IV Escola Regional de Engenharia de Software

ERES

Anais IV ERES

11 a 13 de Novembro 2020

@2020eres @eres2020 @sbc_eres http://eres.sbc.org.br

IV ESCOLA REGIONAL DE ENGENHARIA DE SOFTWARE
4TH REGIONAL SCHOOL OF SOFTWARE ENGINEERING

11 a 13 de novembro de 2020 | *November 11 - 13, 2020*
Maringá, PR, Brazil

ANAIS | *PROCEEDINGS*

Sociedade Brasileira de Computação (SBC)

COORDENAÇÃO GERAL | *GENERAL CHAIRS*
Edson Oliveira Junior (UEM)
Aline M. M. M. Amaral (UEM)
Donizete Carlos Bruzarosco (UEM)
Marcello Erick Bonfim (UniCesumar)

COMITÊ DIRETIVO | *STEERING COMMITTEE*
Edson Oliveira Junior (UEM)
Elder de Macelo Rodrigues (UNIPAMPA)
Fernando dos Santos (UDESC)
Ingrid Nunes (UFRGS)
Maicon Bernardino da Silveira (UNIPAMPA)
Marcelo Hideki Yamaguti (PUCRS)
Rafael Alves Paes de Oliveira (UTFPR)

COORDENADORES DO COMITÊ DE PROGRAMA | *PROGRAM COMMITTEE CHAIRS*
André Takeshi Endo (UTFPR)
Igor Scaliante Wiese (UTFPR)
Katia Romero Felizardo (UTFPR)
Andreia Malucelli (PUCPR)
Avelino F. Zorzo (PUCRS)
Gislaine Camila L. Leal (UEM)

EDITOR | *PROCEEDINGS CHAIR*
Marco Aurélio Graciotto Silva (UTFPR)

CAPA | *COVER*
Ricardo Theis Geraldi (PUCPR)

RESPONSÁVEL PELA PUBLICIDADE | *PUBLICITY CHAIR*
André F. R. Cordeiro (UEM)

PROMOÇÃO | *PROMOTION*
Sociedade Brasileira de Computação (SBC)
REALIZAÇÃO | *REALIZATION*
Departamento de Informática (DIN) – Universidade Estadual de Maringá (UEM)

APOIO | *SUPPORT*
Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq)
Superinterdência Geral de Ciência, Tecnologia e Ensino Superior do Governo do Estado do Paraná
Fundação Araucária
Associação Comercial e Empresarial de Maringá (ACIM)
APL Software Maringá
Incubadora Tecnológica de Maringá
MaringáTech Parque Tecnológico
UniCesumar

PATROCINADORES | *SPONSORS*
Patrocínio Diamente: Elotech
Patrocínio Prata: Platão, Google
Patrocínio Bronze: MentorsTec, Software by Maringá, Mandic Cloud Solutions

Prefácio

É com grande satisfação que apresentamos os artigos aceitos na IV Escola Regional de Engenharia de Software (ERES 2020) e que compõem os anais do evento. A ERES é um evento promovido anualmente pela Sociedade Brasileira de Computação (SBC) com o objetivo de promover e disseminar o conhecimento e boas práticas de Engenharia de Software, tanto do ponto de vista profissional quanto acadêmico. A ERES 2020 ocorreu nos dias 11, 12 e 13 de novembro de 2020, de forma virtual (online streaming), sob a organização do Departamento de Informática (DIN) da Universidade Estadual de Maringá (UEM, campus sede), com o apoio do Centro Universitário de Maringá (UniCesumar).

Mantendo a tradição, esta edição da ERES aceitou submissões de artigos em três trilhas: Graduação, Pós-Graduação, e Indústria de Software. O objetivo da Trilha de Graduação foi apresentar os trabalhos de pesquisa ou relatos de experiência em Engenharia de Software desenvolvidos por acadêmicos de graduação. Já a Trilha da Pós-Graduação teve como objetivo oferecer um espaço para apresentação de trabalhos de alunos de pós-graduação visando incentivar a troca de experiências e divulgação pesquisas em andamento e resultados obtidos. Por fim, na Trilha da Indústria de Software o objetivo foi apresentar trabalhos de profissionais do setor produtivo de tecnologia da informação ou trabalhos acadêmicos resultantes de aplicações práticas na indústria de software, proporcionando a troca de experiências entre profissionais da academia e indústria de software.

A Trilha da Graduação recebeu 42 submissões, das quais 26 foram aceitas (61,9%). A Trilha de Pós-Graduação recebeu 18 submissões, sendo que 8 delas foram aceitas (44,4%). Já na Trilha da Indústria de Software foram 2 submissões, sendo 1 aceita (50%). O processo de submissão e avaliação foi double blind e todos os artigos foram avaliados por pelo menos dois membros do comitê científico da ERES 2020. Dez sessões técnicas foram realizadas ao longo dos três dias do evento para apresentação dos artigos aceitos.

O sucesso da ERES 2020 é resultado da dedicação e entusiasmo de um grande número de pessoas. Agradecemos aos autores pela submissão de seus trabalhos e aos membros do comitê científico e avaliadores que fizeram revisões de excelente qualidade. Agradecemos também aos docentes, discentes, e técnicos-administrativos da UEM que se dedicaram para garantir a realização do evento.

Gostaríamos de agradecer aos palestrantes convidados, Profa Dra Ana Regina Cavalcanti da Rocha (COPPE/UFRJ), Alan Gularte (Dell) e Prof. Me. Leandro Bento Pompermaier (Tecnopuc/PUCRS) e aos professores participantes da Sessão Técnica Especial sobre COVID-19 e a Engenharia de Software: Prof. Dr. Marco Túlio Valente (UFMG), Prof. Dr. Rafael Prikladnicki (PUC-RS), Prof. Dr. Marcos Kalinowski (PUC-Rio) e Profa. Dra. Letícia Machado (UFPA). Ainda, gostaríamos de agradecer enormemente aos nossos patrocinadores: Elotech, Google, MentorsTec, Software by Maringá e Mandic. Por último, mas não menos importante, agradecemos muito ao CNPq, à Fundação Araucária, ao Departamento de Informática da UEM e ao Centro de Tecnologia da UEM pelo apoio financeiro ao evento.

Esperamos que a ERES se estabeleça como uma escola duradoura e contribua com a formação de uma rede de colaboração entre pesquisadores e profissionais da comunidade de Engenharia de Software.

Edson Oliveira Junior
Aline Maria M. M. Amaral
Donizete Carlos Bruzarosco
Marcello Erick Bonfim
Coordenadores Gerais da ERES 2020

Coordenadores de Eventos | *Program Chairs*

Fórum de Graduação | *Undergraduation forum*

André Takeshi Endo (UTFPR)
Igor Scaliante Wiese (UTFPR)

Fórum de Pós-Graduação | *Graduation forum*

Katia Romero Felizardo (UTFPR)
Andreia Malucelli (PUCPR)

Trilha da Indústria | *Industry track*

Avelino F. Zorzo (PUCRS)
Gislaine Camila L. Leal (UEM)

Comitê Científico | *Program committee*

Rio Grande do Sul

Aline Vieira de Mello (UNIPAMPA)
Andreia Sabedra Bordin (UNIPAMPA)
Avelino F. Zorzo (PUCRS)
Claudio Schepke (UNIPAMPA)
Cristiano Tolfo (UNIPAMPA)
Elder de Macedo Rodrigues (UNIPAMPA)
Ewerson Carvalho (UNIPAMPA)
Ingrid Nunes (UFRGS)
Jean Felipe Patikowski Cheiran (UNIPAMPA)
Juliano Varella De Carvalho (FEEVALE)
Kleinner Silva Farias (UNISINOS)
Kurt Werner Molz (UNISC) Lisandra Manzoni Fontoura (UFSM)
Lisane Brisolara de Brisolara (UFPEL)
Maicon Bernardino da Silveira (UNIPAMPA)
Marcelo Hideki Yamaguti (PUCRS)
Marta Rosecler Bez (FEEVALE)
Sabrina Marczak (PUCRS)
Vanessa Gindri Vieira (UFSM)
Raquel Aparecida Pegoraro (UFFS)

Santa Catarina

Adilson Vahldick (UDESC)
Carlos Alberto Barth UDESC)
Daniel Gomes Soares (IFC)
Everaldo Artur Grahl (FURB)
Fabiane Benitti (UFSC)
Fernando dos Santos (UDESC)
Marcela Leite (IFC)
Marilia Guterres Ferreira (UDESC)
Maurício F. Galimberti (UFSC)
Odorico Machado Mendizabal (UFSC)
Pablo Schoeffel (UDESC)
Patrícia Vilain (UFSC)
Raul Sidnei Wazlawick Brasil (UFSC)
Ricardo Pereira e Silva (UFSC)
Rodrigo Curvêllo (IFC)

Paraná

Adilson Luiz Bonifácio (UEL)
Anderson da Silva Marcolino (UFPR)
André Felipe Ribeiro Cordeiro (UEM)
André Luís Andrade Menolli (UENP)
André Takeshi Endo (UTFPR)
Andreia Malucelli (PUCPR)
Gilleanes Thorwald Araujo Guedes (UTFPR)
Gislaine Camila L. Leal (UEM)
Heloise Mânica Paris Teixeira (UEM)
Igor Scaliante Wiese (UTFPR)
Igor Steinmacher (UTFPR)
Itana M. S. Gimenes (UEM)
Katia Romero Felizardo (UTFPR)
Marco Aurélio Graciotto Silva (UTFPR)
Paulo Nardi (UTFPR)
Rafael Alves Paes de Oliveira (UTFPR)
Reginaldo Ré (UTFPR)
Renato Balancieri (UNESPAR)
Sheila Reinehr (PUCPR)
Silvia Regina Vergilio (UFPR)
Silvia Amélia Bim (UTFPR)
Thelma Elita Colanzi (UEM)
Wesley Klewerton Guêz Assunção (UTFPR)
Willian Watanabe (UTFPR)

Demais estados e países

Adenilso Simão (USP)
Awdren Fontão (UFMS)
Bruno Barbieri de Pontes Cafeo (UFMS)
Fabiano Ferrari (UFSCar)
Elisa Nakagawa (USP)
Ellen Francine Barbosa (USP)
Isabel Villanes (UFAM)
José C. Maldonado (USP)
Rodrigo Pereira dos Santos (UNIRIO)
Simone do Rocio Senger de Souza (USP)
Thais Christina Webber dos Santos (University of St Andrews, UK)

Artigos técnicos | *Technical papers*

Trilha da Graduação | *Undergraduate Track*

Open Services for Lifecycle Collaboration: Um Estudo de Mapeamento Sistemático

Bruno Marcelo S. Ferreira[1], Rafael S. Torres[1], Fábio P. Basso[1], Elder Rodrigues[1], Maicon Bernardino[1], Rafael Z. Frantz[2]

[1]Universidade Federal do Pampa (UNIPAMPA), PPGES
Alegrete - RS

[2]Universidade Regional do Noroeste do Estado do Rio Grande do Sul (UNIJUÍ)
Ijuí - RS

brunoferreira.aluno@unipampa.edu.br

Abstract. *The software industry invests in modern tools throughout the software development lifecycle. However, there are challenges to achieve an end-to-end integrated environment such as data integration and artifact traceability. To mitigate these challenges, many approaches have been proposed for integration. In this context, Open Services for Lifecycle Collaboration (OSLC) is an open standard for tool interoperability, which allows data federation throughout Software Engineering (SE) application lifecycles. This study presents a systematic mapping study about OSLC, analyzing 59 primary studies and addressing integration issues.*

Resumo. *A indústria de software investe em ferramentas modernas ao longo de todo o ciclo de desenvolvimento de software. No entanto, existem desafios para alcançar um ambiente integrado de ponta a ponta, como por exemplo estabelecer a rastreabilidade dos artefatos. Para mitigar esses desafios, muitas abordagens foram propostas para integração de ferramentas de software. Nesse contexto, o Open Services for Lifecycle Collaboration (OSLC) é um padrão aberto para interoperabilidade de ferramentas, que permite a federação de dados ao longo do ciclo de vida de aplicações de Engenharia de Software (ES). Este artigo apresenta um estudo de mapeamento sistemático sobre OSLC, analisando 59 estudos primários e abordando questões de integração.*

1. Introdução

Ferramentas de software são usadas para apoiar times a concluir atividades ao longo do ciclo de vida de desenvolvimento de software. No entanto, essas ferramentas geralmente são projetadas sem o suporte necessário para serem integradas a outras ferramentas de software. Além disso, possuem configurações, funcionalidades, fabricantes e manipulam diversos tipos de artefatos. Esse cenário adiciona desafios para a obtenção de um ambiente integrado de ponta a ponta, como por exemplo estabelecer e manter relacionamentos entre artefatos através de toda cadeia de ferramentas.

Ambientes integrados de forma totalmente automatizada são um fenômeno raro na indústria devido ao número de características envolvidas no processo de interoperabilidade de ferramentas [Wicks and Dewar 2007]. Apesar disso, muitas abordagens foram

propostas para *assets* fornecidos como serviços [Biehl et al. 2014]. Essas abordagens visam a minimizar os problemas relacionados ao processo de integração de ferramentas adotados nos contextos de Engenharia de Software (ES) por meio de uma arquitetura típica de serviços [Zhang and Møller-Pedersen 2014].

Nesse contexto, uma emergente solução industrial foi proposta: Open Services for Lifecycle Collaboration (OSLC) [OSLC 2020]. O OSLC define um modelo de representação comum para artefatos produzidos ao longo do projeto, bem como métodos que permitem que outras aplicações manipulem. Essas especificações são definidas na especificação principal do OSLC, que consiste em regras e padrões para o uso de tecnologias como o RESTful e do protocolo HTTP. Além da especificação principal, existem domínios OSLC que a estendem e definem regras para seus respectivos domínios (Gerenciamento de Requisitos, Gerenciamento de Qualidade, Gerenciamento de Configuração e Mudanças). Dessa maneira, o OSLC fornece interoperabilidade em diferentes domínios de desenvolvimento de software sem obter conhecimento sobre como as ferramentas operam internamente.

O OSLC é baseado nos princípios de Dados Ligados e usa tecnologias como protocolo HTTP e RESTful. Além disso, cada artefato é representado por arquivos RDF/XML estruturados com base em domínios OSLC. Os relacionamentos entre os artefatos são obtidos através de uma estrutura composta por três elementos: sujeito, predicado e objeto. Por exemplo, em um ambiente formado pelas ferramentas de gerenciamento de requisitos (Ferramenta A) e gerenciamento de testes (Ferramenta B), um requisito na Ferramenta A pode ser validado por um caso de teste na Ferramenta B por meio do OSLC.

Em uma cadeia de ferramentas OSLC, uma ferramenta pode ser classificada como um provedor ou consumidor. As ferramentas provedoras são projetados para armazenar e prover dados, permitindo as ferramentas consumidoras fácil acesso para navegar, criar e recuperar dados [Aichernig et al. 2014]. Existem três abordagens para prover suporte OSLC em ferramentas de software: abordagem nativa, *plugin* e adaptador. A abordagem mais recomendada é o desenvolvimento de adaptadores, pois não é preciso possuir conhecimento das tecnologias que envolvem a implementação da ferramenta.

Uma vez que a comunidade OSLC está ativa desde 2008, uma pergunta em aberto é ”qual é o estado da arte e a prática do OSLC para integração de ferramentas e ambientes de Engenharia de Software?”. A seguir, é apresentado um Estudo de Mapeamento Sistemático (SMS) que analisa 59 estudos primários que relatam o uso do OSLC.

O restante do artigo está organizado da seguinte forma: na seção 2 o protocolo do estudo de mapeamento é exposto e a Seção 3 discute os resultados do SMS. A Seção 4 discute as ameaças à validade do SMS. Por fim, a Seção 5 conclui este trabalho.

2. Estudo de Mapeamento Sistemático

Nesta seção, é descrito o processo aplicado em nosso estudo de mapeamento sistemático. O protocolo seguido foi proposto por Petersen *et al.* [Petersen et al. 2015], que define um processo para execução de mapeamentos sistemáticos na área de ES. O protocolo completo deste trabalho está disponível no Google Drive [1].

[1] `https://drive.google.com/drive/folders/14Fus_yehjlRxRXP53_V1ATxgzsdfo7Bd?usp=sharing`

2.1. Questões de Pesquisa

O Estudo de Mapeamento Sistemático possui as seguintes questões de pesquisa (Q):

Q1. *Em quais fases do ciclo de vida do software o trabalho utiliza o padrão OSLC como solução?* Os estudos de integração de aplicações voltados para contextos de ES são normalmente associados a determinados ciclos de vida da aplicação. Assim, esta questão de pesquisa busca caracterizar os estudos que contribuem para determinadas disciplinas de ES, fornecendo um mapa de cobertura que permite inferir se os processos de desenvolvimento de software são totalmente automatizados.

Q2. *Quais são as facetas da contribuição do estudo?* Nosso objetivo é caracterizar as contribuições da área de acordo com sua faceta: seja com ferramentas, modelagem, processos e metodologias.

Q3. *Quais os tipos de pesquisa mais frequentes sobre OSLC?* Por fim, para compreender a maturidade dos estudos do ponto de vista empírico, também buscamos classificar os estudos de acordo com o tipo de avaliação adotada.

2.2. Strings de Busca

A busca pelos trabalhos aconteceram em 5 bibliotecas digitais amplamente utilizada pela comunidade científica: ACM Digital Library[2], IEEEXplore[3], Springer Link[4], Science Direct[5] e Scopus[6]. As *strings* de busca foram elaboradas com o objetivo de obter uma ampla cobertura de estudos sobre o padrão OSLC, como mostra a Tabela 1.

Tabela 1. Strings de Busca

Base de Dados	String de busca
ACM DL	(“Open Services for Lifecycle Collaboration” OSLC) AND (Integration Interoperability “Linked Data” Lifecycle Traceability)
IEEE Xplore	(“Open Services for Lifecycle Collaboration” OR OSLC) AND (Integration OR Interoperability OR “Linked Data” OR Lifecycle OR Traceability)
Springer Link	(“Open Services for Lifecycle Collaboration” OR OSLC) AND (Integration OR Interoperability OR “Linked Data” OR Lifecycle OR Traceability)
Science Direct	(“Open Services for Lifecycle Collaboration” OR OSLC) AND (Integration OR Interoperability OR “Linked Data” OR Lifecycle OR Traceability)
Scopus	TITLE-ABS-KEY ((“Open Services for Lifecycle Collaboration” OR OSLC) AND (Integration OR Interoperability OR “Linked Data” OR Lifecycle OR Traceability))

3. Análise dos Resultados

Esta seção apresenta a análise dos resultados obtidos através da execução do nosso protocolo. Os resultados são baseados nos artigos selecionados durante o processo de triagem, conforme ilustrado na Figura 1. Na primeira etapa, as strings de busca foram utilizadas

[2]https://dl.acm.org/
[3]https://ieeexplore.ieee.org/
[4]https://link.springer.com/
[5]https://www.sciencedirect.com/
[6]https://www.scopus.com/

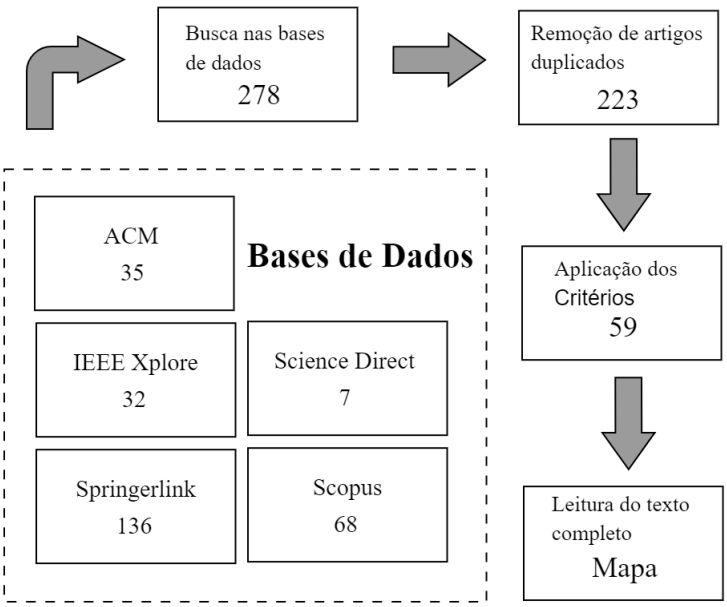


Figura 1. Processo de Triagem dos Estudos

nas bases de dados e foram retornados 278 artigos. Em seguida, os estudos duplicados entre as bases de dados foram desconsiderados, restando 223 artigos. Assim, após aplicar os critérios de inclusão e exclusão com base no título, resumo, palavras-chave e introdução desses artigos, foram considerados 59 estudos de acordo com o tema de pesquisa.

3.1. Q1. Em quais fases do ciclo de vida do software o padrão OSLC é utilizado?

Para responder a Q1, foram mapeadas as fases do ciclo de vida do software em que a solução do estudo foi empregada. Para isso usamos as disciplinas do Rational Unified Process (RUP). De acordo com os resultados do SMS, a maioria das soluções propostas são para os domínios de Gerenciamento de Requisitos, Análise & Projeto, Gerenciamento de Configuração e Mudanças e Testes, como mostra a Tabela 2. Todas as ferramentas no conjunto de ferramentas foram consideradas parte da solução OSLC. Portanto, os estudos podem ser categorizados em mais de uma disciplina.

Tabela 2. Estudos Distribuídos pelas Disciplinas do RUP

Disciplina	Quantidade de artigos
Modelagem de Negócios	1
Requisitos	21
Análise & Projeto	29
Implementação	6
Teste	25
Implantação	0
Gerenciamento de Configuração e Mudanças	14
Gerenciamento de Projetos	6
Ambiente	2
N/A	11

As abordagens OSLC se empenham em resolver problemas comuns entre os artigos. No domínio de requisitos, o OSLC é proposto para estabelecer a rastreabilidade entre os requisitos e o restante do projeto. Além disso, as ferramentas de requisitos de software são frequentemente usadas nos exemplos de conjuntos de ferramentas como um ponto de partida para o fluxo de trabalho.

A partir de conceitos de Dados Ligados, artigos como [VanZandt 2015], [Buffoni et al. 2017] apresentam abordagens para rastrear o histórico das mudanças por meio das propriedades do OSLC, incluindo as partes interessadas que o modificaram e tarefas nas quais os artefatos estão associados. Nesse sentido, abordagens de domínio Gerenciamento de Configuração e Mudanças como [Lednicki et al. 2016], [Pikus et al. 2019], [Adjepon-Yamoah et al. 2015], [Regan et al. 2015], [Aoyama et al. 2014], [Aoyama et al. 2013] são propostas para controle de versão e automação de tarefas por meio dos serviços OSLC.

Os resultados do SMS reforçam a necessidade de integrar diferentes domínios de desenvolvimento de software desde suas fases iniciais. Portanto, podemos citar ferramentas de Engenharia Orientada a Modelos (MDE), que são citadas na fase de Análise & Projeto como soluções para minimizar esses esforços de integração. Nesse contexto, o Eclipse Lyo [El-khoury 2016] propõe um gerador de código-fonte aberto para interfaces de ferramentas em conformidade com OSLC no ambiente eclipse, que foi citado por artigos que foram usados no desenvolvimento de adaptadores OSLC. Ainda, há trabalhos que propõem abordagens para apoiar a integração ou transformação de modelos heterogêneos como BPMN, UML, SysML, SoaML e EMF [d. Martino et al. 2016], [Lu et al. 2018], [Arnould 2018], [Mustafa and Labiche 2017], [Zhang and Møller-Pedersen 2013], [Biehl et al. 2012], [Biehl et al. 2012].

Outra observação é um número considerável de soluções aplicadas à disciplina de Teste devido aos contextos em que o OSLC é normalmente aplicado. Por se tratarem de sistemas críticos de segurança, os conjuntos de ferramentas são compostos por ferramentas de V&V.

3.2. Q2. Quais são as facetas da contribuição do estudo?

Identificar quais facetas de contribuição são mais exploradas em uma área de pesquisa é importante para encontrar lacunas de pesquisa ou avaliar tendências na área. Em relação à faceta de contribuição dos estudos, estabelecemos quatro categorias: modelo, ferramenta, método e processo.

Os resultados mostram que a maioria dos artigos contribui para a faceta de Ferramentas (30). Esta categoria está relacionada à implementação de adaptadores de ferramenta OSLC, serviços da web OSLC ou ferramentas para visualizar artefatos de dados. Esses trabalhos propõem exemplos de conjuntos de ferramentas que usam adaptadores OSLC como uma solução de integração. Em relação à categoria Modelo (20), trabalhos como: [Zhang and Møller-Pedersen 2014], [Gürdür et al. 2018], [Alvarez-Rodríguez et al. 2018], [Gallina et al. 2016], [Mustafa and Labiche 2017], cobrem a proposta para estender as especificações de domínios OSLC. Outro tópico representativo na categoria de modelo são abordagens para gerar especificações OSLC em ambientes baseados em MDE. Nos exemplos da categoria Processo (3) encontrados, o OSLC é usado para automatizar atividades que envolvem a rastreabilidade de artefa-

tos, conforme mostrado em trabalhos de [Baumgart and Ellen 2014], [Regan et al. 2015] e [Regan et al. 2014].

Em outros trabalhos, são propostos Métodos (20) ou novas abordagens para o uso de OSLC. Assim, foi possível identificar que as contribuições dos estudos estão voltadas para a resolução de problemas em um domínio específico, como uma atividade ou um tipo de indústria. É um indicativo da lacuna de pesquisa de abordagens para soluções globais sobre OSLC, em processos especialmente planejados para ES.

3.3. Q3. Quais os tipos de pesquisa mais frequentes sobre OSLC?

A maturidade de uma pesquisa pode ser determinada pela forma como o estudo afeta a prática na área. Por esse motivo, adotamos as categorias propostas pelo estudo de Wieringa *et al.* [Wieringa et al. 2005], mapeando os tipos de pesquisa empregados por cada pesquisa. Assim, são considerados os seguintes tipos de estudos: *Evaluation Research*, *Validation Research*, *Solution Proposal*, *Experience Report*, *Opinion Paper* e *Philosophical Paper*.

A maioria dos artigos é classificada como *Solution Proposal* (36), reforçando a ideia de que o OSLC possui implementações como prova de conceito e protótipos em ambientes reais. Alguns trabalhos sugerem a realização de estudos de caso, embora sejam apresentados dados insuficientes, levando a um pequeno número de estudos empíricos, incluindo *Validation Research* (3) e *Evaluation Research* (5). Os artigos classificados como *Philosophical Papers* (12) discutem como usar o OSLC para solucionar problemas, mas não apresentam sua implementação. Existem também diretrizes, opiniões e relatórios baseados em experiências profissionais na indústria, resultando em artigos de *Experience Report* (5) e *Opinion Paper* (1), afirmando que o OSLC é utilizado em ambientes reais.

4. Ameaças à Validade

Os trabalhos empíricos devem ser capazes de conter informações suficientes para que seja possível replicá-lo apenas com o acesso ao protocolo. Entretanto, durante sua execução existem alguns fatores que podem interferir nos resultados, sendo tratados como ameaças à validade. A seguir seguem algumas ameaças deste SMS:

Validade interna: Esse tipo de ameaça refere-se à validade da análise realizada, validando se as conclusões derivadas dos dados são válidas internamente. Nesse sentido, perguntas de pesquisa bem definidas possibilitaram concluir resultados de acordo com o tópico de pesquisa. Além disso, uma preocupação constante do estudo foi garantir que o processo de seleção de trabalhos ocorresse conforme os critérios de inclusão e exclusão definidos. Para esse fim, foi cuidadosamente investido um grande esforço para filtrar os estudos primários, com o auxílio de ferramentas para organização e filtros dos trabalhos utilizados pelos pesquisadores.

Validade externa: Diz respeito sobre o quão generalizadas são as descobertas do estudo de mapeamento. Nesse sentido, a representatividade estatística pode ser uma ameaça. Por considerar um número expressivo de artigos, essa ameaça é tratada pelo refinamento da *string* de busca por mais de um pesquisador, permitindo a inclusão de 59 estudos primários, os quais caracterizam um bom poder estatístico e a adaptação do texto para cada uma das bases selecionadas.

Validade de construção: O trabalho foi executado com base em um protocolo bem definido e amplamente utilizado na área de Engenharia de Software. Para garantir a extração dos dados, o esquema de classificação foi revisado e discutido por um segundo pesquisador. Além disso, o trabalho conta com critérios de qualidade para identificar os trabalhos mais relevantes para o tema de pesquisa. Entretanto, as categorias escolhidas para o esquema de classificação podem ser consideradas genéricas demais. Para mitigar esse viés, as categorias foram escolhidas a partir da base de conhecimento do RUP.

Validade da conclusão: Uma ameaça a conclusão pode ocorrer na fase de classificação, no qual pode ser influenciada pelo viés do *know-how* dos autores. Para superar esse viés, foi utilizado um formulário de extração de dados que contém todos os dados necessários para responder as questões de pesquisa.

5. Considerações Finais

As especificações OSLC estão se tornando cada vez mais populares na indústria de software. Ao longo dos anos, desafios e práticas foram relatados na literatura para conectar artefatos entre domínios, aplicações e organizações. Nesse sentido, este trabalho empírico ajuda a compreender as áreas potenciais de emprego do OSLC com base na análise de artigos selecionados. Por exemplo, nossos resultados mostram as motivações e contribuições do OSLC para apoiar a integração de ferramentas com base nas disciplinas da Engenharia de Software, contexto industrial, facetas da contribuição e tipo de avaliação de tais trabalhos.

Os trabalhos encontrados enfocam a geração de código-fonte para conectar ferramentas. Os autores procuram maneiras de representar os artefatos bem como explorar ontologias e a extensão dos domínios OSLC. Além disso, é consenso da indústria integrar ferramentas por meio de adaptadores, visto que as ferramentas geralmente não possuem suporte nativo ao OSLC. Nesse contexto, as abordagens de Desenvolvimento Dirigido por Modelos (MDD) e MDE são de interesse da indústria para gerar essas interfaces e ao longo dos anos, vem se consolidando como a melhor alternativa para o desenvolvimento de adaptadores OSLC. Apesar disso, a geração de código não é 100% automática e ainda não atingiu o nível de integração de todo o ambiente. Este cenário pode estar relacionado com o elevado desafio técnico necessário para integrar tais aplicações.

O primeiro trabalho que encontramos foi publicado em 2010 e relata a pesquisa e o potencial para uma nova especificação para integração de ferramentas. Desde então, pesquisadores exploraram o OSLC e propuseram novas abordagens para melhorar a integração de ferramentas em ambientes ALM [Tüzün et al. 2019].

As principais vantagens do OSLC estão relacionadas aos Dados Ligados e envolve não apenas adaptadores de ferramentas para integração ponto a ponto, mas também abordagens propostas para promover a substituição de ferramentas na cadeia de ferramentas, incluindo modificação de especificações de domínio OSLC e soluções para atividades automatizadas para integração de ferramentas. Todos esses elementos podem ser realizados por meio de abordagens como o MDE, que através de uma representação comum, pode fornecer todo o dispositivo que permite a automação das atividades e integração de todo o ambiente de desenvolvimento de software.

O OSLC é motivado por problemas de integração de contextos industriais de alta complexidade. Nesse sentido, as soluções de integração propostas pelos autores geral-

mente são projetadas para domínios específicos de aplicações de software de engenharia de sistemas. Por exemplo, a maioria dos estudos é dedicada a sistemas embarcados e a outros contextos de aplicações da indústria automobilística, naval, entre outros. Embora o OSLC proponha a integração de ponta a ponta de todo o ciclo do software, os estudos mostram que as cadeias de ferramentas existentes são, em sua maior amostra, adotadas nos estágios iniciais do projeto de software e também nas que envolvem atividades de verificação e validação.

Por fim, nós seguimos um protocolo replicável que pode ser implementado ao longo dos anos para identificar novas contribuições na área. Esse fator é especialmente importante para a comunidade acadêmica, pois fornece uma visão geral do estado da arte e da prática, sobre as áreas exploradas, as deficiências para adoção do OSLC e também as áreas inexploradas. Além disso, este trabalho pode servir para tomada de decisão no desenvolvimento de soluções de integração na indústria.

6. Agradecimentos

Este estudo foi parcialmente financiado através da Pró-Reitoria de Pesquisa (PROPESQ), com bolsa do programa AGP (Apoio a Grupos de Pesquisa), e pela FAPERGS, por meio do projeto ARD N. 19/2551-0001268-3.

Referências

Adjepon-Yamoah, D., Romanovsky, A., and Iliasov, A. (2015). A reactive architecture for cloud-based system engineering. In *Proceedings of the 2015 International Conference on Software and System Process*, ICSSP 2015, pages 77–81, New York, NY, USA. ACM.

Aichernig, B. K., Hörmaier, K., Lorber, F., Nickovic, D., Schlick, R., Simoneau, D., and Tiran, S. (2014). Integration of requirements engineering and test-case generation via oslc. In *2014 14th International Conference on Quality Software*, pages 117–126.

Alvarez-Rodríguez, J., Mendieta, R., Vara, J., Fraga, A., and Llorens, J. (2018). Enabling system artefact exchange and selection through a linked data layer. *Journal of Universal Computer Science*, 24(11):1536–1560. cited By 1.

Aoyama, M., Yabuta, K., Kamimura, T., Inomata, S., Chiba, T., Niwa, T., and Sakata, K. (2013). Promis: A management platform for software supply networks based on the linked data and oslc. In *2013 IEEE 37th Annual Computer Software and Applications Conference*, pages 214–219.

Aoyama, M., Yabuta, K., Kamimura, T., Inomata, S., Chiba, T., Niwa, T., and Sakata, K. (2014). A resource-oriented services platform for managing software supply chains and its experience. In *2014 IEEE International Conference on Web Services*, pages 598–605.

Arnould, V. (2018). Using model-driven approach for engineering the system engineering system. In *2018 13th Annual Conference on System of Systems Engineering (SoSE)*, pages 608–614.

Baumgart, A. and Ellen, C. (2014). A recipe for tool interoperability. In *2014 2nd International Conference on Model-Driven Engineering and Software Development (MODELSWARD)*, pages 300–308.

Biehl, M., El-Khoury, J., Loiret, F., and Törngren, M. (2014). On the modeling and generation of service-oriented tool chains. *Software & Systems Modeling*, 13(2):461–480.

Biehl, M., El-Khoury, J., and Törngren, M. (2012). High-level specification and code generation for service-oriented tool adapters. In *2012 12th International Conference on Computational Science and Its Applications*, pages 35–42.

Biehl, M., Gu, W., and Loiret, F. (2012). Model-based service discovery and orchestration for oslc services in tool chains. In Brambilla, M., Tokuda, T., and Tolksdorf, R., editors, *Web Engineering*, pages 283–290, Berlin, Heidelberg. Springer Berlin Heidelberg.

Buffoni, L., Pop, A., and Mengist, A. (2017). Traceability and impact analysis in requirement verification. In *Proceedings of the 8th International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools*, EOOLT '17, pages 95–98, New York, NY, USA. ACM.

d. Martino, B., Esposito, A., Nacchia, S., and Maisto, S. A. (2016). Towards a uniform semantic representation of business processes, uml artefacts and software assets. In *2016 10th International Conference on Complex, Intelligent, and Software Intensive Systems (CISIS)*, pages 543–548.

El-khoury, J. (2016). Lyo code generator: A model-based code generator for the development of oslc-compliant tool interfaces. *SoftwareX*, 5:190 – 194.

Gallina, B., Padira, K., and Nyberg, M. (2016). Towards an iso 26262-compliant oslc-based tool chain enabling continuous self-assessment. In *2016 10th International Conference on the Quality of Information and Communications Technology (QUATIC)*, pages 199–204.

Gürdür, D., Feljan], A. V., El-khoury, J., Mohalik], S. K., Badrinath, R., Mujumdar], A. P., and Fersman, E. (2018). Knowledge representation of cyber-physical systems for monitoring purpose. *Procedia CIRP*, 72:468 – 473. 51st CIRP Conference on Manufacturing Systems.

Lednicki, L., Sapienza, G., Johansson, M. E., Seceleanu, T., and Hallmans, D. (2016). Integrating version control in a standardized service-oriented tool chain. In *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*, volume 1, pages 323–328.

Lu, J., W ang, J., Chen, D., Wang, J., and TöRngren, M. (2018). A service-oriented tool-chain for model-based systems engineering of aero-engines. *IEEE Access*, 6:50443–50458.

Mustafa, N. and Labiche, Y. (2017). Employing linked data in building a trace links taxonomy. pages 186–198. cited By 2.

OSLC (2020). Open services for lifecycle collaboration primer web page. Accessed at February 2020.

Petersen, K., Vakkalanka, S., and Kuzniarz, L. (2015). Guidelines for conducting systematic mapping studies in software engineering: An update. *Information and Software Technology*, 64:1 – 18.

Pikus, Y., Weissenberg, N., Holtkamp, B., and Otto, B. (2019). Semi-automatic ontology-driven development documentation: Generating documents from rdf data and dita templates. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, SAC '19, pages 2293–2302, New York, NY, USA. ACM.

Regan, G., Biro, M., Flood, D., and McCaffery, F. (2015). Assessing traceability - practical experiences and lessons learned. *Journal of Software: Evolution and Process*, 27(8):591–601. cited By 6.

Regan, G., Biro, M., Mc Caffery, F., Mc Daid, K., and Flood, D. (2014). A traceability process assessment model for the medical device domain. In Barafort, B., O'Connor, R. V., Poth, A., and Messnarz, R., editors, *Systems, Software and Services Process Improvement*, pages 206–216, Berlin, Heidelberg. Springer Berlin Heidelberg.

Tüzün, E., Tekinerdogan, B., Macit, Y., and İnce, K. (2019). Adopting integrated application lifecycle management within a large-scale software company: An action research approach. *Journal of Systems and Software*, 149:63 – 82.

VanZandt, L. (2015). Enabling rational decision making with provenance-annotated oslc relationships. In *2015 IEEE International Symposium on Systems Engineering (ISSE)*, pages 346–352.

Wicks, M. N. and Dewar, R. G. (2007). Controversy corner: A new research agenda for tool integration. *J. Syst. Softw.*, 80(9):1569–1585.

Wieringa, R., Maiden, N., Mead, N., and Rolland, C. (2005). Requirements engineering paper classification and evaluation criteria: A proposal and a discussion. *Requir. Eng.*, 11(1):102–107.

Zhang, W. and Møller-Pedersen, B. (2013). Establishing tool chains above the service cloud with integration models. In *2013 IEEE 20th International Conference on Web Services*, pages 372–379.

Zhang, W. and Møller-Pedersen, B. (2014). Modeling of tool integration resources with oslc support. In *2014 2nd International Conference on Model-Driven Engineering and Software Development (MODELSWARD)*, pages 99–110.

Estudo Exploratório Sobre o Uso da Arquitetura de Microsserviços em Empresas da Cidade de Maringá - PR

Tamires G. B. Deprá[1], **Douglas R. Tanno**[1], **Aline Maria M. M. Amaral**[1], **Thelma E. C. Lopes**[1]

[1]Departamento de Informática – Universidade Estadual de Maringá (UEM)
Maringá – PR – Brasil

{tami.g.bsp,douglas.tanno}@gmail.com, {ammmamaral,thelma}@din.uem.br

Resumo. *A arquitetura de microsserviços, embora ainda muito recente, já demonstra grande impacto na indústria de software, onde empresas dos mais variados tamanhos e segmentos vêm adotando esse estilo arquitetural. Neste trabalho buscou-se identificar se as empresas de software da cidade de Maringá - PR utilizam a arquitetura de microserviços e como a utilizam. Para este propósito, um questionário foi aplicado em uma amostra de empresas da cidade de Maringá - PR. Com os dados resultantes da aplicação do questionário, foi possível observar que um número expressivo de empresas utilizam a arquitetura de microsserviços e que as mesmas aplicam as boas práticas observadas na literatura para seu uso.*

Abstract. *The microservice architecture, although still very recent, already shows great impact in the software industry, where companies of the most varied sizes and segments have been adopting this architectural style. In this work, we aimed to identify whether software companies in the city of Maringá - PR use microservices architecture and how they use it. For this purpose, a questionnaire was applied to a sample of companies in the city of Maringá - PR. Considering the data resulting from the application of the questionnaire, it was possible to observe that a significant number of companies use the microservice architecture and that they apply the good practices observed in the literature for their use.*

1. Introdução

A arquitetura de software de um programa ou sistema computacional é a estrutura ou estruturas do sistema, que abrange os componentes do software, as propriedades externamente visíveis desses componentes e as relações entre eles [Bass et al. 2003].

Segundo [Newman 2015], microsserviços são pequenos serviços autônomos que trabalham juntos. A arquitetura de microsserviços propõe o desenvolvimento de um sistema distribuído, onde cada pequena parte do sistema é um serviço independente, facilitando a criação de novos serviços para acoplamento no sistema e manutenção dos serviços que já se encontram em produção.

Muitas questões com relação a adoção da arquitetura de microserviços por empresas de desenvolvimento de software ainda estão em aberto, tais como: se esta tecnologia tem um impacto positivo em empresas de pequeno e médio porte; qual estratégia

para implantação está sendo realizada pelas empresas; além das perspectivas futuras com relação ao uso dessa tecnologia. Nesse sentido, o objetivo deste trabalho foi realizar um estudo exploratório sobre a arquitetura de microsserviços, entendendo melhor o seu uso em empresas, por meio da aplicação de um questionário em empresas de desenvolvimento de software da cidade de Maringá - PR. Com base nas respostas do questionário, foi realizada uma análise para determinar se as empresas utilizam essa arquitetura e de que forma elas a utilizam.

Este trabalho está organizado da seguinte forma: a Seção 2 apresenta uma descrição da arquitetura de microsserviço; na Seção 3 o método de pesquisa utilizado no trabalho é descrito; na Seção 4 são detalhados os resultados obtidos com a realização deste trabalho juntamente com a discussão dos mesmos; finalmente, na Seção 5 são apresentadas as conclusões do trabalho, assim como as motivações para a realização de trabalhos futuros.

2. Microsserviços

Microsserviços são pequenos serviços autônomos que trabalham juntos. Além disso, cada microsserviço possui seus próprios dados, sua própria regra de negócio e uma interface bem definida, de modo que cada serviço possa ter seu próprio repositório, controlador de versões, ciclo de vida e controlador de tarefas. Além disso, outra proposta desse estilo arquitetural é que cada microsserviço tenha sua própria equipe de desenvolvimento e implantação. A arquitetura de microsserviços trabalha com extensa granularidade do sistema e alta escalabilidade possibilitando o desenvolvimento em diversas linguagens e plataformas diferentes, com banco de dados distribuídos e até equipes distribuídas [Meloca 2017, Newman 2015].

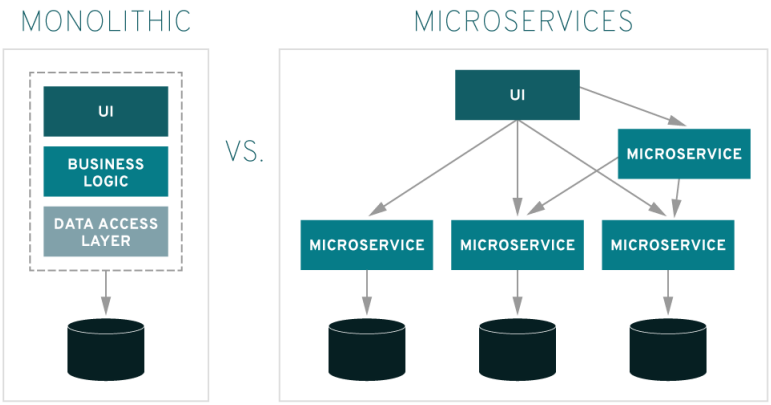


Figura 1. Critérios utilizados para decisão sobre a extração dos microsserviços.

Na Figura 1 pode-se observar um comparativo entre a estrutura de um sistema monolítico e um sistema que utiliza o estilo arquitetural de microsserviços.

A arquitetura de microsserviços é particularmente adequada para infraestruturas de nuvem, pois se beneficia muito da elasticidade ativada pela nuvem e do rápido provisionamento de recursos. Fazer uso da arquitetura de microsserviços, no entanto, não é uma tarefa fácil, pois exige o gerenciamento de uma arquitetura distribuída e seus desafios, que incluem latência e falta de confiabilidade da rede, tolerância a falhas, orquestração

de serviços complexos, consistência de dados e gerenciamento de transações e balanceamento de carga. Infraestruturas em nuvem e novas tecnologias desempenham um papel fundamental no uso de arquiteturas de microsserviços e no gerenciamento dos desafios e complexidades associados [Di Francesco et al. 2019].

É comum comparar a arquitetura de microsserviços com a arquitetura monolítica, pois a arquitetura monolítica é amplamente utilizada e difere em diversos aspectos da arquitetura de microsserviços. Uma aplicação monolítica é feita como uma única unidade lógica executável. A arquitetura de microsserviços deriva da SOA (do inglês *Service Oriented Architecture*) cuja finalidade é reduzir o acoplamento entre os diversos módulos de uma aplicação com o propósito de facilitar sua escalabilidade. Na arquitetura monolítica nenhuma funcionalidade do sistema existe e opera por si só [Fowler and Lewis 2014, Meloca 2017].

Como resultado da adoção da arquitetura de microsserviços espera-se adquirir benefícios, por exemplo, escalabilidade e responsabilidade, mas também pode-se ter alguns malefícios, por exemplo, dificuldade na refatoração e dificuldade na comunicação distribuída, a depender do aspecto sobre o qual se olha, de maneira que há um limiar a ser estudado entre as curvas da arquitetura monolítica e da arquitetura de microsserviços considerando o número de usuários pelo tempo de resposta [Meloca 2017, Newman 2015].

3. Método de Pesquisa

A pesquisa foi conduzida em 4 etapas: Levantamento Bibliográfico, Definição do Método de Avaliação, Coleta de Dados e Avaliação dos Resultados.

Para a realização do estudo exploratório, foi necessário, inicialmente, realizar o levantamento bibliográfico sobre as características da arquitetura de microsserviços, seus benefícios e detalhes de implementação, com objetivo de explorar a aplicação dessa arquitetura na prática por empresas de software da cidade de Maringá-PR e validar os dados a serem obtidos na pesquisa.

A definição do método de avaliação consistiu em eleger o meio mais adequado para coletar os dados de acordo com a proposta do trabalho. Foi definido o desenvolvimento e aplicação de um questionário para realização de um estudo exploratório na cidade de Maringá-PR, contendo perguntas objetivas e discursivas sobre a arquitetura de microsserviços. As perguntas do questionário foram elaboradas nos seguintes formatos: (*i*) múltipla escolha, restringidas a seleção de uma opção, (*ii*) tipo sim/não, (*iii*) tipo seleção, onde é possível ao respondente selecionar mais de uma opção, (*iv*) e questões discursivas.

A coleta dos dados foi realizada de forma virtual, onde os participantes da pesquisa receberam um *link* para responder ao questionário. O critério de escolha das empresas para participarem da pesquisa foi o de serem indústrias de software localizadas em Maringá-PR, sem outros requisitos como número de funcionários ou ramo do desenvolvimento de software.

A última etapa da pesquisa consistiu na análise e discussão sobre os resultados obtidos com a aplicação do questionário, comparando os dados coletados com os materiais disponíveis na literatura.

O questionário completo desenvolvido nesta pesquisa pode ser consultado cli-

cando aqui.

4. Resultados e discussão

Atualmente em Maringá-PR existem aproximadamente 400 empresas de software. O questionário foi enviado à algumas dessas empresas, dentre as quais 23 participaram da pesquisa. Das 23 empresas, 39,1% utilizam a arquitetura de microsserviços, sendo esse percentual referente a 9 empresas. Considerando as respostas dadas ao questionário pelas empresas participantes da pesquisa são apresentados a seguir: os percentuais correspondentes a cada questão de múltipla escolha (opção única de resposta); o gráfico referente às questões de seleção; e as respostas das questões discursivas.

A primeira questão refere-se ao cargo ocupado pelo respondente na empresa, onde 56,5% são desenvolvedores e 17,4% são analistas. Quanto ao restante dos respondentes, há uma grande variação de cargos dentro da empresa, como: arquiteto, gerente de projetos, coordenador, testador, engenheiro de software, diretor de inovação e gerente de tecnologia.

A questão seguinte é "A empresa se organiza por área de desenvolvimento (interface, negócio, banco de dados, etc.) ou por área do negócio (exemplo: vendas, contabilidade, financeiro, etc.) com times multifuncionais?". A maior parte das empresas, 65,2%, é dividida em áreas de negócio, 30,4% são divididas por áreas de desenvolvimento e 4,4% por ambas.

A pergunta seguinte é sobre a distribuição das equipes. É comum em empresas que utilizam microsserviços que as equipes de trabalho estejam alocadas em locais diferentes, como observado na fundamentação teórica. Nas empresas de Maringá podemos observar que a maioria, 52,2%, das empresas, possui esse esquema de trabalho, sendo que 47,8% delas não possui.

Em seguida, foi questionado se a empresa adota a arquitetura de microsserviços. Um percentual de 39,1% empresas usa microsserviços, enquanto que 60,9% delas não utiliza esse estilo arquitetural. Após essa pergunta o questionário foi dividido em duas seções diferentes, uma para as empresas que utilizam a arquitetura de microsserviços e outra seção para as empresas que utilizam outras arquiteturas para o desenvolvimento de seus sistemas.

A próxima pergunta do questionário, "Como foi a escolha pela arquitetura de microsserviços?", já faz parte da seção para as empresas que utilizam a arquitetura de microsserviços. Na Figura 2 são apresentadas todas as opções e seus respectivos percentuais. Pode-se observar que a maioria das empresas, 55,6%, optou pela migração gradual do sistema e 11,1% alegam que o sistema está passando pela migração atualmente, também de forma gradual. O gráfico com as respostas à esta questão pode ser observado na Figura 1. Tais informações corroboram com [Balalaie et al. 2015], que afirmam que a migração do sistema para a arquitetura de microsserviços não é simples e deveria ser realizada de forma incremental e em muitos passos sem afetar o usuário final. Nesse sentido, pode-se observar que as empresas, assim como sugerido na literatura, realizam a migração para a arquitetura de microsserviços de forma gradual.

Na questão sequinte, "Quais critérios para decisão de extração de microsserviços?", a escalabilidade foi apontada como critério para decisão de extração

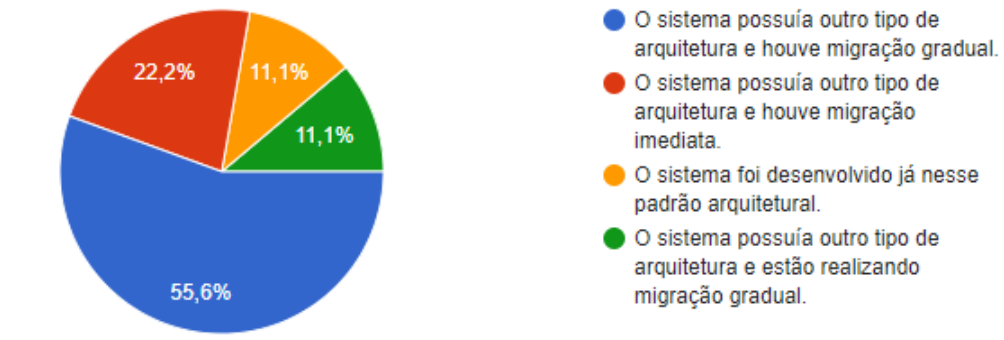


Figura 2. Estratégia para implantação da arquitetura de microsserviços.

por todas as empresas que utilizam a arquitetura de microsserviços. A segunda característica mais apontada foi acoplamento. Na Figura 3, pode-se observar as opções selecionadas pelos respondentes. Segundo [Meloca 2017], dada a adoção da arquitetura de microsserviços, como benefício, dentre outros, espera-se obter um bom nível de escalabilidade do sistema. E para [Newman 2015], uma característica da arquitetura de microsserviços é sua autonomia. Esse autor destaca que toda a comunicação entre os serviços é feita via chamadas através da rede, para forçar a separação entre eles e evitar alto acoplamento. Portanto, assim como o formato da migração, os critérios de extração também vão de encontro ao proposto pela literatura.

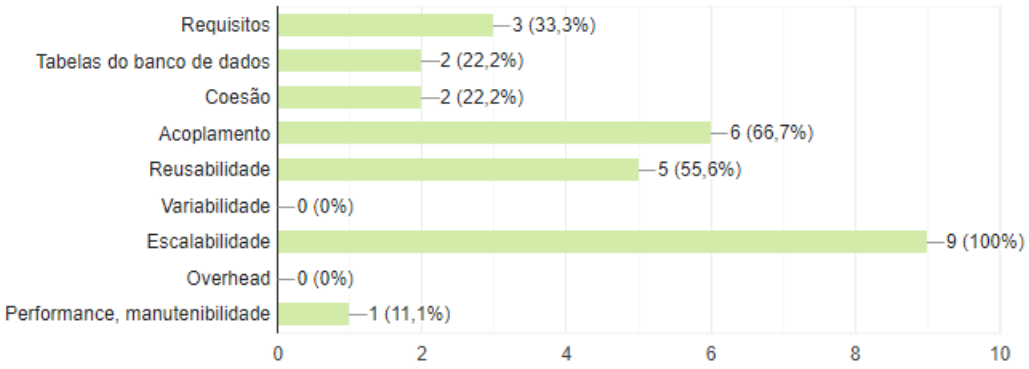


Figura 3. Critérios utilizados para decisão sobre a extração dos microsserviços.

A próxima questão do questionário aplicado refere-se aos *frameworks* utilizados no desenvolvimento de microserviços. Na Figura 4 pode-se observar as opções selecionadas pelos respondentes, sendo ASP .Net o *framework* mais utilizado, indicado por 4 das 9 empresas que utilizam a arquitetura de microsserviços, e o Spring Boot o segundo mais utilizado, indicado por 3 empresas.

Na Figura 5 são apresentados os resultados referente a questão "Qual o meio de comunicação utilizado entre os microsserviços?". Pode-se observar que predominantemente é utilizada a comunicação por APIs REST, abrangendo 66,7% e a

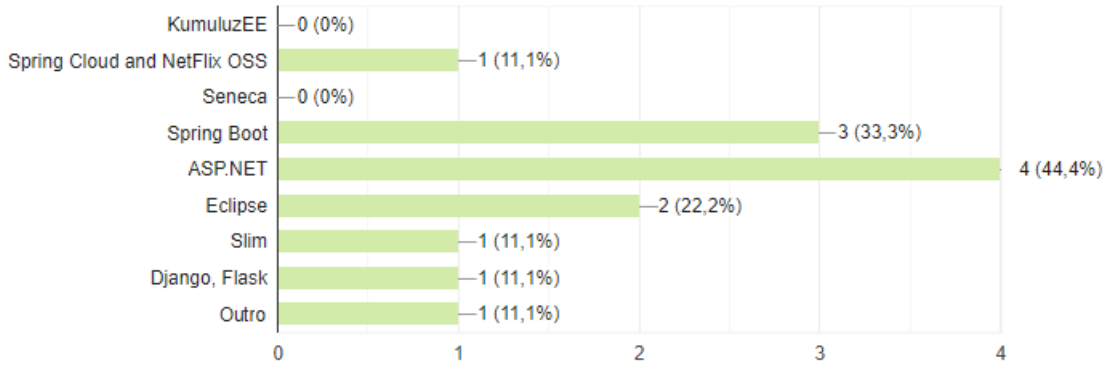


Figura 4. Frameworks utilizados para desenvolvimento de microserviços.

opção SOAP não é utilizada por nenhuma das empresas. Para [Balalaie et al. 2015] e [Fowler and Lewis 2014] microserviços visam transformar sistemas de software em pacotes de pequenos serviços, cada um entregável em uma plataforma diferente e rodando seus próprios processos através de uma comunicação com mecanismos leves como, por exemplo, RESTFull APIs ou API HTTP. Sendo assim, as empresas que utilizam APIs REST ou HTTP, que na pesquisa realizada corresponderam a 77,8%, também trabalham de acordo com o proposto pela literatura.

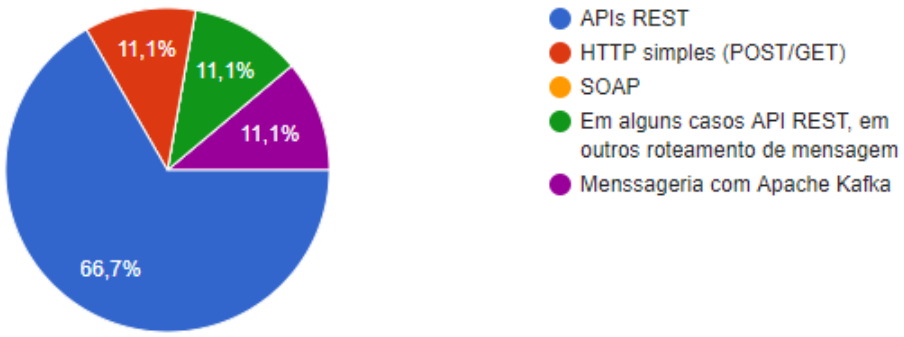


Figura 5. Tipo de comunicação utilizada entre os microsserviços.

A Figura 6 apresenta os percentuais obtidos com a aplicação de uma questão sobre a estrutura do banco de dados utilizada pelas empresas que utilizam microserviços. Dentre as respostas, a estrutura mais utilizada é a de um banco de dados dedicado a cada microsserviço, 55,6 %. Para [Meloca 2017], cada microsserviço possui seus próprios dados, sua própria regra de negócio e uma interface bem definida, de modo que cada serviço deve ter seu próprio repositório, controlador de versões, ciclo de vida e controlador de tarefas. Levando isso em conta, pode-se entender que, ao optar por um banco de dados para

cada serviço, espera-se ter melhor desempenho e menor acoplamento entre os serviços, benefícios almejados ao utilizar a arquitetura de microsserviços.

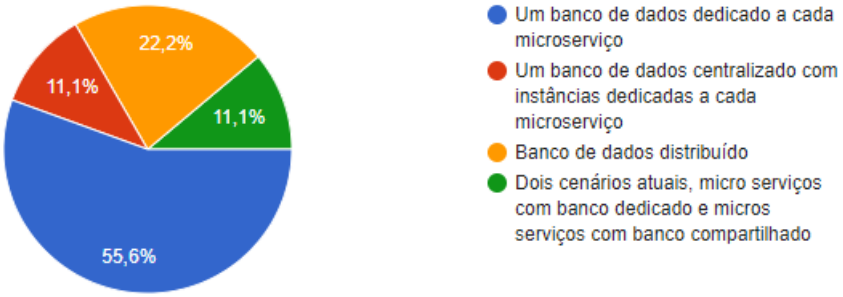


Figura 6. Estrutura de banco de dados para suporte aos microserviços.

Uma questão sobre às ferramentas de integração utilizadas pelas empresas também foi aplicada, sendo as opções de resposta: *TeamCity*; *Jenkins*; *Azure*; *nenhuma* e *outro* com a possibilidade do respondente citar a ferramenta utilizada. Como pode ser visto na Figura 7, a ferramenta mais utilizada é a *Jenkins*. Essa questão é relevante pois uma característica dos microsserviços é a facilidade do *deploy*, o que geralmente ocorre por meio de integração contínua, e as ferramentas mencionadas tem a proposta de facilitar essa integração contínua. Segundo [Balalaie et al. 2015] a arquitetura de microsserviços está intimamente ligada ao conceito de entrega contínua. Na prática, isso só é possível ao adotar ferramentas apropriadas que promovam essa integração do projeto com o *deploy*.

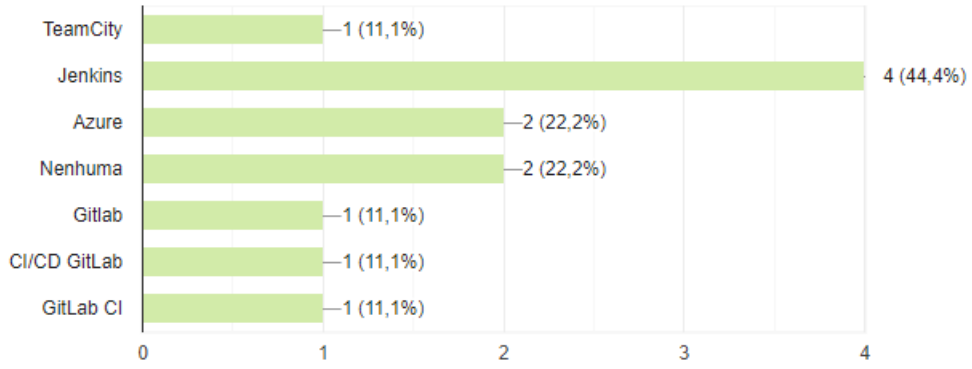


Figura 7. Ferramentas para realização de integração contínua.

A Figura 8 se refere as respostas da pergunta "Qual ferramenta de controle de log utiliza?". As opções dadas foram: *Flume*; *ELK*; *Fluentd*; *nenhuma* e *outra* com opção de indicação da ferramenta utilizada. A maior parte das empresas, como pode ser observado no gráfico, utiliza alguma ferramenta de controle de log, porém não há uma que mais se destaca. Um dos respondentes marcou a opção outra e descreveu a forma do trabalho como segue: "Registros de eventos no banco de dados — log dos frameworks de roteamento". O controle de log é uma importante ferramenta no momento de rastrear um problema e permite diagnosticar anormalidades em relação ao propósito

do sistema e questões de segurança e acessibilidade. O registro de logs é gerado e incrementado ao longo do tempo, e possui informações que permitem diagnosticar anormalidades em relação ao propósito do sistema e questões de segurança e acessibilidade. Para [Brown 2016], os seguintes conceitos são importantes na implementação da arquitetura de microsserviços: ID de Correlação, Agregador de Log e Disjuntor. Mais especificamente sobre o agregador de log, ele complementa os IDs de Correlação permitindo que os logs de diversos microsserviços diferentes sejam agregados em um único repositório que possa ser pesquisado. Juntos, eles permitem a depuração eficiente e compreensível de microsserviços, independentemente do número de serviços ou da profundidade de cada pilha de chamada.

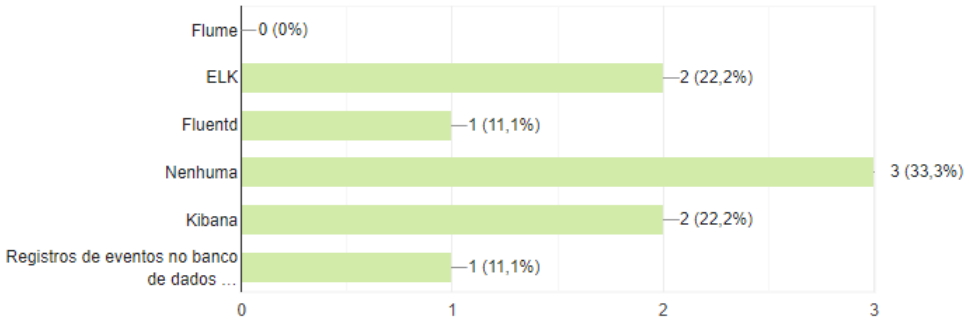


Figura 8. Ferramentas para controle de log.

As duas últimas questões da seção de perguntas para empresas que utilizam a arquitetura microsserviços foram discursivas, onde a empresa poderia responder de forma livre. Dentre as 9 empresas que utilizam a arquitetura de microsserviços apenas 5 responderam essas questões. Esses respondentes foram nomeados P1, P2, P3, P4 e P5.

A primeira questão discursiva realizada relaciona-se às preocupações da empresa com a granularidade dos microsserviços. Com base nas respostas podemos observar que a principal preocupação das empresas é o domínio dos serviços e a manutenção do código. É possível observar esse padrão na resposta fornecida por P1 *"Manutenção de código/estrutura do banco, balanceamento de carga"* ou ainda pela resposta fornecida por P2 *"Responsabilidade única, a orientação é que cada microsserviço seja responsável por um contexto específico"*. Para [Newman 2015], microsserviços são pequenos serviços autônomos que trabalham juntos. Sendo assim, um princípio base da arquitetura de microsserviços é justamente a granularidade dos serviços.

A segunda questão discursiva refere-se sobre o gerenciamento do *overhead*, "Como a empresa tem gerenciado o overhead (balanceamento de carga)?". Pode-se observar que a maior parte das empresas adota como estratégia utilizar ferramentas para o controle do *overhead*. Na resposta de P3, por exemplo, *"Frameworks escolhidos para roteamento/balanceamento"*, ou ainda na resposta de P4 *"Utilizamos Kubernetes rodando no EKS da Amazon"e na resposta de P5 "Nginx, AWS EC2 ELB"*. Como destaca [Balalaie et al. 2015], para ser escalonável, um aplicativo deve ser capaz de distribuir a carga de um serviço individual entre suas várias instâncias. Esse é o dever de um balanceador de carga e, nesse caso, deve obter instâncias disponíveis do componente de descoberta de serviço. Como o escalonamento é um dos benefícios da arquitetura de

microsserviços a empresa deve se preocupar com o *overhead* do sistema.

A parte final do questionário aplicado neste trabalho, algumas questões foram dedicadas para entender por que as empresas não utilizam a arquitetura de microsserviços.

A primeira questão para as empresas que não utilizam foi se a empresa possui conhecimento sobre o tema, sendo que a grande maioria, 71,4%, apesar de não utilizar, alegou ter conhecimento sobre o microsserviços, enquanto que 28,6% não possuem conhecimento.

A segunda questão foi "Qual arquitetura utiliza atualmente?". Foi constatado que todas utilizam arquitetura monolítica. Mesmo SOA sendo uma arquitetura importante, não foram encontradas, dentro da amostra estudada, empresas em Maringá - PR que utilizem essa arquitetura.

A última questão objetivou indentificar a possibilidade/intenção de uma migração dos sistemas da empresa para a arquitetura de microsserviços. Nesse sentido 42,9% dos respondentes acreditam que *talvez seja possível* a migração, 28,6% indicam que *é possível* a migração e outros 28,6% acreditam que *não é* possível a migração.

Das empresas que não utilizam a arquitetura de microsserviços, 71,4% alegam ter conhecimento sobre a tecnologia de microsserviços. Além disso 42,9% acreditam que talvez seja possível realizar a migração para esse estilo arquitetural e 28,9% acreditam que é possível realizar tal migração. Essas informações indicam um cenário positivo para uma futura adoção desse estilo arquitetural pelas empresas participantes da pesquisa.

Com o objetivo de consolidar algumas das informações obtidas pelas empresas que utilizam microserviços a Tabela 1 apresenta o cruzamento dos dados sobre a escolha da arquitetura e os critério de extração dos microsserviços adotados pelas empresas. No total, 5 empresas estão realizando a migração gradual, 2 empresas realizaram a migração imediata, 1 empresas está realizando a migração atualmente e 1 empresa construiu o software já utilizando a arquitetura de microsserviços. Após observar esses dados pode-se notar que o segundo fator mais relevante para extração dos microsserviços por empresas que realizaram a migração gradual é a reusabilidade. Já para as empresas que realizaram a migração imediata são os requisitos. O critério de acoplamento também é uma escolha das empresas que estão passando por migração ou o sistema já foi construído utilizando a arquitetrua de microsserviços.

5. Conclusão

Este trabalho teve como objetivo realizar um estudo exploratório sobre o uso da arquitetura de microsserviços por empresas de software de Maringá - PR. Com base nos resultados apresentados e na análise dos mesmos, pode-se concluir que há um número expressivo de empresas de software dentre as que responderam o questionário que utilizam a arquitetura de microsserviços, sendo 9 das 23 que participaram da pesquisa. Também é possível observar, conforme a análise dos dados, que as empresas que utilizam a arquitetura de microsserviços, em geral, trabalham conforme as boas práticas já apresentadas e descritas na literatura. Das empresas que não utilizam a arquitetura, pode-se observar que há o conhecimento sobre o tema e muitas empresas acreditam que uma migração seria viável.

É possível expandir a pesquisa ao âmbito nacional, possibilitando a comparação entre a forma de trabalho regional, observada na cidade de Maringá - PR, e a forma de

Estratégia de migração	Critérios para extração dos serviços	Nº de empresas
Migração gradual	Escalabilidade	5
	Reusabilidade	4
	Coesão	1
	Acoplamento	3
	Requisitos	1
Migração imediata	Escalabilidade	2
	Acoplamento	1
	Requisitos	2
	Coesão	1
	Tabelas do banco de dados	2
Em processo de migração	Escalabilidade	1
	Acoplamento	1
	Performance, manutenibilidade	1
Foi construido assim	Escalabilidade	1
	Reusabilidade	1
	Acoplamento	1

Tabela 1. Estratégia de migração e critérios para extração dos microsserviços.

trabalho encontrada em outras regiões do país. Também pode ser expandida esta pesquisa para outros aspectos não abordados ou não explorados de forma profunda nesta pesquisa, tais como aspectos da migração dos sistemas, os benefícios obtidos com o uso da arquitetura de microsserviços, entre outros tópicos.

Como forma de divulgação dos resultados da pesquisa em âmbito privado, os autores pretendem promover lives e palestras no futuro para discussão do tema.

Referências

Balalaie, A., Heydarnoori, A., and Jamshidi, P. (2015). Migrating to cloud-native architectures using microservices: an experience report. In *European Conference on Service-Oriented and Cloud Computing*, pages 201–215. Springer.

Bass, L., Clements, P., and Kazman, R. (2003). *Software architecture in practice*. Addison-Wesley Professional.

Brown, K. (2016). Além de palavras da moda: um breve histórico sobre padrões de microsserviços.

Di Francesco, P., Lago, P., and Malavolta, I. (2019). Architecting with microservices: A systematic mapping study. *Journal of Systems and Software*, 150:77–97.

Fowler, M. and Lewis, J. (2014). Microservices.

Meloca, R. M. (2017). Um comparativo entre frameworks para microsserviços. B.S. thesis, Universidade Tecnológica Federal do Paraná.

Newman, S. (2015). *Building microservices: designing fine-grained systems*. "O'Reilly Media, Inc.".

Explorando o Refinamento de uma DSL para Versões Baseadas em EMF, Eclipse Sirius e XText

Jaqueline Moura[1], **Natiele Lucca**[1], **Fábio P. Basso**[1], **João Pablo S. da Silva**[1], **Elder Rodrigues**[1], **Maicon Bernardino**[1]

[1]Campus Alegrete – Universidade Federal do Pampa (UNIPAMPA) – Alegrete – RS – Brasil

{moura.jak,natielelucca}@gmail.com

{fabiobasso,joaosilva,elderrodrigue}@unipampa.edu.br, bernardino@acm.org

Abstract. *Model Driven Engineering (MDE) allows specify abstract models at high levels and then transform them. Self-adaptive systems are able to assess and change their own behavior at run time. Domain-specific languages (DSLs) can be described to abstract out such complexity. This article presents an exploratory study on the refinement of a DSL in this domain. Three versions are implemented: one textual (in Xtext) and two graphics (based on EMF tree and Eclipse Sirius). Afterwards, transformations of these models are carried out for target platform code. Finally, conclusions from the practice obtained with the apparatus used in the Eclipse ecosystem are exposed.*

Resumo. *A Engenharia Dirigida por Modelos (MDE) possibilita descrever modelos abstratos em altos níveis e então transformá-los. Sistemas autoadaptativos são capazes de avaliar e alterar seu próprio comportamento em tempo de execução. Linguagens específicas de domínio (DSLs) podem ser descritas para abstrair tal complexidade. Este artigo apresenta um estudo exploratório no refinamento de uma DSL desse domínio. Três versões são implementadas: uma textual (em Xtext) e duas gráficas (baseada em árvore EMF e Eclipse Sirius). Após, são realizadas transformações destes modelos para código de plataforma alvo. Finalmente, são expostas conclusões da prática obtida com o aparato utilizado no ecossistema do Eclipse.*

1. Introdução

Um sistema autoadaptativo é capaz de avaliar e alterar seu próprio comportamento, sempre que a avaliação mostrar que o software não está realizando o que se pretendia fazer, ou quando uma melhor funcionalidade ou desempenho pode ser possível. Assim, a investigação de abordagens sistemáticas de engenharia de software é necessária, a fim de desenvolver sistemas autoadaptativos que possam, idealmente, ser aplicados em vários domínios [Macás-Escrivá et al. 2013]. Estes sistemas podem ser aplicados a diversas áreas de conhecimento como: 1) saúde – monitoramento de sinais vitais, pressão arterial e temperatura); 2) aplicações tecnológicas como veículos autônomos; 3) dispositivos móveis como controlar a luminosidade e o acelerômetro; entre outros[1].

[1]Mais exemplos de sistemas auto-adaptativos podem ser encontrados em https://www.hpi.uni-potsdam.de/giese/public/selfadapt/exemplars/

Sistemas autoadaptativos vem sendo alvo de investigação em pesquisas aplicadas com MDE [Schmidt 2006]. MDE é uma abordagem de desenvolvimento de software que se concentra na criação de modelos que descrevem os elementos de um sistema. Os modelos respeitam um modelo preciso (sintaxe) e um significado (semântica) [Petricic et al. 2008] conforme uma Linguagem Específica de Domínio (DSL) [Voelter 2009].

Pesquisas sobre MDE identificaram a carência literária de estudos exploratórios considerando diferentes tecnologias de construção de DSLs [Neto et al. 2020, Neto et al. 2019]. Em virtude disso, apresenta-se neste artigo um estudo exploratório caracterizando três opções disponíveis no ecossistema do Eclipse: EMF, Eclipse Sirius e XText. Para tal, uma investigação para abstrair os elementos de sistemas autoadaptativos em modelos foi conduzida, tendo como eleita uma versão preliminar e simplificada da DSL denominada SaSML [da Silva 2018]. Nossa contribuição é um relato de experiência sobre o desenvolvimento de três protótipos desta mesma DSL, utilizando estratégias de representação diferentes.

O restante do trabalho está organizado da seguinte maneira: A Seção 2 discute a metodologia adotada nessa pesquisa exploratória. A Seção 4 introduz o metamodelo da SaSML e ambas as versões de DSL criadas. A Seção 3 apresenta brevemente as tecnologias utilizadas. O mapeamento e transformação de modelos é discutido na Seção 5. Por fim, a Seção 6 aborda trabalhos relacionados e a Seção 7 as considerações finais acerca do trabalho.

2. Motivação

Como exemplificação do estudo, a Figura 1(A) apresenta uma versão preliminar dos elementos esperados da DSL SaSML. Ela introduz o elemento AdaptiveBehavior como forma de encapsular um modelo complexo composto de contextos e comportamentos. Este exemplo descreve o identificador do elemento como Comportamento-Adaptativo, as informações de contextos (C) e os comportamentos alternativos(B).

Alguns desafios para a concepção dessa DSL como uma ferramenta importante para a área incluem: 1) uma vez que a transferência de tecnologia de MDE é altamente dependente de experiências, percebe-se que não há um corpo de conhecimento em português que relate sobre a utilização de diferentes tecnologias na construção da mesma DSL; 2) uma vez que algumas DSLs em nossos grupos de pesquisa estão em seu estágio embrionário, não se tem dados práticos sobre qual tecnologia é viável para construí-las; e 3) uma vez que cada domínio de aplicação pode ter maior aceitação por determinados tipos de DSL [Voelter 2009], não se tem dados sobre qual forma de representação (se em árvore, gráfica ou a textual) é mais bem aceita pelos usuários finais na modelagem de sistemas em geral.

Diante destes desafios, buscou-se resolver o primeiro. Nosso foco neste estudo é realizar uma pesquisa exploratória sobre três tecnologias na construção da SaSML: EMF [Steinberg et al. 2008], Eclipse Sirius [Eclipse Sirius 2018] e XText [XText 2018]. Portanto, apresenta-se uma fundação que serve como base para a execução das próximas pesquisas, mais empíricas, sobre os dois desafios subsequentes.

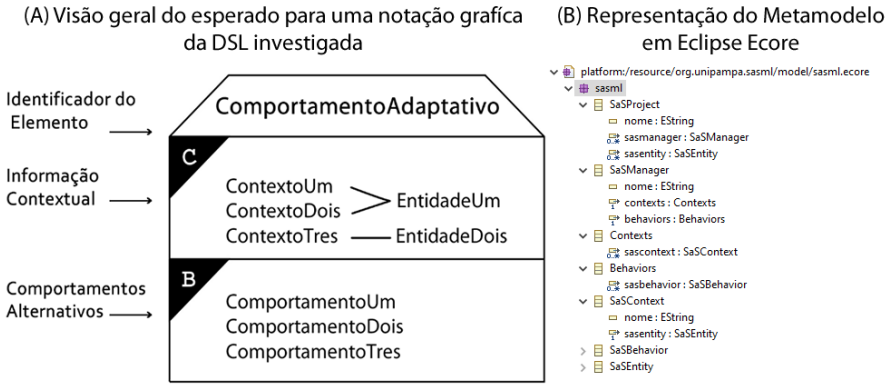


Figura 1. Elementos representacionais da DSL e Metamodelo Ecore

3. Tecnologias Utilizadas

Esta seção descreve brevemente as principais tecnologias que foram integradas para produzir esse trabalho.

3.1. EMF

O Eclipse Modeling Project provém e evolui diversas tecnologias de desenvolvimento baseada em modelo. Seu guarda chuva possui um conjunto unificado de *frameworks* de modelagem, implementações de padrões e fornece o aparato ferramental para sua utilização.

Dentro de um projeto de modelagem, precisamos inicialmente definir a estrutura do modelo de domínio, essa definição é denominada meta modelo. No eclipse, o meta modelo é determinada pela linguagem Ecore por meio do editor EcoreTools. Modelos Ecore possuem suporte de tempo de execução, notificação quando alterações são realizadas e são persistidos em XMI serializado. Sendo, portanto, parte essencial do *framework* EMF.

3.2. Eclipse Sirius

O eclipse Sirius utiliza as tecnologias mencionadas anteriormente para produzir um ambiente de modelagem integrado. Com ele é possível definir uma bancada de trabalho com todas as ferramentas necessárias para a produção de modelos gráficos personalizados, como diagramas, tabelas e arvores [**?**]. Ele provém facilidade de especificação de todas as características do modelo, comportamentos da linguagem modelada e regras validação sincronizada dinamicamente. Além disso, ele permite a especificação de diversas representações *(viewpoint especification)* para um único modelo. Podendo então, prover diversas camadas de visualização de acordo com as diferentes preocupações de cada tipo de usuário.

3.3. Xtext

Xtext é uma estrutura do Eclipse para implementar linguagens de programação e DSL textuais. Permite implementar idiomas rapidamente e, acima de tudo, abrange todos os aspectos de uma infra-estrutura completa de linguagem, desde o analisador, gerador de código, ou intérprete, até uma integração completa do Eclipse IDE com

todos os recursos típicos da IDE [Bettini 2016]. Ele também depende do EMF e utiliza seus modelos como representação na memória de qualquer arquivo de texto analisado.

4. Execução do Estudo

O objeto de estudo é uma versão preliminar da SaSML, uma extensão UML que mapeia as abstrações e restrições de nível mais alto de sistemas autoadaptativos ao introduz um novo elemento chamado AdaptiveBehavior para fornecer suporte adequado à modelagem conceitual desses sistemas[2]. Para desenvolver esta DSL, primeiro, temos que modelar o conceito, os relacionamentos e as restrições do domínio. Depois, temos que identificar os elementos UML com a semântica adequada para as nossas necessidades, implementar as extensões e verificar a integridade do modelo.

4.1. Metamodelo em EMF

Um metamodelo Ecore comporta a criação de DSLs em EMF, Xtext e Eclipse Sirius. Portanto, realizamos a definição do metamodelo da linguagem exibido na Figura 1(B)) a partir da identificação dos elementos que compõe o AdaptiveBehavior. A seguir os componentes do metamodelo são detalhados.

Como componente raiz denominou-se o SaSProject. Um SaSProject é composto de zero ou muitos SaSManager e zero ou muitas SaSEntity. Um SaSManager deve levar o nome do Identificador do Elemento e é composto de um Contexts e um Behaviors, compartimentos demarcados na parte superior com um triangulo preto contendo as letras C e B, respectivamente, na Figura 1(A).

Através do Contexts obtém-se acesso às informações de contexto, ele é composto pelos SaSContext e deve ser associado a SaSEntity respectiva. Os comportamentos que o sistema pode adotar em tempo de execução são acessados a partir de Behaviors, que está associado a zero ou muitos SaSBehaviors.

A partir do metamodelo gerou-se três plug-ins para o Eclipse, discutidos logo mais. A apresentação da versão em árvore do Eclipse EMF foi omitida por motivos de espaço. Ela é parte necessária para a geração do *plugin* em Eclipse Sirius.

4.2. Versão SaSML em Modo Textual com Xtext

A versão desenvolvida em Xtext possui elementos compostos por uma descrição e um identificador - nome. Onde são determinadas regras, o início de um projeto deve possuir o estereótipo SaSProject em seguida deve haver uma cadeia identificadora e um símbolo "{" . Em sequência, podem ser inseridos SaSEntity ou SaSManager.

A regra para o elemento SaSEntity é apenas um identificador seguido dos símbolos "{}", que posteriormente pode evoluir e possuir atributos. Já o SaSManager possui as mesmas regras que o anterior, além dos atributos SaSConext e SaSBehavior. O SaSContext possui um nome um simbolo " : " e uma referencia aos elementos SaSEntity já criados, caso esteja vazio (ainda não descritos) será apresentado erro sintático. O SaSBehavior possui um identificador e uma opção de inserir ou não o

[2]Para fins de simplificação do estudo a especificação restringiu-se ao novo elemento

tipo de retorno (caso não seja informado não mostra erro ou inconsistência). Ao fim é necessário informar o(s) "}".

Os elementos podem ser repetidos, apenas não é o caso do SaSProject. Logo, para que fosse possível inserir um novo elemento sem sobrescrever o anterior, foi necessário informar a estrela de uma linguagem, ou seja, inserir o símbolo " * " no bloco correspondente as repetições. A Figura 2(B) representa o elemento ControladorAnticolisao no modo textual no *framework* Xtext. Essa linguagem é baseada em conceitos de compiladores, como podem ser vistos na Figura 2(A) como uma gramática elaborada com o Xtext que define a sintaxe da DSL.

A) Gramática da linguagem no Xtext, gerada pela transformação do modelo Ecore

```
grammar org.edm.projmde.ProjMde with org.eclipse.xtext.common.Terminals

generate projMde "http://www.edm.org/projmde/ProjMde"

Model:
    {Model} ('SaSProject' nomeProjeto=ID '{' ( entidades+=Entity)*
                    ('SaSManeger' (nameElemento=ID) '{'
                                (elements+=Manager)*
                    '}')*'}'    )
;

Manager:
    AbstractContext | AbstractBehavior ;

Entity:
     'SaSEntity' name=ID  '{}';

AbstractContext:
    'SaSContext' nomeContext=ID ':' ref=[Entity];

AbstractBehavior:
    'SaSBehavior' nomeBehavior=ID (retornoBehavior=ID)?;
```

B) Representação textual de um modelo conforme a DSL SaSML, construída em Eclipse Xtext

```
SaSProject SaSML{

        SaSEntity Veiculo {}
        SaSEntity  Objeto {}

        SaSManeger ControladorAntiColisao {
            SaSContext Velocidade : Veiculo
            SaSContext Distancia : Objeto
            SaSBehavior Desaceleracao void
            SaSBehavior Frenagem void

        }

}
```

Figura 2. Gramática da linguagem em Xtext, gerada na transformação do Ecore

4.3. Versão SaSML em Modo Gráfico com Eclipse Sirius

No eclipse Sirius, as notações gráficas são definidas em projeto de especificação de ponto de vista. Portanto, o primeiro passo foi cria-lo. A partir do modelo especificação de ponto de vista *(Viewpoint Specification Model)*, temos acesso ao editor em árvore Sirius, ele permite determinar a respectiva representação de cada elemento Ecore. Fornecendo portanto, todas as ferramenta necessárias para definir cada elemento que compõe o modelo.

Os componentes que compõe a DSL são definidos de acordo com a especificação do metamodelo conforme explicado a seguir: 1) Cria-se o elemento gráfico (contêiner, nó ou aresta); 2) Determina-se sempre a qual classe de domínio ele corresponde e quais os candidatos semânticos irão ser representados naquele elemento especifico; 3) Define-se de que modo ele será exibido, para tal, isso são fornecidas diversas personalizações de acordo com o elemento selecionado; 4) Opcionalmente, define-se customizações e regras de validação adicionais.

Esse processo, aplicado a SaSML, teve como resultado a Figura 3(A), que é aprofundado a seguir. Na tentativa de uma representação mais fiel à notação gráfica, optou-se pela representação em diagrama. Ele utiliza o metamodelo o Ecore SaSML discutido anteriormente tem como classe de domínio a raiz SaSProject.

Previsivelmente, o elemento AdaptiveBehavior foi definido como um contêiner que é composto por seus três compartimentos de identificação, de controle e

comportamental, que também são contêineres e são detalhados hierarquicamente a seguir.

No topo, encontra-se o compartimento identificador que possui seu estilo característico, a figura de trapézio, para facilitar o alinhamento do rótulo do SaS-Manager, um nó foi criado para exibi-lo.

Na parte central, está o compartimento contextual. Como pode ser visto no elemento proposto, ele é composto de duas colunas verticais, a esquerda estão os contextos (SaSContext) propriamente ditos, e a direita as entidades (SaSEntity) que estão sendo mapeadas. Para que a representação aconteça de maneira correta, foram criados dois contêineres de alinhamento vertical, subordinados ao contêiner de alinhamento horizontal de contexto que mapeia a classe Contexts. Embora uma entidade possa estar relacionadas a mais de um contexto, pela notação ela deverá ser representada apenas uma vez e terá múltiplas arestas ligando os contextos a ela. Para evitar essa duplicação, para SaSEntitys de eleição de candidatos semânticos foi definido um método, na classe de serviços *(Services)*, que recebe como parâmetro o Contexts, e por meio dele acessa os seus SaSContexts buscando suas respectivas SaSEntity e os adicionada em um conjunto (Set) e por fim o retorna, portanto, sem repetições.

Na parte inferior, o compartimento de comportamentos foi definido, ele uma coluna com todos os comportamentos alternativos. Seu estilo foi definido utilizando uma figura de retângulo, que contem a marcação na ponta superior (um triangulo preto com a letra B), processo semelhante foi realizado no compartimento anterior, porém, com sua letra respectiva, a C.

Finalizando, foi criada uma aresta *(Edge)* que representa o relacionamento entre SaSContext e seu respectivo SaSEntity, mapeando os como fonte *(Source)* e alvo *(Target)* respectivamente. Por não ser o objetivo do estudo a criação de um editor SaSML, não foram criadas ferramentas, filtros ou outras camadas de visualização para manipular esses elementos.

Um exemplo da representação gráfica é exibido na figura 3(B). Ela exibe um fragmento de um modelo de carro autoadaptativo, responsável por realizar seu controle anticolisão.

5. Mapeamento e Transformações de Modelos

Uma vez que as DSLs foram desenvolvidas, modelos de cada versão foram criados e deu-se seguimento ao processo de desenvolvimento para testar as mesmas. Nesta seção são discutidas as transformações que assistem o processo, permitindo a manipulação do modelo SaSML de EMF (XText, Eclipse Sirius) em nível independente de plataforma para a plataforma alvo.

5.1. Mapeamento da SaSML para Plataforma Alvo

O modelo SaSML proposto em nível independente de plataforma, apresentado na Figura 3 pode ser mapeado para plataformas específicas, como linguagens e frameworks de desenvolvimento de sistemas auto-adaptativos. A plataforma alvo testada corresponde ao conjunto de características presentes no domínio expresso na Figura 4, um

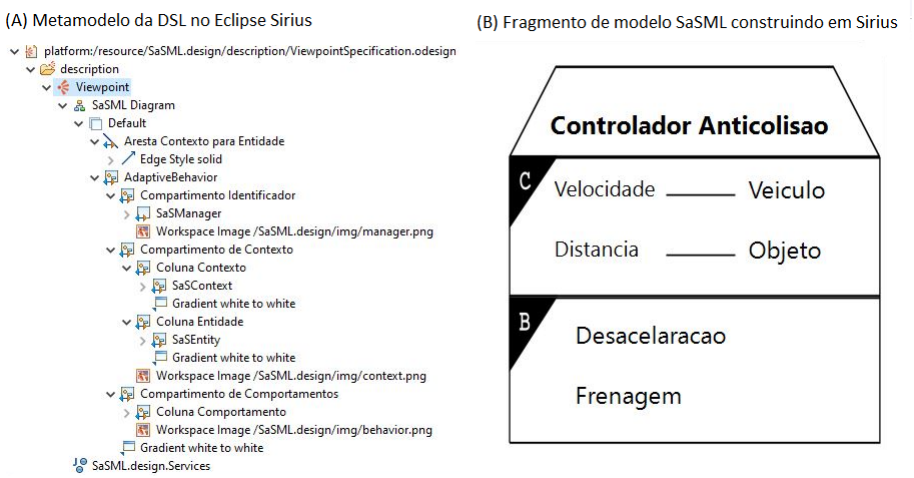


Figura 3. Construção da DSL em Eclipse Sirius

Framework contendo as associações entre as classes AbstractEntity, AbstractBehavior, AbstractManager, AbstractContext. Cada componente do modelo foi mapeado com a devida herança para a super classe respectiva.

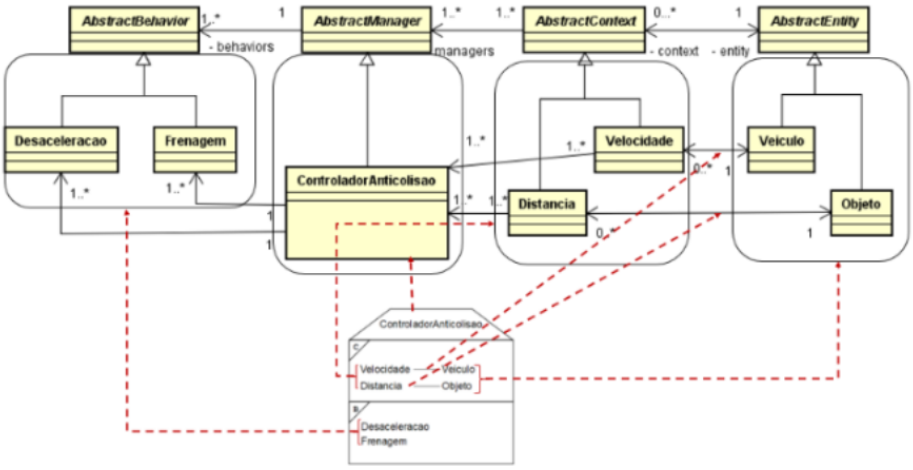


Figura 4. Mapeamento de elemento do modelo "ControladorAnticolisão"para classes específicas de um framework

5.2. Transformação para a Plataforma Alvo

A transformação permite migrar de um modelo a outro a partir de uma estrutura lógica programada de recuperação inteligente. Com auxílio da API WCT, realizamos uma prova de conceito, transformando o modelo gerado para a Plataforma Alvo. Onde identificamos as classes envolvidas (AbstractEntity, AbstractManager, AbstractContext e AbstractBehavior).

A transformação foi realizada por meio da API WCT [Basso et al. 2014]. A WCT possui um aparato robusto para manipulação modelos, permitiu realizar as associações e heranças necessárias entre as classes do modelo.

Inicialmente importou-se o modelo em SaSML, bem como o arquivo com as classes da plataforma. Uma vez que o elemento raiz do metamodelo da SaSML é o

SaSProject, usamos ele para ter acesso aos demais elementos. Para todo SaSEntity presente no SaSProject, foi criado uma classe com seu respectivo nome herdado de AbstractEntity. Para todo SaSManager, presente no SaSProject, foi criado uma classe com seu respectivo nome e realizada a herança com a classe AbstractManager. Por meio de cada AbstractManager, acessou-se Contexts e Behaviors contido nele.

Utilizou-se o Contexts para acessar os SaSContext. Para todo SaSContext, foi criado uma classe com seu respectivo nome que herda de AbstractContext e é associado a sua respectiva SaSEntity e ao SaSManager no qual ele se encontrava. Utilizamos o Behaviors para acessar os SaSBehaviors. Para cada SaSBehavior, foi criado uma classe com seu respectivo nome que herda de AbstractBehavior e também é associado com o SaSManager no qual ele se encontrava. A prova de conceito realizada foi um sucesso, a API Fomda [Basso et al. 2014] gerou o arquivo de saída XMI contendo todas as associações representando o modelo gerado conforme a UML. O fragmento do transformador responsável pela de transformação é exibido em Figura 5(A) Utilizamos a ferramenta WCT para visualização do arquivo e obtivemos o resultado apresentado abaixo exibido na Figura 5(B).

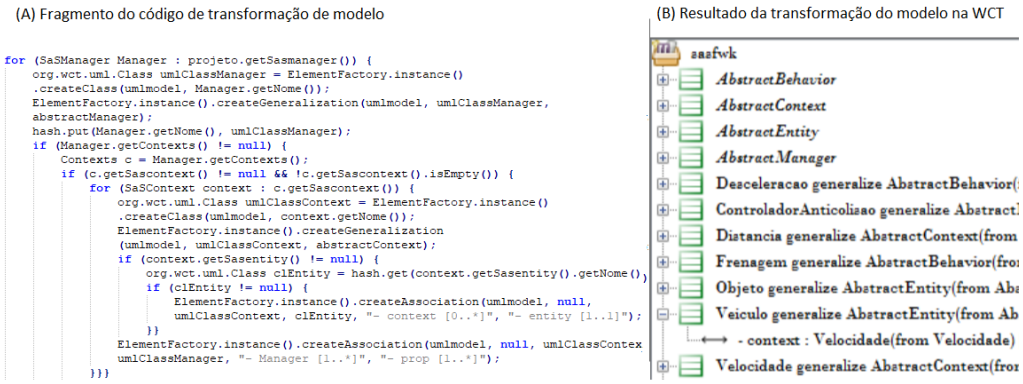


Figura 5. Elementos finais de um processo de transformação de modelos

6. Trabalhos Relacionados

Em [Lima et al. 2019] os autores comparam características de dois construtores de DSLs textuais: o Eclipse *workbench* para XText e *workbench* MPS. Este é um estudo analítico que descreve os pontos positivos e negativos de cada abordagem. A conclusão é que o MPS apresenta características mais interessantes para a construção de DSLs textuais. Em [Favero et al.] os autores apresentam um estudo exploratório para identificar os pontos positivos e negativos do Eclipse *workbench* disponível para a construção de DSLs em ambientes web. O estudo inclui as tecnologias EMF, EMF Forms e Angular, concluindo que apesar de o *workbench* possuir mecanismos de transformações entre estas tecnologias, o refinamento para uma DSL em estágio final necessita de muitos ajustes manuais.

Nosso estudo é complementar e investiga um aparato de tecnologias diferente, dentro do ecossistema do Eclipse para a construção de DSLs. Observou-se que este ecossistema é integrado e permite migrar de uma representação em árvore (EMF) para representação textual e gráfica por meio de transformações. No entanto,

este não é um procedimento linear, pois os artefatos gerados sempre necessitam de refinamento.

7. Conclusão

Uma vez que cada domínio de aplicação pode ter maior aceitação por determinados tipos de DSLs, este estudo exploratório apresenta um relato de experiência no desenvolvimento de três versões de um DSL chamada SaSML. Apoiados no metamodelo em EMF, especificou-se versões em modo árvore (o padrão do EMF), modo textual (Xtext) e em modo gráfico (Eclipse Sirius). Através destas representações, também realizou-se o mapeamento para a plataforma alvo, o que simplificou a compreensão e o processo de transformação entre os modelos.

Sobre a cadeia de ferramentas utilizadas para o desenvolvimento da prova de conceito, destacam-se as seguintes observações: 1) A API WCT permitiu agrupar as informações das versões geradas e transformar o modelo para a plataforma alvo, sendo uma boa opção para o desenvolvimento das provas de conceito futuras; 2) Os *plugins* existentes dentro do Eclipse Modelling Tools (EMT) são integrados, possuem recursos integrados que viabilizam uma simples recuperação e manipulação das informações do metamodelo, e de forma eficiente migram informações do metamodelo para configurações para protótipos executáveis para EMF em Árvore, Eclipse Sirius e XText; e 3) O conjunto ferramental adotado é possível de assimilação por estudantes de graduação sem experiência em MDE, apesar de que observou-se bastante dificuldades na etapa de análise de domínio para extrair as informações gráficas da DSL em metaclasses do metamodelo EMF.

Não foram realizadas avaliações empíricas por meio de grupo focal nas DSLs implementadas. Ficam como pontos em aberto, portanto, os dois outros desafios motivados: 1) uma vez que DSLs em nossos grupos de estudo estão em seu estágio embrionário e não se tem dados empíricos sobre qual tecnologia é viável para construí-las, pretende-se seguir no estudo de outras alternativas como UML Profiles [Schmidt 2006], METACASE [Kelly and Tolvanen 2008] e outras alternativas híbridas [Addazi et al. 2017]; e 2) pretende-se avaliar, com grupo focais de alunos de Engenharia de Software e de Ciência da Computação, qual forma de representação (árvore, gráfica ou textual) é mais aceita para representação de diversos domínios de aplicações investigados em nosso grupo de pesquisa, como: de sistemas de automação e controle (via a DSL M4PIA), sistemas de informações (via a DSL MockupToME) e sistemas de IoT (via a DSL MARTE Profile).

8. Agradecimentos

Este estudo foi parcialmente financiado através da Pró-Reitoria de Pesquisa (PROPESQ), com bolsa do programa AGP (Apoio a Grupos de Pesquisa), e pela FAPERGS, por meio do projeto ARD N. 19/2551-0001268-3.

Referências

Addazi, L., Ciccozzi, F., Langer, P., and Posse, E. (2017). Towards seamless hybrid graphical–textual modelling for uml and profiles. In *Modelling Foundations and Applications*, pages 20–33. Springer International Publishing.

Basso, F. P., Pillat, R. M., Oliveira, T. C., and Fabro, M. D. D. (2014). Generative adaptation of model transformation assets: Experiences, lessons and drawbacks. In *Proceedings of the 29th Annual ACM Symposium on Applied Computing*, SAC '14, pages 1027–1034.

Bettini, L. (2016). *Implementing domain-specific languages with Xtext and Xtend*. Packt Publishing Ltd.

da Silva, J. P. S. (2018). Sasml : a uml based domain specific modeling language for self adaptive systems conceptual modeling.

Eclipse Sirius (2018). Disponível em : <https://www.eclipse.org/sirius/doc/> acesso em junho de 2018.

Favero, E. S., de Oliveira, I. A., nez, P. S. Z. N., and Basso, F. P. Estudo exploratório no refinamento de uma dsl para versões baseadas em emf, emf forms e angular.

Kelly, S. and Tolvanen, J.-P. (2008). *Domain Specific Modeling: Enabling Full Code Generation*. IEEE Computer Society - John Wiley & Sons.

Lima, Y. A., Modesto, S., Medeiros, J. M., Carbonell, J. B. P., and de Macedo Rodrigues, E. (2019). Mps x xtext: Uma comparação de languages workbenches para o desenvolvimento de dsls. In *Anais da III Escola Regional de Engenharia de Software*, pages 97–104, Porto Alegre, RS, Brasil. SBC.

Macás-Escrivá, F. D., Haber, R., del Toro, R., and Hernandez, V. (2013). Self-adaptive systems: A survey of current approaches, research challenges and applications. *Expert Systems with Applications*, 40(18):7267 – 7279.

Neto, A., Carbonell, J., Marchezan, L., Rodrigues, E. M., Bernardino, M., Basso, F. P., and Medeiros, B. (2020). Systematic mapping study on domain-specific language development tools. *Empir. Softw. Eng.*, 25(5):4205–4249.

Neto, V. V. G., Basso, F. P., dos Santos, R. P., Bakar, N. H., Kassab, M., Werner, C., de Oliveira, T. C., and Nakagawa, E. Y. (2019). Model-driven engineering ecosystems. In *Proceedings of the 7th International Workshop on Software Engineering for Systems-of-Systems and 13th Workshop on Distributed Software Development, Software Ecosystems and Systems-of-Systems, SESoS-WDES 2019, Montreal, QC, Canada, May 28, 2019*, pages 58–61. IEEE / ACM.

Petricic, A., Crnkovic, I., and Zagar, M. (2008). Models transformation between uml and a domain specific language. In *Eight Conference on Software Engineering Research and Practice in Sweden (SERPS 08)*.

Schmidt, D. C. (2006). Model-driven engineering. *IEEE Computer*, 39(2).

Steinberg, D., Budinsky, F., Paternostro, M., and Merks, E. (2008). *EMF: Eclipse Modeling Framework (2nd Edition)*. Addison-Wesley Professional.

Voelter, M. (2009). Best practices for dsls and model-driven development. *Journal of Object Technology*, 8(6):79–102.

XText (2018). Disponível em : <http://www.eclipse.org/xtext/documentation/> acesso em junho de 2018.

Investigando Técnicas de Recomendação de Assets Híbridos de Software: Um Mapeamento Sistemático

[1]Rafael S. Torres, [1]Bruno Marcelo S. Ferreira, [1]Fábio P. Basso,
[1]Elder de Macedo Rodrigues, [1]Maicon Bernardino

[1]Universidade Federal do Pampa (UNIPAMPA), PPGES, Alegrete - RS - Brasil

torresrafa22gmail.com

Abstract. *Software reuse is an effective alternative in creating software with quality, for it allows the creation of new software from existing ones. Thanks to the vast availability of resources available to developers and to software houses, software reuse becomes more promising, yet more challenging. In order to make opportunistic reuse a more effective approach, machine learning techniques have been added to opportunistic reuse processes. This study summarizes a systematic mapping, characterizing how the machine learning area has supported the reuse of hybrid software assets. Our results indicate that there is no single approach that stands out from the others, opening space for comparative studies in the area.*

Resumo. *O reuso de software é uma alternativa efetiva para criar software com qualidade, por permitir criar novos produtos a partir de software já existente. Graças à vasta disponibilidade de recursos disponíveis para desenvolvedores e empresas, o reuso de software se torna mais promissor, e ao mesmo tempo, mais desafiador. De modo a tornar o reuso oportunista uma abordagem mais efetiva,técnicas da área de aprendizado de máquina vem sendo acrescentadas aos processos de reúso. Este estudo sumariza um mapeamento sistemático sobre como a área de aprendizado de máquina tem dado suporte ao reúso de assets híbridos de software. Nossos resultados apontam que não há uma única abordagem que se destaque das demais, abrindo espaço para estudos comparativos.*

1. Introdução

O reúso de software é tido como uma alternativa sólida para lidar com os problemas enfrentados no desenvolvimento de software, como falhas no produto e prazos limitados, já que permite criar novos produtos de software a partir de software já existente [Sametinger 1997], evitando parte do esforço e dos custos envolvidos no processo.

Atingir o objetivo de reusar com sucesso artefatos de software, porém, demanda planejamento e, dependendo da forma que é praticado o reúso, exige certo esforço por parte dos mantenedores e comsumidores dos *assets*. Para que esses esforços sejam minimizados e a eficiência do reúso seja intensificada, é necessário o suporte de técnicas/ferramentas para encontrar os artefatos corretos levando em consideração o problema a ser resolvido pelo desenvolvedor.

Neste artigo, relatamos os resultados obtidos a partir de um mapeamento sistemático que objetiva identificar como a área de aprendizado de máquina vem dando

suporte ao chamado reúso oportunista, caracterizado por buscas e recomendações de *assets* por meio de repositórios de reúso.

O restante deste trabalho está organizado como segue: A Seção 2 introduz os dois principais conceitos para o acompanhamento teórico deste trabalho. A Seção 3 apresenta o protocolo adotado na revisão de literatura e a Seção 4 discute os resultados do estudo executado. A Seção 5 apresenta a análise dos artigos para as questões de pesquisa investigadas. Por fim, a Seção 6 encerra com algumas considerações finais.

2. Embasamento

Data Mining pode ser definido como o processo de analisar conjuntos de dados e inferir padrões destes, através de algoritmos de aprendizado de máquina e métodos estatísticos. De acordo com [Fayyad et al. 1996], já era notória a necessidade de adotar técnicas de aprendizado de máquina a fim de automatizar a análise de dados, que antes era realizada manualmente, devido ao crescimento exponencial de informações disponíveis.

Ainda de acordo com [Fayyad et al. 1996], pode-se categorizar as diferentes abordagens para mineração de dados como: Classificação, Regressão, Aglomeração, Sumarização, Modelo de Dependência, e Detecção de anomalias.

Neste trabalho, buscou-se compreender qual é a relação entre data mining e reúso oportunista, que ocorre por meio de *assets* e repositórios. Um *asset*, no contexto da engenharia de software, pode ser caracterizado como qualquer artefato reutilizável produzido durante o ciclo de vida do desenvolvimento de software. De acordo com a definição dada pela OSLC [ams 2012], *asset* é qualquer coisa que pode ser mantida a fim de gerar valor. No contexto de software, *assets* podem ser código-fonte (funções, classes, ...), modelos, ferramentas e componentes. *Assets* híbridos por sua vez, representam artefatos de diferentes naturezas, um grupo heterogêneo de recursos. Podem se diferenciar por estarem em diferentes linguagens, ferramentas, abordagens, etc.

3. Mapeamento Sistemático

Seguindo as diretrizes de [Petersen et al. 2015], esta seção apresenta o protocolo de mapeamento sistemático utilizado. Para especificar o objetivo dessa pesquisa, foi utilizada a abordagem GQM [Caldiera and Rombach 1994], apresentada na Tabela 1

Tabela 1. Objetivo da pesquisa.

Objetivo	Com o objetivo de identificar e classificar
Questão	técnicas, metodologias, desafios, tendências
Objeto	mineração de dados de repositórios de *assets*, componentes e código
Viewpoint	do ponto de vista do pesquisador

Então, foram concebidas as seguintes questões de pesquisa:

- **Q1**: Quais são as propostas de técnicas/metodologias/guidelines propostas e aplicadas na área de mineração de dados úteis à aquisição/reúso de software/desenvolvimento?
- **Q2**: Que mecanismos de recomendação são utilizados?
- **Q3**: Quais são as tendências das pesquisas na área?

Tabela 2. Bases utilizadas na pesquisa.

Scopus	https://www.scopus.com/home.uri
IEEE	https://ieeexplore.ieee.org
Springer	https://www.springer.com
ACM	https://dl.acm.org/

A pesquisa foi realizada em 4 bases digitais de artigos amplamente conhecidas e utilizadas pela comunidade científica [Chen et al. 2010], apresentadas na Tabela 2.

A *string* de busca foi formulada através da estratégia PICo [Pregunta 2007], como mostra a Tabela 3. Vale ressaltar que foram necessárias algumas modificações específicas para algumas das bases pesquisadas, suprimidas neste artigo por motivos de espaço.

Tabela 3. Aplicação da estratégia PICo na formação da string de busca genérica

P	”data mining”OR ”big data”OR ”data extraction”OR “knowledge management” OR “knowledge discovery”
I	”asset management specification”OR ams OR ”reuse repository”OR ”component repository”OR ”mde repository”OR ”code recommender”OR ”recommender system”OR ”asset recommender”OR ”software asset”OR ”component asset”OR ”reusable asset”OR RAS
Co	“technology transfer” OR “software acquisition” OR “integration” OR “software engineering” OR “software development” OR “software reuse”

A seguir, apresenta-se os critérios de inclusão e de exclusão.

Critérios de Inclusão:

1. Artigos que apresentem técnicas/metodologias/guidelines para mineração de dados.
2. Artigos que relatam estudos de caso sobre técnicas/metodologias/guidelines para mineração de dados.
3. Artigos que tratem de sistemas de recomendação de *assets*/componentes/código em cenários de aquisição de software.

Critérios de Exclusão:

1. Estudos que não envolvam mineração de dados
2. Estudos que não estejam em inglês
3. Estudos duplicados
4. Estudos que não estejam disponíveis para download
5. Estudos que sofreram de retratação ou foram invalidados
6. Estudos que não são primários
7. Estudos que não tratem de transferência de tecnologia do domínio da engenharia de software nem de aquisição ou reúso de software.

4. Execução

Inicialmente, foram obtidos 888 artigos, como mostra a Tabela 4. A busca foi realizada considerando título, resumo e palavras-chave.

Após obter-se os resultados das bases, foram aplicados os critérios de exclusão. A Tabela 5 apresenta a quantidade de artigos desconsiderados.

Além disso, 55 artigos estavam repetidos entre as bases, sendo que destes, 44 estavam entre os descartados, e 11 entre os aceitos, resultando em 42 artigos aprovados após a leitura dos títulos, *abstract* e palavras-chave, e 21 após a leitura do texto completo.

Tabela 4. Artigos retornados

Base	Resultados
IEEE	78
ACM	294
Scopus	201
Springer	315
	888

Tabela 5. Resultado da aplicação dos critérios de exclusão.

Base \ Critério	CE1	CE6	CE7	Total
IEEE	4	4	61	69
ACM	14	7	262	281
Scopus	12	29	147	185
Springer	71	81	145	297
				832

A Figura 1 apresenta o processo descrito.Os artigos selecionados se encontram no Google Drive [1].

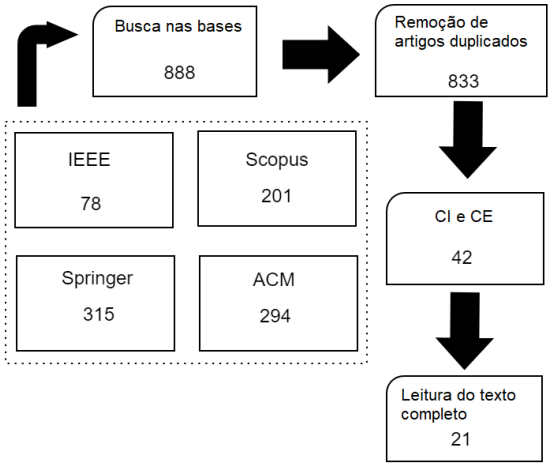


Figura 1. Processo de seleção de artigos.

5. Análise

5.1. Técnicas aplicadas (Q1)

Esta seção responde a **Q1**: Quais são as propostas de técnicas/metodologias/guidelines propostas e aplicadas na área de mineração de dados úteis à aquisição/reúso de software/desenvolvimento?

Pode-se observar dois focos diferentes em relação às propostas dos trabalhos obtidos: 1) utilizar técnicas de clusterização e organização de repositórios de *assets*; e 2) utilizar técnicas de filtragem e exploração de dados à fim de recomendar *assets* que possam ser úteis dependendo do contexto. Em um contexto de reúso oportunista, quando não há o reúso planejado, um repositório organizado daria suporte à busca manual por

[1] `https://drive.google.com/drive/folders/1lTQPbAVb4qpij_MVuULYwZSh5THz-lBz?usp=sharing`

um *asset*. Como apontado por [Wang and Ren 2011], um repositório organizado de maneira lógica é uma maneira efetiva de facilitar a busca de componentes. Um método de filtragem também poderia dar suporte a uma recomendação automatizada por parte do ambiente de desenvolvimento, baseando-se no contexto do desenvolvedor. Além disso, alguns trabalhos também buscam recursos em ambientes web. A relação de fases abordadas na recomendação está representada na Tabela 6.

Tabela 6. Fases identificadas nos trabalhos obtidos.

Artigo	Busca	Organização	Recomendação
[Ye and Lo 2000]		x	
[Nakkrasae and Sophatsathit 2004]		x	
[Li et al. 2004]			x
[Wang et al. 2004]		x	
[McCarey et al. 2005]			x
[Wu et al. 2007]		x	
[Bajracharya et al. 2009]	x	x	
[Martins et al. 2009]			x
[Wang and Ren 2011]		x	
[Dumitru et al. 2011]	x		x
[Heinemann 2012]			x
[Kumar et al. 2011]		x	
[Sayyad et al. 2012]			x
[Vodithala and Pabboju 2015]		x	
[Vescan 2015]			x
[Elkamel et al. 2016]		x	x
[Basciani et al. 2016]		x	
[Bawa and Kaur 2017]		x	
[Kögel 2017]			x
[Le et al. 2018]			x
[Diamantopoulos et al. 2018]	x	x	

Na Tabela 7 é apresentada a relação de quais técnicas foram citadas nos trabalhos e em quais conjuntos de *assets* elas foram aplicadas.

5.2. Mecanismos de recomendação (Q2)

Esta seção responde a **Q2**: Que mecanismos de recomendação são utilizados?

Com relação aos mecanismos de recomendação identificados, não obtivemos nenhum indício de que há uma técnica que se destaque. Ou seja, a área carece de estudos comparativos, principalmente de estudos primários quantitativos. Por exemplo, como pode ser observado na Tabela 8, a única técnica de aprendizado de máquina que foi aplicada em mais de um trabalho foi algoritmo evolutivo. Portanto, visto que tratam-se de estudos exploratórios, trata-se de um indicativo de lacuna de pesquisa, já que existe a necessidade de experimentar diferentes abordagens, bem como realizar estudos comparativos e explicativos.

A maioria dos trabalhos apresentaram alguma forma de validação para suas abordagens, o que abre espaço para algumas observações:

1. **A pesquisa na área carece de estudos experimentais.** A qualidade da abordagem de recomendação pode ser medida através das métricas *Precision* e *Recall*.
   (a) Nos artigos [Heinemann 2012] e [Kögel 2017], os autores usaram *Precision* e *Recall*, com o primeiro estudo mostrando uma precisão de 0.66 e *Recall* de 0.49. O segundo estudo teve como *Precision* 0.43 e *Recall* de 0.82. Sendo *Precision* a relevância dos itens selecionados, e *Recall* a

Tabela 7. Técnicas aplicadas aos tipos de *assets*.

Artigo	Técnica	Tipo de *Assets*
[Ye and Lo 2000]	Self-Organizing Map	Componentes
[Nakkrasae and Sophatsathit 2004]	RPCL(Neural Network)	Componentes
[Li et al. 2004]	Information Entropy Theory	Componentes
[Wang et al. 2004]	Self-Organizing Map	Componentes
[McCarey et al. 2005]	Collaborative filtering, Content-based Filtering	Componentes
[Wu et al. 2007]	Ontologia, text mining	Documentos
[Bajracharya et al. 2009]	Topic Model e Author Topic Model	Source Code
[Martins et al. 2009]	Association Rule	Componentes
[Wang and Ren 2011]	Clustering Index Tree	Componentes
[Dumitru et al. 2011]	Text Mining, K-Nearest-Neighbor, Incremental Diffusive Clustering	Domain-Specific Features
[Heinemann 2012]	Collaborative Filtering e Association Rules	Tool/DSL
[Kumar et al. 2011]	K-Means	Casos de uso em MDL (Rational Rose)
[Sayyad et al. 2012]	Range Ranking Method	Configuração de linha de produtos
[Vodithala and Pabboju 2015]	Clustering	Componente (Funções)
[Vescan 2015]	Algoritmo evolutivo	Componentes
[Elkamel et al. 2016]	CACB	Modelo (Classes UML)
[Basciani et al. 2016]	Hierarchical clustering	Metamodelos
[Bawa and Kaur 2017]	Clustering	Componentes
[Kögel 2017]	Algoritmo evolutivo	Modelos
[Le et al. 2018]	Sequential pattern	Source code
[Diamantopoulos et al. 2018]	K-Means	Source code

Tabela 8. Técnicas usadas para recomendação.

Artigo	Recomendação
[Li et al. 2004]	Information Entropy Theory
[McCarey et al. 2005]	Collaborative Filtering
[Martins et al. 2009]	Association Rule
[Dumitru et al. 2011]	Association Rule e k-Nearest-Neighbor
[Heinemann 2012]	Collaborative Filtering e Association Rule
[Sayyad et al. 2012]	Range Ranking
[Vescan 2015]	Algoritmo Evolutivo
[Elkamel et al. 2016]	CACB
[Kögel 2017]	Algoritmo Evolutivo
[Le et al. 2018]	Sequential Pattern

quantidade de itens relevantes no grupo selecionado, pode-se considerar que [Heinemann 2012] obteve uma melhor seleção de *assets*, enquanto [Kögel 2017] obteve um número maior de *assets* relevantes dentro do grupo de resultado de busca.

(b) O estudo [Martins et al. 2009] apresentou um experimento controlado. O resultado do experimento mostrou uma taxa de sugestão de *assets* de 100%. Mesmo com o resultado positivo, os autores apontaram que mais teste em cenários e dados reais são necessários para corroborar tal resultado.

(c) No artigo de [McCarey et al. 2005], os autores mostraram que a abordagem proposta teve um *Recall* médio de 39.8% e *Precision* média de 22.7%. Os resultados foram considerados promissores comparados a *baseline* prevista.

(d) O número pequeno de contribuições empíricas e quantitativas, é um indicativo de que estudos futuros poderiam explorar as métricas citadas para que seja possível comparar resultados. Sem tais medições, se torna mais

difícil a comparação de abordagens.

2. **Tecnologias antigas podem apresentar bons resultados e lições**. Como exemplo, o estudo [Li et al. 2004], é classificado como recomendação de componente de software e apresentou bons resultados em comparação com repositórios modernos conhecidos que suportam MDE, como MDEForge [Basciani et al. 2016], SEMAT [Jacobson et al. 2012] e SHARE [Gorp and Mazanek 2011]. Os dois últimos introduzidos intencionalmente para fins de discussão e não obtidos através do protocolo executado. O estudo [Li et al. 2004] compara uma abordagem baseada em busca, que é adotado pelos três já mencionados anteriormente, com uma estratégia de seleção de atributo randômico. A estratégia de seleção resultou em um resultado de busca mais eficiente e um conjunto de componentes mais confinado.
3. **Ferramentas de recomendação baseadas em OSLC não foram encontradas.** O OSLC é baseado em *Asset Management Specification* (AMS) e provê meios para mineração de dados e recomendações baseada em *linked data*. Essa limitação na literatura caracteriza oportunidades de pesquisa de recomendação baseada no padrão OSLC.

5.3. Tendências de pesquisa (Q3)

Esta seção responde a **Q3**: Quais são as tendências das pesquisas na área?

Neste sentido, pode-se observar uma distribuição comum do número de estudos por técnicas identificadas, como ilustra a Figura 2. Isto é um indicador de que ainda não foi encontrada/concebida/provada uma abordagem próxima ao considerado ideal para a mineração e recomendação de *assets* de *software*.

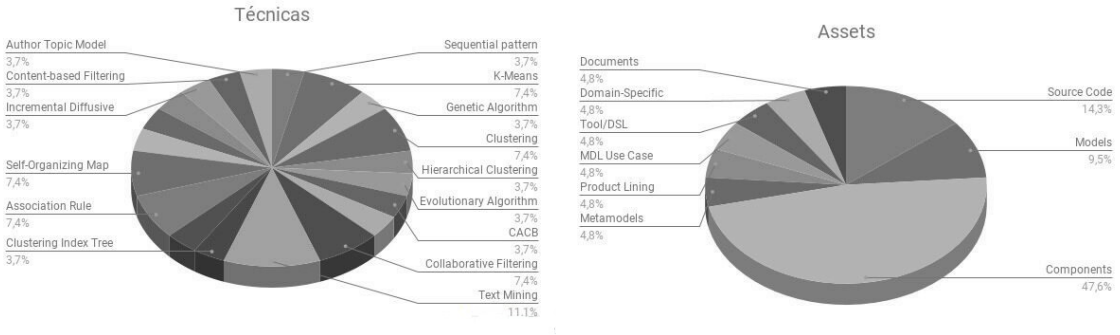


Figura 2. Proporção de técnicas identificadas (a esquerda) e proporção entre os tipos de *assets* identificados (a direita).

Em relação aos tipos de *assets* descritos nos respectivos estudos, encontrou-se que aproximadamente 47% dos artigos abordam *assets* como sendo componentes de software. Demonstrando a diversidade dos *assets*, encontrou-se que aproximadamente 53% dos trabalhos trata de um tipo específico de artefato reutilizável, categorizados em 9 tipos bem definidos. A Figura 2 mostra em detalhes a proporção de *assets* focados nos estudos.

6. Conclusão

O principal objetivo deste trabalho foi relatar os resultados de um mapeamento sistemático de literatura. Com o objetivo de compreender como a área de aprendizado

de máquina da suporte ao reúso oportunista de *assets*, caracterizou-se as técnicas e os processos envolvidos na mineração de *assets* híbridos por meio de questões dirigidas ao suporte de aprendizado de máquina neste contexto. Este trabalho, portanto, apresenta uma análise de como técnicas de recomendação vem contribuindo para a área de reúso oportunista, um estudo singular sobre o estado da arte no tema. Por fim, no âmbito da industria de software e, tendo em vista que ela compartilha do nosso interesse em otimizar os processos de reúso, percebe-se nos estudos analisados duas grandes lacunas para a seleção de opções para recomendação: 1) faltam estudos primários que avaliam as propostas em cenários reais, e 2) a área carece de estudos comparativos de técnicas e também de outros do tipo explicativos que busquem avaliar critérios de adoção na prática.

7. Agradecimentos

Este estudo foi parcialmente financiado através da Pró-Reitoria de Pesquisa (PROPESQ), com bolsa do programa AGP (Apoio a Grupos de Pesquisa), e pela FAPERGS, por meio do projeto ARD N. 19/2551-0001268-3.

Referências

[ams 2012] (2012). Oslc asset management 2.0 specification.

[Bajracharya et al. 2009] Bajracharya, S., Ossher, J., and Lopes, C. (2009). Sourcerer: An internet-scale software repository. In *Proceedings of the 2009 ICSE Workshop on Search-Driven Development-Users, Infrastructure, Tools and Evaluation*, SUITE '09, pages 1–4, Washington, DC, USA. IEEE Computer Society.

[Basciani et al. 2016] Basciani, F., Di Rocco, J., Di Ruscio, D., Iovino, L., and Pierantonio, A. (2016). Automated clustering of metamodel repositories. In *International Conference on Advanced Information Systems Engineering*, pages 342–358. Springer.

[Bawa and Kaur 2017] Bawa, R. K. and Kaur, I. (2017). Classification and clustering for efficient storage and retrieval of component repositories. In *2017 International Conference on Computer Communication and Informatics (ICCCI)*, pages 1–6. IEEE.

[Caldiera and Rombach 1994] Caldiera, V. R. B. G. and Rombach, H. D. (1994). The goal question metric approach. *Encyclopedia of software engineering*, pages 528–532.

[Chen et al. 2010] Chen, L., Ali Babar, M., and Zhang, H. (2010). Towards an evidence-based understanding of electronic data sources.

[Diamantopoulos et al. 2018] Diamantopoulos, T., Karagiannopoulos, G., and Symeonidis, A. (2018). Codecatch: extracting source code snippets from online sources. In *2018 IEEE/ACM 6th International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering (RAISE)*, pages 21–27. IEEE.

[Dumitru et al. 2011] Dumitru, H., Gibiec, M., Hariri, N., Cleland-Huang, J., Mobasher, B., Castro-Herrera, C., and Mirakhorli, M. (2011). On-demand feature recommendations derived from mining public product descriptions. In *Proceedings of the 33rd International Conference on Software Engineering*, ICSE '11, pages 181–190, New York, NY, USA. ACM.

[Elkamel et al. 2016] Elkamel, A., Gzara, M., and Ben-Abdallah, H. (2016). An uml class recommender system for software design. In *2016 IEEE/ACS 13th International Conference of Computer Systems and Applications (AICCSA)*, pages 1–8. IEEE.

[Fayyad et al. 1996] Fayyad, U., Piatetsky-Shapiro, G., and Smyth, P. (1996). From data mining to knowledge discovery in databases. *AI magazine*, 17(3):37–37.

[Gorp and Mazanek 2011] Gorp, P. V. and Mazanek, S. (2011). Share: a web portal for creating and sharing executable research papers. *Procedia Computer Science*, 4:589 – 597. International Conference on Computational Science, ICCS 2011.

[Heinemann 2012] Heinemann, L. (2012). Facilitating reuse in model-based development with context-dependent model element recommendations. In *Proceedings of the Third International Workshop on Recommendation Systems for Software Engineering*, pages 16–20. IEEE Press.

[Jacobson et al. 2012] Jacobson, I., Ng, P.-W., McMahon, P. E., Spence, I., and Lidman, S. (2012). The essence of software engineering: The semat kernel. a thinking framework in the form of an actionable kernel. *ACMQUEUE. Development 9. Networks*, 10(10):1–12.

[Kögel 2017] Kögel, S. (2017). Recommender system for model driven software development. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, ESEC/FSE 2017, pages 1026–1029, New York, NY, USA. ACM.

[Kumar et al. 2011] Kumar, S., Bhatia, R. K., and Kumar, R. (2011). K-means clustering of use-cases using mdl. In *International Conference on Computing and Communication Systems*, pages 57–67. Springer.

[Le et al. 2018] Le, D.-T., Lê, L.-S., Thoai, N., and Nguyen, T.-V. (2018). An old problem with a new therapy: Coupling topic modeling and mining sequential patterns in recommending source code. In *2018 International Conference on Advanced Computing and Applications (ACOMP)*, pages 117–124. IEEE.

[Li et al. 2004] Li, G., Pan, Y., Zhang, L., Xie, B., and Shao, W. (2004). Attribute ranking: An entropy-based approach to accelerating browsing-based component retrieval. In *International Conference on Software Reuse*, pages 232–241. Springer.

[Martins et al. 2009] Martins, A. C., Garcia, V. C., de Almeida, E. S., and d. L. Meira, S. R. (2009). Suggesting software components for reuse in search engines using discovered knowledge techniques. In *2009 35th Euromicro Conference on Software Engineering and Advanced Applications*, pages 412–419.

[McCarey et al. 2005] McCarey, F., Cinnéide, M. Ó., and Kushmerick, N. (2005). Rascal: A recommender agent for agile reuse. *Artificial Intelligence Review*, 24(3):253–276.

[Nakkrasae and Sophatsathit 2004] Nakkrasae, S. and Sophatsathit, P. (2004). An rpcl-based indexing approach for software component classification. *International Journal of Software Engineering and Knowledge Engineering*, 14(05):497–518.

[Petersen et al. 2015] Petersen, K., Vakkalanka, S., and Kuzniarz, L. (2015). Guidelines for conducting systematic mapping studies in software engineering: An update. *Information and Software Technology*, 64:1 – 18.

[Pregunta 2007] Pregunta, L. (2007). A estratégia pico para a construção da pergunta de pesquisa e busca de evidências. *Rev Latino-am Enfermagem*, 15(3).

[Sametinger 1997] Sametinger, J. (1997). *Software engineering with reusable components*. Springer Science & Business Media.

[Sayyad et al. 2012] Sayyad, A. S., Ammar, H., and Menzies, T. (2012). Software feature model recommendations using data mining. In *Proceedings of the Third International Workshop on Recommendation Systems for Software Engineering*, RSSE '12, pages 47–51, Piscataway, NJ, USA. IEEE Press.

[Vescan 2015] Vescan, A. (2015). An evolutionary multiobjective approach for the dynamic multilevel component selection problem. In *International Conference on Service-Oriented Computing*, pages 193–204. Springer.

[Vodithala and Pabboju 2015] Vodithala, S. and Pabboju, S. (2015). A clustering technique based on the specifications of software components. In *2015 International Conference on Advanced Computing and Communication Systems*, pages 1–6. IEEE.

[Wang and Ren 2011] Wang, C. and Ren, Y. (2011). A component clustering index tree based on semantic. In *International Conference on Web Information Systems and Mining*, pages 356–362. Springer.

[Wang et al. 2004] Wang, Z., Liu, D., and Feng, X. (2004). Improved som clustering for software component catalogue. In Yin, F.-L., Wang, J., and Guo, C., editors, *Advances in Neural Networks – ISNN 2004*, pages 846–851, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Wu et al. 2007] Wu, Y., Siy, H., Zand, M., and Winter, V. (2007). Construction of ontology-based software repositories by text mining. In Shi, Y., van Albada, G. D., Dongarra, J., and Sloot, P. M. A., editors, *Computational Science – ICCS 2007*, pages 790–797, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Ye and Lo 2000] Ye, H. and Lo, B. W. (2000). A visualised software library: nested self-organising maps for retrieving and browsing reusable software assets. *Neural Computing & Applications*, 9(4):266–279.

Algoritmo para Cálculo da Similaridade Semântica entre Nomes de Conferências Cadastradas no Curriculum Lattes e na Base Qualis

Rafael Soares[1], **Rafael Z. Frantz**[1]

[1]Universidade Regional do Noroeste do Estado do Rio Grande do Sul (UNIJUÍ)
Departamento de Ciências Exatas e Engenharias
Rua Lulu Ilgenfritz, 480 — 98700-000 — Ijuí/RS -– Brasil

rafael.dcs@unijui.edu.br, rzfrantz@unijui.edu.br

Abstract. *The Lattes Platform is an environment in which researchers register data related to their academic activities, including publications in journals and events. In Brazil, these journals and events may have an evaluation indicator known as Qualis. Automating the extraction of data from journals published in the Lattes curriculum and obtaining the corresponding Qualis is a trivial task thanks to the use of ISSN as an identification key. However, when it comes to papers published in events, the only identification key available is the name of the event, which makes this search difficult. In this paper, we propose an algorithm for an efficient comparison of sentences and apply it in the comparison of event names so that it is possible to search Qualis for publications in events registered in a curriculum with the utmost accuracy.*

Resumo. *A Plataforma Lattes é um ambiente no qual os pesquisadores registram dados relacionados às suas atividades acadêmicas, dentre elas as publicações em periódicos e eventos. No Brasil, esses periódicos e eventos podem ter um indicador de avaliação conhecido como Qualis. A automatização da extração de dados de publicações de periódicos do currículo Lattes e a obtenção do Qualis correspondente é uma tarefa trivial graças ao uso do ISSN como chave de identificação. No entanto, quando se trata de artigos publicados em eventos, a única chave de identificação disponível é o nome do evento, o que dificulta essa busca. Neste artigo, propomos um algoritmo para uma comparação eficiente de sentenças e o aplicamos na comparação de nomes de eventos para que seja possível buscar com o máximo de assertividade o Qualis para as publicações em eventos registradas em um currículo.*

1. Introdução

O Brasil é um dos poucos países no mundo que possui um sistema único e integrado para registro das produções científicas dos pesquisados. Esse sistema, conhecido como Plataforma Lattes, permite que cada pesquisador possa registar seus projetos, orientações, produções, detalhes sobre sua formação acadêmica, colaboração com outros pesquisadores, etc. Pesquisadores que possuem um currículo registrado nessa plataforma podem acrescentar diversos tipos diferentes de produções, tais como artigos em periódicos, artigos em eventos, livros e capítulos de livros. Outro sistema importante para pesquisadores é o sistema Qualis. Desenvolvido inicialmente em 1998 com a intenção de

ser usado como um dos critérios de avaliação de programas de pós-graduação das universidades brasileiras, grande parte das produções científicas são classificadas de acordo com esse sistema. Além de possuir uma base de dados com os mais importantes periódicos e suas classificações, o Qualis também classifica eventos de algumas áreas como a da Ciência Computação. Essa base de periódicos, eventos e classificações é disponibilizada online pela Capes através do Portal Sucupira [Sucupira 2020].

Uma atividade frequente nas instituições de ensino pelos setores administrativos é a construção de uma lista de publicações para um determinado pesquisador ou grupo de pesquisadores, sendo importante constar nesta lista para cada artigo, alguns indicadores de avaliação do foro onde os trabalhos foram publicados, tais como o Qualis, JCR, h-index, Quartil, dentre outros. Embora a extração das publicações dos currículos seja uma tarefa geralmente automatizada, o processo de enriquecimento de informações com os indicadores para cada artigo é feito de forma manual. Para artigos publicados em periódicos a busca dos valores para cada indicador de avaliação pode ser feita por meio do ISSN do periódico, já que as bases que gestionam esses indicadores costumam usá-lo como identificador único. Assim, um sistema automatizado pode encontrar a classificação de um periódico na base Qualis usando o ISSN do periódico como chave da busca. No entanto, essa tarefa não é trivial quando se trata de automatizar a busca dos indicadores para artigos publicados em eventos, uma vez que neste caso o nome do evento passa a ser o identificador único adotado pelas bases. Isso representa um desafio pois o nome de um evento cadastrado pelo pesquisador no seu currículo Lattes pode estar incompleto ou ligeiramente distinto de seu correspondente na base Qualis.

Uma sentença é tecnicamente representada por uma *string*, que, na maior parte das linguagens de programação, nada mais é do que um conjunto de caracteres de 1 byte que formam um texto a ser computado e pode ser utilizado para representar o nome de um evento. Uma comparação simples entre essas *strings*, ainda que utilizando métodos nativos da linguagem de programação, depende que ambas as *strings* comparadas possuam os mesmos caracteres na mesma ordem para que sejam consideradas idênticas. A similaridade semântica entre sentenças é um campo de pesquisa amplamente explorado na área de processamento de linguagem natural e com as mais diversas aplicações. [Ho et al. 2010] argumentam que muitos trabalhos tendem a calcular a similaridade entre sentenças a partir de uma análise individual das palavras que integram a sentença e seus significados próximos, mas que essa abordagem é falha pois nem sempre o significado próximo reflete o real significado da palavra em questão na sentença. No trabalho desses autores, eles propõe comparar duas sentenças a partir do significado real de cada palavra que compõe a sentença. Na proposta os autores mostram e comprovam estatisticamente que sua abordagem é mais eficaz. [Farouk 2018] propõe uma abordagem hibrida que combina o uso de um banco de dados léxico para o idioma inglês com a representação vetorial de palavras. Na sua abordagem o autor ainda propõe considerar a posição das palavras na sentença para resolver conflitos de significado. [Achananuparp et al. 2008] revisaram a literatura e analisaram um conjunto amplo de 14 métricas para calcular a similaridade entre duas sentenças. O trabalho destes autores comparou o uso destas métricas e, embora tenha sido realizado há alguns anos, ainda pode se constituir em um ponto de partida interessante para o estudo e desenvolvimento de novas métricas.

Neste artigo apresentamos um algoritmo que realiza uma comparação morfológica entre sentenças que, a partir da sequência em que as palavras estão dispostas na sentença e da sequência em que as letras estão dispostas nas palavras, utiliza um sistema de pontuação para quantificar a semelhança entre duas sentenças. Para avaliar nosso algoritmo o implementamos em um sistema real que extrai publicações de eventos de um currículo Lattes em formato XML e enriquece os dados de cada artigo com seu Qualis. Avaliamos a assertividade de nosso algoritmo a partir da comparação dos resultados com outra implementação que faz uma comparação de *strings* usando métodos nativos da linguagem de programação. Nosso algoritmo apresenta uma boa assertividade já que desconsidera as pequenas diferenças entre dois nomes do mesmo evento, porém não ao ponto de considerar dois eventos diferentes o mesmo. Trata-se de uma proposta preliminar, que acreditamos mesmo assim, possa ser utilizada no sistema real onde foi implantado.

O restante deste trabalho está organizado da seguinte forma: a Seção 2 detalhamos o algoritmo que desenvolvemos; na Seção 3 introduzimos o sistema real no qual implementamos o algoritmo para poder avaliá-lo e compará-lo com uma implementação simples de comparação de *strings*; na Seção 4 apresentamos nossas conclusões e discutimos os trabalhos futuros que pretendemos desenvolver para evoluir esta proposta.

2. Algoritmo Proposto

Nesta seção descreve-se o algoritmo desenvolvido na pesquisa. Uma implementação foi desenvolvida em PHP e pode ser acessada online [1]. As funções externas nativas da linguagem foram utilizadas na construção do algoritmo e estão descritas no pseudocódigo do Algoritmo 1 com assinaturas diferentes do encontrado na linguagem. Esta função recebe três entradas, o nome do evento que se encontra no curriculo Lattes do pesquisador, o nome do evento da Base Qualis e uma pontuação mínima para ajustar o limite entre o que será considerado mesmo evento ou evento diferente. O algoritmo retorna um valor booleano que representa a correspondência da comparação. Será verdadeiro se a pontuação for maior ou igual a pontuação mínima definida, e falso caso a pontuação seja menor. As funções externas utilizadas na construção deste algoritmo são: `explodir()`, que divide uma *string* em um ponto especificado pelo programador e retorna um vetor com as sub *strings*, `contar()`, que devolve um inteiro com a quantidade de itens de um vetor e `ord()`, que devolve o valor ASCII correspondente ao caractere de entrada ou do primeiro caractere de uma *string* de entrada. Nas linhas 3 e 4 a função `explodir()` é usada para separar os títulos nos caracteres de espaçamento, criando um vetor onde cada item é uma palavra.

A partir da linha 7 o algoritmo pode ser visto como três blocos distintos. O primeiro, que vai da linha 7 à linha 16 analisa a ordem das palavras em cada um dos nomes observando apenas a primeira letra de cada palavra. Em um laço de repetição que passa por todas os itens de `vetPalavrasLattes` é feito uma comparação com os itens de `vetPalabrasBase` usando `ord()`, de forma a definir se a primeira letra de cada palavra aparece na mesma ordem em ambos os vetores. Esse bloco tem o propósito de descartar rapidamente os nomes que possuem a menor probabilidade de coincidir, já que a base Qualis conta com 1179 títulos diferentes e uma comparação mais detalhada de cada nome pode tomar muito tempo e recurso de processamento.

[1] www.gca.unijui.edu.br/project/conference.php

Algoritmo 1: COMPARADOR DE TÍTULOS

Entrada: $conferenciaLattes, conferenciaBase, pontuacaoMinima$
Saída: Valor booleano que indica correspondência

```
1  início
2      pontos := 0, inicio := -1, i := 0
3      vetPalavrasLattes := explodir(' ', conferenciaLattes)
4      vetPalavrasBase := explodir(' ', conferenciaBase)
5      para j := 0 até contar(vetPalavrasLattes) faça
6          se ord(vetPalavrasLattes[j]) == ord(vetPalavrasBase[i]) então
7              i++
8              inicio == -1 ? inicio = j : inicio = inicio
9          senão
10             se i > 0 e i < contar(vetPalavrasBase) então
11                 i := 0
12             fim
13         fim
14     fim
15     se i == contar(vetPalavrasBase) então
16         para i := 0 até contar(vetPalavrasBase) faça
17             vetLetrasBase := explodir(vetPalavrasBase[i])
18             vetLetrasLattes := explodir(vetLetrasLattes[inicio])
19             contador := 0
20             para j := 0 até contar(vetLetrasBase) faça
21                 se ord(vetLetrasBase[j]) == ord(vetLetrasLattes[j])
                    então
22                     contador++
23                 fim
24             fim
25             pontos += contador
26             inicio++
27         fim
28     fim
29     pontos := pontos/tamanhoString(conferenciaBase)
30     se pontos >= pontuacaoMinima então
31         retorna TRUE
32     senão
33         retorna FALSE
34     fim
35 fim
```

A variável `i` é o contador que incrementará dependendo de quantas palavras estiverem em ordem. Se `i` tiver o mesmo valor que a quantidade de itens dentro de `vetPalavrasBase`, o segundo bloco, que vai da linha 17 à linha 30, é acionado. Caso contrário, o algoritmo irá considerar nomes diferentes e retornará falso. Dentro do segundo bloco cada palavra em cada vetor é comparada letra por letra com a sua possível correspondente seguindo a ordem definida no primeiro bloco pela variável `inicio`. Um

ponto será somado para cada letra que coincidir em ambos os nomes e esse somatório será levado ao terceiro bloco assim que finalizada esta etapa. No terceiro bloco um índice é calculado dividindo os pontos somados, ou a quantidade de letras que coincidem, pela quantidade de letras no total em `conferenciaBase`. Esse índice será comparado com a pontuação mínima definida em `pontuacaoMinima` pelo programador e retornará o valor booleano correspondente.

3. Estudo de Caso

Para demonstrar o algoritmo proposto foi utilizado o currículo Lattes de dez pesquisadores [2]. Os currículos foram lidos por um sistema automatizado criado previamente, que extraiu todos os artigos publicados em eventos por esses pesquisadores. O título de cada evento extraído foi comparado com a base Qualis utilizando tanto um método nativo de comparação quanto o algoritmo proposto. Uma pesquisa manual de cada evento do Lattes dos pesquisadores foi feita para avaliar a eficiência do algoritmo. No total 869 nomes de eventos foram comparados à base Qualis, sendo que 335 possuem uma classificação Qualis. A Tabela 1 demonstra os dados obtidos.

Tabela 1. Quantidade de identificações de nomes de eventos com Qualis utilizando cada método.

Currículos	Publicações em Eventos	Publicações com Qualis	Técnicas de Comparação	
			Método Nativo	Algoritmo Proposto
Currículo 1	36	6	0	4
Currículo 2	50	7	0	5
Currículo 3	62	14	1	8
Currículo 4	142	23	2	11
Currículo 5	187	102	5	45
Currículo 6	91	40	4	28
Currículo 7	155	83	2	51
Currículo 8	70	20	2	15
Currículo 9	47	30	2	15
Currículo 10	29	10	0	7
Total	869	335	18	189

No currículo 1, de 6 nomes de eventos que possuem Qualis, o algoritmo proposto identificou 4 e o método nativo, nenhum. No currículo 2, o algoritmo identificou 5 de 7 eventos com Qualis, comparado ao método nativo que novamente não encontrou nenhum. No currículo 3, entre 14 eventos com Qualis, o algoritmo identificou 8 e o método nativo, 1. No currículo 4, entre 23 eventos registrados que possuiam Qualis, 11 foram identificados pelo algoritmo proposto em contrapartida a 2 identificados pelo método nativo de comparação. No currículo 5, 187 nomes de eventos foram analisados no total, dentre eles 107 possuíam Qualis, de forma que o algoritmo proposto identificou 45 e o método nativo, 5 eventos. No currículo 6, entre 40 eventos encontrados na base Qualis, o algoritmo proposto identificou 28 e o método nativo identificou 4. No currículo 7 foram analisados 155 nomes de eventos sendo que destes, 83 possuem Qualis, o algoritmo proposto identificou 51 e o método nativo identificou apenas 2. De 20 nomes de eventos que possuem Qualis

[2] Currículos disponíveis em: www.gca.unijui.edu.br/publication/data/curriculos.zip

no currículo 8, o algoritmo proposto encontrou 15 e o método nativo, 2. O currículo 9, que possui 30 nomes de eventos encontrados na base Qualis, teve 15 nomes identificados pelo algoritmo proposto e apenas 2 pelo método nativo. Por fim, no currículo 10 o algoritmo proposto encontrou 7 dos 10 eventos que possuem Qualis e o método nativo não encontrou nenhum.

A pontuação mínima utilizada no algoritmo foi 0.8, ou seja, seriam considerados mesmo evento se 80% das letras em um nome de evento na base Qualis fossem encontradas em um nome de evento registrado em um currículo. Dentre os 335 eventos que possuem Qualis encontrados entre os currículos Lattes desses pesquisadores, considerando eventos duplicados entre currículos, um total de 189 nomes foram identificados pelo algoritmo proposto, representando cerca de 56% de acerto total na comparação de nomes com uma média de acerto de 61,32%. O método nativo de comparação oferecido pela linguagem de programação utilizada identificou um total de 18 nomes de eventos, cerca de 5% de acerto total, tendo uma média de acerto de 4,98%.

4. Conclusão

Neste artigo foi apresentado uma proposta preliminar de algoritmo para comparação de nomes de eventos com o intuito de classificar artigos registrados no currículo Lattes de pesquisadores utilizando a base Qualis e outros indicadores de avaliação de forma automatizada. Este algoritmo foi desenvolvido de forma a fazer uma análise morfológica das palavras, ou seja, comparar a estrutura de cada palavra e sua ordem na sentença com seu possível correspondente. Foram utilizados currículos de pesquisadores de diferentes instituições que possuem grande número de trabalhos publicados em eventos voltados à computação pois é visível que nesta área a base Qualis de eventos é mais completa. Estes currículos foram analisados por um sistema automatizado de extração de dados, porém cada nome de evento foi também comparado manualmente com a base Qualis para que fosse possível avaliar a eficiência do algoritmo proposto.

A partir dos resultados obtidos no estudo de caso, é possível observar a grande diferença de eficiência de um método nativo de comparação e o algoritmo desenvolvido nesta pesquisa, considerando que este algoritmo é uma proposta preliminar de solução. Porém, apesar de se mostrar uma solução mais eficiente, a analise dos resultados obtidos mostrou quais são os problemas do algoritmo e as possíveis melhorias que podem ser feitas. Uma das dificuldades apresentadas pelo algoritmo é identificar a sigla dos eventos, sendo que muitas vezes pesquisadores registram apenas isso como nome do evento participado. Outro problema está em um pesquisador registrar o nome de um evento em um idioma diferente do registrado na base Qualis, característica que foi percebida na coleta de dados graças ao registro conjunto da sigla.

Como trabalho futuro deve-se resolver os problemas encontrados na proposta atual do algoritmo, onde um identificador de siglas pode ser adicionado junto com alguns ajustes técnicos necessários. Novas técnicas de comparação que avaliam a similaridade semântica de sentenças podem ser incrementadas ao algoritmo no futuro de forma a aumentar seu índice de acerto e disponibilizar novas métricas para que haja uma avaliação mais precisa de seu funcionamento. Eventualmente pode-se até mesmo desenvolver um método baseado em inteligência artificial, sendo que a utilização de cada nova versão do algoritmo será comparada com a de suas versões anteriores, servindo como guia para

alcançar a maior eficiência.

Agradecimentos

Este trabalho tem o apoio da Fundação de Amparo à Pesquisa do Estado do Rio Grande do Sul (FAPERGS) no contexto do Programa Pesquisador Gaúcho, com termo de outorga número 17/2551-0001206-2 e do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) pela bolsa de Iniciação Científica.

Referências

Achananuparp, P., Hu, X., and Shen, X. (2008). The evaluation of sentence similarity measures. In *Data Warehousing and Knowledge Discovery*, pages 305–316.

Farouk, M. (2018). Sentence semantic similarity based on word embedding and wordnet. In *International Conference on Computer Engineering and Systems (ICCES)*, pages 33–37.

Ho, C., Murad, M. A. A., Kadir, R. A., and Doraisamy, S. C. (2010). Word sense disambiguation-based sentence similarity. In *International Conference on Computational Linguistics: Posters (COLING)*, pages 418—-426.

Sucupira (2020). Portal sucupira. https://sucupira.capes.gov.br. Último acesso em 05/03/2020.

Modelagem da Solução de integração Café por meio de Rede de Petri Coloridas

Luis Gustavo Tabile, Fabricia Roos-Frantz

[1]Universidade Regional do Noroeste do Estado do Rio Grande do Sul (UNIJUÍ)
Departamento de Ciências Exatas e Engenharias
Rua Lulu Ilgenfritz, 480 — 98700-000 — Ijuí/RS -– Brasil

luis.tabile@sou.unijui.edu.br,frfrantz@unijui.edu.br

Abstract. *Nowadays it is very common for companies to use a software ecosystem, these ecosystems can be made up of several applications, and it is increasingly necessary for these applications to be integrated with the intention that data and functionality can be shared between them, therefore, it is advisable to integrate applications because it brings a better cost benefit to the company that opts for this alternative. To this purpose, there are platforms that provide support for modeling and implementing integration solutions, one example of which is Guaraná. Before an integration solution can be implemented, it should be checked to avoid possible errors in the modeling, and thus prevent errors from being corrected only in late stages of its development. The purpose of this article is to model the integration solution known as Café using the Petri Nets formalism, thus generating a model that can serve as a basis for the verification and validation of this solution. We present a translation of the Café solution model, developed with the Guaraná DSL platform modeling language, to a model in Colored Petri Nets. It is hoped that this modeling can contribute to the process of verification and validation of integration solutions.*

Resumo. *Hoje em dia é muito comum o uso de um ecossistema de software por empresas, esses ecossistemas podem ser compostos por diversas aplicações, e cotidianamente se vê mais necessário que essas aplicações sejam integradas com a intenção de que possam ser compartilhado dados e funcionalidades entre elas, sendo assim é aconselhável a integração de aplicações por trazer um melhor custo beneficio a empresa que opta por essa alternativa. Para esse fim, existem plataformas que fornecem suporte a modelagem e implementação de soluções de integração, um exemplo dessas plataformas é o Guaraná. Antes que uma solução de integração possa ser implementada, ela deveria ser verificada para evitar possíveis erros na modelagem, e assim evitar que erros sejam corrigidos apenas em etapas tardias do seu desenvolvimento. O objetivo deste artigo é modelar a solução de integração conhecida como Café utilizando o formalismo de Redes de Petri, gerando assim um modelo que pode servir como base para a verificação e validação desta solução. Apresentamos uma tradução do modelo da solução Café, desenvolvido com a linguagem de modelagem da plataforma Guaraná DSL, para um modelo em Redes de Petri Coloridas. Espera-se que esta modelagem possa contribuir com o processo de verificação e validação de soluções de integração.*

1. Introdução

Com o aumento do uso da tecnologia ao longo dos anos, aplicações de software foram desenvolvidas ou adquiridas de terceiros para dar suporte aos processos de negócios das empresas, resultando em um diversificado ecossistema de software. Neste contexto, mudanças nos processos de negócio existentes ou necessidade de inclusão de novos processos ocorrem constantemente, e o ideal é fazer uso das aplicações já existente no ecossistema de software para reutilizar dados e funcionalidades e dessa forma não ter que desenvolver novas aplicações a partir do zero. Em meio a essa demanda, as empresas buscavam soluções mais eficientes e com menor custo, o que deu origem ao termo integração de aplicações empresariais, do inglês *Enterprise Application Integration (EAI)* [Linthicum 2000] . A integração de aplicações consiste em integrar aplicações para que elas possam trocar dados e funcionalidades necessários para novos processos de negócio. Essas aplicações foram criadas geralmente sem a intenção de ser integradas, o que constitui um ecossistema de software heterogêneo e difícil de integrar, pois umas aplicações precisam ser adaptadas aos formatos e dados das outras aplicações, o que gera um custo adicional, muitas vezes alto.

A área de EAI proporciona um conjunto de metodologias, técnicas e ferramentas para a integração de aplicações de um ecossistema de software. Atualmente, existem diversas plataformas que dão suporte ao projeto, implementação, execução e monitoramento de soluções de integração [He and Xu 2014]. Geralmente, essas soluções possuem arquitetura *pipes-and-filters* e são baseadas nos padrões de integração documentados em [Hohpe and Woolf 2004]. Dentre as diversas plataformas de integração, estão: a OIC (ORACLE Integration Cloud) [Metin 2018], a plataforma Betalike-EAI [1], e a Guaraná [Frantz et al. 2016], a qual oferece uma Domain Specific Language (DSL) para projetar soluções de integração de aplicações empresarias e um motor de execução.

A análise de uma solução de integração antes de sua implementação é um fator importante que deve ser considerado, pois melhora a qualidade da solução e reduz gastos com possíveis correções de erros, ainda na fase de projeto. Para essa análise deve ser utilizada uma ferramenta que seja capaz de modelar e representar formalmente os modelos conceituais desenvolvidos com a Guaraná DSL. Um formalismo bastante utilizado para a análise e simulação de sistemas discretos que envolvem processos estocásticos, que é o caso de um processo de integração, é o formalismo das Redes de Petri. Introduzidas por Carl Adam Petri [Petri 1966], elas oferecem uma representação gráfica, baseada em grafos, que facilita o processo de modelagem, além de possuir um formalismo e um conjunto de ferramentas que dão suporte à verificação, validação e simulação de sistemas [Molloy 1982, Jensen et al. 2007, Jensen and Kristensen 2009, van der Aalst and Stahl 2011].

O objetivo do trabalho é desenvolver um modelo em Rede de Petri Colorida para a solução Café, modelada em Guaraná DSL; o problema de integração Café é um exemplo clássico da literatura na área de EAI [Hohpe 2005]. Para a modelagem foi utilizada a ferramenta CPN, que pode ser acessada em: [2], a qual permite a construção de um modelo de simulação para verificação e validação de uma solução de integração, na fase de projeto. A tradução do modelo Café para Redes de Petri co-

[1] http://www.betalike.com.br

[2] http://cpntools.org/

loridas foi baseada nos trabalhos de outros autores que já fizeram este tipo de modelagem [Fahland and Gierds 2013, Roos-Frantz et al. 2015].

O restante deste artigo está organizado da seguinte maneira: Na Seção 2, o modelo da solução Café é apresentado e o fluxo de execução da solução modelada na linguagem Guaraná DSL é explicado. Na Seção 3, é apresentado o modelo da solução com Redes de Petri e uma explicação sobre a linguagem e a plataforma que foi utilizada para modelar essa solução.

2. Solução de Integração Café

O modelo conceitual da solução de integração Café em Guaraná DSL é composto basicamente por portas (de entrada, de saída ou de solicitação/resposta), tarefas, *slots* e aplicações (veja Figura 1). O modelo representa como os pedidos são requisitados e processados em uma cafeteria. A comunicação entre os pedidos que chegam na solução e os pedidos que saem da solução é realizada por meio de portas que leem e portas que mandam mensagens. Depois que o caixa enviar o pedido à solução, ele não tem mais acesso a ele; o pedido vai para uma fila de mensagens, percorre o fluxo de execução de tarefas e é encaminhado para os baristas que preparam os pedidos e os devolvem a solução, por meio das portas de comunicação. Dessa forma, enquanto a mensagem do pedido flui pela solução, outros pedidos podem ser enviados, sem que haja a necessidade de espera.

O fluxo de execução da solução começa na porta de entrada P1, que tem a função de leitura dos pedidos, e após a leitura desses pedidos eles são encaminhados para a tarefa T1, que vai separar esses pedidos em diversas mensagens para cada tipo de bebida. Logo, a mensagem é encaminhada para uma tarefa T2, que tem função de despachar os pedidos, os quais podem ir para quaisquer um dos dois baristas: para o barista de bebidas quentes, caso seja uma bebida quente ou para o barista de bebidas frias, caso seja uma bebida fria. Em seguida, a tarefa T3 replica a mensagem para o barista, para que a aplicação solicite o preparo da bebida por meio das portas P2 e P3 e outra mensagem fica aguardando a resposta do barista sobre as informações do preparo da bebida. A tarefa T5 correlaciona a mensagem com as informações do pedido vindo do barista, com a mensagem que estava aguardando a reposta. Quando todas as bebidas de um pedido tiverem sido preparadas, elas são enviadas para o garçom por meio da porta P6, que realizará a entrega do pedido. O mesmo corre no fluxo de execução que comunica com as portas P4 e P5.

3. Modelo da solução em Redes de Petri

As Redes de Petri possuem dois elementos básicos: as transições (retângulos), que são responsáveis por executar as tarefas e os lugares (círculos), que são variáveis de estado [Desel and Reisig 2015]. As Redes de Petri são grafos bipartidos, nos quais os lugares sempre são direcionados para transições e transições para lugares. Lugares e transições são conectados por meio de arcos que são representados por flechas, as quais indicam o sentido e fluxo do *token*. Com o CPN Tools é possível modelar Redes de Petri Coloridas [Jensen et al. 2007], nas quais os *tokens* podem ser individualizados por meio de tipos diferentes, denominados cores.

No modelo da solução em Redes de Petri, ilustrado na Figura 2, foram utilizados 5 lugares que correspondem às portas de entrada P1, P3 e P5 e às portas de saída P2, P4 e P6. As transições ”T”representam as tarefas a serem processadas, por exemplo, a tarefa

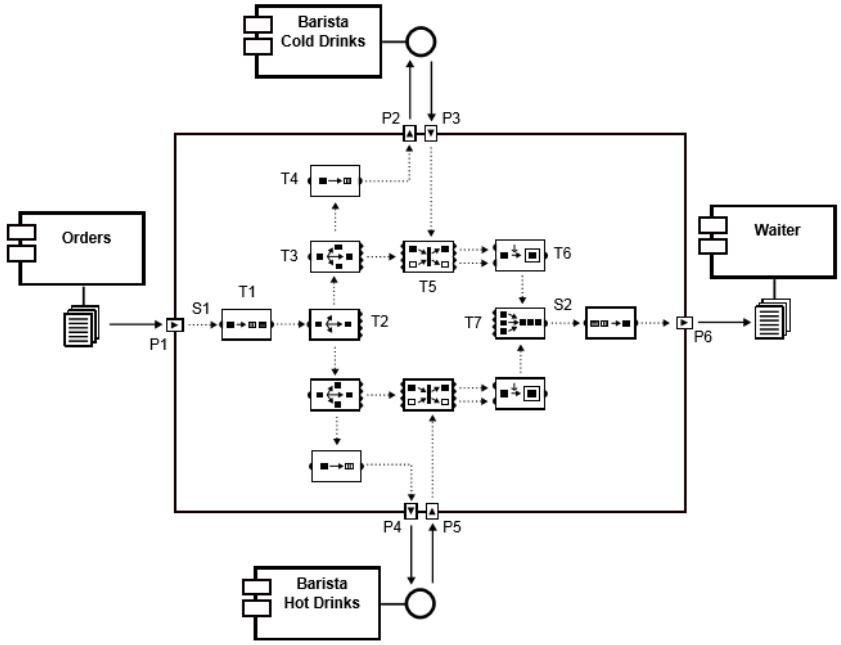


Figura 1. Solução de integração Café [Frantz 2012]

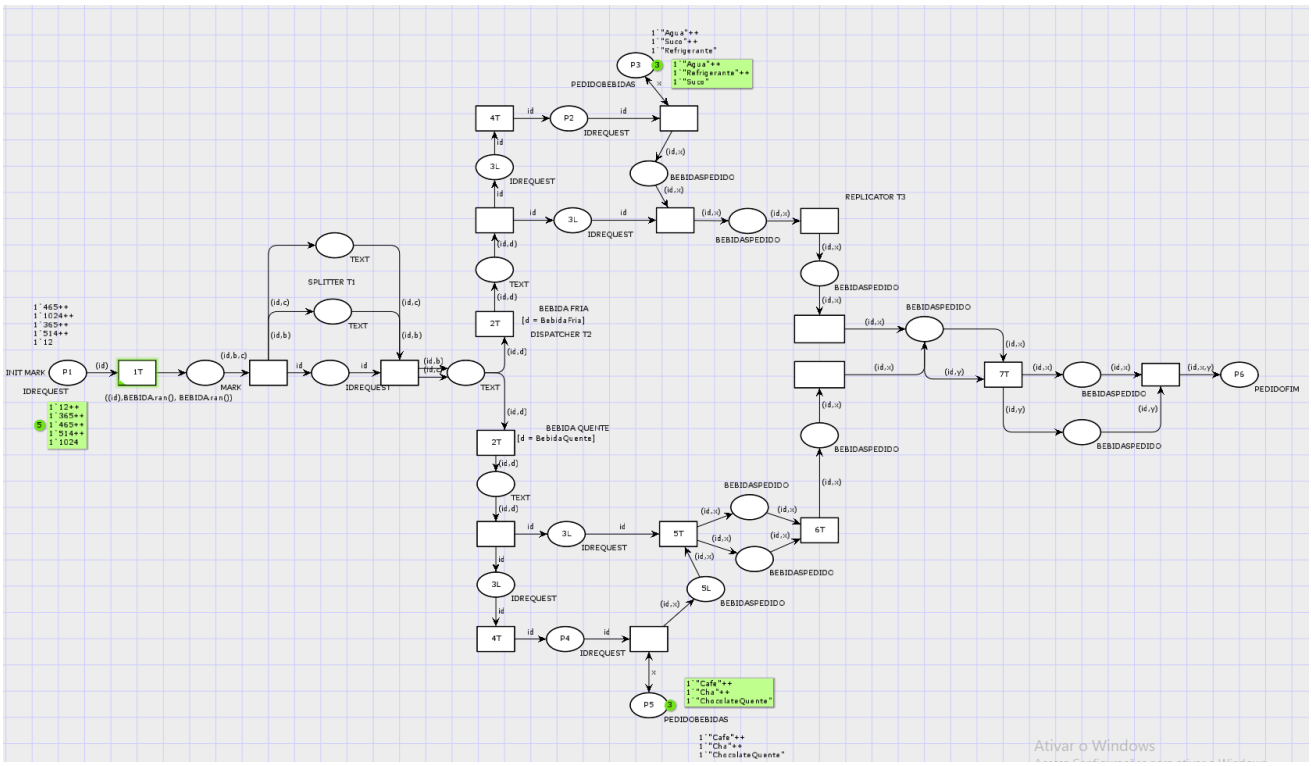


Figura 2. Representação da solução em Redes de Petri usando a ferramenta CPN Tools

T1 é responsável por dividir as mensagens em outras mensagens, que são direcionadas para as bebidas quentes e para as bebidas frias. A transição T1 representa a primeira tarefa, pronta para a execução, ou seja, ela está habilitada na marcação inicial porque possui *tokens* no lugar conectado ao seu arco de entrada. Quando T1 é disparada, ela

consome esses *tokens* e produz outros *tokens* no lugar conectado ao seu arco de saída. Os lugares são representados pela letra ”P”que representam variáveis de estado.

A solução começa na porta P1, que já contém 5 *tokens* prontos para a execução, nessa solução o cliente tem que fazer dois pedidos, não importa o tipo dos pedidos, e sim a quantidade deles, os pedidos podem ser dois frios, dois quentes ou um quente e um frio, cada *token* representa um número do pedido de um cliente. Esse pedido é encaminhado até a tarefa T1, a qual seleciona o tipo da bebida e em seguida é separa em dois *tokens*, que contêm o mesmo número do pedido, para que possam ser agrupados novamente no final da solução. Cada tarefa 2T possui uma condição, para o pedido ir para o barista de bebidas frias o *token* tem que ser do tipo bebida fria, e para ir para o barista de bebidas quentes o *token* tem que ser do tipo bebida quente. Após ser orientado para o barista correto, a P3 é responsável pelo envio da bebida, é interessante observar que nessa P3 possui dois arcos direcionais, esses dois arcos funcionam de modo para que quando um pedido é feito o item que foi enviado para o cliente seja reabastecido.

Após a execução da aplicação, a porta P6 possui todos os pedidos que foram feitos (veja Figura 3). Ao total foram realizados 95 passos durante a execução. Pretende-se com essa modelagem criar uma solução com redes hierárquicas para ver o tempo que seria necessário para realizar esses 5 pedidos com 10 itens no total.

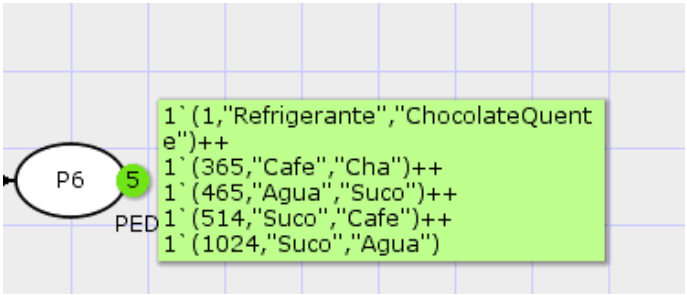


Figura 3. Resultado final após a execução da Solução de Integração Café

4. Processos de Verificação do Modelo de Simulação

Para a verificação do modelo de simulação apresentado nesse artigo foi utilizada uma técnica de verificação proposta por Sargent [Sargent 2005], a qual descreve o fluxo de mensagens para verificar se o modelo está implementado de forma correta. Esse modelo foi dividido em 3 blocos, separados por cores. O lugar P1 representa o inicio, que após 1000 passos realizados resultou em um total de 121 *tokens* disparados, a transição T5 e T10 possuem dois contadores de disparos, os demais numerais representam o número de mensagens que ficaram acumuladas ao final da execução.

A Figura 4, representa o diagrama de aplicação da técnica de Sargent, separada em 3 blocos de cores. As tarefas correlacionadoras (T8a, T8b) são alimentadas pelos *slots* S6a, S8a, S6b e S8b, em ambos lados, tanto na parte inferior quanto na superior é possível ver sinais de acúmulo de mensagens superior a média nesses *slots*, o que pode causar gargalos, pois as tarefas S8a e S8b dependem da execução da transição T9a e T9b, que dependem da execução das portas P2,P3,P4 e P5. O *slot* S4 acumulou cerca de 2 vezes mais mensagens que os demais, assim sendo, o *slot* que possui o maior acúmulo de mensagens, esse acúmulo acontece devido as condições que as tarefas T5a e T5b possuem,

sendo a tarefa T5a possuindo uma condição caso a bebida seja Fria e T5b caso a bebida seja Quente.

A Figura 5 representa a aplicação da técnica de Sargent, é separada em 3 colunas, a primeira coluna é o bloco a qual a técnica foi realizada, a segunda coluna calcula o valor inicial de disparos da transição e compara realizando a soma do bloco, caso a soma seja a mesma do que o número de disparo da transição a coluna três dará um OK mostrando que naquele setor o modelo foi implementado de maneira correta. Após a analise da tabela chegamos ao resultado que todos os blocos atendem o principio da técnica de Sargent, pois o número de disparos da transição de entrada é igual a soma da fila com as saídas.

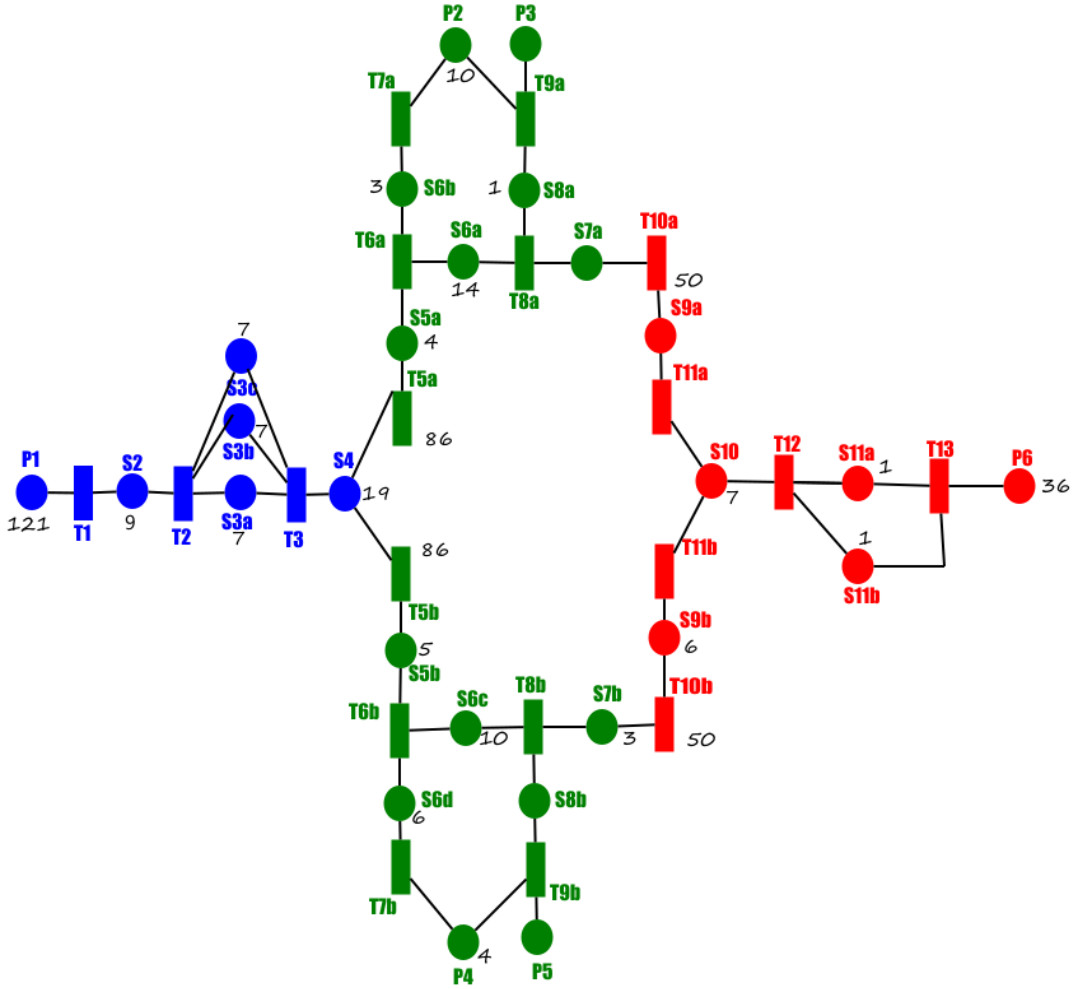


Figura 4. Diagrama de Aplicação da Técnica de Sargent

5. Trabalhos Relacionados

Na literatura pouco se é encontrado sobre uma solução de integração modelada em algum ambiente de simulação. Porém existem muitos trabalhos que tem alguma relação com Redes de Petri, pois é uma rede que abrange muitas áreas, como por exemplo, na modelagem e análise de processos de negócios. Durante a elaboração deste trabalho foram estudadas diversas tecnologias com objetivo parecidos ao que foi apresentado nesse artigo.

Bloco	Contagem	Resultado
Azul	D(P1) = D(T1) + F(S2) + D(T2) + F(S3a) + D(T3) + F(S4) + D(T5)	OK
Azul	121 = 9 + 7 + 19 + 86 = 121	OK
Verde	D(T5) = F(S5a,b) + D(T6a,b) + F(S6a,b) + D(T7a,b) + P2 +P4 + D(T8a,b) + F(S7a,b) + F(S8a,b) + D(T10)	OK
Verde	86 = 4 + 14 + 5 + 10 + 3 + 50 = 86	OK
Vermelho	D(T10) = F(S9a,b) + D(T11a,bd) + F(S10) + D(T12) + F(S11a,b) + T13 + P6	OK
Vermelho	50 = 6 + 7 + 1 + 36 = 50	OK

Figura 5. Aplicação da Técnica de Verificação .

O trabalho apresentado por Carl Adam Petri [Petri 1966] dá início a um modelo matemático, o qual chamamos hoje em dia de Rede de Petri, que é o alicerce dessa pesquisa. Hohpe [Hohpe and Woolf 2004] cita uma coleção de padrões de integração, 65 no total, usados para implementar um sistema de mensagens. A tradução feita por Fahland [Fahland and Gierds 2013] dos padrões de integração básicos descritos por [Hohpe and Woolf 2004] permitiu a criação da Modelagem da Solução de Integração Café por meio de Redes de Petri Coloridas apresentada nesse artigo.

O trabalho realizado por [Roos-Frantz et al. 2015] é o primeiro a propor o uso de Redes de Petri Estocásticas para analisar o comportamento de uma solução de integração feita com o Guaraná. Neste trabalho é proposto uma simulação usando Redes de Petri, para poder analisar soluções baseadas no Guaraná, o trabalho mostra que as Redes de Petri são uma ferramenta útil para simular soluções de integração. Também, demonstra que o modelo conceitual do Guaraná pode ser traduzido em um modelo em Redes de Petri estocásticas. Esse estudo é baseado em um problema de integração real, que tem como objetivo automatizar um processo de registro de novos usuários.

6. Considerações Finais

O desenvolvimento do presente estudo possibilitou uma representação de um modelo em Guaraná DSL para Redes de Petri Coloridas. Esse processo de modelagem servirá como suporte para a investigação e análise de modelos conceituais em Guaraná DSL para previsão de possíveis erros e gargalos, ainda na fase de projeto.

Acreditamos que este trabalho pode contribuir com um projeto mais completo sobre análise de processos de integração. O trabalho apresenta uma forma de transformar uma solução representada com Guaraná DSL em um modelo mais formal, para possibilitar verificação e simulação dessa solução em fase de projeto. O objetivo desta modelagem é contribuir com uma proposta de metodologia para a formalização da linguagem Guaraná DSL por meio de Redes de Petri

Ressaltamos que as maiores dificuldades encontradas na modelagem ocorreu frente as tarefas empregadas no modelo conceitual, e principalmente diante da criação de um método que possa aceitar mais pedidos ao mesmo tempo, que deve sempre ter um número fixo no caso desse modelo.

7. Agradecimentos

Este trabalho tem o apoio da Fundação de Amparo à Pesquisa do Estado do Rio Grande do Sul (FAPERGS) no contexto do Programa Pesquisador Gaúcho, com termo de outorga número 17/2551-0001206-2 e da Universidade Regional do Noroeste do Estado do Rio Grande do Sul (UNIJUÍ) no contexto do Programa PIBIC/UNIJUÍ.

Referências

Desel, J. and Reisig, W. (2015). The concepts of petri nets. *Software & Systems Modeling*, 14(2):669–683.

Fahland, D. and Gierds, C. (2013). Analyzing and completing middleware designs for enterprise integration using coloured petri nets. In *Advanced Information Systems Engineering*, pages 400–416. Springer Berlin Heidelberg.

Frantz, R. Z. (2012). *Enterprise Application Integration - An Easy-to-Mantain Model-Driven Engineering Approach*. PhD thesis, Universidad de Sevilla.

Frantz, R. Z., Corchuelo, R., and Roos-Frantz, F. (2016). On the design of a maintainable software development kit to implement integration solutions. *Journal of Systems and Software*, 111:89–104.

He, W. and Xu, L. D. (2014). Integration of distributed enterprise applications: A survey. *IEEE Transactions on Industrial Informatics*, 10(1):35–42.

Hohpe, G. (2005). Your coffee shop doesn't use two-phase commit. *IEEE software*, 22(2):64–66.

Hohpe, G. and Woolf, B. (2004). *Enterprise integration patterns: Designing, building, and deploying messaging solutions*. Addison-Wesley Professional, 1st edition.

Jensen, K. and Kristensen, L. M. (2009). *Coloured Petri nets: modelling and validation of concurrent systems*. Springer Science & Business Media.

Jensen, K., Kristensen, L. M., and Wells, L. (2007). Coloured petri nets and cpn tools for modelling and validation of concurrent systems. *International Journal on Software Tools for Technology Transfer*, 9(3):213–254.

Linthicum, D. S. (2000). *Enterprise application integration*. Addison-Wesley Professional.

Metin, Y. (2018). Die nutzung von oracle integration cloud als ipaas-lösung für eine hybride integration. pages 48–67.

Molloy, M. K. (1982). Performance analysis using stochastic petri nets. *Computers, IEEE Transactions on*, 100(9):913–917.

Petri, C. (1966). Kommunikation mit automaten. schriften des iim nr. 2, institut fur instrumentelle mathematic. Technical report, English translation: Technical Report RADCTR-65-377, Griffiths Air Base, New York.

Roos-Frantz, F., Binelo, M. O., Frantz, R. Z., Sawicki, S., and Basto-Fernandes, V. (2015). Using petri nets to enable the simulation of application integration solutions conceptual models. In *International Conference on Enterprise Information Systems (ICEIS)*, pages 87–97.

Sargent, R. G. (2005). Verification and validation of simulation models. *Proceedings of the 37th conference on Winter simulation*, pages 130–143.

van der Aalst, W. and Stahl, C. (2011). *Modeling Business Processes: A Petri Net-Oriented Approach*. The MIT Press.

Uma abordagem para auxiliar estudantes com deficiência visual na modelagem de sistemas: um estudo piloto

Pedro O. de Azevedo[1], **Vinícius S. L. Vieira**[1], **Alinne C. Correa Souza**[1], **Francisco Carlos M. Souza**[1]

[1]Coordenação de Engenharia de Software
Universidade Tecnológica Federal do Paraná – Dois Vizinhos – PR – Brasil

azevedo.2015@alunos.utfpr.edu.br, vsantos.lvieira@gmail.com

{franciscosouza, alinnesouza}@utfpr.edu.br

Abstract. *In recent years there has been a growing interest in attempts to include people, with special needs, performing Software Engineering activities such as programming and systems modeling. In this context, we present a tool called Blind Modeling system (B-Model) which aims to assist visually impaired people on system modeling activity supported by Unified Modeling Language (UML). For making the use of the B-Model aproach feasible, we implemented a langague called Blind Modeling Language (BML) and a prototype. The BML was evaluated with a blind student and the prototype was assessed through a pilot study using the requirements specification of two different scenarios performed by blind student for use case diagrams generation. The result shows the effectivennes of the B-Model approach, since the generated diagrams correspond to the specified functional requirements using the BML language.*

Resumo. *Nos últimos anos houve um interesse crescente em tentar incluir pessoas com necessidades especiais nas atividades de Engenharia de Software, como programação e modelagem de sistemas. Nesse contexto, neste artigo é apresentada uma ferramenta chamada Blind Modeling system (B-Model), que visa auxiliar pessoas com deficiência visual na atividade de modelagem de sistema utilizando a Linguagem de Modelagem Unificada. Para viabilizar o uso da abordagem B-Model foi desenvolvida uma linguagem chamada Blind Modeling Language (BML) e implementado um protótipo. A BML foi avaliada com uma estudante deficiente visual e o protótipo foi avaliado por meio de um estudo piloto usando a especificação de requisitos de dois cenários diferentes realizada pela estudante deficiente visual para geração de diagramas de casos de uso. O resultado mostra a eficácia da B-Model, pois os diagramas gerados correspondem aos requisitos funcionais especificados utilizando a linguagem BML.*

1. Introdução

Segundo o Instituto Brasileiro de Geografia e Estatística [IBGE 2010], 45 milhões de pessoas, ou seja, 23,9% da população total, apresentaram algum tipo de incapacidade ou deficiência. No que diz respeito à deficiência visual, atualmente, existem 6,5 milhões de brasileiros, sendo 582 mil cegos e 6 milhões com baixa visão. Neste contexto, a inclusão de estudantes com baixa visão ao ensino superior tem sido considerada desafiante.

Esse desafio é decorrente da baixa disponibilidade de livros e/ou materiais em formato acessível, leitores textuais, além das dificuldades no aprendizado de notações gráficas e a falta de capacitação de profissionais [Konecki et al. 2016].

Um obstáculo que estudantes com deficiência visual enfrentam no ensino superior é o aprendizado de disciplinas que exijam interpretações visuais [Giroto et al. 2012]. Dentre tais disciplinas é importante destacar Interação Humano Computador, programação para web, programação para dispositivos móveis e Análise Orientada a Objetos (AOO).

A disciplina de AOO visa demonstrar como é realizada a modelagem de sistemas utilizando a Linguagem de Modelagem Unificada (do inglês, *Unified Modeling Language - UML*). Essa modelagem apoia as especificações de um sistema por meio de diagramas, que permitem a compreensão das funcionalidades e comportamentos do software antes de desenvolvê-lo. Neste contexto, é importante destacar que os diagramas são utilizados com frequência para explicar conceitos, projetar sistemas ou mesmo documentar sistemas existentes. Diante deste cenário, os diagramas se tornam inacessíveis para os estudantes com deficiência visual devido à sua natureza gráfica.

Este artigo visa apresentar um uma ferramenta chamada *Blind Modeling system (B-Model)*, a fim de auxiliar o aprendizado de estudantes com deficiência visual na modelagem de sistemas utilizando UML. A ferramenta é viabilizada por meio de uma linguagem chamada *Blind Modeling Language (BML)* para a especificação de requisitos, que a partir da qual possibilita gerar automaticamente o diagrama de caso de uso correspondente.

Este artigo está organizado da seguinte forma: na Seção 2 são apresentados os trabalhos relacionados. Na Seção 3 é detalhado a ferramenta B-Model. Na Seção 4 são apresentados a forma como foi conduzido o estudo de caso e experimento piloto. Na Seção 5 são apresentados e discutidos os resultados alcançados. Por fim, na Seção 6 as considerações finais e trabalhos futuros são apresentados.

2. Trabalhos Relacionados

A modelagem de sistemas é realizada por meio de diagramas que encapsulam informações visuais, compostos por formas geométricas e conectores relacionando-as. Diversos estudos tem investigado diferentes formas de auxiliar o ensino e a aprendizagem de modelagem de sistemas para deficiente visuais, tais como: ferramentas [Doherty and Cheng 2015], [King et al. 2004], [Pansanato et al. 2012]; representações físicas [Owen et al. 2014]; e notações textuais [Vieritz et al. 2012], [Grillo and de M. Fortes 2014] e [Loitsch et al. 2018], sendo esta última forma o foco deste artigo.

Apesar dos três estudos abordarem notação textual, somente dois [Harmain and Gaizauskas 2000] e [Elallaoui et al. 2018] utilizaram PLN para a geração de diagramas. No estudo de Harmain e Gaizauskas [Harmain and Gaizauskas 2000] foi desenvolvido um protótipo para geração do diagrama de classes a partir de uma especificação de requisitos em linguagem natural. Por outro lado, o trabalho de Elallaoui, Nafil e Touahni [Elallaoui et al. 2018] apresentou um processo de transformação de histórias de usuários em casos de uso utilizando técnicas de PLN juntamente com o analisador *TreeTagger*.

As diferenças da ferramenta B-Model em relação aos trabalhos apresentados são: *(i)* o foco no diagrama de caso de uso, uma vez que a partir deste é possível gerar o diagrama de atividades e por estar diretamente relacionado aos requisitos funcionais; *(ii)* o diagrama gerado está seguindo as notações da UML; e *(iii)* o uso da teoria de Chomsky [Chomsky 1956].

3. Ferramenta B-Model

Nesta seção serão delineados os detalhes da ferramenta B-Model cujo propósito é auxiliar estudantes com deficiência visual no aprendizado da modelagem de sistemas utilizando UML. Conforme pode ser visto na Figura 1, a B-Model é composta por três fases: *(i)* especificação dos requisitos funcionais em linguagem natural; *(ii)* interpretação dos requisitos; e *(iii)* geração do diagrama de caso de uso. Cada fase será detalhada nas próximas seções.

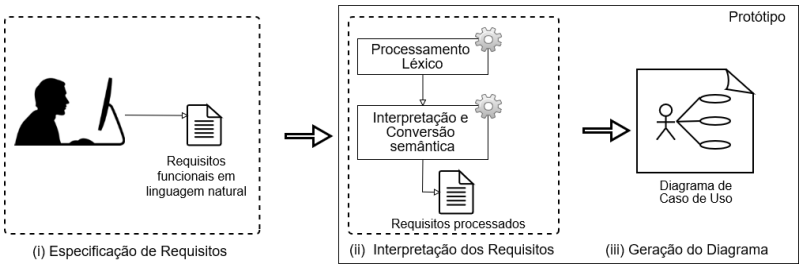


Figura 1. Visão geral da ferramenta B-Model

3.1. Fase 1: Especificação de Requisitos

Os requisitos de software representam as necessidades apresentadas pelos usuários ou clientes, que por sua vez devem ser contempladas pelo sistema a ser desenvolvido. Nesta fase o estudante deverá, em linguagem natural, especificar textualmente os requisitos de um cenário disponibilizado pelo professor, por exemplo.

Para o contexto deste artigo, somente serão considerados os requisitos funcionais, uma vez que estes são necessários para a geração do diagrama de casos de uso. As especificações dos requisitos serão escritos em português salvos em um arquivo de texto que será utilizado como entrada na próxima fase da B-Model.

3.2. Fase 2: Interpretação dos Requisitos

Para auxiliar a interpretação dos requisitos foi definida e desenvolvida uma linguagem chamada *Blind Modeling Language (BML)*. BML é uma linguagem que permite uma padronização na especificação dos requisitos para gerar o diagrama de caso de uso. Nesta fase é realizado o processamento e interpretação dos requisitos especificados utilizando a técnica PLN. A BML foi dividida em duas etapas: *(i)* processamento Léxico; e *(ii)* interpretação e conversão semântica.

No processamento léxico (etapa *(i)* da BML), os requisitos funcionais (entradas) são classificados em categorias conhecidos como *Parts Of Speech (POS)* [Allen 1995]. Nesta etapa foram consideradas as seguintes categorias:

- *determiners*: palavras que determinam o significado de um nome. Neste caso foram considerados os artigos [artigo] definidos no singular (O, o, a, A);

- *nouns:* representam os atores [ator], por exemplo usuário, os objetos [obj], por exemplo login, sistema. Ambos são substantivos e devem ser especificados sempre no singular.
- *verbs:* verbos no infinitivo, por exemplo realizar, que representam a ação que será executada no sistema.
- *prep*: representam as preposições para auxiliar a generalização entre casos de uso, por exemplo entre, em, com.

Além disso, algumas palavras-chaves foram definidas com base no domínio de caso de uso. Para isso, foram criadas três categorias especiais:

- *aux*: auxiliar na definição do nome do caso de uso por meio das palavras "deve"e "devem";
- *conj*: conjunção para auxiliar na generalização de atores e de caso de uso, por meio das palavras "e", "ou", respectivamente;
- *dep*:representam os relacionamentos de *include* e *extend* por meio das palavras "incluindo" e "estendendo", respectivamente;
- *símbolo*: representado por ",";

Para a interpretação e conversão semântica dos requisitos funcionais foi utilizada a teoria de Chomsky [Chomsky 1956] que assume que as sentenças podem ser divididas em *Noun Phrase (NP)* e em *Verb Phrase (VP)*. Essa teoria foi aplicada devido utilizar a estrutura de sentenças como: sujeito, verbo e predicado; e facilitar a especificação de requisitos uma vez que é próxima da linguagem natural.

Além da teoria de Chomsky, regras foram definidas para auxiliar a interpretação e a conversão semântica. É importante ressaltar que a definição dessas regras é fundamental, pois permitem normalizar a escrita de uma sentença, ou seja, padronizar os requisitos especificados. A Tabela 1 apresenta as regras da BML utilizadas para geração do diagrama de caso de uso com base na especificação dos requisitos.

Tabela 1. Regras específicas da BML para a geração do diagrama de caso de uso com base na especificação dos requisitos

Elementos	Regras	Exemplo
Ator	[artigo] + [ator]	O usuário
Caso de uso	[verbo] + [obj]	realizar login
Requisito	[artigo] + [ator]+ [aux] + [verbo] + [obj]	O usuário deve realizar login.
Generalização de ator	[artigo] + [ator] + [conj] + [ator]+ [aux] + [verbo] + [obj]	O atendente e gerente devem realizar login.
Generalização de caso de uso	[artigo] + [ator] + [aux] + [verbo] + [obj] + [simbolo] + [verbo] + [prep] + [obj] + [conj] + [verbo] + [prep] + [obj]	O atendente deve realizar pagamento, pagar com dinheiro ou pagar com cartão.
Include	[artigo] + [ator]+ [aux] + [verbo] + [obj] + [dep] + [verbo] + [obj]	O atendente deve gerar relatório incluindo realizar login.
Extend	[artigo] + [ator]+ [aux] + [verbo] + [obj] + [dep] + [verbo] + [obj]	O atendente deve gerar relatório estendendo realizar impressão.

3.3. Fase 3: Geração do Diagrama

Para a geração do diagrama, o protótipo realizará o processamento léxico e a conversão semântica a partir dos requisitos funcionais especificados em língua portuguesa. É importante destacar que neste estudo os diagramas gerados contemplam atores, casos de uso, suas respectivas relações, e a generalização de atores.

O protótipo foi desenvolvido utilizando a linguagem python e duas bibliotecas dessa linguagem: (i) Spacy[1]; e (ii) Tkinter[2]. A biblioteca Spacy foi utilizada para realizar a etiquetagem das palavras em cada sentença buscando por partes do texto que representam os atores, os casos de uso e seus respectivos relacionamentos. Apesar da existência de outras bibliotecas similares, a seleção da *Spacy* foi devido permitir o processamento da língua portuguesa.

Após a etiquetagem das palavras, o processamento e a interpretação dos requisitos especificados são realizados utilizando PLN por meio da biblioteca *Spacy*. Cada parte de um requisito especificado é associado a um componente que irá compor o diagrama de caso de uso. Por fim, com a biblioteca *Tkinter* foi possível gerar o diagrama, uma vez que permite rotular os atores e casos de uso inserindo *labels* aos respectivos componentes.

4. Avaliação experimental

Dois estudos estudos experimentais pilotos foram realizados: *(i)* estudo de caso para avaliar a a utilidade da linguagem BML; *(ii)* experimento piloto para avaliar a eficácia da ferramenta B-Model. Em ambos os estudos foram utilizados as diretrizes propostas por Wholin et al. [Wohlin et al. 2012].

4.1. Estudo de Caso

Este estudo foi realizado para avaliar a aceitação e uso da linguagem BML com uma estudante deficiente visual do curso de Bacharelado em Engenharia de Software durante a disciplina de AOO no período de agosto a dezembro de 2019. É importante destacar que algumas melhorias na BML foram realizadas durante o período da disciplina a partir dos *feedbacks* da estudante.

A avaliação da BML foi realizada com base no Modelo de Aceitação de Tecnologia (do inglês, *Technology Acceptance Model (TAM)* [Davis et al. 1989]. O modelo é baseado no pressuposto de que a intenção de utilizar um dispositivo tecnológico é determinada pela percepção do usuário sobre sua utilidade e sua facilidade de uso.

Após a definição e planejamento do estudo de caso, sua fase de condução foi realizada em duas etapas: *(1)* especificação de requisitos usando a BML e *(2)* análise da aceitação e uso da BML a partir dos requisitos especificados na primeira etapa. A primeira etapa visa especificar os requisitos de dois cenários diferentes (C = c_1 e c_2) utilizando as regras previamente estabelecidas na BML apresentada na Tabela 1. Esta etapa foi precedida por duas atividades. Na primeira atividade, o docente responsável pela disciplina explicou a linguagem para a estudante e criou um material com explicações e exemplos das regras da BML. Na segunda atividade, a descrição dos cenários foram disponibilizados à estudante, conforme apresentado na Tabela 2. Os requisitos de cada cenário foram especificados em um arquivo de texto.

Por fim, a etapa de análise da aceitação e uso está relacionada às respostas obtidas por meio de um questionário. O questionário foi elaborado com 11 questões em um arquivo de texto, as quais foram agrupadas em três categorias: *(i)* facilidade de uso da linguagem BML; *(ii)* utilidade da linguagem BML; *(iii)* intenção de usar a linguagem

[1] *Spacy* documentation - https://spacy.io/api/doc

[2] *Tkinter* documentation - https://docs.python.org/3/library/tk.html

Tabela 2. Descrição dos dois cenários utilizados

c_1. Sistema de vendas de produtos eletrônicos: uma loja possui notebooks e celulares para venda. A loja possui um atendente cuja função é vender os produtos para os clientes e gerar relatórios semanais. A loja também possui um gerente cuja função é administrar o estoque e administrar fornecedores. Na ausência do atendente, o gerente também realiza as suas atividades. Além disso, após a compra o sistema gera a nota fiscal do produto.
c_2. Sistema de gerenciamento de festas: um sistema online deve gerenciar a compra e venda de ingressos de festa. Nesse sistema tanto o administrador como o gerente podem gerenciar as festas, gerar relatório e consultar festas. Além disso, o gerente pode adicionar patrocínios. O usuário do sistema realiza cadastro, compra ingresso e emite ingresso. Durante a compra do ingresso, a Administradora do cartão autentica a venda. Ao final da compra, o sistema envia um email com ticket do ingresso.

BML no futuro. A estudante realizou a avaliação da BML utilizando a escala *Likert* que consiste em cinco níveis, sendo o nível mais alto representado por "Extremamente"e o nível mais baixo "Nenhum pouco" [Likert 1932].

4.2. Experimento Piloto

Após a avaliação da BML, foi conduzido um experimento piloto para avaliar a eficácia da ferramenta proposta. Para a definição do experimento, o modelo *Goal-Question-Metric (GQM)* proposto por Basili e Weiss [Basili and Weiss 1986] foi utilizado para definir os objetivos deste experimento que consiste em: *"Analisar **a ferramenta B-Model** com o propósito de **avaliar** com respeito à **eficácia** do ponto de vista de **pesquisadores** no contexto de **dois cenários diferentes**."*

Para alcançar o objetivo, a seguinte Questão de Pesquisa (QP) foi investigada: **O quão eficaz é a ferramenta B-Model para especificar requisitos e gerar o diagrama de caso de uso correspondente?** Para responder essa QP, a eficácia da abordagem foi mensurada por meio da relação do diagrama de caso de uso gerado com a especificação de requisitos utilizando a linguagem BML. Para o contexto deste experimento piloto, a primeira versão do protótipo foi avaliada somente com alguns os conceitos que contemplam o diagrama de caso de uso: *(i)* ator; *(ii)* caso de uso; (iii) associação; e *(iv)* generalização de atores. Portanto, os demais conceitos como *include*, *extend* e generalização de casos de uso serão incluídos na próxima versão do protótipo.

Para realizar a avaliação da ferramenta B-Model foram utilizados os dois cenários apresentados na Tabela 2. Apesar dos cenários serem fictícios, os mesmos compreendem os elementos necessários que um estudante deve abstrair para a modelagem de sistema. Por fim, para a condução do experimento piloto foram utilizadas como entrada as especificações dos requisitos dos cenários, realizadas pela estudante com deficiência visual utilizando a BML. Essas especificações foram inseridas no protótipo, as quais foram processadas e interpretadas para a geração do diagrama de caso de uso correspondente.

5. Análise e Discussão dos Resultados

Nesta seção são detalhados os resultados alcançados para a linguagem BML e para a ferramenta B-Model. Na Tabela 3 são apresentados os resultados das respostas da estudante em relação as categorias apresentadas na Seção 4.1. Os resultados sugerem que a BML é uma alternativa eficaz que pode ser utilizada para a especificação de requisitos e geração de diagrama de caso de usos. Um dos principais benefícios da BML é prover uma

especificação mais próxima da linguagem natural, garantindo assim, uma especificação mais consistente, inequívoca e completa para gerar os diagramas de caso de uso.

Tabela 3. Resultados referentes à facilidade de uso, utilidade e intenção de uso no futuro da BML

Categoria	Perguntas	Avaliação
Facilidade de uso	1. A sua interação com a linguagem BML foi clara e compreensível?	4
	2.Interagir com a linguagem BML exigeu muito esforço mental?	2
	3. Você considera a linguagem BML fácil de usar?	5
	4. Você considera a linguagem BML fácil de ser aprendida?	4
Utilidade	5. A linguagem BML melhorou seu desempenho?	5
	6. A linguagem BML permitiu aumentar sua produtividade?	5
	7. A linguagem BML foi útil para o aprendizado?	5
	8. A linguagem BML lhe permitiu realizar trabalhos em grupo?	5
	9. A linguagem BML pode lhe ajudar no mercado de trabalho?	5
Intenção de uso no futuro	10. Você pretende usar novamente a linguagem BML em outras disciplinas?	4
	11. Você recomendaria a linguagem BML para outras pessoas?	5

No que diz respeito ao experimento piloto, nas Figuras 2 e 3 são apresentadas as especificações de requisitos e os diagramas de caso de uso gerados para os cenários c_1 (sistema de vendas de produtos eletrônicos) e c_2 (sistema de gerenciamento de festa), respectivamente.

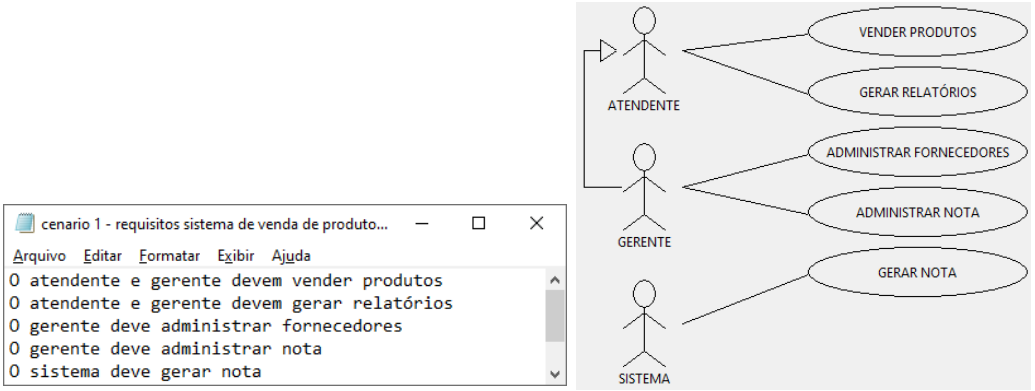


Figura 2. Especificação de requisitos do sistema de vendas de produtos eletrônicos e Diagrama de caso de uso correspondente gerado

É importante destacar que o enfoque deste experimento piloto foi corroborar se o protótipo gera o diagrama de caso de uso correspondente à especificação de requisitos utilizando a BML. Para isso, primeiramente, foi realizado um estudo de caso com uma estudante deficiente visual para a avaliação da BML. Nesse contexto, os resultados alcançados indicam que a ferramenta é eficaz, e consequentemente viável para auxiliar estudantes com deficiência visual na modelagem de sistemas. Conforme pode ser observado nas Figuras 3, por exemplo, a ferramenta B-Model foi capaz de reconhecer todos os atores e suas respectivas relações com seus casos de uso; diferenciou corretamente atores com identificadores similares, como é o caso de “Administrador” e “Administradora”; e atribuiu a generalização identificada aos atores que a possuem, considerando a sequência dos mesmos na especificação de requisitos.

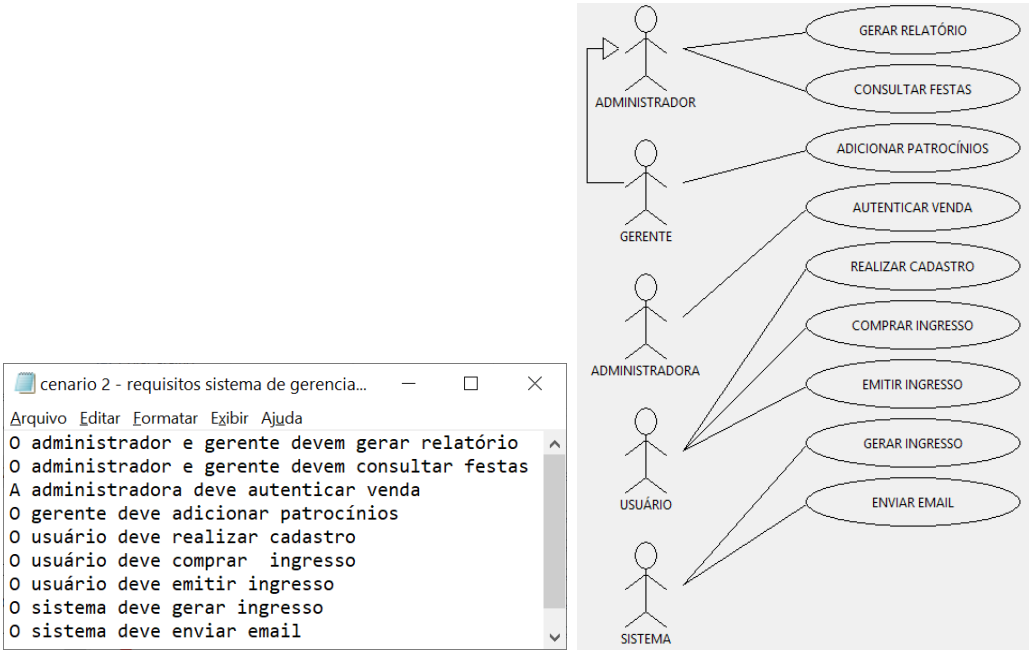


Figura 3. Especificação de requisitos do sistema de gerenciamento de festa e o seu respetivo diagrama de caso de uso gerado

No âmbito real de ensino e aprendizagem, por exemplo, estudantes com deficiência visual utilizam ferramentas de edição de texto para inserção do conteúdo e leitores de tela para verificação. Especificamente no contexto de modelagem de sistemas, o processo de ensino e construção de diagramas é realizado por meio de uma descrição textual, uma vez que estruturas visuais podem não fazer sentido ou nenhum sentido para estudantes que são deficientes visuais de nascença.

Portanto, não existe problema em realizar uma modelagem de sistemas por meio de uma descrição textual, pois é uma prática comum entre os estudantes com essa deficiência, seja no ensino, médio ou superior. No entanto, um artefato visual, como diagramas, é importante para inclusão destes estudantes em diferentes âmbitos, tais como:

- **ensino:** a modelagem de sistemas atualmente é realizada utilizando o padrão universal UML. Dessa forma, o artefato visual se torna imprescindível para a avaliação do estudante, bem como a sua interação com os demais colegas sem deficiência.
- **indústria:** assim como no ensino, os diagramas de UML são utilizados como documentos e artefatos para auxiliar no processo de desenvolvimento de software. Portanto, por meio de uma ferramenta de geração de diagramas via linguagem natural é possível que estudantes com deficiência visual tenham as mesmas oportunidades para ingressar em um cargo de engenheiro de software, por exemplo.

5.1. Ameaças a Validade

Nesta seção são detalhadas as possíveis ameaças à validade que podem afetar os valores e as conclusões do experimento piloto conduzido. As ameaças à validade externa estão relacionadas à generalização dos resultados. A representatividade dos cenários pode ser um problema, pois não existem teorias que garantam que um determinado conjunto

de cenários selecionados seja uma amostra representativa para a condução dos estudos empíricos. Com a finalidade de minimizar esse problema, foram utilizados dois cenários pertencentes a diferentes contextos . No entanto, não é possível afirmar que os resultados podem ser generalizados para qualquer cenário. Neste contexto, replicações futuras são necessárias para corroborar os resultados obtidos.

As ameaças à validade por construção estão preocupadas com a relação entre a teoria e o que é observado. No desenvolvimento da ferramenta proposta bem como na análise dos cenários, possíveis equívocos podem ter sido cometidos. Para mitigar esse risco foram realizados diversos testes das regras a fim de assegurar que durante a condução do experimento a linguagem e a ferramenta tivessem êxito na sua execução.

Por fim as ameças à validade interna caracterizam o grau de confiabilidade entre os resultados esperados e os resultados obtidos. Todas as variáveis do experimento piloto foram controladas para mitigar possíveis ameaças. Além disso, para a ampliar a confiabilidade dos resultados, os dados obtidos foram validados junto as regras definidas para assegurar que os resultados apresentados sejam realmente ortogonais, coerentes e interpretados de forma apropriada.

6. Considerações Finais

Neste artigo foram apresentados e discutidos os resultados da ferramenta B-Model que visa auxiliar estudantes com deficiência visual na modelagem de sistemas utilizando UML. Para a geração do diagrama de caso de uso, os requisitos funcionais especificados utilizando a linguagem BML foram processados por meio da técnica PLN juntamente com a teoria de Chomsky. Considerando a importância do diagrama de caso de uso, o foco que este recebeu ao longo do trabalho é justificado pelo fato de ser o principal diagrama utilizado no diálogo com o usuário ao identificar e validar os requisitos. Os casos de uso constituem elementos que estruturam todas as etapas do processo de software e, a partir deste, é possível gerar o diagrama de atividades, por exemplo.

Portanto, os resultados indicam que a ferramenta é promissora e eficaz pelas seguintes razões: *(i)* permite a descrição dos requisitos por meio de uma linguagem específica reduzindo alguns riscos presentes na especificação de requisitos como incompletude e ambiguidade; e *(ii)* gera automaticamente o diagrama de caso de uso baseado nos requisitos especificados utilizando a linguagem. Como trabalhos futuros, pretende-se: *(i)* avaliar a BML com outros estudantes deficientes visuais; *(ii)* incluir na ferramenta os relacionamentos de dependência do tipo *include* e *extend* e generalizações entre caos de uso; e *(iii)* validar a ferramenta com estudantes deficientes visuais.

Referências

Allen, J. (1995). *Natural Language Understanding*. Addison-Wesley, 15^{th} edition.

Basili, V. R. and Weiss, D. M. (1986). A methodology for collecting valid software engineering data. *IEEE Transactions on Software Engineering*, 10(6):728–738.

Chomsky, N. (1956). Three models for the description of language. *RE Transactions on Information Theory*, 2(3):113–124.

Davis, F. D., Bagozzi, R. P., and Warshaw, R. P. (1989). User acceptance of computer technology: A comparison of two theoretical models. *Manage. Sci.*, 35(8):982–1003.

Doherty, B. and Cheng, B. H. C. (2015). UML modeling for visually-impaired persons. In *Proceedings of the 1st International Workshop on Human Factors in Modeling co-located with 18th International Conference on Model Driven Engineering Languages and Systems*, pages 4–10.

Elallaoui, M., Nafil, K., and Touahni, R. (2018). Automatic transformation of user stories into uml use case diagrams using nlp techniques. *Procedia Computer Science*, 130:42–49.

Giroto, C. R. M., Poker, R. B., and Omote, S. (2012). *As tecnologias nas práticas pedagógicas inclusivas*. Editora Unesp, São Paulo.

Grillo, F. D. N. and de M. Fortes, R. P. (2014). Accessible modeling on the web: A case study. *Procedia Computer Science*, 27:460–470.

Harmain, H. M. and Gaizauskas, R. (2000). Cm-builder: an automated nl-based case tool. In *Proceedings in 15th IEEE International Conference on Automated Software Engineering*, pages 45–53.

IBGE (2010). *Características Gerais da População, Religião e Pessoas com Deficiência*. IBGE, Rio de Janeiro, RJ.

King, A., Blenkhorn, P., Crombie, D., Dijkstra, S., Evans, G., and Wood, J. (2004). Presenting uml software engineering diagrams to blind people. In *Proceedings of the International Conference on Computers for Handicapped Persons*, pages 522–529.

Konecki, M., Ivković, N., and Kaniški, M. (2016). Making programming education more accessible for visually impaired. In *Proceedings of the 39th International Convention on Information and Communication Technology, Electronics and Microelectronics*, MIPRO, pages 887–890.

Likert, R. (1932). A technique for the measurement of attitudes. *Journal Archives of Psychology*, 22(40):1–55.

Loitsch, C., Müller, K., Seifermann, S., Henß, J., Krach, S., and Stiefelhagen, G. J. R. (2018). UML4ALL syntax – a textual notation for UML diagrams. In *Proceedings of the 16th International Conference on Computers Helping People with Special Needs*, ICCHP'18, pages 598–605. Springer.

Owen, C. B., Coburn, S., and Castor, J. (2014). Teaching modern object-oriented programming to the blind: An instructor and student experience. In *Proceedings of the 121st American Society for Engineering Education*, ASSE'14, pages 1–13.

Pansanato, L. T. E., Bandeira, A. L. M., dos Santos, L. G., and do Prado Pereira, D. (2012). Projeto D4ALL: acesso e manipulação de diagramas por pessoas com deficiência visual. In *Proceedings of the 11th Brazilian Symposium on Human Factors in Computing Systems*, IHC'12, pages 33–36.

Vieritz, H., Schilberg, D., and Jeschke, S. (2012). Access to uml diagrams with the HUTN. In *Proceedings of the 14th International ACM Sigaccess Conference on Computers and Accessibility*, ASSETS'12, pages 237–238.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., and Wesslén, A. (2012). *Experimentation in Software Engineering: An Introduction*. Springer-Verlag Berlin Heidelberg, 1^{st} edition.

Aplicação da Ferramenta Google Colaboratory para o Ensino da Linguagem Python

Martony Demes da Silva [1]

[1] Departamento de Ciência da Computação – Universidade Federal do Piauí (UFPI)
Teresina – PI – Brasil

Abstract. *This work presents the application of the Google Collaboratory tool (or colab) in the teaching of Python programming language. Through this environment, it is possible for student interaction in the practice of programming activities in an easy and deductive way. To validate this ease of use, an assessment was made using the Perceived Utility and Ease of Use perceived in this tool. The results showed that the two analyzes had rates above 90%. In this context, the platform is easy to use and effortless*

Resumo. *Esse trabalho apresenta a aplicação da ferramenta Google Colaboratory (ou colab) no ensino de linguagem de programação Python. Por meio desse ambiente, é possível a interação do estudante na prática de atividades de programação de maneira fácil e dedutiva. Para validar essa facilidade de uso, fez-se uma avaliação por meio da Utilidade Percebida e Facilidade de Uso percebida nesta ferramenta. Os resultados apontaram que as duas análise tiveram índices superiores a 90%. Nesse contexto, a plataforma é de fácil utilização e sem necessidade de esforço*

1. Introdução

A ferramenta digital *Google Colaboratory* ou *Colab* é um ambiente digital de acesso aberto e disponibilizado pela *Google* para aplicação de conhecimentos de programação na linguagem *Python* [Carneiro et al. 2018]. Por meio dos recursos providos, a *Colab* facilita práticas de programação. De tal maneira, utiliza-se essa ferramenta em outras disciplinas: matemática [Access 2020] e medicina [Balaraman 2020].

O ensino de programação ainda é um desafio no contexto da computação [Neto and Schuvartz 2007], [Amaral et al. 2017] e [Blatt et al. 2017]. Entre os motivos tem-se a questão de estudantes iniciantes precisam adquirir habilidades como a abstração e o raciocínio lógico. Diante disso, o professor precisa motivar os alunos pelo interesse na disciplina.

Por outro lado, o professor não consegue realizar um atendimento individualizado. Isso se deve ao grande número de alunos e reduzida disponibilidade de tempo. Dado esse desafio, as disciplinas de programação ainda tem um alto índice de reprovação em programação [Amaral et al. 2017].

Diante do contexto de Engenharia de Software, verificou-se que o uso de ferramentas colaborativas e integradas propiciam novas oportunidades ao processo de ensino-aprendizagem na computação. E práticas com novas abordagens no âmbito educacional, como planejamento integrado e uso de funcionalidades específicas, permitem executar as tarefas na internet de modo colaborativo [Abegg et al. 2010].

Nesse cenário, esta pesquisa objetiva avaliar a ferramenta colaborativa e online *google colab* aplicada no ensino de programação em *Python*. O uso da ferramenta favorece a interação do aluno na prática por meio dos recursos facilitadores da plataforma *colab*. No contexto virtual, a ferramenta permite que vários alunos interajam ao mesmo tempo ou no mesmo arquivo. A validação desta pesquisa foi feita por meio de uma avaliação com estudantes do curso de *Python* na modalidade a distância (EaD).

Vale ressaltar que a motivação na escolha da *colab* se deve por ser uma ferramenta integrada da plataforma da google e possui maior disponibilidade de recursos comparado-se com outras tecnologias similares como a *jupyter*. Essas tais características são: bibliotecas pre-instaladas, uso de recursos da máquina virtual da *google*, acessado de qualquer lugar entre outras que serão especificadas na Seção 3.

Este trabalho está organizado da seguinte forma: na Seção 1 abordou sobre os trabalhos relacionados. A Seção 3 apresenta a ferramenta desta pesquisa. A Seção 4 detalha a metodologia da pesquisa. A Seção 5 ilustra os resultados da pesquisa e a Seção 6 finaliza com as considerações finais.

2. Referencial Teórico

Conforme já exposto, o ensino de programação é um grande desafio em cursos técnico e de graduação. Para tanto, alguns métodos e ferramentas tem sido propostos para facilitar o processo de ensino aprendizagem.

Um trabalho relevante foi feito por [Blatt et al. 2017]. Nessa pesquisa, fez-se um mapeamento sistemático da literatura sobre metodologias e ferramentas para apoio ao ensino de programação. Entre os resultados apresentados, tem-se a preferência dos professores em utilizar *Scratch* como ferramenta de apoio. O *Scratch* possibilita que estudantes (inclusive crianças) elaborem blocos visuais por meio de animações, histórias interativas ou jogos.

Outra alternativa para facilitar o ensino de programação é o auxilio de mecanismos de inteligencia artificial (IA) por meio de tutores inteligentes[Neto and Schuvartz 2007]. Essa pesquisa propôs um ambiente ativo, com suporte aos estudantes e auxiliando nas dificuldades encontradas.

No mesmo contexto teórico relacionado, pesquisas propuseram e validaram o uso de ferramenta de apoio ao processo de ensino e aprendizagem de lógica e programação para iniciantes como em [Amaral et al. 2017]. Com objetivo semelhante, a pesquisa de [Farias et al. 2015] propõe um ambiente que facilita a interação e comunicação de estudantes iniciantes e professor.

Por fim, outra proposta relevante é a apresentada por [Torezani and de Lira Tavares 2014]. Nesta pesquisa, é validado um editor para desenvolver atividades de aprendizagem de programação para crianças. Os resultados mostraram que foi possível elaborar atividades adequadas ao perfil dos estudantes.

No âmbito desse tema, à luz da engenharia de software, não encontrou-se trabalhos com uso de ferramentas aplicadas no ensino de Python e ferramentas colaborativas e onlines. As abordagens com uso da ferramenta *Google colab* encontradas são restritas a comparação com hardware real [Carneiro et al. 2018], a comparação de algoritmos de classificação [Balaraman 2020] e em aplicação em estudos de matemática

[Alves and Machado Vieira 2019] e [Access 2020].

3. Ambiente Google Colaboratory - Colab

O *Google Colab* ou "*Colaboratório*" é um ambiente digital disponível na nuvem, gratuito e hospedado pelo Google. O objetivo desta ferramenta é prover serviços de desenvolvimento em python. O ambiente é uma máquina virtual, denomidada de notebook, em que o usuário desenvolver práticas de programação em python. Ao mesmo tempo que programa, o usuário pode fazer comentário, compilar, criar relatório, texto, tudo junto no mesmo trabalho.

As vantagens do colab são:

1. Suporte para linguagem Python;
2. Aceleração de GPU grátis;
3. Bibliotecas pré-instaladas: Todas as principais bibliotecas Python, como o TensorFlow, o Scikit-learn, o Matplotlib, entre muitas outras, estão pré-instaladas e prontas para serem importadas [Vishakha Lall 2018];
4. Construído com base no Jupyter Notebook;
5. Recurso de colaboração (funciona com uma equipe igual ao Google Docs): permite que os desenvolvedores usem e compartilhem o Jupyter notebook entre si sem precisar baixar, instalar ou executar qualquer coisa que não seja um navegador;
6. Suporte a comandos bash;
7. Os notebooks do Google Colab são armazenados no drive.

Uma particularidade do *colab* é o mecanismo de dividir blocos de textos e de código no mesmo arquivo. Com isso, é possível facilitar os comentários do código, a compilação do código modularizada (separada), elaboração simultânea de relatório, clareza e organização dos estudos. Essas possibilidades favorecem o ensino de programação. A Figura 1, a seguir, ilustra a segregação de bloco de código e bloco de texto.

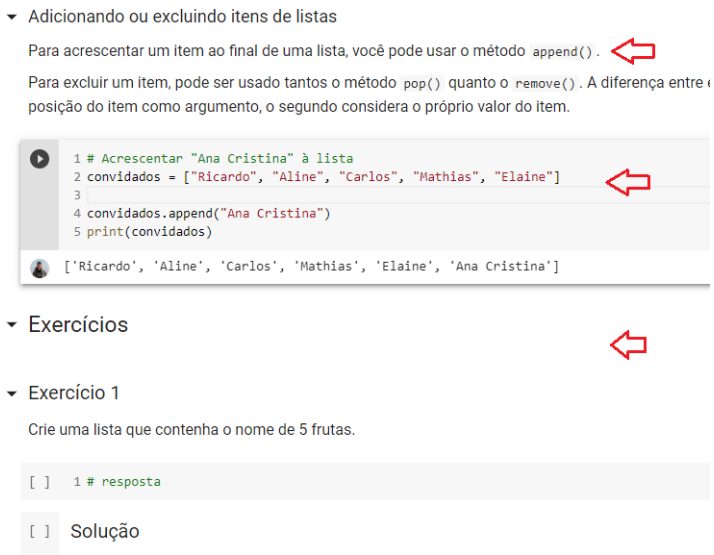
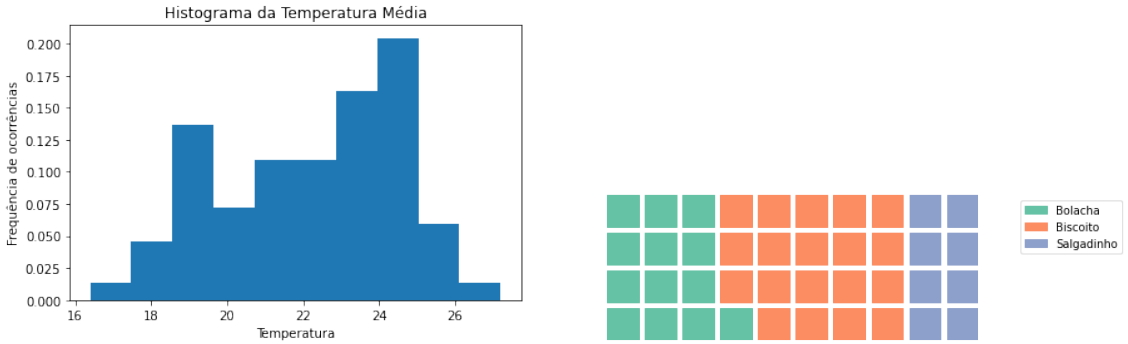


Figura 1. Separação de códigos, textos e atividades

Entre as funcionalidades do *colab*, existe a possibilidade de elaboração de gráficos e tabelas diretamente na ferramenta. A Figura 2 ilustra exemplos de gráficos criados no *colab*. A compilação é modular, sem afetar outras partes



(a) Gráfico tipo histograma (b) Gráfico do tipo Waffle

Figura 2. Exemplos de gráficos no colab

Dada as diversas funcionadas, o *colab* apresenta ainda uma performance compatível com servidores reais[Carneiro et al. 2018], em um ambiente controlado. Como apresentado, *colab* pode ser utilizado por qualquer estudante mesmo que não tenha experiencia com programação. Na Seção 4 será validado essa facilidade por meio de uma avaliação realizada no *colab* com estudantes do curso EaD de *Python*. Importante frisar que alguns encontros do curso são online (em tempo real) do professor com os alunos.

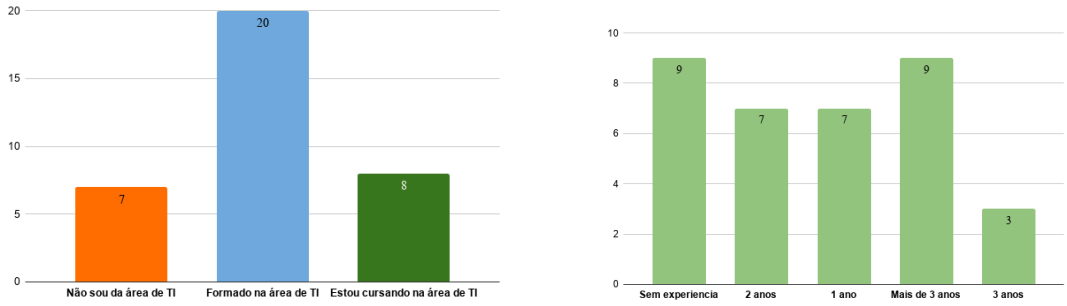
4. Metodologia

Para validar a flexibilidade e facilidade do ambiente em pesquisa, elaborou-se uma avaliação baseado em um experimento quantitativo com alunos do curso de online *Python*. A descrição do experimento será apresentado a seguir.

4.1. Participantes

Os participantes do experimento foram **35 estudantes** de um curso online de *Python*. Inicialmente, foram levantadas as seguintes informações dos envolvidos: a idade, experiência com programação e formação na área de TI.

Em uma análise preliminar dos participantes, a partir do resultado exposto na Figura 3, tem-se as seguintes inferências: dos 35 alunos, 7 não são da área de Tecnologia da Informação. Além disso, 57% dos estudantes já são formados na área de TI. Outro dado relevante é que cerca de 66% dos participantes não têm nenhuma experiencia ou somente até 2 anos de experiência com programação.



(a) Formação na área de TI (b) Experiência com programação

Figura 3. Informações sobre os participantes

4.2. Procedimentos

O experimento foi realizado por meio de um questionário online com 12 questões, ilustradas na Figura 2. Cada questão tem como opções de resposta escores de 1 a 7, conforme o grau de resposta dada por cada participante. O ambiente digital de referência é o *colab*, utilizado no curso de online de programação em *Python*. Os participantes responderam o questionário com base em suas experiências de práticas transcorridas no curso.

Vale frisar que o questionário foi aplicado no período em que os alunos estavam com dois meses de curso. O quantitativo de participantes foi limitado pelo total de estudantes à época no curso

Como citado, a investigação foi guiada por meio do questionário online. Cada questão segue o método da escala Likert [Wainerman 1976] e a metodologia *Technology Acceptance Model (TAM)* [Davis et al. 1989]. TAM apresenta as intenções de uso de um sistema. Essas intenções são fundamentadas em dois aspectos: **Utilidade Percebida (UP)**, que mede o nível em que o indivíduo acredita que o uso do sistema pode melhorar suas atividades; e **Facilidade de Uso Percebida (FUP)**, que mede o grau em que o usuário acredita que o uso do sistema de informação livre de esforço.

Como citado, a escala Likert verifica o nível de concordância do sujeito com várias afirmações que expressam algo favorável ou desfavorável em relação a um objeto psicológico. Com base na abordagem de [Neto et al. 2018], os participantes da pesquisa escolhem entre opções e marcam a resposta conforme sua atitude ou opinião. As opções de respostas disponíveis para cada questão, nesta pesquisa, são: Discordância Total (DT); Discordância Parcial (DP); Discordância Leve (DL); Neutro (N); Concordância Leve (CL); Concordância Parcial (CP); Concordância Total (CT).

Apos aplicação do questionário aos 35 participantes, as respostas foram avaliadas levando em consideração dois pontos [Macnaughton 1996]: quantidade de concordantes (ConP) e Discordantes (DisP). ConP é o calculo das soma entre a quantidade de respostas totalmente concordantes e parcialmente concordantes (Equação 1). Já o DisP é calculado pela soma entre a quantidade de respostas totalmente discordantes e parcialmente discordantes (Equação 2)

$$ConP = CT + CP + CL + \frac{N}{2} \quad (1) \qquad DisP = DT + DP + DL + \frac{N}{2} \quad (2)$$

Com base no resultado de ConP e DisP, calcula-se o grau de concordância de cada proposição (GCP), pela Equação 3, com base na abordagem de [Wilder 1978].

$$GCP = 100 - \left(\frac{100}{\frac{ConP}{Disp} + 1} \right) \tag{3}$$

Diante dos resultados do GCP, utiliza-se a Tabela 1 de referência de valores de GCP a qual indica o quanto o participante concorda ou discorda sobre cada questão [Davis et al. 1989].

Tabela 1. Referência de valores do GCP

Valor de GCP	Frase adequada
90 ou mais	Uma concordância muito forte
80 a + 89,99	Uma concordância substancial
70 a + 79,99	Uma concordância moderada
60 a + 69,99	Uma concordância baixa
50 a + 59,99	Uma concordância desprezível
40 a + 49,99	Uma discordância desprezível
30 a + 39,99	Uma discordância baixa
20 a + 29,99	Uma discordância moderada
10 a + 19,99	Uma discordância substancial
9,99 ou menos	Uma discordância muito forte

5. Resultados

O produto resultante do questionário apresenta dois aspectos: a **UP**, que constitui 6 proposições e a **FUP**, compondo também 6 proposições. A Tabela 2 ilustra as questões definidas para esta pesquisa.

Tabela 2. Proposições da Pesquisa

Utilidade Percebida (UP)	
nº	**Questão**
1	A utilização do *Colab* é importante para o meu aprendizado
2	As funcionalidades do *Colab* oferecem condições semelhantes a um ambiente de IDE (Ambiente de desenvolvimento Integrado)
3	O *Colab* facilita meu entendimento de assuntos passados em aula
4	Usar o *Colab* agrega valor ao meu aprendizado em *Python*
5	As universidades deveriam utilizar o *Colab* para treinamento de alunos de curso de computação
6	Estou motivado a continuar usando o *Colab*
Facilidade de Uso Percebida (FUP)	
7	No *Colab* eu sempre sei onde estou e como chegar onde quero chegar
8	O *Colab* tem uma interação compreensível e clara
9	Os recursos de programação e navegação são fáceis de encontrar
10	O *Colab* possui uma boa amigabilidade (fácil de aprender a usar)
11	Consigo utilizar o *Colab* sem auxílio de um instrutor
12	Utilizar o *Colab* é agradável

Com base nas respostas das questões da pesquisa dispostas na Tabela 2, calculou-se os valores das respostas de 35 participantes da pesquisa. O resultado é apresentado na Tabela 3.

Ao examinar o quadro de **Utilidade Percebida**, da Tabela 3, verifica-se que todas as seis questões tem *Uma concordância muito forte*, com índice GCP acima de 90%. Isso, tomando como parâmetro a Tabela 1. A questão 2 apresenta um valor relativamente abaixo. Isso é considerável pois a pergunta faz um comparativo com IDE. Diante desses resultados, pode-se sustentar que o *Colab* apresenta uma boa utilidade percebida pelos estudantes.

Os resultados para a **Facilidade de Uso Percebida** são equivalente ao UP. A FUP avalia o grau em que o usuário acredita que o uso da ferramenta é livre de esforço.Na Tabela 3 é apresentado, em todas as questões, valores GCP superior a 90%. O resultado reproduz que o *Colab* é de fácil utilização e sem necessidade de esforço - ”Uma concordância muito forte”, descrito na Tabela 1.

Tabela 3. Resultados da Pesquisa

Utilidade percebida											
Questão	DT	DP	DL	N	CL	CP	CT	QTR	DisP	ConP	GCP
1	0	0	0	1	2	5	27	35	0,5	34,5	98,57
2	0	0	2	3	6	10	14	35	3,5	31,5	90,0
3	0	0	0	2	4	5	24	35	1	34	97,14
4	0	0	0	1	0	6	28	35	0,5	34,5	98,57
5	0	1	0	0	1	4	29	35	1	34	97,14
6	0	0	0	1	2	8	24	35	0,5	34,5	98,57
Facilidade de uso percebida											
Questão	DT	DP	DL	N	CL	CP	CT	QTR	DisP	ConP	GCP
7	0	0	0	2	2	7	24	35	1	34	97,14
8	0	0	0	1	2	5	27	35	0,5	34,5	98,57
9	0	0	0	3	2	9	21	35	1,5	33,5	95,71
10	0	0	0	2	7	6	20	35	1	34	97,14
11	0	0	0	3	1	10	21	35	1,5	33,5	95,71
12	0	0	0	1	3	9	22	35	0,5	34,5	98,57

Conforme exposto, não encontrou-se estudos com uso de ferramentas computacionais colaborativas no ensino de linguagem de programação *Python*. Diante dessa lacuna, este trabalho apresentou resultados significativo no uso da ferramenta *Colab*, destacando a facilidade de uso.

6. Considerações Finais

Este trabalho apresentou a o ambiente colaborativo *Google Colab* como ferramenta de apoio ao ensino de programação Python. No âmbito da engenharia de software e com base nos resultados apresentados, na Seção 5, o uso desta ferramenta propicia a prática de programação e reduz a complexidade que surgem durante aprendizagem de programação.

O diferencial ou ganho computacional apresentado nesta pesquisa é a alternativa de uso de uma ferramenta que facilita o ensino de programação em *Python* aos estudantes de computação.

Nesse âmbito, o estudante de programação pode concentra-se dividir seus estudos em módulos organizados e acompanhado pelo professor colaborativamente. Para trabalhos futuros, pretende-se realizar uma avaliação comparativa de performance de processamento do *colab* com outras ferramentas IDE.

Referências

[Abegg et al. 2010] Abegg, I., Bastos, F. d. P. d., and Müller, F. M. (2010). Ensino-aprendizagem colaborativo mediado pelo wiki do moodle. *Educar em Revista*, 1(38):205–218.

[Access 2020] Access, O. (2020). Visualizing the Newtons Fractal from the Recurring Linear Sequence with Google Colab : An Example of Brazil X Portugal Research. *International Electronic Journal of Mathematics Education*, 15(3).

[Alves and Machado Vieira 2019] Alves, F. R. V. and Machado Vieira, R. P. (2019). The Newton Fractal's Leonardo Sequence Study with the Google Colab. *International Electronic Journal of Mathematics Education*, 15(2).

[Amaral et al. 2017] Amaral, E., Camargo, A., Gomes, M., Richa, C. H., and Becker, L. (2017). ALGO+ Uma ferramenta para o apoio ao ensino de Algoritmos e Programação para alunos iniciantes. *Anais do XXVIII Simpósio Brasileiro de Informática na Educação (SBIE 2017)*, 1(Cbie):1677.

[Balaraman 2020] Balaraman, S. (2020). Comparison of Classification Models for Breast Cancer Identification using Google Colab. *Preprints*, pages 1–12.

[Blatt et al. 2017] Blatt, L., Becker, V., and Ferreira, A. (2017). Mapeamento Sistemático sobre Metodologias e Ferramentas de apoio para o Ensino de Programação. *Anais do XXIII Workshop de Informática na Escola (WIE 2017)*, 1(Cbie):815.

[Carneiro et al. 2018] Carneiro, T., Da Nobrega, R. V. M., Nepomuceno, T., Bian, G. B., De Albuquerque, V. H. C., and Filho, P. P. R. (2018). Performance Analysis of Google Colaboratory as a Tool for Accelerating Deep Learning Applications. *IEEE Access*, 6:61677–61685.

[Carneiro et al. 2018] Carneiro, T., Medeiros Da NóBrega, R. V., Nepomuceno, T., Bian, G., De Albuquerque, V. H. C., and Filho, P. P. R. (2018). Performance analysis of google colaboratory as a tool for accelerating deep learning applications. *IEEE Access*, 6:61677–61685.

[Davis et al. 1989] Davis, F. D., Bagozzi, R. P., and Warshaw, P. R. (1989). User acceptance of computer technology: a comparison of two theoretical models. *Management science*, 35(8):982–1003.

[Farias et al. 2015] Farias, H., Bonifácio, B., and Ferreira, R. (2015). Avaliando o Uso da Ferramenta Scratch para Ensino de Programação através de Análise Quantitativa e Qualitativa. *Anais do XXVI Simpósio Brasileiro de Informática na Educação (SBIE 2015)*, 1(Sbie):947.

[Macnaughton 1996] Macnaughton, R. J. (1996). Numbers, scales, and qualitative research. *The Lancet*, 347(9008):1099–1100.

[Neto et al. 2018] Neto, A. S., Neto, F. M., Lima, R., Silva, S., and de Oliveira, E. J. (2018). Avaliação de um ambiente virtual gamificado para auxiliar o ensino-aprendizagem de estudantes de medicina. In *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*, volume 29, page 496.

[Neto and Schuvartz 2007] Neto, W. C. B. and Schuvartz, A. A. (2007). Ferramenta Computacional de Apoio ao Processo de Ensino-Aprendizagem dos Fundamentos de Programação de Computadores. *SBIE - Simpósio Brasileiro de Informática na Educação*, 17:520–528.

[Torezani and de Lira Tavares 2014] Torezani, C. and de Lira Tavares, O. (2014). Eanewprog-um editor de atividades para o ambiente online newprog. In *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*, volume 25, page 422.

[Vishakha Lall 2018] Vishakha Lall (2018). Google colab — the beginner's guide. Access date: 1 jun. 2020.

[Wainerman 1976] Wainerman, C. (1976). *Escalas de medición en ciencias sociales*. Ediciones Nueva Visión.

[Wilder 1978] Wilder, J. W. (1978). *New concepts in technical trading systems*. Trend Research.

Hábitos de estudos de graduandos em Ciência da Computação e Engenharia de Software: uma análise no período de pandemia

Flávia A. Barbosa, Ícaro M. Crespo, Lucas A. G. Garais, Andréa S. Bordin

[1]Universidade Federal do Pampa (Unipampa) – *Campus* Alegrete
Av. Tiarajú, 810 – 97.546-550 – Alegrete – RS – Brazil

{flaviabarbosa.aluno, icarocrespo.aluno}@unipampa.edu.br

{lucasgarais.aluno, andreabordin}@unipampa.edu.br

Abstract. *During the pandemic period, the teaching activities from most universities were suspended. As a consequence, it was observed that a lot of students felt unmotivaded. This study objective to analise study habits of Computer Science (CS) and Software Engineering (SE) students from Unipampa/Alegrete. For this, a survey was carried out in which 81 responses were obtained. The data analisys revealed that students of SE felt more motivated and dedicatd more time to studies, unlike the CS students who indicated they miss the face-to-face classes. The results of this research can be used to plan remote teaching activities, as well as assist in the curriculum management of the courses in question.*

Resumo. *Durante o período de pandemia, as atividades em grande parte das universidades foram suspensas. Como consequência, observou-se que muitos estudantes se sentiram desmotivados. Este estudo objetiva analisar os hábitos de estudos de alunos dos cursos de Ciência da Computação (CC) e Engenharia de Software (ES) da Unipampa/Alegrete. Para isso, foi realizado um survey no qual foram obtidas 81 respostas. A análise dos dados revelou que alunos de ES se sentiram mais motivados e dedicaram mais tempo aos estudos, diferente dos alunos de CC que indicaram sentir falta das aulas presenciais. Os resultados desta pesquisa podem ser utilizados para o planejamento de atividades de ensino remotas, assim como auxiliar na gestão curricular dos cursos em questão.*

1. Introdução

O ano de 2020 trouxe uma mudança brusca em diversos segmentos da sociedade em virtude da pandemia do vírus Sars-CoV-2 (Síndrome Respiratória Aguda Grave) [BRASIL 2020b]. Como forma de prevenção à pandemia foram aplicadas medidas de isolamento e distanciamento social. As aulas presenciais nos diferentes níveis educacionais foram suspensas, por decisão do Ministério da Educação brasileiro [BRASIL 2020a].

De acordo com [Van Nguyen et al. 2020], a parada repentina neste aprendizado resulta em perda de interesse em adquirir novos conhecimentos. Como consequências desta ação, podemos citar queda no rendimento, alienação, desmotivação para estudar e baixa na participação em atividades acadêmicas, além de perda do sentimento de pertencimento aos estabelecimentos de ensino [Marques 2020].

Como alternativas à suspensão de atividades presenciais, muitas instituições de ensino propuseram ações *online* nos eixos de ensino, pesquisa e extensão para a manutenção do vínculo proporcionado pelo ensino da universidade. Na Universidade Federal do Pampa (Unipampa), campus Alegrete, ações com este objetivo também foram desenvolvidas. Dentre essas ações, destaca-se o I Ciclo *Online* de Palestras dos cursos de Ciência da Computação (CC) e Engenharia de Software (ES).

O presente estudo tem como motivação a avaliação das palestras do ciclo supracitado, onde questionou-se sobre hábitos de estudos no período de isolamento. A análise das respostas revelou que parte dos alunos dedicava pouco tempo às atividades de estudo, assim como apresentava um quadro de desânimo à situação de isolamento. No entanto, outra parte manifestou que estava buscando estudar assuntos da área de Computação. Como a avaliação era voluntária, o número de respondentes não foi expressivo. Dessa forma, entendeu-se que era necessário estender esses questionamentos para um maior número de discentes de CC e ES.

Uma busca na literatura encontrou estudos que investigaram os diversos impactos da pandemia de COVID-19 no ensino de uma forma mais global. O estudo de [Van Nguyen et al. 2020] apresenta dados sobre o comportamento e percepção de estudantes de graduação de universidades do Vietnã, onde cerca de 81% dos respondentes alegaram que a COVID-19 afetou significativamente as atividades de trabalho/estudo. Além desse, o estudo de [Onyema et al. 2020] analisou os impactos da pandemia em diferentes graus da educação de diferentes países, no qual mais de 70% dos participantes concordaram que infraestrutura inadequada, como falta de computadores e internet, foi o principal fator que limitou seu envolvimento na educação *online*. Não foram encontrados estudos especificamente relacionados a alunos de graduação da área de Computação.

Neste artigo é apresentada uma análise dos hábitos de estudos e envolvimento em atividades acadêmicas dos alunos dos cursos de Ciência da Computação e Engenharia de Software da UNIPAMPA durante o período de isolamento da pandemia. Para isso, foi conduzido um *survey* com questões demográficas, de infraestrutura e de hábitos de estudos. Dentre as contribuições deste estudo estão a identificação de perfis de alunos com hábitos de estudos diferentes durante a pandemia, assim como a compilação das fontes de estudos mais utilizadas pelos alunos e as áreas e assuntos pelos quais mais se interessaram.

2. Metodologia

O *survey* desta pesquisa seguiu o processo proposto por [Kasunic 2005], no qual tem-se as seguintes etapas: Planejamento, Elaboração, Teste piloto, Execução e Análise. Na etapa de planejamento, o escopo e o conjunto de questões da pesquisa aperfeiçoaram-se por meio de reuniões entre os pesquisadores. Definiu-se uma pesquisa de caráter quantitativo e qualitativo, direcionada a discentes dos cursos de CC e ES da Unipampa. Embasado nos tipos de amostragem da pesquisa, optou-se pela técnica probabilística de amostragem aleatória simples, na qual todos os elementos da população, ou seja, alunos dos cursos da área de Computação, têm a mesma probabilidade de pertencerem à amostra [Grzybovski 2005].

Na etapa de elaboração, definiu-se o mecanismo utilizado para a coleta de dados,

sendo este um formulário disponibilizado via plataforma Google Forms[1]. O formulário constituiu-se em 18 questões subdivididas em 5 grupos, sendo eles: identificação; infraestrutura necessária ao estudo/pesquisa; hábitos de estudos; fontes de consultas e atividades acadêmicas.

Para efeito de teste e refatoração de possíveis inconsistências no instrumento, na etapa de teste piloto o instrumento foi aplicado inicialmente a um grupo focal, constituído por 2 discentes de ES e 1 discente de CC. Os discentes foram encorajados a apontar problemas pertinentes ao instrumento de pesquisa. Feito isso, cumpriu-se a análise e correção dos potenciais problemas. Um novo pré-teste foi realizado, desta vez utilizando 8 discentes, sendo 2 de CC e 6 de ES. A versão final das questões é exibida na Tabela 1. O convite de participação na pesquisa foi feito para todos os alunos dos dois cursos (348 matriculados), por meio do endereço eletrônico institucional. Além desse meio, redes sociais foram utilizadas para maior alcance de participações.

Na etapa de análise de dados, primeiramente verificou-se a consistência e integridade das respostas obtidas na coleta. Ademais, realizou-se um processo de reestruturação nos dados para facilitar a análise. A reestruturação concretizou-se a partir da organização dos dados de forma ordenada em planilhas eletrônicas compartilhadas entre a equipe, de acordo com as questões e problemáticas impostas.

Tabela 1. Pesquisa sobre hábitos de estudos e envolvimento em atividades acadêmicas durante a pandemia

Informações demográficas
Q1. Gênero; Q2. Curso; Q3. Período; Q4. Você trabalha?
Questões relacionadas a hábitos de estudos
Q5. Durante este período de isolamento, como você avalia o seu envolvimento com atividades de estudo e pesquisa em relação a antes do isolamento?
Q6**.** Durante este período de isolamento, quais os fatores que conduziram positivamente o seu envolvimento?
Q7. Durante este período de isolamento, quais os fatores que conduziram negativamente o seu envolvimento?
Q8. Durante este período de isolamento, qual a percentagem dispendida do seu tempo disponível para estudar assuntos relacionados ao seu curso?
Q9. Seus interesses de estudo se enquadram em quais áreas?
Q10. Indique o(s) assunto(s) que você andou estudando:
Q11. Quais os tipos de fontes de consulta que você mais utiliza?
Questão relacionada às fontes de consultas
Q12. Liste (copie e cole) até 5 fontes de consulta e aprendizagem mais utilizados:
Questões relacionadas às atividades acadêmicas
Q13. Durante este período de isolamento, você fez alguma atividade acadêmica?
Q14. As atividades acadêmicas estavam relacionadas a algum projeto de ensino, pesquisa, extensão e / ou eram orientadas por algum docente?
Q15. Qual o tipo do projeto?
Q16. Qual o nome do projeto e / ou do orientador?
Questões relacionadas à infraestrutura necessária ao estudo/pesquisa
Q17. Você tem acesso a qual tipo de equipamento?
Q18. Você tem acesso a qual tipo de internet?

[1]O instrumento de pesquisa pode ser acessado em: `http://bit.ly/habitosInstrumento`

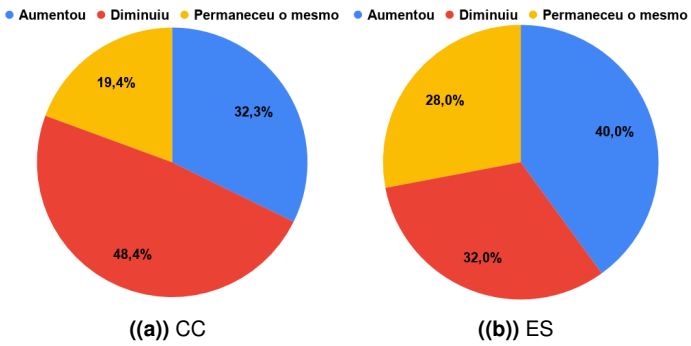
3. Resultados

Nesta seção, são apresentados os resultados obtidos a partir de 81 respostas, sendo 31 de discentes de CC e 50 de ES, representando, respectivamente, 17,7% e 29,0%, dos alunos matriculados até o período deste estudo. Os dados são apresentados por grupos de questões, tal como no instrumento de coleta.

Em relação ao perfil demográfico dos respondentes, em ambos os cursos o percentual de participantes do sexo masculino foi bastante superior ao do sexo feminino. Destaca-se também, que o maior número de respondentes está cursando entre o 1º e o 4º semestre em ambos os cursos, e que o menor índice de participação no *survey* está entre os discentes do 5º e 6º semestre. Quando perguntados se trabalham, no curso de CC, 64,5% responderam que sim e 35,5% que não. Já entre os respondentes de ES, 58% dos respondentes trabalham e 42,2% não.

Na análise referente às questões de infraestrutura necessária ao estudo/pesquisa, verificou-se que todos os respondentes possuem acesso a um ou mais aparelhos eletrônicos, sendo eles: celular, desktop/notebook e tablet. É válido ressaltar que em média 10,4% (12,9% de CC e 8,0% de ES) dos respondentes afirmaram usar tablet, quando comparado ao uso de celular ou desktop/notebook. Entretanto, quando verifica-se qual tipo de internet os respondentes utilizam, os dados mostram que em média apenas 41,3% (38,7% de CC e 44,0% de ES) dos discentes de ambos os cursos têm acesso a internet móvel. Em comparação, a banda larga mostrou ser utilizada por cerca de 99% dos respondentes, nota-se que o uso é consideravelmente maior.

Quanto aos hábitos de estudos durante a pandemia, foi possível perceber que 48,4% dos respondentes de CC tiveram o seu envolvimento com atividades de estudo e pesquisa diminuídos. Em contrapartida, 40,0% dos respondentes de ES apresentaram um aumento do envolvimento com atividades de estudo durante a pandemia, como visto na Figura 1.



((a)) CC **((b))** ES

Figura 1. Nível de envolvimento em atividades

A respeito de influências positivas nos hábitos de estudos, destacou-se o fator flexibilidade de tempo entre os discentes de ES e a liberdade de escolha dos assuntos estudados, entre os de CC. A frequência dos fatores infraestrutura da casa e proximidade com a família se sobressaíram na CC em relação à ES, como observado na Figura 2. Dentre as influências negativas, observa-se que três fatores foram mencionados por cerca de 30% a 40% dos discentes de ambos os cursos, são eles: problemas pessoais; falta de

contato com colegas e professores; e preocupação com o cenário imposto pela pandemia. Em especial, o fator falta de aulas presenciais é salientado com 61,3% de frequência na CC, como demonstrado na Figura 3.

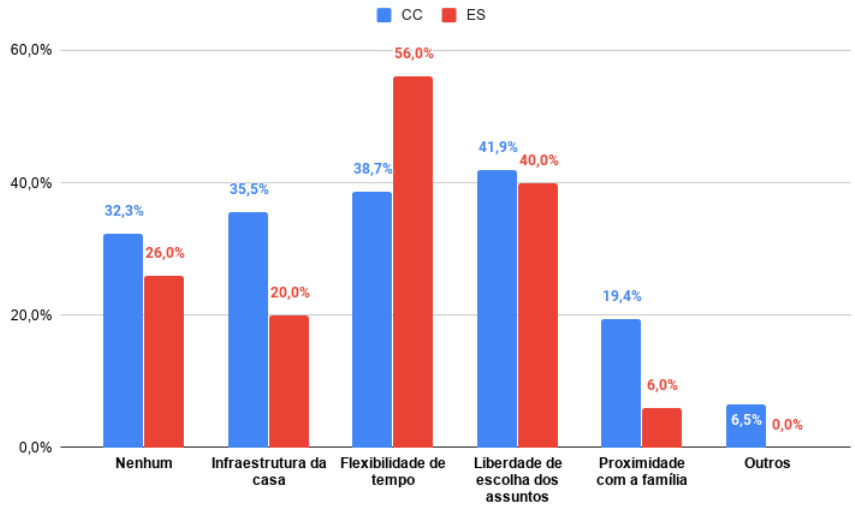


Figura 2. Influências positivas

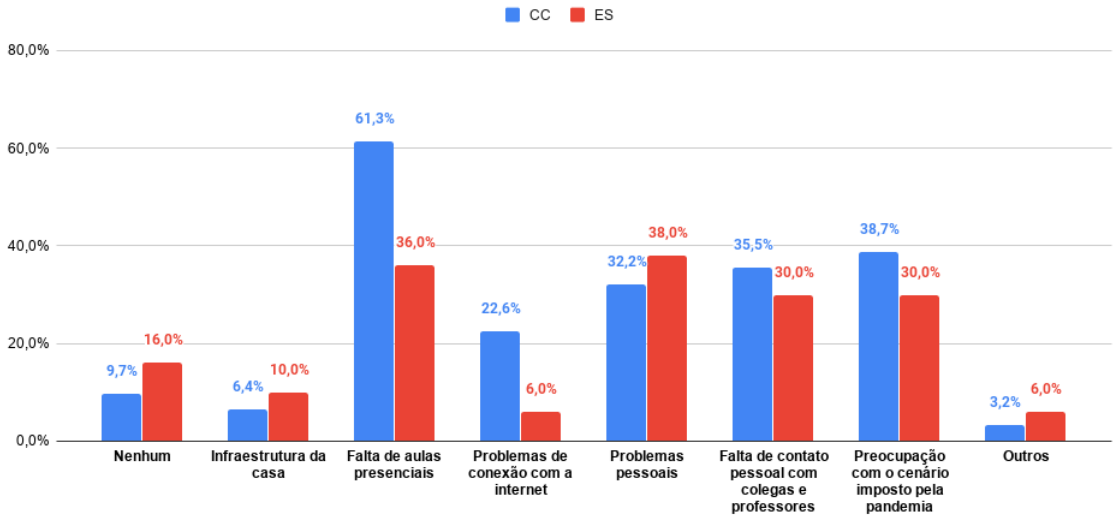


Figura 3. Influências negativas

Quando se refere ao tempo disponível dedicado a estudar, cerca de 38,7% dos respondentes de CC destinaram entre 0 a 10,0% do seu tempo para estudar assuntos relacionados ao seu curso. Por outro lado, houve uma polarização no que diz respeito aos respondentes de ES, com cerca de 20,0% destinando pouco tempo (entre 0 a 10,0%) e também aproximadamente 18,0% dedicando mais da metade do tempo. Apenas 6,4% dos participantes de CC dedicaram mais da metade do tempo para estudar, um contraste significativo em relação aos 18,0% de respondentes da ES que afirmaram o mesmo, como evidenciado na Figura 4.

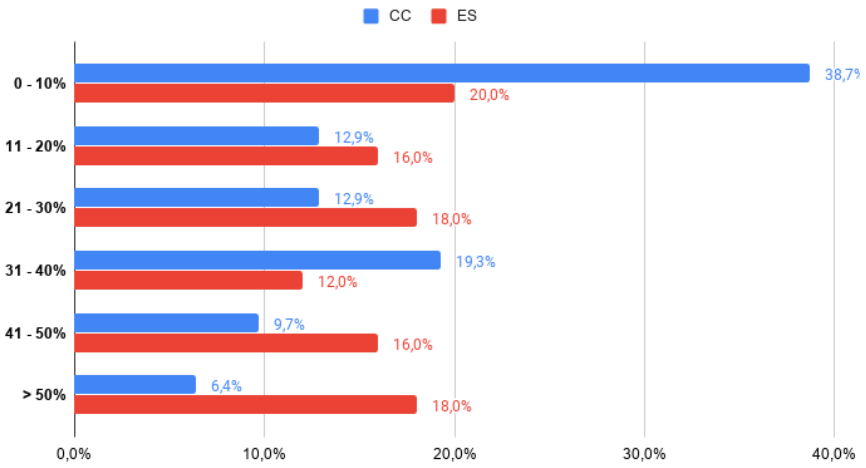
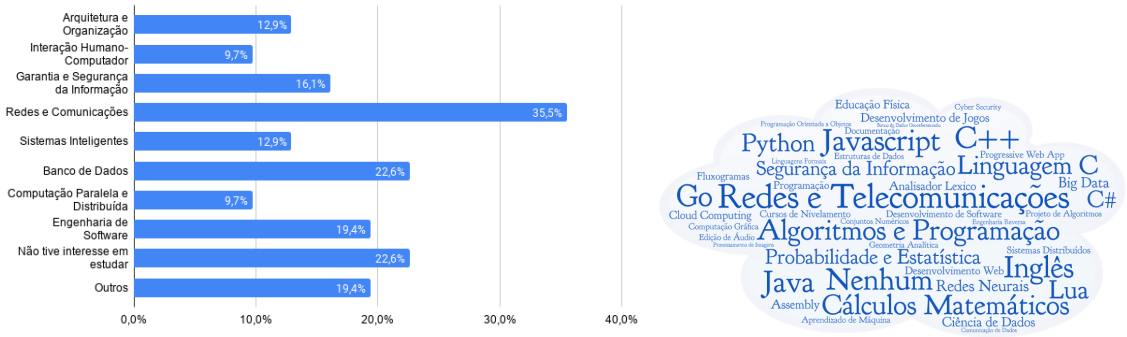


Figura 4. Percentagem de tempo disponível

Quanto às áreas de interesse de estudo, observa-se que o instrumento disponibilizava algumas áreas de conhecimento para alunos de CC e outras para alunos de ES, além de permitir a inserção de outras. Dentre aquelas consideradas pelos discentes da CC, Redes e Comunicações se sobressaiu em relação as demais com 35,5% de frequência, sendo acompanhada por Banco de Dados (22,6%) e Engenharia de Software (19,4%). Outras áreas foram apontadas como de interesse pelos de CC, como: Desenvolvimento *Web*, Inglês, *Data Science* e Lógica de Programação, além disso, cerca de 22,6% dos alunos afirmaram não ter interesse em estudar. No que diz respeito a assuntos estudados, dentre as respostas da CC, se destacam os tópicos: Redes e Telecomunicações, Javascript, Algoritmos e Programação, Python, Linguagem C e Cálculos Matemáticos. Como observado na Figura 5, outros temas relacionados à CC foram identificados, por exemplo: diversas linguagens de programação e tópicos matemáticos.

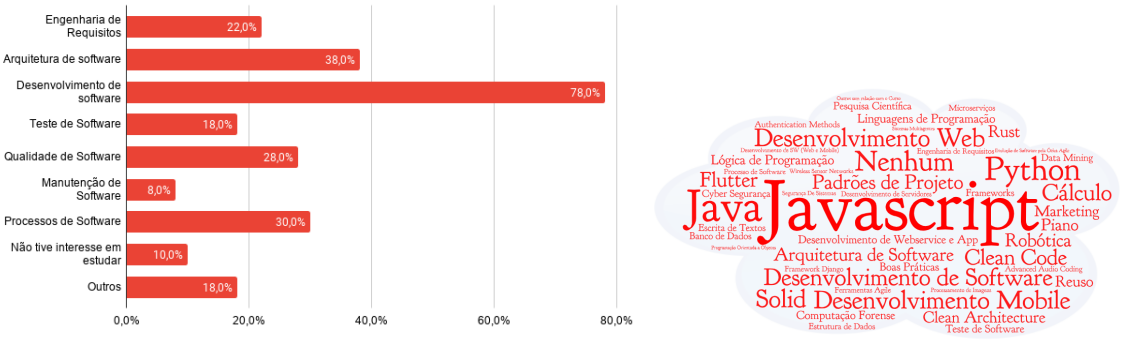


((a)) Áreas de Interesse **((b))** Assuntos

Figura 5. Áreas de interesse e assuntos da CC

Em relação às áreas de estudo escolhidas pelos participantes de ES durante o período de isolamento, Desenvolvimento de Software se sobreleva em relação as demais com 78,0%, seguido das áreas: Arquitetura de Software (38,0%) e Processos de Software (30,0%). Ademais, 18,0% declararam que não tiveram interesse em estudar. No que se refere aos tópicos estudados pelos respondentes de ES, destacam-se os temas: Javascript,

Desenvolvimento de Software, *Mobile* e *Web*, Arquitetura de Software, Python, Padrões de Projeto, dentre outros, como visto na Figura 6.



((a)) Áreas de Interesse **((b))** Assuntos

Figura 6. Áreas de interesse e assuntos da ES

Ainda em relação aos hábitos de estudos, foram analisadas as fontes de consulta[2] mais utilizadas pelos discentes. Ao total foram informadas 203 fontes de consulta, sendo 83 (48,9%) provenientes de alunos de CC e 120 (59,1%) de ES. Para fins de categorização e análise dos dados, as fontes foram agrupadas e rotuladas em: AVA (Ambiente Virtual de Aprendizagem); blog; buscador; fórum; livro; página *web* de documentação; plataforma de cursos *online*; plataforma de versionamento; rede social; repositório científico; Youtube; e outros (para quando é impossível associar um *link* ao texto informado pelo respondente).

A Figura 7 mostra o percentual de fontes de consulta utilizadas pelos respondentes de CC e ES em cada grupo. Percebe-se que o Youtube é o mais utilizado pela CC (29,1%) e aparece em segundo lugar para a ES com 20,0%. Já as plataformas de cursos *online* são mais utilizadas pela ES (25,8%). Na sequência, página *web* de documentação e blog, com o mesmo percentual, são mais utilizados pela CC. Quanto às fontes menos consultadas, tem-se rede social em ambos os cursos.

A análise das questões relacionadas às atividades acadêmicas[3] revelou que 45,2% dos respondentes de CC realizaram atividades acadêmicas durante a pandemia, contra 54,8% que não realizaram. Dentre os que realizaram atividades, 85,7% responderam que elas estavam relacionadas a algum projeto e/ou eram orientadas por algum docente e 14,3% responderam que não. Desses, 35,7% envolveram-se em projetos de ensino, 57,1% em projeto de pesquisa e 7,1% em projetos de extensão.

Já entre os respondentes de ES, 60,0% realizaram atividades acadêmicas e 40,0% não. Dentre os que realizaram atividades, 76,7% responderam que elas estavam relacionadas a algum tipo de projeto dentro da instituição e 23,3% responderam que não. Desses, 26,7% se envolveram em projetos de ensino, 60,0% de pesquisa e 13,3% de extensão.

[2]As fontes de consulta e os respectivos links podem ser acessadas em: `https://bit.ly/32dh2ki`.

[3]São consideradas atividades acadêmicas quaisquer atividades de cunho organizacional ou de geração de conhecimento, relacionada à universidade. Exemplos: organização de eventos, atividades de iniciação científica (pesquisa), dentre outros.

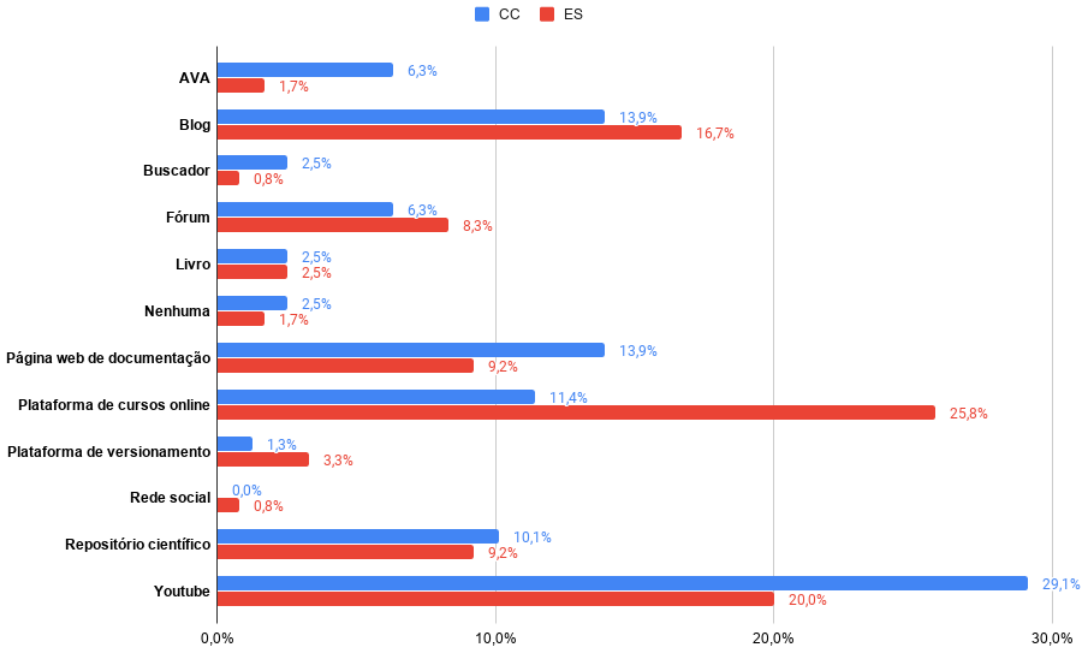


Figura 7. Fontes compiladas

4. Discussão

Em relação à infraestrutura, entende-se que o cenário de acessibilidade positivo de ambos os cursos esteja relacionado ao fato de que a grande maioria dos alunos de Computação já possui acesso a essas ferramentas, uma vez que são necessárias para a realização das atividades dos cursos. No entanto, existe a possibilidade que não reflita a realidade da maioria, uma vez que a pesquisa não atingiu a totalidade de matriculados.

A diminuição do nível de envolvimento em atividades de estudo entre os alunos de CC, pode estar relacionado com a falta de aulas presenciais, apontado como influência negativa pelos próprios. Além disso, foram os que dispenderam menos tempo aos estudos e os que menos realizaram atividades acadêmicas. Já entre os alunos de ES, o nível de envolvimento aumentou, o que, como mencionado por eles, pode ter sido influenciado pela flexibilidade de tempo. Ademais, foram os que mais dispenderam tempo aos estudos e os que realizaram mais atividades acadêmicas.

Esse cenário caracteriza alunos com perfis diferentes e pode estar relacionado à metodologia de ensino empregada nos diferentes cursos. O curso de CC é conduzido através de metodologia de ensino tradicional, onde os componentes curriculares são ministrados de uma forma essencialmente expositiva. Já no curso de ES é adotada a metodologia ativa de Aprendizagem Baseada em Problemas (ABP) com uma carga horária significativa, na qual os alunos são desafiados a resolver problemas, sendo estimulados a aprender de forma autodidata. Dessa forma, alunos de CC podem se sentir menos motivados a estudar de forma autônoma, sentindo falta das aulas presenciais, ao contrário dos alunos de ES, que, em função da ABP, se sentem mais confortáveis em fazê-lo.

Dentre os fatores, elencados por ambos os cursos, que mais influenciaram positivamente o envolvimento em atividades, aparece a flexibilidade de tempo. Entende-se que

as demandas dos cursos em tempos normais exigem uma gestão de tempo mais rigorosa por parte dos alunos, o que não foi necessário durante a pandemia, onde eles possivelmente conseguiram gerenciar melhor o seu tempo, sendo capazes de estipular uma rotina de estudos adaptada a diferentes horários e condições. Na sequência tem-se liberdade de escolha dos assuntos, que pode estar relacionada à diversidade e flexibilidade que a área de Computação oferece, o que por sua vez incentiva os estudantes a serem mais curiosos e buscarem novos conhecimentos.

Ainda em relação às influências positivas, a infraestrutura da casa e a proximidade com a família foram mais apontados pelos discentes de CC, com um significativo percentual de diferença em relação aos de ES. Entende-se que isso possa estar relacionado a um perfil de aluno que, em sala de aula, é exposto a um ambiente de aprendizagem tradicional, mais estável e controlado. Nessa linha, os discentes da CC tendem a encontrar tais características nos fatores supracitados.

Sobre as três influências negativas que impactaram o envolvimento com os estudos, apontadas de forma equilibrada por ambos os cursos, entende-se que são fatores esperados diante deste cenário de isolamento devido à pandemia, independentemente de área de conhecimento. A população em geral vem apresentando problemas de natureza pessoal, como por exemplo, problemas de convivência com a família, problemas financeiros, problemas psicológicos, dentre outros. A ausência de contato pessoal tende a prejudicar o envolvimento com os estudos. Segundo [Riess 2010] a sociabilização é um fator importante para a aprendizagem, uma vez que ela possibilita a troca de conhecimentos e a sedimentação dessa aprendizagem.

Em relação a assuntos estudados pelos respondentes de CC, predominam aqueles associados a linguagens de programação científicas, como C++, C, Lua, Python. Tal ocorrência parece estar alinhada ao perfil de alunos deste curso, que oferta em sua estrutura curricular componentes curriculares que demandam a utilização de tais linguagens. Por outro lado, na ES a área de desenvolvimento de software e os assuntos relacionados a essa área, como as linguagens para desenvolvimento de aplicações *web* e *apps*, teve um expressivo destaque. Isso também parece estar alinhado ao perfil de alunos de ES, onde a metodologia ABP, adotada nas disciplinas de Resolução de Problemas, demanda aos discentes o desenvolvimento de aplicações utilizando diferentes tecnologias.

Uma grande adesão a meios eletrônicos para consulta foi vista nos dados. A evidente preferência pelo Youtube e pelas plataformas de cursos *online* pode ser reflexo de que ambos os tipos de fonte oferecem conteúdo de uma forma semelhante à sala de aula, com uma pessoa (ou uma voz) falando enquanto apresenta algo, o que provavelmente faz com que os alunos se sintam mais familiarizados com esses meios. Mais especificamente, em relação à preferência de ES pelas plataformas de cursos *online*, entende-se que, novamente, possa estar relacionado ao perfil autodidata que os alunos apresentam em função da metodologia utilizada no curso. Da mesma forma, entende-se que a escolha de CC por páginas *web* de documentação, parece estar alinhado a um perfil de discente mais acostumado a acessar uma informação no seu estado original, sem subterfúgios.

5. Considerações Finais

As evidências indicam que alunos de Ciência da Computação e Engenharia de Software, apesar de serem da área de Computação, se comportaram de forma diferente durante

a pandemia, no que diz respeito aos hábitos de estudos e envolvimento em atividades acadêmicas. Alunos de ES tiveram maior participação na pesquisa e demostraram ter uma postura mais ativa, o que parece estar relacionado à metodologia ABP.

Ademais, o entendimento dos perfis permite que sejam tomadas decisões em nível de gestão curricular dos cursos, independentemente de período de isolamento. O mapeamento das áreas e assuntos de interesse podem auxiliar na decisão de oferta de componentes curriculares complementares ou eletivos e podem gerar pontos de reflexão em discussões sobre estruturação curricular, dentre outros.

Como ameaças à validade da pesquisa destaca-se o número da amostra que, para efeitos de generalização dos resultados, ficou aquém do esperado. É válido ressaltar que os resultados refletem majoritariamente os dados de alunos do gênero masculino, matriculados nos dois primeiros anos dos cursos.

Como trabalhos futuros, pretende-se aprimorar o *survey* para contemplar questões relacionadas à hábitos de estudos em contextos tanto de aulas remotas como de aulas presenciais, permitindo também a análise do impacto das metodologias de ensino distintas aplicadas nos dois cursos. Ademais, deseja-se investigar o processo de escolha das fontes de consultas por partes dos alunos de CC e ES e, a partir disso, propor um processo automatizado de curadoria colaborativo, onde alunos e professores possam interagir para qualificar as fontes de consulta mais relevantes.

Referências

BRASIL, M. d. E. (2020a). Portaria nº 343, de 17 de março de 2020. Disponível em: https://pesquisa.in.gov.br/imprensa/jsp/visualiza/index.jsp?data=18/03/2020jornal=515&pagina=39. Acessado em 11 de ago. de 2020.

BRASIL, M. d. S. (2020b). Sobre a doença. Disponível em: https://coronavirus.saude.gov.br/sobre-a-doenca. Acessado em 13 de ago. de 2020.

Grzybovski, D. (2005). Revisão teórica sobre pesquisa quantitativa, mensuração, amostragem e análise multivariada. *Texto para Discussão*, (13):2005.

Kasunic, M. (2005). Designing an effective survey. Technical report, Carnegie-Mellon Univ Pittsburgh PA Software Engineering Inst.

Marques, R. (2020). A ressignificação da educação e o processo de ensino e aprendizagem no contexto de pandemia da covid-19. *Boletim de Conjuntura (BOCA)*, 3(7):31–46.

Onyema, E., Nwafor, C., Faith, A., Sen, S., Atonye, F., Sharma, A., and Alsayed, A. (2020). Impact of coronavirus pandemic on education. *Journal of Education and Practice*, 11:108–121.

Riess, M. L. R. (2010). Trabalho em grupo: instrumento mediador de socialização e aprendizagem.

Van Nguyen, D., Pham, G. H., and Nguyen, D. N. (2020). Impact of the covid-19 pandemic on perceptions and behaviors of university students in vietnam. *Data in Brief*, page 105880.

RecorDS: Um Aplicativo para Extração de Diagramas UML

Douglas M. Giordano[1], **Arthur M. Becker**[1], **João P. S. da Silva**[1]

[1]Engenharia de Software – Universidade Federal do Pampa (UNIPAMPA)
97.546-550 – Alegrete – RS – Brazil

douglasmontanhagiordano@gmail.com

arthmalbeck@gmail.com, joaosilva@unipampa.edu.br

Abstract. *Teams that use agile modeling practices often use whiteboards, as they offer a dynamic and collaborative modeling environment. The problem occurs in the generation of software artifacts, due to the fact that normally the sketches made on whiteboards have to be recreated in a UML editor or else they are stored in images, causing the documentation to be out of date. Thus, in order to minimize this integration between sketches made in frames and UML editors, this article has the bias of presenting an application for recognizing sketches of class diagrams. For the purposes of developing this application, image processing and analysis techniques are used.*

Resumo. *Equipes que utilizam práticas de modelagem ágil muitas vezes fazem uso de quadros brancos, por oferecerem um ambiente dinâmico e colaborativo de modelagem. O problema ocorre na geração dos artefatos de software, devido ao fato de que normalmente os esboços feitos em quadros brancos são recriados em um editor UML ou então armazenados em imagens, ocasionando uma desatualização da documentação. Assim, com o objetivo de minimizar essa integração entre esboços feitos em quadros e editores UML, esse artigo tem o viés de apresentar um aplicativo de reconhecimento de esboços de diagramas de classe. Para fins de desenvolvimento desse aplicativo, técnicas de processamento e análise de imagens são utilizados.*

1. Introdução

Os métodos ágeis são adotados por muitas empresas de software durante o processo de desenvolvimento, sendo esses normalmente interativos, evolutivos, adaptativos e incrementais. Os métodos ágeis, segundo alguns autores, são flexíveis em relação aos métodos tradicionais. A modelagem ágil é uma prática ágil comumente usada por empresas para modelar o software, auxiliando na comunicação da equipe e geralmente utilizando a UML (*Unified Modeling Language*).

Dessa forma, equipes de desenvolvimento que seguem tais práticas ágeis, frequentemente recorrem a quadros brancos para a modelagem de diagramas. Devido ao fato de os quadros brancos não imporem regras [Costagliola et al. 2014] na hora de desenhar o modelo.

Embora a modelagem em quadros brancos seja simples e fácil, ela apresenta alguns problemas em relação à integração dos artefatos de software e a sua documentação em imagens [Wüest et al. 2013]. Tais problemas podem ser minimizados por meio de uma ferramenta que viabilize a integração do esboço com um editor UML.

Nesse sentido, esse presente trabalho tem como objetivo apresentar uma ferramenta para a extração de esboços de diagramas de classe desenhados em quadros brancos. A ferramenta, vai possibilitar a integração entre esboços feitos em quadros brancos e editores UML, viabilizar o reconhecimento de diagramas de classe em uma perspectiva conceitual, possibilitar a utilização de dispositivos móveis para capturar e reconhecer a imagem e gerar arquivos no formato XMI (*XML Metadata Interchange*) dos diagramas reconhecidos.

2. Fundamentação Teórica

Nessa seção será abordado os conceitos relacionados com os sistemas de reconhecimento de esboços e modelagem de software. Na seção 2.1 é apresentada a UML, uma linguagem de modelagem de software. Na seção 2.2 é descrito os principais conceitos relacionados ao processamento de imagens, responsável pela melhoria ou retirada de características indesejadas em imagens. Por fim, na seção 2.3 apresentamos a análise de imagens, que visa a extração e compreensão do conteúdo de uma imagem.

2.1. Linguagem Unificada de Modelagem

A UML (Linguagem Unificada de Modelagem) é uma linguagem de modelagem de software padrão que pode ser empregada na visualização, especificação, construção e documentação de artefatos de software [Booch et al. 2000].

A UML possui três tipos de blocos de construção denominados de itens, relacionamentos e diagramas. Os itens podem ser divididos em estruturais, comportamentais, de agrupamento e notacionais. Os diagramas da UML 2.5 podem ser divididos em duas categorias, os diagramas estruturais que tem como objetivo mostrar o sistema de uma perspectiva estática, e os diagramas comportamentais que tem como objetivo mostrar o sistema em uma perspectiva mais dinâmica [OMG 2015].

2.2. Processamento de Imagens Digitais

O processamento de imagens consiste em um conjunto de técnicas para capturar, representar e transformar imagens com auxílio de computador [Pedrini and Schwartz 2008]. Podemos definir uma imagem digital como uma função bidimensional onde em cada par de coordenadas (x,y) possui um valor representando a densidade, também chamado nível de cinza. Esse par de coordenada é o Pixel.

O sistema de registro é um dispositivo utilizado para impressão ou apresentação da imagem. O hardware especializado em processamento de imagens normalmente consiste em um digitalizador (conversor de estímulos elétricos em dados digitais) e um hardware para realizar operações que requerem um rápido processamento. O Software de processamento de imagens é um conjunto de módulos especializados para a realização de tarefas específicas. Os sensores são os dispositivos responsáveis pela aquisição da imagem. Por último temos a rede de comunicação que faz a transferência de dados entre os componentes.

Segundo [Gonzalez and Woods 2010] alguns passos são fundamentais no processamento digital de imagens. Podemos considerar como métodos de processamento de imagem a aquisição da imagem, realce de imagens, restauração de imagens, processamento de imagens coloridas e a compreensão de imagens. Os métodos de processa-

mento morfológico, segmentação, representação, descrição e reconhecimento de objetos são métodos de análise de imagens.

2.3. Análise de Imagens Digitais

A análise de imagens é, tradicionalmente, baseada na forma, textura, níveis de cinza ou nas cores dos objetos presentes em uma imagem. A análise de imagens tem como foco compreender o conteúdo da imagem. Algumas técnicas comuns na análise da imagem são a segmentação da imagem em regiões ou objetos de interesse, descrição dos objetos e reconhecimento ou classificação dos mesmos [Pedrini and Schwartz 2008].

A segmentação e suas técnicas buscam detectar descontinuidades e similaridades da imagem, assim sendo um processo importante para identificação de objetos. A imagem é subdividida em regiões ou objetos de interesse. A segmentação é uma das tarefas mais difíceis em um sistema de análise ou processamento de imagens onde a precisão da segmentação determina o sucesso, ou o fracasso final de um procedimento [Gonzalez and Woods 2010]. A divisão das regiões e objetos também depende do processamento da imagem onde, por exemplo, imagens com muito ruído podem distorcer a extração dos dados da imagem [Pedrini and Schwartz 2008].

As regiões e objetos gerados da segmentação necessitam ser representados e descritos para os próximos passos da análise da imagem. O objeto pode ser representado em termos de suas características externas (bordas) e características internas (pixels). A descrição depende da representação escolhida e é necessário a extração de características e medidas mínimas não permitindo ambiguidades.

A morfologia matemática é uma ferramenta para extrair componentes das imagens que são úteis na representação e na descrição da forma de uma região [Gonzalez and Woods 2010].

A classificação de padrões ou também chamada de reconhecimento de objetos visa mapear amostras extraídas com um conjunto de rótulos possuindo algumas restrições para mapear amostras com características semelhantes [Pedrini and Schwartz 2008].

3. Trabalhos Relacionados

Nessa seção será apresentada as questões e seus resultados obtidos por meio do mapeamento sistemático para investigar o estado da arte dos trabalhos relacionados aos sistemas de reconhecimento de diagramas UML ou estruturas semelhantes.

3.1. Quais técnicas de aquisição de imagem foram utilizadas?

No domínio de reconhecimento de fluxograma [Awal et al. 2011] utilizam telas sensíveis ao toque para aquisição de imagem. Outra ferramenta com proposta parecida é a FlexiSketch criada por [Wüest et al. 2013], tem como objetivo simular um quadro branco e receber esboços de diagramas UML desenhados diretamente na ferramenta.

[Buchmann 2012] propõe uma simulação de um quadro branco, oferecendo liberdade ao desenvolvedor. A aquisição da imagem é feito pelo desenho do desenvolvedor na tela do software. Uma ferramenta de modelagem denominada GMF (*Graphical Modeling Framework*) possibilitou ao usuário desenhar os esboços na própria tela do ambiente de desenvolvimento. Esse tipo de aquisição de imagens facilita a modelagem de diagramas em telas sensíveis ao toque, sem perder a liberdade de um quadro branco.

3.2. Quais filtros foram utilizados?

Todos os trabalhos relacionados encontrados recebiam como entrada um desenho feito pelo usuário diretamente no sistema. Logo, é eliminado grande parte do ruído ou qualquer outro tipo de distorção na imagem gerada muitas vezes pelo ambiente. Levando em consideração esse fator, poucos filtros foram utilizados nos artigos encontrados.

3.3. Quais técnicas de segmentação de imagem foram utilizadas?

A segmentação também foi pouco utilizada nos artigos encontrados, pois a maior parte dos desenhos eram analisados de forma isolada logo após o usuário desenhar. Contudo, foram encontradas algumas técnicas para separação do diagrama e do texto.

[Awal et al. 2011] tem como um dos principais problemas a separação do texto e dos gráficos, pois cada um deles é tratado de forma diferente, então para separá-los é apresentado uma técnica para medir a complexidade chamada de entropia. Por meio da entropia o grau de desordem é medido utilizando ângulo de pontos. Isso possibilita verificar a complexidade dos elementos, como o texto possui linhas mais complexas é possível separar a partir desta técnica. Outro método também utilizado para separar o diagrama do texto é a binarização linear absoluta que [Maggiori et al. 2014] aplicou com outras técnicas nos pixeis.

[Jiang et al. 2011] utilizaram algoritmo de partição de gráficos para separar em vários subgrafos cada mapa conceitual. A partir destes subgrafos é utilizada programação dinâmica para extrair os blocos de conceitos.

3.4. Quais técnicas de reconhecimento de padrões foram utilizadas?

Para o reconhecimento de esboço em um domínio geral [Liao and Duan 2012] utilizam uma sequência de caracteres que representam cada primitiva desenhada pelo usuário. São recebidos duas sequências de caracteres e é calculado a distância entre as mesmas utilizando métodos Kernel.

A falta de naturalidade das ferramentas CASE (*Computer-Aided Software Engineering*) e o retrabalho de transferir tais documentos para o computador levou [Costagliola et al. 2014] a apresentar uma ferramenta de reconhecimento de diagramas. A Metodologia foi dividida 4 em camadas: camada de separação de texto e gráficos, camada do reconhecimento de símbolos, camada de detecção do contexto local e por fim a camada de detecção do diagrama. O reconhecimento é feito em contexto local utilizando aprendizado de máquina.

Também em um domínio de reconhecimento de diagramas [Maggiori et al. 2014] utilizaram OpenCV para reconhecimento de diagramas arquiteturais. Para a detecção das relações foi utilizado Transformada de Hough e uma série de outras operações em cima dos pixeis da imagem. Também foi utilizado algoritmos de aprendizado de máquina para verificar a forma no final da linha que representa a relação. Os textos foram reconhecidos utilizando um OCR de código aberto chamado de Tesseract.

4. RecorDS: Recognizer Diagram Sketch

Nesta seção apresentamos o aplicativo denominado RecorDS (Recognizer Diagram Sketch). Na subseção 4.1 abordamos uma visão geral do trabalho, retomando os pro-

blemas e soluções ao qual motivarão a criação da ferramenta, além dos requisitos encontrados. Já na subseção 4.2 é tratado os detalhes da implementação do sistema.

4.1. Funcionalidades Gerais

A maioria dos editores UML não possibilitam a edição de diagramas UML por vários membros da equipe ao mesmo tempo, fazendo com que equipes prefiram quadros brancos [Wüest et al. 2013]. Assim, alguns problemas são encontrados na modelagem de software colaborativa utilizando um quadro branco ou mesmo papel e caneta. O primeiro é o retrabalho ocasionado pela reconstrução do diagrama manualmente em uma ferramenta CASE. Outro problema é que como muitas vezes a documentação é uma imagem não tem como gerar o código automaticamente.

Analisando os problemas descritos, é possível propor um produto que consiga minimizar o retrabalho de reconstruir todo diagrama. Uma ferramenta que capture a imagem com um aplicativo e possa extrair diagramas UML a partir da imagem, logo após gerar um arquivo para ser importado em um editor UML, resolveria o retrabalho de integrar o esboço de um diagrama com uma ferramenta CASE, consequentemente minimizando o trabalho das equipes.

A partir dos problemas e soluções encontramos os requisitos necessários ao sistema. Organizamos os requisitos por intermédio de uma lista apresentada abaixo.

- Detecção e reconhecimento de classes em um diagrama de classes.
- Detecção e reconhecimento de associações em um diagrama de classes.
- Detecção e reconhecimento de multiplicidades em um diagrama de classes.
- Reconhecimento de caracteres do diagrama de classes
- Geração do diagrama de classes no formato XMI seguindo a especificação UML.
- Utilização de um dispositivo Android para capturar e reconhecer o diagrama de classes.

4.2. Detalhes da Implementação

O processo de reconhecimento dos elemento para por um conjunto de passos. Utilizando um diagrama de atividades na Figura 1 mostramos cada passo necessário para reconhecer o diagrama. Adquirir a imagem por meio da câmera de um dispositivo Android é o primeiro passo. Após a imagem capturada pelo dispositivo, é realizada a detecção e reconhecimento dos elementos, na qual abrangem as seguintes etapas: extrair elementos, classificar elementos, segmentar nome da classe, interpretar nome da classe, interpretar conexão das relações, interpretar conexão das multiplicidades, interpretar texto da multiplicidade. Após o reconhecimento é gerado o arquivo XMI para ser utilizado em alguma ferramenta de edição de modelos UML.

4.2.1. Detecção dos Elementos

A detecção dos elementos é feita em duas etapas. A primeira etapa ocorre em tempo real quando o usuário abre o aplicativo. Neste momento nossa ferramenta processa cada imagem por segundo e localiza os elementos, como se pode observar na Figura 2. O Segundo passo é o momento de extração dos elementos, onde além de serem detectados os mesmos são classificados e interpretados. Os elementos das classes são extraídos por

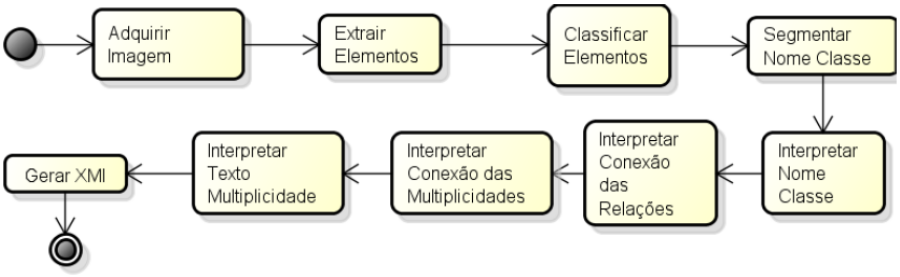


Figura 1. Processo de reconhecimento.

meio de uma série de processos bastante simples. Primeiramente captura-se a imagem por meio de um toque na tela no smartphone ou tablet. Este toque na tela inicia o processo de reconhecimento, pausando a detecção em tempo real e iniciando o reconhecimento completo da imagem.

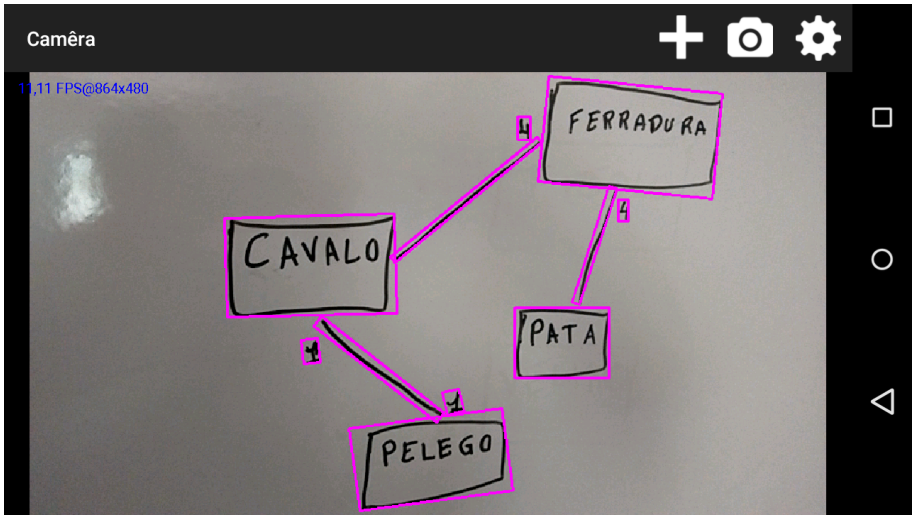


Figura 2. Detecção dos elementos em tempo real.

A imagem capturada é processada de modo a transformá-la para uma escala de cinza e facilitar a extração de características. Com a imagem em escala de cinza, utilizamos um operador de morfologia matemática denominada de *Morphological Gradient* que é a diferença entre a erosão e dilatação da imagem.

Com as bordas da imagem tratadas, utilizamos uma técnica para segmentação de bordas denominada de Limiarização (*Threshold*).

Com a imagem segmentada, utilizando o método "findContours" do Open CV todos os contornos são procurados. Extraímos apenas os contornos externos primeiramente, com o objetivo de saber onde está localizado cada elemento. Nesse momento ainda não se faz necessário pegar outras informações, como as letras da classe.

4.2.2. Reconhecimento de Classe

O reconhecimento de classe passa por duas etapas denominadas de classificação e de interpretação de nome. Na qual na primeira etapa vários elementos são extraídos da ima-

gem. Um elemento pode ser considerado uma classe quando o mesmo possuir o tamanho da área maior que 1000 pixeis e também possuir um nome.

Na segunda etapa o nome da classe é interpretado. Primeiramente é necessário pegar a imagem contendo apenas a classe. A partir dessa imagem utilizamos o método de limiarização para segmentar as bordas da imagem. Como se tem a conveniência de fazer o reconhecimento apenas do nome da classe e não de toda estrutura, teve-se que separar as bordas da classe do nome, para isso é utilizado a diferença de matrizes. Pegamos as bordas externas da classe e depois todas as bordas e subtraímos os valores, sobrando apenas o nome da classe.

Com o nome da classe já tratado, levamos a imagem para o reconhecimento por meio do OCR (*Optical Character Recognition*) chamado de Tesseract. Como estamos trabalhando e operando em um ambiente fechado, o OCR está treinado apenas para reconhecer letras de forma.

4.2.3. Reconhecimento das Associações

Atualmente nosso software suporta o reconhecimento de associações. As associações simples são diferenciadas dos demais elementos pela diferença entre sua largura e altura. O elemento é classificado como uma relação, caso a altura for cinco vezes maior que a largura ou vice-versa.

Depois de encontrada as relações do diagrama, deve-se verificar a quais classes elas estão relacionadas. As relações não devem encostar nas classes por que existe uma limitação no aplicativo (não foi encontrada nenhuma técnica viável para separação das linhas), ou seja, devem apenas estar perto. Por meio de um cálculo de proximidade, utilizando as extremidades da relação, é possível verificar quais classes estão conectadas.

Utilizando uma fórmula derivada do Teorema de Pitágoras, mostrada na Figura 3, medimos a distância entre o centro da classe e a extremidade da associação.

$$\sqrt{ditancia = (pontoBX - pontoAX)^2 + (pontoBY - pontoAY)^2}$$

Figura 3. Fórmula derivada do Teorema de Pitágoras

A melhor solução é utilizar a média de tamanho da classe, considerando altura e largura. Então pegamos o resultado da distância e verificamos com a média de tamanho da classe. Caso a distância for menor que a média de tamanho da classe, a associação está conectada.

4.2.4. Reconhecimento de Multiplicidade

São definidos como multiplicidade os elementos com área maior que 100 pixeis e menor que 1000 pixeis. As multiplicidades são ligeiramente menores que as classes, então nesse contexto foi suficiente para diferencia-las.

O cálculo de proximidade é o mesmo utilizado nas relações, com a diferença que nesse contexto as multiplicidades ficam mais próximas das relações. Então, apenas é ve-

rificado se cada multiplicidade está próxima da sua respectiva extremidade da associação. Com a proximidade verificada, os caracteres são reconhecidos com o OCR.

5. Verificação e Validação da Solução

Com o propósito de identificar se o software possui defeitos e está de acordo com o especificado, elaboraram-se alguns testes. Para a verificação da solução e a realização desses testes capturamos 10 diagramas de classe com seus respectivos XMI, a captura da imagem foi realizada por meio do smartphone Moto G2 com a resolução minima de 864 x 480. Os diagramas testados estão inseridos em um ambiente controlado no qual deveriam seguir as seguintes regras:

- As associações não devem encostar nas classes.
- As multiplicidades não devem encostar nas relações.
- As classes devem estar fechadas (aberturas mínimas são retiradas por meio da morfologia matemática, porém aberturas maiores não).
- O nome da classe deve ser escrito em letra de forma o mais legível possível.
- O nome da classe não devem encostar nas bordas da classe.

Comparamos cada classe com o seu respectivo arquivo XMI. Com os valores de reconhecimento fizemos uma média geral conseguindo como resultado a % de acerto na detecção e reconhecimento, como se pode observar na Tabela 1.

Tabela 1. Resultado dos testes

Elemento	**% Detecção**	**% Reconhecimento**
Classes	100%	80%
Associações	100%	90%
Multiplicidades	70%	30%

De modo a validar a ferramenta, foi marcado um horário onde se contou com a participação de 4 acadêmicos do curso de engenharia de software (7 Semestre e 8 Semestre), esses alunos possuem em sua grade curricular, disciplinas que fazem uso da UML. Cada aluno efetuou a instalação da ferramenta em seu dispositivo móvel, em seguida, desenhou alguns diagramas. Por fim, foi repassado um formulário para preenchimento, compondo-se por perguntas relacionadas a facilidade de instalação, facilidade de uso, tolerância a falhas, relevância da ferramenta e qual seria o próximo diagrama mais relevante para ser reconhecido pela ferramenta.

Vale ressaltar que a criação do diagrama de classe efetuado por cada graduando respeitou algumas regras de um ambiente controlado de validação de uso, limitando-se: criação de classes, multiplicidades, relações de associação simples não direcionais, elementos desenhados respeitando uma pequena distância de cada outro elemento, o uso de letras garrafais e o uso em ambientes com poucas sombras. Assim sendo, essa é uma validação do funcionamento do todo e não podendo ser usada para validar a assertividade do reconhecimento.

Na Figura 4 encontram-se as respostas dos alunos, onde se pode observar grande interesse por parte dos mesmos na ferramenta. Outro fator foi que alguns tiveram certas dificuldades na instalação, pois necessita-se baixar o Open CV inicialmente antes da

utilização do aplicativo. Ressalta-se também que, encontraram-se problemas com a conexão das relações, mas problema esse que já foi resolvido na versão atual do software.

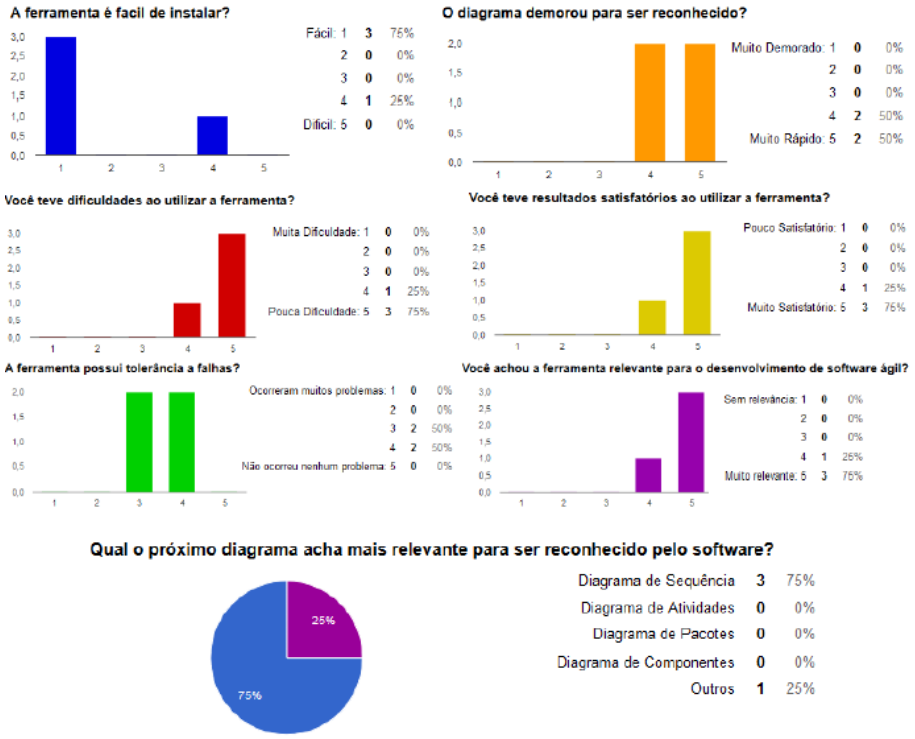


Figura 4. Respostas dos alunos que participaram da validação.

6. Conclusão

O principal objetivo do nosso trabalho foi criar uma ferramenta para extrair esboços de diagramas de classe da UML. Esta ferramenta busca minimizar o trabalho de equipes de software, no qual normalmente necessitam reconstruir os diagramas feitos em quadros brancos em uma ferramenta CASE apenas para geração do código ou atualização da arquitetura do software.

A construção da ferramenta de reconhecimento de esboço passou por alguns desafios. O primeiro desafio foi limitar o uso da ferramenta, no qual seu funcionamento encontra-se limitado a um ambiente controlado, tratando apenas de elementos simples de um diagrama de classes da UML. O segundo é a grande variação do esboço. Cada pessoa possui uma forma de desenhar ou escrever, o que dificulta o reconhecimento, porém não tornou a tarefa impossível.

A ferramenta causa um impacto muito grande em quem a usa (em ambiente controlado), pois apesar do ambiente controlado, ela automatiza um processo que muitas vezes é demorado e causa retrabalho. Outro ponto é que apenas com um smartphone ou tablet é possível reconhecer um diagrama e enviar direto para a nuvem, sem gerar retrabalhos.

Os trabalhos futuros em cima da ferramenta serão focados na utilização de inteligência artificial para realizar a segmentação dos elementos do quadro branco e a

classificação dos elementos do esboço. Logo, será possível aumentar o número de diagramas UML reconhecidos pela ferramenta, extrapolando o reconhecimento apenas do diagrama de classes.

Referências

Awal, A.-M., Feng, G., Mouchere, H., and Viard-Gaudin, C. (2011). First experiments on a new online handwritten flowchart database. In *IS&T/SPIE Electronic Imaging*, pages 78740A–78740A. International Society for Optics and Photonics.

Booch, G., Rumbaugh, J., and Jacobson, I. (2000). *UML-GUIA DO USUARIO: TRADUÇAO DA SEGUNDA EDIÇAO*. Elsevier Brasil.

Buchmann, T. (2012). Towards tool support for agile modeling: sketching equals modeling. In *Proceedings of the 2012 Extreme Modeling Workshop*, pages 9–14. ACM.

Costagliola, G., De Rosa, M., and Fuccella, V. (2014). Local context-based recognition of sketched diagrams. *Journal of Visual Languages & Computing*, 25(6):955–962.

Gonzalez, R. C. and Woods, R. E. (2010). Processamento digital de imagens. traduçao: Cristina yamagami e leonardo piamonte.

Jiang, Y., Tian, F., Zhang, X. L., Dai, G., and Wang, H. (2011). Understanding, manipulating and searching hand-drawn concept maps. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 3(1):11.

Liao, S. and Duan, M. (2012). Sketch recognition via string kernel. In *Natural Computation (ICNC), 2012 Eighth International Conference on*, pages 101–105. IEEE.

Maggiori, E., Gervasoni, L., Antúnez, M., Rago, A., and Díaz Pace, J. A. (2014). Towards recovering architectural information from images of architectural diagrams. In *XLIII Jornadas Argentinas de Informática e Investigación Operativa (43JAIIO)-XV Simposio Argentino de Ingeniería de Software (Buenos Aires, 2014)*.

OMG (2015). Omg unified modeling language.

Pedrini, H. and Schwartz, W. R. (2008). *Análise de imagens digitais: princípios, algoritmos e aplicações*. Thomson Learning.

Wüest, D., Seyff, N., and Glinz, M. (2013). Flexisketch: A mobile sketching tool for software modeling. In *Mobile Computing, Applications, and Services*, pages 225–244. Springer.

Estimativa de Esforço em Atividades de Manutenção de Software: Um Mapeamento Sistemático

Kaliane L. Viesseli[1], **Arielyn P. Silva**[1], **Gustavo Santos**[1]

[1]COENS – Universidade Tecnológica Federal do Paraná (UTFPR)
Estrada para Boa Esperança, km. 04 – Zona Rural, Dois Vizinhos - PR, 85660-000

{kaliane, arielyn}@alunos.utfpr.edu.br, gustavosantos@utfpr.edu.br

Abstract. *During the software lifecycle, good effort estimation allows teams to make decisions on how development activities will proceed, on the feasibility of new changes, and whether the project will be delivered according to deadlines. In this paper, we describe a systematic mapping to identify evidence with respect to metrics, approaches and frameworks for calculating effort estimation during the software maintenance phase. We analyzed 521 studies and we selected 17 primary studies; most approaches use metrics and historical data do estimate effort for new activities. However, only one study proposed a framework, which emphasizes the importance of proposing tools to calculate effort estimation for maintenance activities.*

Resumo. *Durante o ciclo de vida do software, o correto levantamento de estimativas de esforço permite que a equipe tome decisões sobre como será o andamento das atividades de desenvolvimento e manutenção, qual a viabilidade de novas alterações, e se estas serão entregues de acordo com os prazos. Neste artigo, é apresentado um mapeamento sistemático com o objetivo de identificar evidências na literatura com relação a métricas e abordagens para o cálculo de estimativa de esforço durante a fase de manutenção de software. Foram analisados 521 estudos e selecionados 17 estudos primários; a maioria das abordagens utiliza de métricas ou dados históricos para estimativa de novas atividades, e apenas um framework foi proposto, o que ressalta a importância de gerar novas ferramentas que facilitem a geração das novas estimativas.*

1. Introdução

A manutenção de software abrange os processos de modificação de um sistema após a sua entrega, sejam estes para correções de falhas, melhorias de desempenho ou adaptações do sistema para um ambiente diferente. A manutenção é considerada a fase mais custosa do ciclo de vida de um software, atingindo até 80% dos custos totais de um projeto, e cerca de 75% dos custos são dedicados em melhorias [Erlikh 2000].

As atividades de manutenção em maioria são reativas; isto é, erros são identificados no sistema em uso, melhorias não planejadas no início do projeto são requisitadas pelos clientes, e mudanças na plataforma exigem atualizações no sistema. Independente da atividade, as empresas demandarão esforço e tempo para realizar as alterações necessárias, considerando o estado atual das equipes, a complexidade da atividade, o nível de experiência com o sistema, entre outros fatores. Para isso, as estimativas são responsáveis por determinar quanto de esforço será empregado, qual o custo, qual será o cronograma e o escopo do projeto [Vazquez et al. 2013].

O presente trabalho abordará em específico a estimativa de esforço, a qual almeja identificar qual o tempo necessário para a conclusão de uma atividade. Neste sentido, existem muitos fatores que podem dificultar na geração de uma estimativa mais precisa, como por exemplo requisitos imprecisos e mal detalhados, falta de conhecimento dos estimadores e nova atividade muito diferente do que já foi realizado no projeto, não sendo possível utilizar atividades semelhantes para realizar uma estimativa por analogia.

O objetivo deste trabalho é identificar as abordagens propostas na literatura sobre estimativa de esforço, sumarizar as características principais dos trabalhos encontrados e, consequentemente, situar a atual pesquisa e sugerir novas áreas de investigação para futuras pesquisas. Para realização desse objetivo, foi conduzido um Mapeamento Sistemático (MS) [Kitchenham 2004] em busca de trabalhos relevantes publicados em conferências e periódicos de impacto.

O restante do artigo está organizado da seguinte forma: a Seção 2 descreve o protocolo de pesquisa e condução do MS. A Seção 3 apresenta os resultados e lições aprendidas com o estudo. A Seção 4 discute limitações e ameaças à validade. E, por fim, a Seção 5 apresenta as considerações finais.

2. Mapeamento Sistemático

Com o objetivo de identificar métodos, métricas ou ferramentas de estimativa de esforço utilizados durante a manutenção de software, foi realizado um MS, o qual visa identificar, interpretar e avaliar todas as pesquisas pertinentes à questão de pesquisa definida [Kitchenham 2004]. Este MS foi realizado seguindo quatro passos: (i) definição da questão de pesquisa (QP), (ii) pesquisa dos estudos primários relacionados ao tema, (iii) triagem desses estudos e (iv) extração de dados. Para a condução deste MS, a seguinte QP foi definida:

- **QP:** Quais métodos, métricas ou ferramentas de estimativa de esforço têm sido aplicadas à manutenção de software?

Esta QP visa identificar diferentes abordagens para estimativa de esforço, especificamente aplicadas para apoiar atividades referentes à manutenção de software. Durante esta fase, a complexidade do sistema tende a aumentar, a qualidade do código decai e novas alterações irão demandar mais esforço para compreensão do estado atual do sistema. Consequentemente, estimativas realizadas com base nas primeiras versões do projeto não refletem o real esforço para manutenção.

A partir da QP, foram identificadas as palavras chaves relacionadas ao tema, mais especificamente os termos "estimativa de esforço" e "manutenção de software". A *string* de busca foi obtida pela combinação dos sinônimos em inglês destes termos, obtendo então a seguinte *string*: (***"effort estimation" AND "software maintenance"***).

Posteriormente, quatro bases de dados foram selecionadas para extração dos estudos primários, sendo elas: *ACM Digital Library*, *IEEE Xplore*, *Scopus* e *Springer*.[1]. Por meio da *string* de busca, são buscados automaticamente estudos em conferências e periódicos de impacto na área.

[1] `http://dl.acm.org/` — `http://ieeexplore.ieee.org/` — `http://scopus.com/` — `http://link.springer.com/`

Para a triagem dos estudos obtidos na etapa anterior, foram definidos critérios de inclusão e exclusão. Ambos são importantes para classificar os estudos mais relevantes e que possam contribuir com a QP deste trabalho. Os **Critérios de Inclusão (CI)** agregam: **CI_1**: estudos primários que abordem alguma metodologia ou ferramenta propondo estimativas de esforço voltadas para a área de manutenção de software; e **CI_2**: estudos primários em que a estimativa de esforço seja voltada a tarefas ou a pequenos projetos.

Quanto aos **Critérios de Exclusão (CE)**, são definidos a seguir: **CE_1**: estudos primários que não sejam *full paper* ou *short paper*; **CE_2**: estudos primários que estejam escritos em outro idioma que não seja Inglês ou Português; **CE_3**: estudos primários incompletos e que não possuem avaliação dos métodos; **CE_4**: estudos primários indisponíveis para *download*; **CE_5**: estudos primários que são versões anteriores de um estudo mais completo sobre a mesma investigação.

A condução deste MS é apresentado na Figura 1. As buscas automáticas nas bases de pesquisa foram realizadas no período de outubro a novembro de 2019.

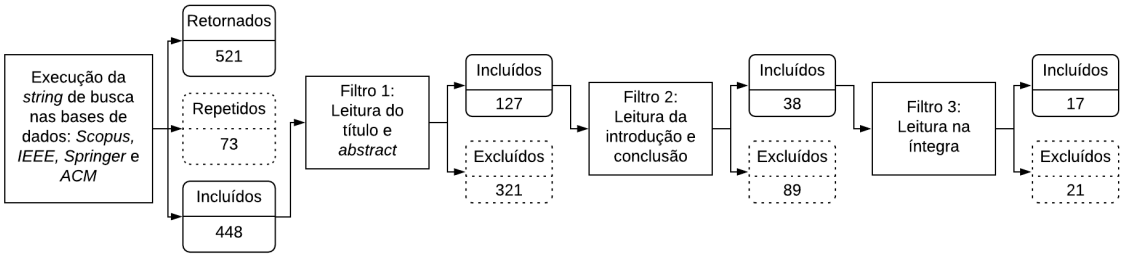


Figura 1. Processo de condução do MS

Após a execução da *string* de busca nas bases selecionadas, foram retornados 521 estudos, sendo 19 estudos na *ACM*, 204 no *IEEE*, 95 no *Scopus*, e 203 no *Springer*. A partir dos resultados obtidos, identificou-se 73 artigos repetidos, os quais foram excluídos do mapeamento, assim restando 448 estudos a serem analisados. Na próxima fase, foram aplicados os critérios de inclusão e exclusão a partir da leitura do título e resumo dos estudos selecionados, sendo incluídos 127 estudos e excluídos 321 estudos. Em seguida, os mesmos critérios foram aplicados, desta vez a partir da leitura da introdução e conclusão dos estudos; foram incluídos 38 estudos e excluídos 89 estudos nesta fase. Por fim, o último filtro foi baseado na leitura integral dos estudos, dos quais 17 estudos foram incluídos e 21 foram excluídos.

É importante destacar que, para melhor compreensão do contexto de cada estudo primário, foram criadas três categorias para classificação desses estudos conforme os objetivos da QP. As categorias são:

- **C_1 – Métricas:** reúne estudos que sugerem o uso de métricas, *e.g.*, linhas de código e pontos de função, ou fórmulas para o cálculo da estimativa de esforço;
- **C_2 – Métodos:** reúne estudos que propõem um método, técnica ou abordagem para o cálculo da estimativa de esforço;
- **C_3 – Ferramentas:** reúne estudos que apresentam uma ferramenta, protótipo, sistema, software, ou *framework* para o cálculo de estimativa de esforço.

Após a leitura na íntegra, foi possível associar alguns estudos a mais de uma categoria. Os conjuntos de categorias observadas foram: C_1 e C_2; e C_2 e C_3.

3. Resultados

A partir da execução do MS, foram selecionados 17 estudos que apresentavam algum método para estimativa de esforço na área de manutenção de software. Considerando o meio de publicação, nota-se que a maioria dos estudos foi publicado em conferências, correspondendo a 58,8% (10/17), já os demais foram publicados em periódicos – 29,4% (5/17), simpósio – 5,9% (1/17) e workshop – 5,9% (1/17).

Quanto ao ano de publicação, pode-se observar que essa área possui estudos ao passar dos anos, tendo artigos desde 1997 até 2019, sendo que os anos que obtiveram mais artigos foram 2008, 2011 e 2018, possuindo dois estudos cada ano. Utilizando como base o país de origem do primeiro autor, os 17 artigos estão distribuídos em nove países diferentes, sendo Índia o país com mais estudos nessa área, totalizando quatro artigos; Estados Unidos e Brasil produziram três estudos cada, e a Coréia do Sul publicou dois artigos. Nas Figuras 2 e 3 são apresentadas as distruibuições de estudos por ano de publicação e país do primeiro autor, respectivamente.

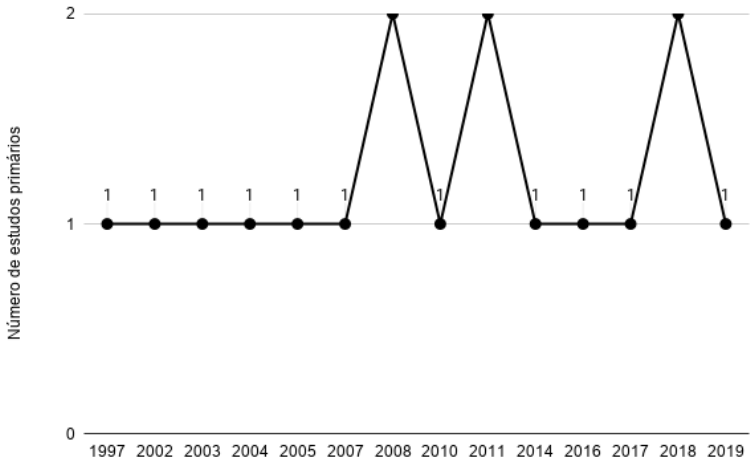


Figura 2. Visão geral dos estudos primários de acordo com o ano de publicação

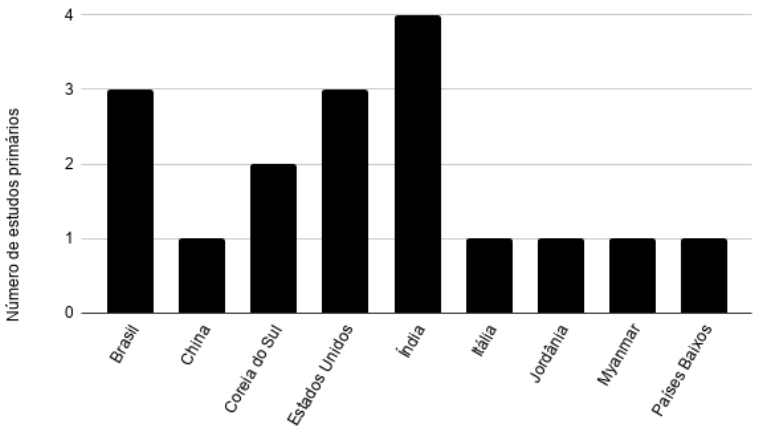


Figura 3. Visão geral dos estudos conforme o país de origem do primeiro autor

Segundo [Lientz and Swanson 1980], existem quatro tipos de manutenção de software: corretiva, adaptativa, perfectiva e preventiva. Observou-se que, referente ao tipo de manutenção a qual seriam empregadas métricas, métodos e ferramentas, cinco estudos não especificaram o tipo de manutenção. Três estudos aplicaram estimativa de esforço relacionado à manutenção corretiva, um estudo focou em manutenção adaptativa, dois estudos focaram em manutenção corretiva e adaptativa, dois focaram em manutenção corretiva e evolutiva e quatro estudos focaram em três tipos de manutenção: corretiva, adaptativa e evolutiva. Conforme ilustrado na Figura 4, pode-se observar a quantidade de estudos considerando separadamente cada tipo de manutenção.

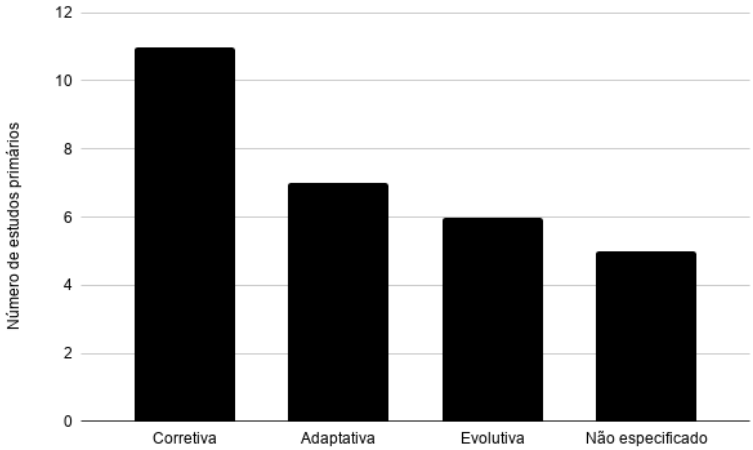


Figura 4. Visão geral dos estudos de acordo com o tipo de manutenção

Para responder a QP, na Tabela 1 são apresentados os métodos utilizados para estimativa de esforço encontrados no MS, juntamente com as categorias nas quais os artigos foram classificados. Observa-se que muitos dos artigos utilizaram métricas (11 de 17) tanto para gerar a estimativa (C_1), quanto para auxiliar outros métodos a gerarem a estimativa (C_1 e C_2). Dos 17 artigos, 13 apresentaram algum método de estimativa de esforço, porém apenas cinco apresentaram apenas o método sem utilizar de métricas ou uma nova ferramenta (C_2). Analisando a categoria C_3, apenas um estudo apresentou um *framework* em conjunto com métodos de estimativa de esforço (C_2 e C_3).

Tendo em vista os estudos que utilizam métricas para o cálculo da estimativa de esforço (categoria 1), observa-se que a métrica mais utilizada é o número de linhas de código (6 de 11 estudos). No entanto, os estudos utilizaram cálculos diferentes dessa métrica: alguns mediram o tamanho do sistema para complementar as demais métricas, outros exploravam a quantidade de linhas alteradas, inseridas ou excluídas para gerar a estimativa. Dos demais estudos, quatro utilizaram pontos de função e apenas um estudo [Chandra et al. 2017] aplicou métricas de orientação a objetos, como número de métodos por classe e profundidade de herança. A Tabela 2 mostra as métricas utilizadas para o cálculo de estimativa de esforço para cada estudo baseado em métricas (C_1).

Quanto à categoria 2, ou seja, técnicas e abordagens para o cálculo da estimativa, observa-se que o método mais usado foi a análise de regressão linear (5 de 13 estudos), o qual permite chegar a um valor estimado a partir de outros valores analisados de uma base de dados históricos. Além disso, quatro estudos utilizaram rede neural, o que acabou

Tabela 1. Visão geral dos estudos primários e seus métodos utilizados

Artigo	Categoria	Métodos
[Niessink and Van Vliet 1997]	C1, C2	Pontos de Função e Analogia
[Leung 2002]	C2	Analogia com o método vizinho virtual (AVN)
[Ahn et al. 2003]	C1	Pontos de Função
[Hayes et al. 2004]	C1	Modelo adaptável baseado em métricas
[De Lucia et al. 2005]	C1, C2	Regressão linear multivariada
[Song et al. 2007]	C1, C2	Modelo probabilístico baseado em rede Bayesiana
[Shukla and Misra 2008]	C1, C2	Rede Neural
[Tenório Jr et al. 2008]	C1, C2	Regressão Linear pelo Menor Quadrado (LSLR) e Pontos de Função
[Thaw et al. 2010]	C2	Mapa auto-organizado em Rede Neural
[Bharathi and Shastry 2011]	C2	Rede Neural
[Nguyen et al. 2011]	C1	COCOMO
[Alomari et al. 2014]	C1, C2	Análise de regressão e fatiamento do código
[Miguel et al. 2016]	C2, C3	Similaridade textual, reputação dos desenvolvedores e análise de regressão
[Chandra et al. 2017]	C1, C2	Análise de regressão e Support Vector Machine (Univariada e Multivariada)
[Hira and Boehm 2018]	C1	COSMIC Pontos de Função com SNAP
[Lélis et al. 2018]	C2	Reputação dos desenvolvedores
[Pillai and Madhukumar 2019]	C2	Julgamento de especialista

Tabela 2. Visão geral dos estudos primários e suas métricas utilizadas

Artigo	Métricas
[Niessink and Van Vliet 1997]	Pontos de Função
[Ahn et al. 2003]	Pontos de Função
[Hayes et al. 2004]	Linhas e operadores alterados
[De Lucia et al. 2005]	Tamanho do sistema (KLOC), número de tarefas
[Song et al. 2007]	Experiência do mantenedor, tamanho do software, características estruturais
[Shukla and Misra 2008]	Complexidade, número de linhas, número de arquivos
[Tenório Jr et al. 2008]	Pontos de Função
[Nguyen et al. 2011]	Linhas inseridas, modificadas e excluídas
[Alomari et al. 2014]	Tamanho total do sistema, tempo de atraso, intervalo de tempo entre versões, número de *hashes* modificado
[Chandra et al. 2017]	Número de métodos por classe, acoplamento, falta de coesão, número de filhos, profundidade da herança
[Hira and Boehm 2018]	Pontos de Função

se tornando o segundo método mais aplicado. Dos demais métodos podemos citar: julgamento de especialista na atividade, similaridade textual e analogia que fazem a busca de atividades já realizadas para assim gerar a estimativa de esforço, dentre outros métodos apresentados na Tabela 1.

Ao analisar a categoria 3, referente a ferramentas e *frameworks*, obteve-se apenas um estudo que se enquadrou nesta categoria [Miguel et al. 2016]. O estudo apresentou um *framework* que além de gerar as estimativas segundo diversas abordagens, apresenta também seis tipos de visualizações em formato de diagramas para ajudar no acompanhamento das atividades de manutenção. Dentre as visualizações, se destacam a visualização da reputação dos desenvolvedores ao longo do tempo, e uma representação em regressão linear do esforço real utilizado nas atividades de manutenção.

4. Ameaças à Validade

A *validade interna* é relacionada ao tratamento dos resultados obtidos. Durante a seleção dos artigos, o fator humano pode ter tendenciado a pesquisa e, consequentemente, afetado os resultados do MS. Para mitigar esta ameaça, toda a condução do MS foi realizada em pares (primeiro e segundo autor) e documentada via planilha compartilhada, a qual foi inspecionada posteriormente por um revisor (terceiro autor).

A *validade externa* é relacionada às chances da generalização dos estudos obtidos. Este MS extraiu estudos de quatro grandes bases de busca a partir de uma *string* de busca abrangente, com apenas dois termos. Em consequência dessa busca ampla, uma grande quantidade de estudos foram retornados no início da busca. No entanto, não foi utilizado um estudo de controle. Dessa forma, há a possibilidade de existirem outros estudos relevantes e que não foram incluídos no MS.

5. Considerações Finais

Neste trabalho foi realizado um mapeamento sistemático a partir das diretrizes propostas por [Kitchenham 2004], com o principal objetivo de encontrar quais métodos, métricas ou ferramentas estavam sendo utilizadas na manutenção de software para realizar estimativa de esforço de atividades. Deste modo, foram apresentados e discutidos os 17 estudos selecionados do mapeamento, os quais foram divididos em três categorias: C_1 – Métricas, C_2 – Métodos e C_3 – Ferramentas.

Os resultados apontam que os métodos são mais utilizados que as métricas, apesar dos dois poderem ser usados em conjunto. Podemos citar que a métrica mais utilizada foi linhas de código, podendo ser usada para medir o tamanho total do sistema, ou identificar linhas alteradas, incluídas ou excluídas. Já o método mais utilizado foi a análise de regressão linear, o que representa que muitos estudos preferem utilizar dados históricos para gerar novas estimativas. Quanto às ferramentas, obteve-se apenas um *framework* proposto, composto por seis tipos de visualizações para facilitar para o usuário o acompanhamento das atividades.

Entre os estudos é possível analisar que poucos levaram em consideração a opinião dos especialistas no projeto quando se trata de estimativas, os quais podem fornecer dados baseados em suas experiências. Diante dos estudos resultantes do MS, observa-se que todos apresentaram um método que aprimora a estimativa, porém apenas o ar-

tigo [Miguel et al. 2016] apresentou uma maneira de ajudar o usuário a gerar essas estimativas. Assim, uma lacuna de pesquisa seria o desenvolvimento de *frameworks* ou ferramentas que facilitassem ao usuário a geração das novas estimativas de esforço para manutenção de software.

Portanto, a principal contribuição deste trabalho é a obtenção de uma visão geral das pesquisas que estavam sendo realizadas de estimativa de esforço e que focassem nas atividades de manutenção de software. Em vista disso, como trabalho futuro, pretende-se realizar um survey, o qual teria como objetivo identificar o que os desenvolvedores levam em consideração no momento de estimar uma nova atividade e quais métodos estão sendo utilizados na indústria. Desse modo, visa-se aprimorar o método de geração de estimativa e deixá-la mais próxima do contexto atual do projeto.

Referências

Ahn, Y., Suh, J., Kim, S., and Kim, H. (2003). The software maintenance project effort estimation model based on function points. *Software maintenance and evolution: Research and practice*, 15(2):71–85.

Alomari, H. W., Collard, M. L., and Maletic, J. I. (2014). A slice-based estimation approach for maintenance effort. In *30th International Conference on Software Maintenance and Evolution*, pages 81–90.

Bharathi, V. and Shastry, U. (2011). Neural network based effort prediction model for maintenance projects. In *11th International Conference on Software Process Improvement and Capability Determination*, pages 236–239.

Chandra, D., Choudhary, M., and Gupta, D. (2017). Prophecy of software maintenance effort with univariate and multivariate approach: By using support vector machine learning techniquewith radial basis kernel function. In *6th International Conference on Computing, Communication and Automation*, pages 876–880.

De Lucia, A., Pompella, E., and Stefanucci, S. (2005). Assessing effort estimation models for corrective maintenance through empirical studies. *Information and Software Technology*, 47(1):3–15.

Erlikh, L. (2000). Leveraging legacy system dollars for e-business. *IT Professional*, 2(3):17–23.

Hayes, J. H., Patel, S. C., and Zhao, L. (2004). A metrics-based software maintenance effort model. In *8th European Conference on Software Maintenance and Reengineering*, pages 254–258.

Hira, A. and Boehm, B. (2018). COSMIC function points evaluation for software maintenance. In *11th Innovations in Software Engineering Conference*, page 4.

Kitchenham, B. (2004). Procedures for performing systematic reviews. *Keele, UK, Keele University*, 33(2004):1–26.

Lélis, C. A., Miguel, M. A., Araújo, M. A. P., David, J. M. N., and Braga, R. (2018). AD-reputation: a reputation-based approach to support effort estimation. In *Information Technology – New Generations*, pages 621–626. Springer.

Leung, H. K. (2002). Estimating maintenance effort by analogy. *Empirical Software Engineering*, 7(2):157–175.

Lientz, B. and Swanson, E. (1980). *Software maintenance management: a study of the maintenance of computer application software in 487 data processing organizations*. Addison-Wesley.

Miguel, M. A., Araujo, M. A. P., David, J. M. N., and Braga, R. (2016). A framework to support effort estimation on software maintenance and evolution activities. In *12th Brazilian Symposium on Information Systems: Information Systems in the Cloud Computing Era*, pages 232–239.

Nguyen, V., Boehm, B., and Danphitsanuphan, P. (2011). A controlled experiment in assessing and estimating software maintenance tasks. *Information and Software Technology*, 53(6):682–691.

Niessink, F. and Van Vliet, H. (1997). Predicting maintenance effort with function points. In *13th International Conference on Software Maintenance*, pages 32–39.

Pillai, S. and Madhukumar, R. (2019). An experiment to improve expert judgment software estimation through work breakdown structure. *Innovative Technology and Exploring Engineering*, 8(7):2278–3075.

Shukla, R. and Misra, A. K. (2008). Estimating software maintenance effort: a neural network approach. In *1st India Software Engineering Conference*, pages 107–112.

Song, T.-H., Yoon, K.-A., and Bae, D.-H. (2007). An approach to probabilistic effort estimation for military avionics software maintenance by considering structural characteristics. In *14th Asia-Pacific Software Engineering Conference*, pages 406–413.

Tenório Jr, N. N., Ribeiro, M. B., and Ruiz, D. D. (2008). A quasi-experiment for effort and defect estimation using least square linear regression and function points. In *32nd Annual IEEE Software Engineering Workshop*, pages 143–151.

Thaw, T., Aung, M. P., Wah, N. L., Nyein, S. S., Phyo, Z. L., and Htun, K. Z. (2010). Comparison for the accuracy of defect fix effort estimation. In *2nd International Conference on Computer Engineering and Technology*, pages 550–554.

Vazquez, C. E., Simões, G. S., and Albert, R. M. (2013). *Análise de Pontos de Função: Medição, estimativas e gerenciamento de projetos de software*. Érica, São Paulo.

Avaliação de Dependências no Desenvolvimento de Sistemas

Carlos Magno Barbosa[1], **Flávio Luiz Schiavoni**[1]

[1] Arts Lab in Interfaces, Computers, and Everything Else - ALICE
Federal University of São João del-Rei - UFSJ
São João del-Rei - MG

cmagnobarbosa@gmail.com, fls@ufsj.edu.br

Abstract. *The library re-usage is one of the most common practices to increase the efficiency of software development and the quality of developed systems. However, this re-usage of libraries can be harmful if it adds problematic dependencies to system maintenance. This paper brings an approach of topics and possible issues that must be taken in considerations during software development and maintenance according to dependencies and used libraries. This analysis can help the dependency optimization relating it with longevity, stability, maturity and maintenance of the system. To exemplify thus purpose it is presented a study of case that describes the analyze on the recovery process in the Harpia / Mosaicode application.*

Resumo. *O reúso de bibliotecas é uma das práticas mais comuns para aumentar a eficiência do processo de desenvolvimento e a qualidade dos sistemas desenvolvidos. Contudo, esse reúso pode ocorrer de forma não apropriada e adicionar dependências externas problemáticas à manutenção do sistema. Diante desse contexto, este artigo tem o objetivo de propor uma abordagem de tópicos e questões que devem ser levadas em consideração durante o desenvolvimento ou manutenção de um sistema com o intuito de analisar a dependência desse sistema para com as bibliotecas utilizadas. Essa análise tem como objetivo a otimização das dependências e está relacionada com a longevidade, estabilidade, maturidade e manutenibilidade do sistema. Para exemplificar a proposta, um estudo de caso que descreve a análise no processo de recuperação da ferramenta Harpia / Mosaicode.*

1. Introdução

Reutilizar trechos de código (reuso) tornou-se uma prática comum no desenvolvimento de software para possibilitar a entrega de aplicações complexas de forma eficiente em termos de tempo e custo [Shiva and Abou Shala 2007]. Nesse contexto, bibliotecas de software são alguns dos artefatos mais reutilizados, pois são soluções comumente fornecidas por diversos desenvolvedores, em diversas linguagens de programação para os mais variados problemas. Neste artigo, o termo biblioteca de software é utilizado para descrever trechos de códigos reutilizáveis, como APIs de funções, componentes ou *frameworks*.

Os desenvolvedores também podem criar as suas próprias bibliotecas para reutilizá-las em diferentes projetos. Isso permite a reutilização de trechos de código para criar novos sistemas de software. A reutilização dessas bibliotecas pode resultar em um ganho de produtividade dos desenvolvedores e da qualidade do código-fonte dos sistemas [Caldiera and Basili 1991].

Apesar das vantagens mencionadas, a utilização de bibliotecas provenientes de terceiros pode trazer alguns problemas para o código de um sistema. Caso as bibliotecas sejam muito genéricas ou específicas demais, os desenvolvedores precisam realizar adaptações que podem resultar em um código-fonte pouco eficiente. Outro ponto a ser avaliado é que a reutilização de uma biblioteca de terceiros não garante a integração dessa biblioteca ao projeto de maneira adequada ao sistema. [Pressman and Maxim 2016]. Pra garantir uma integração eficaz e adequada é necessária uma avaliação da maturidade e confiabilidade, análise das licenças, análise das dependências, verificação da portabilidade e adequação das bibliotecas com os requisitos do sistema [Gimenes and Huzita 2005].

Além desse problema, a utilização de uma biblioteca em um projeto pode gerar uma dependência da aplicação resultante do projeto para com esse artefato. Consequentemente, a descontinuidade dessa biblioteca pode comprometer o funcionamento e a qualidade de uma aplicação que dependa da mesma. Além disto, depender de diversas bibliotecas pode resultar em um aumento de custo de manutenção e no aumento da complexidade de um projeto. Por esta razão, a reutilização de código por meio da utilização de bibliotecas de terceiros deveria passar por uma análise criteriosa de escolha que ajude a decidir quais e como as bibliotecas devem ser reutilizadas em um determinado sistema, de modo que essa reutilização traga benefícios reais para o projeto.

Diante desse contexto, este artigo tem o objetivo de propor uma abordagem de tópicos e questões que podem ser levadas em consideração durante o processo de desenvolvimento ou de manutenção de um sistema. O intuito dessa abordagem é analisar a dependência desse sistema para com as bibliotecas utilizadas, visando reduzir o impacto negativo que este reúso pode causar na evolução desse sistema. Este artigo apresenta ainda um caso de uso em que a abordagem proposta foi aplicada no processo de refatoração da ferramenta Harpia / Mosaicode [Schiavoni and Gonçalves 2017], que estava descontinuada e encontrava-se não funcional devido a problemas em suas dependências.

2. Background

Bibliotecas de software são trechos de códigos previamente construídos que podem ser utilizados para auxiliar na implementação da funcionalidade de um sistema. Algumas bibliotecas são fornecidas em conjunto com os compiladores das linguagens de programação e fornecem soluções para problemas simples e recorrentes, como, por exemplo, entrada e saída da dados, números randômicos, funções matemáticas e obtenção de dados do sistema operacional. Há também bibliotecas obtidas separadamente, comumente baixadas a partir do site do desenvolvedor/fornecedor. Normalmente, esse último tipo de biblioteca fornece componentes ou *frameworks* que auxiliam na solução de problemas mais complexos, como por exemplo, a manipulação de bancos de dados ou a construção da arquitetura do sistema.

Um componente de software é um artefato de software reutilizável no estilo caixa-preta. Ele possui uma interface bem definida que define quais dados devem lhe ser passados e qual saída é obtidas. Diferentemente de bibliotecas, um componente resolve um problema pontual sendo que diversos componentes podem atuar em conjunto para resolver um problema mais complexo podendo formar bibliotecas de componentes. O exemplo mais comum de componentes são os *widgets* de interface gráfica com o usuário (GUI).

Framework é um artefato de software que implementa a funcionalidade de um

domínio sendo formado por um conjunto de classes, ou componentes, que possuem uma dependência entre si. Um *framework* pode ser reutilizado na forma de uma caixa-preta, caixa-branca (herança de seus classes), ou um meio termo entre esses dois últimos (chamado de caixa-cinza). Outra diferença, em comparação às bibliotecas de funções e de componentes, é que os *frameworks* assumem o controle da execução da funcionalidade, o que nos permite dizer que é o *framework* que utiliza o código do sistema, não o contrário [Abi-Antoun 2007, Shiva and Abou Shala 2007].

Quando uma aplicação faz reúso de um artefato, como, por exemplo, uma biblioteca, cria-se uma dependência dessa aplicação para com esse artefato. Essa dependência pode ser amenizada por meio de um projeto de software flexível e coeso que oculta os detalhes dos artefatos reutilizados da implementação da funcionalidade que a reutiliza. Uma forma de se conseguir isso é por meio de padrões de projeto de software.

Padrões de software são modelos de descrição de problemas recorrentes e suas soluções [Pressman and Maxim 2016]. Eles possibilitam ao desenvolvedor reutilizar a experiência e os procedimentos que foram adotados por outros desenvolvedores em situações semelhantes. Assim, ao invés de criar um solução própria que possivelmente não será a ideal, o desenvolvedor pode adotar a solução proposta por um padrão de software, que é reconhecida como a mais adequada [Group 2017].

Existem padrões para diferentes níveis do desenvolvimento de software. Entre eles, os padrões de projeto são os mais conhecidos e estão relacionados com a construção de código flexível e de manutenção facilitada [Manolescu et al. 2007]. Alguns deles objetivam diminuir a dependência entre as classes da aplicação, como, por exemplo, os padrões Abstract Factory, Decorator, Observer e Strategy , permitindo que classes possam ser removidas, modificadas e/ou inseridas sem que sejam necessárias alterações nos demais módulos da aplicação.

Não existe um consenso sobre o que é arquitetura de software. Em geral, a arquitetura está relacionada com a definição do objetivo e das interfaces de cada componente/módulo do software [Rosik et al. 2011]. A organização dos componentes de um software, seja ele construído com bibliotecas, *frameworks* e projetado ou não com o uso de padrões e outros recursos, compõe a arquitetura desse software. A arquitetura de software é um dos artefatos mais importantes no ciclo de vida de um sistema. Ela interfere nos objetivos de negócios, objetivos funcionais e na qualidade do sistema [Bessa et al. 2016, Melo et al. 2016].

3. Método

O presente trabalho propõe uma metodologia para a análise de dependências na adoção de bibliotecas em projetos de software. A metodologia proposta passa por alguns passos ou etapas, conforme será apresentado. Esta metodologia pode ser utilizada ao encontrarmos uma determinada característica do sistema que é comum a outros sistemas computacionais ou um problema comum a ser resolvido. Diante deste problema, há duas possibilidades: implementar algo que já pode estar implementado ou adotar uma solução já existente. Primeiramente, será analisada a possibilidade de desenvolver uma solução própria.

3.1. Desenvolvendo soluções próprias

A possibilidade de desenvolver uma solução própria é viável, principalmente, no caso de esta solução ser ótima e atender a todos os requisitos do projeto. Certamente espera-se que os requisitos estejam muito bem elucidados e que haja experiência na equipe com o tipo de problema que a solução pede. Desse modo, alguns passos podem ser dados:

- Garantir a componentização da solução e/ou o desenvolvimento da solução como uma biblioteca que atenda diretamente ao problema comum e que poderá ser reutilizada em outros projetos para resolver o mesmo problema.
- Documentar e incluir exemplos desta biblioteca de forma a garantir que seu uso por terceiros seja facilitado.
- Realizar testes e incluí-los na biblioteca.
- Disponibilizar esta biblioteca em um repositório próprio incluindo o código-fonte e uma licença de software (livre).
- Divulgar a sua biblioteca e convidar outras pessoas a ajudarem no projeto.

Realizando esses passos, é possível conseguir uma solução própria que atenda os requisitos da metodologia, garantindo um manutenibilidade futura para essa biblioteca. Vale lembrar que o desenvolvimento de uma solução própria pode ser a única decisão a ser tomada caso a adoção de uma biblioteca de terceiro não seja possível por qualquer dos critérios que serão apresentados a seguir.

3.2. Adotando bibliotecas de terceiros

No caso de ser decidido adotar uma biblioteca de terceiro, um processo de avaliação das características desejáveis desta biblioteca deve ser iniciado. Esta avaliação deve ser feita para tentar reduzir os impactos negativos da adição de uma biblioteca externa ao projeto e evitar problemas como, por exemplo, de portabilidade e obsolescência.

Manutenção e auditoria
A distribuição de uma biblioteca pode ser feita por meio de arquivos binários ou por meio de seu código-fonte. A distribuição do código-fonte não é comum em bibliotecas proprietárias. Por isto, essas bibliotecas não podem ser auditadas ou adaptadas pelos desenvolvedores do Sistema e, apesar de muitas vezes serem funcionais e aparentemente inofensivas, podem trazer problemas de desempenho ou de segurança para o sistema. Consequentemente, elas se tornam um ponto fraco no sistema, uma vez que o desenvolvedor não consegue garantir o funcionamento desejado. Já bibliotecas *Free Libre Open Source Software* (FLOSS) são distribuídas com seu código-fonte. Isso permite que as mesmas sejam alteradas, modificadas, distribuídas e auditadas pela equipe desenvolvedora do Sistema [Mancinelli et al. 2006]. Por este motivo, para garantir a manutenção e auditoria do código do sistema, recomenda-se a utilização de bibliotecas de código aberto.

Longevidade
A longevidade do Sistema é diretamente influenciada pela longevidade de suas bibliotecas, pois, se um produto de software depende de uma biblioteca que não possui mais atualização, o mesmo poderá se tornar obsoleto. Neste ponto, é importante que um projeto possua dependências apenas de códigos de terceiros que sejam distribuídos por entidades reconhecidas, sejam elas empresas ou comunidades de software, e que a velocidade de adequação destas bibliotecas às novas necessidades do mercado sejam adequada e de

acordo com os requisitos do Sistema. Caso uma biblioteca esteja sem sofrer atualizações há muito tempo ou se sua comunidade de desenvolvimento e/ou empresa responsável parece inativa, a utilização desta biblioteca poderá trazer problemas futuros ao sistema que dela depender. Para analisar a longevidade de um projeto é possível analisar o tempo de vida do projeto, versão em que ele se encontra e número de atualizações do mesmo ao longo de sua existência.

Maturidade
Uma biblioteca madura e testada em diversos projetos tende a ter menos falhas do que um código criado exclusivamente para resolver um único problema. Isso porque:

1. diversos desenvolvedores já analisaram esse código;
2. após ter sido utilizada em inúmeros sistemas, a biblioteca já teve diversos problemas identificados e solucionados. Algo improvável em um projeto iniciante;
3. alguns testes, como o de carga e estresse, só são possíveis após os usuários entrarem em contato com o sistema.

Por esta razão, a incorporação de uma biblioteca madura ao projeto pode adicionar ao projeto uma solução de qualidade testada e melhorada. Esta característica nos leva a acreditar que é melhor adotar de bibliotecas que tenham sido amplamente utilizadas por outros projetos e a evitar a adoção de bibliotecas novas que foram pouco testadas.

Portabilidade
A portabilidade de um Sistema é sua capacidade de funcionar em diversas plataformas. Para que isto ocorra, é necessário que as bibliotecas que o mesmo utiliza como dependência estejam disponíveis nas plataformas as quais o software será utilizado. Por esta razão, a escolha das bibliotecas deve ser embasada no suporte da plataforma de destino a essas bibliotecas. Caso a análise aqui mencionada não seja feita nos instantes iniciais do desenvolvimento do projeto, o custo na portabilidade do software poderá ser muito alto, pois será necessário portar também todas as dependências externas do mesmo.

Compatibilidade de Licenças
Um ponto que deve ser observado ao compilar, integrar e distribuir uma biblioteca externa juntamente com o Sistema é a sua licença. Quando se deseja distribuir um compilado ou fechar o código de um produto, as permissões das dependências utilizadas devem ser avaliadas pois podem haver incompatibilidades entre a licença do Sistema e as licenças de suas bibliotecas. Algumas licenças não podem ser usadas simultaneamente em um projeto e possuem restrições de compilação [German and Hassan 2009]. Portanto deve ser avaliado se as permissões de uso dessas bibliotecas atendem aos objetivos do projeto.

Independência de outras bibliotecas
Uma biblioteca pode atender aos requisitos de um sistema inicialmente mas possuir dependências de outras bibliotecas que não atendem a estes requisitos. Ao assumir a dependência de uma biblioteca que possui dependências de outras bibliotecas é necessário fazer esta mesma análise com todas as dependências sucessivamente [Kula et al. 2014]. Por esta razão, deve-se evitar criar dependência de uma biblioteca que depende de muitas bibliotecas devido ao aumento da complexidade que esta adoção pode trazer ao Sistema.

Legibilidade e documentação
A adoção de uma biblioteca aumenta a complexidade do código-fonte de um Sistema pois para entender o Sistema pode ser necessário entender a API da biblioteca em questão. Por esta razão, para garantir a legibilidade do código e o aprendizado do mesmo é necessário que a biblioteca possua documentação disponível para o aprendizado de sua API e preferencialmente possua exemplos de código para facilitar o seu aprendizado.

3.3. Integrando a solução

Independentemente de ter desenvolvido uma solução própria na forma de uma biblioteca ou ter adotado bibliotecas de terceiros, uma série de decisões podem ser tomadas na integração da solução ao projeto. Estas decisões estão associadas ao nível do acoplamento da biblioteca ao código implicar em uma dependência forte ou fraca.

Abordagem arquitetural
Para reduzir o acoplamento de um código, é possível utilizar uma abordagem arquitetural que consiga isolar a dependência. Um exemplo é a utilização de uma arquitetura em camadas onde apenas uma camada irá ter contato com a biblioteca a ser acoplada. Isso evita que a dependência seja propagada por todo o projeto e permite a substituição da dependência sem grandes custos ao projeto.

Padrões de projeto
Na programação orientada a objetos há diversos padrões de projeto que permitem isolar componentes de um sistema de maneira a criar uma interface para os mesmos, como por exemplo, o Facade ou o Bridge, e isolar a complexidade de um sistema. Com o uso desses padrões de projeto é possível encapsular a dependência em uma classe do próprio sistema e isolá-la de maneira que a substituição desta biblioteca seja possível sem grandes refatorações de código no sistema. Apesar de padrões de projeto serem uma metodologia de desenvolvimento voltada para programação orientada a objetos, este isolamento também é possível em linguagens de programação não orientadas a objetos.

3.4. Resumo do método

As características levantadas no método proposto permitem auxiliar a decisão de projeto quanto à adoção de código de terceiros no desenvolvimento de um Sistema. Parte do método proposto pode ser automatizada (eg. independência de outras bibliotecas).

4. Estudo de Caso: Harpia / Mosaicode

Este estudo e avaliação das dependências foi feito para auxiliar o processo de refatoração e empacotamento de uma ferramenta já existente chamada Harpia. Esta ferramenta tem o código aberto disponível e foi incluída nos repositórios Debian e Ubuntu por bastante tempo mas em determinado momento deixou de estar disponível devido ao abandono da manutenção de seu código. Apesar de não existir uma necessidade evidente de manutenção na funcionalidade da ferramenta, a mesma possuía dependências para com bibliotecas que tiveram seu desenvolvimento descontinuado. Por esse motivo, a ferramenta encontrava-se totalmente inoperante para os sistemas atuais. Eu sua refatoração, a ferramenta foi renomeada e passou a ser chamada de Mosaicode.

Dentre as dependências que impossibilitavam o funcionamento da ferramenta, destacam-se a biblioteca **Amara**, usada para a persistência de dados em XML, e a biblioteca **GLADE**, usada na construção da GUI da ferramenta.

Além disto, as bibliotecas Amara e Glade estavam entrelaçadas de maneira forte com o sistema, o que fez com que a refatoração tivesse que ser feita de maneira a alcançar diversas classes da aplicação. Com isto, também foi notado que seria mais adequado que a dependência ocorresse de maneira menos transversal ao código-fonte tornando uma futura modificação menos impactante.

4.1. Desenvolver ou adotar?

No instante inicial da refatoração do código do Harpia e com a ferramenta inoperante devido a dependências quebradas, surgiu a oportunidade de aplicarmos o método aqui apresentado. Diante da decisão imposta neste método, optamos por Adotar solução de terceiros. O desenvolvimento de soluções próprias para estes requisitos foi descartado pois existem soluções maduras e reconhecidas tanto para a implementação de persistência XML quanto para componentes gráficos e GUI.

Além disto, desenvolver uma solução própria teria um custo alto para o projeto, implicaria em reimplementar soluções existentes, poderia trazer erros já solucionados em códigos de terceiros, traria problemas de portabilidade e estava fora do escopo do projeto. Desse modo, foi decidido utilizar bibliotecas de terceiros e que seria necessário uma criteriosa análise de dependências para reduzir o impacto da obsolescência futura dessas bibliotecas.

Após um levantamento inicial das bibliotecas disponíveis para solucionar os pontos que causavam a obsolescência do Sistema, optou-se pelas seguintes substituições:

- A persistência XML feita anteriormente pelo **Amara** foi substituído pela biblioteca **Beautiful Soup**[1];
- Os componentes de GUI gerenciados anteriormente pela ferramenta **Glade** foi substituído pelo **PyGObject - GTK**[2];

As bibliotecas destacadas BeautifulSoup e PyGObject - GTK possuem código aberto, uma grande comunidade de desenvolvimento e suas licenças são compatíveis com a licença original da ferramenta. Elas também são utilizadas por diversos projetos de software reconhecidos e são portáveis para os Sistemas Operacionais Windows, MacOS e Linux. Além disto, estas bibliotecas possuem mais de 10 anos de existência, ampla documentação disponível para o seu aprendizado e pouca ou nenhuma dependência de outras bibliotecas de terceiros.

Considerando que as bibliotecas utilizadas atenderam de forma satisfatória todas as as características listadas anteriormente, se optou por isolar essas dependências e torná-las dependências fracas cuja substituição no futuro não implicará na reescrita de todo o sistema.

Persistência XML

Os trechos de código que liam e escreviam arquivos XML estavam espalhados por diversas classes da ferramenta. A refatoração do código e a adoção da nova biblioteca para

[1] `https://www.crummy.com/software/BeautifulSoup/bs4/doc/`
[2] `https://pygobject.readthedocs.io/en/latest/`

esta funcionalidade iniciou-se pelo isolamento de todos os métodos de persistência em algumas classes responsáveis pela persistência.

Depois disto, foi implementada uma classe proxy que traz para o sistema todas as funcionalidades da nova dependência e todas as classes que trabalham com persistência chamam métodos desta classe do sistema. Desta maneira, caso a dependência quebre, somente será necessário realizar a manutenção na camada de abstração.

Nova GUI

No caso da interface gráfica, criar uma classe proxy resultaria em um esforço considerável de reescrita do código. Por isto, foi utilizada uma técnica para isolar a dependência baseada no padrão *Model-View-Controller* (MVC). Este padrão separa o desenvolvimento do sistema em camadas, facilitando a alteração da interface gráfica sem afetar a parte funcional. Assim, com a adoção do padrão MVC, a solução implementada separou os componentes de GUI da funcionalidade da ferramenta. Está abordagem permite inclusive que o sistema funcione sem GUI.

Além da adoção do padrão MVC, toda a utilização da API Gtk aconteceu por meio de classes que implementam componentes e estendem a API Gtk e fornecem uma interface simples para os demais componentes do sistema. Isto tornou as dependências do projeto mais fracas ou seja uma modificação de um componente implica na modificação de um trecho reduzido de código. Portanto o impacto de futuras modificações foi reduzido. A componentização dos elementos de GUI é a estratégia geral para enfraquecer a dependência e segue o principio de reduzir o número de funções de cada componente, modularizando o sistema o máximo possível.

5. Trabalhos Relacionados

Alguns trabalhos propõem abordagens de mineração de padrões de uso de APIs [Niu et al. 2017, Mendez et al. 2013]. Um padrão de uso de uma API documenta a sequência necessária de chamadas de métodos das classes dessa API para que a sua funcionalidade seja corretamente reutilizada. Esses padrões de uso são definidos por meio de processos de mineração que identificam as sequências de chamadas de métodos das classes das APIs [Zhong et al. 2009]. O intuito dessas abordagens é fazer com que as APIs sejam reutilizadas de maneira correta e eficaz pelo sistema.

Outro nicho de pesquisa aborda os desafios provenientes do reúso de software, como custo e dificuldades para a manutenção de bibliotecas, ausência de ferramentas de suporte e o custo para identificação e adaptação dessas bibliotecas [Hummel 2010]. Alguns trabalhos propõem como solução para esses desafios a reutilização sistemática de software por meio de ambientes integrados de reúso [Ahmed 2011]. Esse levantamento permitiu aos autores obter uma compreensão das atuais abordagens de reutilização sistemática e respectivos ambientes de reutilização, bem como identificar as deficiências correspondentes.

6. Conclusão

Este artigo apresentou um método para a avaliação de dependências no desenvolvimento de um Sistema partindo do entendimento que as escolhas das dependências são um passo essencial do desenvolvimento de um projeto e que essas escolhas podem ser decisivas

para o aumento de complexidade do código, custo de desenvolvimento, qualidade, portabilidade, longevidade, liberdade e produtividade de um sistema. Para que a decisão no momento de adotar ou não uma biblioteca tenha uma maior probabilidade de ser correta, este artigo se apresenta como um guia que pode auxiliar o desenvolvedor a tomar decisões quanto a criação de dependências externas em um código. Este estudo não traz uma resposta objetiva à questão da adoção de componentes de terceiros, mas pode auxiliar a equipe a considerar diversos fatores na decisão de gerar novas dependências com a adição de um componente externo.

Certamente, não utilizar dependências externas podem tornar o desenvolvimento de um Sistema mais custoso e até mesmo inviável. Por isto, apresentamos a possibilidade de garantir a manutenibilidade futura do Sistema por meio de adoção de técnicas de engenharia de software que permitem isolar bibliotecas de modo a enfraquecer a dependência do sistema em relação a códigos de terceiros. Com isto, é possível trazer para o projeto os ganhos e a maturidade da reutilização de bibliotecas de terceiros sem aumentar a complexidade do código, impacto e reduzir os custos de manutenção do sistema. Destacamos que o método proposto pode ser aplicado em sistemas legados e no desenvolvimento de novas aplicações.

Este artigo trouxe ainda um estudo de caso que utilizou o método aqui proposto na refatoração da ferramenta Harpia, que se encontrava inoperante devido a depreciação de suas dependências antigas. Essa depreciação da ferramenta Harpia permitiu ao grupo de pesquisa observar como as dependências estão diretamente relacionadas à longevidade de um Sistema e como uma dependência forte com uma biblioteca descontinuada pode resultar em um sistema não funcional e com alto custo de manutenção. Ao fim da refatoração de seu código, a ferramenta Harpia foi renomeada e passou se chamar Mosaicode.

Há uma tomada de decisão não elucidada neste trabalho que remete ao caso de nossa análise apontar para várias bibliotecas com características similares, que cumprem nossos requisitos e que atendem às necessidades. Como trabalhos futuros, pretendemos investigar metodologias que ajudem o desenvolvedor a escolher entre estas bibliotecas. Também é parte de nossos trabalhos futuros encontrar outras arquiteturas de sistemas e padrões de projetos que possam diminuir o acoplamento de bibliotecas e sistemas no que tange suas dependências de código externo.

Referências

Abi-Antoun, M. (2007). Making frameworks work: a project retrospective. In *Companion to the 22nd ACM SIGPLAN conference on Object-oriented programming systems and applications companion*, pages 1004–1018. ACM.

Ahmed, M. A. (2011). Towards the development of integrated reuse envirnoments for uml artifacts. In *Proceedings of the 6th International Conference on Software Engineering Advances*, pages 426 – 431.

Bessa, S., Valente, M. T., and Terra, R. (2016). Modular specification of architectural constraints. In *2016 X Brazilian Symposium on Software Components, Architectures and Reuse (SBCARS)*, pages 31–40.

Caldiera, G. and Basili, V. R. (1991). Identifying and qualifying reusable software components. *Computer*, 24(2):61–70.

German, D. M. and Hassan, A. E. (2009). License integration patterns: Addressing license mismatches in component-based development. In *Proceedings of the 31st International Conference on Software Engineering*, pages 188–198. IEEE Computer Society.

Gimenes, I. M. d. S. and Huzita, E. H. M. (2005). Desenvolvimento baseado em componentes: conceitos e técnicas. *Rio de Janeiro: Ciência Moderna*.

Group, T. H. (2017). Design patterns library.

Hummel, O. (2010). Facilitating the comparison of software retrieval systems through a reference reuse collection. In *Proceedings of the Workshop on Search-Driven Development: Users, Infrastructure, Tools and Evaluation*, pages 17–20.

Kula, R. G., De Roover, C., German, D., Ishio, T., and Inoue, K. (2014). Visualizing the evolution of systems and their library dependencies. In *Software Visualization (VISSOFT), 2014 Second IEEE Working Conference on*, pages 127–136. IEEE.

Mancinelli, F., Boender, J., Di Cosmo, R., Vouillon, J., Durak, B., Leroy, X., and Treinen, R. (2006). Managing the complexity of large free and open source package-based software distributions. In *Automated Software Engineering, 2006. ASE'06. 21st IEEE/ACM International Conference on*, pages 199–208. IEEE.

Manolescu, D., Kozaczynski, W., Miller, A., and Hogg, J. (2007). The growing divide in the patterns world. *IEEE Software*, 24(4):61–67.

Melo, I., Santos, G., Serey, D. D., and Valente, M. T. (2016). Perceptions of 395 developers on software architecture's documentation and conformance. In *X Brazilian Symposium on Software Components, Architectures and Reuse*, pages 81–90.

Mendez, D., Baudry, B., and Monperrus, M. (2013). Empirical evidence of large-scale diversity in api usage of objected-oriented software. In *Proceedings of 13th IEEE International Working Conference on Source Code Analysis and Manipulation (SCAM)*, pages 43–52.

Niu, H., Keivanloo, I., and Zou, Y. (2017). Api usage pattern recommendation for software development. *Journal of Systems and Software*, 129:127 – 139.

Pressman, R. and Maxim, B. (2016). *Engenharia de Software-8ª Edição*. McGraw Hill Brasil.

Rosik, J., Gear, A. L., Buckley, J., Babar, M. A., and Connolly, D. (2011). Assessing architectural drift in commercial software development: a case study. *Software: Practice and Experience*, 41(1):63–86.

Schiavoni, F. L. and Gonçalves, L. L. (2017). Teste de usabilidade do sistema mosaicode. In *Anais [do] IV Workshop de Iniciação Científica em Sistemas de Informação (WICSI)*, pages 5–8, Lavras - MG - Brazil.

Shiva, S. G. and Abou Shala, L. (2007). Software reuse: Research and practice. In *Fourth International Conference on Information Technology (ITNG'07)*, pages 603–609. IEEE.

Zhong, H., Xie, T., Pei, P., and Mei, H. (2009). Mapo: mining and recommending api usage pattern. In *Proceedings of European Conference on Object-Oriented Programming*, page 318–343.

Search-based Test Data Generation for Mutation Testing: a tool for Python programs

Matheus M. Ferreira[1], **Lincoln M. Costa**[2], **Francisco Carlos M. Souza**[1]

[1]Departament of Software Engineering, Federal University of Technology - Paraná, Dois Vizinhos-PR, Brazil

[2]Computer Systems Engineering Program, Federal University of Rio de Janeiro, Rio de Janeiro-RJ, Brazil

matheusf.2016@alunos.utfpr.edu.br,costa@cos.ufrj.br,

franciscosouza@utfpr.edu.br

Abstract. *Test data generation for mutation testing consists of identifying a set of inputs that maximizes the number of mutants killed. Mutation Testing is an excellent test criterion for detecting faults and measuring the effectiveness of test data sets. However, it is not widely used in practice due to the cost and complexity to perform some activities as generating test data. Although test suites can be produced and selected manually by a tester this practice is susceptible to errors and tools are needed to facilitate it. Several tools have been developed to automate mutation testing, but, only a few address the test data generation. The present paper proposes an automated test data generation tool based on weak mutation for Python programming language using the Hill Climbing algorithm. For evaluation, we performed an empirical study concerning the effectiveness and cost computational of the tool in a database composed of 348 mutants and we compare it with random generation. Overall, the experiment achieved an average mutation score of 86% for our proposed tool and random testing 64% on average.*

1. Introduction

Mutation testing is the most common type of software fault-based testing and this is based on producing hypothetical faulty programs by creating variants of the program under test (PUT). This criterion was proposed by [DeMillo and Offutt 1991] for detecting faults and measuring the effectiveness of tests. The process works from faults that are injected into the program under test to produce faulty program versions called mutants. In general, to perform software testing using any technique is required to create a set of test cases to execute the artifact under test, be it code, interfaces, or requirements. A test case is composed of a pair of, *i)* test data which are inputs to execute the program under test and *ii)* expected output. Thus, test data generation is an activity focused on finding valid input according to specific test criteria.

Mutation testing is widely considered as effective and powerful. However, this criterion also is known as extremely costly, mainly due to three factors: i) the high number of mutants produced, ii) the number of test data necessary to identify the mutants, and iii) the difficulty of finding the equivalent mutants which are mutants with the same behavior regarding the original program. The number of test data to detect a fault into a mutant is

related to the strategy applied for generating them since if the test data set is generated manually can be labor-intensive and susceptible to errors task [Papadakis and Malevris 2010].

The main objective of the mutation testing is to generate a test data set that can reveal the faults in the mutant programs so that they fail, that is, to distinguish the outputs of the mutant programs from the original ones. The test data generation is an activity that has been a growing interest in the community due to being hard and it not having reached a high level of automation. Thus, an approach able to completely automate this activity is an important step to reduce mutation testing costs and increase more reliability in the tests. For these reasons intelligent approaches to solving this problem have been arising amongst them the utilizing of search-based algorithms.

Mutation testing is considered to be a powerful test criterion, due to its effectiveness in revealing faults. Nevertheless, it is a very expensive technique, for this reason, alternatives have emerged to reduce its costs which were named, weak mutation, firm mutation, and strong mutation that refers to the traditional mutation testing. In this case, mutants killed in this context is said as mutants strongly killed. In weak mutation, mutants are killed if immediately after the execution of the mutated statements there is a state difference between the original and mutant programs, and they are called weakly killed mutants. In Firm mutation, the aim is to verify if there is a difference between the states of the original and mutant programs at a later point after the execution of the mutation point, the more differences, the more likely the mutant has to be killed by a test die.

In this context, the paper presents an automated approach for generating test data based on the weak mutation to strongly kill mutants using a Hill Climbing algorithm for programs written in the Python language. In this study, we utilized an objective function derived from a study proposed by Weneger et al. [Wegener et al. 2001] and applied for the same context in [Souza et al. 2016] called Reach distance and Mutation distance. The difference from the previous works is that it aims at generating test data to kill mutants in Python, the frameworks in the literature only automatize the process of mutants production and tests execution that shows there is still the problem in this area.

2. Mutation testing for Python

Mutation testing aims to verify whether the program under test is not present defects and also assess the quality of the test suite. Assessing of the test sets, faulty versions of the program original are produced, which simulate the mistakes made by programmers, these versions are defined as mutants. Hence, the goal is to execute test data that causes the mutants to behave differently from the original program. A test data that identifies a fault makes the mutant be considered dead, otherwise, it is said to be alive [DeMillo and Offutt 1991]. However, there are two possibilities if a mutant remains alive after executing each input of the test data set. The first one, the mutant is considered equivalent, i.e,. for all inputs, the mutant will produce the same output as the original program. The second possibility is that the test set is weak to kill the mutant, i.e., improvements are necessary to carry out the mutants identifications. To generate the mutants, it is necessary to consider the features of a programming language, since the process for generating mutants is performed though applying mutation operators. The mutant operators are usually based on typical errors that occur during the software development and they determine the type

of syntactic change that must be made to generate mutants, such as command mutations, operator mutations, variable mutations, and constant mutations.

Performing mutation testing depends largely on the existing tools to automate its process since applying this technique manually is impracticable. Despite producing mutants and generating test sets can be performed non-automatic by a tester, this practice is complex and time-consuming. For Python programming language, according to Derezinska and Halas [Derezinska and Halas 2014] the first mutation tool was created in 2002 based on a Java mutation tool Jester and currently has emerged better tools, but not sufficiently automated to perform all mutation testing activities as can be seen in Table 1.

Currently, we consider four main tools to test python programs based on mutation testing and they are defined as Cosmic Ray, MutMut, MutPy, PyMuTester. Cosmic Ray is a mutation testing tool for Python 3 and it has been successfully used on a variety of projects ranging such as assemblers to oil exploration software. MutMut is a mutation testing system also for Python 3 it focuses mainly on ease of use and supports to all test runners. MutPy is a tool for Python programs from 3.3+ version, it uses the standard unittest module, generates YAML/HTML reports, and has colorful output, also supports high order mutations (HOM) and code coverage analysis. PyMuTester is a tool to execute mutation testing, its main purpose is to assist faults in if-condition negation and loop skipping

Table 1 shows information about whether the tools can produce mutants (second column), they generate test data sets automatically (third column), execute the test sets against the mutants (fourth column), if the tools have some features to analyze equivalent mutants by the tester (fifth column), the number of mutant operators available (sixth column) and the year of the first and the last version (seventh and eighth column). As we can see, the features analyzed in the tools, most of them can produce mutants and execute the tests against mutants, however, none of them generate test data sets, i.e., the inputs must be produced manually to execute the tests.

Table 1. Python Mutation Testing Tools

Tools	Generate Mutants	Generate Test sets	Execute Test	Marking of Eq. Mutants	Mutant Operators	First Version	Current Version
Cosmic Ray	Yes	No	Yes	Yes	14	2017	2020
MutMut	Yes	No	Yes	No	12	2016	2020
MutPy	Yes	No	Yes	No	17	2011	2019
PyMuTester	Yes	No	Yes	No	2	2011	2017

3. Related work

Several publications have appeared in recent years documenting different techniques for test data generation in mutation testing, these studies address the use of search-based algorithms such as Hill Climbing [Souza et al. 2016], particle swarm optimization (PSO) [Jatana et al. 2016] and, genetic algorithm (GA) [Rani et al. 2019]. There are also a few studies specifically about mutation testing for Python [Derezinska and Halas 2015] and [Derezinska and Halas 2014]. However, as far we know no research addresses automatic test data generation for this context.

Papadakis and Malevris [Papadakis and Malevris 2010] proposed an approach that generates test data to kill weak and strong mutants using Dynamic Symbolic Exe-

cution (DSE) for Java programs. The approach transforms the original program into a meta-program, containing all the weak-mutant-killing constraints, and then the test data to cover all the branches the meta-program are generated through DSE. Thus, the DSE produces test data to strongly kill the mutants automatically based on weak-mutant-killing constraints. The work concludes that a high level of automation of the generation of test cases for killing the mutants can be achieved.

According to Souza et al. [Souza et al. 2016], a Hill Climbing algorithm was used to generate the test data for weak and strong mutants for programs written in C. In this study, an objective function involving three parts was used to find the best result. The first one, with the help of Branch Coverage Testing, is guided to the test data to reach the mutation point. The second function is used to assist in the generation of data test that can infect the mutation point, and the third one makes this infection to impact the flow of the program in such a way that the fault is propagated to the output. More recently, Rani et al. [Rani et al. 2019] proposed a genetic algorithm to generate the test cases automatically for mutation testing. This research employed a selective mutation technique to minimize the mutation testing cost i.e, to create and execute fewer mutants instead of all the traditional mutation operators. An experiment was performed out from a set of mutants in Java language.

4. Approach Description

Test data generation is a fundamental activity to improve the automatizing of mutation testing. These automatizing consists mainly of mutants production, test data generation, programs, and mutants execution using the test sets and test oracle. Excepts for test data generation, the mentioned activities are already automated in several tools. A test data is considered as an effective one if it kills one or more mutants, by producing different outputs from the original program and mutants, thus checking whether the test set can identify the changes injected into the source code. In general, to distinguish the behavior between original program P and a mutant M, a test data t should satisfy three conditions known RIP model, Reachability, Infection, and Propagation [DeMillo and Offutt 1991]:

1. Reachability: t must be able to reach the original statement S and mutated statement S'. If S and S' are not reached by t, t is not guaranteed to kill M.
2. Infection: t must be able to cause different internal states on P and M immediately after executing S and S'. Thus, for t to kill M, it is necessary that S and S' are reached and the state after its execution must be different.
3. Propagation: t must be able to cause the final state of the M to be different from P, i.e., the infected state must propagate to some point in P at which it can be observed. Thus, the different state caused by satisfying the necessity condition must be propagated through the program's execution to produce a different output.

In this context, the proposed approach attempts to automate the mutation-based test data generation for Python programs. This project aims to create a library that utilizes search-based algorithms to produce test data sets and can be employed in different mutation testing tools. As we can see in Figure 1, the process is composed of the following steps: *(1)* searching and generating of test data; *(2)* executing the test suite against mutants; *(3)* analyzing the outputs by the test oracle, removing the killed mutants and

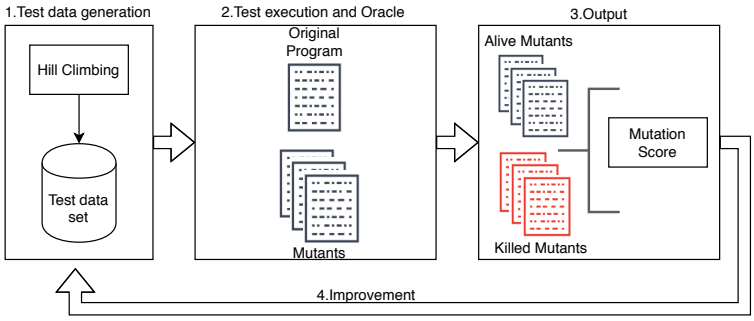


Figure 1. Scheme of the proposed tool

computing of mutation score; e *(4)* return and improving the generated test data from Hill Climbing. This process is repeated until reaching a predefined score threshold.

Hill Climbing is a search algorithm that combines a general search method, i.e., generate and test solutions through objective functions to evaluate the states produced by the method. This algorithm aims to identify the best path to be followed in the search and then, it returns the optimal or a satisfactory result for a given problem. Therefore, the algorithm consists of selecting an initial solution randomly, evaluating it, and improving it step by step, from the investigation of the neighborhood of the current solution [Russell and Norvig 2009].

An important element to achieve optimal results in search-based algorithms is how to assess the candidate solutions, this element is described as objective functions. Objective functions consist of mathematical functions that guide the search to find the best solutions and they are created using specific information on the problem. In this paper, we utilize an objective function based on RIP conditions, more specifically in Reachability and Infection and it is composed into two parts: reach distance and mutation distance, as used in [Souza et al. 2016]. The first part of the function is regarding Reachability (RD) and it guides the search towards the mutation point. The second one called MD is a reference to infection condition and it aims to infect the program state at the mutation point, i.e., making a difference in the execution between the two program versions.

The approach consists of generating test data for Python programs from a Hill Climbing algorithm based on weak mutation, i.e., considering reachability and infection condition to kill strongly mutants. The objective function is composed of Reach distance and Mutation distance which represents the weak mutation. Reach distance is a function that combines two metrics and it has been widely used in previous studies [Korel 1990], [McMinn and Holcombe 2006], and [Papadakis et al. 2010]. This function is formed by metrics used in structural testing described as approach level and branch distance. Approach level measures the distance that a test data needs to cover the targeted statement using the number of the target mutant's control dependent nodes that were not executed by the test data. For nodes in which the execution flow was diverted, the distance for a branch to be taken as true, and it is performed from the values of the variables or constants in the condition statements, this metric is called Branch distance.

The second part of our objective function defined as mutation distance was proposed by Papadakis et. al [Papadakis and Malevris 2013] and it is a generalization of the study introduced by Bottaci [Bottaci 2001] in which a fitness function for a genetic

algorithm in mutation testing was presented. Mutation distance computes how close test data are to expose a difference between the original and mutated statement according to branch distance.

For the tool to handle mutations, we deal with the codes on an Abstract Syntax Tree (AST) and reach branches using branch coverage. Assuming a mutant as a branch, the aim is to generate test data to traverse a tree, in other words, a test data that achieves coverage of the mutant branch means that the reachability condition has been fulfilled. Whether this condition has not been fulfilled the process continues until to reach the mutation. A phase of test data improvements is started utilizing the distance necessary for an input reaching the target branch and this is calculated with branch distance and approach level metrics as presented in [Wegener et al. 2001]. After the reachability condition is achieved, we verify using the same test data generated if the infection condition has been fulfilled. For achieving infection state a test data should lead to a different state on a mutant program in the mutation point and original statement. Finally, if the infection is achieved for a selected mutant, the test data are executed in all mutants to verify how many have been strongly killed. The process continues to improve the test data generate until killed all mutants.

The approach above mentioned was developed into a tool that is accessed using a command terminal as well as the reports produced as illustrated in Figure 2. The main features of the tool is about the following items: i) –function, ii) –method, iii) –int-min and –int-max as presented in Table 2. The tool was developed using as a base a Python implementation of automated test data generation for branch testing [1]. From this implementation, we can deal with mutants as branches to apply weak mutation, thus we developed an extension to execute the original programs and mutant programs to verify if they have been killed from a given test data. It is worth mentioning that our tool has no mutants production, for the experiment we use an external tool.

Table 2. Flags arguments to execute the proposed tool

Flag argument	Function
–function <target function name>	define the function to be tested
–method <Hill Climbing>	define search algorithm
–int-min	minimum value of initial parameters for the technique
–int-max	maximum value of initial parameters for the technique

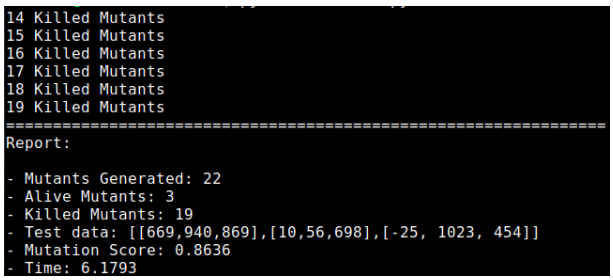


Figure 2. Report from the proposed tool

[1] https://pypi.org/project/covgen/

5. Experimental Study

We conducted experiments to analyze and evaluate the proposed tool for test data generation on a set of Python programs. In this study, the guidelines recommended by Wholin et al. [Wohlin et al. 2012] were used. The experiment was performed through a laptop with Intel Core i7 2.4GHz CPU, 8GB memory in Ubuntu 19.10 operating system.

5.1. Experiment Definition

We used the Goal-Question-Metric (GQM) model [Basili and Weiss 1986] to set out the objectives of the experiment that can be summarized as follows: *"Analyse **proposed tool** for the purpose of **evaluation** with respect to **the mutation-based test data generation** from the point of view of **experimenters** in the context of **the Python programs**"*.

For achieving the goal, we investigate the following Research Question (RQ): **How effective is the proposed tool for test data generation to kill mutants in Python programs?** To answer this RQ, the effectiveness of the tool was measured using the mutation-based test data generation for eight Python programs. We also performed this experiment 30 times computing the average mutation score and the number of test data.

5.2. Procedure of Experiment

To answer the RQ, we carried out the experiment, as follows: *(i)* generating test data to kill the mutants using the proposed approach; and *(ii)* computing mutation score. These experiments were performed in three steps: (1) we chose eight Python programs P = (p_1, p_2,..., p_8) of different size as experimental subjects. Table 3 presents name and LOC of the programs; (2) we generate mutants using mutpy tool[2]; (3) we generated the test data to kill the mutants using the proposed tool; and (4) the total of mutation score is computed.

Table 3. Python programs used in the experiment

Programs	LOC
boolop	25
calender	94
changerMoney	43
coordinates	30
grades	18
orelse	17
triangle	37
typeTriangle	17

5.3. Results

In this section, we answer the RQ presented in Section 5 from the analysis of results concerning the effectiveness of the proposed approach. For this, we compare the proposed tool with a random generation. Table 4 shows the number of mutants (second column), test data generation using proposed tool (T) e (R) random generation (fourth and fifth column), the number of alive mutants by (T) e (R) in sixth column and the number of killed mutants by (T) e (R) in last one.

Analyzing the last column is possible to notice that the performance of the proposed tool was very significant since it killed more than random testing in all programs.

[2]https://pypi.org/project/MutPy/

Table 4. Comparison between the number of killed mutants by the proposed tool (T) and random generation (R)

Programs	Mutants	Equivalentes	Test Data		Alive		Killed	
			T	R	T	R	T	R
boolop	37	3	8	7	4	13	30	21
calendar	87	7	6	11	4	21	76	59
changerMoney	64	8	7	10	9	24	47	32
cordinates	52	4	4	8	12	21	36	27
grades	33	5	4	6	8	11	20	17
orelse	22	3	3	4	0	6	19	13
triangle	20	2	3	5	3	6	15	12
typeTriangle	33	2	5	8	3	7	28	24
Total	348	34	40	59	43	109	271	205
Average	35	3.5	4.5	7.5	4	12	29	22.5

We compared the proposed tool with random testing by evaluating the number of the test data generated capable of killing the largest number of mutants (RQ). The program with the most test data was *calendar* and the program with minimum test data was *orelse*. As expected, the bigger program required more test data than the smaller ones. As presented in Table 4, test data generation using the proposed tool is more efficient than a random generation, since it generates 19 fewer test data with high quality able to kill 77.81% of all mutants. Thus, it was clear that the random generation was unable to kill the whole set of mutants since most of the test data produced are not sufficient.

Figure 3 shows the mutation score achieved by the proposed tool and random generation against the subject programs. As the results show, the proposed tool achieved a mutation score of 22% more than the random generation. In all programs, independent of size, it was possible to observe that the proposed tool obtained better mutation scores than the random generation. Also, we compute the time to generate test data sets and execute it against the set of mutants for each program. Table 5 presents the average time obtained by our tool, the first column indicates the name of programs, the second column reports the average mutation score achieved per program, and the third column shows the average time required for the whole execution.

Table 5. Time and mutation score

Programs	Mutation Score	Time
boolop	0.88	7.0164
calendar	0.95	8.0102
changerMoney	0.84	8.184
cordinates	0.75	6.5821
grades	0.71	7.9134
orelse	1	6.1793
triangle	0.83	5.3258
typeTriangle	0.9	6.7134
Average	0.86	6.8649

We can notice that the test data set generated based on reachability and infection condition is adequate to strongly kill mutants. However, even our benchmark be composed of simple programs, for some mutants are necessary more efforts to find adequate test data, this is due to the difficulty of satisfying more complex requirements or difficulty for a test data to propagate the wrong state to the program's output. Thus, we notice that

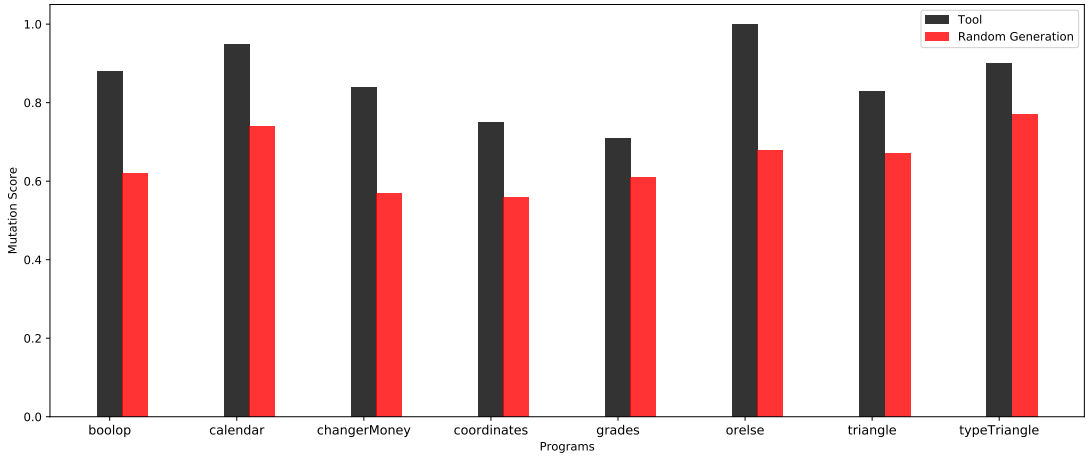


Figure 3. Comparison of MS achieved between the tool and random generation

it is important to add in our objective function the propagation condition.

6. Conclusion

Test data generation is an important activity in software testing, the goal is generating a large number of test data to fulfill testing criteria. Although this activity is known to be performed manually by a Tester, it demands a lot of effort and automation is necessary. For several years a great effort has been devoted to the study of techniques and methods to address this issue, but only a few tools are considered sufficiently adequate to be applied in industrial environments. Generating test data based on mutation for python programs is still a gap, the existing tools focus on mutants generation, engines to execute the test data against mutants and test oracles. An automated approach can lead to the many opportunities to produce quality test data sets and ensuring the cost reduction for mutation testing.

The paper proposes an attempt to automate the test data generation activity through a tool. Our tool is based on approaches presented in previous studies as [Papadakis and Malevris 2013], [Fraser and Arcuri 2015] and [Souza et al. 2016] which use search-based algorithms to generate adequate test data employing the RIP conditions. For guiding the test generation we employed an objective function based on weak mutation, i.e., it is composed of Reachability and Infection conditions to kill strongly mutants. From our preliminary experimental, the results indicate that for toy Python programs the tool was able to kill 86% on average of all mutants and 22% more than random generation. Future work comprises in *i)* conducting additional experiments using real programs, *ii)* increase the impact condition on the objective function and *iii)* finally, we will attempt to provide a tool for Python programs that can be employed in different mutation testing tools which no generates test data automatically.

References

Basili, V. and Weiss, D. (1986). A methodology for collecting valid software engineering data. *IEEE Transactions on Software Engineering*, 10(6):728–738.

Bottaci, L. (2001). A genetic algorithm fitness function for mutation testing. In *Proceedings of the First International Workshop on Software Engineering using Metaheuristic Innovative Algorithms*, pages 3–7, Toronto, Ontario, Canada. IEEE Computer Society.

DeMillo, R. A. and Offutt, A. J. (1991). Constraint-based automatic test data generation. *IEEE Trans. Softw. Eng.*, 17:900–910.

Derezinska, A. and Halas, K. (2014). Experimental evaluation of mutation testing approaches to python programs. In *2014 IEEE Seventh International Conference on Software Testing, Verification and Validation Workshops*, pages 156–164.

Derezinska, A. and Halas, K. (2015). Improving mutation testing process of python programs. In *CSOC*.

Fraser, G. and Arcuri, A. (2015). Achieving scalable mutation-based generation of whole test suites. *Empirical Software Engineering*, 20(3):783–812.

Jatana, N., Suri, B., Misra, S., Kumar, P., and Choudhury, A. R. (2016). Particle swarm based evolution and generation of test data using mutation testing. In *Computational Science and Its Applications – ICCSA 2016*, pages 585–594. Springer International Publishing.

Korel, B. (1990). Automated software test data generation. *IEEE Transactions on Software Engineering*, 16(8):870–879.

McMinn, P. and Holcombe, M. (2006). Evolutionary testing using an extended chaining approach. *Evolutionary Computation*, 14(1):41–64.

Papadakis, M. and Malevris, N. (2010). Automatic mutation test case generation via dynamic symbolic execution. In *2010 IEEE 21st International Symposium on Software Reliability Engineering*, pages 121–130.

Papadakis, M. and Malevris, N. (2013). Searching and generating test inputs for mutation testing. *Springer Plus*, 2(1):1–12.

Papadakis, M., Malevris, N., and Kallia, M. (2010). Towards automating the generation of mutation tests. In *Proceedings of the 5th Workshop on Automation of Software Test (AST)*, pages 111–118, Cape Town, South Africa. ACM.

Rani, S., Dhawan, H., Nagpal, G., and Suri, B. (2019). Implementing time-bounded automatic test data generation approach based on search-based mutation testing. In *Progress in Advanced Computing and Intelligent Engineering*, pages 113–122. Springer Singapore.

Russell, S. and Norvig, P. (2009). *Artificial Intelligence: A Modern Approach*.

Souza, F. C. M., Papadakis, M., Le Traon, Y., and Delamaro, M. E. (2016). Strong mutation-based test data generation using hill climbing. In *Proceedings of the 9th International Workshop on Search-Based Software Testing*, pages 45–54. ACM.

Wegener, J., Baresel, A., and Harmen, S. (2001). Evolutionary test environment for automatic structural testing. *Information and Software Technology*, 43(14):841–854.

Wohlin, C., Runeson, P., Host, M., Ohlsson, M. C., Regnell, B., and Wesslen, A. (2012). *Experimentation in Software Engineering: An Introduction*. Springer-Verlag Berlin Heidelberg, 1st. edition.

Usando o teste ponta a ponta para garantia de confiabilidade de um Sistema Integrado de Gestão: uma prova de conceito

Yury Alencar Lima[1], **Elder de Macedo Rodrigues**[1], **Rafael Alves Paes de Oliveira**[2], **Maicon Bernardino da Silveira**[1]

[1]Universidade Federal do Pampa (UNIPAMPA)
Código Postal 97.546-550 – Alegrete – RS – Brasil

yuryalencar19@gmail.com, elderrodrigues@unipampa.edu.br

bernardino@acm.org

[2]Universidade Tecnológica Federal do Paraná (UTFPR)
Código Postal 85.660-000 – Dois Vizinhos – PR – Brasil

raoliveira@utfpr.edu.br

Abstract. *In order to improve management and decision making, companies tend to organize their information through Integrated Management Systems (ERPs). ERPs must be reliable systems, as they represent the business and influence decision making. However, ensuring the reliability of an ERP through tests is not a trivial task, as they are generally large systems. The use of automated end-to-end testing can be an effective solution. This study presents a proof of concept of the use of end-to-end tests, through an architecture, metrics and ways of performing interactions with each layer of the application. Thus resulting in a reduction in the time needed to resolve and detect errors in the system.*

Resumo. *Com o intuito de melhorar o gerenciamento e tomada de decisões, as empresas tendem a organizar suas informações através dos Sistemas Integrados de Gestão (ERPs). Os ERPs devem ser sistemas confiáveis, pois representam o negócio e influenciam nas tomadas de decisões. Entretanto garantir a confiabilidade de um ERP através de testes não é uma tarefa trivial, pois são geralmente sistemas de grande porte. O uso de testes automatizados de ponta a ponta pode ser uma solução eficaz. Este estudo apresenta uma Prova de Conceito (POC) do uso de testes ponta a ponta, através de uma arquitetura, métricas e métodos de realizar interações com cada camada da aplicação. Resultando assim em uma redução do tempo necessário para resolver e detectar erros no sistema.*

1. Introdução

A crise do *software* que ocorreu por volta da década dos anos de 1960 e foi referente à baixa qualidade das aplicações desenvolvidas, ocasionando algumas vezes até mesmo a inutilização dos sistemas [Naur and Randell 1969]. A fim de aumentar a confiabilidade dos sistemas, a *Engenharia de Software* (ES) surgiu para sistematizar o processo de desenvolvimento, de modo com que a qualidade das aplicações pudesse ser replicável, independentemente do contexto e domínio de aplicação [Sommerville 2011]. Uma das alternativas dentro da ES para garantir a confiabilidade das aplicações é por meio de atividades de verificação, validação e testes de *software*. Dentre as diversas ações de verificação, os testes funcionais têm como finalidade verificar se o sistema implementado funciona como o esperado a partir da especificação [Delamaro et al. 2013]. Esses testes são parte do ciclo de desenvolvimento e demandam um tempo significativo dentro do processo de desenvolvimento, aumentando a qualidade do produto, mas gerando altos custos [Pressman and Maxim 2016].

Com a finalidade de reduzir esses custos é possível realizar a automatização dos testes. A automatização reduz o tempo de verificação quando a adição de uma nova funcionalidade ou manutenção modifica o sistema, pois não são mais necessários testadores para verificar toda a aplicação, apenas executar novamente os *scripts* de teste [Ivory and Hearst 2001]. Entretanto, a automatização de testes de sistema visa identificar erros por meio da interface com o usuário [Delamaro et al. 2013]. Todavia essa prática pode não ser eficaz em garantir a qualidade necessária em alguns contextos, como, por exemplo, em aplicações *web* atuais mais precisas que requerem uma alta confiabilidade [Clerissi et al. 2017].

Um exemplo de *software* que possui a necessidade de testes constantes é o *Sistema Integrado de Gestão* (ERP – do inglês *Enterprise Resource Planning*), uma vez que ele possui todas as regras de negócio e dados da empresa que não podem ser perdidos, pois influenciam nas tomadas de decisão [Dechow and Mouritsen 2005]. Esses sistemas podem originar novos desafios para a automatização como o gerenciamento dos dados de teste, tempo de execução devido a grandes fluxos e imprecisão para delimitar em qual camada da aplicação se encontra o defeito [Palmér and Waltré 2015].

Uma possível abordagem para mitigar esses desafios, aumentar a eficácia e reduzir os custos é o uso de testes de ponta a ponta. Os testes ponta a ponta verificam as funcionalidades do sistema como um todo, analisando não somente a *interface* com o usuário mas também a estrutura interna da aplicação [Palmér and Waltré 2015]. Além do uso do sistema do ponto de vista do usuário, o teste também é capaz de verificar as outras camadas da aplicação como o banco de dados e chamadas à *Interface* de Programação de Aplicativos (API – do inglês *Application Program Interface*), o que pode facilitar a localização do defeito [Paul 2001].

A fim de analisar o impacto no uso dos testes de ponta a ponta na qualidade de ERPs, esse estudo apresenta uma prova de conceito realizada em um módulo crucial de um sistema real de gestão em nuvem. Além das métricas, questões e objetivos, são apresentadas as tecnologias utilizadas, estrutura dos testes, avaliação empírica e resultados. A fim de apresentar uma visão geral do uso de testes de ponta a ponta no cenário real, e quais foram os impactos do seu uso na qualidade de um produto de *software*. Além de viabilizar a aplicação desses testes, através das tecnologias e estrutura utilizada.

Com base nisso, o artigo foi estruturado da seguinte forma: A Seção 2 apresenta o referencial teórico apresentando os testes funcionais e de ponta a ponta, juntamente com as tecnologias utilizadas para a prova de conceito; A Seção 3 apresenta os trabalhos relacionados ao respectivo estudo; A Seção 4 apresenta a arquitetura dos testes ponta a ponta e como foi realizada a Avaliação Empírica; A Seção 5 apresenta os resultados baseado nas métricas juntamente com as ameaças à validade do estudo, e; A Seção 6 apresenta as considerações finais e os trabalhos futuros relacionados.

2. Referencial Teórico e Técnico

Esta seção é responsável por apresentar os conceitos e tecnologias que foram utilizadas para a realização desse estudo. Com base nisso, esta seção foi subdividida em Aspectos Conceituais apresentado na Seção 2.1 e Tecnologias Utilizadas presente na Seção 2.2.

2.1. Aspectos Conceituais

O presente estudo foi baseado nos conceitos de testes funcionais de *software* e testes de ponta a ponta, a fim de aumentar a confiabilidade e qualidade de um Sistema. Normalmente, as equipes de teste buscam o aumento da qualidade de um sistema através da aplicação de diferentes abordagens e técnicas de teste. Por exemplo, o teste funcional visa

revelar falhas e detectar defeitos em uma aplicação, aumentando a qualidade do produto antes da entrega para o cliente [Delamaro et al. 2013]. Além disso, essa prática auxilia na verificação da aplicação em relação ao requisitado pelo cliente [Rios et al. 2007]. Nesses testes, o sistema é considerado uma caixa-preta, ou seja, a aplicação é analisada do ponto de vista do usuário final, não sendo verificada sua estrutura interna, somente seu comportamento [Delamaro et al. 2013]. A especificação desse comportamento é derivada dos requisitos do cliente e transformadas em casos de teste, os quais são um conjunto de condições para a realização dos testes, possuindo alguns elementos principais como, por exemplo, as entradas do sistema, ações a serem realizadas e resultados esperados [Myers et al. 2011]. Assim, possibilita a detecção de problemas funcionais antes do produto ser entregue a seu usuário final, reduzindo os custos relacionados às correções após a entrega da aplicação, sendo esses os mais altos do ciclo de vida de *software* [Delamaro et al. 2013]. Esses conceitos influenciaram na criação dos casos de teste, levando em consideração o ponto de vista do usuário final. Resultando assim na verificação não somente dos comportamentos das telas e sim do fluxo total das ações realizadas pelo cliente.

Assim como o teste funcional, os testes de ponta a ponta também possuem como principal objetivo a detecção de defeitos dentro de uma aplicação, entretanto essa abordagem leva em consideração todas as camadas do *software* [Paul 2001]. Essa abordagem visa identificar falhas em qualquer camada do produto de *software*, por meio da automação e verificação não somente da interação do usuário com a interface, mas também da comunicação com as APIs (*Application Programming Interface*) e banco de dados [Palmér and Waltré 2015]. Assim, os conceitos de teste ponta a ponta surgiram como uma alternativa dentro da POC para localizar precisamente os defeitos da aplicação. Dessa forma, reduzindo tanto no tempo necessário para o desenvolvedor ou testador encontrar a origem do problema, quanto na identificação do impacto do erro dentro da aplicação.

2.2. Tecnologias Utilizadas

A fim de viabilizar a aplicação tanto dos testes funcionais de *software*, quanto dos testes de ponta a ponta automatizados, o presente estudo utilizou um conjunto de ferramentas. Essa seção visa apresentar as tecnologias Robot *Framework*, Selenium WebDriver, Database Library e HTTP Library Requests, bem como detalhar qual sua função na automação dos testes no contexto do trabalho realizado.

O *Robot Framework* é uma tecnologia *Open Source* com o objetivo de realizar a criação de testes de aceitação de sistema. O seu uso é baseado em uma sintaxe tabular e de fácil aprendizado [Bisht 2013]. O *framework* também utiliza de linguagem natural tanto para a definição das *Keywords* que são como métodos de teste, quanto para os *Steps* que são os passos dentro de cada *Keyword*. Dentre os pontos positivos para seu uso, destacam-se a compatibilidade com várias bibliotecas externas e a possibilidade de criação de bibliotecas para domínios específicos [Bisht 2013]. Essa personalização pode ser realizada por meio de *scripts* criados em linguagem Python [Lutz 2001] ou, então em Java [Gosling et al. 2000]. Nessa prova de conceito foi utilizado o *Robot Framework* devido a possibilidade de aplicação em diversos tipos de sistemas como *web*, *desktop* e mobile. A tecnologia foi utilizada juntamente com o Python decorrente da grande quantidade de bibliotecas de suporte para o respectivo domínio. Além disso, o *framework* teve grande influência na escrita dos casos de teste representando o domínio sem restrições, beneficiando também a documentação das funcionalidades testadas.

O Selenium WebDriver é uma tecnologia responsável por realizar a manipulação do navegador web, por meio de interações com a *interface* gráfica da aplicação *web*,

representando as ações do usuário [Avasarala 2014]. A tecnologia captura os elementos gráficos por meio dos atributos presentes dentro do *HyperText Markup Language* (HTML), que por sua vez representam a *interface* com o usuário, viabilizando a realização de manipulações e ações com esses elementos. Sua adoção dentro da prova de conceito foi decorrente da existência de uma biblioteca nativa para o *Robot Framework* e ser uma recomendação da *World Wide Web Consortium* (W3C) sendo esta a principal organização de padronização de aplicações *web* [W3C 2018]. Assim, todas as ações realizadas na aplicação por meio da *interface* gráfica representando o usuário final dentro da POC foram através do Selenium WebDriver.

O *Database Library* é uma biblioteca externa que provê ao *Robot Framework* a realização da comunicação com diferentes tipos de banco de dados [See 2019]. A biblioteca possui duas versões uma para Python e outra focada em Java. Ambas são compostas por uma série de palavras chave que realizam a comunicação com o banco de dados da aplicação [See 2019]. Explorando as palavras reservadas do *Robot Framework* é possível realizar diversas validações nessa camada da aplicação, desse modo sua adoção na verificação do ERP utilizado nessa prova de conceito foi necessária. Viabilizando assim, a adoção dos conceitos de teste ponta a ponta, levando em consideração outra camada da aplicação, sendo essa responsável pela comunicação com o banco de dados.

O *HTTP Library Requests* é uma biblioteca que tem como finalidade realizar chamadas por meio do *HyperText Transfer Protocol* (HTTP) que tem o intuito de contactar serviços implementados pela parte lógica da aplicação, que neste caso estão presente nas APIs utilizadas [Evcimen 2020]. Desse modo, é possível verificar também o processamento com chamadas para a API que provê o serviço. Assim, analisando outra camada geralmente presente nas aplicações *web*. Essa biblioteca também é voltada para o uso por meio do *Robot Framework* nas versões em Python e em Java. Como o *Database Library* essa biblioteca também é composta por um conjunto de palavras chave para simplificar essa atividade provendo uma fácil automação [Evcimen 2020]. Com base nisso também foi necessária para a prova de conceito, a fim de testar os retornos da lógica da aplicação e tornando possível o teste de ponta a ponta do Sistema de Gestão Integrado em Nuvem.

3. Trabalhos Relacionados

Dentre os trabalhos relacionados ao estudo conduzido, é possível citar [Clerissi et al. 2017] que apresenta a importância das aplicações *web* dentro do cotidiano das pessoas e a necessidade de garantia da qualidade desses sistemas. Esse estudo também contém os conceitos de teste de ponta a ponta, frisando sua eficácia dentro do cenário de qualidade. Com base nisso, os autores apresentam uma abordagem com o foco na geração de *scripts* de testes de ponta a ponta, utilizando especificações textuais ou, então baseadas em *Unified Modeling Language* (UML) [Clerissi et al. 2017].

[Debroy et al. 2018] apresenta os desafios relacionados à automação de aplicações *web*, resultando em um conjunto de lições aprendidas e informações técnicas de como gerenciar os testes em ambientes ágeis de desenvolvimento e, assim manter a confiabilidade da aplicação testada [Debroy et al. 2018].

4. Prova de Conceito

Nessa seção são apresentados todos os detalhes relacionados à prova de conceito. Pontualmente, apresentam-se: (1) a estrutura do projeto de testes; (2) o módulo do ERP denominado RadarXML, no qual as tecnologias foram aplicadas e, por fim, (3) a estruturações e condução da avaliação empírica.

4.1. Estrutura para o uso do teste ponta a ponta

A fim de permitir a replicabilidade do projeto de testes para outros domínios e facilitar o uso de padrões de projetos, foi definida a estrutura apresentada nessa seção.

Essa arquitetura conta com a divisão dos testes em seis diretórios que são apresentados a seguir: (1) **config:** diretório que contém os arquivos *Robot* referentes à configuração do projeto com informações generalizadas como *host* e *URL* base da aplicação, usuários e o navegador que será utilizado nos testes; (2) **libs:** Nesse diretório é onde são inseridos os *scripts* em Python que realizam procedimentos do domínio não detectados em outras bibliotecas do Robot Framework, como busca em arquivos JSON, geração de CNPJs, dentre outros; (3) **pages:** Esse diretório é voltado para o uso do padrão de projeto *Page Object*, no qual cada página da aplicação é um arquivo, e todas suas ações são inseridas em formas de métodos, possibilitando um melhor reúso e entendimento dos *scripts* de teste; (4) **services:** Representa o diretório de serviços utilizados dentro dos testes, geralmente utilizados para realizar as pré-condições dos cenários, por meio de interação com o banco de dados, API e *Interface* Gráfica; (5) **tests:** Todos os *Steps* dos testes e *Keywords* são definidas em arquivos neste diretório, o uso de *Keywords* proporciona a definição de ações mais representativas para a validação com o usuário final; (6) **resources:** Esse diretório possui todos os arquivos contendo os cenários de teste utilizando termos do domínio da aplicação, juntamente com uma breve documentação das funcionalidades.

Dentro desses diretórios podem ser criadas outras pastas a fim de organizar as páginas, serviços, cenários e os *Steps* dos testes de acordo com o módulo a ser testado.

4.2. RadarXML

O Sistema Integrado de Gestão em Nuvem, no qual esta prova de conceito foi realizada é um produto real utilizado por mais de 200 clientes. Além disso, este ERP é um sistema completo de gestão para varejo, que requer uma alta confiabilidade. Dentre os módulos existentes na aplicação a POC foi focada apenas no RadarXML. Esse módulo é responsável por consultar em tempo real todas as notas fiscais na Secretaria da Fazenda direcionadas à empresa que utiliza o sistema. Partindo disso, o estoque, preços dos produtos, parte financeira, contábil e fiscal são atualizadas gerando uma série de indicadores e relatórios. Além disso, também é possível realizar a importação manual de notas fiscais, ajustes de fornecedores, produtos e faturamento por meio do módulo.

4.3. Avaliação Empírica

Essa seção apresenta as informações relacionadas à Avaliação Empírica, como o *Design* Experimental, Objetivos e Métricas e por fim os procedimentos realizados para facilitar a análise dos resultados encontrados.

4.3.1. *Design* Experimental

O ERP utilizado na avaliação impõe restrições aos testes realizados somente através da *interface* gráfica da aplicação, como não permitir a exclusão de nenhum usuário, nota fiscal, fornecedor, dentre outras informações presentes. Isto acontece para possibilitar uma maior integridade e disponibilidade dos dados. Entretanto, isso implica que os testes sempre deveriam inserir novos dados geralmente aleatórios, mas alguns dados como notas fiscais acabam sendo mais complicados de serem gerados automaticamente. Uma alternativa seria realizar sempre a limpeza do banco de dados antes de realizar a execução de cada teste, todavia esta prática não é viável devido ao tempo de espera para a base de

dados voltar a ficar disponível, levando em consideração as características da infraestrutura onde estas informações estão hospedadas. Assim, o teste de ponta a ponta foi uma solução idealizada tanto para mitigar o problema de gerenciamento de dados de teste, quanto para utilizar dados reais nas execuções, simulando assim ações similares a aquelas executadas pelos usuários reais. A fim de reduzir o esforço para a aplicação da solução proposta foram escolhidas as tecnologias apresentadas na Seção 2.2.

4.3.2. Objetivos e Métricas

O objetivo dessa prova de conceito foi melhorar a qualidade e confiabilidade de ERP utilizado por diversos clientes, por meio de testes ponta a ponta, verificando cada camada da aplicação. A fim de analisar o uso dos testes ponta a ponta e das tecnologias escolhidas foram definidas as seguintes questões (Q): **Q01:** Quantos erros foram detectados no módulo RadarXML devido ao uso dos testes de ponta a ponta? **Q02:** Qual a diferença no tempo necessário para realizar a manutenção dos *scripts* de teste, comparando os testes de ponta a ponta e funcionais tradicionais?

Com a finalidade de responder a **Q01** foi utilizado o número de erros encontrados. Por fim a **Q02** pode ser analisada com base no tempo em horas necessárias para resolver um problema em comum, tanto na automação ponta a ponta, quanto na realizada somente por meio da *Interface* Gráfica.

4.3.3. Procedimentos

Os procedimentos que serão apresentados foram executados durante duas homologações distintas do ERP em nuvem na empresa responsável pelo produto. A quantidade de erros encontrados por meio dos testes de ponta a ponta, referente à **Q01**, foram coletadas com base nos resultados dos automatizados e computadas em uma planilha somente após a confirmação do defeito. Devido à grande quantidade de testes e diversas alterações para serem verificadas, na segunda homologação foram detectados falsos positivos provocados por alterações em funcionalidades já automatizadas. As métricas referentes à **Q02**, foram coletadas por meio do tempo de detecção e correção dos defeitos encontrados nos testes automatizados. Nessa atividade foram necessários dois automatizadores. As informações coletadas foram armazenadas em uma planilha somente após o teste corrigido ser executado conforme o comportamento esperado.

5. Resultados e Discussões

Após a execução dos procedimentos acima, foi realizada uma análise dos testes de ponta a ponta no impacto na qualidade. O resultado dessa análise e as ameaças à validade da prova de conceito são apresentados nessa respectiva seção.

5.1. Análise de Impacto na Qualidade

Durante as homologações foram encontrados cinco erros no módulo RadarXML, que foi considerado como um bom resultado, tendo em vista que antes de chegar à esta etapa já haviam sido executados diversos testes em ambientes de desenvolvimento e de qualidade. Além disso, como a empresa utiliza de práticas ágeis de desenvolvimento como entrega e integração contínua, o tempo entre as homologações é relativamente menor, podendo levar de uma a duas semanas incluindo o tempo destinado a implementação das funcionalidades. Sabendo disso, os erros encontrados pelos testes de ponta a ponta não foram somente relativos à *Interface* Gráfica da aplicação, o que impactou na redução do tempo

Métrica	Testes de Ponta a Ponta	Testes Funcionais
Quantidade de Erros Encontrados	5	0
Tempo de Manutenção	30 minutos	1 hora

Tabela 1. Métricas Coletadas

de correção, devido ao relatório de execução apresentar em qual camada foi manifestada a falha. Assim, respondendo à **Q01**.

Com a atualização constante do ERP, nesse espaço de tempo foram detectados falsos negativos. Com base nisso, foi possível realizar a medição em relação a manutenção dos testes ponta a ponta com *Robot Framework*, com os testes funcionais utilizando o *framework Cucumber*. O que resultou em uma diferença de tempo de trinta minutos a mais para resolver o mesmo problema utilizando o *Cucumber*, respondendo à **Q02**. Isto aconteceu pois como a aplicação não permite a exclusão dos dados, os testes realizados somente através da *Interface* Gráfica utilizavam dados gerados automaticamente. O que aumentou a complexidade para o entendimento do cenário através da execução dos testes. O uso de dados gerados aleatoriamente também dificultou a análise do cenário dentro da aplicação, pois é necessário saber exatamente quais eram estes dados e o que representavam. Enquanto nos testes de ponta a ponta todas as pré-condições foram realizadas diretamente no banco de dados ou com chamadas de serviços através das APIs. O que tornou não só o teste mais rápido mas também possibilitou a inserção de dados reais e representativos que foram facilmente encontrados. As métricas coletadas podem ser visualizadas através da Tabela 1.

5.2. Ameaças à Validade

Naturalmente, devido ao escopo específico dos estudos, diversas ameaças à validade foram ponderadas. Considerando as ameaças de *Construção*, que considera a relação entre a teoria proposta e os resultados observados, estes pesquisadores consideram a existência de ameaças baixas e poucas chances da existência de alguma relação casual entre a implementação da estratégia automatizada de teste e os defeitos revelados. Essa mesma análise é estendida para a avaliação das ameaças às validades *Internas* do estudo, apesar de não terem sido realizadas avaliações estatísticas, acredita-se que o tratamento realizado teve influência direta nos resultados aferidos. No tocante às ameaças *Externas* (generalização de resultados fora do escopo estudado), estes autores consideram as ameaças como sendo médias e moderadas. É importante destacar que estudos mais amplos precisam ser realizados para aferir a generalização dos resultados e da aplicabilidade da estratégia e diminuição de tais ameaças. Sendo assim, a generalização do trabalho configura o principal ponto a ser melhorado em trabalhos futuros nessa mesma linha investigativa. Por fim, considerando as ameaças à *Conclusão* do estudo, é possível afirmar que tais ameaças são consideradas baixas, haja vista que os resultados e os defeitos na aplicação industrial utilizada como cobaia somente puderam ser revelados a partir da estratégia de teste automatizada proposta.

6. Considerações Finais

Com base nos impactos detectados na qualidade do ERP em nuvem, o uso dos testes de ponta a ponta se mostrou eficaz para o respectivo contexto. O tempo de execução, detecção do local do defeito e uso de dados reais e representativos foram os pontos fortes encontrados do uso destas tecnologias. Entretanto, a sua aplicação pode requisitar um grupo de testadores com mais conhecimento relacionado a automação e à arquitetura do sistema ser testado. Baseado nos benefícios encontrados o uso dos testes de ponta a ponta

nas aplicações web ainda serão utilizados, e como trabalhos futuros foi planejado novas provas de conceito do seu uso em outros tipos de *software* como *mobile* e *desktop*.

Referências

Avasarala, S. (2014). *Selenium WebDriver practical guide*. Packet.

Bisht, S. (2013). *Robot framework test automation*. Packet.

Clerissi, D., Leotta, M., Reggio, G., and Ricca, F. (2017). Towards the generation of end-to-end web test scripts from requirements specifications. In *IEEE 25th International Requirements Engineering Conference Workshops*, pages 343–350. IEEE.

Debroy, V., Brimble, L., Yost, M., and Erry, A. (2018). Automating web application testing from the ground up: Experiences and lessons learned in an industrial setting. In *IEEE 11th International Conference on Software Testing, Verification and Validation*, pages 354–362.

Dechow, N. and Mouritsen, J. (2005). Enterprise resource planning systems, management control and the quest for integration. *Accounting, organizations and society*, pages 691–733.

Delamaro, M., Jino, M., and Maldonado, J. (2013). *Introdução ao teste de software*. Elsevier Brasil.

Evcimen, B. (2020). HTTP Library (Requests). `https://github.com/bulkan/robotframework-requests/#readme`. Online; acessado 28 Fevereiro de 2020.

Gosling, J., Joy, B., Steele, G., and Bracha, G. (2000). *The Java language specification*. Addison-Wesley Professional.

Ivory, M. Y. and Hearst, M. A. (2001). The state of the art in automating usability evaluation of user interfaces. *ACM Computing Surveys (CSUR)*, pages 470–516.

Lutz, M. (2001). *Programming python*. "O'Reilly Media, Inc.".

Myers, G. J., Sandler, C., and Badgett, T. (2011). *The art of software testing*. John Wiley & Sons.

Naur, P. and Randell, B. (1969). Software engineering: Report of a conference sponsored by the nato science committee, garmisch, germany, 7th-11th october 1968. *NATO Science Committee*.

Palmér, T. and Waltré, M. (2015). Automated end-to-end user testing on single page web applications.

Paul, R. (2001). End-to-end integration testing. In *Proceedings Second Asia-Pacific Conference on Quality Software*, pages 211–220. IEEE.

Pressman, R. and Maxim, B. (2016). *Engenharia de Software*. McGraw Hill Brasil.

Rios, E., Cristalli, R., Moreira, T., and Bastos, A. (2007). Base de conhecimento em teste de software. 2ª edição S. Paulo.

See, F. A. V. (2019). Database Library. `http://franz-see.github.io/Robotframework-Database-Library/`. Online; acessado 29 Fevereiro de 2020.

Sommerville, I. (2011). *Software Engineering 9th Edition*. Pearson.

W3C (2018). WebDriver W3C Recommendation. `https://www.w3.org/TR/webdriver1/`. Online; acessado 29 Fevereiro de 2020.

Engenharia de Software em Empresas de Pequeno e Médio Porte: Um Mapeamento Sistemático

Márcio Vitor dos Santos[1], **Wesley K. G. Assunção**[1], **Ivonei Freitas da Silva**[2]

[1]COTSI – Universidade Tecnológica Federal do Paraná (UTFPR). Toledo, Brasil.

[2]Universidade Estadual do Oeste do Paraná (UNIOESTE). Cascavel, Brasil.

mvs_designer@hotmail.com,wesleyk@utfpr.edu.br,ivonei.silva@unioeste.br

Resumo. *A engenharia de software (ES) estabelece práticas para auxiliar o desenvolvimento de software com qualidade, custo e prazos controlados. Apesar das práticas da ES serem amplamente conhecidas, pequenas e médias empresas (PMEs) por vezes enfrentam dificuldades em aplicá-las. Para identificar os estudos existentes que abordam a aplicação de práticas da ES no contexto de PMEs, este trabalho descreve um mapeamento sistemático da literatura para um melhor entendimento do corpo de conhecimento de ES no contexto de PMEs. Deseja-se obter uma visão geral desta área e direcionar pesquisas futuras neste tópico. Os resultados apontaram que as PMEs de fato preocupam-se em adotar práticas da ES, mas essas práticas muitas vezes não estão adequadas as suas características. Pode-se concluir que novos estudos devem ser conduzidos em tópicos relacionados à ES para atender as necessidades das PMEs.*

1. Introdução

A produção de software possui uma participação relevante na economia de muitos países. Por exemplo, de acordo com a Associação para Promoção da Excelência do Software Brasileiro (SOFTEX), o Brasil é o sétimo maior mercado de tecnologia da informação do mundo, representando aproximadamente 200 bilhões de reais em receita, envolvendo mais de 415000 funcionários empregados na área [SOFTEX 2019]. Um estudo da Associação Brasileira das Empresas de Software (ABES) apontou que em 2019, das 5519 empresas que atuam principalmente com desenvolvimento de software, cerca de 95% são classificadas como micro e pequenas empresas (até 99 funcionários) [ABES 2020]. Essas Pequenas e Médias Empresas (PMEs) têm grande importância no cenário sócio-econômico de qualquer país [de Barros Sampaio et al. 2015].

Apesar da expressividade na quantidade de PMEs, tais empresas possuem portfólio reduzido de produtos, poucos recursos humanos, muitas vezes não empregam processos de produção de software adequados, e apresentam um grande intervalo entre a concepção e a comercialização de seus produtos de software. Apesar das atividades propostas na Engenharia de Software (ES) contornarem esses problemas, ainda existe uma lacuna entre teoria e prática [Sánchez-Gordón and O'Connor 2016].

Há muitas técnicas, métodos, ferramentas, processos e práticas ofertadas pela ES para que as empresas e desenvolvedores possam gerenciar, desenvolver e manter seus produtos com foco nos interesses do mercado. E, como consequência do uso da ES, o desenvolvimento de software poderá ocorrer de forma otimizada, aumentando a produtividade e qualidade do produto final. Contudo, PMEs têm um cenário com características

distintas, como o número limitado de funcionários, a estrutura organizacional plana, uma infraestrutura limitada e a flexibilidade que as governam podem dificultar a aplicação das práticas da ES, inclusive essas PMEs podem não reconhecer as ferramentas e processos adequados para as suas atividades [Degwitz 2014]. Como resultado, as PMEs podem encarar vários desafios. Por exemplo, há poucos recursos para dar suporte ao investimento e crescimento, assim há resistência na adoção da ES, gerando um processo deficitário desde o desenvolvimento até a entrada do produto no mercado [O'Connor and Coleman 2009, Sánchez-Gordón and O'Connor 2016, Moll 2013, Majchrowski et al. 2016].

Com base no cenário exposto, este trabalho tem por objetivo descrever um mapeamento sistemático acerca da ES em PMEs. Deseja-se contribuir para uma melhor compreensão do cenário de utilização de práticas da ES em PMEs e identificar o direcionamento para novas pesquisas na área. O mapeamento sistemático (Seção 2) visa identificar estudos que tem como foco o relato do uso de práticas da ES em PMEs. Foram encontrados 26 estudos que serviram como base para uma análise acerca dos problemas que as PMEs tiveram na adoção de práticas da ES, das barreiras encontradas para esta adoção e dos benefícios que estas empresas tiveram com esta adoção. Os resultados (Seção 3) apontaram que as PMEs de fato preocupam-se em adotar práticas da ES, mas essas práticas muitas vezes não estão adequadas às suas características. Foi possível observar que nossos achados complementam os trabalhos existentes na literatura (Seção 4).

2. Processo do Mapeamento Sistemático

Um mapeamento sistemático é um método baseado em evidência usado para construir ou classificar um corpo de conhecimento em um tópico de interesse. O objetivo é obter uma visão geral e identificar lacunas e tendências de pesquisa para o tópico em questão [Petersen et al. 2015]. O mapeamento descrito neste artigo seguiu as diretrizes definidas por [Petersen et al. 2015], que são: (1) definição de questões de pesquisa, (2) busca e filtro dos estudos, (3) definição de um esquema de classificação, e (4) coleta e análise dos dados. Esses passos são apresentados em detalhes nas próximas seções.

2.1. Questões de Pesquisa

Deseja-se obter evidências na literatura para responder as seguintes questões de pesquisa:

- **Questão 1:** *Quais práticas de ES são comumente adotadas pelas PMEs?* Deseja-se identificar práticas da ES que as PMEs aplicam durante o desenvolvimento de software. Possivelmente, muitas práticas são adotadas, assim, queremos entender o quadro geral do desenvolvimento de software em PMEs.
- **Questão 2:** *Quais os benefícios/melhorias que são relatados sobre a aplicação das práticas de ES em PMEs?* Deseja-se investigar os benefícios e as melhorias que as práticas da ES podem prover para PMEs, motivando a sua ampla adoção.
- **Questão 3:** *Quais os desafios/barreiras que são relatados sobre a aplicação das práticas de ES em PMEs?* PMEs talvez observam pontos negativos ao adotar práticas da ES. Identificar e discutir esse pontos pode ajudar a motivar ou fortalecer pesquisas em tópicos específicos.

2.2. Busca e Filtro dos Estudos

A partir das questões de pesquisa - e considerando que o termo *"software engineering"*, quando presente no estudo primário, é amplo o suficiente para contemplar também demais

termos como atividade e valor/vantagem, neste caso, sinônimos para práticas e benefícios - estabeleceu-se três palavras-chave. Estas palavras e seus sinônimos que foram usados para a busca são apresentadas na Tabela 1. A partir desses termos, a seguinte consulta foi definida: *("Software Engineering"OR "SE") AND ("Small and medium-sized enterprises"OR "Small and medium enterprises"OR "SME"OR "SMEs"OR "Small medium enterprises"OR "Small enterprises"OR "Medium enterprises") AND (challenge OR limitation OR constraint OR barrier).*

Palavras-chave	Termos e Sinônimos
Engenharia de Software	Software Engineering, SE.
Pequenas e Médias Empresas	Small and medium-sized enterprises, Small and medium enterprises, SME, SMEs, Small medium enterprises, Small enterprises, Medium enterprises.
Limitações	Challenge, Limitation, Constraint, Barrier.

Tabela 1. Termos usados na busca de estudos

A consulta foi utilizada em três indexadores amplamente utilizados em estudos sistemáticos, como apresentado na Tabela 2. A última coluna desta tabela mostra a quantidade de estudos encontrados ao executar a consulta.

Indexador	URL	Resultado
SCOPUS	`https://www.scopus.com`	88
IEEE Xplore	`https://ieeexplore.ieee.org/`	61
ACM Digital Library	`https://dl.acm.org/`	22

Tabela 2. Indexadores utilizados e quantidade de estudos encontrados

Foram encontrados um total de 171 estudos que passaram por quatro filtros, conforme descritos a seguir: (i) leitura do título, resumo e palavras-chave para remover estudos que não eram relevantes para o objetivo do estudo (*97 mantidos e 74 removidos*), (ii) remoção de estudos repetidos encontrados em diferentes indexadores (*78 mantidos e 19 removidos*), (iii) remoção de literatura cinzenta. Documentos nessa categoria são: cartas editoriais, resumos de palestras, prefácio de conferências, slides de apresentações, propostas e estudos em andamento (*61 mantidos e 17 removidos*), e (iv) leitura da introdução e conclusão para identificar os estudos com informações suficientes para responder as questões de pesquisa (*26 mantidos e 35 removidos*). Ao final, os 26 estudos apresentados na Tabela 3 foram selecionados para análise e classificação dos dados.

2.3. Esquema de Classificação

A Tabela 4 apresenta as dimensões e categorias utilizadas para coletar os dados e classificar os estudos identificados como relevantes após a busca de filtro. As categorias foram definidas considerando as descrições nos estudos primários selecionados, focando nas atividades realizadas e não em fases (atividades macros), tais como "desenvolvimento"ou "manutenção". As categorias apresentadas nessa tabela foram utilizadas para as análises quantitativas. Além dessa classificação, também foram coletadas informações sobre como as práticas mencionadas foram aplicadas pelas PMEs, as quais foram usadas para as análises qualitativas.

ID	Referência
AC-003	JANES, Andrea; LENARDUZZI, Valentina; STAN, Alexandru Cristian. A continuous software quality monitoring approach for small and medium enterprises. In: Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering Companion. ACM, 2017. p. 97-100.
AC-004	HLAD, Nicolas. Facilitating the development of software product lines in small and medium-sized enterprises. In: Proceedings of the 23rd International Systems and Software Product Line Conference-Volume B. ACM, 2019. p. 95.
AC-005	SETH, Ashish; AGARWAL, Himanshu; SINGLA, Ashim Raj. Designing a SOA based model. ACM SIGSOFT Software Engineering Notes, v. 36, n. 5, p. 1-7, 2011.
AC-006	HERING, Dominik et al. Integrating usability-engineering into the software developing processes of SME: a case study of software developing SME in Germany. In: Proceedings of the Eighth International Workshop on Cooperative and Human Aspects of Software Engineering. IEEE Press, 2015. p. 121-122.
AC-007	THÖRN, Christer; GUSTAFSSON, Thomas. Uptake of modeling practices in SMES: initial results from an industrial survey. In: Proceedings of the 2008 international workshop on Models in software engineering. ACM, 2008. p. 21-26.
AC-008	BASTOS, Jonatas Ferreira et al. Software product lines adoption: an industrial case study (keynote). In: Proceedings of the Third International Workshop on Conducting Empirical Studies in Industry. IEEE Press, 2015. p. 35-42.
AC-009	JONES, Stephen; NOPPEN, Joost; LETTICE, Fiona. Management challenges for DevOps adoption within UK SMEs. In: Proceedings of the 2nd International Workshop on Quality-Aware DevOps. ACM, 2016. p. 7-11.
AC-017	SELVARANI, R.; MANGAYARKARASI, P. A Dynamic Optimization Technique for Redesigning OO Software for Reusability. ACM SIGSOFT Software Engineering Notes, v. 40, n. 2, p. 1-6, 2015.
IEEE-006	BESROUR, Souhaib; RAHIM, Lukman Bin AB; DOMINIC, P. D. D. A quantitative study to identify critical requirement engineering challenges in the context of small and medium software enterprise. In: 2016 3rd International Conference on Computer and Information Sciences (ICCOINS). IEEE, 2016. p. 606-610.
IEEE-012	BUCHAN, Jim; EKADHARMAWAN, Christian Harsana; MACDONELL, Stephen G. Insights into domain knowledge sharing in software development practice in SMEs. In: 2009 16th Asia-Pacific Software Engineering Conference. IEEE, 2009. p. 93-100.
IEEE-013	AKBAR, Rehan; HASSAN, Mohd Fadzil; ABDULLAH, Azrai. A framework of software process tailoring for small and medium size IT companies. In: 2012 International Conference on Computer and Information Science (ICCIS). IEEE, 2012. p. 914-918.
IEEE-018	BASHARAT, Iqra et al. Requirements engineering practices in small and medium software companies: An empirical study. In: 2013 Science and Information Conference. IEEE, 2013. p. 218-222.
IEEE-024	RECH, Jörg; BOGNER, Christian; HAAS, Volker. Using wikis to tackle reuse in software projects. IEEE software, v. 24, n. 6, p. 99-104, 2007.
IEEE-030	SULAYMAN, Muhammad; MENDES, Emilia. An extended systematic review of software process improvement in small and medium web companies. In: 15th Annual Conference on Evaluation and Assessment in Software Engineering (EASE 2011). IET, 2011. p. 134-143.
IEEE-031	BORGES, Pedro; MONTEIRO, Paula; MACHADO, Ricardo J. Tailoring RUP to small software development teams. In: 2011 37th EUROMICRO Conference on Software Engineering and Advanced Applications. IEEE, 2011. p. 306-309.
IEEE-034	BESROUR, Souhaib; RAHIM, Lukman Bin AB; DOMINIC, P. D. D. Investigating requirement engineering techniques in the context of small and medium software enterprises. In: 2016 3rd international conference on computer and information sciences (ICCOINS). IEEE, 2016. p. 519-523.
IEEE-043	KNAUBER, Peter et al. Applying product line concepts in small and medium-sized companies. IEEE Software, v. 17, n. 5, p. 88-95, 2000.
IEEE-045	SCOTT, Louise et al. Practical software process improvement-the IMPACT project. In: Proceedings 2001 Australian Software Engineering Conference. IEEE, 2001. p. 182-189.
IEEE-057	AKBAR, Rehan et al. Software development process tailoring for small and medium sized companies. In: 2014 International Conference on Computer and Information Sciences (ICCOINS). IEEE, 2014. p. 1-5.
SC-005	NOMAN, Muhammad; TAHIR, Touseef; RASOOL, Ghulam. A systematic mapping study on success factors of implementing measurement processes in SMEs. In: 2018 International Conference on Advancements in Computational Sciences (ICACS). IEEE, 2018. p. 1-8.
SC-008	TAHIR, Touseef; RASOOL, Ghulam; NOMAN, Muhammad. A systematic mapping study on software measurement programs in SMEs. e-Informatica Software Engineering Journal, v. 12, n. 1, 2018.
SC-027	SANCHEZ-GORDON, Sandra; SÁNCHEZ-GORDÓN, Mary-Luz; LUJÁN-MORA, Sergio. Towards an Engineering Process for Developing Accessible Software in Small Software Enterprises. In: ENASE. 2016. p. 241-246.
SC-032	KOUZARI, Elia et al. Critical success factors and barriers for lightweight software process improvement in agile development: A literature review. In: 2015 10th International Joint Conference on Software Technologies (ICSOFT). IEEE, 2015. p. 1-9.
SC-034	DE BARROS SAMPAIO, Suzana Cândido et al. Reflecting, adapting and learning in small software organizations: an action research approach.
SC-046	MIRNA, Munoz et al. Expected Requirements in Support Tools for Software Process Improvement in SMEs. In: 2012 IEEE Ninth Electronics, Robotics and Automotive Mechanics Conference. IEEE, 2012. p. 135-140.
SC-076	CATER-STEEL, Aileen; ROUT, Terry; TOLEMAN, Mark. Short and long-term impacts of SPI in small software firms. In: Proceedings of the 17th Australasian Conference on Information Systems (ACIS 2006). Australasian Association for Information Systems, 2006.

Tabela 3. Lista dos estudos primários selecionados.

Dimensão	Categorias
Práticas da ES	Engenharia de Requisitos. Design de software. Teste. Codificação. Métricas. Gestão. Processo. Reutilização.
Benefícios	Produtividade. Qualidade. Custos. Maturidade.
Limitações	Burocracia. Custo. Esforço. Complexidade. Irreal para a empresa. Sem suporte de ferramenta.

Tabela 4. Esquema de classificação utilizado para a coleta de dados dos estudos

3. Resultados e Análises

Este artigo apresenta os resultados com uma análise quantitativa (Seção 3.1) e qualitativa (Seção 3.2) dos 26 estudos primários. Essas análises fornecem as respostas às questões de pesquisas previamente mencionadas neste artigo.

3.1. Análise Quantitativa

Esta análise compreende a categorização dos 26 estudos primários em quatro dimensões, conforme apresentado na Tabela 4.

3.1.1. Práticas da Engenharia de Software

As oito categorias desta dimensão obtiveram o seguinte resultado: *Processo de Software*, com 11 (42,3%) estudos mapeados; *Reutilização de Software*, com cinco (19,2%) estudos mapeados; *Engenharia de Requisitos*, com quatro (15,4%); *Design de Software*, com três (11,5%) estudos mapeados; *Métricas*, com dois (7,7%) estudos mapeados; *Codificação*, com um (3,8%) estudo mapeado; e, por fim, *Teste de Software* e *Gestão de Software* que não tiveram estudos mapeados. É possível observar que o foco dos trabalhos analisados está em processo de software. Esses estudos apontam a relevância de um processo bem definido e sistemático para o desempenho e motivação das equipes. A reutilização é justificada pela redução dos custos à longo prazo, principalmente na produção das variantes de um mesmo produto de software.

3.1.2. Benefícios para as PMEs

Os resultados para as categorias desta dimensão são: *Produtividade*, com dez (38,5%) estudos mapeados; *Qualidade*, com nove (34,6%) estudos mapeados; *Custos*, com seis (23,1%) estudos mapeados; e, por fim, *Maturidade*, com um (3,8%) estudo mapeado. Nota-se que produtividade, qualidade e custos representam mais de 96% dos estudos.

3.1.3. Desafios/Barreiras para as PMEs

As sete categorias desta dimensão obtiveram o seguinte resultado: *Custo*, com 11 (42,3%) estudos mapeados; *Complexidade*, com sete (26,9%) estudos mapeados; *Esforço* e *Irreal para a empresa*, com um (3,8%) estudo mapeado para cada categoria; e, por fim, *Burocracia* e *Sem Suporte de Ferramenta* não foram citadas nos estudos. Nota-se que as barreiras

relacionadas ao custo e complexidade das atividades de ES para PMEs dominam (≈70% dos estudos) a preocupação dessas organizações quanto a sua adoção.

3.2. Análise Qualitativa

A análise qualitativa apresenta uma descrição textual sobre as discussões dos autores dos estudos considerando as três questões de pesquisa apresentadas na seção 2.1.

3.2.1. Práticas da Engenharia de Software

Em quase todos os estudos primários foi possível identificar a adoção de técnicas, métodos, ferramentas, processos ou soluções adaptadas da ES. O que chama a atenção é a ausência de atividades relacionadas aos testes e gestão de software. Um levantamento de dados em campo através de um *survey* com as PMEs fornecerá mais evidências para caracterizar as atividades realizadas, mas principalmente, o motivo da não adoção de atividades relacionadas ao testes ou gestão de software, por exemplo. Para as atividades adotadas, os autores deixam claro a necessidade de adaptações para as características das PMEs.

3.2.2. Benefícios da Engenharia de Software para as PMEs

Redução do custo de desenvolvimento é citado como benefício dado pela ES às PMEs (AC-004, AC-005, AC-006, AC-017, IEEE-024 e IEEE-043). Alguns autores associam redução do custo de desenvolvimento à rápida recuperação dos recursos investidos (AC-008 e SC-032) que pode estar associado à redução no tempo de comercialização (IEEE-030) ou no aumento da produtividade (IEEE-012 e IEEE-018). Os autores que tratam de Linha de Produto de Software (LPS) associam como benefícios a facilidade de gerar novas variantes de produtos (AC-004 e AC-008), distribuindo o custo e esforço de desenvolvimento em vários sistemas e não em um único sistema (IEEE-043), por promover uma reutilização em larga escala ao desenvolver componentes ou funcionalidades comuns ou similares em vários sistemas.

Outro benefício apontado é a diminuição com gastos de manutenção (AC-003, AC-009, AC-017 e IEEE-018). A baixa manutenção é um pré-requisito para melhora na qualidade, mencionada por vários estudos (AC-004, AC-008, AC-009, AC-017, IEEE-012, IEEE-024, IEEE-030, IEEE-057, SC-005, IEEE-006, IEEE-031 e SC-027). A Engenharia de Requisitos (ER) é tratado em quatro estudos primários desta amostragem (IEEE-006, IEEE-012, IEEE-018 e IEEE-034) todavia o tema também surge em documentos cujo o foco da pesquisa não seja especificamente a ER (AC-007) onde os autores afirmam que requisitos bem elaborados são fundamentais para as demais etapas do desenvolvimento, aumentando qualidade e gestão de riscos, em especial àquelas que estão iniciando suas atividades.

Em relação à melhoria nos processos, autores relatam que esta torna a atividade de desenvolvimento mais visíveis para as equipes (SC-034), oferece estabilidade ao progresso das atividades (IEEE-057) de forma mais planejada e estruturada, conscientizando sobre a importância de processos bem definidos (IEEE-045 e AC-008), produzindo uma abordagem mais disciplinada com tarefas bem definidas e distribuídas (IEEE-031).

Melhoria na documentação após a adoção de práticas da ES (AC-007, AC-008, IEEE-045, SC-076) é destacada por alguns autores. Estes mencionam que a documentação bem elaborada evita erros, previne problemas, traz uma estimativa de prazo mais precisa e ajuda a definir processos mais adequados para as fase do desenvolvimento.

Benefícios aos recursos humanos das empresas oriundos da ES são citados por autores em sete estudos primários. É dito que a aplicação de práticas de ES melhora as atividades organizacionais da empresa (IEEE-045), fazendo com as equipes busquem o aperfeiçoamento (SC-046) ampliando seus conhecimentos e os horizontes da empresa. As equipes sentem-se mais motivadas (IEEE-024), pois há uma melhoria significativa no planejamento (SC-076) e na comunicação (SC-046), aumentando assim a autoestima e confiança (SC-032), minimizando os riscos (IEEE-057), aumentando a maturidade e a satisfação dos clientes (IEEE-030).

3.2.3. Problemas/Barreiras/Limitações da Engenharia de Software para as PMEs

Dezesseis estudos primários citam a limitação de recursos (humanos, temporal, físicos, tecnológicos ou monetários) como principal obstáculo para as PMEs adotarem as soluções da ES. Por exemplo, SC-046 os autores citam "um orçamento muito limitado"; AC-007 mencionam restrições em termos de "tempo, dinheiro e equipe" além de ausência de "ferramentas e métodos apropriados" que possam ser rapidamente adotados de forma correta; AC-003, AC-007, IEEE-013, IEEE-57, SC-005, SC-032, SC-046 e IEEE-030 mencionam limitações monetárias pois elas têm economia vulnerável. Esta vulnerabilidade pode ser potencializada nas novas PMEs, onde ainda não há receitas e os investimentos são limitados. Três estudos primários (AC-004, AC-008 e SC-005) destacam o "investimento inicial" como uma barreira para adoção de práticas da ES. Em AC-017 o estudo indica "custo elevado", porém ele se refere a todo o custo de adoção das atividades da ES.

Recursos humanos é uma barreira tratada como "equipe pequena", "acúmulo de funções pelos colaboradores", "falta de experiência", "resistência às mudanças por parte dos experientes" e "imaturidade dos profissionais", mencionada nos estudos AC-003, AC-004, AC-005, AC-007, AC-017, IEEE-057, SC-008, SC-032, SC-034, SC-046 e SC-076. Em SC-076 vale salientar que os autores citam como barreiras o comprometimento da alta administração e o envolvimento, tempo e treinamento da equipe.

Outra barreira identificada na amostragem desta pesquisa é o fato de existirem poucos estudos e pesquisas do meio científico voltados às características especiais das PMEs (AC-005). Esta barreira pode-se associar também à falta de ferramentas e/ou técnicas adequadas para a realidade delas (IEEE-031 e IEEE-034). Temos ainda autores afirmando que as soluções da ES foram elaboradas para grandes empresas e não contemplam as necessidades das PMEs (IEEE-043, IEEE-045 e SC-032).

Alguns autores ainda citam como barreira a "dificuldade no entendimento" e "aplicabilidade das soluções de ES". Em AC-006 os autores associam essa dificuldade ao fato das PMEs necessitarem de processos de ES ágeis e dinâmicos para um retorno rápido do investimento. Essa falta de compreensão das soluções da ES acaba que gerando outras barreiras, como o de "processos mal planejados". Por exemplo, os autores de IEEE-013 apresentam que este problema afeta negativamente a empresa em termos de tempo,

dinheiro e demais recursos. Em IEEE-043 é afirmado que os processos foram desenvolvidos para as grandes empresas e em AC-008 cita como barreira o fato das PMEs não possuírem processos bem definidos. É possível ainda identificar a dificuldade das PMEs de possuírem documentação completa e bem elaborada (IEEE-024 e SC-008), em alguns casos foi possível identificar que a empresa nem ao menos possuía documentação de seus produtos (AC-008 e AC-009). Esse fator pode amplificar a dificuldade de adaptar sistemas legados para soluções da ES (AC-009).

Outro problema observado, é quando as PMEs não adotam as soluções da ES durante o levantamento de requisitos (IEEE-018), erros nos requisitos exigem muito retrabalho, esforço e despesas adicionais (IEEE-006). Requisitos mal elicitados ou mal compreendidos causam elevado custo de detecção de erros e correção tardia (IEEE-012).

Em resumo, observa-se o Custo e a Complexidade como principais barreiras, os quais podem causar maior Esforço e, inclusive, tornar a atividade Irreal para as características da PMEs. Nenhum estudo apontou Burocracia ou Falta de Ferramentas como problema para aplicar alguma prática da ES.

4. Trabalhos Relacionados

Sánchez-Gordón e O'connor conduziram um estudo com gerentes e desenvolvedores de três PMEs para entender as lacunas entre as práticas para o processo de software e as práticas adotadas. Eles identificaram que PMEs tem seu próprio processo adaptado para seu ambiente particular sem adotar programas formais de melhoramento de processo de software. Quase toda documentação é omitida, com exceção das questões relacionadas à comunicação com o cliente. As PMEs investigadas não adotavam padrões para processo de software pois elas não se sentem prontas para implementá-los [Sánchez-Gordón and O'Connor 2016].

Berg *et al.* conduziram um mapeamento sistemático analisando 74 artigos sobre ES em Startups. As áreas de modelos, métodos, qualidade, gerenciamento de configuração e testes têm recebido pouca atenção da comunidade de pesquisa entre 1994 e 2017. Eles indicam que mais evidências são necessárias para generalizar as práticas de trabalho e o contexto de engenharia para as Startups [Berg et al. 2018]. Tegegne *et al.* também descrevem um mapeamento sistemático sobre as práticas e metodologias de desenvolvimento de software adotadas em Startups. 37 estudos primários, publicados de 2006 a 2017, indicaram que metodologias ágeis e *Lean startup* foram as mais adotadas, assim como, um total de 144 práticas de desenvolvimento de software. Eles destacam que a natureza flexível e a facilidade de empacotamento dessas metodologias e práticas justificam sua adoção nas Startups [Tegegne et al. 2019].

Laport *et al.* fornecem uma visão geral do padrão ISO/IEC 29110. Este padrão é um guia sobre a engenharia e o gerenciamento de software para entidades muito pequenas (empresas, times, departamentos de desenvolvimento de software até 25 pessoas). No artigo, uma introdução à parte central da configuração básica de engenharia de software é apresentada, assim como, casos de sucessos de adoção do padrão. Eles mencionam que é preciso acelerar a transferência do conhecimento do padrão para as PMEs [Laporte et al. 2018]. Fayad *et al.* argumentam sobre as limitações da engenharia de software para pequenas empresas, desde que a grande maioria dos estudos científicos, técnicos e práticos foram realizados com grandes times, departamentos ou empresas em

mente. Eles, portanto, concluem que engenharia de software para pequenas empresas não é apenas um caso degenerado da engenharia de software para grandes empresas, mas uma área relevante para futuras pesquisas [Fayad et al. 2000]. Kouzari *et al.* resumem os resultados de uma revisão de literatura informal sobre os fatores de sucesso e barreiras críticos em pequenas e médias empresas em melhorias de processos de software. Eles destacam que processos "leves" não são suficientes para garantir o sucesso em PMEs. Eles observaram que a maioria dos fatores críticos e barrerias estão relacionados ao custo e ao retorno do investimento [Kouzari et al. 2015].

Estes trabalhos relacionados contribuem para entender como as PMEs executam práticas, metodologias, e técnicas da ES. Porém, dado que os estudos concentram-se em um período do tempo ou em sub-áreas específicas, torna-se necessário novos estudos para ratificar ou ajustar o corpo de conhecimento de como as PMEs estão adotando as práticas da engenharia de software. Neste sentido, este mapeamento busca ampliar essa compreensão através do mapeamento sistemático.

5. Conclusão

PMEs representam uma grande parcela das empresas que desenvolvem software. Ao passo que a ES é a disciplina responsável por guiar o desenvolvimento de software, este trabalho apresentou um mapeamento sistemático para identificar como a ES atende as PMEs, levando em consideração a suas características específicas.

Com mapeamento sistemático, foi possível observar nos estudos primários que as PMEs em grande parte preocupam-se com o Processo de Software, citado como fator para aumentar produtividade e qualidade dos produtos de software. Contudo, se observa que é sempre necessário a adaptação dos processos para atender as características das PMEs. Outra prática adotada refere-se ao reúso de software, em que se observou uso de linhas de produtos de software para facilitar o desenvolvimento e reduzir o tempo de entrega. Em relação aos problemas, a falta de recursos é evidente. Sobre os recursos aqui cabe ressaltar que não se trata apenas de valores monetários, mas engloba recursos humanos, físicos, de tempo e de materiais. Outra limitação observada é a necessidade que as PMEs têm em adaptar ou criar suas próprias ferramentas e métodos. Sobre as barreiras, foi observado que PMEs possuem equipes limitadas, com pouca experiência ou maturidade e em muitos casos os profissionais desempenham mais de uma função. Ainda é possível identificar o baixo comprometimento dos próprios administradores, ou resistência deles, em relação à adoção de práticas da ES. Além das dificuldades, foi possível identificar uma variedade de benefícios que as PMEs podem obter com as atividades da ES. Um processo sistemático de desenvolvimento gera produtos mais rapidamente, com menos erros, exigindo menos manutenção. O produto final tem mais qualidade e atende as expectativas dos clientes. Se a adoção da ES eleva os custos iniciais, as PMEs recuperam este investimento de forma mais rápida daquelas que não a adotaram.

Um trabalho futuro é conduzir entrevistas com profissionais de PMEs para observar se as atividades, os benefícios e os desafios mapeados são observados na prática. Assim, os benefícios e desafios poderão ser atrelados às atividades identificadas. Além disso, pretende-se identificar outros benefícios e desafios enfrentados durante o desenvolvimento ou a manutenção de seus sistemas. Por exemplo, atividades fundamentais para a qualidade do software, como testes, refatoramentos e inspeção de código.

Referências

ABES (2020). Mercado Brasileiro de Software: panorama e tendências. http://central.abessoftware.com.br/Content/UploadedFiles/Arquivos/Dados Acessado: 26-Set-2020.

Berg, V., Birkeland, J., Nguyen-Duc, A., Pappas, I. O., and Jaccheri, L. (2018). Software startup engineering: A systematic mapping study. *Journal of Systems and Software*, 144:255–274.

de Barros Sampaio, S. C., Marinho, M. L. M., de O. Luna, A. J. H., and Moura, H. P. (2015). Reflecting, adapting and learning in small software organizations: an action research approach. In *27th International Conference onSoftware Engineering and Knowledge Engineering (SEKE)*, pages 632–637.

Degwitz, L. T. N. (2014). Software engineering in small projects: The most essential processes. Master's thesis, University of Miami.

Fayad, M. E., Laitinen, M., and Ward, R. P. (2000). Thinking objectively: Software engineering in the small. *Communications of the ACM*, 43(3):115–118.

Kouzari, E., Gerogiannis, V. C., Stamelos, I., and Kakarontzas, G. (2015). Critical success factors and barriers for lightweight software process improvement in agile development: A literature review. In *10th International Joint Conference on Software Technologies (ICSOFT)*, volume 1, pages 1–9. IEEE.

Laporte, C. Y., Munoz, M., Mejia Miranda, J., and O'Connor, R. V. (2018). Applying software engineering standards in very small entities: From startups to grownups. *IEEE Software*, 35(1):99–103.

Majchrowski, A., Ponsard, C., Saadaoui, S., Flamand, J., and Deprez, J.-C. (2016). Software development practices in small entities: An iso29110-based survey. *Journal of Software: Evolution and Process*, 28.

Moll, R. (2013). Being prepared -a bird's eye view of smes and risk management. *ISO Focus+. The Magazine of the Intern. Organization for Starndardization*, 4(2):16–18.

O'Connor, R. and Coleman, G. (2009). Ignoring "best practice": Why irish software smes are rejecting cmmi and iso 9000. *Australasian J. of Inf. Systems*, 16.

Petersen, K., Vakkalanka, S., and Kuzniarz, L. (2015). Guidelines for conducting systematic mapping studies in software engineering: An update. *Information and Software Technology*, 64:1–18.

Sánchez-Gordón, M.-L. and O'Connor, R. V. (2016). Understanding the gap between software process practices and actual practice in very small companies. *Software Quality Journal*, 24(3):549–570.

SOFTEX (2019). Associação para promoção da excelência do software brasileiro. relatório das atividades 2019. https://softex.br/booksoftex/. Acessado: 26-Set-2020.

Tegegne, E. W., Seppänen, P., and Ahmad, M. O. (2019). Software development methodologies and practices in start-ups. *IET Software*, 13:497–509(12).

Utilização do *Scrum* no desenvolvimento de uma aplicação *web*: um Estudo de Caso

Thales F. Dal Molim[1], **Francisco Carlos M. Souza**[1]

[1]Universidade Tecnológica Federal do Paraná (UTFPR)
Dois Vizinhos – PR – Brasil

Abstract. *Scrum stands out for its ability to deal with the unpredictability inherent in complex projects, therefore, one of the concepts used is that of self-managed and multifunctional teams. However, when ensuring greater autonomy to a team, greater commitment is required, especially in uncertain environments, such as the academic environment. Thus, this article aims to demonstrate the application of Scrum in the development of a web application in an academic environment, through a case study. After the execution, it was concluded that, although the essential requirements were delivered, initially there were moments of high productivity, however, during the execution there were delays and unproductiveness, mainly due to the lack of regular meetings, which led to a demotivation of the team.*

Resumo. *O Scrum destaca-se pela capacidade de lidar com a imprevisibilidade inerente a projetos complexos, para tanto, um dos conceitos utilizados é o de equipes autogeridas e multifuncionais. Entretanto, ao garantir maior autonomia a uma equipe, requer-se um maior comprometimento da mesma, especialmente em ambientes incertos, como o meio acadêmico. Desta forma, o presente artigo tem por objetivo demonstrar a aplicação do Scrum no desenvolvimento de uma aplicação web em ambiente acadêmico, por meio de um estudo de caso. Após a execução, concluiu-se que, embora os requisitos essenciais tenham sido entregues, inicialmente houve momentos de alta produtividade, porém, ao longo da execução houve atrasos e improdutividade, devido principalmente à falta de reuniões regulares, que levaram a uma desmotivação da equipe.*

1. Introdução

Desenvolver softwares é uma tarefa complexa. De um modo geral, esta complexidade está relacionada à própria natureza dos sistemas de software, isto é, não são regidos pelas leis da física nem por processos fabris, não havendo limites naturais para seu potencial. Dada sua inerente complexidade, em pouco tempo tornam-se difíceis de entender e de modificar [Schwaber 2004, Sommerville 2011].

De acordo com [Sommerville 2011], a Engenharia de Software surge para auxiliar no desenvolvimento profissional de software de qualidade, oferecendo, para tanto, diferentes processos que devem ser adequados ao contexto específico de um determinado projeto. Sob o mesmo ponto de vista, para [Pressman and Maxim 2015], o processo não deve ser rígido, mas adaptável ao problema, ao projeto, à equipe e à cultura organizacional aos quais será inserido.

Neste contexto, as metodologias ágeis destacam-se por serem capazes de lidar com projetos complexos, uma vez que são capazes de controlar a imprevisibilidade íntrinseca

a estes projetos. Para tanto, devem se adaptar ao projeto em que estão inseridas, garantindo entregas frequentes por meio de uma série de incrementos, onde cada incremento corresponde a uma nova funcionalidade do sistema.

Sendo assim, o presente trabalho pretende demonstrar a utilização de uma metodologia ágil por meio de um estudo de caso em ambiente acadêmico, visando descrever as atividades e relatar as vantagens ou mesmo desvantagens encontradas no processo. Para tanto, utilizou-se o *framework Scrum* para desenvolver uma aplicação *web* com o objetivo de facilitar trocas e empréstimos de livros entre usuários dentro de uma rede de consumo colaborativo.

O restante do artigo está estruturado da seguinte maneira: a seção 2 descreve o *framework Scrum* e seu funcionamento; a seção 3 apresenta o estudo de caso e as atividades nele desenvolvidas; e, por fim, a seção 4 apresenta as considerações finais.

2. Scrum

De acordo com [Rubin 2012], a origem do *Scrum* pode ser traçada desde um artigo de 1986 [Takeuchi and Nonaka 1986] que evidenciava a importância de equipes multidisciplinares e autogeridas, utilizando o termo *"scrum"* em referência às equipes do *rugby*, propondo que as equipes permaneçam intactas do começo ao fim e sejam responsáveis por combinar as diversas etapas de desenvolvimento, criando, portanto, uma certa continuidade.

Baseando-se nesta premissa, Jeff Sutherland e sua equipe criam, nos anos 1990, o *framework Scrum* para desenvolvimento de software. Em 1995, Ken Schwaber publica o primeiro artigo apresentando o processo então criado [Schwaber 1995]. Nos próximos anos, Sutherland e Schwaber realizariam diversas publicações referentes ao *Scrum* [Rubin 2012].

O *Scrum*, na definição de [Schwaber and Sutherland 2017], é um *framework* estrutural capaz de lidar com problemas complexos, visando entregar maior valor em menor tempo. Para tanto, o *Scrum* define papéis, eventos e artefatos que devem ser associados ao time e são essenciais ao processo.

2.1. Papéis, eventos e artefatos

O desenvolvimento inicia-se com a definição do *Product Backlog*, artefato composto por uma lista de requisitos funcionais e não funcionais que devem materializar os anseios dos *stakeholders*. O responsável por gerenciar este artefato é o *Product Owner (PO)*, que deve, também, garantir que o Time de Desenvolvimento compreenda com clareza os itens listados, visando otimizar o valor do trabalho desenvolvido.

O Time de Desenvolvimento é multifuncional e autogerido, ou seja, a equipe possui todas as competências necessárias para realizar o trabalho, bem como a liberdade para decidir a melhor forma de realizá-lo, sem interferências externas. Um papel chave no andamento do projeto, é o *Scrum Master (SM)*. É o responsável pela manutenção correta do processo, implementando e ensinando as práticas do *Scrum* a todos os envolvidos no projeto. O *SM* deve garantir que as reuniões ocorram corretamente e que a equipe entenda seu propósito.

A natureza do *Scrum* é iterativa, sendo assim, o trabalho flui por meio de uma sequência de iterações, chamadas *Sprints*. Cada *Sprint* se inicia com uma reunião de Planejamento da *Sprint*, onde planeja-se quais funcionalidades do *Product Backlog* serão implementadas, as quais constituirão um novo artefato: o *Sprint Backlog*. São então realizadas reuniões diárias (*Daily Scrum*) para verificar o andamento da *Sprint*. Ao final da *Sprint* é realizada a *Sprint Review*, uma reunião mais longa que visa conferir o que foi realizado até então, os problemas ocorridos, bem como discutir o que poderá ser feito a seguir [Schwaber and Sutherland 2017].

Finalmente, outro artefato utilizado é o *Burndown Chart*, uma representação gráfica que visa representar a quantidade de trabalho restante a ser desenvolvido em uma *Sprint*, no eixo vertical, e o tempo restante, no eixo horizontal. Por meio deste artefato, é possível mensurar a produtividade da equipe, alertar possíveis atrasos, e calcular quando o trabalho será finalizado [Schwaber 2004].

3. Estudo de Caso

A pesquisa coordenada e organizada por [Failla 2016] na 4ª edição de Retratos da Leitura no Brasil, aponta dados importantes sobre hábitos de leitura no país. Da amostra utilizada na pesquisa, 56% dos entrevistados considera-se leitor(a)[1], crescimento de 6% em comparação à mesma pesquisa realizada em 2011, denotando um aumento no número de brasileiros que se identificam como leitores.

Dentre os que se consideram leitores, a pesquisa [Failla 2016] elencou também as razões para não se ter lido mais, das quais evidenciam-se os seguintes números: 8% não leu mais devido à falta de bibliotecas perto de onde mora; 7% porque acha o preço dos livros caro; 5% porque não tem dinheiro para comprar; e 3% por não ter um local próximo de onde mora para comprar livros, como mostra a Figura 1.

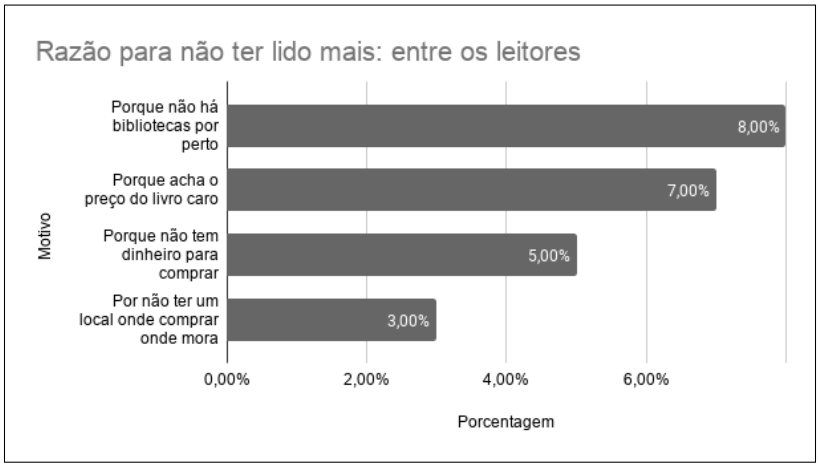


Figura 1. Razão para não ter lido mais: entre os leitores [Failla 2016]

Portanto, temos que 23% dos leitores identificados leram menos devido a motivos relacionados à condição financeira ou pela falta de proximidade de bibliotecas ou

[1]De acordo com o critério utilizado por [Failla 2016], define-se leitor como sendo aquele que leu, inteiro ou em partes, pelo menos 1 livro nos 3 meses anteriores à pesquisa.

lojas físicas. Outro dado relevante constatado pela pesquisa [Failla 2016] é o fato de que 81% dos leitores são usuários da *internet*. Sendo assim, entendemos que a *internet* pode constituir um meio efetivo de atingir o leitor e influenciar no hábito da leitura.

Desta forma, levando em consideração os dados levantados em relação às razões que levaram os leitores a lerem menos, e em relação ao consumo de *internet* pelos mesmos, foi proposto o desenvolvimento de uma ferramenta *online* que visa facilitar a aquisição de livros, via trocas e empréstimos entre os próprios usuários, a ser desenvolvida utilizando o *framework Scrum*.

3.1. Proposta

A ferramenta proposta foi nomeada *Escambook* (aglutinação das palavras "escambo" e *"book"*) e baseia-se no princípio do consumo colaborativo, definido por [Botsman and Rogers 2011] como sendo um modelo de consumo baseado na colaboração entre membros de uma comunidade, podendo ser algo local ou através da *internet*, que busca formar grupos de pessoas interessadas em compartilhar bens, por meio de trocas, empréstimos ou mesmo doações.

Considerando os dados levantados anteriormente em relação ao uso da *internet* por leitores no Brasil, em conjunto com dados levantados por [Wroblewski 2011] referentes ao crescente uso de *smartphones* pela população global, a ferramenta foi proposta dentro do paradigma *web*, utilizando o conceito *mobile first*[2].

3.2. Planejamento

Por tratar-se de um projeto desenvolvido como parte de um trabalho a ser entregue visando a conclusão de uma disciplina acadêmica, o *Scrum* precisou ser adaptado para tanto. O desenvolvimento deveria durar, no máximo, o período de um semestre letivo, correspondendo ao tempo da disciplina ministrada. Portanto, as *Sprints* foram previamente planejadas da seguinte maneira: três *Sprints* iniciais com a duração de uma semana cada, e duas *Sprints* finais com a duração de duas semanas cada.

A equipe, composta por seis alunos, todos com conhecimento prévio acerca do *framework* utilizado, foi dividida devidamente em: um *PO* e cinco desenvolvedores, sendo um destes o *SM*. Foram criadas contas, por cada membro da equipe, nas plataformas: *Slack*, para auxiliar na comunicação; *Trello*, para simular um quadro de tarefas e visualizar o fluxo de trabalho; e *GitHub*, para controlar o versionamento do produto. Além destas, outras ferramentas auxiliares foram utilizadas pela equipe, como exposto na Tabela 1.

[2]Abordagem na qual o desenvolvimento de aplicações tem foco direcionado a dispositivos móveis [Wroblewski 2011].

Tabela 1. Principais ferramentas utilizadas pela equipe

Ferramenta	Objetivo
draw.io	Elaborar diagramas, dentre outras representações de alto nível.
GitHub	Controlar o versionamento do produto.
Google Drive	Armazenar, criar e editar documentos e artefatos.
Insomnia	Testar as requisições feitas à API.
pgAdmin 4	Gerenciar o banco de dados do aplicativo.
Slack	Auxiliar na comunicação da equipe.
Trello	Simular um quadro de tarefas para visualizar o fluxo de trabalho.
Visual Studio Code	Produzir o código-fonte do aplicativo.

Devido ao ambiente em que o projeto estava inserido, as reuniões do *Scrum* precisaram ser adaptadas, de modo que: o Planejamento das *Sprints* deveria ocorrer na primeira aula da semana correspondente à *Sprint*, juntamente com a *Sprint Review* da *Sprint* anterior. Já as reuniões diárias, ocorreram informalmente e somente quando possível, devido a restrições relacionadas ao horário disponível de cada membro da equipe

Após definido o escopo do projeto, o *PO* em conjunto com a equipe levantou os requisitos funcionais e não funcionais que comporiam o *Product Backlog*. Foi criada então uma pasta compartilhada em um serviço de armazenamento (*Google Drive*), onde cada membro teria acesso a este e aos demais artefatos.

Para a implementação, optou-se pela composição de duas aplicações distintas: uma aplicação *frontend*, responsável pela interação do usuário com o produto, e uma API *backend*, responsável por gerenciar a interação entre o *frontend* e a base de dados. Foi escolhido o padrão MVC (*model-view-controller*[3]) para estruturar o aplicativo como um todo, como demonstrado na Figura 2.

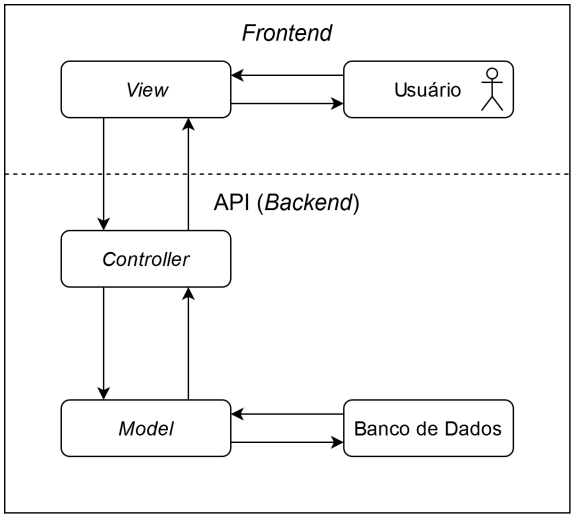


Figura 2. Arquitetura do aplicativo proposto

[3]Padrão de arquitetura de software na qual a interface de usuário é desacoplada das funcionalidades da aplicação e dos dados armazenados [Pressman and Maxim 2015].

Para o desenvolvimento da aplicação *frontend*, decidiu-se pela utilização do *Vue.js*, *framework JavaScript* de código-aberto, englobando, portanto, além da linguagem *JavaScript*, a linguagem de marcação HTML e a linguagem de folhas de estilo CSS. Para a implementação da API *backend*, utilizou-se *Node.js*, interpretador de *JavaScript* de código-aberto, associado ao SGBD *PostgreSQL*, também de código-aberto.

3.3. Primeira *Sprint*

A primeira *Sprint* foi utilizada para o planejamento geral do projeto, nela foram levantados os requisitos iniciais e determinados os aspectos técnicos, como as tecnologias e padrões a serem utilizados. Foram iniciados, também, os trabalhos de documentação, elaboração de diagramas, bem como foram levantados e reunidos materiais de estudo objetivando nivelar a equipe quanto ao conhecimento técnico.

Decidiu-se, também, por dividir o Time de Desenvolvimento em dois: uma equipe para desenvolver a aplicação *frontend* e outra para desenvolver a API *backend*, pois alguns integrantes não eram familiarizados com algumas das tecnologias empregadas, assim cada equipe poderia ter um foco de estudo mais direcionado. Para tanto, a comunicação entre as duas equipes deveria ser constante. Não houveram grandes dificuldades, todos empenharam-se e entregaram o que havia sido estipulado no Planejamento da *Sprint*.

3.4. Segunda *Sprint*

Para a segunda *Sprint*, foram alocadas seis funcionalidades a partir do *Product Backlog*, nesta iteração surgiram as primeiras dificuldades. Embora a documentação e os papéis estivessem bem definidos, o desenvolvimento falhou em grande parte devido à inexperiência de alguns integrantes acerca das tecnologias empregadas, como fora apontado pelos mesmos na *Sprint Review*. Das seis funcionalidades que deveriam ser entregues, a equipe do *backend* concluíra quatro, ficando o *frontend* inteiramente por fazer (Tabela 2).

Tabela 2. Funcionalidades a serem entregues na segunda *Sprint*

ID	Nome	*Frontend*	*Backend*
RF001	Cadastrar usuário	Pendente	Entregue
RF005	Fazer login	Pendente	Entregue
RF006	Fazer logout	Pendente	Entregue
RF011	Cadastrar livro	Pendente	Entregue
RF015	Visualizar mural de livros	Pendente	Pendente
RF016	Acessar biblioteca pessoal	Pendente	Pendente

3.5. Terceira *Sprint*

No planejamento da terceira *Sprint*, definiram-se três novas funcionalidades a serem desenvolvidas, além das que permaneceram incompletas na *Sprint* anterior. Desta vez, a comunicação e o desenvolvimento em equipe fluiram melhor, havendo maior envolvimento do time nas tarefas. O *backend* concluiu o que lhe fora designado, enquanto o *frontend* deixou apenas uma funcionalidade por fazer, como visto na Tabela 3.

Sendo assim, embora o desempenho da *Sprint* tenha sido no geral positivo, os integrantes responsáveis pelo *frontend* destacaram, na *Sprint Review*, que não haviam se

dedicado o quanto gostariam, pois, embora as funcionalidades tivessem sido entregues, não estavam satisfeitos com a qualidade de seu trabalho.

Tabela 3. Funcionalidades a serem entregues na terceira *Sprint*

ID	Nome	*Frontend*	*Backend*
RF001	Cadastrar usuário	Entregue	-
RF005	Fazer login	Entregue	-
RF006	Fazer logout	Entregue	-
RF011	Cadastrar livro	Entregue	-
RF015	Visualizar mural de livros	Entregue	Entregue
RF016	Acessar biblioteca pessoal	Entregue	Entregue
RF017	Adicionar livro na biblioteca pessoal	Entregue	Entregue
RF018	Remover livro da biblioteca pessoal	Entregue	Entregue
RF019	Marcar ou desmarcar livros disponíveis	Pendente	Entregue

3.6. Quarta *Sprint*

Na quarta *Sprint* foram alocadas as últimas quatro funcionalidades previstas, além da que havia ficado para trás na última *Sprint*. Esta seria a primeira *Sprint* dentre as duas de maior duração (duas semanas). O número reduzido de funcionalidades para um maior período de tempo se justifica pelo fato de estas serem funções mais complexas do aplicativo, como o registro de trocas e empréstimos de livros entre usuários.

Porém, esgotadas as duas semanas, o *backend* havia implementado todas as funcionalidades, enquanto o *frontend* havia entregue apenas uma (Tabela 4). Na *Sprint Review*, concluiu-se que, além de tudo, o *frontend* do aplicativo estava deixando a desejar em qualidade, e propôs-se que, estando o *backend* concluído, na próxima *Sprint*, que seria utilizada para melhorias e correções gerais, as duas equipes se reuniriam para uma refatoração total do *frontend*, caso contrário o produto final não atingiria as expectativas.

Ainda durante a *Sprint Review*, os próprios integrantes da equipe responsável pelo *frontend* admitiram que faltou comprometimento e dedicação de sua parte, apesar dos constantes estímulos e cobranças do *SM*. Dessa forma, percebeu-se também que, separar o Time de Desenvolvimento em duas equipes pode não ter sido uma boa experiência, pois enquanto uma das equipes mostrou total dedicação ao projeto, a outra deixou a desejar.

Tabela 4. Funcionalidades a serem entregues na quarta *Sprint*

ID	Nome	*Frontend*	*Backend*
RF014	Buscar livros	Entregue	Entregue
RF019	Marcar ou desmarcar livros disponíveis	Pendente	-
RF024	Registrar troca	Pendente	Entregue
RF027	Registrar empréstimo	Pendente	Entregue

3.7. Quinta *Sprint*

A quinta e última *Sprint*, foi, portanto, voltada para uma refatoração do *frontend*. Constatou-se que a parte mais complexa acabou sendo implementada pelo *SM*, que

possuía maior conhecimento técnico, delegando as tarefas menos complexas ao restante do time, bem como, reaproveitando o possível do que já havia sido produzido, uma vez que faltava pouco tempo para a entrega final do produto.

Ao final das duas semanas, tanto o *backend* quanto o *frontend* estavam finalizados. Porém, faltou tempo para possíveis melhorias que agregariam maior valor ao produto. Nesta última *Sprint Review*, o time concordou que a comunicação entre as equipes poderia ter sido melhor, e que a falta de reuniões diárias (*Daily Scrum*), que ocorriam apenas quando possível, prejudicou o projeto, pois o desenvolvimento por vezes não manteve constância.

3.8. *Product Release*

A entrega final do produto consistiu em uma apresentação do processo de desenvolvimento do mesmo, seguida de uma demonstração de seu uso, realizadas em sala de aula pelo *PO* e pelo *SM* aos demais integrantes da disciplina.

O aplicativo desenvolvido permite ao usuário, após cadastrar-se no sistema, acessar sua página pessoal, onde encontra-se um dos elementos centrais da plataforma, a biblioteca pessoal (Figura 3), onde poderá visualizar e controlar todos os livros que possui, assim como livros que recebeu através de empréstimos ou trocas com outros usuários.

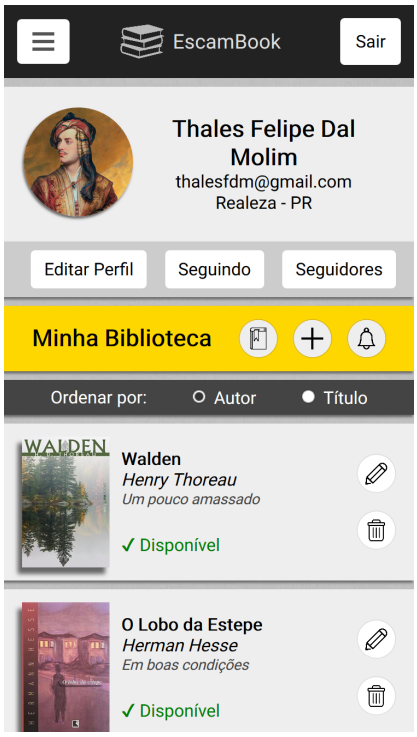


Figura 3. Interface da biblioteca pessoal

A ferramenta consta também com um "mural de livros" (Figura 4), onde é possível buscar livros por título, autor ou ISBN. Através deste, usuários cadastrados podem adicionar à sua biblioteca pessoal cópias que já possuam ou procurar cópias de outros usuários para trocar ou emprestar.

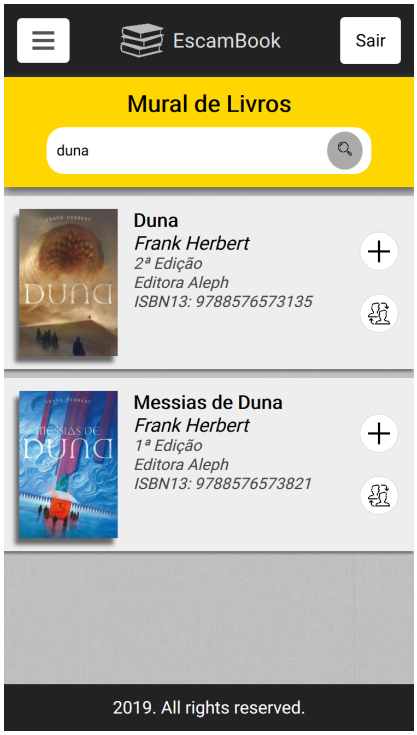


Figura 4. Interface do mural de livros

O produto final atingiu os objetivos propostos, entregando os requisitos essenciais inicialmente propostos no *Product Backlog*. Entretanto, ainda há espaço para aperfeiçoamentos e novas funcionalidades que agregariam maior valor ao produto, como, por exemplo: um *chat*; um sistema de seguidores; um sistema de avaliações; e a busca por usuários de acordo com determinada região geográfica.

4. Conclusão

Buscando demonstrar o uso do *Scrum* no desenvolvimento de uma aplicação *web*, propôs-se a criação de uma ferramenta *online* com o objetivo de facilitar empréstimos e trocas de livros entre usuários, a ser desenvolvida em ambiente acadêmico no período de um semestre letivo.

Inicialmente, foi descrita a origem, a definição e as características do *Scrum*, assim como os papéis, eventos e artefatos que o compõem. Em seguida, foi apresentado o estudo de caso, onde justificou-se a proposta, para que, então, fosse relatado o processo de planejamento e desenvolvimento do aplicativo por meio do *framework Scrum*.

Verificou-se, durante o desenvolvimento, a importância de seguir corretamente as instruções do *Scrum*, uma vez que, o uso dos artefatos e a definição dos papéis de acordo com o perfil de cada integrante manteve a equipe melhor organizada, visto que o *PO* concentrou-se em levantar e gerenciar as funcionalidades do *Product Backlog*, enquanto o *SM* focou-se em manter o funcionamento do processo até o final.

Evidenciou-se, também, que a falta ou irregularidade das reuniões diárias prejudicou o projeto como um todo, visto que houveram picos de produtividade em certos momentos quando a equipe se reunia, em contraste aos momentos de hiato entre os encontros, quando houve procrastinação e improdutividade.

Concluiu-se que, apesar das dificuldades encontradas, o produto final foi entregue dentro do prazo e os requisitos essenciais implementados, e que, portanto, a aplicação do *Scrum* foi bem sucedida. Finalmente, em relação ao produto que foi desenvolvido, considerou-se que o mesmo pode ainda ser estendido com novas funcionalidades que agregariam maior valor ao mesmo.

Referências

Botsman, R. and Rogers, R. (2011). *What's mine is yours: the rise of collaborative consumption*. HarperCollins.

Failla, Z. (2016). *Retratos da leitura no Brasil 4*. Sextante.

Pressman, R. and Maxim, B. (2015). *Software Engineering: a practitioner's approach*. McGraw-Hill Higher Education.

Rubin, K. (2012). *Essential Scrum: a practical guide to the most popular agile process*. Addison-Wesley.

Schwaber, K. (1995). Scrum development process. *Advanced Development Methods*.

Schwaber, K. (2004). *Agile project management with Scrum*. Microsoft Press.

Schwaber, K. and Sutherland, J. (2017). *Guia do Scrum*. (n.p.).

Sommerville, I. (2011). *Engenharia de Software*. Pearson Brasil.

Takeuchi, H. and Nonaka, I. (1986). The new new product development game. *Harvard Business Review*.

Wroblewski, L. (2011). *Mobile First*. A Book Apart.

The Use of Design Thinking in a Global Information Technology Company

Gabriel Kryvoruchca[1], Lauriane Correa[1], Rafael Parizi[1], Sabrina Marczak[1]

[1]MunDDoS Research Group – School of Technology
Pontifícia Universidade Católica do Rio Grande do Sul - PUCRS
Porto Alegre, RS, Brazil

gabriel.kryvoruchca@acad.pucrs.br
{lauriane.pereira,rafael.parizi}@edu.pucrs.br
sabrina.marczak@pucrs.br

Abstract. *Software companies have been using Design Thinking to support software development, fostering the creation of innovative features and products. However, there is not so much knowledge of what matters for the application of Design Thinking being successful. Thus, this study aims to describe how does the adoption of Design Thinking with software development take place in a global information technology company through an interview-based case study with 16 professionals. Our analysis indicates that Design Thinking is used in software products, software improvements, processes, user experience identification and solution discovery. Also, there are perceived benefits and challenges during the Design Thinking activities. As a result, this paper describes the Design Thinking phenomenon in software development, serving as a guide for practitioners on how to set up and implement Design Thinking activities and publishing more academic research.*

1. Introduction

Information technology companies want to deliver software products in less time with higher-quality [Subih et al. 2019]. Putting the user at the center of the software development process has been pointed out for years as the key to success [Luedeke et al. 2018]. It has been argued that Design Thinking (DT) is relevant in this attempt to better establish user-centric activities to support the understanding of user needs and to develop a fit solution [Hehn et al. 2020].

The use of DT to aid software development activities fosters the development of human-centered solutions more effectively and, therefore, it is important to know in-depth about how organizations have used DT for their activities [Hehn and Uebernickel 2018].

This way, we focused on describing this phenomenon, answering the following research questions (RQ) *"RQ1. How does the adoption of Design Thinking with software development take place?"* and *"RQ2. What are the perceived benefits and challenges of Design Thinking adoption?"*. It aims to identify how the adoption of Design Thinking with software development takes place in a global information technology company (called ORG - confidential name), located in Brazil.

Thus, this study conducted an interview-based case study with 10 coaches and 6 participants, totaling 16 professionals to understand how DT is used, what techniques and

tools are used, how knowledge is shared to the team, which professionals (or team roles) get involved in Design Thinking activities and the perceived benefits and challenges of Design Thinking usage.

This paper reports a case study in a global information technology company. Our main contributions are: (i) a set of characteristics and techniques used in this context; (ii) perceived benefits by the teams; (iii) perceived challenges by the teams so far; and compared studies highlighting similarities and differences.

The paper is organized as follows: Section 2 presents the Design Thinking background. Section 3 provides the research method used. Section 4 describes the preliminary results. Section 5 discusses our findings. Section 6 covers related work. Section 7 concludes with a summary of this article and future studies.

2. Design Thinking in Software Development

Design thinking has gained recognition as an approach to problem-solving that relies on interdisciplinary teamwork, exploration of human needs, rapid prototyping and interactive learning cycles in the earlier stages of product, service and system development processes [Brown 2008]. Design thinking proposes to assist software development by supporting the understanding of the users needs [Vetterli et al. 2013].

Focusing on user needs, the purpose of DT is to collaboratively and innovatively identify opportunities to solve problems. It can be defined in three perspectives: as a process, as a mindset or as a toolbox [Brenner et al. 2016]. As a mindset, DT is focused on a strong orientation to discover the obvious and hidden customers' and users' needs and prototype the possible solutions. As a process, DT is seen as a combination of a micro-process as an innovation process and a macro-process as the milestones manifested in prototypes that must fulfill defined requirements. Finally, as a toolbox, DT refers to the application of numerous methods and techniques from design, software engineering and psychology [Brenner et al. 2016], for instance, personas, brainstorming, others.

DT and requirements engineering are distinct when it comes to the underlying philosophies, but many artifacts are complementary or even overlapping, the Design Thinking follows a philosophy of domain understanding and the learning curve leading to it – regardless of the surrounding processes [Hehn et al. 2020]. When looking through the lens of process, DT can be defined as a set of distinct working spaces, which can be adapted and executed non-sequentially according to the context [Thoring et al. 2011].

There is a wide variety of Design Thinking models, emerging and being updated to support the distinct contexts and problems that Design Thinking can support [Hasso-Plattner Institute 2020]. There are models such as double diamond design model [Council 2020] and d-school model of the Hasso-Plattner-Institute [Hasso-Plattner Institute 2020]. Although exists the guides and manuals on Design Thinking, there are professionals who do not know how to conduct and decide techniques [Mateusz Dolata 2017].

3. Research Method

This study conducted an interview-based case study. We did design our study considering its qualitative nature [Dybå et al. 2011] based on the procedure by [Runeson and Höst 2009], which is: to establish the study goal and scope, to define the case and criteria to

Table 1. Professionals' Profiles

ID	Role	Description	Yrs at ORG	Yrs in DT
C1	Developer	Development team member	10	5
C2	Developer	Development team member	9	2
C3	Developer	Development team member	8	3
C4	Developer	Development team member	6	1
C5	Product Support Manager	Manages the entire product support organization	10	7
C6	Partner Service Delivery	Responsible to provide support, training and assistance to certified partners	10	6
C7	Customer Relations	Responsible for representing the customer within ORG. Will coordinate and speed up trainings, assistance, etc	9	5
C8	Customer Center	Responsible for offering ORG applications and enabling co-innovation with partners and customers	7	4
C9	Product Support	Support engineer working with ORG applications	5	2
C10	Product Support	Support team member	5	4
P1	Product Support	Support team member	10	4
P2	Product Support	Support team member	7	2
P3	Product Support	Support team member	7	3
P4	Product Support	Support team member	5	2
P5	Product Support	Support team member	4	3
P6	Product Support	Support team member	3	3

select it, to select the data collection method, to decide on the data analysis method, and to get ethics approval to conduct the study within our institution and from the company management. Our research within this study was guided by the following research questions (RQ) *"RQ1. How does the adoption of Design Thinking with software development take place?"* and *"RQ2. What are the perceived benefits and challenges of Design Thinking adoption?"*.

3.1. Case Study Setup

We set up a descriptive case study in a global information technology company, called ORG. ORG develops and sells software systems solutions. This European-based company counts with over 10,000 information technology professionals around the world. Also, ORG follows D-school model in DT.

First, we observed a DT session for 2 hours to understand how it was applied. After that, we invited 20 employees from different teams, which 16 accepted to participate in this study. These professionals have different roles, such as Developer, Product Support, Manager and others, they are identified in Table 1.

We conducted an interview-based case study with 10 coaches and 6 participants, totaling 16 information technology professionals located in Brazil site. ORG classifies the professionals in two profiles: Design Thinking coach who receives a formal training and conducts sessions and, Design Thinking participant, who attends a session. A Design Thinking session can be organized in a few hours or a set of days, depending on demand.

Table 2. Interview Script

RQ	ID	Question
RQ1	Q1	When, how and for what purpose is used Design Thinking?
	Q2	Who are the stakeholders involved during the Design Thinking?
	Q3	Which resources and tools do you use to support the Design Thinking?
	Q4	How the gathered knowledge during Design Thinking is shared to the development team?
RQ2	Q5	What are the perceived benefits of Design Thinking usage?
	Q6	What are the perceived challenges of Design Thinking usage?
	Q7	What the perceived quality aspects are identified when Design Thinking is used?

3.2. Data Collection and Analysis Methods

The interviews were semi-structured, organized in 7 open-ended questions, see Table 2. To define this interview script we followed the guidelines suggested by [Kitchenham and Pfleeger 2002]. We performed the activities: (1) interview script creation, (2) script evaluation with a senior researcher who has industry experience in software development (12 years) and Design Thinking (3 years), (3) professionals selection, (4) interviews conduction, (5) interview transcription, and (6) data analysis and consolidation.

All professionals accepted the consent form. Each interview took on average of one-hour long. Notes were taken by the interviewer and used to aid data analysis. The study used content analysis technique to gain insights into the different variations of the Design Thinking phenomenon in a specific scenario through the subjects' feelings and thoughts. This way, all interviews were transcribed and analyzed using the this technique based on Krippendorff [Krippendorff 2018].

It aims to reveal perspective and patterns of behaviors among the professionals. All answers were analyzed to classify the results. The results were organized following the interview script. The authors discussed the results classification and categorization, achieving a consensus of all perspectives, as reported in Section 4.

4. Results

In this section, we present the questions from our interview script and its consolidation based on professionals' perspectives. They are identified in Table 1, where 'C' represents a DT coach and 'P' a DT participant. Table 3 and Table 4 show an extract of the results for the interviews questions, grouped for RQ1 and RQ2, respectively.

The Design Thinking model used at ORG is D-school [Hasso-Plattner Institute 2020]: **understand** working space aims to discover the user needs, the professionals highlighted the following techniques: (i) *briefing* to start the conversation among all stakeholders; (ii) *statement/challenge* to explore the needs; (iii) *customer journey map* to understand the process as a whole; (iv) *charting* to define possible solutions, and; (v) *creative reframe* to rewrite the challenge, focused on discovering new aspects.

In the **observe** working space, they mentioned the ways: (i) *interviews* with users; (ii) *shadowing* to observe the users in their activities; (iii) *desk research* to explore about the topic, and; (iv) *competitors' research* to explore how the competitor are doing it. In **point of view** working space, the professionals mentioned the following techniques: (i) *storytelling* to describe users' perspectives; (ii) *personas* to identify the people who will

Table 3. Interview's questions and professionals' answers for RQ1.

RQ1. How does the adoption of Design Thinking with software development take place?	
Question	Result (Professionals – ['C': Coach - 'P': Participant])
Q1. When, how and for what purpose is used Design Thinking?	(i) discover new requirements (C1, C2, C5, C6, P2) (ii) promote technological development (C8, C9, P1, P4, P5) (iii) improve projects and processes (P3, P4, P5, P6, C8), and (iv) build a mindset (C4, C5, C8, P2)
Q2. Who were the stakeholders involved during the Design Thinking?	(i) interdisciplinary team (e.g. C9, C10, P1, P2, P3) (ii) consultants (C1) (iii) business specialists (C3, C6) (iv) product owner (C5, C10) (v) designers (C2, C3, C7) (vi) managers (C1, C7, C9, P6) (vii) end-users (C4, C5, P2, P3) (viii) developers (C1, C5, C8, C9, C10), and (ix) Design Thinking coaches(P5, P6)
Q3. Which resources and tools did you use to support the Design Thinking?	(i) google drive (C5) (ii) github (P4) (iii) boards (C3, C8, P1, P3, P4) (iv) prototyping tool (C2, C10), and (v) flip-charts and pens (C1, C5, C10 e P1)
Q4. How the gathered knowledge during Design Thinking was shared to the development team?	(i) storyboard (C6) (ii) storytelling (C5, C6, C8) (iii) pictures (e.g. C9, C10, P4, P5, P6) (iv) sheets (C2, C8, P6) (iv) documents (C2, C8, C9), and (v) emails (C7, C9, C10)

interact with the solution, and; (iii) *point of view*[1] to represent the users' visions.

During the **ideate** working space, the professionals described the following techniques: (i) *brainstorming* to generate the participants' insights (ii) *2x2 matrix* to organize insights; (iii) *reverse brainstorming* to find a viable solution. In **prototype** working space, the professionals mentioned the following techniques: (i) *storyboard* to align the ideas in a graphical visualization way; (ii) *low-fidelity prototypes* to evaluate the solutions; (iii) *canvas* to visualize the business aspects (iv) *flowchart* to draw the flows; (iv) *mock-ups* to give an idea of how it would be and then start a prototype, and; (v) *physical prototypes*.

In **test** working space, the professionals reported the (i) *test matrix* to map the positive and negative points, and questions about the solution; (ii) *user validation* using paper prototype; (iii) *usability tests*, and; (iv) *user feedback* to understand the users' perceptions.

5. Discussion

This study identifies how the adoption of Design Thinking with software development takes place in a global information technology company. Our discussion is organized to answer our two research questions.

[1] Apache MADlib: https://madlib.apache.org/

Table 4. Interview's questions and professionals' answers for RQ2.

RQ2. What are the perceived benefits and challenges of Design Thinking adoption?	
Question	Result (Professionals)– ['C': Coach - 'P': Participant])
Q5. What were the perceived benefits of DT usage?	(i) creativity (P1, P4) (ii) discovery of innovative solutions (C4, C10, P3) (iii) cost reduction and time optimization (C3, C6, C8) (iv) users' collaboration (C2, C3, C6, C8, P6) (v) focus on end users (C3, C5, C9, P2, P5) and (vi) problem identification (C1, C2, C4, C5, P1, P2)
Q6. What were the perceived challenges of DT usage?	(i) availability of rooms (P1, P2, C7) (ii) lack of valorization (C1, C10, P1, P5) (iii) lack of enough time to solve the problem (C2, C3, P5, P6), and (iv) lack of goals' definitions (P5, C10)
Q7. What the perceived quality aspects are identified when DT is used?	(i) effort maintenance reduction (C5) (ii) quality increase (C4, P6, P2, C10, P4) (iii) focus on final users (C1, C6, C9, C10, P5)

RQ1. How does the adoption of DT with software development take place?

Through our case study, we observed that period of time to use Design Thinking depends on the goal, e.g. a Design Thinking can take months or hours. The participants use it to focus on solving problems, improving solutions or building a mindset among participants. The interpretation of Design Thinking as a mindset and as a process were dominant in coaches' and participants' perceptions. This is in line with research that claims that practicing Design Thinking can lead to the development of a Design Thinking mindset [Brenner et al. 2016].

The professionals reported that Design Thinking was essential to achieve specific project goals, through working spaces with tools and techniques to achieve and explore these goals. We found that Design Thinking, using d-school model, is applied in different ways with software development, showing that there is indeed a need for different levels of Design Thinking application within the software development process. Existing studies suggest that the level of Design Thinking application depends on the project's goal, e.g. [Dobrigkeit et al. 2019].

Additionally, we identified that Design Thinking has a strong focus on understanding the users. Thus, the participants need to work in interdisciplinary teams and explore problems and solutions, matching with the Design Thinking principles mentioned within the literature [Brenner et al. 2016, Brown 2009, Díaz et al. 2014].

Luedeke et al. [Luedeke et al. 2018] argue that deploying the appropriate methods during Design Thinking is a key success factor. All participants of this study use the D-school model, however, each coach uses techniques to achieve the specific goal.

Overall, the participant roles and their understanding of Design Thinking suggests that different roles can conduct Design Thinking, but it is important to consider the experience or good knowledge of the Design Thinking coach. Because of it, the company gives the training to empower the professional to conduct Design Thinking sessions,

called Design Thinking coach. Also, our findings highlight the importance of transfer the knowledge gathered during the Design Thinking among all involved.

RQ2. What are the perceived benefits and challenges of Design Thinking adoption?

Our findings suggest that Design Thinking is a good way to identify the users' and clients' needs, exploring it in depth and developing products focusing on their needs. It can be observed the relationship between participants who have a Design Thinking mindset and implementation of it. We identified that the coaches not only use Design Thinking to improve their software products but also to improve their processes or working spaces.

The perceived benefits were the cost reduction and time optimization because the participants are engaged to build a solution together, so it increases the quality during solution definition. Also, the users' collaboration to discover and gather software requirements, reducing conflicts between people. It confirms that Design Thinking adds value to deliverable solving practical problems, as mentioned in literature [Brown 2008].

Our findings indicate that DT reflects the needs in designed solutions, promoting satisfaction between stakeholders. However, our findings suggest some challenges such as unavailability of rooms to conduct Design Thinking for an extended period of time, coaches unavailability to conduct sessions and enough time to solve the problem.

6. Related Work

Studies are discussing how to use Design Thinking in software development, they focus on understanding an issue or phenomenon, illustrated in Table 5. In our study, we conducted a case study to explore the Design Thinking adoption and the perceived benefits and challenges in a global software company.

Carlgren, Elmquist and Rauth [Carlgren et al. 2014] (column A) argued that Design Thinking can be understood as a user-centered innovation approach, a process to develop new ideas, a mindset, or a combination of mindset and methods. In line with these findings, we identified this understanding among our professionals.

Hehn and Uebernickel [Hehn and Uebernickel 2018] (column B) highlighted the integration among stakeholders, better usability requirements elicitation and different viewpoints shared for an in-depth requirements elicitation during Design Thinking [Hehn and Uebernickel 2018]. Also, Lucena et al. [Lucena et al. 2016] (column C) identified that time spent contributes to improve software development goals and deliver better results and Jensen, Lozano and Steinert [Jensen et al. 2016] (column D) observed that Design Thinking involves the customer on the products' context understanding [Jensen et al. 2016]. De Paula, Amancio and Flores [de Paula et al. 2020] (column E) identified that Design Thinking adds value to the product, project, people and whole organization.

In our study (column F), we identified the same characteristics mentioned above and cost reduction because the stakeholders are engaged during the Design Thinking. De Paula, Amancio and Flores highlighted the known solutions emerged and many ideas generated were executed as challenges [de Paula et al. 2020]. Our findings suggest that innovative solutions can emerge during the DT, however, some ideas are not executed too.

Table 5. Related work

	A) Carlgren, Elmquist and Rauth	B) Hehn and Uebernickel	C) Lucena et al.	D) Jensen, Lozano and Steinert	E) de Paula, Amancio and Flores	F) Our Study
Objective	Explored how DT is used in large organizations	Identified the potential between DT and Requirements Engineering	Identified how IBM uses DT	Described the research fields behind the DT methods	Evaluated a DT light version usage at IBM	Describes the adoption of DT in a global IT company
Method	Interviews with participants from 16 companies	Case study	Survey	Case study with professionals from SAP	Survey with two groups from IBM	Case study with 16 professionals from a global IT company
RQ1. Adoption	DT as a user-centered approach, a process to develop new ideas, a mindset, or a combination of mindset and methods	n/a	DT is used for product development	71% use Scrum with DT and 14% Kanban	n/a	DT is used in software products, software improvements, processes, user experience identification and solution discovery
RQ2. Perceived benefits	n/a	Integration among stakeholders, better usability requirements elicitation and different viewpoints shared for an in-depth requirements elicitation	The time spent contributes to improve agile software development goals and deliver better results	Provide an interactive development with stakeholders, gain a holistic problem overview and involve the customer on the products' context understanding	DT adds value to the product, project, people and whole organization.	Cost reduction, time optimization, users' collaboration to discover and gather the software requirements, conflicts reduction and increased quality
RQ2. Perceived challenges	n/a	n/a	n/a	n/a	Participants used already known solutions and many of the ideas generated were not executed	Unavailability of rooms to use during DT, coaches unavailable to conduct DT and enough time to solve the problem

7. Final Considerations

This study aims to describe how the adoption of DT with software development takes place in global information technology. This way, we identified how DT, through working spaces, is used, what techniques and tools are used, which artifacts are produced and which professionals (or team roles) get involved in DT activities. Also, we identified the perceived benefits and challenges of DT usage. With these parameters in mind, we conducted semi-structured interviews with 10 coaches and 6 participants, totaling 16

professionals who have experience with DT in software development.

Our findings can serve as a guide for practitioners on how to set up and implement DT activities. Thus, we focus on answering the research questions (RQ) *"RQ1. How does the adoption of DT with software development take place?"* and *"RQ2. What are the perceived benefits and challenges of DT adoption?"*.

As a result, we identified that DT is used in software products, software improvements, processes, user experience identification and solution discovery. Also, there are perceived benefits like cost reduction, time optimization, users' collaboration to discover and gather the software requirements, conflict reductions and increased quality. However, there are challenges such as unavailability of rooms to use during DT, coaches unavailable to conduct DT and enough time to solve the problem.

The study has limitations inherent to any empirical study. For instance, we interviewed only 16 employees, which might not represent the company view on the topic. To mitigate that we did select employees with different levels of expertise, time working in the company, and from different company areas.

For future studies, we intend to explore in-depth knowledge about DT and its tools, and techniques used in software development, as well as conduct case studies with other software companies to create a wider vision of DT integrated to software process, its benefits, challenges and ways of the application.

Acknowledgements

This study was partially financed by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001.

References

Brenner, W., Uebernickel, F., and Abrell, T. (2016). Design Thinking as Mindset, Process, and Toolbox. In *Design Thinking for Innovation: Research and Practice*, pages 3–21. Springer.

Brown, T. (2008). Design Thinking. *Harvard Business Review*, 86:84–92.

Brown, T. (2009). Change by Design: How Design Thinking Transforms Organizations and Inspires Innovation. page 272. HarperCollins, New York, United States.

Carlgren, L., Elmqvist, M., and Rauth, I. (2014). Exploring the use of Design Thinking in Large Organizations: Towards a Research Agenda. *Swedish Design Research Journal*, 11:55–63.

Council, D. (1944 (accessed October 10, 2020)). The Design Process: What is the Double Diamond? Available in: http://designcouncil.org.uk/.

de Paula, T. R., Amancio, T. S., and Flores, J. N. (2020). Design Thinking in Industry. *IEEE Software*, 37(02):49–51.

Dobrigkeit, F., de Paula, D., and Uflacker, M. (2019). InnoDev: A Software Development Methodology Integrating Design Thinking, Scrum and Lean Startup. In *Design Thinking Research*, pages 199–227. Springer.

Dybå, T., Prikladnicki, R., Rönkkö, K., Seaman, C., and Sillito, J. (2011). Qualitative Research in Software Engineering. *Empirical Software Engineering*, 16(4):425–429.

Díaz, P., Aedo, I., and Cubas, J. (2014). CoDICE: Balancing Software Engineering and Creativity in the Co-Design of Digital Encounters with Cultural Heritage. In *Proceedings of the Workshop on Advanced Visual Interfaces*, pages 253–256, Como, Italy. ACM.

Hasso-Plattner Institute ((accessed October 10, 2020)). *What is Design Thinking?* Available in: https://hpi-academy.de/en/design-thinking/what-is-design-thinking.html.

Hehn, J., Mendez, D., Uebernickel, F., Brenner, W., and Broy, M. (2020). On Integrating Design Thinking for Human-Centered Requirements Engineering. *IEEE Software*, 37(2):25–31.

Hehn, J. and Uebernickel, F. (2018). The Use of Design Thinking for Requirements Engineering: An Ongoing Case Study in the Field of Innovative Software-Intensive Systems. In *Proc. of the Int'l Requirements Eng. Conf.*, pages 400–405, Banff, Canada. IEEE.

Jensen, M. B., Lozano, F., and Steinert, M. (2016). The Origins of Design Thinking and the Relevance in Software Innovation. In *Procedings of the Conference on Product-Focused Software Process Improvement*, pages 675–678, Trondheim, Norway. Springer.

Kitchenham, B. and Pfleeger, S. L. (2002). Principles of Survey Research Part 4: Questionnaire Evaluation. *SIGSOFT Software Engineering Notes*, pages 20–23.

Krippendorff, K. (2018). *Content Analysis: An Introduction to Its Methodology*. Sage, New York, United States.

Lucena, P., Braz, A., Chicoria, A., and Tizzei, L. (2016). IBM Design Thinking Software Development Framework. In *Proceedings of the Brazilian Workshop on Agile Methods*, pages 98–109, Curitiba, Brazil. Springer.

Luedeke, T., Köhler, C., Conrad, J., Grashiller, M., Sailer, A., and Vielhaber, M. (2018). Cyber-Physical Systems/Property-Driven Design in the context of Design Thinking and Agile Development of Cyber-Physical Systems: Use Cases and Methodology. In *Proceedings of the NordDesign Conference*, pages 1–24, Linköping, Sweden.

Mateusz Dolata, Falk Uebernickel, G. S. (2017). The power of words: Towards a methodology for progress monitoring in design thinking projects. In *International Conference on Wirtschaftsinformatik*, pages 1190–1204, St. Gallen, Switzerland.

Runeson, P. and Höst, M. (2009). Guidelines for Conducting and Reporting Case Study Research in Software Engineering. *Empirical software engineering*, 14(2):131.

Subih, M. A., Malik, B. H., Mazhar, I., Yousaf, A., Sabir, M. U., Wakeel, T., Izaz-ul Hassan, W. A., Bilal-bin Ijaz, M. S., and Nawaz11, H. (2019). Comparison of Agile Method and Scrum Method with Software Quality Affecting Factors. *International Journal of Advanced Computer Science and Applications*, 10(5):531–535.

Thoring, K., Müller, R. M., et al. (2011). Understanding Design Thinking: A Process Model Based on Method Engineering. In *Proc. of the Int. Conference on Engineering and Product Design Education*, pages 493–498, London, UK. Design Society.

Vetterli, C., Brenner, W., Uebernickel, F., and Petrie, C. (2013). From Palaces to Yurts: Why Requirements Engineering Needs Design Thinking. *IEEE Internet Computing*, pages 91–94.

Avaliação de Deep Learning para Predição de Mensagens Processadas pela Plataforma de Integração Guaraná

Matheus H. Rehbein[1], **Rafael Z. Frantz**[1]

[1]Departamento de Ciências Exatas e Engenharias
Universidade Regional do Noroeste do Estado do Rio Grande do Sul
98700-000 – Ijuí – RS – Brasil

matheus.rehbein@sou.unijui.edu.br, rzfrantz@unijui.edu.br

Abstract. *Companies have many applications in their software ecosystems that need to be integrated. Integration frameworks allow integrating with efficiency those applications, however there are parameterization that must be defined manually, like the number of available threads. This paper presents an experimental study that created and evaluated deep learning models for realizing the prediction of the number of messages that will be processed by using a determinate number of threads and the message rate as input in the model, to allow the parametrization of the number of threads automatically and in real time.*

Resumo. *Empresas possuem diversas aplicações em seus ecossistemas de softwares que precisam ser integradas. Plataformas de integração permitem uma eficiência para realizar a integração, entretanto, algumas configurações devem ser definidas de forma manual, como o número de threads disponíveis. Este artigo apresenta um estudo experimental que criou e avaliou modelos de deep learning para realizar a predição do número de mensagens que serão processadas a partir de um determinado número de threads e da taxa de mensagens, para permitir uma definição de forma automática e em tempo de execução do número de threads nas plataformas de integração.*

1. Introdução

Para dar suporte aos seus processos de negócio, geralmente as empresas utilizam diversas aplicações em seu ecossistema de software [Manikas 2016]. Para agilizar as rotinas diárias da empresa, é necessário que as aplicações operem de forma integrada [Hohpe and Woolf 2004], portanto é necessário integrá-las. Atualmente, existem diversas plataformas que visam auxiliar o desenvolvimento e manutenção de integração entre diferentes aplicações, abstraindo as principais atividades envolvidas na integração. Destacam-se Apache Camel [Ibsen and Anstey 2018], Guaraná [Frantz et al. 2016], Mule ESB [Dossot et al. 2014], e Spring Integration [Fisher et al. 2012]. Essas são plataformas desenvolvidas a partir do paradigma de mensageria, que visa realizar a troca de informações entre as aplicações por meio de mensagens [Frantz et al. 2020]. Algumas configurações são necessárias para obter maior desempenho durante a integração, como a quantidade de *threads*.

Quando realizado uma superestimação ou subestimação da quantidade de *threads* a aplicação pode ficar lenta ou causar uma sobrecarga no servidor. A partir dessas questões, a configuração das plataformas de integração se torna fundamental, e usar uma

configuração de forma automática pela própria plataforma de integração é necessário, para isso modelos matemáticos específicos para cada processo de integração se constituem em uma boa alternativa de solução ao processo de configuração.

O campo de *machine learning* é constituído por modelos capazes de obter conhecimento a partir de dados de exemplos, e então realizar predições [Géron 2017]. Sendo um dos principais modelos o modelo de *deep learning*, que é composto por múltiplas camadas de processamento capazes de aprender com diversos níveis de abstração. Para realizar uma boa definição da quantidade de *threads* nas plataformas de integração, o modelo de *deep learning* pode ser utilizados, a partir de predições da quantidade de mensagens que serão processadas. O conhecimento inferido no modelo pode ser obtido por meio de logs e de experimentos realizados anteriormente.

Neste artigo, será utilizado um conjunto de dados (*dataset*) extraído da plataforma Guaraná, que foi desenvolvida pelo grupo de pesquisa, para criar modelos capazes de predizer a quantidade de mensagens que serão processadas. Além da disso, será realizada a comparação entre os modelos de *deep learning*, com diferentes configurações para encontrar qual configuração teve maior assertividade. O restante do artigo está organizado da seguinte forma: na Seção 2 será apresentado um conhecimento prévio sobre os assuntos, seguida pela Seção 3 que apresenta os trabalhos relacionados; a Seção 4 apresentará a metodologia utilizada; o objeto de estudo será apresentado na Seção 5, e os experimentos na Seção 6; por fim, será realizada as conclusões do artigo na Seção 7.

2. Background

Nesta Seção serão apresentados os conceitos chaves para entendimento do trabalho. Será descrito o desenvolvimento da técnica de *deep learning*, seguido pela evolução histórica. Em seguida, é apresentado o Guaraná e suas principais definições.

2.1. Deep Learning

Rede Neural é uma técnica de *machine learning* composta por uma série de elementos (chamados de neurônios) caracterizados como elementos simples e possui como principal característica a forte conexão entre os neurônios; essa ideia vem da forma como o cérebro humano é estruturado. O neurônio é constituído por: sinapses, entradas, pesos, função de ativação e saída. Sendo que as sinapses são as ligações lógicas entre os neurônios. Haverão diversas sinapses em cada neurônio; as entradas consistem nos valores que serão inferidos no neurônio; o peso é um valor gerado pelo modelo, que multiplicado pela entrada será utilizado para auxiliar na excitação/inibição do neurônio. O peso é iniciado de forma randômica e posteriormente será ajustado no treinamento. O peso não é atribuído diretamente no neurônio, mas sim em cada sinapse; uma função de ativação é utilizada para realizar o processamento interno no neurônio, para representar se o neurônio será excitado ou inibido; por fim a saída é o resultado da função de ativação [Géron 2017].

O *perceptron* é um exemplo de uma arquitetura de rede neural, na qual suas entradas e saídas são números e cada entrada é associada com um peso [Géron 2017]. Uma rede *perceptron* é composta por apenas uma camada onde todos os neurônios estão internamente conectados. Muitos problemas foram encontrados na rede *perceptron*, e foram solucionados com um conjunto de redes *perceptron*, chamada de *multi-layer perceptron*. Uma rede *multi-layer perceptron* consiste em diversas redes *perceptron* interligadas, ou

seja, com diversas camadas. Sendo necessário uma camada de entrada, uma ou mais camadas invisíveis, e uma camada de saída, como representado na Figura 1. É considerado um modelo de *deep learning* quando uma rede possui duas ou mais camadas invisíveis.

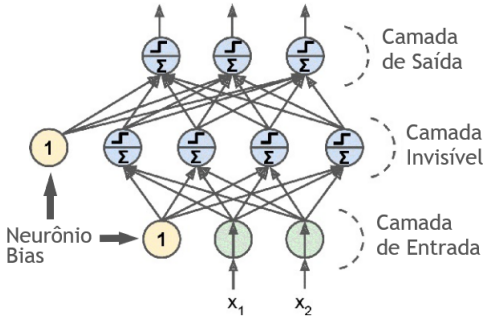


Figura 1. Multi-Layer Perceptron. Adaptado de: [Géron 2017]

Cada camada de um modelo de *deep learning* é constituída por um otimizador, quantidade de neurônios, um neurônio *bias* (que está sempre inferindo um valor constante, como mostrado na Figura 1), e uma função de ativação.

Otimizador é uma função aplicada no treinamento do modelo que visa diminuir as perdas. Ou seja, busca diminuir o erro do valor predito no momento atual do treinamento e do valor real. O otimizador é um dos principais elementos de uma rede neural, tendo em vista que ele é o responsável por ajustar os pesos em cada neurônio. Um dos principais otimizadores é o Adam [Kingma and Ba 2014], que se destaca devido a uma série de fatores, como: eficiência computacional, é adequado para trabalhar com grande quantidade de dados, possui baixa quantidade de consumo de memória, velocidade.

A função de ativação é uma função responsável pela excitação/inibição dos neurônios na camada. A ativação significa que o neurônio foi ativado. Nela o objetivo é calcular os pesos e então verificar se o neurônio será excitado ou não. Essas funções podem variar a saída; algumas funções, como *sigmoid*, possuem valores de saída entre 0 e 1, outras variam de 0 até infinito, e também saídas binárias (0 ou 1).

2.2. Guaraná

Guaraná [Frantz et al. 2016] é uma plataforma desenvolvida com o objetivo de auxiliar a realização de integração de aplicações. Ele realiza a integração a partir da troca de mensagens entre as aplicações. O Guaraná proporciona uma DSL, um SDK, e um motor.

A DSL permite modelar soluções de integração com um alto nível de abstração. Seus construtores estão apresentados na Figura 2. São eles: aplicação, tarefa, slot, processo de integração, porta de entrada, porta de saída, porta de solicitação, e porta de resposta. Uma aplicação é o software que será integrado; a Tarefa é o elemento que implementa uma atividade atômica que é executada sob as mensagens; o Slot é responsável por interligar as tarefas, permitindo o processamento assíncrono do processo de integração; o Processo de Integração implementa um *workflow* de Tarefas dessincronizadas por Slot. Esse processo se comunica com os meios externos a partir de portas; a Porta de Entrada é um canal de comunicação que permite ao processo de integração obter informações de uma aplicação, enquanto que a Porta de Saída é responsável por enviar informações às aplicações; a Porta de Solicitação é utilizada por uma tarefa para enviar solicitações e

receber respostas de/para uma aplicação; por fim, a Porta de Resposta é usada por uma tarefa para receber solicitações e responder de/para uma aplicação [Frantz 2012].

Notação	Conceito	Notação	Conceito
	Aplicação		Porta de Solicitação
	Processo de Integração		Porta de Resposta
	Porta de Entrada		Tarefa
	Porta de Saída		Slot

Figura 2. Descrição dos conceitos da DSL. Fonte: [Frantz et al. 2015]

O SDK consiste em um conjunto de classes que permitem transformar o modelo conceitual em código executável. O motor de execução é a parte responsável por executar o código resultante da transformação do modelo conceitual. Um elemento fundamental durante a execução é o número de *threads*. Engenheiros de Softwares necessitam configurar o motor de execução, sendo uma das configurações a quantidade de *threads*. No Guaraná, cada tarefa é executada de forma sequencial, ou seja, para executar a tarefa T2 da mensagem X é necessário que a T1 da mesma mensagem ter sido executada. Devido a sua política de tarefas, é possível que diversas mensagens sejam executadas paralelamente, pois cada *thread* pode executar a mesma tarefa para diferentes mensagens [Frantz 2012].

3. Trabalhos Relacionados

Em [Nemirovsky et al. 2017], os pesquisadores propuseram um escalonador de CPU heterogêneo, que visa maximizar o *throughput* a partir do uso de redes neurais artificiais, sendo que obtiveram uma melhoria de 25% a 31% em relação aos escalonadores heterogêneos convencionais de CPU e memória intensiva. Embora o trabalho dos pesquisadores tenha sido de otimização com técnica de inteligência artificial, ele se diferencia deste artigo pois este está dentro do contexto de integração de aplicações.

Uma estratégia de programação que usa um Algoritmos Genéticos para escalonar tarefas heterogêneas em processadores heterogêneos para minimizar o tempo total de execução foi proposta por [Page and Naughton 2005]. Embora seja um problema de otimização e tenham resolvido ele com inteligência artificial, não foi utilizado no contexto de integração de aplicações e também não foi utilizada a técnica de *deep learing*.

No trabalho [Beer and Hassan 2018] os pesquisadores utilizaram redes neurais artificiais na prevenção de ataques SOA em *web services* RESTful. Embora o trabalho dos pesquisadores seja no campo de integração de aplicações e tenha utilizado inteligência artificial, ele se difere deste pois foi no contexto de segurança e não de otimização.

4. Metodologia

A pesquisa realizada neste artigo é considerada aplicada, tendo em vista que busca gerar conhecimento para aplicações práticas voltados a resoluções de problemas específicos. Neste artigo utilizou-se o método científico indutivo com o objetivo de generalizar a resposta de qual técnica possui maior assertividade a partir de experimentos sob um conjunto

de dados específico. Essa generalização é verificada pois não foram realizados experimentos com todas as combinações possíveis. Seus objetivos de estudo são explicativos, pois busca realizar o entendimento dos resultados encontrados. Os procedimentos são experimentais sob parâmetros controlados, juntamente com uma abordagem quantitativa.

Para realizar a implementação dos modelos utilizou-se a linguagem *Python*, juntamente com a biblioteca *TensorFlow* [Abadi et al. 2016]. O desenvolvimento foi separado da seguinte forma: i) análise e pré-processamento do *dataset*; ii) implementação do problema com a biblioteca *TensorFlow*; iii) execução e coleta dos resultados.

5. Objeto de Estudo

A Figura 3 apresenta a solução de um problema encontrado em um café, que é utilizado como *benchmark* no contexto de integração de aplicações. O processo de integração da solução Café inicia com os pedidos inseridos na integração pela porta de entrada denominada *Orders* e posteriormente encaminhados a tarefa *T1* do tipo *Splitter* que dividirá o pedido original em dois pedidos distintos, um contendo bebidas quentes e outro geladas. Em seguida, os pedidos gerados são encaminhados para a tarefa *T2* do tipo *Dispacher*, que encaminhará os pedidos para os seus respectivos balcões, bebidas geladas e bebidas quentes. Após essa etapa é então iniciado o processo assíncrono pelos baristas, sendo possível realizar o preparo tanto da bebida quente, quanto da bebida gelada.

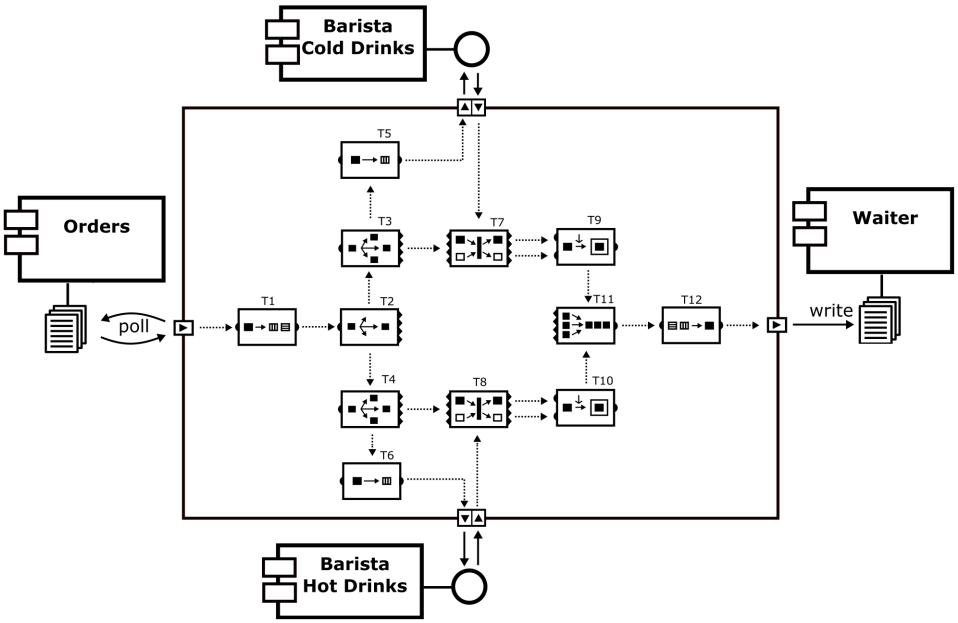


Figura 3. Solução Café implementada pelo Guaraná DSL. Fonte: [Frantz 2012]

Após passar pela tarefa *T2* o pedido é encaminhado a tarefa *T3* do tipo *Replicator*, que gerará duas cópias do pedido recebido por ela, uma será encaminhada a tarefa *T5*, *Translator* e a outra à tarefa *T7*, *Correlator*. As tarefas do tipo *Translator* aplicam modificações no pedido recebido de maneira a criar um novo pedido que possa ser lido pela aplicação destino, que no modelo é chamada de *Barista Cold Drinks*, que será responsável pela preparação do pedido do cliente. Tarefas *Correlator* fazem com que a cópia da mensagem só seja encaminhada a tarefa seguinte no momento em que o *Correlator* receber duas ou mais mensagem com o mesmo identificador.

A tarefa *T9* é do tipo *Context Based Enricher* que adicionará a bebida ao pedido que foi correlacionado pela tarefa *T7*. Posteriormente, será encaminhado para a tarefa *T11*, um *Merger* que juntará as mensagens de diferentes slots em apenas um, e então encaminhará para tarefa *T12*, do tipo *Aggregator*, que juntará as mensagens do slot em uma única mensagem, essa mensagem será composta pelo pedido e as bebidas, e então será encaminhado ao garçom para realizar a entrega.

6. Experimentos

Nesta Seção são apresentados os experimentos. São discutidos os seguintes assuntos: questão de pesquisa e hipóteses, variáveis, ambiente de execução, execução e coleta dos dados, resultados, discussão dos resultados, e ameaças a validade.

6.1. Questão de Pesquisa e Hipóteses

Neste experimento, busca-se responder como o modelo de *deep learning* se comporta em predições a partir do mesmo conjunto de dados de treinamento utilizado a variação de seus parâmetros. A hipótese H_0 é de que *Deep Learning não possui assertividade alta para a predição de mensagens processadas pela plataforma de integração Guaraná*, enquanto que H_1 busca apresentar que *Deep Learning apresenta uma alta assertividade na predição de mensagens processadas pela plataforma de integração Guaraná, a fim de ser possível a sua utilização como base para a definição da quantidade ideal de threads*.

6.2. Variáveis

As variáveis independentes deste experimento são responsáveis pelas configurações realizadas nos modelos: quantidade de neurônios e camadas. A fim de realizar a combinação entre as variações das variáveis, foram realizadas as combinações das variações. A quantidade de camadas variou de um a seis, enquanto que para a quantidade de neurônios utilizou-se a variação exponencial de base dois, iniciando com o expoente dois e finalizando em oito, sendo os valores 2 e 256 o mínimo e o máximo, respectivamente.

As saídas de cada modelo representam a assertividade que ele teve na predição da quantidade de mensagens processadas, ou seja, as variáveis dependentes que foram utilizadas neste experimento para realizar a comparação entre os modelos. Foi utilizada a métrica de avaliação *r-squared*, que tem sua variação ligada diretamente ao quão corretas/erradas foram as predições, o valor retornado estará na escala de zero e um.

6.3. Ambiente de Execução

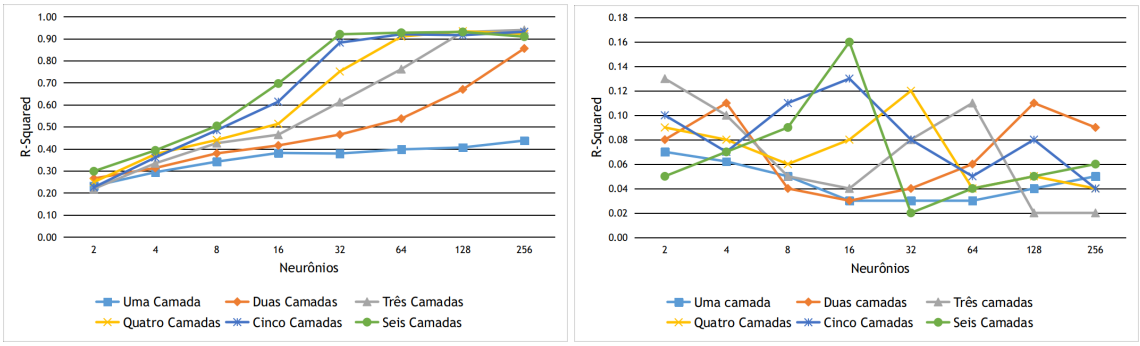
A Execução foi realizada em um computador com o processador Intel(R) Core(TM) $i3-3217U$ CPU, que opera na frequência de 1.80 GHz, possui 2 núcleos e 4 *threads*, e 4 GB de memória. O Sistema Operacional utilizado foi o Deepin/Debian $4.15.0-30$, 64 bits.

6.4. Execução e Coleta de Dados

Cada modelo foi executado 25 vezes e coletadas suas assertividades, a fim de obter as médias. Alguns modelos não foram capazes de realizar o aprendizado dos dados, e então realizou-se a exclusão dos valores no cálculo. Os dados foram gerados a partir de cada execução do modelo e então salvos em um arquivo CSV constando todas as repetições da combinação dos parâmetros. A quantidade de arquivos gerados é representado pela equação: Quantidade Camadas x Quantidade de Variações de Neurônios (6x8): 48.

6.5. Resultados

A assertividade de um modelo é o indicador de quão bom é o modelo, sendo assim os resultados utilizados para análise dos dados foram obtidos a partir da média da assertividade de cada modelo. A Figura 4a apresenta as assertividades com os modelos de *deep learning*, enquanto que a Figura 4b apresenta os desvios-padrões dos modelos. Sendo separados pela quantidade de camadas e neurônios. A visualização dos dados deve ser realizada a partir da quantidade de camadas (cor e formato) e de neurônios (eixo x). A assertividade está no eixo y e deve ser utilizada para realizar a comparação dos modelos.



(a) Assertividade dos Modelos **(b) Desvio Padrão dos Modelos**

Figura 4. Assertividade e Desvio Padrão dos modelos

Com os resultados obtidos, é possível observar que todos os modelos com três ou mais camadas e pelo menos 128 neurônios tiveram a assertividade acima de 90%, enquanto que ao utilizar apenas menos camadas as assertividades foram menores. Todos os modelos apresentaram baixa assertividade com dois neurônios. Também é visível o crescimento dos assertos dos modelos conforme o incremento de suas variáveis.

6.6. Discussão

Na Figura 4a, é possível verificar a tendência de incremento da assertividade em relação a quantidade de neurônios, o mesmo ocorre para a quantidade de camadas. O crescimento é explicado devido a maior possibilidade de interconexões entre os neurônios, permitindo assim melhores ajustes de seus pesos. O modelo de uma camada que obteve a maior assertividade foi o modelo com 256 neurônios, obtendo 0.44 de assertividade em seus testes. O modelo mais próximo foi o modelo de 128 neurônios, que obteve a assertividade de 0.41. Enquanto isso, o modelo de apenas 2 neurônios obteve o pior desempenho, de 0.23. Além disso, é possível verificar que o comportamento do desvio padrão foi mais suave em relação aos modelos de mais camadas, mostrando que a variação média da assertividade esteve mais balanceada em relação aos modelos com mais camadas.

Os modelos de duas camadas obtiveram um crescimento de semelhante ao crescimento exponencial, havendo uma diferença considerável da assertividade entre os modelos de 2 e 256 neurônios, sendo possível verificar um crescimento da assertividade com o incremento da quantidade de neurônios nos modelos. O melhor resultado desse modelo foi de 0.85 enquanto que o pior foi de 0.26, resultando na diferença de 0.59. Existe ainda uma diferença de 0.19 de assertividade entre os modelos de 128 (0.67) e 256 (0.86) neurônios. Essa diferença não é vista entre os outros modelos com essa quantidade de camadas, entretanto ela pode ser explicada pois o modelo de 256 neurônios possui o dobro

de neurônios se comparado ao outro modelo, também é possível verificar que há um valor de 0.11 no desvio padrão do modelo de 128 neurônios, enquanto que o modelo de 256 neurônios teve 0.09, ou seja, os modelos tiveram uma variação próxima.

Um crescimento linear é apresentado na variação de 16 até 128 neurônios nos modelos de três camadas, variando de 0.46 para 0.93. A maior assertividade desses modelos foi do modelo de 256 neurônios, com 0.94. Próximo ao modelo de 256 neurônios, esteve o modelo de 128 neurônios, que obteve a assertividade de 0.93, apresentando a primeira estabilização da assertividade entre os modelos. A menor assertividade dos modelos dessa quantidade de camadas, foi do modelo com apenas 2 neurônios, que possui apenas 0.22. É possível verificar o alto desvio padrão com 2 e 64 neurônios, mostrando que os valores variaram mais em relação aos outros modelos com essa quantidade de camadas.

Assim como nos modelos de três camadas, a assertividade dos modelos de quatro camadas tenderam a estabilização após um determinado número de neurônios, que nesse modelo foi de 64, havendo três modelos que ficaram acima de 90% de assertividade, os modelos com 64, 128 e 256 neurônios. A maior assertividade do modelo de quatro camadas ocorreu quando teve 128 neurônios disponíveis para suas interconexões, obtendo o valor de 0.93, enquanto que com 256 neurônios a assertividade ficou em 0.90. Entretanto, é possível verificar que com 32 neurônios houve um elevado desvio padrão, mostrando que houve uma variação maior com essa configuração.

Com cinco camadas é possível visualizar um crescimento da assertividade dos modelos com até 32 neurônios. Sendo que três modelos estiveram acima de 0.90 de assertividade, os modelos de 64, 128 e 256. Entretanto, o modelo de 32 neurônios esteve próximo dessa assertividade, obtendo 0.88. A menor assertividade ficou em 0.22 e a maior em 0.93, com 2 e 256 neurônios, respectivamente. Também é visível a diferença da assertividade entre os modelos de 16 e 32 neurônios. Já o desvio padrão se mostrou instável em relação aos outros modelos, sendo que o maior foi de 0.13 com 16 neurônios, o que pode justificar a diferença em relação ao modelo de 32 neurônios.

Por fim, utilizando seis camadas, é possível verificar que quatro modelos estiveram acima de 0.90 de assertividade, os modelos de 32, 64, 128 e 256 neurônios, obtendo o *r-squared* de 0.91, 0.92, 0.93 e 0.90, respectivamente. O modelo de menor assertividade foi o modelo com 2 neurônios, que obteve 0.29 de assertividade. Além disso, o desvio padrão se mostrou instável, sendo que com 16 e 32 neurônios obteve os valores de 0.16 e 0.02, respectivamente, sendo o maior e o menor valor entre todos os modelos.

Observando os resultados, é possível observar que o modelo de maior assertividade foi o de 3 camadas e 256 neurônios. Além disso, foi um dos modelos que menos variou a assertividade, com 0.02 de desvio padrão. Portanto, é possível verificar que embora outros modelos tiveram mais camadas, eles também tiveram maior variação da assertividade, ou seja, um desvio padrão maior.

6.7. Ameaças à Validade

Este estudo se mostrou válido pois com ele foi possível encontrar qual modelo se melhor adéqua na realização de predições da quantidade de mensagens que serão processadas pela plataforma de integração guaraná. Utilizando os modelos de *deep learning* pode-se realizar predições com alto nível de confiabilidade, utilizar as predições durante a execução da plataforma de integração e realizar a alteração da quantidade de *threads* em tempo

real. Ao realizar a alteração das variáveis independentes dos modelos de *deep learning*, é verificado que as variáveis dependentes se alteraram de forma positiva.

Neste estudo realizou-se a utilização de um *dataset* de uma experimentação da solução de integração café, sendo assim, não é possível generalizar os resultados obtidos para todas as soluções de integração. Portanto, essa é a principal ameaça aos resultados.

Para que este artigo possa responder a questão de pesquisa e testar as hipóteses, algumas etapas foram realizadas. Levantamento do referencial teórico; as informações sobre o objeto de estudo são apresentadas, seguido pelas configurações utilizadas, e também os diferentes cenários de execução são apresentadas. Por fim, é realizado a execução de diferentes combinações para representar diferentes modelos que possam ser utilizados.

Para apresentar resultados que possuam conclusões corretas, diversos cuidados foram levados em consideração. A quantidade de execução de cada modelo, que representa um número que permite realizar análise e conclusões de forma confiável. As alterações das variáveis independentes foram realizadas a fim de verificar com maior intensidade as alterações dos resultados. Por fim, a separação dos dados de treinamento foi feita de forma randômica, para que não tenha causado resultados inconsistentes.

7. Conclusão

Muitos desafios são encontrados e resolvidos pelas plataformas de integração, como a definição do número ideal de *threads*. Neste artigo, utilizou-se modelos de *deep learning* para realizar a predição da quantidade de mensagens que seriam processadas a partir da taxa de entrada e da quantidade de *threads* configuradas pela plataforma de integração Guaraná. A técnica *r-squared* foi utilizada para mensurar os modelos. O objetivo do artigo foi encontrar a configuração ideal para um modelo de *deep learning* na realização da predição do número de mensagens que seriam processadas a partir da taxa de entrada e da quantidade de *threads* disponíveis. Com esse modelo, um gerenciador de *threads* automático e em tempo de execução poderia ser construído, para que fosse possível uma melhor definição do número de *threads* que fosse capaz de executar todas as mensagens que chegassem, além de não sobrecarregar o servidor. A partir das distintas configurações de modelos, é possível verificar que o aumento da quantidade de camadas e neurônios é capaz de aumentar a precisão das predições, até chegar a estabilidade de, no máximo, 94%. Por outro lado, a baixa assertividade do modelo de apenas uma camada também é notável, pois o número de interconexões entre os neurônios é baixo.

Agradecimentos

À Fundação de Amparo à Pesquisa do Estado do Rio Grande do Sul (FAPERGS) (termo de outorga número 17/2551-0001206-2) e Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES).

Referências

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y., and Zheng, X. (2016). Tensorflow: A system for large-scale machine learning. In *12th USENIX Symp. on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283.

Beer, M. I. and Hassan, M. F. (2018). Adaptive security architecture for protecting restful web services in enterprise computing environment. *Service Oriented Computing and Applications*, 12(2):111–121.

Dossot, D., d'Emic, J., and Romero, V. (2014). *Mule in action*. Manning Publications Co.

Fisher, M., Partner, J., Bogoevici, M., and Fuld, I. (2012). *Spring integration in action*. Manning Publications Co.

Frantz, R. Z. (2012). *Enterprise application integration: an easy-to-maintain model-driven engineering approach*. PhD thesis, Universidad de Sevilla.

Frantz, R. Z., Corchuelo, R., Basto-Fernandes, V., Rosa-Sequeira, F., Roos-Frantz, F., and Arjona, J. L. (2020). A cloud-based integration platform for enterprise application integration: a model-driven engineering approach. *Software: Practice and Experience. (in-press)*, 1(1):1–25.

Frantz, R. Z., Corchuelo, R., and Roos-Frantz, F. (2016). On the design of a maintainable software development kit to implement integration solutions. *Journal of Systems and Software*, 111(1):89–104.

Frantz, R. Z., Sawicki, S., Roos-Frantz, F., Yevseyeva, I., and Emmerich, M. (2015). On using markov decision processes to model integration solutions for disparate resources in software ecosystems. In *Intl. Conf. on Enterprise Information Systems (ICEIS)*, pages 260–268.

Géron, A. (2017). *Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems*. "O'Reilly Media, Inc.".

Hohpe, G. and Woolf, B. (2004). *Enterprise integration patterns: Designing, building, and deploying messaging solutions*. Addison-Wesley Professional.

Ibsen, C. and Anstey, J. (2018). *Camel in action*. Manning Publications Co.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. In *3rd Intl. Conf. for Learning Representations (ICLR)*, pages 1–13.

Manikas, K. (2016). Revisiting software ecosystems research: A longitudinal literature study. *Journal of Systems and Software*, 117:84–103.

Nemirovsky, D., Arkose, T., Markovic, N., Nemirovsky, M., Unsal, O., and Cristal, A. (2017). A machine learning approach for performance prediction and scheduling on heterogeneous cpus. In *29th Intl. Symp. on Computer Architecture and High Performance Computing (SBAC-PAD)*, pages 121–128.

Page, A. J. and Naughton, T. J. (2005). Framework for task scheduling in heterogeneous distributed computing using genetic algorithms. *Artificial Intelligence Review*, 24(3):415–429.

Open Coding Tool: Uma Ferramenta de Codificação Colaborativa para Análise de Dados Qualitativos

Maurício dos Santos Escobar, Alex Severo Chervenski e Andréa Sabedra Bordin

[1]Curso de Engenharia de Software, Universidade Federal do Pampa - UNIPAMPA
Av. Tiaraju, 810 - Ibirapuitã, Alegrete - RS, 97546-550

mauricioescobar.aluno@unipampa.edu.br, alex.chervenski@gmail.com

andreabordin@unipampa.edu.br

Abstract. *In software engineering research problems have been studied from a qualitative perspective. Several studies adopt qualitative text-based methods, which have a procedure called coding, through which data are broken down to produce new knowledge. As it is an analytical task, it requires people and execution time. There are some software to perform this procedure. However, most are not free and do not allow coding to be performed by a group of people. This article presents a collaborative encoding tool for textual data. In a case study, the use of the tool significantly reduced the coding time, without prejudice to the quality of the generated codes.*

Resumo. *Na engenharia de software problemas de pesquisa vêm sendo estudados sob a ótica qualitativa. Vários estudos que adotam métodos qualitativos baseados em texto, possuem um procedimento denominado codificação, através do qual os dados são desmembrados para a produção de um novo conhecimento. Por ser uma tarefa analítica, a codificação demanda pessoas e tempo de execução. Existem alguns softwares para realização desse procedimento, no entanto, a maioria não é gratuita e não permite que a codificação seja executada por um conjunto de pessoas. Este artigo apresenta uma ferramenta de codificação colaborativa de dados textuais. Em um estudo de caso, o uso da ferramenta diminuiu significativamente o tempo de codificação, sem prejuízo para a qualidade dos códigos gerados.*

1. Introdução

Métodos de pesquisa qualitativa foram desenvolvidos nas Ciências Sociais para habilitar pesquisadores a entender pessoas e os contextos culturais e sociais no qual eles vivem [Denzin and Lincoln 2005]. De acordo com [Gibbs 2009], uma característica essencial dos dados qualitativos é que eles se originam de praticamente qualquer forma de comunicação humana – escrita, auditiva, ou visual. No entanto, o tipo mais comum de dado qualitativo usado em análises é o texto, que pode ser transcrições de entrevistas, notas de campo de trabalho etnográfico ou outros tipos de documentos.

Na Engenharia de Software questões complexas podem, quando estudadas sob a ótica qualitativa, gerar hipóteses bem fundamentadas e resultados que incorporam a complexidade do fenômeno estudado. Além disso, estudos qualitativos oferecem explicações mais ricas e novas oportunidades para estudos futuros. Eles são apropriados quando as variáveis não estão definidas ou quantificadas e existe pouco estudo

teórico ou empírico [Dybå et al. 2011]. Algumas pesquisas de Engenharia de Software que empregaram métodos qualitativos são as de [Salinger et al. 2013], [Hoda et al. 2012] e [McLeod et al. 2011].

Alguns métodos de análise de dados qualitativos baseados em texto como a Análise de Conteúdo, a Análise Temática e a Teoria Fundamentada em Dados, utilizam um procedimento denominado de codificação. A codificação de dados qualitativos, segundo [Gibbs 2009], é a forma como o pesquisador define de que tratam os dados em análise, através da aplicação de nomes a passagens de texto, de sua categorização, de forma a estabelecer uma estrutura temática, que possibilite possíveis interpretações do seu conteúdo. De acordo com [Elliott 2018], a codificação é quase um processo universal em pesquisa qualitativa, sendo um aspecto fundamental do processo analítico e a maneira pela qual pesquisadores desmembram seus dados para produzir algo novo.

A codificação é uma tarefa analítica e, por isso, muito dependente da intervenção humana. Assim, demanda tempo para ser realizada. É comum que esse procedimento seja realizado através de Softwares para Análise de Dados Qualitativos (SADQs). No entanto, poucos softwares permitem a colaboração de vários pesquisadores em um mesmo projeto e, mais importante, essa forma de colaboração pode não ser tão eficiente, uma vez que permite que todos os pesquisadores tenham acesso ao mesmo conjunto de dados, sem atribuição de responsabilidade de codificação.

Uma proposta de solução para esse problema partiria do uso do conceito de *crowdsourcing*, onde membros de uma comunidade podem realizar de forma colaborativa uma determinada tarefa, neste caso, o procedimento de codificação. Em um processo de codificação colaborativa, gasta-se menos tempo, pois várias pessoas (codificadores) podem codificar dados textuais ao mesmo tempo. Além disso, se implementar mecanismos de verificação dos códigos pelos pesquisadores, o resultado pode ser mais confiável.

O objetivo deste artigo é apresentar uma proposta de ferramenta para criação colaborativa de códigos a partir de dados textuais. A ferramenta permite criar grupos de dados textuais que podem ser atribuídos a diferentes grupos de usuários. Dessa forma, um mesmo conjunto de dados pode ser codificado por várias pessoas. Além disso, a ferramenta permite que especialistas verifiquem a qualidade da codificação. A avaliação da ferramenta foi realizada através de um estudo de caso real, onde a codificação colaborativa de dados textuais foi executada com celeridade e qualidade.

O artigo está organizado da seguinte forma: na Seção 2 é apresentada a fundamentação teórica com a descrição de alguns métodos qualitativos e funcionalidades encontradas em softwares de análise de dados qualitativos; na Seção 3 as relacionadas a esta proposta; na Seção 4 é apresentada a ferramenta; na Seção 5 o estudo de caso envolvendo o uso da ferramenta e, por fim, as considerações finais.

2. Fundamentação Teórica

2.1. Métodos de Pesquisa Qualitativa

Dentre os métodos qualitativos, destacam-se alguns onde o procedimento de codificação é mais utilizado, são eles: Análise de Conteúdo, Análise Temática e Teoria Fundamentada em Dados (*Grounded Theory*).

A Análise de Conteúdo consiste em uma abordagem sistemática de codificação e categorização utilizada para explorar grandes quantidades de informações textuais, para determinar as tendências e padrões de palavras utilizadas, a sua frequência, as suas relações e as estruturas e discursos de comunicação [Mayring 2004]. Aceita-se que o seu foco seja qualificar as vivências do sujeito, bem como suas percepções sobre determinado objeto e seus fenômenos [Bardin 1977].

A Análise Temática (AT) é um método para identificar, analisar e relatar padrões (temas) dentro dos dados. Ela organiza e descreve o conjunto de dados de forma detalhada [Boyatzis 1998]. Segundo [Braun and Clarke 2006], a AT é muito semelhante à Análise de Conteúdo. No entanto, difere no fato de que os temas geralmente não são quantificados. Além disso, a AT também se concentra em encontrar temas em vez de criar categorias em que a unidade de dados é mais do que uma palavra ou frase, como na Análise de Conteúdo.

Na Teoria Fundamentada em Dados os dados são revisados usando uma análise comparativa constante para marcar códigos e agrupar em conceitos para determinar temas. A análise comparativa constante (categorizar, comparar e conceituar) reduz os dados para identificar conceitos e desenvolver uma teoria fundamentada [Strauss and Corbin 2008]. Esta metodologia serve para que o pesquisador consiga gerar explicações (e.g teorias) de um processo, ação ou uma interação moldada pelas visões de um grande número de participantes [Creswell 2014].

2.2. Software de Análise de Dados Qualitativos

Softwares de Análise de Dados Qualitativos (SADQs) possuem diferentes recursos, cuja aplicabilidade pode variar de acordo com os objetivos e a natureza da pesquisa. Assim, cabe ao pesquisador avaliar a utilidade desse tipo de software em sua pesquisa e quais ferramentas utilizar. As principais funcionalidades encontradas em um SADQ são:

- Pesquisa de conteúdo: permite coletar dados qualitativos, extraindo conteúdo de arquivos de vídeo, áudio, documentos de texto, gráficos e outros.
- Visualização e relatórios de dados: permite visualizar todas as formas de dados eletrônicos, incluindo entrevistas, pesquisas, vídeos com imagens e dados bibliográficos;
- Armazenamento e codificação: permite aos analistas de dados e outros usuários de software executar diferentes formas de codificação, como codificação de palavras-chave e texto. Permite codificar sistematicamente dados em diferentes formatos e categorias;
- Ligação de dados: permite que os usuários formem clusters, redes ou categorias de dados;

De acordo com [Moreira 2007], não existe consenso entre os pesquisadores quanto às vantagens do uso desses softwares, sobretudo na pesquisa qualitativa. Assim, o autor aponta uma série de argumentos contra e a favor ao uso deles. Grande parte dos argumentos a favor dizem respeito à facilidade e rapidez proporcionada pelo uso do computador e seus recursos. Já os argumentos contra estão relacionados aos riscos da mecanização de um processo interpretativo.

Os SADQ fazem, a partir de um computador, o que os pesquisadores vêm fazendo manualmente há décadas: a armazenagem, o gerenciamento e a recuperação de dados. Como fio condutor dessas funções, está o processo de codificação que consiste na

designação de códigos para pequenos trechos do texto. Esses códigos, por sua vez, podem ser sobrepostos, permitindo que vários trechos de texto sejam recuperados a partir de um mesmo código.

Contudo, essa codificação não é executada de forma autônoma pelo software, mas depende da indicação do pesquisador. Assim, embora o processo de análise seja mecanicamente facilitado e acelerado pelo software, a codificação é resultado do raciocínio e da versatilidade do pesquisador. E, considerando que durante o processo de análise o pesquisador passa a ter uma visão mais geral sobre os dados, esses softwares permitem a revisão dos códigos, combinando-os ou dividindo-os [Moreira 2007].

3. Trabalhos Relacionados

Segundo [Gibbs 2009], três SADQs parecem ser os mais utilizados pelos pesquisadores, são eles: Atlas.ti, MAXqda e NVivo. Contudo, essas ferramentas não são gratuitas e não permitem que a codificação seja realizada de forma colaborativa. Nesta pesquisa, entende-se o requisito de colaboração de uma forma mais específica, como a capacidade de permitir que o conteúdo textual seja dividido em vários conjuntos de dados, que sejam criados vários grupos de codificadores e que cada grupo possa acessar um ou mais conjuntos de dados.

Uma pesquisa por SADQs gratuitos, identificou a existência de 14 softwares para uso em diversas metodologias de pesquisa qualitativa. Desses a maioria são aplicações desktop, que não permitem qualquer tipo de atividade colaborativa. Na Tabela 1, são exibidos os únicos SADQs gratuitos e disponíveis para uso na plataforma web, sendo este último critério um indicador da possibilidade de atividades colaborativas.

Dos três softwares encontrados, QCAmap e Computer Assisted Text Markup and Analysis (CATMA) não permitem qualquer colaboração. Já o software Coding Analysis Toolkit (CAT) permite criar subcontas e, a partir de um titular de conta principal, é possível convidar colaboradores para o processo de codificação. São 2 tipos de conta: o usuário com tipo de conta especializada têm permissão para acessar, fazer upload e bloquear conjuntos de dados; o usuário com tipo de conta regular só pode acessar conjuntos de dados quando recebe permissão de alguém com conta especializada, sendo essa conta a indicada para codificadores.

No teste de uso do CAT percebeu-se que as importações de trechos textuais não são fáceis de serem realizadas, mesmo seguindo-se o passo a passo do tutorial. Além disso, foi percebível um visual antiquado, pouco amigável, algo que pode desestimular os usuários. Por fim, recentemente foi relevado aos usuários a descontinuidade deste software.

A ferramenta proposta neste trabalho, soluciona as limitações destacadas no CAT no que tange, por exemplo, à facilidade de importação de trechos textuais e tem potencial para se tornar uma alternativa viável de uso em pesquisas que demandem um processo de codificação colaborativa.

Softwares gratuitos na web	Colaborativa	Site
QCAmap	Não	www.qcamap.org
CATMA	Não	catma.de
CAT	Sim	cat.texifter.com

Tabela 1. Ferramentas Gratuitas de Análise Qualitativa de Dados

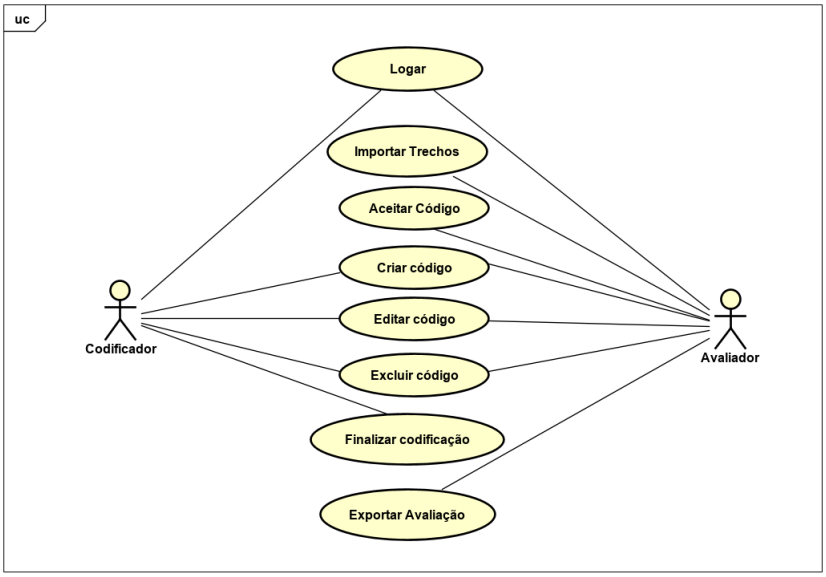
4. Metodologia de Desenvolvimento da Ferramenta

O objetivo da ferramenta ***Open Coding Tool*** é apoiar o processo de codificação de dados textuais de forma colaborativa, permitindo que códigos sejam criados por uma comunidade de codificadores e posteriormente sejam avaliados e chancelados por pesquisadores, de forma online e gratuita.

Para o desenvolvimento da ferramenta, optou-se por um processo ágil de software, entregando em intervalos de tempo curto as funcionalidades solicitadas. A coleta de requisitos foi realizada com uma pesquisadora da área de Engenharia de Software que utiliza métodos qualitativos e seus orientandos; os requisitos foram analisados, modelos foram criados e um documento de especificação de requisitos foi desenvolvido.

A Figura 1 exibe o diagrama de caso de uso com as principais funcionalidades da ferramenta. Existem dois tipos de usuários: o codificador e o avaliador/pesquisador. O codificador tem permissão para analisar os trechos textuais, gerenciar a criação de códigos e finalizar o processo. O avaliador pode avaliar os códigos criados pelos codificadores, podendo aceitar, modificar ou rejeitar os códigos criados anteriormente, além de criar novos códigos, caso os disponíveis não sejam adequados.

Figura 1. Diagrama de Caso de Uso



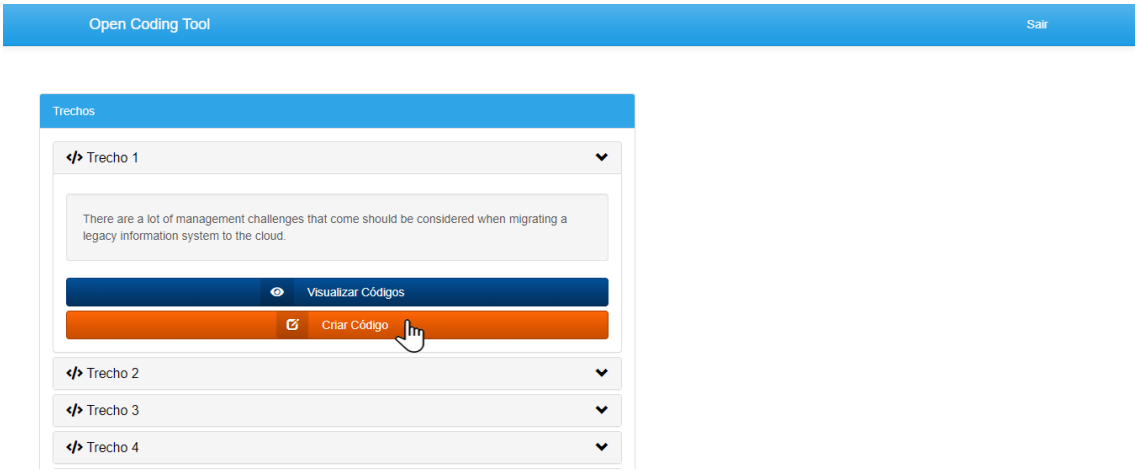
A implementação do software foi feita com a linguagem de programação open source PHP com apoio do framework ZendFramework 1.12.3 para criar o backend. Para o frontend foram utilizados HTML5, CSS3, Javascript, com o apoio do framework Bootstrap3. O gerenciador de banco de dados foi o MySQL.

Antes de ser colocada em produção a ferramenta foi testada junto a grupo de codificadores voluntários. As percepções foram coletadas e serviram para o seu aprimoramento. O deployment da versão final foi feito em um servidor institucional [1].

4.1. Funcionalidades da Ferramenta

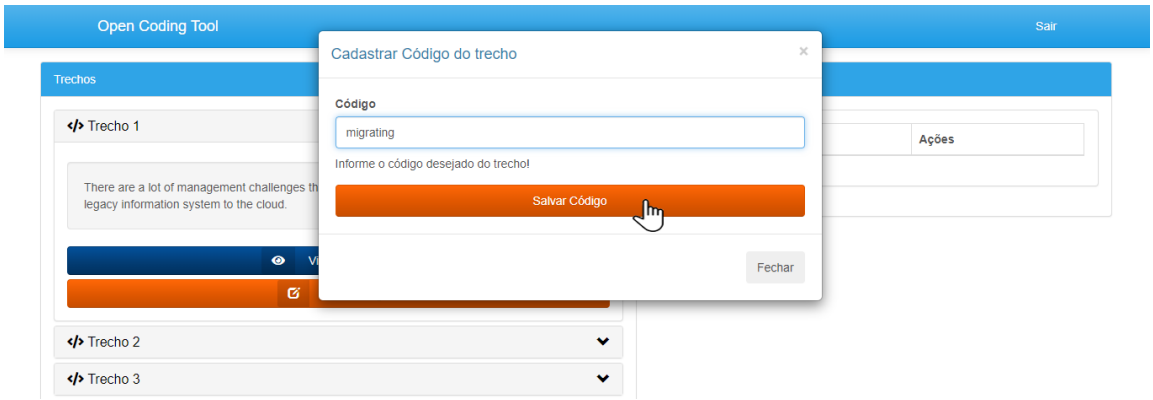
Cada codificador deve possuir um *login* e senha para acessar a ferramenta e visualizar os trechos textuais alocados para a sua análise, como exposto na Figura 2. Após análise do trecho, o codificador pode criar um ou mais códigos referentes à sua visão analítica, de forma individual.

Figura 2. Perfil de Codificador: Análise dos trechos textuais



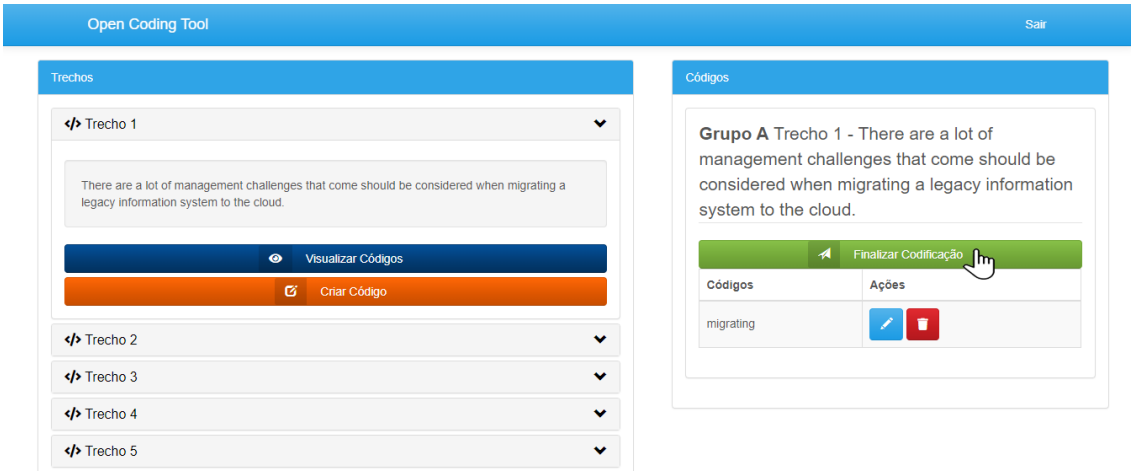
Como exibido na Figura 3 podem ser criados vários códigos relacionados ao trecho textual. Cada código criado poderá ser editado e até mesmo excluído antes de ser salvo conforme explicitado na Figura 4.

Figura 3. Perfil de Codificador: Criação de códigos



[1] Ferramenta Open Coding Tool: `https://opencodingtool.com.br`

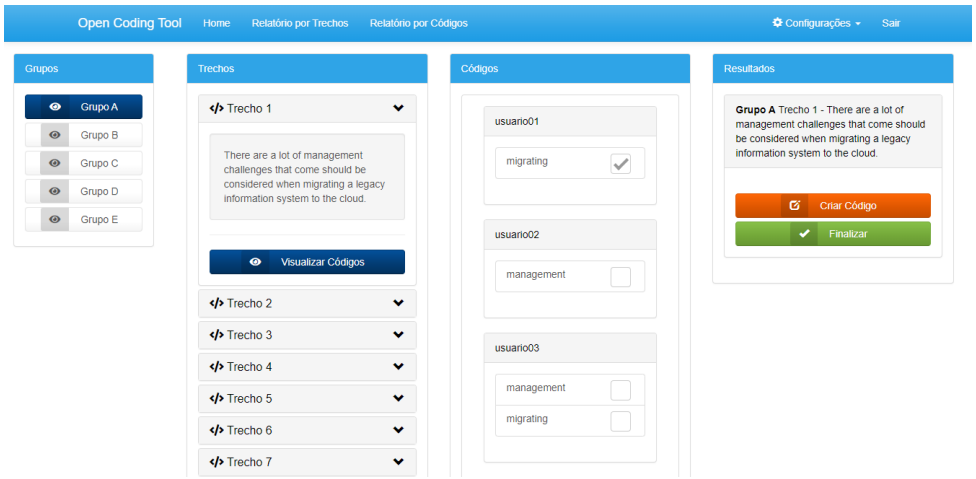
Figura 4. Perfil de Codificador: Salvar código



Após a finalização da codificação, não é mais permitido ao codificador adicionar ou modificar códigos referentes ao trecho textual que estava sendo analisado.

Os avaliadores também devem possuir *login* e senha para acessar a ferramenta. Com esse perfil é possível visualizar todos códigos criados pelos codificadores em todos os trechos, como mostra a Figura 5. O avaliador possui a opção de aceitar (selecionando o *check box*) o(s) código(s) mais apropriados. Também é possível criar códigos de acordo com suas análises, bem como editar códigos já criados pelos codificadores.

Figura 5. Perfil de avaliador: Aceitar ou criar códigos



5. Estudo de caso

A ferramenta *Open Coding Tool* foi desenvolvida para dar apoio ao procedimento de codificação aberta da Teoria Fundamentada em Dados, método utilizado na pesquisa de [Chervenski and Bordin 2020] sobre entendimento de sistemas legados a partir da literatura científica. A referida pesquisa analisou trechos textuais oriundos da literatura com o objetivo que elucidar os elementos que causam ou contribuem para um sistema se tornar legado, as consequências e as possíveis estratégias de evolução de tais sistemas, bem como suas relações, contribuindo para a criação de uma teoria.

No trabalho de [Chervenski and Bordin 2020], a coleta de trechos textuais foi realizada através de um mapeamento sistemático na literatura, seguindo os passos bem definidos por [Petersen et al. 2008]. Foram utilizadas 3 bases digitais (ACM Digital library, IEEE Xplore e Science Direct), após a criação e utilização de *strings* de busca foi possível obter um número de 87 estudos. Após a leitura dos estudos, foi possível identificar 111 trechos textuais que foram armazenados em planilhas no *Google Drive* e importadas para a ferramenta *Open Code Tool* através de arquivos CSV.

O processo colaborativo de codificação foi pensado como alternativa, em razão da necessidade de se obter um consenso confiável de grupos de especialistas em um espaço de tempo reduzido, pois o conjunto de dados textuais analisados era volumoso.

A codificação colaborativa na ferramenta *Open Coding Tool* foi realizada por 14 participantes, todos alunos dos cursos de Engenharia de Software e Ciência da Computação, e por 2 avaliadores que possuiam o papel de validar os códigos criados pelos participantes. A escolha dos participantes foi feita levando em consideração a experiência acadêmica de cada aluno. Os alunos do curso de Engenharia de Software já deveriam ter cursado a disciplina de Evolução de Software e os alunos do curso de Ciência da Computação a disciplina de Engenharia de Software 2, onde tópico Sistemas Legados é abordado.

Os participantes foram separados em 5 grupos, sendo esse número definido em função da quantidade e do tamanho dos trechos textuais. Preferencialmente cada grupo deveria ser composto por 3 participantes, de forma que as decisões de codificação não ficassem polarizadas e que cada grupo tivesse condições de codificar entre 25 a 30 trechos. Para cada grupo foi alocado uma quantidade de trechos textuais diferentes, sem sobreposições. Como alguns trechos eram bem maiores, optou-se por alocar uma quantidade menor aos grupos D e E. Alguns exemplos de trechos textuais extraídos da literatura estão destacados na Tabela 2. Na Tabela 3 é possível visualizar os grupos, números de participantes e quantidade de trechos alocados.

Tabela 2. Exemplos de trechos textuais extraídos da literatura

Exemplo de trecho textual
Older legacy software systems are frequently replaced as they become obsolete.
Recent estimates confirm at least 180 billion lines of legacy, smelly code are target of software refactoring.
legacy systems usually lack complete and updated engineering documents.
In a legacy system, component-aging is unavoidable.
legacy systems are operated across decades.

Tabela 3. Relação de grupos para codificação colaborativa.

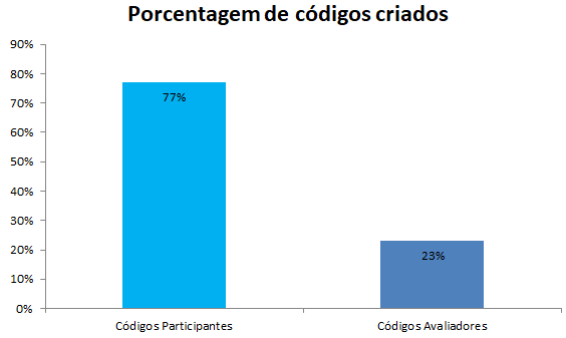
Grupo	Número de participantes	Quantidade de Trechos
A	3	31
B	3	30
C	2	30
D	3	15
E	3	14

Cabe destacar que a construção dessa aplicação *web* possibilitou a todos os participantes a oportunidade de acessarem a ferramenta e realizarem o processo de criação de códigos de qualquer lugar e a qualquer momento que desejassem, permitindo que

o trabalho fosse interrompido e continuado de onde se havia parado. Ao término das análises textuais e criações de códigos, cada participante finalizava o processo. Assim, os códigos criados pelos participantes eram salvos e foram analisados pelos avaliadores/pesquisadores.

Ao final do processo, obteve-se um total de 237 códigos avaliados, desse total cerca de 23% (54 códigos) foram criados pelos avaliadores e 77% (183 códigos) foram criados pelos participantes e aceitos pelos avaliadores, como mostra a Figura 6. Esses números evidenciam a forte contribuição da codificação colaborativa e da ferramenta *Open Coding Tool* para o desenvolvimento desse procedimento de codificação.

Figura 6. Resultados da codificação colaborativa



6. Considerações Finais

Este artigo apresentou uma ferramenta de codificação colaborativa que permite a participação de muitas pessoas (*crowd*) em procedimento de codificação de dados textuais. A ferramenta também garante que os códigos criados sejam validados por um grupo de pesquisadores mais experientes.

A ferramenta traz contribuições para a comunidade acadêmica interessada em pesquisas qualitativas, incluindo a Engenharia de Software, onde estudos dessa natureza estão crescendo, porque é gratuita, pode ser acessada pela web, permite uma configuração que amplia o conceito de colaboração existente em alguns softwares de análise qualitativa e garante que os resultados surjam em período de tempo reduzido com a qualidade necessária.

A ferramenta está operacional, com as funcionalidades já apresentadas, no entanto, testes adicionais, ajustes e novas funcionalidades devem ser realizados, de forma a torná-la disponível para a ampla utilização pela comunidade acadêmica. Como trabalhos futuros de desenvolvimento, elenca-se a disponibilização da ferramenta em inglês, a possibilidade de inserção de vários projetos de análise de dados qualitativos, assim como a criação de agrupamentos de códigos em categorias e o relacionamento entre categorias, permitindo que outros procedimentos presentes em métodos qualitativos, como a categorização no método de Análise de Conteúdo e a codificação axial da Teoria Fundamentada em Dados sejam contemplados. Posteriormente pretende-se avançar com a implementação da coleta e extração semi-automática de trechos de dados textuais, que servirão de insumo aos procedimentos de análise de dados qualitativos.

Referências

Bardin, L. (1977). Análise de conteúdo. *Lisboa: edições*, 70:225.

Boyatzis, R. E. (1998). *Transforming qualitative information: Thematic analysis and code development*. sage.

Braun, V. and Clarke, V. (2006). Using thematic analysis in psychology. *Qualitative research in psychology*, 3(2):77–101.

Chervenski, A. S. and Bordin, A. S. (2020). Understanding Legacy Systems in the Light of Grounded Theory. *34th Brazilian Symposium on Software Engineering (SBES '20)*.

Creswell, J. W. (2014). *Investigação Qualitativa e Projeto de Pesquisa-: Escolhendo entre Cinco Abordagens*. Penso Editora.

Denzin, N. K. and Lincoln, Y. S. (2005). The discipline and practice of qualitative research introduction. *The landscape of qualitative research*, pages 1–43.

Dybå, T., Prikladnicki, R., Rönkkö, K., Seaman, C., and Sillito, J. (2011). Qualitative research in software engineering. *Empirical Software Engineering*, 16(4):425–429.

Elliott, V. (2018). Thinking about the coding process in qualitative data analysis. *The Qualitative Report*, 23(11):2850–2861.

Gibbs, G. (2009). *Análise de dados qualitativos: coleção pesquisa qualitativa*. Bookman Editora.

Hoda, R., Noble, J., and Marshall, S. (2012). Developing a grounded theory to explain the practices of self-organizing agile teams. *Empirical Software Engineering*, 17(6):609–639.

Mayring, P. (2004). Qualitative content analysis. *A companion to qualitative research*, 1(2004):159–176.

McLeod, L., MacDonell, S. G., and Doolin, B. (2011). Qualitative research on software development: a longitudinal case study methodology. *Empirical software engineering*, 16(4):430–459.

Moreira, D. A. (2007). O uso de programas de computador na análise qualitativa: oportunidades, vantagens e desvantagens. *Revista de Negócios*, 12(2):56–58.

Petersen, K., Feldt, R., Mujtaba, S., and Mattsson, M. (2008). Systematic mapping studies in software engineering. *EASE'08 Proceedings of the 12th international conference on Evaluation and Assessment in Software Engineering*, pages 68–77.

Salinger, S., Zieris, F., and Prechelt, L. (2013). Liberating pair programming research from the oppressive driver/observer regime. In *2013 35th International Conference on Software Engineering (ICSE)*, pages 1201–1204. IEEE.

Strauss, A. and Corbin, J. (2008). *Pesquisa qualitativa: técnicas e procedimentos para o desenvolvimento de teoria fundamentada.* 2ª ed. Porto Alegre (RS): Artmed.

Investigando a Integração de Ferramentas com OSLC Através do Eclipse Lyo

Bruno Marcelo S. Ferreira[1], **Fábio P. Basso**[1], **Elder Rodrigues**[1], **Maicon Bernardino**[1], **Rafael Z. Frantz**[2]

[1]Universidade Federal do Pampa (UNIPAMPA), PPGES
Alegrete - RS

[2]Universidade Regional do Noroeste do Estado do Rio Grande do Sul (UNIJUÍ)
Ijuí - RS

brunoferreira.aluno@unipampa.edu.br

Abstract. *The software industry invests in modern tools throughout the software development lifecycle. However, there are challenges to achieving an integrated end-to-end environment, such as dealing with multiple tool configurations and establishing project data interoperation in real time. To mitigate these challenges, many approaches have been proposed for integrating tools. This paper presents an exploratory study on a technology for modeling integration solutions and the respective generation of integration configurations in Open Services for Lifecycle Collaboration (OSLC) standard. The results are interesting and suggest that Eclipse Lyo provides some benefits to software engineers when configuring integration solutions in Software Engineering application lifecycles.*

Resumo. *A indústria de software investe em ferramentas modernas ao longo de todo o ciclo de vida de desenvolvimento de software. No entanto, existem desafios para alcançar um ambiente integrado de ponta a ponta, como por exemplo lidar com múltiplas configurações de ferramentas e estabelecer o compartilhamento de dados do projeto em tempo real. Para mitigar esses desafios, muitas abordagens foram propostas para a integração de ferramentas. Este artigo apresenta um estudo exploratório sobre uma tecnologia para modelagem de soluções de integração e respectiva geração de configurações de integração no padrão Open Services for Lifecycle Collaboration (OSLC). Os resultados são interessantes e sugerem que Eclipse Lyo provê alguns benefícios aos engenheiros de software na configuração de soluções de integração em ciclos de vida de aplicações de Engenharia de Software.*

1. Introdução

Ferramentas de software são utilizadas para apoiar profissionais na execução de atividades ao longo do ciclo de vida do software. Essas ferramentas possuem diferentes funcionalidades, configurações, fabricantes e são desenvolvidas sem o suporte nativo para serem integradas com outras ferramentas. Além disso, manipulam artefatos (requisitos de software, modelos, código-fonte, casos de teste, etc.) produzidos em diferentes fases do desenvolvimento. Este cenário contribui para que ambientes integrados de ponta a ponta de forma totalmente automatizada sejam raros na indústria [Wicks and Dewar 2007].

Organizações de software obtêm novas ferramentas ou desenvolvem suas soluções de software ao longo dos anos, resultando em um ecossistema de software heterogêneo [Messerschmitt 2005]. Isso demanda que as organizações adotem abordagens para melhoria dos processos de software e suporte ferramental que permitam a automação do processo de desenvolvimento através da integração de ferramentas.

Entre as alternativas está o *Application Lifecycle Management* (ALM), que adota a ideia de integração ponta a ponta, evitando silos de informações devido a características como rastreabilidade, visibilidade e automação de processos [Schwaber et al. 2006]. Para reduzir os problemas relacionados à integração de artefatos de software ao longo de várias fases do ALM, padrões de representação para modelos, metamodelos e transformações têm sido propostos na literatura [Kleppe et al. 2003]. Além disso, para mitigar a complexidade desse ambientes, várias ferramentas que auxiliam nas tarefas de ES foram propostas [Ebert 2013]. Essas ferramentas ajudam os desenvolvedores a aproveitar experiências anteriores ao especificar um novo software, enquanto ferramentas baseadas em modelos ajudam a capturar, organizar e armazenar a maioria das informações trocadas como *assets* reutilizáveis.

Para ajudar na integração de ferramentas usadas na produção de software, muitas abordagens foram propostas interoperando dados entre serviços [Biehl et al. 2014]. As abordagens foco destes trabalhos caracterizam-se por minimizar os problemas relacionados ao processo de integração. Elas são adotadas nos contextos de Engenharia de Software (ES), tipicamente implementadas por meio de uma arquitetura comum para serviços [Zhang and Møller-Pedersen 2014], o que demanda um protocolo de comunicação comum entre diferentes ferramentas.

Nesse contexto, uma emergente especificação industrial caracterizada como arquitetura comum para ferramentas de ES foi proposta: *Open Services for Lifecycle Collaboration* (OSLC) [OSLC 2020]. O OSLC é um padrão aberto para interoperabilidade de ferramentas de software, que define um modelo de representação comum para os artefatos produzidos ao longo do ciclo de vida do software, bem como métodos que permitem ferramentas compartilhar dados entre si.

Com base nisso, este trabalho, de caráter exploratório, possui o objetivo de identificar os aspectos envolvidos na modelagem de integração por meio da ferramenta Eclipse Lyo e da geração de código-fonte com base em artefatos em conformidade com o padrão OSLC. Durante a realização deste trabalho buscou-se analisar parte de um cenário de um setor de desenvolvimento de software de uma organização governamental. Este cenário composto por ferramentas de código-fonte aberto, foi parcialmente implementado, resultando em algumas contribuições como segue: (i) identificada a carência no estado da arte de um material que sintetiza ou organiza atividades para a geração de soluções de integração em OSLC, derivou-se dessa experiência um tutorial para desenvolvimento de integradores; e (ii) comprovação na prática de que o padrão OSLC e o Eclipse Lyo funcionam na integração dos artefatos do cenário explorado.

Este trabalho é organizado como segue: Na seção 2 aborda tecnologias investigadas durante a realização deste trabalho, como os conceitos do padrão OSLC e do Eclipse Lyo. A Seção 3 caracteriza o estudo conduzido no Eclipse Lyo e a Seção 4 as alternativas ao Eclpse Lyo para o desenvolvimento de adaptadores OSLC. Por fim, A Seção 5 discute

os resultados deste estudo.

2. Tecnologias Investigadas

Esta seção apresenta os conceitos das tecnologias investigadas durante a execução deste trabalho.

2.1. Open Services for Lifecycle Collaboration

Open Services for Lifecycle Collaboration (OSLC) é um padrão aberto para integração de ferramentas de software. O OSLC define uma forma de representação comum para artefatos, bem como métodos para o compartilhamento de dados em todos os domínios do projeto. As especificações OSLC consistem em regras para criar, atualizar, recuperar e ligar assets estruturados nos formatos RDF/XML e JSON. Existem também os domínios OSLC que estendem a especificação principal e definem como representar artefatos em domínios como gerenciamento de requisitos, gerenciamento de qualidade, gerenciamento de configuração e mudanças, etc.

O OSLC é baseado nos princípios dos dados ligados e segue as regras definidas por Tim Berners-Lee [Linked Data 2006] para ligar dados na web. Nesse sentido, os relacionamentos entre os artefatos em uma cadeia de ferramentas podem ser estabelecidos sem a duplicação de dados por meio de *links*. Esses *links* são mantidos em uma estrutura chamada tripla que consiste em dados de sujeito-predicado-objeto que descrevem o relacionamento entre artefatos, partes interessadas e atividades. Por exemplo, em um ambiente formado pelas ferramentas de gerenciamento de requisitos (Ferramenta A) e gerenciamento de testes (Ferramenta B), um requisito na Ferramenta A pode ser validado por um caso de teste na Ferramenta B através do OSLC.

Em uma cadeia de ferramentas OSLC, uma aplicação pode ser classificada como Provedora ou Consumidora. Provedores destinam-se a armazenar e fornecer dados, permitindo aos consumidores acesso fácil para navegar, criar e recuperar dados [Aichernig et al. 2014].

Existem três abordagens para prover suporte OSLC em ferramentas de software: Abordagem nativa, *plugin* e adaptador. A abordagem de suporte nativo é recomendada para desenvolvedores de ferramentas. As abordagens *plugin* e adaptador são semelhantes, entretanto a construção de *plugins* são recomendadas apenas nos casos em que há o conhecimento das tecnologias que envolvem a construção da ferramenta, como linguagem de programação e arquitetura. Em outros cenários, a abordagem recomendada é a construção de adaptadores OSLC. Nesse contexto, surge a necessidade de abordagens para se implementarem adaptadores OSLC, como por exemplo Desenvolvimento Dirigido por Modelos (MDD) [Fowler and Parsons 2011].

2.2. Eclipse Lyo

O Eclipse Lyo é um projeto aberto que visa ajudar a comunidade interessada em integrações a adotar as especificações OSLC em suas ferramentas [El-khoury 2016]. O Lyo possibilita o desenvolvimento de novas soluções compatíveis com o OSLC por meio da abordagem de MDD, que permite engenheiros a trabalhar com um nível maior de abstração através de Linguagens Específicas de Domínio (DSLs).

O projeto consiste nos seguintes componentes: OSLC4J SDK, que é um kit de desenvolvimento java para interfaces de ferramentas provedoras e consumidoras OSLC e o Lyo Designer, que é um gerador de código-fonte baseado em modelos, permitindo a representação gráfica dos modelos de integrações. Entretanto, o código-fonte gerado possui apenas um conjunto de métodos que permitem a comunicação entre as ferramentas e para acessar os dados armazenados internamente é necessário implementá-los manualmente.

Existem três perspectivas para a representação de modelos no Lyo Designer: Perspectiva Especificação de Domínios, Perspectiva Cadeia de Ferramentas e Perspectiva Interface do Adaptador. Cada perspectiva permite representar um tipo de diagrama, como visto a seguir:

1. **Perspectiva Especificação de Domínios -** Trata sobre a representação dos domínios OSLC. É possível adicionar múltiplos domínios. Cada domínio é composto por *Resources* (artefatos) e *Resources Properties* (valores permitidos, cardinalidade e opcionalidade).
2. **Perspectiva Cadeia de Ferramentas -** Refere-se a estrutura da cadeia de ferramentas e artefatos que serão compartilhados entre as aplicações, definindo quais as ferramentas que serão consumidoras e provedoras.
3. **Perspectiva Interface do Adaptador -** Representa a estrutura dos adaptadores de cada ferramenta representada na perspectiva Cadeia de Ferramentas. Essa estrutura segue a especificação principal do OSLC, sendo a base para a geração de códigos.

Após realizar as três perspectivas de modelagem, as abordagens baseadas em MDD permitem gerar código para adaptadores OSLC, promovendo assim a integração automatizada de um modelo independente para uma representação específica da plataforma.

3. Caracterização do Estudo Exploratório

Esta seção detalha a implementação de adaptadores OSLC por meio da geração de código-fonte baseada em modelos no Eclipse Lyo por meio de um estudo exploratório. Este tipo de estudo não possui uma análise conduzida conforme os protocolos de estudo de caso de outras naturezas como: descritivo, explanatório, e melhoria. Estudos exploratórios possuem a característica de relatar uma experiência, sendo o primeiro tipo de estudo conduzido para caracterizar a aplicabilidade de determinada solução [Runeson and Höst 2008]. A Figura 1 mostra a cadeia de ferramentas do nosso estudo exploratório, composta por três ferramentas de domínios diferentes, sendo elas Excel para o gerenciamento de requisitos, Redmine para o gerenciamento de projetos e Testlink para gerenciamento de testes.

A ferramenta Excel mantém requisitos de software e deseja-se ligar através das propriedades dos dados ligados do OSLC às tarefas gerenciadas pelo Redmine e aos casos de testes do Testlink. Com base nisso, o adaptador OSLC do Excel possui duas interfaces consumidoras para tarefas e casos de testes respectivamente e uma interface que provê requisitos de software. Além disso, as ferramentas Redmine e Testlink possuem interfaces provedoras para o compartilhamento de dados.

Embora nosso cenário tenha definido a estrutura da cadeia de ferramentas do nosso estudo, o primeiro passo para gerar os códigos-fonte dos adaptadores é a representação

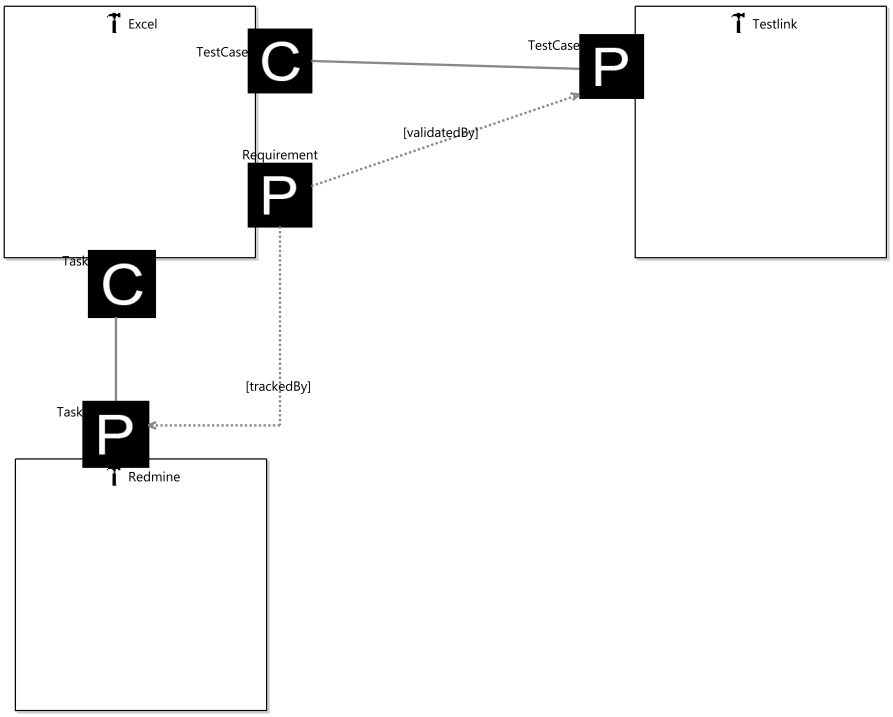


Figura 1. Estrutura da Cadeia de Ferramentas

gráfica dos domínios OSLC. Como mostra a Figura 2, exploramos três domínios OSLC: *Requirements Management (RM)* [1], *Change Management (CM)* [2] e *Quality Management (QM)* [3]. Cada um desses domínios possuem propriedades e *resources* para a representação dos artefatos mantidos pelas ferramentas e seus relacionamentos. Por exemplo, o domínio Requirement Management possui o *resource* Requirement com as propriedades que o descrevem como identificador, data de criação e modificação. Além disso, nessa representação são estabelecidos os relacionamentos entre os artefatos a partir de propriedades como a *validateBy* em que é possível ligar um caso de teste que valida determinado requisito. Além dos domínios OSLC, também exploramos outros domínios auxiliares, que tem objetivo de padronizar informações da web, como o FOAF [4] e Dublin Core [5].

A partir de serviços OSLC, é possível recuperar, editar, atualizar e excluir artefatos de software gerados pela cadeia de ferramentas. Essas funcionalidades são modeladas na perspectiva do adaptador. A Figura 3 mostra a interface do adaptador da ferramenta Redmine. A estrutura desse adaptador segue as especificações da Especificação principal do OSLC. Em adaptadores OSLC, todos os serviços são acessados através de *Uniformd Resource Identifier* (URIs). Ao acessar o adaptador é disponibilzado um catálogo de serviços (*Service Provider Catalog*) que mantém URIs para todos os provedores de serviços (*Service Providers*), que representam os projetos cadastrados no Redmine. Cada provedor de

[1]http://docs.oasis-open.org/oslc-domains/oslc-rm/v2.1/oslc-rm-v2.1-part2-requirements-management-vocab.html

[2]http://docs.oasis-open.org/oslc-domains/cm/v3.0/cs02/part2-change-mgt-vocab/cm-v3.0-cs02-part2-change-mgt-vocab.html

[3]https://docs.oasis-open-projects.org/oslc-op/qm/v2.1/psd02/quality-management-vocab.html

[4]http://xmlns.com/foaf/spec/

[5]https://www.dublincore.org/specifications/dublin-core/dcmi-terms/

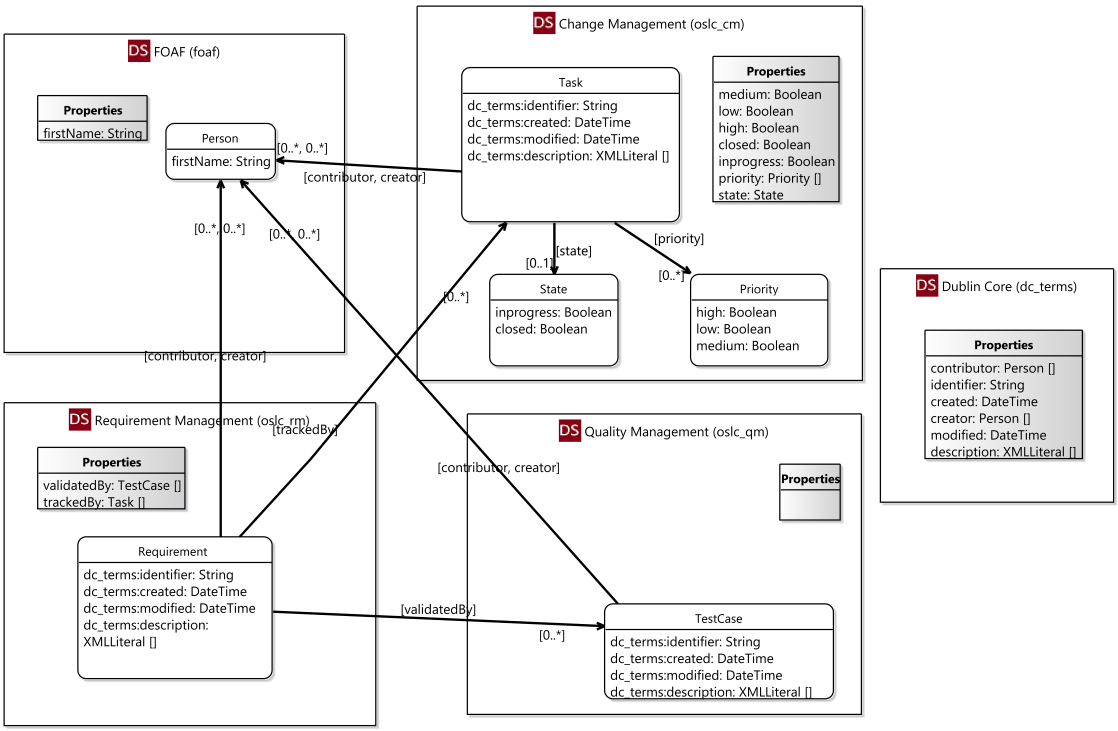


Figura 2. Representação Gráfica dos Domínios OSLC no Eclipse Lyo

serviço disponibiliza URIs para acessar o serviços (*Services*), que nesse cenário são as tarefas de cada um dos projetos. Ao final do fluxo, cada é possível acessar cada tarefa e manipulá-la através das funções Restful que foram modeladas na perspectiva da interface do adaptador.

O Lyo gera apenas os esqueletos de classes java, sendo necessário implementar manualmente a lógica dos adaptadores. Para que isso seja possível, as ferramentas devem possuir alguma interface que possibilite manipular seus dados, como uma API RESTful. Em nosso estudo, usamos três APIs RESTful, uma para cada ferramenta envolvida no processo. Graças ao componente OSLC4J do Lyo, todos os objetos tornam-se possível de serem representados nos formatos RDF/XML e JSON, possibilitando a troca de dados entre as ferramentas após a transformação dos seus dados em objetos java para o padrão OSLC.

4. Trabalhos Relacionados

Integração de ferramentas em ambientes de ES trata sobre como ferramentas com características heterogêneas podem ser compatíveis em diferentes aspectos. Esses aspectos incluem os formatos dos dados, convenções de interface do usuário, funcionalidades da aplicação e outros aspectos da construção da ferramenta [Thomas and Nejmeh 1992].

Ainda que o maior desafio seja prover um ambiente integrado de forma automatizada que compreenda todo o ciclo de vida do software [Brown and Penedo 1992], existem diversas abordagens para integrar ferramentas de software. Cada uma dessas abordagens envolve uma série de características que devem ser levadas em consideração durante sua escolha [Hohpe 2003].

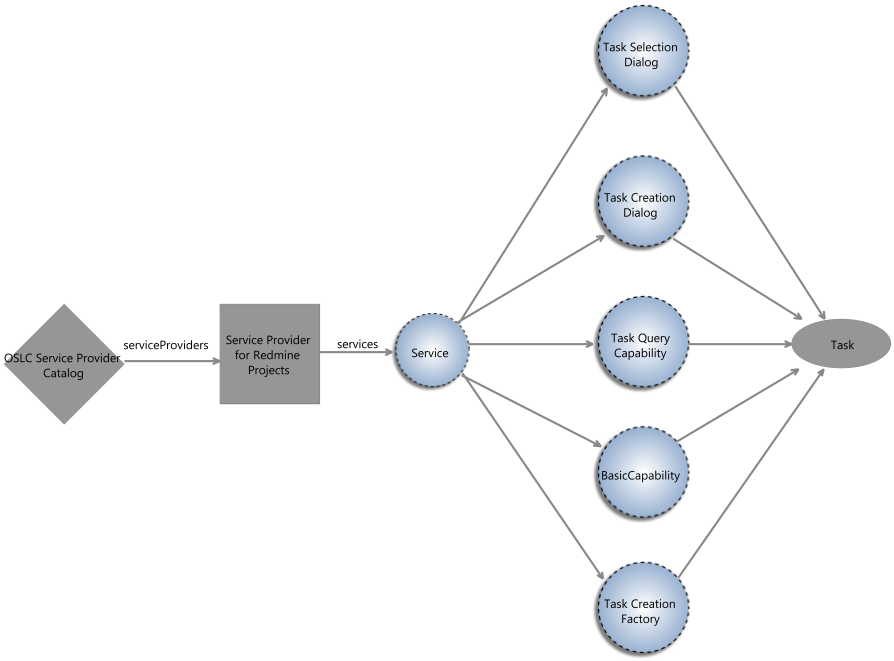


Figura 3. Interface do Adaptador OSLC para a ferramenta Redmine

Nosso estudo foca na análise de apenas uma abordagem. Além do Eclipse Lyo, existem outras alternativas para implementar soluções de integração de ferramentas baseadas em OSLC:

1. **Guaraná -** O Guaraná é uma abordagem para o suporte de implementações de soluções Enterprise Application Integration [Frantz et al. 2016]. Além disso, Guaraná possui uma linguagem gráfica denominada Guaraná DSL, que permite engenheiros de software a projetar soluções de integração independente de plataformas de integração.
2. **Tool Integration Language (TIL) -** É uma linguagem de domínio específico para cadeia de ferramentas [Biehl 2011]. TIL serve para diversos propósitos, entre eles a geração de códigos de adaptadores a partir da representação de modelos independente da tecnologia, inclusive OSLC.

5. Considerações Finais

Esse trabalho apresentou um estudo exploratório sobre a ferramenta Eclipse Lyo com o objetivo de implementar adaptadores OSLC. Para isso, foram utilizadas abordagens para geração de código-fonte baseada em modelos. Essas abordagens propõem mitigar alguns desafios que envolvem o processo de integração de ferramentas pois permitem desenvolvedores a trabalhar com um nível maior de abstração. Apesar disso, foram encontradas poucas alternativas para a construção de adaptadores OSLC baseadas em DSLs na literatura.

O Eclipse Lyo surge como objeto de estudo para se atingir ambientes integrados de ponta a ponta por meio do OSLC. Nosso trabalho possibilitou identificar a estrutura que um projeto OSLC segue, bem como as tecnologias envolvidas no processo do desenvolvimento dos adaptadores.

Não existe uma receita para desenvolver adaptadores OSLC, o que dificultou bastante o aprendizado sobre a DSL Lyo e o respectivo suporte ferramental que suporta a execução das rotinas de integração. Apesar do Eclipse Lyo ser utilizado para desenvolver adaptadores OSLC na indústria, como visto nos trabalhos [Biró et al. 2017], [El-khoury et al. 2019], [Nardone et al. 2020], [Gürdür et al. 2018] e [Zhang and Møller-Pedersen 2014], esses estudos propõem integrações em contextos de sistemas críticos de segurança. Dessa forma, o desenvolvimento de adaptadores OSLC através do Lyo não foi totalmente explorado na área de ES. Além disso, cada cenário de integração em um ciclo de vida de aplicação de uma empresa possui particularidades que devem ser levadas em consideração na hora de tomar as decisões do projeto. Por mais que o estudo tenha considerado representações de ferramentas bastante utilizadas no desenvolvimento de software, o cenário representado jamais pode refletir um ambiente real de uma empresa.

Observou-se que o aparato ferramental de suporte propicia a geração automática de código. Entretanto, apenas os esqueletos dos códigos são gerados dessa forma. É necessário implementar manualmente o conteúdo dos métodos para que seja estabelecida a comunicação entre as ferramentas provedoras e consumidoras, possibilitando a troca dos artefatos mantidos por elas. Isso caracteriza uma limitação do aparato ferramental investigado, uma vez que a geração de 100% do código poderia ser obtida para OSLC baseado na DSL investigada. Os adaptadores desenvolvidos ainda precisam ser customizados e implementar algumas interfaces gráficas para busca e acesso aos artefatos. Apesar disso, com base nesse estudo foi possível identificar os componentes mínimos necessários para implementar um adaptador de ferramentas, e portanto viável para a execução de um estudo exploratório em domínios de ES.

As integrações com OSLC limitam-se a apenas algumas fases do desenvolvimento. Um dos motivos disso é a complexidade para desenvolver adaptadores para essas soluções de integrações. O Estudo do Eclipse Lyo possui entre os objetivos identificar as atividades necessárias para o desenvolvimento de adaptadores e possíveis vantagens e desvantagens que motivem a evolução da ferramenta por ser um projeto de código-fonte aberto ou a proposta de novas alternativas para o desenvolvimento de adaptadores OSLC com o intuito de fomentar a transferência de conhecimento tácito em explícito do OSLC para engenheiros de software.

Os próximos passos da pesquisa incluem finalizar o desenvolvimento dos adaptadores, realizar uma avaliação empírica da viabilidade do uso do Eclipse Lyo em ambientes organizacionais com a execução de um estudo de caso, aprofundar os conceitos de representações comuns de ativos de software para diferentes áreas do conhecimento e explorar outras DSLs identificadas na literatura para a representação de elementos de integração de ferramentas de ES por meio de OSLC.

Por fim, o OSLC possui um grande potencial como tecnologia para automatizar todo o ambiente de desenvolvimento devido a suas características como flexibilidade e reuso. Além disso, há o interesse da indústria no desenvolvimento de processos, metodologias e ferramentas motivadas principalmente para reduzir o custo operacional dessas integrações, inclusive por meio do Eclipse Lyo.

6. Agradecimentos

Este estudo foi parcialmente financiado através da Pró-Reitoria de Pesquisa (PROPESQ), com bolsa do programa AGP (Apoio a Grupos de Pesquisa), e pela FAPERGS, por meio do projeto ARD N. 19/2551-0001268-3.

Referências

Aichernig, B. K., Hörmaier, K., Lorber, F., Nickovic, D., Schlick, R., Simoneau, D., and Tiran, S. (2014). Integration of requirements engineering and test-case generation via oslc. In *2014 14th International Conference on Quality Software*, pages 117–126.

Biehl, M. (2011). Tool integration language (til). Technical Report 2011:14, KTH, Mechatronics. QC 20111130.

Biehl, M., El-Khoury, J., Loiret, F., and Törngren, M. (2014). On the modeling and generation of service-oriented tool chains. *Software & Systems Modeling*, 13(2):461–480.

Biró, M., Kossak, F., Klespitz, J., and Kovács, L. (2017). Graceful integration of process capability improvement, formal modeling and web technology for traceability. In Stolfa, J., Stolfa, S., O'Connor, R. V., and Messnarz, R., editors, *Systems, Software and Services Process Improvement*, pages 381–398, Cham. Springer International Publishing.

Brown, A. W. and Penedo, M. H. (1992). An annotated bibliography on integration in software engineering environments. *SIGSOFT Softw. Eng. Notes*, 17(3):47–55.

Ebert, C. (2013). Improving engineering efficiency with plm/alm. *Software & Systems Modeling*, 12(3):443–449.

El-khoury, J. (2016). Lyo code generator: A model-based code generator for the development of oslc-compliant tool interfaces. *SoftwareX*, 5:190 – 194.

El-khoury, J., Berezovskyi, A., and Nyberg, M. (2019). An industrial evaluation of data access techniques for the interoperability of engineering software tools. *Journal of Industrial Information Integration*.

Fowler, M. and Parsons, R. (2011). *Domain-specific languages*. Addison-Wesley.

Frantz, R. Z., Corchuelo, R., and Roos-Frantz, F. (2016). On the design of a maintainable software development kit to implement integration solutions. *Journal of Systems and Software*, 111(1):89–104.

Gürdür, D., Feljan], A. V., El-khoury, J., Mohalik], S. K., Badrinath, R., Mujumdar], A. P., and Fersman, E. (2018). Knowledge representation of cyber-physical systems for monitoring purpose. *Procedia CIRP*, 72:468 – 473. 51st CIRP Conference on Manufacturing Systems.

Hohpe, G. (2003). *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. Addison-Wesley Professional.

Kleppe, A., Warmer, J., and Bast, W. (2003). Mda explained: The model driven architecture: Practice and promise.

Linked Data (2006). Linked data web page. Accessed at April 2020.

Messerschmitt, D. G. (2005). *Software Ecosystem: Understanding an Indispensable Technology and Industry (The MIT Press)*. The MIT Press.

Nardone, R., Marrone, S., Gentile, U., Amato, A., Barberio, G., Benerecetti, M., Guglielmo], R. D., Martino], B. D., Mazzocca, N., Peron, A., Pisani, G., Velardi, L., and Vittorini, V. (2020). An oslc-based environment for system-level functional testing of ertms/etcs controllers. *Journal of Systems and Software*, 161:110478.

OSLC (2020). Open services for lifecycle collaboration primer web page. Accessed at February 2020.

Runeson, P. and Höst, M. (2008). Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, 14(2):131.

Schwaber, C. et al. (2006). The changing face of application life-cycle management. *Forrester Research*, 18.

Thomas, I. and Nejmeh, B. A. (1992). Definitions of tool integration for environments. *IEEE Software*, 9(2):29–35.

Wicks, M. and Dewar, R. (2007). A new research agenda for tool integration. *Journal of Systems and Software*, 80(9):1569 – 1585. Evaluation and Assessment in Software Engineering.

Zhang, W. and Møller-Pedersen, B. (2014). Modeling of tool integration resources with oslc support. In *2014 2nd International Conference on Model-Driven Engineering and Software Development (MODELSWARD)*, pages 99–110.

Evolução do Ambiente SMartyModeling por meio de Testes Exploratórios

Leandro F. Silva[1], **Bruno Fernandes**[1], **Edson OliveiraJr**[1]

[1]Departamento de Informática
Universidade Estadual de Maringá (UEM)
Maringá, Brasil.

leandroflores7@gmail.com, bruno.fernandes@msn.com, edson@din.uem.br

Abstract. *Software Product Line (SPL) is a software reuse approach that has been consolidated over the past few years. However, the absence of tools that offer native support to the main activities in the life cycle of an SPL, motivated the development of the SMartyModeling environment. In order to identify problems when interacting with the environment, a series of exploratory tests were carried out. Therefore, this article presents the planning and execution of these exploratory tests, from the point of view of a tester who did not participate in the process of implementing the environment. The 12 defects resulting from these tests were discussed, describing the context in which they were identified and whether, to date, they have been corrected. Defects were also categorized by the features of the environment. The defects identified will be corrected in a new version of the environment. The result explains the importance of expanding to new tests that allow a safer and more reliable environment.*

Resumo. *Linha de Produto de Software (LPS) é uma abordagem de reuso de software que vem se consolidando no decorrer dos últimos anos. No entanto, a ausência de ferramentas que ofereçam suporte nativo às principais atividades no ciclo de vida de uma LPS, motivaram o desenvolvimento do ambiente SMartyModeling. Com o objetivo de identificar problemas ao interagir com o ambiente, foram realizados uma série de testes exploratórios. Portanto, este artigo apresenta o planejamento e a execução destes testes exploratórios, a partir do ponto de vista de um testador que não participou do processo de implementação do ambiente. Foram discutidos os 12 defeitos resultantes desses testes, descrevendo o contexto em que foram identificados e se, até o momento, foram corrigidos. Os defeitos foram também categorizados pelas funcionalidades do ambiente. Os defeitos identificados serão corrigidos em uma nova versão do ambiente. O resultado explica a importância de expandir para novos testes que permitam um ambiente mais seguro e confiável.*

1. Introdução

Softwares são constituídos de código-fonte e sua respectiva documentação, sendo o principal ativo de produtos. A preocupação com a competitividade no desenvolvimento de software se tornou uma questão primordial para empresas deste segmento, independentemente do porte e mercado alvo. Diante disso, os principais desafios passaram a ser lidar com o aumento de diversidade, demandas pela diminuição do tempo para entrega e desenvolvimento de software confiável [Sommerville 2011]. Técnicas voltadas ao reúso de

artefatos foram propostas, incluindo a Linha de Produto de Software (LPS), uma abordagem de desenvolvimento, que aplica o reúso de maneira sistemática em um domínio de atuação [Linden et al. 2007].

A ausência de ferramentas que ofereçam suporte às principais atividades relacionadas à LPS motivaram o desenvolvimento do ambiente SMartyModeling. O ambiente foi avaliado inicialmente em relação à modelagem de LPS. E como resultado, foi possível identificar limitações, que motivaram o desenvolvimento de uma nova versão com os defeitos identificados corrigidos e novas funcionalidades incluídas [Silva 2017]. Como disciplina, a Engenharia de Software (ES) se preocupa com todo o ciclo de produção, incluindo desde o levantamento de requisitos, especificação, desenvolvimento, validação e evolução de software. A validação de software tem a intenção de mostrar que um software atende suas especificações. E o teste de software é a principal técnica de validação [Sommerville 2011]. Nesse contexto, o planejamento e a execução de testes é fundamental para a evolução de um software, em especial a execução contínua de diferentes técnicas de testes e avaliações de usabilidade. A aplicação dos testes deve ser feita de maneira independente, incluindo a participação de membros que não participaram diretamente do processo de desenvolvimento, com o objetivo de reduzir o viés [Myers et al. 2011].

Portanto, este trabalho teve como objetivo realizar testes de software no SMartyModeling. Inicialmente, foi realizada uma análise mais minuciosa sobre a arquitetura do SMartyModeling e uma revisão sobre as principais técnicas de teste de software. Posteriormente, foram projetados e realizados testes exploratórios no ambiente, e, finalmente, analisados e discutidos os resultados obtidos. Este trabalho está organizado da seguinte maneira: a Seção 2 descreve os principais conceitos relacionados, a Seção 3 apresenta uma visão geral do ambiente testado neste estudo: o SMartyModeling, com a discussão dos resultados detalhadas na Seção 5, e as contribuições, limitações e trabalhos futuros na Seção 6.

2. Fundamentação Teórica

2.1. Teste de Software

O Teste de Software pode ser definido como um processo, ou conjunto de processos, projetado para garantir que o software execute o que foi planejado e que não realize nenhuma atividade não esperada [Myers et al. 2011]. O teste é destinado a mostrar que um programa realiza o que é proposto e para descobrir os defeitos do programa antes do uso. Os resultados do teste são voltados à procura de erros, anomalias ou informações sobre os atributos não funcionais do programa [Sommerville 2011].

Com objetivo de complementar o conceito de teste, é necessário apresentar a definição dos termos defeito, erro e falha no contexto de software. O defeito consiste em uma deficiência mecânica ou algorítmica que, se ativada, pode levar a uma falha. O erro corresponde à um item de informação ou estado de execução inconsistente. E, por fim, a falha é definida como um evento notável em que o sistema viola suas especificações. Assim, pode-se dizer que erro é a ação humana incorreta, defeito é a forma incorreta que o software foi construído e a falha é desvio percebido ao executar o software [ISTQB 2018].

No entanto, os testes não podem demonstrar se o software é completamente livre de defeitos em qualquer situação. É sempre possível que um novo teste eventualmente encontre mais problemas no sistema [Myers et al. 2011]. Segundo Dijkstra [Dijkstra 1972],

os testes mostram apenas a presença de erros, e não sua ausência. Apesar disso, o teste é parte de um amplo processo de verificação e validação do software, sendo fundamental para estabelecer a confiança de que o software está pronto para atender seu objetivo final [Sommerville 2011].

É praticamente impossível demonstrar a ausência de defeitos em um software, pelo fato de normalmente existir uma quantidade infinita de combinações possíveis para os dados de entrada [Dijkstra 1972]. Neste cenário, foram propostas estratégias econômicas para a execução dos testes. Duas técnicas clássicas de testes descritas foram [Myers et al. 2011]: **Teste de Caixa-Preta:** tem como objetivo testar o software sem qualquer preocupação com a estrutura interna do programa, por isso caixa-preta. Assim, a partir entrada de dados (válidas e inválidas), espera-se uma saída após o processamento. Como exemplo, temos o Teste de Sistema e o Teste de Aceite; e **Teste de Caixa-Branca:** tem como objetivo testar o software examinando a estrutura interna do programa, sendo orientado a lógica. Portanto, busca-se fazer com que cada instrução do programa seja executada pelo menos uma vez para um teste completo, o que se torna inviável. Como exemplo, temos o Teste de Unidade e o Teste de Integração.

Diante da inviabilidade lógica e das limitações para construção dos testes pelas técnicas apresentadas, a técnica de teste baseada na experiência foi proposta. Tal técnica é voltada à uma política mais informal e tem sua modelagem, execução, registro e avaliação feitos dinamicamente durante a execução dos testes. Como exemplo, o teste exploratório é utilizado, principalmente em situações que há especificações limitadas ou pressão de tempo significativa nos testes [ISTQB 2018].

2.2. Linha de Produto de Software e Gerenciamento de Variabilidades

Linha de Produto de Software (LPS) é uma abordagem de desenvolvimento centrado no reuso de forma sistemática em um domínio de atuação [Linden et al. 2007]. LPS é composta por um conjunto de sistemas de software que compartilham características comuns e gerenciáveis, que satisfazem as necessidades de um segmento particular ou de uma missão [Pohl et al. 2005]. A engenharia de LPS abrange todas as atividades envolvidas no planejamento, produção, e manutenção de produtos. Um *framework* é proposta por [Pohl et al. 2005] para a Engenharia de LPS a partir de dois processos: **Engenharia de Domínio:** processo responsável pela identificação e definição das similaridades e variabilidades da LPS; e **Engenharia de Aplicação:** processo responsável pela aplicação da LPS, em que os produtos são construídos reutilizando artefatos de domínio e resolvendo as variabilidades.

A Engenharia de Domínio é composta de cinco atividades: Gerenciamento do Produto, Engenharia de Requisitos do Domínio, Análise do Domínio, Implementação do Domínio e Teste do Domínio. Estes processos produzem uma plataforma incluindo as similaridades entre as aplicações e suas variabilidades que permitem a customização de produtos em massa. A partir da Engenharia de Domínio, são definidos os ativos base da LPS. A Engenharia de Aplicação é composta pelas seguintes atividades: Engenharia de Aplicação, Análise da Aplicação, Implementação da Aplicação e Testes da Aplicação, tendo como resultado a instanciação de produtos finais a partir dos ativos base [Pohl et al. 2005]. A Figura 1 apresenta a visão geral destes dois processos. Na parte superior da Figura 1, a sequência de atividades da Engenharia de Produto é detalhada,

resultando na identificação e definição dos ativos, que formam a base para as atividades da Engenharia de Aplicação, descritas na parte inferior da Figura 1.

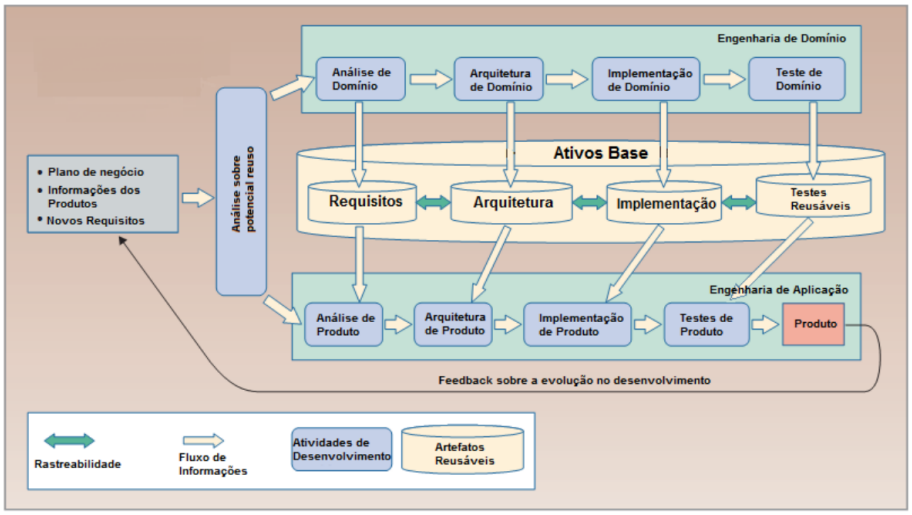


Figura 1. *Framework* de Engenharia de Linha de Produto de Software [Pohl et al. 2005].

Nos últimos anos, a abordagem de LPS vem definindo cada vez mais o Gerenciamento de Variabilidades (GV) como atividade fundamental para a instanciação de produtos em larga escala [Capilla et al. 2013]. Variabilidade de software é a capacidade de um sistema, ativo de software, ou ambiente de desenvolvimento ser configurado, customizado, ou alterado para uso em um domínio específico de uma forma pré-planejada. Em síntese, variabilidade é a forma como os membros de uma família de produtos podem se diferenciar entre si, ou seja, é o que difere os produtos gerados de uma mesma LPS [Capilla et al. 2013, Linden et al. 2007]. Para a abordagem de LPS, várias alternativas ao longo do tempo vêm sendo propostas para representar a variabilidade. No entanto, para a representação de uma variabilidade, existem três elementos fundamentais [OliveiraJr et al. 2010, Linden et al. 2007]:

- **Ponto de Variação:** descreve o ponto em que existe uma diferença nos produtos a serem instanciados, identificando os locais nos quais as variações são combinadas para ocorrer. São responsáveis por definir as diferenças da LPS;
- **Variante:** define os elementos para a resolução de um Ponto de Variação. Corresponde a uma alternativa de projeto para resolver uma determinada variabilidade. Define como uma variabilidade ou ponto de variação varia em uma LPS; e
- **Restrições entre Variantes:** estabelecem os relacionamentos entre uma ou mais Variantes com o objetivo de resolver seus respectivos Pontos de Variação.

Algumas abordagens para a representação de variabilidades em modelos UML são propostas na literatura, como PLUS [Gomaa 2006], Ziadi [Ziadi and Jezequel 2006], e SMarty [OliveiraJr et al. 2013]. A abordagem *Stereotype-based Management of Variability* (SMarty) é uma abordagem anotativa de GV baseada em estereótipos UML. A SMarty é composta por um perfil UML denominado *SMartyProfile*, que representa variabilidades por meio de estereótipos, e por um processo denominado *SMartyProcess*, que guia o usuário na identificação, representação e rastreamento de variabilidades em modelos de LPS [OliveiraJr et al. 2013]. A principal contribuição da SMarty é permitir o

gerenciamento de forma efetiva das variabilidades de uma LPS modeladas em UML. Os estereótipos definidos pela abordagem SMarty estão organizados em um perfil UML denominado *SMartyProfile* e em uma forma de representação de variabilidades por meio de estereótipos. O *SMartyProfile* é formado por um conjunto de estereótipos e meta-atributos para representar variabilidades em modelos UML de LPS [OliveiraJr et al. 2010].

3. O Ambiente SMartyModeling

O SMartyModeling é um ambiente para engenharia de LPSs baseadas em UML, nas quais as variabilidades são modeladas como estereótipos usando qualquer perfil UML compatível, adotando como padrão a abordagem SMarty. O ambiente oferece suporte à modelagem de diagramas de casos de uso, classes, componentes, sequência e atividades. As principais *features* do ambiente são: modelagem de diagramas, definição e restrição de variabilidades, rastreabilidade de elementos, e instanciação de diagramas por meio da resolução de variabilidades, configuração e exportação de produtos [Silva 2017].

A arquitetura do SMartyModeling foi instanciada a partir da VMTools-RA, uma Arquitetura de Referência para ferramentas de variabilidade de software [Allian 2016]. A VMTools-RA não especifica um estilo arquitetural, e a arquitetura do SMartyModeling foi instanciada considerando o padrão *Model-View-Controller* (MVC). O ambiente foi codificado na linguagem JavaSE para Desktop, com a arquitetura organizada em quatro pacotes: Modelo, Controle, Visão e Arquivo [Silva 2017]. A Figura 2 apresenta a interface principal do SMartyModeling, composta pelos seguintes componentes:

- **Painel Principal (Componente A):** contendo as principais operações do sistema: novo, abrir, salvar e fechar Projeto; desfazer, refazer; zoom+ e zoom-; exportar imagem, ajuda e sobre;
- **Painel de Operações (Componente B):** operações de acordo com o respectivo diagrama. Para o diagrama de casos de uso temos: arrastar e manipular elementos do diagrama; inserir novo ator; inserir novo caso e uso; inserir nova variabilidade; editar elemento, excluir elemento e inserir nova associação;
- **Painel do Projeto (Componente C):** organização do Projeto em uma ordem hierárquica, com a seguinte ordem: diagramas, variabilidades, produtos, instâncias, métricas e rastreabilidade;
- **Painel de Modelagem (Componente D):** modelagem dos diagramas; e
- **Painel de Informações (Componente E):** abas com as respectivas informações sobre o elemento selecionado.

No decorrer do desenvolvimento do ambiente, os testes ficaram restritos ao planejamento e execução de testes unitários envolvendo as principais classes de modelo e de controle, concentrando os esforços em cobrir as operações descritas de acordo com cada método. Os testes foram codificados utilizando o *framework* JUnit[1], voltado especificamente à automação de testes na linguagem de programação Java.

4. Testes Exploratórios no SMartyModeling

A partir da compreensão das principais funcionalidades do ambiente SMartyModeling, foram executados testes com o objetivo de analisar minuciosamente a ferramenta e si-

[1]Disponível em: `https://junit.org/junit4/javadoc/latest/index.html`.

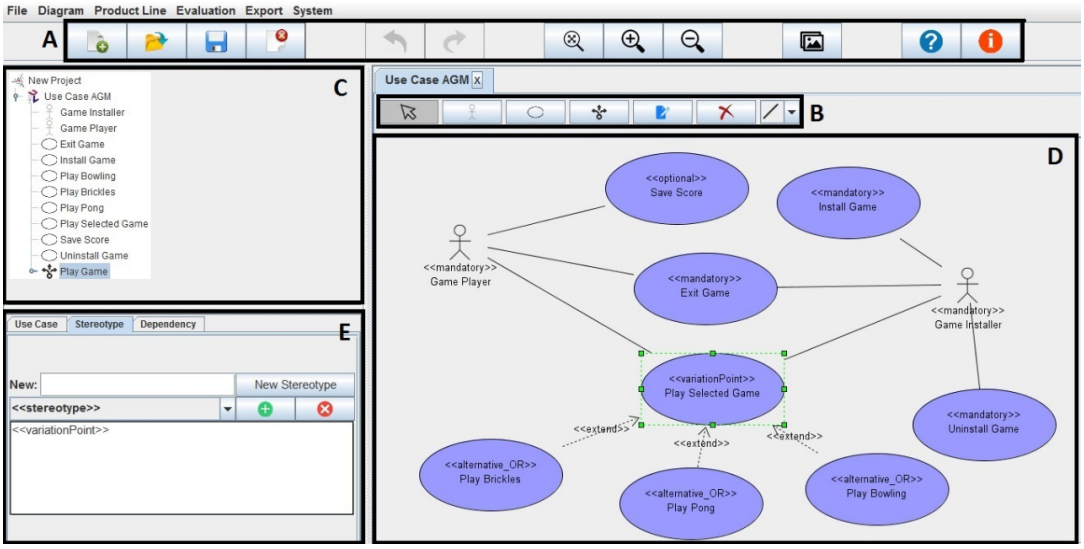


Figura 2. Interface do SMartyModeling.

multaneamente identificar possíveis defeitos[2] relacionados à falta de experiência no ambiente. Os testes foram executados por um usuário que não participou em nenhuma etapa do desenvolvimento da ferramenta. O usuário responsável pela realização do teste exploratório possui experiência em técnicas de testes aplicadas no mercado de trabalho, possuindo igualmente experiência em testes exploratórios, no entanto, não possui conhecimento avançado em LPS, e sobre as funcionalidades do ambiente.

A técnica de testes selecionada para o SMartyModeling foi o teste exploratório [Pfahl et al. 2014], principalmente em função do contexto de aplicação. Os testes exploratórios são especialmente úteis em ambientes em que a documentação é limitada e o responsável por examinar o software não possui conhecimento prévio e experiência sobre o software. A abordagem de testes exploratórios proporciona alguns benefícios que vão ao encontro das limitações mencionadas: contribuem para o aprendizado do funcionamento do sistema em teste, demanda um tempo aceitável de planejamento, além de revelar tipos de falhas que geralmente não são detectados em um plano ou caso de teste pré-estabelecido. Os testes exploratórios de maneira alguma podem ser utilizados como única estratégia de teste, no entanto podem ser utilizados como complemento à demais técnicas de teste com objetivo de obter uma melhoria na qualidade do software, e complementam eventuais falhas não identificadas nos testes unitários e avaliações de usabilidade inicialmente realizadas.

Primeiramente, para organização dos testes realizados, foi criado um quadro *Kanban* no Trello[3], organizando as atividades em listas, e possibilitando um controle mais efetivo sobre os testes realizados. Foi necessário preparar o ambiente para execução. Para isso, foram instalados o Apache NetBeans IDE 11.1[4], o OpenJDK[5] e o sistema de controle

[2]Todos os defeitos e correções estão disponíveis em `https://doi.org/10.5281/zenodo.3706642`.

[3]Trello é um aplicativo na web disponível em `https://trello.com` para gerenciamento de projetos utilizando conceitos de Kanban.

[4]Disponível em `https://netbeans.apache.org`

[5]Disponível em `https://openjdk.java.net`

de versão distribuído Git[6]. Foi realizado também o Clone do repositório da SMartyModeling no Git tendo assim acesso ao código-fonte. Com o ambiente de desenvolvimento preparado, foi possível executá-lo usando o Apache NetBeans IDE.

Portanto, com o ambiente SMartyModeling pronto, foram executados alguns testes exploratórios a fim de conhecer a ferramenta e encontrar defeitos relacionados a falta de experiência no ambiente para usuários iniciantes. As ocorrências de defeitos encontrados no teste foram categorizadas por funcionalidades, resultando em 12 defeitos identificados:

- **Defeito 1 - Redimensionar Tela:** ao redimensionar a janela o menu também é redimensionado e são criadas barras de rolagem horizontal e vertical no menu. Após a correção, foi feita a criação apenas da barra horizontal, sendo assim, possível o acesso a todos os botões mesmo com janela pequena;
- **Defeito 2 - Redimensionar Tela:** ao redimensionar a janela não está criando barra de rolagem, tornando alguns componentes inacessíveis. Após a correção, ao redimensionar a tela são criadas barras de rolagem horizontal e vertical;
- **Defeito 3 - Salvar Projeto:** a opção de salvar projeto não funciona após abrir o `Save Project` uma vez e fechar a opção sem salvar o projeto. Após ajustes realizados pelo desenvolvedor, o erro persistiu. Foi observado que o ambiente controla o salvamento do projeto pela ação do botão `Save Project`, porém apenas deveria identificar que foi salvo após clique na janela de salvamento e caso não ocorressem falhas. Foi repassado ao programador para que seja feita a correção;
- **Defeito 4 - Salvar Projeto:** não existe uma opção `Save As` para poder salvar o projeto com outro nome sem sobrescrever o atual. Após reportado ao programador, foi criada a opção `Save As`;
- **Defeito 5 - Abrir Projeto:** ao acionar `Open Project` ou `Close Project` não é aberta opção para salvar o projeto. O usuário pode acabar perdendo o trabalho feito;
- **Defeito 6 - Salvar Projeto:** ao salvar o projeto não está sendo salvo o arquivo com a extensão padrão SMTY, assim, ao abrir um projeto já salvo é necessário alterar a opção `Files of Type` para "All Files". Foi repassado ao programador para que seja feita a correção;
- **Defeito 7 - Abrir Projeto:** ao acionar a opção `Open Project` está sendo aberta uma tela com nome `Save`, inclusive com botão `Save` ao invés de `Open`. Após correção, o nome da janela e do botão de abertura de projetos está com grafia correta;
- **Defeito 8 - Atributos dos Elementos:** alguns caracteres são conflitantes com a estrutura XML e ao salvar um elemento com algum desses caracteres no nome ocorre falha ao abrir o projeto. Após a correção, este erro foi corrigido;
- **Defeito 9 - Desfazer Ação:** a opção `Undo` não está funcionando. Foi apagado um relacionamento entre "Activities" e não foi possível desfazer a remoção;
- **Defeito 10 - Manipulação de Elementos:** após apagar um relacionamento do tipo *flow* não está sendo possível incluir o mesmo novamente. Após correção, está sendo possível excluir um relacionamento e posteriormente incluir novamente;

[6]Disponível em `https://git-scm.com`

- **Defeito 11 - Painel de Projetos:** foi identificada a ocorrência de uma falha na visualização do painel do projeto ao alterar o campo `Variation Point` em uma Variabilidade. Ao modificar o ponto de variação, o SMartyModeling não estava mostrando esta alteração no painel mesmo ao salvar o projeto. Após a correção, ao selecionar uma nova opção para o ponto de variação já é alterado também no painel do projeto; e
- **Defeito 12 - Manipulação de Elementos:** para organização e legibilidade do diagrama, é possível mover as anotações de tipo de associação. Porém, ao clicar em qualquer um dos elementos do diagrama o texto da anotação volta ao local original.

A Figura 3 apresenta em detalhes o **Defeito 11**. Na tela A, a `Variation Point` está diferente do que o mostrado no painel de projeto. Após a correção (B), a `Variation Point` está condizendo com o mostrado no painel.

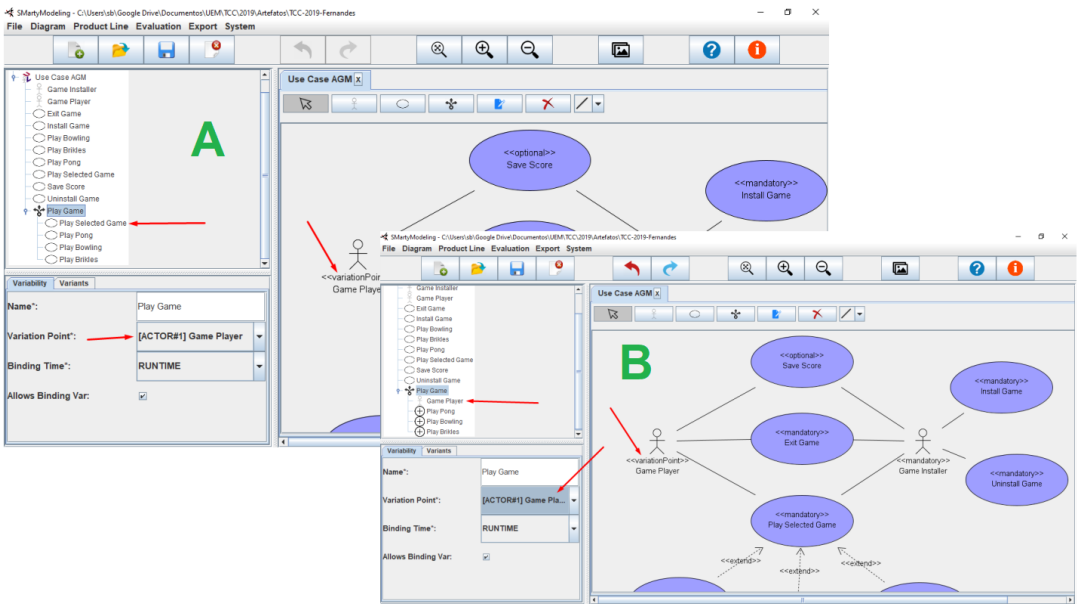


Figura 3. Detalhes do Defeito 11 (A) e após a correção (B).

5. Discussão dos Resultados

Como apresentado, os testes exploratórios resultaram em 12 defeitos no SMartyModeling, sendo categorizados de acordo com as funcionalidades relacionadas a "Salvar/Abrir Projeto", "Elementos dos Diagramas", "Dimensionamento da Janela", "Desfazer Ação" e "Painel de Projeto". A Tabela 1 apresenta os defeitos encontrados, classificando-os conforme a funcionalidade do ambiente.

Conforme a categorização dos defeitos, é possível observar que a considerável parte está concentrada nas funcionalidades implementadas no pacote responsável pela manipulação de arquivos. Portanto, como resultado inicial, podemos citar uma correção destas funcionalidades no ambiente, sendo realizada a correção de parte dos defeitos de maneira imediata, em especial, dos defeitos 1, 2, 4, 6, 7, 8, 10 e 11. Estas correções contribuíram significativamente para a melhoria e evolução da ferramenta.

Tabela 1. Funcionalidade por Identificador e Quantidade dos Defeitos.

Funcionalidade	Identificador dos Defeitos	Quantidade de Defeitos
Salvar/Abrir Projeto	3, 4, 5, 6 e 7	5
Elementos do Diagrama	8, 10 e 12	3
Redimensionar Tela	1 e 2	2
Desfazer Ação	9	1
Painel de Projeto	11	1

O teste exploratório ofereceu uma liberdade maior ao testador, de maneira que o testador interagiu com o SMartyModeling da maneira que considerou mais adequada e utilizou as funcionalidades que considerou importantes dentro do cenário de LPS. E mesmo diante dos defeitos identificados, o testador considerou o ambiente intuitivo, mesmo considerando um usuário com pouca familiaridade com modelagem LPS. O teste exploratório também permitiu, a partir de um ponto vista diferente do desenvolvedor, explorar as funcionalidades do ambiente em cenários alternativos, avaliar a capacidade de execução e encontrar pontos vulneráveis e suscetíveis a erros.

6. Conclusão

Este trabalho apresentou uma revisão sobre Teste de Software, LPS, descreveu em alto nível o ambiente SMartyModeling e descreveu o planejamento e execução de um teste exploratório realizado sobre o SMartyModeling, bem como os respectivos defeitos identificados. A principal contribuição deste trabalho é do ponto de vista prático, considerando a aplicação de técnicas de teste em um sistema real.

Como resultados obtidos deste trabalho, pode-se citar, principalmente, a identificação de falhas e a correção de defeitos no ambiente SMartyModeling. Estas correções contribuíram para a melhoria e evolução da ferramenta, com o objetivo principal de deixa-lá mais intuitiva para uso. O teste exploratório executado permitiu complementar os resultados dos testes unitários realizados durante o desenvolvimento e da avaliação inicial de usabilidade, detectando assim novos defeitos a partir do ponto de vista de um usuário sem experiência em utilizar o ambiente, porém familiarizado com a atividade de teste de software.

O teste exploratório realizado foi muito importante pelo fato de identificar uma série de defeitos que, assim que corrigidos, permitiram a correção e evolução do ambiente. Contudo, os resultados do teste exploratório reforçam a necessidade de realizar testes complementares, expandindo para as demais técnicas apresentadas na literatura e ampliando para cenários mais complexos.

Referências

Allian, A. P. (2016). VMTools-RA: a Reference Architecture for Software Variability Tools. Master's thesis, State University of Maringá. in portuguese.

Capilla, R., Bosch, J., and Kang, K. C. (2013). *Systems and Software Variability Management: Concepts, Tools and Experiences*. Springer.

Dijkstra, E. W. (1972). The humble programmer - ACM Turing Award Lecture. *Communications of the ACM*, 15(10):859–866.

Gomaa, H. (2006). *Designing Software Product Lines with UML 2.0: From Use Cases to Pattern-Based Software Architectures*.

ISTQB (2018). *Certified Tester Foundation Level Syllabus - International Software Testing Qualifications Board (ISTQB)*. URL: https://www.bstqb.org.br/.

Linden, F. J. V. D., Schmid, K., and Rommes, E. (2007). *Software product lines in action: The best industrial practice in product line engineering*, volume 20. Springer-Verlag New York, Inc.

Myers, G. J., Sandler, C., and Badgett, T. (2011). *The Art of Software Testing*. Wiley Publishing, 3rd edition.

OliveiraJr, E., Gimenes, I. M. S., Maldonado, J. C., Masiero, P. C., and Barroca, L. (2013). Systematic Evaluation of Software Product Line Architectures. *Journal of Universal Computer Science (JUCS)*, 19(1):25–52.

OliveiraJr, E., Maldonado, J. C., and Gimenes, I. M. S. (2010). Systematic Management of Variability in UML-based Software Product Lines. *Journal of Universal Computer Science (JUCS)*, pages 2374–2393.

Pfahl, D., Yin, H., Mäntylä, M. V., and Münch, J. (2014). How is exploratory testing used? a state-of-the-practice survey. In *ESEM*, pages 1–10, New York, NY, USA. ACM.

Pohl, K., Bockle, G., and van der Linden, F. (2005). *Software Product Line Engineering: Foundations, Principles, and Techniques*, volume 26. Springer-Verlag New York Inc., Secaucus, NJ, USA.

Silva, L. F. d. (2017). SMartyModeling: um Ambiente de Modelagem para Linha de Produto de Software com base no Eclipse Modeling Framework. Monografia (Bacharel em Informática), UEM (Universidade Estadual de Maringá), Maringá - PR, Brasil.

Sommerville, I. (2011). *Engenharia de Software*. Pearson Education do Brasil, São Paulo, 9 edition.

Ziadi, T. and Jezequel, J.-M. (2006). Software Product Line Engineering with the UML: Deriving Products. In *Software Product Lines*, pages 557–588. Springer, Berlin.

Configuração de Algoritmos Genéticos Multiobjetivos para Otimização de Projeto de Arquitetura de Linha de Produto

Narcizo Gabriel Freitas Palioto, Thelma Elita Colanzi[1]

[1]Departamento de Informática – Universidade Estadual de Maringá (UEM)
Maringá – Brasil

narcizo.gabriel2@gmail.com, thelma@din.uem.br

Resumo. *Algoritmos de busca têm sido explorados com sucesso na otimização de projeto de Arquitetura de Linha de Produto de Software (PLA) na abordagem seminal chamada Multi-Objective Approach for Product-Line Architecture Design (MOA4PLA). Tal abordagem produz um conjunto de alternativas de projeto de PLA que melhora os diferentes fatores otimizados. Atualmente, o algoritmo utilizado nesta abordagem é o algoritmo NSGA-II (Non-dominated Sorting Genetic Algorithm II), um algoritmo genético multiobjetivo que otimiza várias propriedades simultaneamente. Apesar de resultados experimentais promissores, estudar a melhor combinação de configuração dos parâmetros do algoritmo genético é imprescindível para obter melhores resultados. Valores de referência para os parâmetros ainda não foram definidos para otimização de projeto de PLA porque este é um tópico de pesquisa incipiente. Nesse contexto, o objetivo deste trabalho é identificar os valores mais adequados para configurar o algoritmo NSGA-II para a otimização de projetos de PLA por meio de um estudo experimental. Uma análise quantitativa baseada no indicador de qualidade hypervolume e em testes estatísticos foi realizada para determinar o valor mais adequado para configurar cada parâmetro do algoritmo.*

1. Introdução

Linha de Produto de Software (LPS) é um conjunto de técnicas e métodos de desenvolvimento que visa a maximização da qualidade e do reuso do software, a minimização dos custos através do reuso de artefatos e a entrega mais rápida do software. Uma LPS contém um conjunto de funcionalidades comuns e que satisfazem as necessidades de um segmento de mercado particular [Linden et al. 2007]. Um dos principais artefatos de uma LPS é a Arquitetura de Linha de Produto (PLA - *Product Line Architecture*). Porém, a obtenção da PLA não é uma tarefa trivial, pois fatores relacionados a diferentes propriedades arquiteturais devem ser atendidos, os quais muitas vezes são conflitantes, por exemplo modularidade de características, reusabilidade e extensibilidade da LPS [Colanzi et al. 2014].

Problemas da Engenharia de Software similares a este têm sido eficientemente resolvidos com algoritmos de busca em um campo de pesquisa conhecido como Engenharia de Software Baseada em Busca (SBSE - *Search Based Software Engineering*) [Harman and Jones 2001]. A SBSE converte um problema de engenharia de software num problema de busca computacional que pode ser resolvido com uma metaheurística.

Algoritmos genéticos são uma metaheurística muito utilizada em SBSE. Esses algoritmos são baseados na analogia com os processos de seleção natural e genética evolucionária [Goldberg 1989]. O objetivo desses algoritmos é copiar a natureza levando

em consideração a adaptação e a sobrevivência do mais forte. Eles incluem operadores de busca de três tipos: seleção, cruzamento e mutação [Goldberg 1989]. Todas as metaheurísticas, incluindo algoritmos genéticos, são algoritmos estocásticos. Logo há um fator de aleatoriedade associado à aplicação dos operadores de busca.

A MOA4PLA (*Multi-Objective Approach for Product-Line Architecture Design*) [Colanzi et al. 2014] é uma abordagem que oferece um tratamento multiobjetivo para avaliar e otimizar projetos de PLA. Essa abordagem foi implementada na ferramenta OPLA-Tool [Freire et al. 2020] utilizando o algoritmo NSGA-II (*Non-dominated Sorting Genetic Algorithm II*) [Deb et al. 2002], um algoritmo genético multiobjetivo muito eficiente e popular para resolver problemas de otimização. NSGA-II é um algoritmo multiobjetivo, pois permite a otimização simultânea de mais de um objetivo (fator). No caso específico de otimização de projetos de PLA, os objetivos são constituídos por métricas de software relacionadas a propriedades arquiteturais que se pretende otimizar.

Para utilizar o NSGA-II é preciso configurar as probabilidades de aplicação dos operadores de mutação e de cruzamento, bem como qual o tamanho da população de indivíduos e o número de gerações do processo evolutivo. Em geral, esses valores variam de acordo com o problema em questão e com características dos indivíduos fornecidos como entrada para o processo de otimização [Goldberg 1989]. Considerando que os algoritmos de busca são estocásticos, recomenda-se a execução de várias rodadas do algoritmo e o uso de indicadores de qualidade para averiguar qual a melhor combinação de valores para os parâmetros [Arcuri and Fraser 2011].

Essa situação se aplica ao uso do NSGA-II para resolver o problema de otimização de projeto de PLA. Os parâmetros de configuração desse algoritmo geralmente são: (a) probabilidade de mutação; (b) probabilidade de cruzamento; (c) tamanho da população; (d) número de gerações (ou de avaliações de *fitness*) e (e) formação da população inicial. Existem na literatura recomendações de valores para esses parâmetros [Goldberg 1989], mas estudos anteriores mostraram que os valores de referência para um algoritmo canônico não são os mais apropriados para otimização de projeto de PLA, o que justifica investigar valores especificamente para esse problema.

Considerando o contexto exposto anteriormente, o objetivo deste trabalho é investigar os valores mais apropriados para calibrar o algoritmo NSGA-II, a fim de otimizar projeto de PLA no contexto da MOA4PLA na ferramenta OPLA-Tool. Para isso, foram realizados estudos experimentais com diferentes valores para os parâmetros de configuração de um algoritmo genético e posterior comparação dos resultados usando o indicador de qualidade de algoritmos multiobjetivo *hypervolume* e testes estatísticos, a fim de obter os valores mais apropriados para resolver o referido problema. Os experimentos são definidos na Seção 4 e os resultados são apresentados na Seção 5. Os conceitos fundamentais envolvidos neste trabalho são apresentados a seguir.

2. Referencial Teórico

2.1. Algoritmos Genéticos

Algoritmos genéticos são inspirados na evolução genética [Goldberg 1989]. Esses algoritmos codificam uma potencial solução ao problema em estruturas de dados que se assemelham a cromossomos e aplicam operadores de recombinação nessas estruturas de modo que informações cruciais não sejam perdidas.

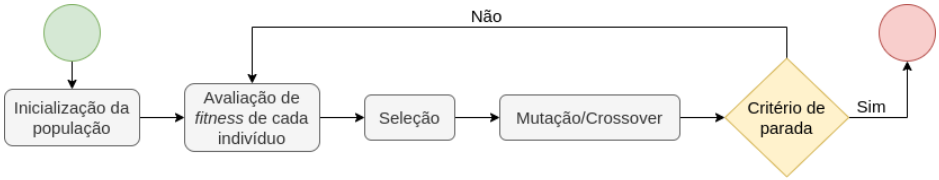


Figura 1. Passos de um algoritmo genético.

A Figura 1 explica os passos de um algoritmo genético. Uma população inicial é gerada aleatoriamente, sendo que cada indivíduo é uma possível solução. Em seguida é feito o cálculo da função de *fitness* de cada indivíduo da população. Essa função de *fitness* mede o quão adaptado um indivíduo está de acordo com o problema. A seleção escolhe os indivíduos mais aptos para continuarem para as próximas gerações e esses indivíduos passam por operadores genéticos de mutação e *crossover* com o objetivo de perpetuar seus genes. Enquanto o critério de parada não é atingido é realizada uma nova geração com os indivíduos gerados pelos operadores de mutação e de *crossover* e, caso o critério de parada seja atingido, o algoritmo se encerra. O operador de cruzamento faz uma combinação de partes de dois indivíduos selecionados para gerar novos indivíduos. Então, o operador de mutação modifica aleatoriamente um indivíduo descendente. A população descendente criada a partir dos operadores de busca substitui a população anterior.

O NSGA-II [Deb et al. 2002] é o algoritmo genético multiobjetivo mais difundido e utilizado na literatura. Ele lida com problemas multiobjetivo, aqueles que dependem de diversos fatores (objetivos) que estão em conflito. Desse modo, geralmente não há uma única solução. Assim, o algoritmo retorna diversas boas soluções que representam o *trade-off* entre os objetivos definidos. Estas soluções são chamadas de não-dominadas e formam a fronteira de Pareto.

2.2. Projeto de Arquitetura de Linha de Produto de Software Baseado em Busca

A Arquitetura de Linha de Produto (PLA) é um meio de garantir o reúso de uma LPS, pois ela envolve todos as características que são compartilhadas por todos os produtos que derivam de uma LPS [Contieri Jr et al. 2011]. Trata-se de um projeto que contém todos os componentes e características possíveis, comuns e variáveis, a todos os produtos a serem derivados da LPS [Linden et al. 2007]. Desse modo, é possível instanciar a arquitetura individual para cada produto da LPS. A importância de uma PLA é evidente, a complexidade dos softwares cresce a cada dia e o custo de desenvolvimento e manutenção deve ser contido. PLAs ajudam nesse sentido, pois funcionam como uma planta para a criação de famílias de produtos similares diminuindo drasticamente o custo e o esforço de *design* de software e o desenvolvimento de vários produtos [Linden et al. 2007].

O projeto de uma PLA modular, extensível e reusável não é uma tarefa simples. Um arquiteto precisa saber analisar os *trade-offs* necessários entre as métricas utilizadas e isso demanda um grande esforço humano.

MOA4PLA é uma abordagem baseada em SBSE e tem como foco avaliar e melhorar uma PLA otimizando propriedades arquiteturais convencionais, modularização de características e extensibilidade de LPS por meio de algoritmos multiobjetivos [Colanzi et al. 2014]. A Figura 2, extraída de [Colanzi et al. 2014], mostra as atividades realizadas pela MOA4PLA, cujas descrições são apresentadas a seguir:

- Construção da Representação de PLA (*Construction of the PLA Representation*): Recebe como entrada um arquivo XML que representa um diagrama de classes

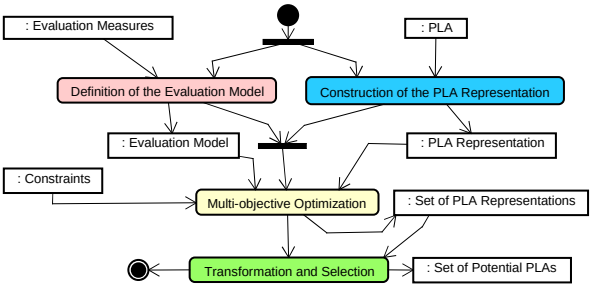


Figura 2. Atividades da MOA4PLA

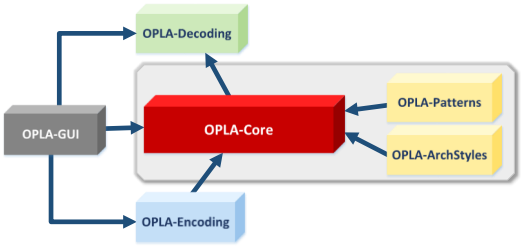


Figura 3. Módulos da OPLA-Tool.

contendo o projeto de PLA e gera uma representação computacional a partir do diagrama de classes da PLA;

- Definição do Modelo de Avaliação (*Definition of the Evaluation Model*): Escolha das métricas a serem utilizadas no processo de otimização para determinar a qualidade das soluções geradas, por exemplo, acoplamento e modularização de características;

- Otimização Multiobjetivo (*Multi-objective Optimization*): após a execução das atividades anteriores o algoritmo de busca NSGA-II gera um novo conjunto de representações de PLA que otimiza as métricas escolhidas pelo arquiteto. Este projeto está relacionado a esta atividade;

- Transformação e Seleção (*Transformation and Selection*): O conjunto de representações de PLAs geradas pela atividade anterior é convertido em diagrama de classes para ser manipulado pelo arquiteto.

A OPLA-Tool[1] (*Optimization for PLA Tool*) [Freire et al. 2020] permite a aplicação da abordagem MOA4PLA. Os módulos que compõem a OPLA-Tool estão descritos na Figura 3, extraída de [Féderle et al. 2015]. O módulo OPLA-GUI apoia o arquiteto permitindo a escolha do algoritmo de busca que deverá ser utilizado bem como seus parâmetros, funções objetivo e operadores genéticos. Esse módulo ainda permite ao arquiteto a inserção de uma PLA que é a entrada para o processo de busca. O módulo OPLA-GUI utiliza serviços dos módulos: OPLA-Encoding, OPLA-Decoding e OPLA-Core. O OPLA-Encoding é responsável por transformar uma PLA em uma representação baseada em um metamodelo predefinido. O OPLA-Core é onde está implementado o algoritmo de busca, esse módulo recebe a representação criada pelo OPLA-Encoding e realiza o processo evolutivo do algoritmo e retorna um conjunto de PLAs. Finalmente o OPLA-Decoding recebe cada representação das PLAs e decodifica para uma versão legível no Papyrus (plugin do Eclipse para modelos UML).

A Figura 4 mostra a tela de configuração de execuções de um experimento na OPLA-Tool. Vários destes parâmetros podem ser alvo dos experimentos de configuração a serem estudados no presente projeto.

3. Trabalhos Relacionados

Nesta seção apresentam-se os trabalhos identificados em uma busca em fontes eletrônicas por trabalhos sobre calibração do NSGA-II. O estudo de Arcuri e Fraser

[1]https://github.com/otimizes/OPLA-Tool

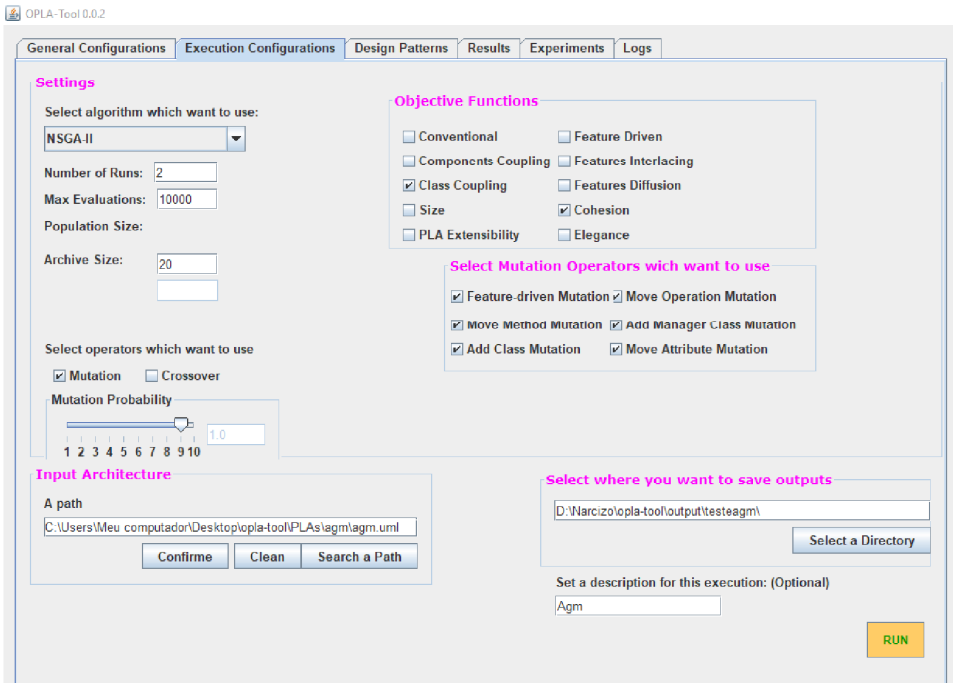


Figura 4. Interface de Configuração dos Parâmetros da OPLA-Tool.

[Arcuri and Fraser 2011], cujo foco é selecionar casos de testes para determinadas classes com o objetivo de maximizar um critério de cobertura enquanto busca-se minimizar o número de testes e sua duração, foram utilizados os parâmetros a seguir: Número de Avaliações de *Fitness* = 300.000, Tamanho da População: {4, 10, 50, 100, 200}, Probabilidade de Mutação: {0, 0.2, 0.5, 0.8, 1}.

Sayyad et al. [Sayyad et al. 2013] compararam os resultados de Arcuri e Fraser no contexto de otimização de modelos de características de LPS. Os experimentos foram realizados utilizando as seguintes configurações: o tamanho da população é formado pelos valores {10, 50, 100, 150, 200} e a probabilidade de mutação de acordo com o conjunto: {0, 0.5, 1, 1.5, 2}. Sayyad et al. [Sayyad et al. 2013] confirmaram as afirmações de Arcuri e Fraser de que configurações de parâmetros diferentes causam uma grande variação na performance, justificando a necessidade de estudar qual a melhor configuração dos parâmetros do NSGA-II para a otimização de projeto de PLA.

No contexto de projeto de PLA, o número de avaliações de *fitness* proposto em [Colanzi et al. 2014] estipulava um valor de 30.000 com populações de 100 indivíduos e probabilidade de mutação de 0.9. Guizzo [Guizzo 2014] propôs derivar e verificar outros dois valores, um dez vezes maior (300000) e outro com um décimo do valor (3000). Os resultados mostraram que o melhor valor depende da PLA dada como entrada.

4. Descrição do Estudo Experimental

Nesta seção descreve-se os procedimentos adotados para realizar o estudo experimental a fim de identificar os melhores valores para os parâmetros do algoritmo NSGA-II. Os parâmetros a serem calibrados e os respectivos valores a serem verificados foram definidos a partir de trabalhos relacionados. Para realizar os experimentos foi utilizada a ferramenta OPLA-Tool aplicando as diferentes combinações de parâmetros com o algoritmo NSGA-II e otimizando dois projetos de PLA.

Tabela 1. Configurações dos experimentos.

ID da Configuração	Tamanho da População	Número de Avaliações de Fitness	Taxa de Mutação
config1	10	3000	0,9
config2		30000	0,9
config3		300000	0,9
config4	50	3000	0,9
config5		30000	0,9
config6		300000	0,9
config7	100	3000	0,9
config8		30000	0,9
config9		300000	0,9
config10	200	3000	0,9
config11		30000	0,9
config12		300000	0,9
config13	melhor entre os anteriores	melhor entre os anteriores	0,8

Projetos de PLA. Dois projetos de PLA foram usados no estudo experimental: *Arcade Game Maker* (AGM) [SEI 2009] e *Mobile Media* [Contieri Jr et al. 2011]. AGM é uma LPS acadêmica criada pelo *Software Engineering Institute (SEI)* que contém 3 jogos de arcade: *Brickles*, *Bowling* e *Pong*. *Mobile Media* (MM) é uma LPS para gerenciamento de mídia (música, vídeo e foto) em dispositivos móveis.

Configuração dos parâmetros. Considerando os trabalhos relacionados (Seção 3), neste trabalho decidiu-se investigar a calibração dos seguintes parâmetros: Número de Avaliações de *Fitness*; Tamanho da População e Probabilidade de Mutação. No presente trabalho foi usada a mesma estratégia proposta em [Guizzo 2014] partindo de 30000 avaliações de *fitness*. O tamanho da população variou entre: 10, 50, 100 e 200. Dois valores foram executados para probabilidade de mutação: 0,8 e 0,9.

Nesse contexto, foram realizados experimentos de configuração de parâmetros para as PLAs AGM e MM utilizando os valores para cada parâmetro conforme discriminado na Tabela 1. A fim de minimizar o número de experimentos a serem executados, optou-se por executar os doze primeiros experimentos (Config 1 a Config12) utilizando o valor 0,9. Sabendo qual o melhor valor de tamanho de população e de avaliações de *fitness*, foi então averiguado o desempenho do algoritmo com taxa de mutação de 0,8 (Config 13). Cada experimento foi executado 15 vezes, sempre com as mesmas funções objetivo: *Cohesion* (mede a coesão relacional das classes), *Feature Modularization* (mede o nível de modularização das características da PLA) e *Class Coupling* (mede o acoplamento entre classes), cujas definições podem ser obtidas em [Verdecia et al. 2017].

Critério para Avaliação dos Resultados. Foram utilizados dois critérios de avaliação sendo eles o tempo de execução e o indicador de qualidade *hypervolume*. Com relação ao critério de tempo de execução quanto menor o tempo de execução melhor o desempenho. Todos os experimentos foram executados na mesma máquina, cujo processador é um Intel Core I5 7500 com 8GB de memória RAM, para assegurar que todas configurações disponibilizavam do mesmo recurso.

O indicador de qualidade *hypervolume* foi adotado para comparar os resultados obtidos pelos experimentos. Ele consiste no volume n-dimensional entre a estimativa de Fronteira de Pareto e um ponto de referência específico [Coello et al. 2007]. Quanto maior o valor do *hypervolume*, maior a área de cobertura refletindo em uma fronteira melhor. Os dados do hipervolume foram normalizados antes de aplicar testes estatísticos

para constatar qual a melhor configuração de parâmetros. O primeiro teste estatístico executado foi o teste de Shapiro-Wilk a fim de saber se uma determinada população tem uma distribuição normal ou não de acordo com o *p-value*. Como a distribuição dos resultados foi não normal, o teste de Kruskal-Wallis foi utilizado para averiguar a diferença estatística entre os resultados obtidos com confiança de 95% (*p-value* $\leq$0.05).

Ameaças à Validade. Uma das principais ameaças à validade do nosso estudo é o uso de uma única medida para avaliar os resultados. A escolha do indicador de qualidade hypervolume foi estratégica para mitigar este risco, uma vez que este indicador avalia tanto a convergência como a diversidade de soluções geradas pelo algoritmo [Li and Yao 2019]. Outra ameaça é o tamanho da amostra utilizada no estudo. Apesar dos resultados não poderem ser generalizados, as duas PLAs utilizadas na calibração têm diferentes características, especialmente no que tange à modularização de *features*, que afeta diretamente os resultados já que a MOA4PLA tem operadores e funções objetivo sensíveis a esta propriedade arquitetural, amenizando ameaça à validade externa.

5. Resultados e Análise

Esta seção apresenta os resultados dos experimentos realizados a fim de determinar a melhor configuração de parâmetros para o NSGA-II no contexto de otimização de projeto de PLA. A Tabela 2 contém o número de soluções não dominadas encontradas pelo NSGA-II após as 15 execuções. É possível identificar que os parâmetros que mais impactam na quantidade de soluções não dominadas são o número de avaliações de *fitness* e o tamanho da população já que config11 e config12 que apresentam os maiores números em relação a esses parâmetros obtiveram o maior número dessas soluções .

Tabela 2. Número de soluções não dominadas e o tempo de execução (“d” para dias, “h” para horas e “m” para minutos) de cada configuração.

	AGM		**MM**	
Configuração	**Número de Soluções**	**Tempo de Execução**	**Número de Soluções**	**Tempo de Execução**
config1	7	1h 3m	9	1h 4m
config2	6	17h 29m	8	8h 53m
config3	4	6d 11h 45m	7	4d 9h 9m
config4	4	2h 8m	6	2h 13m
config5	4	1d 13h 18m	19	13h 10m
config6	8	6d 16h 7m	10	4d 19h 54m
config7	8	2h 44m	8	2h 6m
config8	8	17h 42m	24	14h 20m
config9	7	5d 23h 41m	21	5d 23h 32m
config10	8	5h 46m	7	22m
config11	16	16h 39m	17	15h 13m
config12	19	7d 21h 36m	27	6d 14h 57m
config13	5	2h 25m	4	47m

Configurações com o mesmo tamanho de população foram subdivididas em grupos para a aplicação do teste de Kruskal-Wallis a fim de averiguar a diferença estatística de cada grupo. A Tabela 3 apresenta a formação dos grupos das doze primeiras configurações. Os resultados de *p-value* retornados pelo teste estatístico são apresentados na Tabela 4. O boxplot de cada configuração é apresentado nas Figuras 5 e 6. Para o indicador de qualidade *hypervolume*, quanto maior a mediana melhor o resultado. Nos grupos 1 e 3 da *Mobile Media*, o *p-value* não foi significativo para atestar diferença estatística, então nesses casos foi analisado qual configuração obteve a maior mediana de *hypervolume*, sendo então considerada a melhor configuração do grupo. Logo, observando os

Tabela 3. Agrupamento para os testes estatísticos.

Grupo	Configurações
Grupo 1	{config1, config2, config3}
Grupo 2	{config4, config5, config6}
Grupo 3	{config7, config8, config9}
Grupo 4	{config10, config11, config12}
Grupo 5	Melhores configurações dos 4 primeiros grupos **AGM:** {configs 1, 4, 7 e 10} **MM:** {configs 1, 4, 9 e 10}

Tabela 4. Resultados do teste de Kruskal-Wallis

Grupo de Configurações	AGM p-value	MM p-value
Grupo 1	4.90E-04	8.02E-02
Grupo 2	2.87E-04	6.31E-02
Grupo 3	3.86E-05	6.43E-01
Grupo 4	3.45E-04	1.40E-06
Grupo 5	1.01E-06	5.24E-04

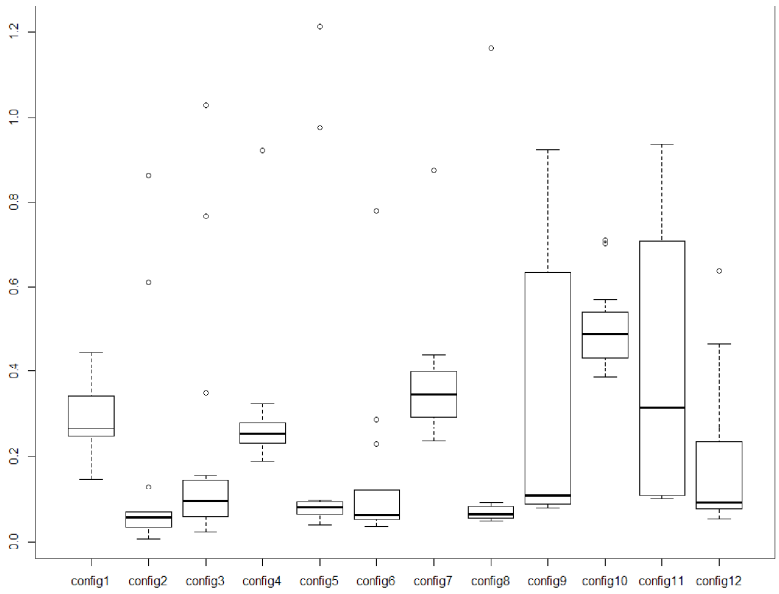


Figura 5. Boxplot referente ao teste de Kruskal-Wallis das configurações da AGM.

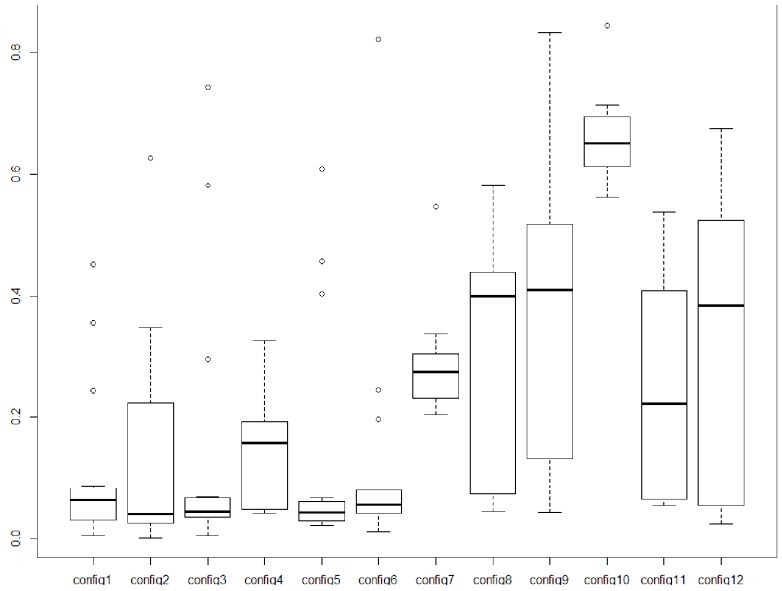


Figura 6. Boxplot referente ao teste de Kruskal-Wallis das configurações da MM.

boxplots e considerando os grupos, pode-se observar que as melhores configurações para a AGM são configs 1, 4, 7 e 10 e para a MM as configs 1, 4, 9 e 10.

Tabela 5. Resultado do teste de Kruskal-Wallis.

	Configurações	*p-value*	Melhor Configuração
AGM	config10, config13	5.20E-01	config10
MM	config10, config13	3.067E-06	config10

Tabela 6. Valores de mediana das configurações config10 e config13.

PLA	Configurações	Mediana	PLA	Configurações	Mediana
AGM	config10	0.49	**MM**	config10	0.64
	config13	0.45		config13	0.07

Após a aplicação do teste de Kruskal-Wallis nos Grupos de 1 a 4 (Tabela 4), foi possível identificar qual a melhor configuração de cada grupo e finalmente pode-se descobrir qual a melhor configuração geral aplicando o teste de Kruskal-Wallis nesse novo grupo formado (Grupo 5). Para a AGM, o Grupo 5 é formado pelas configurações 1, 4, 7 e 10 e para a *Mobile Media*, esse grupo é formado pelas configurações 1, 4, 9 e 10. Para o grupo 5 o teste estatístico atestou que há diferença estatística entre as configurações, como mostra o *p-value* (Tabela 4) e que a melhor configuração para as duas PLAs foi a config 10. Isso quer dizer que o melhor valor de tamanho de população é 200 e de número de avaliações de *fitness* é 3.000.

Como dito anteriormente após encontrada a melhor configuração (config10), ela seria testada variando o parâmetro Probabilidade de Mutação para 0.8. Deste modo, os parâmetros da config13 são: tamanho de população = 200, número de avaliações de *fitness* = 3.000 e probabilidade de mutação = 0.9. O número de soluções não dominadas e o tempo de execução da config13 pode ser visto na Tabela 2.

A Tabela 5 apresenta os resultados do teste estatístico para as configurações config10 e config13. Nesse caso, o teste de Kruskal-Wallis não apontou diferença estatística entre as configurações para a PLA AGM, porém a Tabela 6 mostra que o valor 0,9 de probabilidade de mutação (config10) é superior ao valor 0,8 (config13) em ambos os casos. No caso da AGM os valores foram equivalentes, porém config10 possui uma mediana maior e, por isso, foi considerada melhor.

A melhor configuração de parâmetros do NSGA-II para otimização de projeto de PLA é 3.000 avaliações de *fitness*, tamanho da população = 200 e probabilidade de mutação = 0,9. É interessante notar que a mesma configuração conseguiu os melhores resultados para ambas as PLAs e também foi uma das mais rápidas em termos de tempo de execução. Os parâmetros otimizados encontrados conseguem unir o melhor resultado de projeto de PLA com um tempo baixo de execução comparado com as configurações utilizadas anteriormente em [Colanzi et al. 2014] e [Guizzo 2014]; cada execução com 3.000 de avaliações de *fitness* levaram em média 1 hora para serem executadas enquanto as execuções com 30.000 levaram em média 1 dia para serem executadas.

6. Conclusão

O objetivo deste trabalho foi descobrir empiricamente as melhores configurações de parâmetros do algoritmo NSGA-II no contexto de otimização de projeto de PLA. Com base na literatura foi elaborado um conjunto de configurações de parâmetros focando em duas PLAs, a AGM e a *Mobile Media*, e dentre esse conjunto foi encontrada a configuração que encontra os melhores resultados de acordo com o indicador de qualidade *hypervolume*. Os resultados contribuem para obter melhores resultados em futuros expe-

rimentos de otimização de projetos de PLA e um menor tempo de execução. Pretende-se investigar se uma configuração com população maior pode obter melhores resultados.

Referências

[Arcuri and Fraser 2011] Arcuri, A. and Fraser, G. (2011). On parameter tuning in search based software engineering. In *Symposium on Search Based Software Engineering*, pages 33–47. Springer Berlin.

[Coello et al. 2007] Coello, C. A. C., Lamont, G. B., Van Veldhuizen, D. A., et al. (2007). *Evolutionary algorithms for solving multi-objective problems*. Springer Science & Business Media.

[Colanzi et al. 2014] Colanzi, T. E., Vergilio, S. R., Gimenes, I., and Oizumi, W. N. (2014). A search-based approach for software product line design. In *Proceedings of the 18th International Software Product Line Conference (SPLC'14)*, volume 1, pages 237–241.

[Contieri Jr et al. 2011] Contieri Jr, A. C., Correia, G. G., Colanzi, T. E., Gimenes, I. M., Oliveira Jr, E. A., Ferrari, S., Masiero, P. C., and Garcia, A. F. (2011). Extending uml components to develop software product-line architectures: Lessons learned. In *Proceedings of the 5th European Conference on Software Architecture (ECSA)*, pages 130–138.

[Deb et al. 2002] Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197.

[Féderle et al. 2015] Féderle, É. L., do Nascimento Ferreira, T., Colanzi, T. E., and Vergilio, S. R. (2015). OPLA-Tool: a support tool for search-based product line architecture design. In *Proc. of the 19th SPLC*, pages 370–373.

[Freire et al. 2020] Freire, W. M., Massago, M., Zavadski, A. C., Amaral, A. M. M. M., and Colanzi, T. E. (2020). OPLA-Tool v2.0: a tool for product line architecture design optimization. In *Proceedings of the 34th Brazilian Symposium on Software Engineering (SBES '20)*. ACM.

[Goldberg 1989] Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., USA, 1st edition.

[Guizzo 2014] Guizzo, G. (2014). Uso de padrões em projeto arquitetural baseado em busca de linha de produto de software. Dissertação de Mestrado, Pós-Graduação em Ciência da Computação, Universidade Federal do Paraná, Curitiba.

[Harman and Jones 2001] Harman, M. and Jones, B. F. (2001). Search-based software engineering. *Information and Software Technology*, 43(14):833 – 839.

[Li and Yao 2019] Li, M. and Yao, X. (2019). Quality evaluation of solution sets in multiobjective optimisation: A survey. *ACM Comput. Surv.*, 52(2).

[Linden et al. 2007] Linden, F. J. V. d., Schmid, K., and Rommes, E. (2007). *Software product lines in action: the best industrial practice in product line engineering*. Springer Science & Business Media.

[Sayyad et al. 2013] Sayyad, A. S., Goseva-Popstojanova, K., Menzies, T., and Ammar, H. (2013). On parameter tuning in search based software engineering: A replicated empirical study. In *2013 3rd International Workshop on Replication in Empirical Software Engineering Research*, pages 84–90.

[SEI 2009] SEI (2009). Arcade game maker pedagogical product line. https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=485941.

[Verdecia et al. 2017] Verdecia, Y. D., Colanzi, T. E., Vergilio, S. R., and Santos, M. C. B. (2017). An enhanced evaluation model for search-based product line architecture design. In *XX Ibero-American Conf. on Software Engineering (CIbSE2017)*, Buenos Aires.

Controlled Experiment in Crosscutting Concerns Identification from Software Requirements Specification

Guilherme Legramante[1], **Maicon Bernardino**[1], **João Pablo Silva**[2], **Elder Rodrigues**[1]

[1] Laboratory of Empirical Studies in Software Engineering
Universidade Federal do Pampa (UNIPAMPA)
Caixa Postal 15.064 – 97.546-550 – Alegrete – RS – Brazil

[2]Laboratory of Intelligent Software Engineering
Universidade Federal do Pampa (UNIPAMPA)
Caixa Postal 15.064 – 97.546-550 – Alegrete – RS – Brazil

guilhermelegramante@gmail.com,

{bernardino,joaosilva,elderrodrigues}@unipampa.edu.br

Abstract. *The concern is a group of requirements with the same purpose, which are known as crosscutting concerns when they are scattered and tangled in the system. Identifying and separating these concerns is a matter of huge importance to software maintainability and evolution. For this, some approaches are proposed. There are few experimental studies comparing and analyzing these approaches in detail. In this paper, our aim is to provide empirical evidence about two approaches in this context. We conducted a controlled experiment to compare the effectiveness of two approaches, which identify crosscutting concerns in the requirements specification, followed by an assessment of its perceived utility and ease of use. Our results indicate that, in this given context and planned scenario, there are significant differences between the effectiveness of* `ObasCId` *and* `Theme/Doc` *approaches.*

Resumo. *Interesse é um grupo de requisitos com o mesmo objetivo, conhecidos como interesses transversais quando dispersos e emaranhados no sistema. Identificar e separar esses interesse é uma questão de enorme importância para a manutenção e evolução do software. Para isso, algumas abordagens são propostas com essa finalidade. Há poucos estudos experimentais analisando essas abordagens em detalhes. Este artigo busca fornecer evidências empíricas sobre duas abordagens nesse contexto. Foi realizado um experimento controlado para comparar a eficácia de duas abordagens que identificam interesses transversais na especificação de requisitos, seguido de uma avaliação da sua utilidade percebida e facilidade de uso. Os resultados do experimento indicam que, nesse contexto e cenário planejado, existem diferenças significativas entre a efetividade das abordagens* `ObasCId` *e* `Theme/Doc`*.*

1. Introduction

Software requirements define the features and constraints of a given software product. These characteristics may contribute to the solution related to a real-world problem [Bourque et al. 2014]. The concern is a set of software requirements with the

same purpose[Parnas 1972]. Dijkstra [Dijkstra 1976] describes some concepts within this scope, such as concern separation, cohesion, and coupling in the context of Software Engineering by introducing notions of modularization. The Standard Glossary of Software Engineering Terminology [IEEE 1990] defines modularization as the degree of modifications required in software components that are caused by changes in another component. Therefore, a good modularization could bring benefits like reusability, comprehension, adaptability, maintainability [Meyer 1997]. For this reason, software engineers consider modularization as a key principle [Santanna 2008]. A cross-cutting concern is known as a scattered and tangled concern, by transversely intersecting the other software structures [Kiczales et al. 1997].

To decrease the effects caused by scattering and tangling of crosscutting concerns, the Software Engineer must identify them as early as possible in the software development life cycle [RESENDE 2007]. Nonetheless, this is a non-trivial activity, due to many concerns are implicit in the requirements document. According to Baniassad *et al.* [Baniassad and Clarke 2004], Software Engineers find it difficult to identify crosscutting concerns and two causes may be associated with these problems. The first one refers to a lack of understanding regarding the domain of software concerns. The second cause is related to the scarcity of appropriate resources to support software engineers software concerns identification and classification. Although there are several techniques and approaches for crosscutting concerns identification, there are few studies that perform empirical evidence on these approaches. So, in this paper, our objective is to provide empirical evidence about relevant studies in this context. To achieve the given objective, we conducted a controlled experiment to compare the effectiveness of the two approaches to identify crosscutting concerns in the requirements specification, followed by an assessment of its perceived utility and ease of use. We chose the selected approaches after the conduction of a Systematic Literature Review, which was the main aim of another study that we carried out before this experiment, and that we will not explain it in this paper.

We believe that the main contribution of our study to this field is the outcomes of the conducted empirical study, which were supported by well-defined and replicable protocol. In this context, the controlled experiment results provide empirical evidences that in a specific background there are significant differences between the effectiveness of the `ObasCId` [Parreira Jr. 2018] and `Theme/Doc` [Baniassad and Clarke 2004] approaches. Furthermore, we managed to accomplish a qualitative assessment that provided us with interesting and relevant traces of qualitative perceptions in the proposed study.

The paper is organized as it follows. Section 2 presents the controlled experiment, which is one of the core parts in the present study, and finally, in Section 3, show final remarks.

2. Experimental Study

In this experiment we followed the model by Wohlin [Wohlin et al. 2012].

2.1. Scope Definition

The experiment aims to compare two approaches for identifying crosscutting concerns from Requirements Engineering artifacts. Thereunto, main objective is described according to the GQM (Goal, Question, Metric) paradigm [Caldiera and Rombach 1994]:

For the purpose of: **comparison**, with respect to **effectiveness**, from the viewpoint of **researchers**, in the context of **Software Engineering undergraduate identifying crosscutting concerns in software requirement specification**

2.2. Experiment Context

Regarding the experiment context, we use the *in-vitro* approach since we perform the experiment in the laboratory under controlled conditions. The subjects were undergraduate students from Software Engineering, second year. The experiment compared the effectiveness of the `ObasCId` and `Theme/Doc` approaches for crosscutting concerns identification, addressing a real problem. The experiment was applied in a specific context, with Software Engineering students and two approaches of crosscutting concerns. However, general assumptions regarding the controlled experiment may be applied in other populations and approaches.

In order to implement those approaches, the software requirements specification of the Course Management System proposed by Baniassad *et al.* [Baniassad and Clarke 2004] and `ObasCId Tool` proposed by Junior and Penteado [Parreira Jr. 2018] were used as inputs. The software has a specification that allows identification and also already has the concerns identified and cataloged. Also, they are equivalent in complexity level as a means of having the same number of concerns.

2.3. Research Hypotheses

The mainstay for the statistical analysis of an experiment is the hypothesis test. The hypothesis is statistically formulated and the data was collected during the experiment conduction to, if possible, reject the hypothesis. If we could reject the hypothesis, We can infer some conclusions.

For hypotheses formulation, we analyze the following research question:

RQ. Which of the approaches for identifying crosscutting concerns in the software requirements specification is most effective?

The experiment has the following hypotheses [1]:

Null hypothesis, $H_0 = E_{OC} = E_{TD}$**:** The effectiveness is the same when using `ObasCId` e `Theme/Doc` approaches.

Alternative hypothesis, $H_1 : E_{OC} > E_{TD}$**:** The effectiveness is lower when using `Theme/Doc` than when using `ObasCId`.

Alternative hypothesis, $H_2 : E_{TD} > E_{OC}$**:** The effectiveness is lower when using `ObasCId` than when using `Theme/Doc`.

Furthermore, it is possible to evaluate the results through precision and recall metrics. Hence, Precision is the fraction of the documents already examined that are relevant, and Recall is the fraction of the relevant documents observed among the examined documents [Monteiro 2017]. Thus, to calculate Precision and Recall metrics, some variables are used:

True Positives (TP): Concerns correctly identified using the approach.

[1] E_{OC}/: ObasCId Effectiveness and E_{TD}: Theme/Doc Effectiveness

False Positives (FP): Concerns incorrectly identified using the approach.

False Negatives (FN): Concerns correctly not identified using the approach.

In our context, the effectiveness was inferred from the calculation of *F-measure* or *F-Score*, which is the harmonic mean of precision and recall metrics.

- *Precision (P):* $P = \frac{TP}{TP+FP}$
- *Recall (R):* $R = \frac{TP}{TP+FN}$
- *F-measure (F):* $F = 2 * ((P * R)/(P + R))$

2.4. Subjects Selection

We selected the experiment subjects by non-probabilistic sampling, where there is a deliberate choice, for convenience. Thirty-six (36) undergraduate students from the second year in Software Engineering course at University participated in the controlled experiment. They were students enrolled in Software Verification and Validation disciplines, Experimental Software Engineering, and Problem Solving VI. The subjects were chosen because they composed a representative sample of the software engineers population since they have knowledge related to the area and already perform activities related to software engineering even though in academia.

2.5. Experiment Design

The four general principles for the experiment are as follow:

Standard Design Types: The experiment applied the pattern of *One Factor with Two Treatments*. The Factor is the approach for identifying crosscutting concerns and Treatments are the `ObasCId` and `Theme/Doc`.

Blocking: Thus, we applied a leveling questionnaire so that it was possible to classify the subjects.

Balancing: We divided the selected subjects into two homogeneous groups.

Randomization: We allocated the subjects randomly placed into each group and each approach.

2.6. Instrumentation

The experience level of the subjects was obtained through a questionnaire[2], in order to assess the knowledge level in the area that is the object of the study. With this information it was possible to identify a subject profile.

Objects: among the objects used in the experiment are the requirements documentation of the software `ObasCId` Tool, of the Course Management System and the training guidelines of the subjects. The training sessions were available in videos divided in 3 parts. The first training session addressed main concepts on Requirements Engineering and Aspect-Oriented Requirements Engineering in order to level the subjects' knowledge. The second and third training sessions explain the `Theme/Doc` and `ObasCId` approaches. We provided a printed document containing all the material to each student during the video lessons and was also a possibility to be used by them. The

[2]Questionnaire: `https://bit.ly/2wRInwF`

requirements documentation was properly prepared for the identification of software concerns according to the approaches used in the task. Since participation in the experiment was voluntary, a consent form was presented to the subjects, formalize their agreement to participate in the experiment. Requirements templates were also elaborated for listing the concerns that should be identified by the subjects.

Guidelines: We conducted three sessions of the experiment divided into two stages, training and execution. During the training stage, the subjects received instructions on Software Concerns and Requirements Engineering, then instructions on the approaches to be performed. We defined randomly the order of the instructions on the approaches and remained the same throughout all sessions of the experiment.

Measures: The quantitative metrics encompass relevant elements, selected elements, true positives, false positives, and false negatives. All subjects performed the same tasks under the same conditions, except when sessions were on different days, but with no significant differences.

2.7. Operation

The first activity was to contact the course professors for the sessions. After their agreement, we applied a questionnaire to assess the knowledge level of research subjects. From this, we classified the students into two levels: basic and intermediate. We sent all experiment artifacts for validation with a specialist, Aspect-Oriented Requirements Engineering Ph.D. We made all the corrections suggested by the specialist. The experiment execution has three sessions that occurred on consecutive days, September 11, 12, and 13, 2018, respectively. For this stage, we invited the students of the Software Verification and Validation - V&V (Session 1), Experimental Software Engineering - ESE (Session 2), and Problem Solving VI - PSVI (Session 3) courses to attend. It stands out that the disciplines mentioned are part of the Software Engineering curriculum of the university where the research took place.

Table 1 displays the information about the experiment sessions. The experiment sessions are in the first column, followed by the executed disciplines, the number of subjects that participated with the blocks by basic and intermediate levels, and the date of execution. In Session 1, two participants had the results disregarded, as long as they did not follow the recommendations regarding the non-use of electronic devices and did not avoid communicating with the other subjects during the experiment execution.

Table 1. Experiment Sessions.

Session	Course	Subjects	BL	IL	Date
1	V&V	20	18	2	11/09/2018
2	ESE	12	4	8	12/09/2018
3	PSVI	4	2	2	13/09/2018
Total		**36**	**24**	**12**	

BL: Basic Level, **IL**: Intermediate Level

The first activity Consent Term[3] provision for experiment execution. The subjects who signed the term were selected to participate in the other activities. We informed the

[3]Consent Term and study artifacts available in: `https://bit.ly/2wRInwF`

subjects about the ”non-evaluative” character of the experiment, *i.e.* that the Subjects were not under evaluation. Afterward, we included the subjects in two homogeneous groups, A and B, according to the result of the questionnaire previously applied. In the first stage, we show the video Software Concerns Separation training. This activity lasted approximately 12 minutes. Then, we display the `Theme/Doc` approach training, after a random choice. This activity lasted for approximately 9 minutes. The last activity of the training was the presentation of the video about the `ObasCId` approach, which lasted approximately 12 minutes. It is important to emphasize that the difference in duration between the training sessions of the approaches is due to the differences between them because the `ObasCId` approach requires more activities to be carried out to identify the software concerns and for this reason, it has impacted training duration.

At the end of the training phase, the execution stage began. For this, the subjects received the necessary artifacts to perform the tasks. The main activity to be undertaken was to identify software concerns in the software requirement specification, using the approaches. In the former part of the experiment execution, Group A should use the `ObasCId` approach and Group B, the `Theme/Doc` approach. In the latter part of the experiment execution, Group A used the `Theme/Doc` approach and or Group B to `ObasCId`. Therefore, we provided the material presented in the training sessions to the subjects in a printed document. Also, Group A received the software requirement specification (SRS) of the Course Management System and the Software Concerns Catalog. For Group B, the SRS of the `OC Tool` was also delivered.In the latter part of the experiment execution, Group A received the SRS of the OC Tool and Group B, the SRS of the Course Management System plus the Software Concerns Catalog. The subjects answered the questionnaire about the perception of its utility and ease of use after performing the concerns identification. The three sessions occurred in the same way, following the same activities in the same execution order.

2.8. Threats to Validity

According to Cook [Cook and Campbell 1979], threats are categorized into: Conclusion, Internal, Construct, and External Validity.

2.8.1. Conclusion Validity

It encompasses questions related to results analysis, *i.e.* whether the conclusions reached are correct. The Paired T-test was used to compare the measured values. The *Shapiro-Wilk* test was applied to verify the normality of the data. The fact that there is no relationship between the approaches analyzed and our research, that help us to mitigate this threat. Objective measurements that do not depend on the subjective judgment will be made because the effectiveness of the approaches will be measured. The qualitative metrics collected to verify the perceived utility and ease of use only complement the quantitative metrics, not being the main objective of measurement. We standardize procedures for all subjects. The sessions of the experiment followed the same execution order, having the same artifacts and duration time. The experiment was performed in a controlled environment, avoiding external interactions to experiment, such as interruptions, the exit of the experiment environment, access to electronic devices, etc. We advised the participants to

ban the use of electronic devices and parallel conversations during the experiment. No participant was allowed to leave the experiment environment during the execution.

2.8.2. Internal Validity

We performed the experiment at a time when students were not too overwhelmed with semester projects, papers, or exams. We validated all the artifacts by a Ph.D. specialist in the Aspect-Oriented Requirements Engineering area. We do not select participants with any type of involvement in similar experiments, even the sample used in the pilot experiment was discarded for the real experiment.

2.8.3. Construct Validity

As the experiment follows a paired design, all subjects performed the two treatments. However, this fact was not a threat because there was no learning between the approaches execution. For learning verification, we analyzed experiment data as a random design and, the results remained similar. Hence, we verified that there was no evidence of learning between the execution of the approaches. We do not inform the subjects were about the experiment design details. Also, we inform that the experiment would have no impact on students' grades and that they were not under evaluation but participating in an experiment. Subjects were not provided detailed information about the experiment, only the instructions were given on the tasks to be performed.

2.8.4. External Validity

We performed the study with Software Engineering students who represent a significant sample for the area since the primary responsibility for identifying software concerns in a project is the Software Engineer. However, the fact of carrying out the experiment with students is considered a threat, which could not be mitigated. We used documentation of traditional software requirement specifications found in the industry. Furthermore, the artifacts were validated with a Ph.D. specialist in the area.

2.9. Results

After the subjects of the experiment performed the identification of concerns activities, the correction stage was passed, where concerns identified by the subjects and concerns presented in the template were confronted. The correction was conducted in pairs as a means to attaining a consensus. Figure 1 displays the results of the experiment. The red boxes refer to the `ObasCId` approach and the blue boxes show the results of the `Theme/Doc` approach and also show the results according to the levels: basic and intermediate. It is possible to notice that essentially the `Theme/Doc` approach obtained better rates for the *F-measure*, regardless of the blocking performed. However, the difference between treatments is lower among subjects classified at the intermediate level. The `ObasCId` approach has a smaller variability compared to `Theme/Doc`. The lower limit for the basic level is quite similar, 0.20 for `ObasCId` and 0.22 for `Theme/Doc`, at the intermediate level the difference is larger, 0.33 for `ObasCId` and 0.72 for `Theme/Doc`.

The group that presents the greatest variability between the first and third quartile is the basic level in regard to `Theme/Doc` approach. The least dispersed group is the intermediate level that executed `Theme/Doc` approach, amidst a difference of 0.22 between the first and third quartile. This group also presented the highest value for the lower limit, 0.72. A further relevant aspect is that at all levels and treatments the upper limit was the maximum possible value, which means that in all groups there were subjects who were able to identify all the interests presented in the software under analysis. Based on the data reported, we verified that both treatments obtained better results with groups with greater experience in Requirements Engineering and Aspect-Oriented Requirements Engineering, and the treatments did present significant differences to their effectiveness.

2.10. Hypothesis Testing

According to [Graybill et al. 1998], to be able to accept or reject a particular hypothesis, it is necessary to perform a procedure called Hypothesis Test or Significance Test. The first procedure was the analysis of experiment samples to verify the normality among data distribution. For this task, we used the *Shapiro-Wilk* test, in which one tries to reject the null hypothesis to infer the normality of the data [Wohlin et al. 2012]. Table 2 displays the values of W and *p-value* obtained with the *Shapiro-Wilk* test for the experiment. As p-value = 0.05 and $W_{(calculated)}$ ¡ $W_{(0,05;\ 10)}$, it is possible to state with a significance level of 5% that the sample comes from a normal population, that is, there is 95% chance of the sample being normal.

Table 2. Shapiro-Wilk Test Results

	Session 1	Session 2	Session 3	Total
W	0.9711	0.9616	0.8935	0.9279
p-value	0.7789	0.8064	0.3999	0.2013

Based on the results of the normality test, we have chosen to perform the *Paired Sample T-Test* to check the H_0 of the experiment. Applying this test it was obtained a *p-value* = 2.001e-05 (*p-value* < 0.05), *i.e.* it was possible to reject H_0 and state that there are differences between effectiveness of the `ObasCId` and `Theme/Doc` approaches. According to the data displayed in the Figure 1, we may accept that the H_2: `Theme/Doc` approach has more effectiveness than the `ObasCId` approach. Presented results were obtained for a given sample from a specific population under a given context. Hence, their data do not depict a representative sample and cannot be generalized.

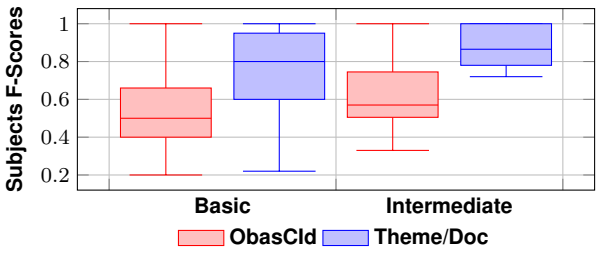


Figure 1. Experiment results.

2.11. Perceived Utility and Ease of Use

Qualitative metrics were collected regarding the perceived utility and ease of use of the approaches. These metrics were not included in the protocol presented in Section 2, because they are subjective. The reference model used to formulate the questions presented in the questionnaire was the TAM (Technology Text Acceptance) Model, proposed by Davis [Davis 1993]. The questionnaire was organized in 12 questions, 6 for each of the approaches under analysis. Figure 2 shows questionnaire results.

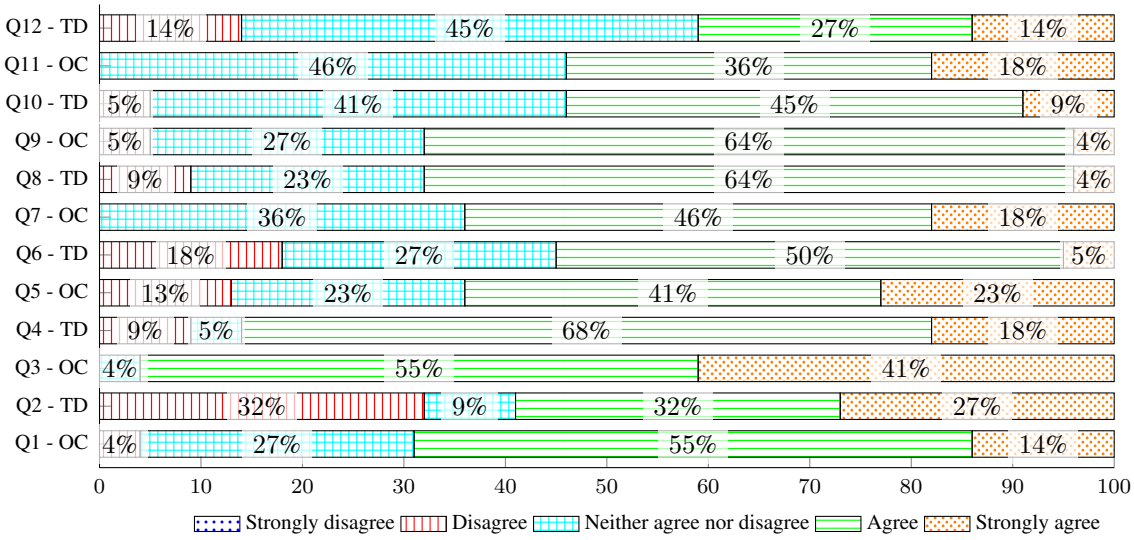


Figure 2. Questionary Results

The following are the questions that we use: - The approach is easy to use; - The approach is useful for identifying and classifying crosscutting concerns in requirements documents; - I enjoyed working with the approach; - Using the approach can increase my performance during work related to the identification and classification of crosscutting concerns; - The approach produces the results I expect from an approach to identifying and classifying crosscutting concerns; - I will recommend using the approach.

To answer the questionnaire the subjects should choose one of 5 possible answers: Strongly Disagree, Disagree, Neither Agree nor Disagree, Agree, Strongly Agree. In general, the `ObasCId` approach presented better results because it shows a higher percentage of responses that obtained total agreement, that is, answers marked with "Strongly Agree". Both approaches were well accepted, according to questions Q6 and Q12, which refer to using the approaches recommendation. The `Theme/Doc` approach had better results in the ease of use questions, questions Q1 and Q7, obtaining 27.3% of responses *Strongly Agree* versus 13.6% of `ObasCId`. In questions Q5 and Q11, regarding the expected results, although the `Theme/Doc` approach obtains a higher result of "Strongly Agree" answers (9.1%), it had an expressive number of subjects that remained neutral, answering "I Neither Agree nor Disagree" and the fact that 63.6% of subjects respond "I Agree" to the same question, but related to the approach `ObasCId`, may represent that the latter had better expected results.

3. Final Remarks

Some general considerations about obtained results. With the Theme/Doc approach, the crosscutting concern participants noticed greater difficulty in identifying were the ones related to usability, a non-functional requirement. Therefore, in our experiment, the Theme/Doc approach proved to be more effective for identifying concerns, not necessarily crosscutting concerns. The difficulty in executing the Theme/Doc approach may have been part of Action View modeling, which requires a certain understanding and special attention to its creation to be correctly modeled.

Regarding the ObasCId approach, the crosscutting concern with a lower number of correct answers of the participants was related to usability in the OCT requirements and student selection in CMS specification. The greatest difficulty encountered in implementing this approach was related to support from the concerns catalog since it induces some participants to mark false positives, as the catalog contains some concerns that might not necessarily be mapped.

Based on our experiment results, we believe that Theme/Doc approaches are due to some characteristics that make it slightly simpler to be applied than the ObasCId approach. Perhaps the catalog of concerns, available at ObasCId, despite being useful, induces the participant not yet familiar with the approach, which was our case, to elicit some false positives. However, we emphasize that for our context, with our representative portion of the selected population, these are the comparative evidence found. Moreover, we did not intend to refute or replicate the experiment carried out by Junior and Penteado [Parreira Jr. 2018], since we have different protocols and objectives.

The general objective of this paper was to provide empirical evidence about relevant studies in this context. `ObasCId` and `Theme/Doc` approaches were evaluated through an experiment, as means to measure and compare their effectiveness. The main contributions of this work are the outcomes of the empirical study conducted, which were supported by well-defined and replicable protocol. Obtained empirical evidence illustrates that, in a given scenario and context, there are significant differences between selected approaches effectiveness The results obtained through qualitative metrics are related to perceived utility and ease of use of approaches listed above, which ones might also be elicited as a contribution to this research area. To the best of our knowledge, no studies were found in the Literature, so far, performing these measurements given the presented context and scenario. As further research, we would suggest extending the results of the controlled experiment performed with a different sample, for instance, by replicating it in an industrial environment, since we applied our experiment in an academic environment. It is also possible to conduct empirical studies in the area of crosscutting concerns in other stages of the software development life cycle.

References

Baniassad, E. and Clarke, S. (2004). Finding aspects in requirements with theme/doc. *Early Aspects: Aspect-Oriented Requirements Engineering and Architecture Design*, pages 15–22.

Bourque, P., Fairley, R. E., et al. (2014). *Guide to the software engineering body of knowledge (SWEBOK (R)): Version 3.0*. IEEE Computer Society Press.

Caldiera, V. R. B.-G. and Rombach, H. D. (1994). Goal question metric paradigm. *Encyclopedia of software engineering*, 1:528–532.

Cook, T. and Campbell, D. (1979). *Quasi-Experimentation: Design and Analysis Issues for Field Settings*. Houghton Mifflin.

Davis, F. D. (1993). User acceptance of information technology: system characteristics, user perceptions and behavioral impacts. 38(3):475 – 487.

Dijkstra, E. W. (1976). *A Discipline of Programming*. Prentice Hall.

Graybill, F., Iyer, H., and Burdick, R. (1998). *Applied Statistics: A First Course in Inference*. Data Warehousing Institute Series from. Prentice Hall.

IEEE (1990). Standard Glossary of Software Engineering Terminology.

Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Lopes, C. V., Loingtier, J.-M., and Irwin, J. (1997). *Aspect-oriented programming*. Springer Berlin Heidelberg.

Meyer, B. (1997). *Object-oriented Software Construction*. Prentice Hall.

Monteiro, S. D. (2017). Sistemas de recuperação da informação e o conceito de relevância nos mecanismos de busca: semântica e significação.

Parnas, D. L. (1972). *On the criteria to be used in decomposing systems into modules*. Communications of ACM.

Parreira Jr., R. A. D. P. (2018). ObasCId(-Tool): an ontologically based approach for concern identification and classification and its computational support. *Journal of the Brazilian Computer Society*, 24(1):3.

RESENDE, A. M. (2007). *Um método para Identificação e Definição de Aspectos Iniciais. 209f.* PhD thesis, Tese de Doutorado-Instituto Tecnológico de Aeronáutica, São José dos Campos.

Santanna, C. (2008). *On the Modularity of Aspect-Oriented Design: a concern-driven measurement approach*. PhD thesis, Centre for Telematics and Information Technology, University of Twente, Enschede, The Netherlands.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., and Wesslén, A. (2012). *Experimentation in Software Engineering*. Springer, Norwell, MA, USA.

Business Intelligence com Qlik Sense aplicado ao Radar Saúde

Leander Alves[1], Joyce Aline Oliveira[1], Marcelo Takao[2], Jeison Santos[3], Vanice Cunha[3], Cristiano Maciel[3]

[1]Faculdade de Engenharias
Universidade Federal de Mato Grosso (UFMT), Cuiabá, MT – Brasil

[2]Secretaria de Controle Externo
Tribunal de Contas do Estado do Mato Grosso, Cuiabá, MT – Brazil

[3]Instituto de Computação
Universidade Federal de Mato Grosso (UFMT), Cuiabá, MT – Brasil

leanderlvda@gmail.com,{joyceoliveira,jeison}@ufmt.br,
mttanaka@tce.mt.gov.br, vanice@ic.ufmt.br, crismac@gmail.com

Abstract. *This paper reports the adoption of Qlik Sense in the Radar Saúde construction process, software by which TCE-MT publishes graphics and indicators on health in the state of Mato Grosso in order to guarantee transparency and accountability to the citizen. The results showed that Qlik Sense is easy to use and has advantages in terms of simplicity, quality, time reduction and increased agility in the implementation of the application and data management. The article helps to increase the body of knowledge related to Qlik Sense. In practical terms, the article is useful for developers looking for more efficient ways to manage data.*

Resumo. *Este artigo relata a adoção do Qlik Sense no processo de construção do Radar Saúde, software pelo qual o TCE-MT divulga gráficos e indicadores sobre a saúde no estado do Mato-Grosso de modo a garantir a transparência e a prestação de contas ao cidadão. Os resultados mostraram que o Qlik Sense é de fácil utilização e apresenta vantagens em termos de simplicidade, qualidade, diminuição de tempo e aumento de agilidade na implementação da aplicação e gerenciamento de dados. O artigo auxilia no aumento do corpo de conhecimento relacionado a Qlik Sense. Em termos práticos, o artigo é útil para desenvolvedores que buscam formas mais eficientes para gerenciar dados.*

1. Introdução

No contexto de organizações públicas, o cidadão é o principal cliente que, pela Constituição, deve se beneficiar de serviços que garantam a sua segurança, locomoção, saúde, inclusão, enfim o seu bem estar enquanto membro da sociedade. Portanto, o setor público deve visar o atendimento das necessidades do cidadão com transparência, integridade e eficiência, além de garantir a prestação de contas. Tanto a transparência quanto a prestação de contas podem ser eficientemente geridas e reportadas por meio do

uso de recursos de BI (Business Intelligence). BI consiste em um conjunto de técnicas e ferramentas através das quais é possível coletar dados brutos, organiza-los e extrair informações úteis para, a partir de metodologias específicas, mostrar aos tomadores de decisão a situação atual do negócio ou organização e induzi-lo à ação (Vashisht e Dharia, 2020). O principal objetivo do BI é permitir a rápida compreensão e interpretação de grandes quantidades de dados (Kalaiarasan et al, 2020). O Qlik Sense é um dos softwares de BI disponíveis no mercado que permite manipular gráficos, além do controle e manuseio destes por meio do gerenciador de dados em tabelas ou do editor de script (Mora, 2020). Este artigo relata a adoção do Qlik Sense na implementação do Radar Saúde no TCE-MT (Tribunal de Contas do Estado de Mato Grosso). O Radar Saúde visa a transparência e a prestação de contas das Unidades Básicas de Saúde e de seus profissionais em todos os municípios do Mato Grosso por meio da apresentação de dashboards atuais e de fácil visualização. O artigo também relata a experiência dos principais desenvolvedores envolvidos no projeto quanto ao aprendizado e manipulação do Qlik Sense. A ferramenta Qlik Sense foi escolhida, diante de muitas outras, por ser de fácil uso, líder no mercado e pioneira no ambiente de BI, além de possuir coleções de "itens reutilizáveis de dados. A nível teórico, o artigo é útil por aumentar o corpo de conhecimento da área. A nível prático, os resultados são úteis para desenvolvedores que queiram adotar o Qlik Sense e que buscam orientações sobre o tema.

2. Qlik Sense

O Qlik Sense é uma ferramenta líder de mercado, sendo uma pioneira no ambiente de BI. Desde sua primeira versão, a chave do negócio foi integrar diferentes fontes de dados para a geração automática de relatórios Vashisht e Dharia (2020). O Qlik Sense permite a visualização de dados de modo a ajudar o usuário a obter insights de forma intuitiva funcionando de modo associativo por meio da criação e detalhamento de modelos de dados (Mora, 2020). Segundo Salas Urbano e Bruce (2020), o Qlik Sense foi criado pela Qlik, que é uma empresa fornecedora de software fundada em 1993 em Lund, Suécia. A Qlik é especializada em visualização de dados, integração de dados e Business Intelligence orientado ao usuário (Salas Urbano e Bruce, 2020).

As edições disponíveis do sistema Qlik Sense são: Desktop, ferramenta direcionada para usuários com relatórios e painéis interativos e personalizados de várias fontes de dados; Enterprise, ferramenta paga e voltada para empresas com aplicativos de análises personalizadas para suportar processos de negócios ou casos de uso mais específicos e Sense Cloud, versão hospedada online do Qlik Sense e que possui níveis gratuitos e pagos (Georg, 2017). O software possui um mecanismo de análise que ajuda o usuário a classificar os dados e escolher o que pode ser interessante para eles analisarem. O Qlik Sense tem mais de quinze diferentes gráficos para ajudar o usuário a visualizar seus dados: gráficos de barra, gráficos de pizza, mapas de árvore, gráficos de dispersão, entre outros (Salas Urbano e Bruce, 2020). O usuário pode com o simples recurso de arrastar e soltar criar uma página de visualização que corresponde às necessidades do usuário. Depois de os gráficos desejados estarem em seus devidos lugares, o usuário pode interagir com eles e ver suas reações em tempo real (Georg, 2017).

3. Trabalhos relacionados

Business intelligence - BI é um Sistema de Suporte à Decisão (Decision Support Systems – DSS) orientado a dados que combina coleta de dados, armazenamento de dados e

gerenciamento de conhecimento com análise para fornecer entrada para o processo de decisão. A inteligência de negócios enfatiza a análise de grandes volumes de dados sobre a empresa e suas operações. Inclui inteligência competitiva (monitoramento de concorrentes) como um subconjunto (Negash e Gray – 2008).

BI direcionado à obtenção dos objetivos estratégicos, utilizando a Governança Corporativa como instrumento de controle. Com o auxílio de ferramentas específicas e com o uso das técnicas de análise léxica, análise de conteúdo e análise de discurso contribuem para a obtenção dos objetivos estratégicos da organização. A ferramenta BI fornece características como: Confiabilidade, Consistência, Disponibilidade e Capacidade Preditiva (Flores, 2011).

Nunes, E. D. (2016) aponta como tendência e realiza uma análise comparativa entre três soluções para SSBI desenvolvidas por fabricantes líderes em Plataformas Analíticas e de Business Intelligence, de modo a identificar como o mercado vem se adaptando a esta nova necessidade e quais as características mais comuns nestas ferramentas, visando permitir que os profissionais da área de negócios possam criar suas próprias análises sem a necessidade de buscar apoio do Setor de Tecnologia da Informação (TI) presente nas organizações. Afirmando que, por meio do Qlik Sense é possível criar uma série de Aplicativos, que consistem em coleções de "itens reutilizáveis de dados (medidas, dimensões e visualizações).

Mungard et al (2017) desenvolveu uma aplicação de inteligência de negócios para análise dos dados do censo da educação superior, permitindo apoio aos gestores das instituições no desenvolvimento de atividades de tomada de decisões com base em dados, esta aplicação foi construída usando a ferramenta Qlik Sense Desktop, por se tratar de uma ferramenta livre, de fácil uso, além de suportar acesso a grandes volumes de dados.

Georg (2017) apresentou uma análise e a descrição do desenvolvimento de um sistema de BI para uma empresa do ramo de vendas online, utilizando a ferramenta Qlik Sense Desktop, por possuir uma interface que pode ser editada conforme necessidade do gestor responsável pela área, com atualização periódica, conforme a necessidade e facilitar a análise e processamento dos dados para gerar informações úteis.

Musskopf (2018) apresentou em seu trabalho as 40 melhores e mais utilizadas ferramentas de BI, utilizou como critério a repetição de aparecimento em ferramentas de pesquisa, as avaliações positivas dos clientes e quantidades de empresas que utilizam a ferramenta. A ferramenta Qlik foi apontada como uma das que possuem a característica de auxiliar qualquer empresa em qualquer ramo, pois não necessita de funcionalidades específicas, visto que sua principal função é a criação de painéis de indicadores, e esses são similares em qualquer organização. A autora também apresenta em seu estudo que muitas empresas que utilizam o Qlik, também utilizam o Microsoft Excel e o Microsoft Power BI.

De Castro e Silva (2018) apresentou a relevância da – BI como processo na tomada de decisões das empresas em seus resultados, adequando-as a realidade de um mercado globalizado, neste contexto os autores fazem uma análise comparativa entre as três ferramentas líderes no mercado, de acordo com os resultados identificados pelo Quadrante Mágico de Gartner, no ano 2018, e mostram que o Qlik é uma das ferramentas de BI líderes do mercado, oferecendo melhores resultados e competitividade.

4. Metodologia

O TCE-MT, com intuito de agregar informações úteis e exibi-las de maneira intuitiva ao cidadão, para fins de transparência e prestação de contas, lançou ao final de 2019 o Portal Cidadão. Este portal utiliza como ferramenta principal de gerenciamento de dados o Qlik Sense, um software de BI (*Business Intelligence*) que facilita a transformação de dados e exposição em gráficos e tabelas. Um dos módulos do Portal Cidadão é o Radar Saúde criado com o propósito de aumentar a transparência aos cidadãos sobre as Unidades Básicas de Saúde e seus profissionais em cada município do estado de Mato Grosso. Neste artigo, é relatada a experiência da implementação deste módulo com o Qlik Sense por meio de uma pesquisa-ação. Este método foi escolhido por possibilitar a modificação do ambiente pelo pesquisador não só com aspectos de algumas variantes da pesquisa observacional, mas é uma forma de pesquisa qualitativa que busca modificar o ambiente em que está sendo estudado através da ação de uma ferramenta (Wainer, 2007), (Torres et al, 2014). O resultado da pesquisa-ação foi descrito seguindo as etapas diagnóstico, planejamento da ação, tomada da ação na implantação da solução, avaliação dos resultados e relato do aprendizado a adaptação das teorias que foram usadas para formular a solução, tendo em vista a avaliação e a descrição (da Silveira Farias e Manzanal, 2013), (de Souza, Vasconcelos, 2010). Para a implementação foi necessário criar um processo ETL (do inglês, *Extract Transform Load*). O processo de ETL foi criado anteriormente pela STI (Secretaria de Tecnologia da Informação, do TCE-MT), portanto não são conhecidos detalhes sobre a sua definição. Mas, o processo padrão consiste em enviar o arquivo que deve ser inserido no sistema ao STI que o converte em QVD e o grava no banco de dados. Após este procedimento, o arquivo é disponibilizado em acesso de arquivos compartilhados.

A experiência da equipe de desenvolvimento com o Qlik Sense começou com a STI fazendo uma apresentação sobre esta ferramenta para a equipe de desenvolvimento. Após isso, a equipe se aprofundou no estudo sobre o Qlik Sense unindo o conhecimento obtido na disciplina de Banco de Dados, cursada na UFMT, com informações e materiais disponibilizados pelo TCE-MT. Antes de utilizar o Qlik Sense, a equipe utilizava o Oracle BI para gerar gráficos. Entretanto, por falta de compatibilidade com outras aplicações do TCE-MT foi necessário adotar outra ferramenta para gerenciamento de dados e geração de gráficos. O Qlik Sense foi escolhido pela facilidade de uso. A princípio eram utilizados o Excel para realizar as edições dos dados para a inserção no Qlik Sense. Posteriormente, os dados foram editados por meio de SQL (*Structured Query Language*) e VBA. Com o aumento da curva de aprendizado, passou-se a utilizar AQL (Associative Query Language), que é a tecnologia usada no Qlik Sense para associar dados. Dentro das aplicações foram criados gráficos e visões, que consistem em poderosos recursos para simplificar buscas. O resultado da implementação pode ser visualizado em: http://radarsaude.tce.mt.gov.br/.

Não é possível afirmar que a ferramenta pode ser viável para ser implantada em qualquer base de dados, mas nas bases de dados, que são mais conhecidas como: Oracle, BigQuery, Salesforce, Azure, e Amazon, são aceitas. Também são aceitos: entradas manuais, nas quais são desenvolvidas tabelas dentro do sistema; e anexados arquivos onde podem-se inserir diretamente de um arquivo tabulado “. XLSX /.CSV” (Excel), por exemplo. Na implementação, é possível realizar a integração de diferentes bases facilmente, por exemplo: na aplicação do radar saúde foram realizadas algumas inserções

pelo servidor já em QVD, algumas .CSV, alterações manuais e aplicações externas como Mapa de geolocalização.

Após a implementação, três dos principais desenvolvedores envolvidos no uso do Qlik Sense foram questionados quanto a facilidade do uso, os benefícios e as desvantagens do Qlik Sense, além de dicas úteis para potencializar o aprendizado sobre a referida ferramenta. Estes desenvolvedores, caracterizados como Desenvolvedor 1 (D1), Desenvolvedor 2 (D2) e Desenvolvedor 3 (D3) foram escolhidos pela imersão e experiência obtida no uso do Qlik Sense na implementação do Radar Saúde. As perguntas foram elaboradas no google forms e estão disponíveis no link: https://forms.gle/pzsuL7K5rTUL1RNu5. As respostas foram analisadas qualitativamente.

5. Resultados e Discussão

Na tela inicial de acesso do Qlik Sense Web na versão empresarial são dispostos os trabalhos já realizados, publicados e os fluxos criados pela equipe do TCE-MT, tendo a opção de configuração de acesso para cada usuário. Dentro de cada fluxo estão os aplicativos. Os dois primeiros fluxos: Trabalho e publicado são de propriedade de cada usuário. O primeiro fluxo "Trabalho", contém todos os aplicativos criados pelo usuário, já o segundo "Publicado" abrange todos os aplicativos publicados pelo usuário na rede da empresa, sendo que todos os usuários que têm acesso liberado a esta rede conseguirão ver o aplicativo. Dentro de cada aplicativo estão as pastas de visões criadas e dentro das pastas encontram-se os gráficos e tabelas. Para o Radar Saúde foram criadas 4 pastas na qual cada uma contém os gráficos publicados no portal. Estes gráficos foram divididos em pastas a fim de facilitar a organização ao serem incluídos no website. A versão Web do Qlik Sense foi utilizada para gerar os gráficos do Radar saúde. A versão Desktop foi utilizada para exportar os aplicativos e os IDs dos gráficos. Para se instalar o Qlik Sense Desktop é necessário ter uma licença do produto. O TCE-MT tem uma licença Entreprise. O Qlik Sense Web roda através dos servidores do TCE-MT também com uso de licença.

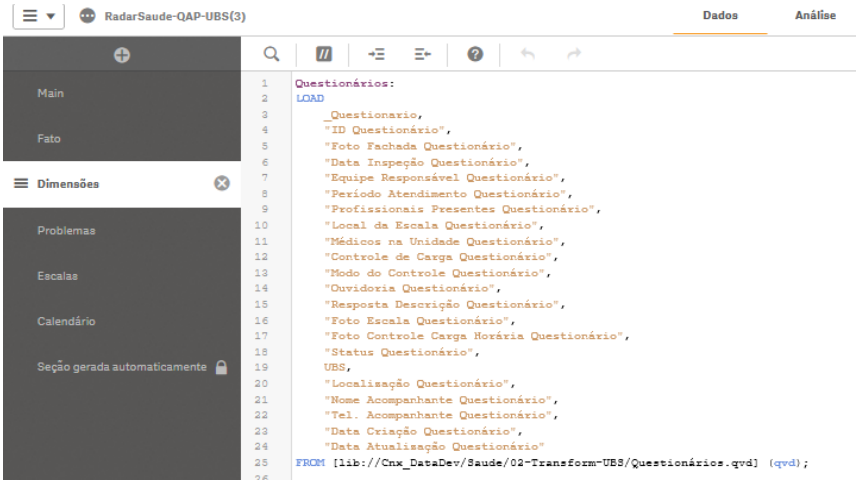


Figura 1. Código em SQL do Radar Saúde

Inicialmente foi gerado um modelo lógico de dados no próprio Qlik Sense para identificação das classes e das relações necessárias. Ao clicar em uma das tabelas do visualizador do modelo de dados, são informados pelo sistema as linhas, os campos, as

chaves e as conexões entre as tabelas. Após a elaboração do modelo lógico de dados, o script em SQL (Structured Query Language) corresponde ao modelo foi gerado no Qlik Sense que ainda fornece as funcionalidades de depuração, carregamento de dados e conexão de dados por bancos de dados ou de forma manual. Um trecho do código em SQL do Radar Saúde é mostrado na figura 1. Este trecho se refere as tabelas de questionários.

Dentro da opção do gerenciador de dados do Qlik Sense foi possível ver as informações inseridas por meio de associação ou tabela. Cada modificação foi acompanhada do carregamento de dados para atualização do sistema. Para melhor gestão, também foi realizada uma modelagem das informações utilizando-se o esquema estrela, o que facilitou o acréscimo de dados ao Radar Saúde (Figura 2).

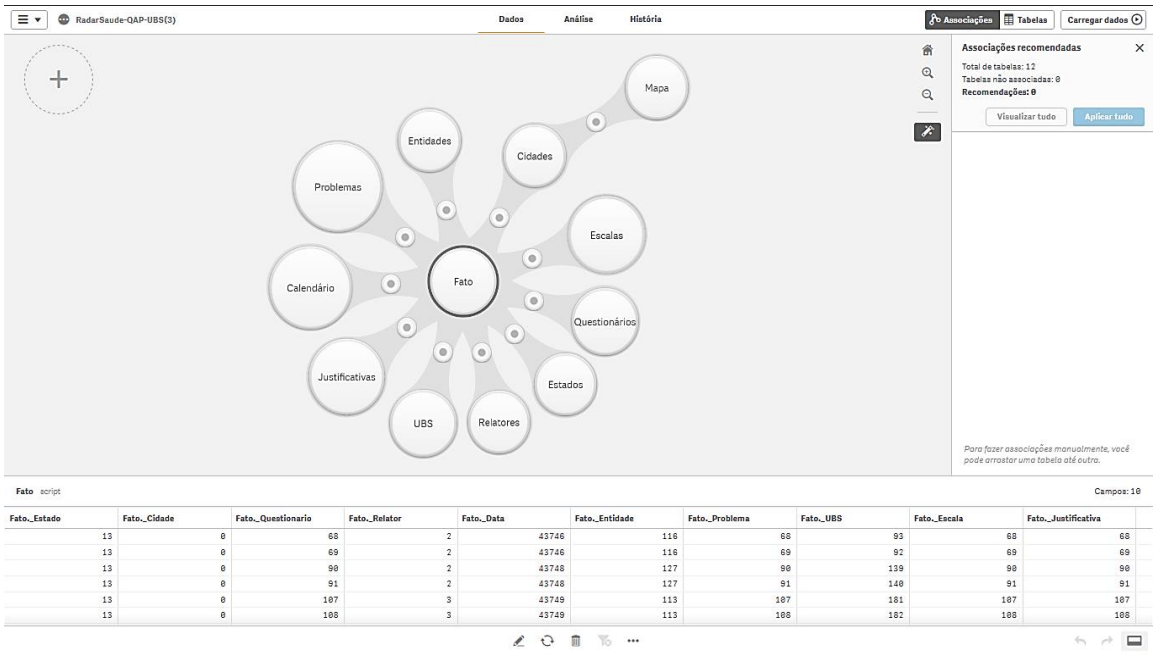


Figura 2. Modelagem em estrela do Radar Saúde no Qlik Sense

Ainda na opção do gerenciador de dados, em tabelas, foi possível ver todas as informações individuais de cada tabela, além de configurar execuções, bloqueios de colunas, alteração de valores de células, etc. Dentro das pastas criadas no Qlik Sense foram criados e editados os campos e os gráficos do Radar Saúde por meio da opção 'gerenciador de dados'. No Qlik Sense, as opções de gráficos são dispostas para utilização como o usuário preferir. Para configurar um gráfico basta arrastar um campo de uma tabela e depois um gráfico para a mesma sessão da tela. A figura 3 mostra a criação do gráfico USBs vistoriadas, que se refere às Unidades Básicas de Saúde que passaram por fiscalização pelos conselhos municipais de saúde.

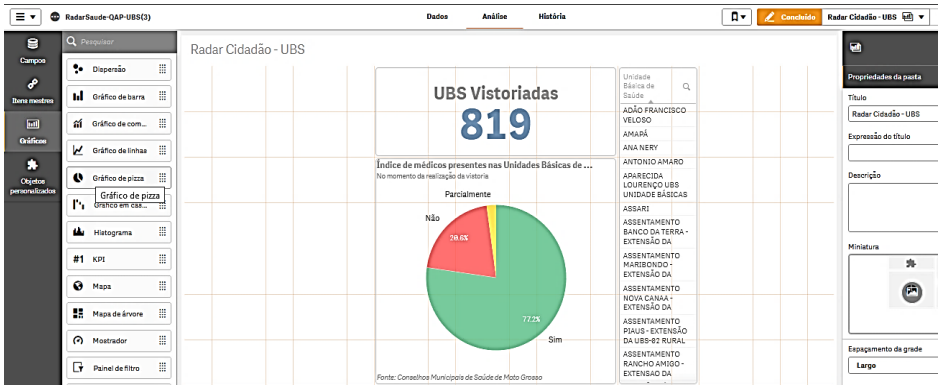


Figura 3. Criação do gráfico UBS Vistoriadas do Radar Saúde no Qlik Sense

Os gráficos criados no Qlik Sense do Radar Saúde indicam a quantidade de municípios fiscalizados, a quantidade de Unidades Básicas vistoriadas, a localização dos profissionais em cada Unidade Básica, o nome, a especialidade, além do início e o término da jornada de cada profissional de saúde. Também foram criados gráficos que mensuram o índice de médicos presentes em cada unidades, a disponibilidade do telefone da ouvidoria de saúde e a forma de controle da carga horária dos profissionais da saúde. Ainda foram criados gráficos referentes a existência de controle de carga horária dos profissionais de saúde, a localização das unidades de Saúde no estado de Mato Grosso, o ranking de trasnparência das escalas médicas e os principais problemas identificados nas Unidades Básicas de Saúde (Figura 4). Para cada município é possível visualizar estas informações detalhadamente.

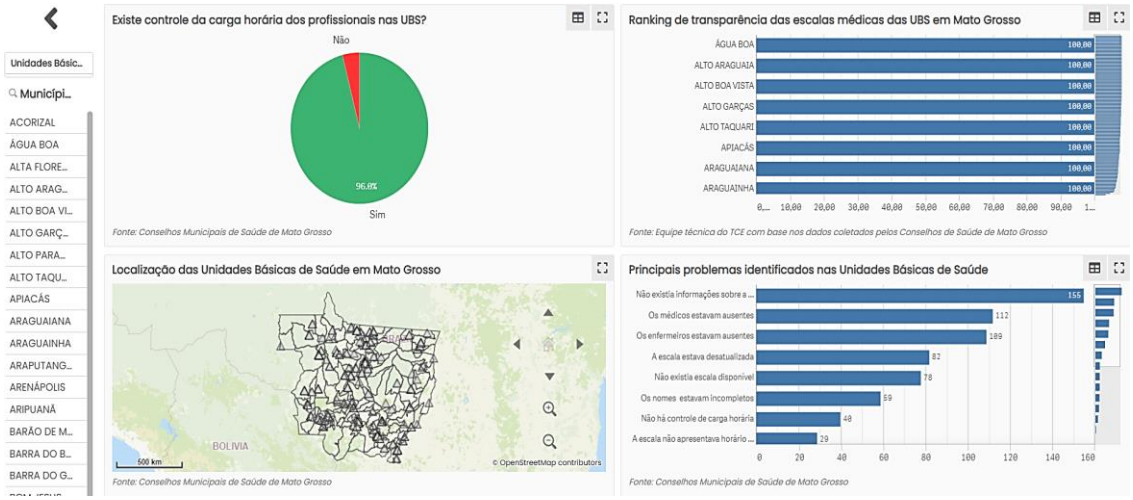


Figura 4. Gráficos do Radar Saúde no Qlik Sense (Controle e transparência)

Após o aplicativo pronto, atráves do Qlik Sense Desktop foi realizada a identificação de cada gráfico/tabela pelo seu ID, para ser inserido no código do site. Por uma questão de boa prática e de padronização faz parte do processo interno do TCE-MT transformar todos os dados em .CSV para o formato QVD (*Qlik View Data*) e grava-los em banco de dados interno antes de inseri-los no Qlik Sense.

Os desenvolvedores envolvidos no projeto Radar Saúde consideraram o grau de dificuldade de aprendizado do Qlik Sense como fácil por terem levado em média um mês para aprender a utilizar esta ferramenta e para criar gráficos simples. Com relação aos

benefícios, o desenvolvedor 1 destacou a utilização de uma ferramenta única para o ETL (Extração/Transformação/Criação dos gráficos). "*Outro ponto de destaque é a possibilidade de realizar um embed em um página HTML utilizando os gráficos desenvolvido do próprio QLIK dando visual mais agradável ao usuário final (D1).*" É necessário ressaltar que embed consiste em tag HTML para incorporar arquivos multimídia de áudio e vídeo. Outro benefício do uso do Qlik Sense foi destacado pelo Desenvolvedor 2 que afirmou que "*o desenvolvimento de dashboards é muito ágil com alta qualidade* (D2)." O terceiro desenvolvedor acrescentou ainda que *"em questão de tempo o Qlik Sense tem vantagem por ter um processo de agendamento da carga de dados simplificado e uma maior facilidade para criação de gráficos e tabelas."* O terceiro desenvolvedor também afirmou que "*a qualidade do Qlik Sense em relação a ferramenta anteriormente utilizada no TCE-MT [Oracle BI] para gerar gráficos é superior por ser feita em uma tecnologia mais recente (HTML5, CSS 3) e ainda ter a opção de integração com páginas web (D3)*."

Os desenvolvedores apontaram como principal desvantagem do Qlik Sense durante o desenvolvimento do Radar Saúde a parte de versionamento da aplicação: "*Apenas um analista edita um projeto, o que é muito ruim mesmo que isso preze pela segurança (D2).*" Outras desvantagens apontadas foram a dificuldade no gerenciamento de backup dos metadados e dos painéis, a dificuldade do compartilhamento de fontes (Exemplo do GIT) e a necessidade de exigir um grande recurso de máquina (memória).

Quando questionados sobre quanto tempo levavam para gerar gráficos e gerencia-los antes de utilizar o Qlik Sense e quanto tempo levam agora com o Qlik Sense, o desenvolvedor 1 respondeu que na parte do transforme dos dados houve ganho de muita agilidade e diminuição de tempo no desenvolvimento da parte gráfica dos Dashboards. "*Posso destacar que é mais ágil devido a facilidade de criação dos gráficos (vários modelos pré-definidos) diminuindo o tempo para atender o usuário uma vez que os dados já estão tratados em nosso data lake* [Sistema de repositório de dados] (D1)." O desenvolvedor 2 por sua vez argumentou que na parte de extração e modelagem dos dados o tempo gasto é semelhante entre o Qlik Sense e a ferramenta anterior [Oracle BI]. Porém a facilidade de agendamento de cargas do Qlik Sense juntamente com a simplicidade para construção dos painéis, que já integram os diversos elementos do painel automaticamente, fazem a construção dos aplicativos ter uma redução no tempo de implementação de cerca de 15% se comparado com a ferramenta anterior. "*No final o analista acaba economizando tempo com a construção final dos gráficos e tabelas no Qlik Sense pela sua facilidade e integração nativa entre os vários elementos adicionados (D3)*."

Quando questionados sobre a qualidade das aplicações antes do uso do Qlik Sense e agora com o uso do Qlik Sense, o Desenvolvedor 1 respondeu que "*a qualidade é maior principalmente se formos falar em parte gráfica em relação ao Oracle BI que não suportava mais HTML 5 e CSS* (D1)." O desenvolvedor 2 declarou que o Qlik Sense foi sua primeira experiência em BI, mas os painéis em Qlik Sense possuem alta qualidade e uma infinidade de análises possibilitando tomadas de decisões precisas. O Desenvolvedor 3 por sua vez afirmou que "*pelo fato de ser uma ferramenta mais recente e com possibilidade de integração com páginas web a qualidade das aplicações subiu*

bastante em comparação com a ferramenta anterior, principalmente por conta da sua interface mais agradável ao usuário (D3)."

Ao serem questionados sobre quais dicas dariam para outros desenvolvedores que pretendem usar o Qlik Sense, o Desenvolvedor 1 indicou que as dicas são: estudar o conceito de Data warehouse, partir para o conceito do motor associativo do Qlik Sense e se aprofundar no AQL (correspondente ao SQL do BI tradicional). *"Não adianta desenvolver no Qlik Sense como desenvolve nos BI tradicionais pois irá perder o melhor recurso da ferramenta que é o modelo associativo* (D1)*."* O Desenvolvedor 1 ainda sugeriu estudar o manual do Qlik Sense e investir em capacitação: "*o manual é bem completo, a comunidade no Brasil é forte, existe curso de introdução na Udemy e muito material bom gratuito* (D1)." O Desenvolvedor 2 ressaltou que "*para quem não está na área, primeira decisão é entender os termos relacionados a área de BI. Para o Qlik em específico é imprescindível entender AQL* (D2)." O Desenvolvedor 3 aconselha ter uma base sólida em banco de dados, com destaque para conceitos de data warehouse, modelagem dimensional, que é a forma como se tratam grandes quantidades de dados e é a base conceitual das ferramentas de BI, além de aprender a linguagem AQL do Qlik Sense que serve para fazer as transformações dos dados e tem características um pouco diferentes do SQL tradicional.

6. Conclusão

O setor público precisa prestar contas dos gastos aos cidadãos de forma transparente, uma vez que todo o dinheiro utilizado advém de impostos que são pagos pela sociedade. Este artigo apresentou o relato do uso do Qlik Sense na implementação do Radar Saúde, que agrega em gráficos dados referentes as Unidades Básicas de Saúde do estado de Mato Grosso permitindo a transparência e a prestação e contas neste contexto. Apesar da facilidade de uso, foi um desafio aprender sobre o Qlik Sense e dominá-lo devido ao curto tempo e a grande demanda de desenvolvimento. Também foi desafiador identificar as classes e atributos necessários para gerar o modelo lógico de dados e as tabelas do Radar Saúde. Entretanto, os ganhos em termos de diminuição de tempo no gerenciamento de dados e aumento da qualidade da aplicação com o Qlik Sense superaram as expectativas. Como implicações teóricas, o artigo auxilia no aumento do corpo de conhecimento relacionado a Qlik Sense. Em termos práticos, o artigo é útil para desenvolvedores que buscam formas mais eficientes para gerenciar dados. Como trabalho futuro pretende-se mensurar quantitativamente por meio de indicadores de tempo, qualidade e custo os benefícios do Uso do Qlik Sense obtidos ao longo do tempo não só no Radar Saúde, mas em outras aplicações do TCE-MT. Este resultado será útil para outras organizações que tenham a intenção de adotar o Qlik Sense em sua rotina de desenvolvimento e de gerenciamento de dados. Os autores agradecem a parceria firmada entre TCE-MT, UFMT e UNISELVA que possibilitou a realização desta pesquisa.

Referencias

da Silveira Farias, E., & Manzanal, M. N. (2013). Pesquisa-Ação em Sistemas de Informação de 2002 a 2012–Uma Revisão Sistemática. IV Encontro de Ensino e Pesquisa em Admnistração e Cotabilidade, Brasília, DF.

de Castro, L. M., & da Silva, M. A. L. (2018). Business Intelligence (BI): Análise comparativa entre as ferramentas líderes no mercado.e-RAC, 8(1).

de Souza Valentim, F., & de Vasconcelos Paiva, V. P (2010). Um Modelo de Gestão e Avaliação de Programas para Melhoria do Desempenho de Instituição do Sistema de Ciência e Tecnologia. XXXIV Encontro da ANPAD, Rio de Janeiro, RJ.

Flores, J. D. S. (2011). Características das ferramentas de Business Intelligence que contribuem para obtenção dos objetivos estratégicos à luz dos princípios de governança corporativa.

Georg, J. 2017. Business Intelligence para gestão de vendas do setor de E-Comerce utilizando Qlik Sense. Trabalho de Conclusão de Curso. Universidade Regional de Blumenau. Blumenau.

Kalaiarasan, T. R., Anandkumar, V., Nivetha, R., & Mathumitha, B. DATA ANALYTICS FOR RETAIL INDUSTRY USING QLIK. Journal of Xi'an University of Architecture & Technology. Volume XII, Issue V, 2020.

Mora, J. M. L. (2020). Qlik sense implementation: dashboard creation and implementation of the test performance methodology (Doctoral dissertation). Universidade Nova de Lisboa, Portugal.

Mungard, T. F., Oliveira, A. A. D., Campos, N. D. S., & Alvares, R. V. (2017). Aplicação Para Gestão Estratégica Dos Dados Do Censo Da Educação Superior. 3º Simpósio Avaliação da Educação Superior, AVALIES 2017, Florianópolis.

Musskopf, Gabriela Witz. Analise das Ferramentas de Business Intelligence Utilizadas por Empresas Brasileiras. 2017. 94 f. Trabalho de Conclusão de Curso – Administração. Universidade Federal do Rio Grande do Sul, Porto Alegre.

Negash, S., & Gray, P. (2008). Business intelligence. In Handbook on decision support systems 2 (pp. 175-193). Springer, Berlin, Heidelberg.

Nunes, E. D. (2016). Uma Análise Comparativa Sobre Ferramentas Para Self-Service Business Intelligence. Trabalho de Conclusão de Curso – Sistemas de Informação, Universidade Federal de Pernambuco, Recife.

Salas Urbano, N., & Bruce, N. (2020). Making Sense of Multivariate Data: An Iterative Design of the Visualization and Interactions with Parallel Coordinates. Master Thesis. Lundi University. Lundi.

Torres, N. N., Guerra, E. L., & Lima, A. M. (2014). Uma pesquisa-ação da metodologia lean startup em um empreendimento de software. In Anais Principais do X Simpósio Brasileiro de Sistemas de Informação (pp. 446-457). SBC.

Vashisht, V., & Dharia, P. (2020). Integrating Chatbot Application with Qlik Sense Business Intelligence (BI) Tool Using Natural Language Processing (NLP). In Micro-Electronics and Telecommunication Engineering (pp. 683-692). Springer, Singapore.

Wainer, J. (2007). Métodos de pesquisa quantitativa e qualitativa para a Ciência da Computação. Atualização em informática, 1, 221-262.

Aprimorando a Interação do Sistema de Ajuda de um Software Previdenciário

Douglas H. Pozzolo[1], Cristiano Maciel[1], Raphael de S. R. Gomes[1]

[1]Instituto de Computação - Universidade Federal de Mato Grosso (UFMT) - Cuiabá, Mato Grosso - Brasil

douglas.pozzolo@gmail.com, cmaciel@ufmt.br, raphael@ic.ufmt.br

Abstract. *In this research, Human-Computer Interaction methods and techniques are used in a social security software under development - WebRecad, specifically on its help system. Based on the results of usability tests and user experience evaluation methods, improvements to the software are proposed. These strategies also allow the company to take ownership of them in order to improve the development of their systems.*

Resumo. *Neste estudo são aplicados métodos e técnicas de Interação Humano-Computador em um software previdenciário em desenvolvimento - o WebRecad, mais especificamente no que se diz respeito ao seu sistema de ajuda. Com base nos resultados das avaliações de usabilidade e user experience, são propostas melhorias ao software. Tais estratégias permitem a empresa se apropriar delas para o desenvolvimento de seus sistemas.*

1. Introdução

A interface de um sistema mal projetado pode causar grandes dificuldades, e até mesmo causar graves danos na realização de trabalhos de uma organização, e ainda assim, poucas organizações consideram este fato quando planejam suas aplicações (SHNEIDERMAN, 1997). Na área previdenciária, alguns *softwares* têm sido desenvolvidos. Entre eles, o WebRecad, um sistema de gerenciamento de segurados de um sistema previdenciário. Em parceria com sua empresa de desenvolvimento, a Webtech - Softwares e Serviços, percebeu-se que o mesmo, em sua fase de construção, não tem passado por um processo de revisão de usabilidade até o início do projeto deste artigo. Em uma primeira avaliação, percebe-se que a ajuda do usuário era falha. Face a esta problemática, questionou-se: como avaliações de usabilidade e de user experience (UX) podem agregar valor ao sistema de ajuda WebRecad, minimizando problemas com o uso dele? Evitar que o *software* falhe em quesitos como usabilidade, principalmente no que diz respeito à ajuda do sistema, tende a aumentar a qualidade de seu uso, é o que motiva o desenvolvimento deste trabalho.

Neste sentido, este artigo visa a utilização de métodos e técnicas da interação humano-computador (IHC) no contexto prático da produção do WebRecad, em seu sistema de ajuda. Para isto, o *software* foi sujeito às técnicas de avaliação: avaliação heurística, *checklist*, teste de usabilidade e *user experience*.

2. Referencial Teórico

No que se refere à experiência de utilização de um sistema, Nielsen (2007) conceitua usabilidade como elemento que garante a facilidade ao uso de algo. Assim, a avaliação

de interface é imprescindível aos sistemas à medida que o desenvolvimento das interfaces dispõe de grandes desafios em relação às metodologias de projeto, que são vitais para ajudar cada ciclo de um sistema (BERTINI et al., 2009).

Antes de lançar um *software* para uso, é necessário identificar se o mesmo apoia de forma adequada seus usuários nas suas tarefas e em seu ambiente de uso. Seguir à risca métodos e princípios de projeto de interfaces não são suficientes para garantir uma alta qualidade de uso de um *software*. Desta forma, não somente testes de funcionalidade serão necessários para verificar a robustez da implementação, mas também a avaliação da interface será necessária para analisar a qualidade de uso do *software* (PRATES, BARBOSA, 2003). Exploraremos a seguir os conceitos das técnicas utilizadas neste artigo: a avaliação heurística, *checklist*, teste de usabilidade e avaliação de *user experience*.

A avaliação heurística é um método analítico de inspeção de interfaces criado por Jacob Nielsen e Molich (1993), que tem por objetivo identificar problemas de usabilidade baseado em conjunto de heurísticas (diretrizes) e até mesmo na experiência dos próprios avaliadores (NIELSEN, 1994). Este método é gerado das melhores práticas definidas por profissionais especialistas em IHC ao longo de anos trabalhados na área (PRATES; BARBOSA, 2003). Segundo Prates e Barbosa (2003), para cada heurística violada, o especialista deve definir a localização do problema e a sua gravidade (calculada com fatores como frequência, impacto na aplicação e persistência). Como produto final da avaliação, o especialista redige um relatório consolidado.

Por sua vez, a *checklist* é uma ferramenta de inspeção de usabilidade que tem poder de diagnosticar rapidamente problemas gerais e repetitivos das interfaces e pode ser usada com muita eficiência, não somente por profissionais especialistas em usabilidade, mas também programadores e analistas, por exemplo (JEFFRIES et al., 1991). Diferentemente da avaliação heurística, é a qualidade da lista de verificação (*checklist*), que determina as possibilidades da avaliação. Quanto melhor elaborada a *checklist*, maior a produção de resultados uniformes e abrangentes na definição dos problemas de usabilidade. Elas reduzem a subjetividade em relação ao processo de avaliação e são de rápida aplicação (CYBIS, 2003).

Por exemplificação, este trabalho faz uso de uma *checklist,* fruto de estudos contextualizados em *softwares* governamentais brasileiros criada por Maciel, Nogueira e Garcia (2005), denominada g-Quality. A *checklist* mapeia itens baseados nas heurísticas de Nielsen (1994) e no e-Ping (2004) - sigla de "Padrão de Interoperabilidade do Governo Eletrônico" - além de permitir avaliar acessibilidade, interoperabilidade, segurança e privacidade, veracidade da informação, agilidade do serviço e transparência de um sítio. Cabe ressaltar que o g-Quality tem visibilidade internacional (GARCIA et al., 2005) e, em alguns estudos, recortes dele tem sido utilizado e até atualizados, como em Silva, Maciel e Silva Junior (2020), neste caso, no que se refere à interoperabilidade.

O teste de usabilidade tem como foco a avaliação da qualidade da interação do usuário com o sistema, objetivando a métrica dos impactos sobre esta e a identificação dos aspectos da interface que geram desconforto ao usuário (CYBIS, 2017). De acordo com Santa Rosa (2008), os avaliadores produzem um roteiro com as tarefas que os participantes da pesquisa deverão realizar. O teste é realizado com a observação desta

interação em um meio real ou sob condições controladas. Os avaliadores recolhem dados dos problemas encontrados através da interação dos usuários com a interface e verificam, posteriormente, se ela suporta o ambiente e as tarefas que o usuário executará (MATIAS, 1995). Procura-se quantificar, por meio deste teste, o desempenho do usuário. Dentre os exemplos de medidas no teste de usabilidade, é possível listar o tempo gasto para realizar uma tarefa, a quantidade de erros cometidos e a porcentagem dos usuários que conseguem se recuperar de um erro (PRATES, BARBOSA, 2003).

Além de usabilidade, deve-se levar em consideração que a atividade de um usuário num sistema tem relação direta na predisposição dele a utilizar um *software*, e a partir disso, se torna necessário que o *software* esteja em conformidade com a personalidade, emoção, humor, objetivos e preferência de seus usuários, e além disso, seu contexto físico, social e virtual onde irão ocorrer a interação com a interface (MCCARTHY; WRIGHT, 2007). Diversas técnicas foram criadas para compreender o estado emocional do usuário sobre cada detalhe do produto, para que a equipe de desenvolvimento seja capaz de tomar ações que causam melhorias na UX. Dentre elas, destaca-se os Emocards. Esta é uma técnica de avaliação não-verbal que consiste em mensurar respostas emocionais por meio de expressões faciais, permitindo assim validar as emoções expressadas independentemente da cultura do usuário (DESMET; OVERBEEKE; TAX, 2001). Ele consiste de 16 cartas com ilustrações faciais, sendo 8 masculinas e 8 femininas, representando possíveis emoções que contemplam dimensões de prazer e estimulação (SILVA; KRONBAUER, 2018).

3. Metodologia

Neste tópico serão descritos os métodos e técnicas utilizados juntamente com motivo das escolhas dos mesmos, além da a especificação da ferramenta WebRecad. Serão apresentados fluxos do processo realizado na produção do estudo e, em sequência, serão explicados em maiores detalhes.

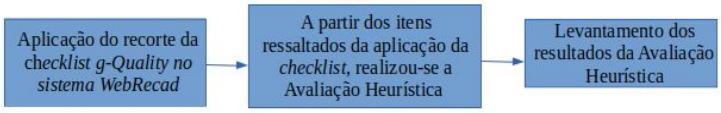


Figura 1 – Fluxo da criação dos resultados da Avaliação Heurística

3.1 WebRecad

O WebRecad é um sistema *web* em produção pela empresa Webtech - Softwares e Serviços designado ao recenseamento previdenciário, isto é, responsável por convocar e agendar segurados de um regime próprio de previdência social (RPPS) de uma determinada entidade governamental, para atualização de seus dados cadastrais e documentação comprobatória a fim de manter seus dados consistentes e seus benefícios validados dentro das regras da previdência social.

O fluxo principal do WebRecad pode ser dividido em cinco etapas: 1) cadastro de parâmetros, onde são definidos todos os parâmetros necessários para o recadastramento, por exemplo, os tipos de documentos que podem ser apresentados; 2) cadastro de projetos de recadastramento, em que se identifica as singularidades de um projeto de recadastramento, como quais documentos serão tidos como obrigatórios, se haverá coleta de biometria e fotos 3) convocação e agendamento, no qual, a partir dos projetos

de recadastramento, é possível realizar a convocação dos segurados, filtrando os segurados de forma personalizada, por mês de nascimento, por exemplo. 4) a triagem, que trata-se da verificação dos documentos apresentados pelo segurado para averiguar se está com toda a documentação obrigatória para realizar a etapa de recadastramento; e 5) o recadastramento, a etapa fundamental de atualização dos dados cadastrais e apresentação de seus documentos comprobatórios.

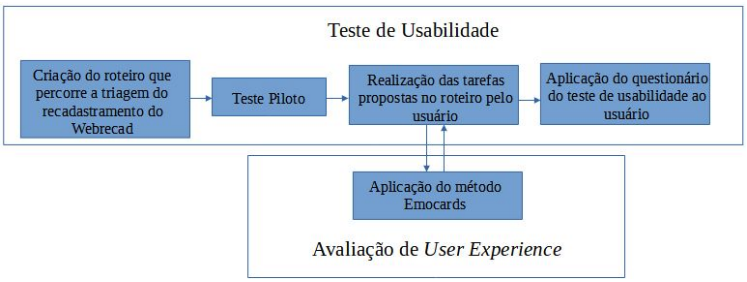


Figura 2 – Aplicação do teste de usabilidade e avaliação de *user experience*

O WebRecad possui, também, uma Central de Ajuda para o usuário, lá encontram-se dois manuais: um no formato PDF possuindo 49 páginas para operadores e 82 páginas para administradores, e outro em formato HTML (Linguagem de Marcação de HiperTexto) que é dividido em tópicos para cada módulo do sistema. No manual em HTML, existe um sistema de busca no qual é possível fazer uma busca pelo nome dos tópicos desejados.

3.2 g-Quality

No intuito de avaliar os métodos de ajuda na interface do WebRecad, utilizou-se um recorte da *checklist* utilizada no estudo do g-Quality (MACIEL; NOGUEIRA; GARCIA, 2005) no que diz respeito à parte que trata da heurística de Nielsen "Ajuda e Documentação". Como este foi criado para análise de interfaces de sítios governamentais, se enquadra como uma potencial ferramenta para a avaliação do WebRecad. Os itens da *checklist* identificados como problema de usabilidade serão utilizados como base para a avaliação heurística no passo posterior. A *checklist* adaptada foi aplicada em dezembro de 2019 e preenchida conforme proposto no método.

3.3 Avaliação Heurística

A partir de cada problema de interface identificado com a *checklist*, foi realizada uma avaliação heurística sobre o quesito. Com no artigo "Avaliação Heurística de Sítios na Web" de Maciel et al. (2004) e no formulário fornecido neste, foi possível classificar, atribuir um grau de severidade ao problema, identificar a causa e os efeitos trazidos por ele sobre o usuário e a tarefa, e propor uma possível solução.

3.4 Teste de Usabilidade

Como método empírico de avaliação de interface, utilizou-se o teste de usabilidade. Para tal, foi criado um roteiro de tarefas as quais deveriam ser realizadas pelos usuários (Figura 2). Este teste teve como objetivo induzir o usuário, por meio de tarefas dentro do WebRecad, a utilizar a central de ajuda do sistema, e assim, criar um feedback sobre a qualidade e eficiência do sistema denominado "Central de Ajuda". O teste foi

realizado na sede da Webtech por 5 participantes com perfil de operador WebRecad, com os critérios de que tivessem acima de 18 anos de idade, noções de informática básica e que não tivessem contato prévio com o sistema. Todo o procedimento foi gravado com o consentimento do usuário, tanto a tela da interface, quanto o rosto do participante durante a realização da avaliação. Realizou-se, anteriormente, um teste piloto com um avaliador para que se testasse a qualidade do roteiro criado e se seus aspectos - objetos de avaliação - fossem postos à prova. Os dados recolhidos neste teste não foram contabilizados.

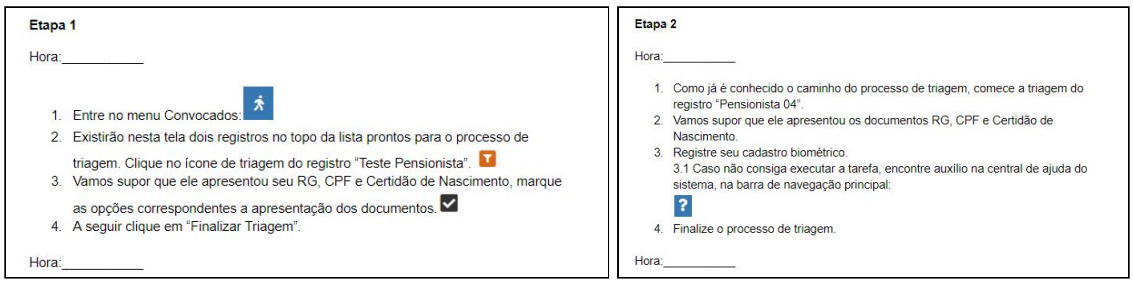


Figura 3 – Etapas do roteiro do Teste de Usabilidade

O roteiro foi escrito para a realização de dois processos de triagem no WebRecad. Com dois registros de segurados já preparados previamente, foi pedido aos usuários que o realizassem de acordo com as instruções do roteiro, que possui duas etapas (Figura 3): a primeira, demonstrando passo a passo de como realizar um processo de triagem simples, sem cadastramento de biometria; na segunda etapa, já com menos detalhamento sobre o procedimento, pediu-se que o usuário realizasse o processo de triagem para outro registro, mas desta vez com cadastro biométrico, uma tarefa mais complexa. No intuito de fazer com que o usuário buscasse auxílio na central de ajuda do sistema existia uma observação abaixo da tarefa: "Caso não consiga executar a tarefa, encontre auxílio na central de ajuda do sistema" com informações de como acessá-lo, indicando a existência de uma central de ajuda, onde se localiza o manual de usuário. Ambas etapas possuem um campo para registro das horas a fim de medir o tempo que o usuário levou para a realização das tarefas. Ao finalizar o teste, o usuário foi submetido a um questionário com perguntas-chave sobre a facilidade de uso do sistema e a necessidade de auxílio na realização das atividades.

3.5 Avaliação de UX

Conforme ilustrado na Figura 2, simultaneamente com o teste de usabilidade, foi realizada uma avaliação de *user experience*. Para tal, foi entregue uma página com Emocards (Desmet; Overbeeke; Tax, 2001) (Figura 4), sendo que, ao realizar cada uma das etapas da tarefa, o usuário foi orientado a selecionar um card que mais se aproxima da emoção sentida ao realizá-la. A técnica foi escolhida para que a emoção do usuário sobre cada uma das etapas seja, de forma não-verbal, reconhecida e analisada a reconhecer o grau de satisfação do usuário ao interagir com a interface em virtude das prováveis dificuldades encontradas. Foi solicitado ao usuário anotar o número (de 1 a 8) que representa a emoção sentida em relação a cada um dos passos do roteiro. Utilizaram-se os Emocards originais, em conjunto com as orientações, sendo que o aplicador do teste os traduziu e esclareceu dúvidas sobre o método aos participantes.

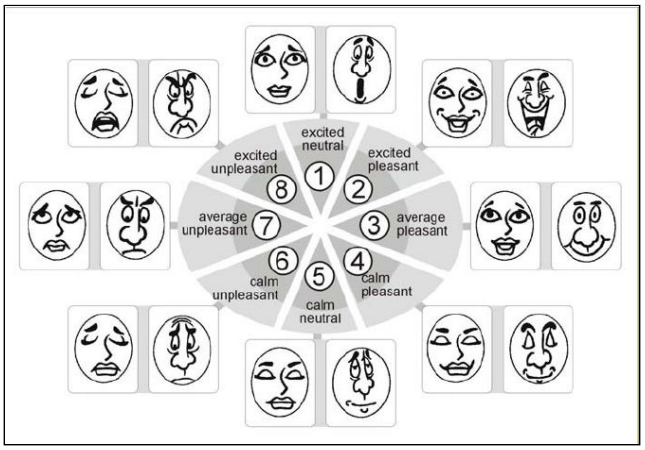


Figura 4 - Emocards de Desmet, Overbeeke & Tax (2001).

4. Resultados

A seguir, serão apresentados os resultados das aplicações das metodologias.

4.1 g-Quality

A aplicação do recorte de "Ajuda e documentação", da *checklist* do método g-Quality no sistema do WebRecad, permitiu verificar os itens conforme elencado na Tabela 1. A primeira coluna identifica qual item da *checklist* está sendo verificado, em "Opção" foram escolhidos números de 0 a 2 baseados nos critérios: 0 para "nunca/opção não testável"; 1 para "às vezes" e 2 para "sempre", de acordo com o proposto no método.

Tabela 1. Verificação da *Checklist - Ajuda e Documentação*

Item da *checklist*	Opção
1. Fornece recurso de ajuda (help) integrado com as páginas do sítio.	2
2. Não disponibiliza ajudas muito extensas.	0
3. Disponibiliza de ajudas contextualizadas.	1
4. A informação está organizada de forma organizada, com índice de separação entre as áreas.	1
5. Partes do sistema não têm acesso a ajuda.	2
6. Falta ajuda para determinados contextos ou ações.	2
7. Falta mecanismo de busca na ajuda.	0
8. Disponibiliza uma FAQ - Frequently Asked Questions (Perguntas Mais Frequentes).	0
9. Utiliza recursos de atendimento pessoal de forma online e em tempo real.	0

4.2 Avaliação Heurística

A avaliação heurística foi realizada com base no proposto na literatura e verificando cada item elencado como problema de usabilidade na *checklist* previamente apresentada. Percebe-se que o WebRecad possui apenas o manual do usuário integrado no sistema como fonte de auxílio e por ser um material muito extenso, pode causar muita hesitação no seu uso por parte dos usuários. Ficar revirando páginas e páginas de texto pode ser um tanto enfadonho, no entanto, o sistema acerta ao possuir um sistema de busca em seu manual HTML, sendo possível procurar palavras-chave dentre os

títulos dos tópicos. Sair do contexto de uso para acessar a central de ajuda também é outro fator que pode causar retrabalho ao usuário, além desconforto e aborrecimento.

Com o preenchimento dos formulários, observa-se que a solução indicada para quase todos os problemas de usabilidade, no que se refere à ajuda do sistema, convergem para a adição de ajuda contextualizada nas páginas do sistema. É sugerido o uso de pequenos ícones próximos a campos e ações que possam gerar dúvidas aos usuários, que ao clicá-los, abre-se um pequeno balão explicativo sobre para que serve tal campo, ou o que faz tal ação. É sugerido também, para maximizar o uso do manual dentro do sistema, links dentro destes balões que, abrem uma segunda janela do navegador exatamente na página em que é explicado seu uso no manual. Outra sugestão é a existência de vídeos tutoriais para as ações que geram as dúvidas mais frequentes. É interessante destacar, logo ao entrar na central de ajuda, uma espécie de FAQ (perguntas mais frequentes, traduzido de Frequently Asked Questions) com a solução ou a descrição dos campos mais complexos, a apresentação do significado das siglas utilizadas, ou ações que mais geram dúvida nos usuários enquanto utilizam o sistema. A pesquisa de da Silva et al. (2013) enfatiza o uso de ajudas contextualizadas em sistemas de governo eletrônico.

4.3 Teste de Usabilidade

O teste de usabilidade foi aplicado na própria sede da Webtech com a participação de 5 usuários que se encaixam no perfil de operador WebRecad. A análise do resultado será dividida em 3 seções: primeira etapa do roteiro, segunda etapa do roteiro e o questionário. A tarefa a ser realizada pelo participante foi de realizar a triagem do registro de um segurado previamente agendado para o momento do teste. O procedimento a ser executado pelo usuário consistia em entrar na página dos registros, encontrar o registro descrito no roteiro e sinalizar de que este apresentou todos os documentos requeridos e a seguir, encerrar o processo de triagem.

A maior dificuldade encontrada pelos usuários na etapa 1 da tarefa (Figura 3) foi a localização do menu, uma vez que estava disposto em ícone, sem legenda visível, sendo necessário passar o ponteiro do mouse sobre ele para encontrar a descrição “Convocados”. É questionável o símbolo do ícone, que mesmo estando explícito no roteiro, pode ser apontado como item de confusão aos usuários, sendo necessário um estudo sobre um ícone mais adequado. Na segunda etapa, a tarefa a ser realizada pelo participante é de realizar a triagem de um segundo segurado, porém desta vez, realizando um cadastro biométrico, utilizando os próprios dedos para simular os do segurado que está passando pela triagem. Mesmo sem possuir uma legenda visível na área de cadastramento de biometria, contendo apenas um ícone de leitura biométrica, a maioria (80%) dos usuários reconheceram rapidamente o local para realizar o procedimento. Ainda que 100% dos usuários tiveram dúvidas em relação ao cadastro biométrico, nenhum deles recorreu ao auxílio da Central de Ajuda. Existiu total hesitação por parte dos usuários de sair do contexto atual (janela de cadastro de biometria) para buscar auxílio no sistema, o fato é realçado ao observar o usuário passando o mouse por cima do ícone em momento de dificuldade, mas ainda assim preferir clicar nos botões dispostos para testar a resposta da interface. Com a aplicação do questionário após a utilização do sistema pelos participantes, observa-se que a central de ajuda precisa ser reestruturada no WebRecad.

Todos os usuários alegaram encontrar dificuldades ao usar o sistema, porém houve grande hesitação ao entrar numa página nova para encontrar ajuda. A preferência por tentativa e erro deve ser evitada ao máximo pela interface, uma vez que pode gerar danos à integridade dos dados e diminuir a satisfação do usuário com a interface. Há entre eles uma grande preferência (80%) por vídeos tutoriais a longos manuais. Ao apresentar uma nova ideia de ajuda no sistema, 100% disseram que recorreriam a uma forma de ajuda na própria janela que está realizando as atividades.

4.4 Avaliação de User Experience

A avaliação de UX foi realizada em conjunto ao teste de usabilidade, assim que o usuário realizasse um passo na lista de tarefas, ele foi orientado a escolher um card de emoção, da técnica Emocards (DESMET; OVERBEEKE; TAX, 2001), correspondente à experiência do contato com a interface. Os emocards são numerados de 1 a 8 conforme a Figura 4. A tabela 2 demonstra o Emocard escolhido pelos usuários na realização dos passos do roteiro do teste de usabilidade (Figura 3).

Tabela 2. Resultados do Teste de UX

	Etapa 1				Etapa 2			
	Passo 1	Passo 2	Passo 3	Passo 4	Passo 1	Passo 2	Passo 3	Passo 4
Usuário 1	6	4	4	4	7	7	8	2
Usuário 2	6	5	4	3	3	3	7	3
Usuário 3	7	3	6	1	3	3	5	3
Usuário 4	3	3	3	3	3	3	4	3
Usuário 5	5	7	5	5	5	5	7	5

Analisando a numeração dos Emocards, as emoções consideradas positivas se referem às de número 1 a 4, 5 é uma emoção completamente neutra, enquanto as emoções de 6 a 8 são emoções negativas que aumentam de intensidade à medida em que se aproximam do card 8. Na etapa 1 observa-se que o passo 1 gerou um grande número de reações negativas (60%), no passo 2 ocorrem duas reações negativas, apenas uma delas com grande intensidade, no passo 3 ocorre grande quantidade de reações neutras e duas consideradas pouco desagradáveis, na etapa 4 ocorre a maior parte de reações positivas. Na segunda etapa do teste, os passos 1 e 2 possuem 60% de reações positivas, 20% muito desagradável e 20% levemente desagradável. O passo 3 é o que gerou maior quantidade de reações negativas com maior intensidade de todo o teste e em seguida, o quarto passo trouxe emoções positivas (80%) em sua maioria.

Ao comparar os resultados do teste de usabilidade com o teste de *user experience* podemos notar que o pico de reações negativas gerada pelo passo 1 da etapa 1 ocorre pela dificuldade do usuário de encontrar o menu, da forma que ele disposto na interface. Já o pico de reações negativas gerado pelo passo 3 da etapa 2 ocorre pela dificuldade enfrentada pelo usuário de realizar a tarefa, nota-se que nesta etapa, mais do que todas as outras, o usuário necessita de ajuda. Com a análise das imagens gravadas, a etapa de começar o cadastro biométrico é facilmente encontrada e o incômodo da dificuldade apresentada nos dados tem início no momento em que o usuário interage com o botão “Capturar” e utiliza o leitor biométrico, e retorna no momento em que é necessário posicionar o dedo no leitor novamente para confirmação da digital.

5. Considerações Finais

Todos os métodos utilizados para a avaliação de interface trouxeram informações primordiais para um estudo mais aprofundado sobre como melhorar a qualidade e a humanização da interação do usuário com a ajuda do sistema. Embora o teste de usabilidade ter sido mais custoso, em questão de tempo de preparo e aplicação, trouxe informações mais ricas sobre os problemas de usabilidade, encontrando até mesmo problemas fora do contexto de ajuda e documentação, como ícones inadequados e menus de navegação confusos.

A avaliação heurística, juntamente com a *checklist*, indicaram que o sistema de ajuda não teria alta probabilidade de eficiência, uma vez que o manual do usuário, que apesar de ser muito rico em detalhes, é muito extenso. Além de destacar que o redirecionamento do usuário a uma página nova, saindo do contexto do seu problema para buscar ajuda, causa hesitação dos usuários ao utilizar a ferramenta. O teste de usabilidade, quando aplicado com o método dos Emocards, não somente confirmou a hesitação dos usuários em utilizar a ferramenta de ajuda como também provou que nenhum sequer a utilizou.

De forma geral, todo o estudo aponta que o WebRecad precisa reformular seu método de ajuda, sugerindo, entre outras formas, a utilização de ajuda contextual além da possível implementação de vídeos tutoriais ou *links* que, além de abrirem uma nova janela (para não ocorrer o risco de sair do contexto do problema), revelam explicação mais detalhada da dúvida dentro do próprio manual do usuário já existente. Todos os resultados obtidos pela aplicação dos testes e conceitos de IHC na interface do WebRecad apontaram elementos que, ao serem trabalhados, trarão maior qualidade ao *software* final. Isso reitera a importância da aplicabilidade de métodos, técnicas e ferramentas de IHC dentro de corporações e projetos de *software* no que diz respeito a qualidade do uso e, por consequência, o produto final.

Os resultados gerados por esta pesquisa foram repassados para a própria empresa, para que se estude, juntamente com suas normas de criação e seus desenvolvedores, o melhor meio de aprimorar a ferramenta de ajuda e documentação dentro, não somente do WebRecad, mas de todos seus sistemas. Assim, ficou claro, no ambiente empresarial em que esse *software* é utilizado, que os aspectos humanos são fundamentais no desenvolvimento dos sistemas, e precisam sempre ser considerados, uma vez que podem afetar, inclusive, os aspectos comerciais da empresa.

6. Referências

Bertini, E., Catarci, T., Dix, A., Gabrielli, S., Kimani, S., & Santucci, G. (2009). Appropriating heuristic evaluation for mobile computing. International Journal of Mobile Human Computer Interaction (IJMHCI), 1(1), 20-41.

Cybis, W. D. A. (2003). Engenharia de usabilidade: uma abordagem ergonômica. Florianópolis: Labiutil.

Cybis, W., Betiol, A. H., & Faust, R. (2017). Ergonomia e usabilidade: conhecimentos, métodos e aplicações. Novatec editora.

da Silva, Liziane A. et al. Designing help system for e-GOV websites: A Brazilian case study. **Information Polity**, v. 18, n. 3, p. 261-274, 2013.

Desmet, P., Overbeeke, K., & Tax, S. (2001). Designing products with added emotional value: Development and appllcation of an approach for research through design. The design journal, 4(1), 32-47.

Garcia, A. C. B., Maciel, C., & Pinto, F. B. (2005). A quality inspection method to evaluate e-government sites. In International Conference on Electronic Government (pp. 198-209). Springer, Berlin, Heidelberg.

Jeffries, R., Miller, J. R., Wharton, C., Uyeda, K. (1991). User interface evaluation in the real world: A comparison of four techniques. In proceedings of CHI'91. In Conference on Human Factors and Computing Systems, New Orleans, LA (pp. 119-124).

Maciel, C., Nogueira, J. L. T., Ciuffo, L. N., Garcia, A. C. B. Avaliação heurística de sítios na web. VII Escola de Informática da SBC-Centro-Oeste, 2004.

Maciel, C.; Nogueira, J.L.T.; Garcia, A.C.B. An x-ray of the brazilian e-gov web sites. In: **IFIP Conference on Human-Computer Interaction**. Springer, Berlin, Heidelberg, 2005. p. 1138-1141.

Maciel, C.; Nogueira, J.L.T.; Garcia, A.C.B. g-Quality: um método para avaliação da qualidade dos sítios de e-Gov. VIII Escola de Informática do SBC – Centro-Oeste, Cuiabá. SUCESU-MT. PAK Multimídia, Cuiabá, 2005.

McCarthy, J.; Wright, P. (2007) Technology as experience. The MIT Press.

Nielsen, J. (1994). Heuristic evaluation. In Usability inspection methods (pp. 25-62). John Wiley & Sons, Inc..

Nielsen, J. (2007). Loranger, Hoa. Usabilidade na web: Projetando websites com qualidade. Rio de Janeiro: Elsevier.

Prates, R. O., Barbosa, S. D. J. (2003). Avaliação de interfaces de usuário–conceitos e métodos. In Jornada de Atualização em Informática do Congresso da Sociedade Brasileira de Computação, Capítulo (Vol. 6, p. 28).

Shneiderman, B. (1997). Designing the user interface: strategies for effective human-computer interaction. New York: Addison-Wesley, 1997.

Silva, B., Maciel, C; Silva Junior, D. Interoperability Inspection Method in Electronic Government based on e-PING Architecture. In XVI Simpósio Brasileiro de Sistemas de Informação, SBC, 2020.

Estratégias Remotas à Avaliação de Interfaces de Usuário

Amanda M. Melo, Ícaro M. Crespo, Gabriela C. Medeiros, Amanda B. de Oliveira

Universidade Federal do Pampa (Unipampa) – Campus Alegrete
Av. Tiarajú, 810 – Ibirapuitã – 97.546-550 – Alegrete – RS – Brasil

amanda.melo@unipampa.edu.br, {icarocrespo, amandapadme, gabiceemi}@gmail.com

Abstract. *There are different approaches to the integration of Human-Computer Interaction with Software Engineering. In these approaches, user interface evaluation is recognized as an important strategy to promote the quality of use of software systems. In the current context, where social isolation is necessary as a way to combat the pandemic by COVID-19, several activities started to be carried out remotely, including the evaluation of user interfaces. In this article, we cover three cases of user interface evaluation that were conducted completely remotely. As a result, we present the protocols for three remote evaluation methods.*

Resumo. *Há diferentes abordagens à integração da Interação Humano-computador à Engenharia de Software. Nessas abordagens, a avaliação de interface de usuário é reconhecida como importante estratégia para promover a qualidade de uso de sistemas de software. No contexto atual, onde o isolamento social se faz necessário como forma de combate à pandemia por COVID-19, várias atividades passaram a ser realizadas remotamente, inclusive avaliações de interfaces de usuário. Neste artigo, abordamos três casos de avaliação de interface de usuário que foram conduzidos de modo totalmente remoto. Como resultados apresentamos os protocolos para três métodos de avaliação remota.*

1. Introdução

Pessoas percebem, compreendem e operam sistemas de *software* através de suas interfaces de usuário. Bons projetos dessas interfaces definem a aceitação ou a rejeição de um sistema de *software*. A área de Interação Humano-computador (IHC), de forma interdisciplinar e interprofissional, tem contribuído ao projeto e à avaliação de sistemas computacionais interativos, assim como à compreensão do uso desses sistemas e os fenômenos relacionados a esse uso [Hewett *et al.* 1992].

Barbosa e Silva (2010) apresentam três abordagens para a integração da Interação Humano-computador à Engenharia de Software (ES):

> definição de características de um processo de desenvolvimento que se preocupa com a qualidade de uso; definição de processos de IHC paralelos que devem ser incorporados aos processos propostos pela ES; indicação de pontos em processos propostos pela ES em que atividades e métodos de IHC podem ser inseridos. [Barbosa e Silva 2010, p. 123]

Em todas essas abordagens, a avaliação da interface de usuário é uma importante estratégia para promover a qualidade de uso. Essa avaliação vai além de observar aspectos da construção interna do *software*. Envolve também compreender se o sistema apoia adequadamente seus usuários a atingirem seus objetivos em determinado contexto de uso ou até mesmo de que modo um sistema afeta seus usuários. Para isso, diferentes métodos e

ferramentas podem ser adotados em diferentes momentos de um processo de desenvolvimento de *software*.

No contexto atual, onde o isolamento social se tornou necessário como modo de combate à pandemia por COVID-19, várias instituições passaram a desenvolver suas atividades de modo remoto, inclusive as Universidades. Atividades que até então eram realizadas presencialmente foram adaptadas, entre elas avaliações de interfaces de usuário de sistemas de *software*.

Em uma revisão de trabalhos relacionados sobre métodos e ferramentas remotos de apoio à avaliação de interfaces de usuário, com ênfase na usabilidade e na resposta afetiva de usuários, observa-se que estes conferem maior flexibilidade, possibilitando a colaboração em tempos e locais distintos. A adaptação de métodos convencionais de forma remota, segundo alguns autores, gera resultados similares [Brush, Ames e Davis 2004][Andreasen *et al.* 2007].

Neste artigo, compartilhamos três casos de avaliação de interfaces de usuário que foram planejados, executados e analisados de forma totalmente remota. Como resultados dessas experiências são apresentados três protocolos de avaliação remota de interfaces de usuário.

2. Fundamentação Teórica

De acordo com Barbosa e Silva (2010), são vários os critérios de qualidade de uso. Nesta seção, a usabilidade e a experiência do usuário são abordadas, assim como estratégias para sua avaliação.

2.1. Usabilidade

A usabilidade está definida na ISO 9244-11 como "O grau em que um produto é usado por usuários específicos para atingir objetivos específicos com eficácia, eficiência e satisfação em um contexto específico de uso." [Barbosa e Silva 2010]. Segundo Nielsen (1993), são cinco os atributos de usabilidade: capacidade de aprendizado, eficiência, capacidade de memorização, erros e satisfação. Para o autor, a usabilidade tipicamente é medida a partir da observação de um número representativo de usuários com apoio de testes de interfaces, com tarefas pré-definidas, mas também pode ser medida a partir de observações de usuários em campo realizando suas tarefas.

A Avaliação Cooperativa propõe a avaliação de interfaces de usuário envolvendo a cooperação entre desenvolvedor e usuário, que exploram um protótipo ou sistema de *software* e desenvolvem uma crítica [Muller, Haslwanter e Dayton 1997]. Melo (2006), ao adaptar o uso da técnica, propõe a realização de tarefas pelo usuário e sua observação pelo desenvolvedor ou responsável pela sessão de avaliação. Nessa adaptação, a observação também pode ser realizada por duas pessoas: uma que interage mais com o usuário e outra que faz anotações. Para cada tarefa realizada pelo usuário, podem ser registrados se ela foi concluída, o tempo envolvido em sua execução, aspectos relacionados às tomadas de decisão do usuário, entre outros. Então, ao final da sessão de avaliação, usuário e observador(es) conversam sobre aspectos negativos e positivos da interface e da experiência, que também devem ser registrados.

O questionário *Usefulness, Satisfaction and Ease of Use* (USE), que significa Utilidade, Satisfação e Facilidade de Uso [Lund 2001], propõe-se a mensurar a usabilidade

de um produto. Esse questionário possui 30 itens que examinam quatro dimensões da usabilidade: utilidade, facilidade de uso, facilidade de aprendizado e satisfação. Essas questões foram construídas numa escala de avaliação Likert de sete pontos, onde os usuários são incentivados a classificar a concordância com as afirmações, variando de discordo totalmente a concordo totalmente. Além disso, os usuários devem citar aspectos mais positivos e os mais negativos do produto.

Além disso, métodos de inspeção de interface, como a Avaliação Heurística de Usabilidade [Nielsen 1993], colaboram à avaliação da usabilidade de interfaces em diferentes graus de fidelidade e funcionalidade. A Avaliação Heurística de Usabilidade possui duas etapas principais: (1) três a cinco especialistas em Interação Humano-computador inspecionam um protótipo ou sistema de *software* e, então, registram os problemas identificados associando-os às heurísticas violadas; (2) reúnem-se os problemas em uma única lista, explicitando as heurísticas violadas, sendo possível atribuir graus de severidade aos problemas para auxiliar na priorização das correções.

2.2. Experiência do Usuário

A experiência do usuário está relacionada aos sentimentos e às emoções de um usuário evocados na utilização de um sistema computacional interativo. Ao se considerar a experiência do usuário como um fator de qualidade, a ideia é projetar interfaces de usuário com características que promovam boas emoções nos usuários, evitando-se sensações desagradáveis [Barbosa e Silva 2010].

Peter Morville (2004) descreve o *design* da experiência do usuário através de sete critérios de qualidade: (1) Útil: a solução deve ser inovadora e útil ao usuário; (2) Utilizável: a aplicação deve ser simples, objetiva e eficiente; (3) Desejável: a aplicação deve possuir elementos do *design* emocional como imagem, identidade e marca; (4) Localizável: a aplicação deve ser navegável e seu conteúdo localizável; (5) Acessível: a aplicação deve permitir que qualquer usuário interaja com a interface do sistema; (6) Confiável: o sistema deve possuir elementos de interface que transmitam segurança ao usuário; (7) Valioso: a aplicação deve entregar valor ao usuário.

Entre os métodos relacionados à avaliação da experiência do usuário, está o SAM (do inglês, *Self Assessment Manikin*). O SAM é um método constituído por imagens, criado para avaliar a emoção de pessoas em relação a eventos ou estímulos. Essa avaliação ocorre através de três dimensões, que possuem uma série de imagens ordenadas de acordo com o seu grau. A dimensão prazer varia de uma figura feliz a uma figura infeliz, a dimensão excitação varia de uma figura animada a uma figura sonolenta e a dimensão dominância varia de uma figura pequena, caracterizando a falta de domínio da situação, a uma figura grande que indica o contrário [Bradley e Lang 1994].

3. Trabalhos Relacionados

Brush, Ames e Davis (2004) comparam a avaliação remota síncrona à avaliação realizada localmente. Todos os participantes avaliaram a mesma interface gráfica do *plugin* UrbanSim, da IDE Eclipse. Alguns participantes foram submetidos aos dois tipos de avaliação para compararem as duas experiências. Os autores apontam que o número médio de problemas encontrados nas duas condições é muito semelhante. Além disso, os participantes demonstraram maior interesse em colaborar para avaliações futuras de forma remota, enquanto que ninguém manifestou interesse em colaborar com avaliações locais.

Andreasen *et al.* (2007) comparam três métodos de teste de usabilidade remoto e um método convencional em laboratório. Entre os testes de usabilidade remota, um dos protocolos envolvia um teste síncrono de modo similar ao método convencional, mas com a coleta de dados realizada com gravação de áudio e de vídeo. Os demais foram conduzidos de modo assíncrono, sendo um realizado com um grupo de usuários e outro com um grupo de especialistas. Em ambos os casos, a apresentação das tarefas e a coleta de dados se deu pela apresentação de um questionário. Os resultados mostraram que o teste remoto síncrono é equivalente ao método convencional, onde foi encontrado quase o mesmo número de problemas de usabilidade e os usuários usufruíram o mesmo tempo para completar as tarefas. Por outro lado, nos testes assíncronos os resultados não foram tão positivos, confirmando que os métodos assíncronos são mais demorados para os usuários e, além disso, auxiliam a identificar menos problemas de usabilidade.

Madathil e Greenstein (2011) apresentam uma abordagem para a realização de estudos de usabilidade síncronos, aplicada a 48 pessoas, sendo 36 participantes e 12 facilitadores, a partir de um laboratório virtual de usabilidade construído com o kit de ferramentas Open Wonderland, que é comparada com outras duas: a abordagem tradicional e WebEx - abordagem com webconferência e compartilhamento de tela. Os dados coletados das três metodologias foram dispostos em três variáveis: identificação dos defeitos; gravidade dos defeitos; tempo gasto nas atividades. O estudo sugere que abordagens virtuais baseadas na realidade geram alegria por parte dos participantes, afetando positivamente o desempenho deles. Além disso, foi apontado que, para a realização de testes síncronos, a realidade virtual do mundo pode ser uma alternativa às tradicionais.

Alcantud *et al.* (2012) propuseram um sistema de gerenciamento de informações relacionadas à avaliação do conteúdo da observação do comportamento humano. A Ferramenta de Monitoramento e Avaliação para Análise Comportamental (do inglês, *Monitoring and Evaluation Tools for Behavioral Analysis* - METBA) permite o tratamento de imagens gravadas tanto em um laboratório de usabilidade como em qualquer outro local controlado. Além disso, a plataforma conta com a integração de dados psicobiológicos obtidos de dados médicos para que se possa realizar uma análise de sinais médicos ao mesmo tempo em que o teste é realizado. O sistema foi projetado para avaliação e monitoramento da análise comportamental humana, permitindo o tratamento de imagens gravadas para análise posterior. Além da captura e reprodução de vídeo com som, o sistema permite a avaliação desses experimentos de qualquer lugar a qualquer hora, podendo ser usado remotamente por diferentes grupos de avaliadores através do acesso à Internet.

Hayashi *et al.* (2016) apresentam o Emoti-SAM *online*, adotado em um experimento realizado em uma escola pública primária para avaliar a experiência de uso de *laptops* pela comunidade escolar. Nessa versão do SAM, as figuras do SAM original foram substituídas por *emojis* ou *emoticons* semelhantes aos utilizados em redes sociais. O instrumento adota a escala de cinco pontos, mantendo as dimensões prazer, excitação e dominância.

Petrovica e Ekenel (2016) apresentam uma adaptação do método SAM, denominada AffectButton, desenvolvida como um botão simples que representa um rosto que permite ao usuário informar a emoção que está sentindo. Essa adaptação foi adotada na avaliação de emoções de alunos na interação com sistema de tutoria durante seu processo de aprendizagem. O AffectButton adota as mesmas dimensões do SAM: prazer, excitação e

dominância. O usuário pode selecionar uma variedade de valores afetivos dessas dimensões, movendo o *mouse* dentro do botão (a expressão muda conforme o *mouse* é movimentado), escolhendo a expressão desejada e clicando para selecioná-la [Broekens e Brinkman 2013].

Rocha e Prazeres (2017), ao avaliarem a usabilidade do protótipo RDFaLiveExtension, uma extensão para navegador *web*, que implementa o modelo LDoWPaN de navegação e de apresentação de dados do tipo Linked Data com ênfase no usuário final, realizam a adaptação do questionário USE, desenvolvido por Lund (2001). Para conduzir a avaliação, de forma remota, o questionário foi disponibilizado na *web*. Os autores também disponibilizaram um tutorial *online* para orientar os 25 usuários convidados a respeito da instalação do protótipo. Após o questionário ter sido respondido, os resultados foram organizados em gráficos de barras empilhados - um para cada dimensão de usabilidade - e, então, analisados.

Em síntese, alguns desses métodos propõem adaptação de abordagens convencionais com apoio de tecnologias digitais de informação e comunicação para acompanhamento e registro. Outros são conduzidos de forma semiautomatizadas, gerando e organizando dados para análise posterior. Conferem, assim, maior flexibilidade, possibilitando a colaboração em tempos e locais distintos. Contudo, os autores não apresentam os protocolos subjacentes de modo a facilitar sua adoção por outros avaliadores. Além disso, Brush, Ames e Davis (2004) e Andreasen *et al.* (2007) indicam que a adaptação de métodos convencionais de forma remota gera resultados similares.

4. Estudos de Caso

Este trabalho está baseado em uma abordagem qualitativa de pesquisa, de caráter exploratório [Triviños 2011]. Envolve a observação e a interpretação das experiências de avaliação de interface de usuário no contexto do projeto de ensino GEIHC – Grupo de Estudos em Interação Humano-computador do *Campus* Alegrete da Universidade Federal do Pampa. Nesta seção são apresentadas três avaliações realizadas de maneira totalmente remota com adaptações de métodos e técnicas apresentados na seção de fundamentação teórica deste artigo. As experiências de avaliação síncronas, ou seja, realizadas no mesmo momento, foram apoiadas pela ferramenta de comunicação Google Meet, substituindo a interação face a face.

4.1. Avaliação de protótipos de aplicação *mobile* para apoiar estudantes universitários

A aplicação *mobile* proposta serve para apoiar os estudantes universitários no gerenciamento de suas tarefas, onde eles podem adicionar e organizar componentes curriculares que estão cursando. Além disso, conseguem se informar sobre a disponibilidade de Atividades Complementares de Graduação presentes em uma agenda eletrônica.

Sobre protótipos dessa ferramenta, foram realizadas avaliações visando identificar aspectos de usabilidade a melhorar. A primeira avaliação se deu através de um grupo de foco, que contou com a participação de 11 estudantes de graduação do *Campus* Alegrete da Unipampa. O encontro com esses estudantes tinha como propósitos validar os requisitos e avaliar a usabilidade do protótipo de alta fidelidade com apoio do questionário USE [Lund 2001], adaptado com auxílio da ferramenta Google Forms. Nesse formulário, além de informações pessoais dos participantes, foram apresentadas as questões presentes no

instrumento USE. Para a análise dos resultados, foram consideradas as classificações entre 5 e 7 como “boa aceitação”, entre 1 e 3 como “rejeição” e 4 como “aceitação neutra”.

Na primeira dimensão (utilidade) da avaliação remota, para os diferentes quesitos de avaliação, 65% a 100% dos usuários relataram boa aceitação, enquanto 9% a 35% indicaram aceitação neutra. Na segunda dimensão (facilidade de uso), cerca de 80% a 100% apontaram uma boa aceitação e apenas 9% a 20% relataram aceitação neutra. Na terceira dimensão (facilidade de aprendizagem), quase todos os usuários classificaram o protótipo como fácil de aprender. Na última dimensão (satisfação), mais de 90% dos usuários relataram uma boa aceitação. Entre os aspectos positivos, estão a organização das atividades na palma da mão de forma fácil e melhor produtividade em relação às atividades acadêmicas. Entre os aspectos negativos, muitos apontaram que não puderam identificá-los e apenas um estudante revelou estar com dúvidas sobre como algumas funcionalidades se comportariam.

Após ajustes no protótipo, uma nova sessão de avaliação foi conduzida, desta vez com apoio do protocolo da Avaliação Heurística de Usabilidade [Nielsen 1993]. Participaram dessa avaliação quatro estudantes que já conheciam o método e integram o projeto de ensino GEIHC. Na primeira etapa da avaliação, foram apresentadas as Heurísticas de Usabilidade, o protótipo funcional, assim como foi distribuída uma planilha para cada avaliador registrar problemas. Na segunda etapa da avaliação, os avaliadores compartilharam os problemas identificados na primeira etapa, gerando uma única lista de problemas e com os respectivos graus de severidade. Nesse processo de avaliação, identificaram-se onze problemas de usabilidade.

4.2. Avaliação Heurística de Usabilidade do protótipo e-SAM

O e-SAM é uma solução para a aplicação da técnica SAM na avaliação do estado afetivo do usuário na interação com sistemas computacionais. Essa ferramenta permite a criação de avaliações utilizando o método SAM em sua forma original, assim como a configuração de suas dimensões, escalas e imagens para utilizá-lo de forma adaptada. Essas avaliações podem ser impressas ou disponibilizadas aos participantes através de um *link*. Antes de sua implementação, foi desenvolvido um protótipo de alta fidelidade do sistema para definir as características da sua interface.

Uma Avaliação Heurística de Usabilidade [Nielsen 1993] foi aplicada a esse protótipo, contando com a participação de três avaliadores do GEIHC. Na primeira etapa, foi apresentado o protótipo e foram distribuídas três planilhas a cada avaliador. Na primeira planilha, os avaliadores deveriam indicar os problemas encontrados, identificando, para cada problema, a(s) tela(s) do protótipo em que o encontraram, as heurísticas relacionadas e um grau de severidade. Na segunda e terceira planilhas foram disponibilizadas, respectivamente, a descrição das Heurísticas de Usabilidade de Nielsen e a descrição dos graus de severidade. Na segunda etapa, os avaliadores eram convidados a dissertar sobre os problemas encontrados enquanto exibiam o protótipo do sistema no modo de compartilhamento de tela. Durante essa dinâmica, os avaliadores chegaram a um consenso sobre como enunciar os problemas e quanto ao grau de severidade associado a cada um. Ao final, chegou-se a uma lista de trinta problemas.

4.3. Avaliação do *site* do *Campus* Alegrete da Unipampa

Em resposta a uma solicitação institucional ao GEIHC, duas estratégias foram adotadas, de forma remota, para a avaliação da usabilidade do *site* do *Campus* Alegrete da Unipampa com a participação de estudantes universitários: uma Avaliação de Usabilidade, com uma adaptação do questionário USE [Lund 2001]; e uma adaptação da Avaliação Cooperativa [Muller, Haslwanter e Dayton 1997], com a colaboração de usuários que demonstraram interesse em colaborar após responderem ao questionário.

Estudantes, convidados através de lista de *e-mails*, responderam ao questionário USE no período de uma semana. No questionário, organizado na plataforma Google Forms, além das questões presentes no instrumento USE, foram solicitadas informações pessoais, sugestões de melhoria e realizada uma pergunta fechada sobre o interesse em colaborar em outras etapas de avaliação. É importante destacar que a matrícula foi solicitada a fim de explicitar dados sobre o público trabalhado, já que na Universidade ela indica o ano de ingresso de um estudante. Além disso, junto às instruções para adesão ao questionário, foram apresentadas informações para o consentimento livre e esclarecido.

Ao todo, obtiveram-se 47 respostas, de estudantes dos diferentes cursos do *Campus*, com ingresso em diferentes anos, sendo o mais antigo em 2011. Pôde-se perceber que a dimensão mais bem avaliada foi a facilidade de aprendizagem e a pior foi a satisfação. Além disso, houve significativa contribuição nos questionamentos dissertativos. Ao todo, 33 respondentes mencionaram ter interesse em continuar colaborando com a avaliação do *site*. Destes, 14 foram convidados a colaborarem em uma adaptação da Avaliação Cooperativa.

Para a condução da Avaliação Cooperativa, o grupo de avaliadores se organizou em duplas, adaptou seu protocolo para condução *online*, de forma síncrona, e formalizou os convites pelo envio de *e-mails*, com orientações sobre o processo de avaliação e o termo de consentimento livre e esclarecido. Dentre os convidados, cinco participaram das sessões síncronas de avaliação, que duraram entre 20min e 40min cada e foram gravadas.

No início de cada sessão de avaliação, na qual participou um usuário e dois avaliadores, a atividade foi contextualizada e explicada. Os participantes compartilharam suas telas com os avaliadores e foram convidados a realizarem cinco tarefas. Cada tarefa, em sua vez, era lida em voz alta por um dos avaliadores e disponibilizada no *chat* da ferramenta. O tempo de sua realização era cronometrado. Durante o desenvolvimento da tarefa, era solicitado ao usuário que narrasse cada uma de suas ações. Ao término da realização de uma tarefa ou esgotado o tempo de 4min, o usuário era convidado a expressar seu estado afetivo respondendo ao artefato SAM original, em cinco escalas, adaptado em um formulário *online*. Então, passava à próxima tarefa até que todas fossem realizadas. Após realizar as cinco tarefas, o usuário e os avaliadores conversaram sobre a experiência de avaliação, com apoio do seguinte roteiro: (1) De modo geral, considerando as tarefas realizadas, quais suas impressões sobre o *site* do *Campus* Alegrete da Unipampa?; (2) Há algo que gostaria de destacar sobre a experiência de avaliação do *site*?; (3) Mencione aspectos negativos do *site*; (4) Mencione aspectos positivos do *site*.

Pôde-se observar, durante a sessão de avaliação, que uma das tarefas se tornou confusa em função da organização das informações no *site*, que não respeitava o modelo mental do usuário. Além disso, ao tentarem utilizar o mecanismo de busca do *site*, este não foi eficaz, o que levou alguns usuários a utilizarem ferramenta de busca fora do domínio da

instituição. Ademais, quando uma tarefa não era concluída, percebia-se a sensação de falta de domínio da situação, o que gerava frustração no usuário.

5. Resultados e Discussões

Inicialmente, o recrutamento de avaliadores e usuários ocorreu, respectivamente, por grupo de Whatsapp e por *e-mail*. A principal modificação realizada, contudo, foi a adoção de uma ferramenta de conferência *online* – o Google Meet – para apoiar o desenvolvimento da Avaliação Heurística de Usabilidade e da Avaliação Cooperativa. Observa-se, ainda, que na Avaliação Heurística de Usabilidade, foi adotada planilha eletrônica compartilhada remotamente com os avaliadores. Já a sessão de Avaliação Cooperativa foi gravada com o recurso da ferramenta de comunicação, as tarefas foram lidas em voz alta e apresentadas no *chat* da ferramenta, as ações dos usuários foram narradas por eles e o método SAM foi utilizado ao final da realização de cada tarefa.

Embora tenha havido mediação por ferramenta de conferência *online* e documentos compartilhados, as experiências de avaliação se aproximaram bastante das experiências face a face desenvolvidas no contexto do GEIHC e em sala de aula, no componente curricular Interação Humano-computador. Contudo, na aplicação da Avaliação Cooperativa, a impossibilidade de contato face a face com possíveis colaboradores pode explicar a baixa adesão pelos convidados em sua adoção remota.

A seguir, são apresentados três protocolos de avaliação remota de interfaces de usuário, documentados a partir das experiências dos estudos de caso reportados, com o objetivo de facilitar sua replicação. Tais protocolos podem ser empregados em atividades de Verificação e Validação no desenvolvimento de *software.*

5.1. Avaliação Heurística de Usabilidade *Online*

Resumo: Apoiado por planilhas eletrônicas, três a cinco avaliadores inspecionam uma interface de usuário e elaboram uma lista de problemas, associando-os a heurísticas de usabilidade e graus de severidade. Modelo de objeto: protótipo em diferentes níveis de fidelidade e funcionalidade ou sistema de *software*. Modelo de processo: O responsável pela sessão de avaliação apresenta o protocolo e rememora as heurísticas de usabilidade. Cada avaliador, com auxílio das Heurísticas de Usabilidade de Nielsen, explora o protótipo ou sistema de *software*. Individualmente, apoiado por uma planilha eletrônica, registra os problemas identificados, associando-os a heurísticas. Posteriormente, em grupo, os avaliadores compartilham os problemas identificados e geram uma única lista de problemas de usabilidade associados a heurísticas e graus de severidade. Observa-se que um problema deve ser enunciado de forma clara, inclusive com a indicação de sua localização e/ou abrangência. Resultados: lista de problemas de usabilidade associados a Heurísticas de Usabilidade e graus de severidade. Métodos formais complementares: Avaliação Heurística de Usabilidade. Tamanho do grupo: 3-5 avaliadores. Referência: Nielsen (1993).

5.2. SAM *Online*

Resumo: Apoiado por um formulário *online*, após o usuário realizar tarefas ou explorar um protótipo ou sistema de *software*, este pode responder ao instrumento SAM (do inglês, *Self Assessment Manikin*) em relação à experiência vivenciada. Modelo de objeto: protótipo ou sistema de *software*. Modelo de processo: Usuário explora ou utiliza um protótipo ou sistema de *software* e responde ao instrumento SAM, apresentando sua resposta afetiva ao

sistema relacionada a três dimensões: prazer, excitação e dominância. Resultados: resposta afetiva do usuário a um protótipo ou sistema de *software*. Métodos formais complementares: Teste de Usabilidade. Tamanho do grupo: 2, sendo 1 usuário e 1 avaliador. Referências: Bradley e Lang (1994), Nielsen (1993).

5.3. Avaliação Participativa Remota

Resumo: Apoiado por uma ferramenta de conferência *online* e um formulário *online*, um usuário explora um protótipo ou sistema de *software* guiado por tarefas. Através do compartilhamento da tela do usuário, esse uso é observado por um ou dois avaliadores. Ao final da realização das tarefas, o usuário submete o formulário com registro de sua resposta afetiva para a realização de cada tarefa. Modelo de objeto: protótipo funcional ou sistema de *software*. Modelo de processo: Os avaliadores propõem um conjunto de tarefas para ser realizado pelo usuário, atribuindo um tempo máximo à execução de cada uma delas. Os avaliadores configuram o instrumento SAM para registro da resposta afetiva do usuário à cada tarefa proposta para a sessão de avaliação. Um dos avaliadores explica a proposta da sessão de avaliação ao usuário, solicita que compartilhe sua tela e, com sua anuência, inicia a gravação da sessão. Enquanto um avaliador interage mais com o usuário, o outro observa e realiza anotações durante toda a sessão de avaliação. Um dos avaliadores compartilha o instrumento SAM com o usuário e o instrui a preencher suas informações pessoais. Em seguida, conduz o usuário à tela inicial do sistema em avaliação e enuncia a primeira tarefa, que também fica registrada no *chat* do sistema de conferência *online*. A partir desse momento, o tempo de execução da tarefa é cronometrado. Ao usuário, é solicitado que narre o desenvolvimento da tarefa. Encerrada a execução da tarefa ou o tempo previsto para sua conclusão, um dos avaliadores registra o tempo transcorrido e interrompe o usuário. Então, o usuário preenche o instrumento SAM, indicando sua resposta afetiva à realização da tarefa. Enquanto houver tarefas a serem executadas, repete-se esse processo, desde a condução do usuário à tela inicial até a resposta ao instrumento SAM. Após a realização de todas as tarefas, é solicitado ao usuário o envio do instrumento SAM. Finalmente, usuário e avaliadores conversam sobre impressões a respeito do protótipo ou sistema de *software*, a experiência de avaliação, aspectos negativos e positivos do protótipo ou sistema de *software*. Resultados: para cada tarefa, seu *status* de realização (ex.: não realizada, tempo transcorrido) e resposta afetiva do usuário; crítica a um protótipo ou sistema de *software*; vídeo para análise posterior; anotações dos avaliadores. Métodos formais complementares: Teste de Usabilidade. Tamanho do grupo: 2-3, sendo 1 usuário e 1 ou 2 avaliadores. Referências: Melo (2006), Muller, Haslwanter e Dayton (1997), Nielsen (1993).

6. Considerações Finais

O período pandêmico forçou uma nova abordagem à realização de avaliações de interface de usuário relacionadas a trabalhos de conclusão de curso e projeto de ensino do *Campus* Alegrete da Unipampa. A abordagem adotada pelos envolvidos foi a adaptação de métodos existentes para viabilizar sua aplicação de forma remota.

Neste artigo, foram reportados três casos de adaptações de diferentes estratégias de avaliação. Os resultados reportados baseiam-se nas experiências de avaliação no contexto do GEIHC. Para facilitar que essas experiências possam ser reproduzidas, os protocolos com suas adaptações foram documentados.

Para aplicação do método SAM, em particular, foi criado um instrumento *online* específico para o caso que o adotou. Conforme apresentado, está em desenvolvimento uma

ferramenta de apoio à sua realização de forma *online*, de modo que o instrumento possa ser utilizado em outros estudos. Tem-se, ainda, em perspectiva a revisão sistemática do estado da arte e do estado da prática em avaliação remota de interfaces de usuário, síncronas e assíncronas, com apoio de tecnologias digitais de informação e comunicação.

Referências

Alcantud, F., Coret, J., Jiménez, E. *et al.* (2012). "Usability remote evaluation: METBA system". In *ICL 2012*, 1-8, Villach.

Andreasen, M. S., Nielsen, H. V., Schrøder, S. O., Stage, J. (2007). "What happened to remote usability testing? an empirical study of three methods." In CHI '07, 1405–1414, San Jose.

Barbosa, S.; Silva, B. (2010), Interação Humano-computador, Elsevier Brasil, 1. ed.

Bradley, M. M., Lang, P. J. (1994) Measuring emotion: the self-assessment manikin and the semantic differential. *Journal of behavior therapy and experimental psychiatry*, p. 49–59, v. 25, n. 1.

Broekens, J., Brinkman, W. P. (2013) Affectbutton: A method for reliable and valid affective self-report. *International Journal of Human-Computer Studies*, p. 641–667, v. 71, n. 6.

Brush, A. J. B., Ames, M., Davis, J. (2004). "A comparison of synchronous remote and local usability studies for an expert interface". In *CHI EA 04*, 1179–1182, Vienna.

Hayashi, E. C. S., Posada, J. E. G., Maike, V. R. M. L. (2016) "Exploring new formats of the self-assessment manikin in the design with children". In: *IHC'16*. 1–10, São Paulo.

Hewett, T. T., Baecker, R., Card S. *et al.* (1992) "ACM SIGCHI Curricula for Human-Computer Interaction", In: ACM SIGCHI Report, ACM, NY.

Lund, A. M. (2001) Measuring usability with the use questionnaire. *Usability interface*, p. 3-6, v. 8, n. 2.

Madathil, K. C., Greenstein, J. S. (2011). "Synchronous remote usability testing: a new approach facilitated by virtual worlds". In CHI 11, 2225–2234, Vancouver.

Melo, A. M., Baranauskas, M. C. C. (2006). "Uma Opção Inclusiva à Avaliação Cooperativa de Interfaces de Usuário". In *SEMISH CSBC 2006*.

Morville, P. (2004) "User Experience - Design. Semantic Studios", http://semanticstudios.com/user_experience_design/.

Muller, M. J., Haslwanter, J. H., Dayton, T. (1997) "Participatory Practices in the Software Lifecycle", Handbook of Human-Computer Interaction, Elsevier.

Nielsen, J. (1993), Usability Engineering, Morgan Kaufmann.

Petrovica, S.; Ekenel, H. K. (2016) "Emotion recognition for intelligent tutoring. In *BIR-WS 2016*, Prague.

Rocha, A., Prazeres, C. (2017) LDoW-PaN: Linked Data on the Web—Presentation and Navigation. *ACM Trans. Web* 11, 4.

Triviños, A. N. S. (2011), Introdução à pesquisa em ciências sociais: a pesquisa qualitativa em educação, Atlas.

Towards a Process for Migrating Legacy Systems into Microservice Architectural Style

Daniele Wolfart[1], Ederson Schmeing[1], Gustavo C.L. Geraldino[1], Guilherme L.D. Villaca[1], Diogo do N. Paza[1], Diogo C.P. Domingos[1], Wesley K.G. Assunção[2,1], Ivonei F. da Silva[1], Victor F.A. Santander[1]

[1]PPGComp – Western Paraná State University (UNIOESTE). Cascavel, Brazil.

[2]COTSI – Federal University of Technology - Paraná (UTFPR). Toledo, Brazil.

{danielewolfart, edersonschmeing, gclgeraldino, guidvillaca, diogopaganinidomingos, diogopazacvel}@gmail.com, wesleyk@utfpr.edu.br, {ivonei.silva, Victor.Santander}@unioeste.br

Abstract. *Microservice architectural style is a paradigm to develop systems as a suite of small and autonomous services, communicating thought a lightweight protocol. Currently, one of the most common ways of adopting microservice architectures is by the modernization of legacy monolith systems. The migration of a legacy system into a microservice architecture is not a trivial task. In addition, there is a lack of recommendation or guidelines on how to perform such process. In view of this, this paper presents a preliminary process for conducting the migration of legacy systems into microservice architectures. This process was defined by analyzing and discussing pieces of work on the topic. As result, we propose a process composed of eight steps, grouped in four phases, which we describe together with their common input and output.*

1. Introduction

The majority of industrial systems are long-lived applications, i.e., legacy system, usually having a decayed and degraded monolithic architecture [Lewis et al. 2003]. In order to remain competitive, these monolithic legacy systems must be modernize. Nowadays we can observe a trend on migrating legacy systems into microservice architectures [Knoche and Hasselbring 2018]. The microservice architectural style is a paradigm where a system is a suite of small and autonomous services that work together [Newman 2015]. The benefits of adopting microservices are: reduced effort for maintenance and evolution, increased availability of services, ease of innovation, continuous delivery, ease of DevOps incorporation, and facilitated scalability [Taibi et al. 2017].

We can find in the literature mappings and reviews in the topic of migrating legacy system into microservices [Ponce et al. 2019, Di Francesco et al. 2019], classification of refactoring approaches for this migration [Fritzsch et al. 2018], industrial reports and surveys with practitioners [Carvalho et al. 2019b, Fritzsch et al. 2019, Di Francesco et al. 2018]. These papers describe, respectively, approaches to conduct the migration, refactoring approaches for deal with implementation artifacts, and experience reports on how migrations were applied in practice. Despite the grown of interest in migrating legacy systems into microservices, in the literature, to the best of our knowledge, there is no study that presents recommendations/guidelines on how to perform the entire migration process, covering activities of legacy comprehension, architecture definition, execution of the migration, and microservices monitoring.

The goal of this paper is to define a preliminary process for the migration of legacy systems into microservice architectures. For the definition of such process, we analyzed and extensively discussed a set of studies describing migration processes. Our study (presented in details in Section 3) was conducted in a subject of a master course with six students and one professor. The resulting process (described in Section 4) is composed of eight steps, grouped in four phases. The phases are: comprehension of the monolithic legacy system, definition of the new architecture, execution of the transformation, and monitoring after the migration. In addition, we also provide the observed input and output for each step. The contribution of preliminary process defined in our study (differences from related work are discussed in Section 2) is to serve as basis for those ones envisaging the conduction of a migration process.

2. Related work

In this section, we discuss pieces of work that are related to our study. In special the studies presented in [Mayer and Weinreich 2018], [Balalaie et al. 2018], [Chen et al. 2018], [Hassan et al. 2017], [Taibi et al. 2017], [Carvalho et al. 2019b], and [Ahmadvand and Ibrahim 2016].

Ahmadvand and Amjad Ibrahim present a methodology for microservice decomposition from system requirements. Security and scalability requirements are input for the decomposition. This methodology is base for architectural design decisions [Ahmadvand and Ibrahim 2016]. This approach is related to the steps one, two, and three of our proposed process. Balalaie *et al.* describe fifteen patterns to migrate a monolithic system into a set of microservices considering the variation of several factors such as requirements, current situation and skills of team members. The set of migration patterns focus on several steps in our proposed process, although the authors do not guarantee the completeness of patterns repository to realize the migration [Balalaie et al. 2018]. Chen *et al.* use the data flow diagram (DFD) to capture the business requirements of a monolithic system, then, an algorithm combines the same operations with the same type of output data into a decomposable data-flow, finally, a function is run to identify microservices candidates from the decomposable DFD [Chen et al. 2018]. This approach is related to the steps two and three of our proposed process. Hassan *et al.* describe an architecture-centric approach to model microservice granularity. They extend the *ambient* concept [Ali et al. 2010] by introducing *microservice ambient* that models microservices and uses *aspects* to support changes in granularity at runtime. As a result, this approach can provide architectural modeling to aid the candidate solution for granularity adaptation [Hassan et al. 2017]. This approach is related to the steps three and four of our process.

Taibi *et al.* after performing a survey with practitioners that migrate monolithic systems to microservices, describe a process framework adopted by the practitioners during the migration. This framework contains activities to migrate an existing monolithic system to a microservice from scratch. There is also activities to implement new features as microservices to replace gradually the existing system. This approach is related to the all steps of our proposed process, although the manuscript does not present details [Taibi et al. 2017]. Carvalho *et al.* also conducted a survey with practitioners and describe adopted criteria to extract microservices on monolithic systems. Requirements, modularity, cohesion, coupling, database scheme, visual models, and reuse were criteria more important mentioned by practitioners participants of the study [Carvalho et al. 2019b].

This approach is related to the step one and three of our proposed process. Mayer and Weinreich define an approach for continuously extracting the architecture of REST-based microservice software systems. The approach retrieves static and dynamic data from different involved and distributed microservices to overview the software architecture and supervise it over time [Mayer and Weinreich 2018]. This approach is related to the step eight of our proposed process.

Our study presented in this paper differs from these mentioned approaches by describing an entire process, capturing common steps, input and output. Therefore, the proposed process in this paper has the goal broader. Despite of existing secondary studies such as [Di Francesco et al. 2019, Fritzsch et al. 2018, Ponce et al. 2019, Francesco et al. 2017], which map primary studies describing characteristics of the migration, they do not describe a consolidated and comprehensive process. So, to the best of our knowledge, our study is the first one that presents recommendations and/or guidelines on how to perform the migration process.

3. Study Design

This section presents details of the methodology of our study.

3.1. Participants

The study was conducted by six master students and supervised by one professor, which is a researcher on the topic of this study. The students were enrolled in a course of Requirements Engineering from the Graduate Program X[1] at University Y[1]. The professor had experience on conducting research on migrating legacy systems to microservice architectures. As part of the course, the professor taught four classes, summing 3h30min, about the migration of legacy systems to microservices. These classes were designed to serve as a background for the students. In addition, the professor assisted the students and participated in all the activities of the study.

3.2. Primary Sources

The source of information to define the preliminary process were obtained from searches on Google Scholar[2] with different combinations of the keywords "Migration", "Microservices", "Legacy Systems", and "Monolith". The primary sources are: [Balalaie et al. 2018, Chen et al. 2018, Taibi et al. 2017, Ahmadvand and Ibrahim 2016, Mayer and Weinreich 2018, Hassan et al. 2017]. These papers were selected based on their content, which is strictly related to the goal of our work.

3.3. Data Extraction

Next we describe the steps we followed to collect relevant information from the papers and the discussions performed to define the preliminary process.

1. **Individual reading**: each paper was assigned to one student, which had one week to read the paper and answer the following questions: "*What are the driving forces to adopt a microservice architecture?*", "*What are the advantages and disadvantages of adopting microservices?*", and "*What does the paper describe about migrating a legacy system to a microservice architecture?*".

[1]Omitted due to double blind review.

[2]`https://scholar.google.com/` on November 12th, 2019.

2. **Individual presentation and group discussion**: the students individually presented their answers for the previous questions. During the presentation, all other students and the professor could ask questions, complement affirmations, or relate the information presented with information collected by other student. This activity took 3h30min.
3. **Papers filtering and pair reading**: after the individual presentations, we selected only three papers that were more related to the topic of migrating legacy systems to microservices. These papers were: [Balalaie et al. 2018, Chen et al. 2018, Taibi et al. 2017]. The professor defined pairs of students and assigned one of the selected papers to each pair. The pairs were asked to answer the following questions: "*Which are the steps to perform the migration of legacy systems to microservices?*" and "*What are the common input and output artifacts of each step?*". They had one week to read the papers and answer the questions.
4. **Pair presentation**: each pair of students presented the steps they found in the paper, as well as the input and output for each step. A open discussion was also allowed, similarly to Step 2. For the presentations and to support the discussion, we used a whiteboard, that was split in three parts, where each pair of students could draw the information they collected. Figure 1[3] presents the whiteboard after the presentation of the three pairs of students.
5. **Information merging**: the last step of our study was merging all the information presented by the students and discussed in group. For this, we relied on the drawing on the whiteboard (Figure 1). During the discussion for merging the initial processes found by each pairs, the students were asked to take notes. The resulting process after merging all information, is described in the next section.

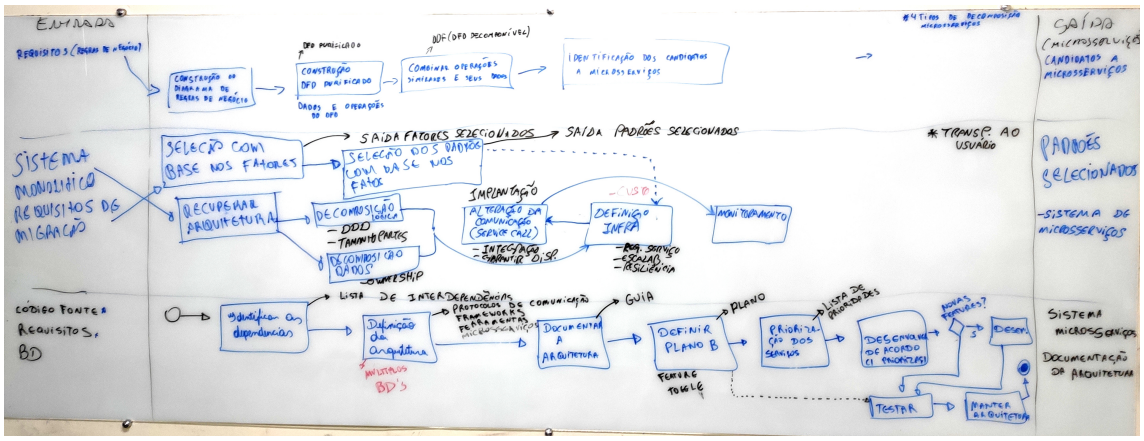


Figure 1. Whiteboard with the migration process identified the pairs of students.

4. Proposed Preliminary Process

After conducting the study presented in the previous section, we defined a preliminary process for migrating monolith legacy systems to microservice architectures. This preliminary process has four phases and a total of eight steps. Figure 2 presents the defined process, together with the input and output of each step. We stress here that this is a high-level process, and each step can be decomposed in lower level steps/tasks. In the next subsection we describe each step in details.

[3]The text in the whiteboard is in Portuguese, the language used in the course.

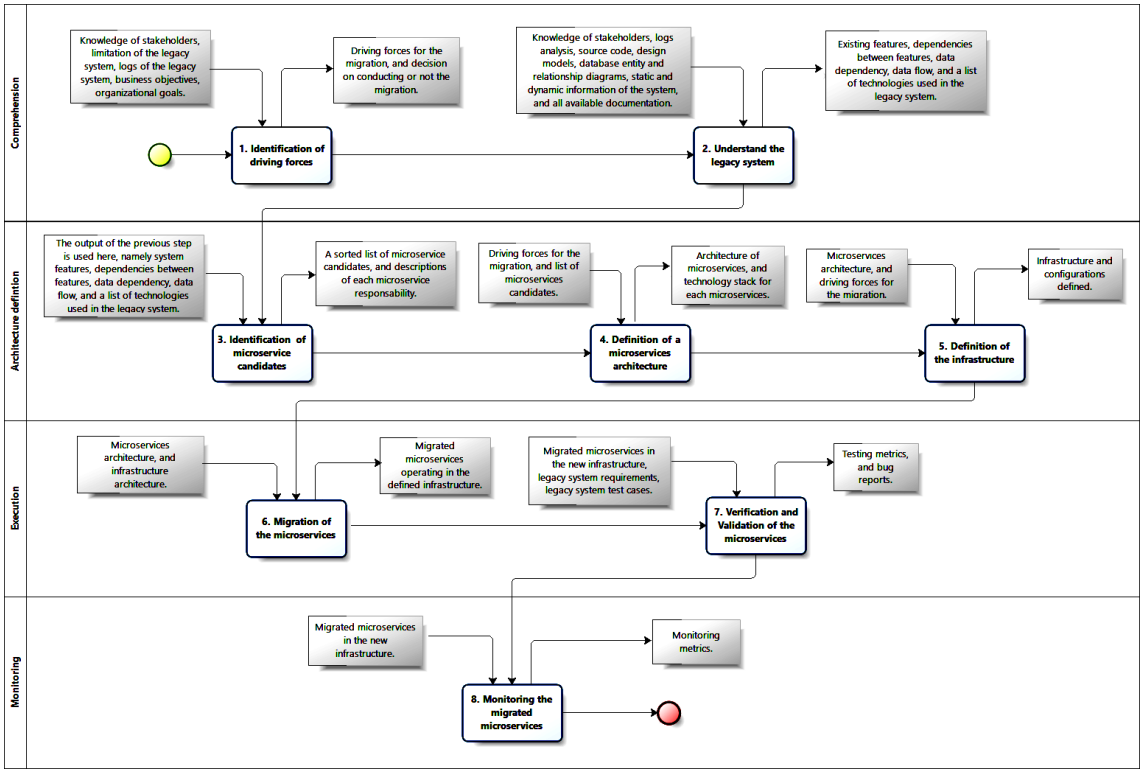


Figure 2. Process for migrating legacy systems to microservice architecture.

4.1. Phase 1: Comprehension the monolithic legacy system

This phase is responsible for providing a understanding of the legacy system and the needs on migrating it to a microservice architecture, i.e. the driving forces for the migration.

- **Step 1: Identification of driving forces**. The goal of this step is to identify the forces, requirements, needs, or objectives that motivate the migration. These forces can be related to technological, organizational, and/or business aspects. For example, in the primary sources we observe driving forces for the migration related to easier maintainability, allow scalability, easier management of development teams, moving towards adoption of DevOps [Taibi et al. 2017]. Other authors mention as motivation the opportunity to use of different technologies, reduced time-to-market, and better modularization [Balalaie et al. 2018].
  - *Input*: knowledge of stakeholders, limitation of the legacy system, logs of the legacy system, business objectives, organizational goals.
  - *Output*: driving forces for the migration, and decision on conducting or not the migration.
- **Step 2: Understand the legacy system.** This step is dedicated to an analysis of the actual legacy system. The goal is to understand the source code organization, existing features, dependencies among features, database structure, and all available documentation. During this step, the responsible for the migration also need to consider: (i) the business characteristics and goals such as keeping the company innovative and competitive; (ii) existing technologies related to programming languages, database management system, and software tools used; and (iii) organization aspects such as size of development teams, maturity and expertise of the

developers, development process adopted in the company, etc. The legacy system must be well-understood. In addition, the behavior of the systems and data flow within the features either [Chen et al. 2018].

  - *Input*: knowledge of stakeholders, logs analysis, source code, design models, database entity and relationship diagrams, static and dynamic information of the system, and all available documentation.
  - *Output*: existing features, dependencies between features, data dependency, data flow, and a list of technologies used in the legacy system.

4.2. Phase 2: Definition of the new architecture

The new architecture depends on what was chosen as migration goal. An incremental migration or big bang migration. In the former case, the architecture will be hybrid, having both the legacy system parts and the new microservices that will replace other of its parts. In the case of a big bang migration, an entire new microservice architecture should be defined. In addition to the architecture of the system under migration, the engineers need to define the infrastructure where the microservices will operate.

- **Step 3: Identification of microservice candidates**. This step has the goal of decomposing the legacy system in units that will be transformed in microservices. At first, the engineers should define the tasks each microservices will be responsible for, indirectly defining the size of the microservices in the new architecture [Hassan et al. 2017]. Based on the importance of defining the responsibility and size of a microservice, this step is a crucial for the migration process. The identification of microservice candidates can be performed by investigating the coupling and cohesion among parts of the legacy system, considering the migrating whole features to a microservice architecture, or focusing on migrate most reused parts of the legacy [Carvalho et al. 2019b]. A factor that must be taken into account during the identification of potential microservices is the overhead in communication that will be includes when decoupling to dependent task. In a monolithic system, all the communication among units are based on direct memory access. However, microservices use the network for the communication, having great impact in the time execution. For an incremental migration, Balalaie et al. recommend starting with a fewer number of microservices and incrementally perform the migration of new microservices, as the system grows and/or the development team acquire a better understanding of microservices characteristics [Balalaie et al. 2018].
  - *Input*: the output of the previous step is used here, namely system features, dependencies between features, data dependency, data flow, and a list of technologies used in the legacy system.
  - *Output*: a sorted list of microservice candidates, and descriptions of each microservice responsibility.
- **Step 4: Definition of a microservice architecture**. For example, here engineers have to describe the APIs for the microservices, define the communication protocol, and choose a strategy of services discovering. The architecture will be greatly affected by the decision of conducting an incremental or a big bang migration. In case of an incremental migration, the decisions on how integrate the microservices with legacy systems should be defined. For that, we observe the use of

Feature Toggles [Carvalho et al. 2019a], which ease to revert the use of the legacy system in case of any problem with the new microservices, allowing a better transition. Choosing a big bang migration, engineers have to figure out how to move all the legacy to microservices at once. A challenge with the big bang migration is to define a complete architecture beforehand. During the construction of this entire architecture some maintenance in the legacy during the migration should be updated in the new architecture, delaying the migration [Casey et al. 2017]. Another aspect that must be taken into account is the microservicification of the data. Commonly, legacy systems rely in one central database [Taibi et al. 2017]. But in the microservices context engineers can migrate the data of a microservice handle to a specific database managed only by this microservices and all access for such data is done by API requests. A central shared database also can be kept, however, this lead to data coupling among microservices, making complex activities of maintenance. In addition, it also affects the interdependence among different teams on charge of specific microservices.

  - *Input*: driving forces for the migration, and list of microservices candidates.
  - *Output*: architecture of microservices, and technology stack for each microservices.

- **Step 5: Definition of the infrastructure**. This step is devoted to define the environment where the microservices will operate. Here the engineers should decide about having a local hosts for the microservices or rent a cloud platforms such as Amazon Web Services[4], Microsoft Azure[5], or Google Cloud[6]. The infrastructure must provide resiliency and high availability in microservices, allowing management of scalability.
  - *Input*: microservice architecture, and driving forces for the migration.
  - *Output*: infrastructure and configurations defined.

4.3. Phase 3: Execution of the transformation

This phase is devoted to execute the migration considering all the planning of previous phases.

- **Step 6: Migration of the microservices**. Here the transformation of the implementation artifacts is done to move the parts of the legacy system to microservices. The infrastructure should be configured and be available for deploying the microservices. As we aforementioned, feature toggle[7] can be used as strategy to incrementally integrate the microservices with the legacy system, in a incremental migration.
  - *Input*: microservice architecture, and infrastructure architecture.
  - *Output*: migrated microservices operating in the defined infrastructure.

[4] `https://aws.amazon.com/`

[5] `https://azure.microsoft.com/`

[6] `https://cloud.google.com/`

[7] Feature Toggle is basically a flag variable used in a conditional statement with code blocks. Their goal is either enabling or disabling the feature code in those blocks [Rahman et al. 2016]

- **Step 7: Verification and Validation of the microservices**. After moving parts of the legacy systems to a microservice architecture, regression tests must be performed in order identify possible bugs. We highlight the importance of having proper test cases before starting the migration. Such tests are intend to identity bugs introduced during the migration and also confirm if the requirements of the migrated parts are adequate. Once interdependent microservices are migrated, integration tests are also required. For this step of verification and validation of the migration, the legacy system can be used as oracle for the testing activity.
  - *Input*: migrated microservices in the new infrastructure, legacy system requirements, legacy system test cases.
  - *Output*: testing metrics, and bug reports.

4.4. Phase 4: Monitoring after the migration

Monitoring is responsible to assess the health of the migrated microservices in the new infrastructure.

- **Step 8: Monitoring the migrated microservices**. The goal here is to keep control of the behavior of the migrated microservices in the new infrastructure. The monitoring should focus on availability of the services, bottlenecks, performance, use of infrastructure resources, and the like. Since a desired scalability is one of the main driving forces for migrated to a microservice architecture, here the engineers should analyze if this goal has been achieved. Loris Degioanni[8] describes five principles of monitoring microservices: (i) monitor containers and what runs inside them, (ii) use orchestration systems, (ii) prepare for elastic, multilocation services, (iv) monitor APIs, and (v) map monitoring to the organizational structure.
  - *Input*: migrated microservices in the new infrastructure.
  - *Output*: monitoring metrics.

5. Limitation of our study

Three limitations need be highlighted about our study, as follows:

- *Number of primary sources:* the process was based on some six primary studies. There are others that could be considered. In this case, we can only claim the process as a preliminary version based on relevant primary studies.
- *Process validation:* the proposed process was not evaluated in practice, then we cannot affirm that the process strictly adhere to practical needs and industrial scenarios. However, we alleviate this limitation by comparing the proposed process with information available in survey with practitioners, namely considering the studies [Taibi et al. 2017, Carvalho et al. 2019b, Carvalho et al. 2019a]. In addition, we believe the process was based on relevant literature for the field.
- *Expertise of the authors:* despite the professor that supervised this study being a researcher on the topic of this study, a threat to validity is the expertise of the students on migrating legacy systems to microservice architecture. To mitigate such threat, the professor provided a background training and assisted all steps of the study. In addition, the professor participated in all discussions regarding the primary sources.

[8]Available at `https://thenewstack.io/five-principles-monitoring-microservices/` accessed on March 11th, 2020.

6. Final Remarks

The microservice architectural style has emerged in industry[9] and only in the recent year has been focus of research in academia. Therefore, there is still scarce literature on the process of migrating monolith legacy systems to microservice architectures.

Most of existing studies on migrating legacy systems to microservices have focus specific scenarios or tasks of the migration process. Based on the lack of a broader view of the entire process, this paper describes a preliminary process composed of four phases that together have eight steps. As part of our preliminary process, we also described the input and output for each step. This proposal claims to be a starting point to consolidate a process of migrating monolithic systems into microservices.

Based on the limitations of our study, we plan future work based on mainly two directions. Firstly, we have been conducting an extend study including new primary sources to provide a more in-depth discussion of each step of the migration process. We intent to provide a process based on more primary studies to ratify or rectify the identified phases and steps of our preliminary process and increase new elements such as stakeholders, artifacts, guidelines and templates. Secondly, to move into an appropriate process that will be accepted and adopted by the practitioners, we intend to evaluate/validate the proposed process in real world case studies.

References

Ahmadvand, M. and Ibrahim, A. (2016). Requirements reconciliation for scalable and secure microservice (de)composition. In *IEEE 24th International Requirements Engineering Conference Workshops (REW)*, pages 68–73.

Ali, N., Ramos, I., and Solís, C. (2010). Ambient-prisma: Ambients in mobile aspect-oriented software architecture. *Journal of Systems and Software*, 83(6):937 – 958. Software Architecture and Mobility.

Balalaie, A., Heydarnoori, A., Jamshidi, P., Tamburri, D. A., and Lynn, T. (2018). Microservices migration patterns. *Software Practice and Experience*, 48(11):2019–2042.

Carvalho, L., Garcia, A., Assunção, W. K. G., Bonifácio, R., Tizzei, L. P., and Colanzi, T. E. (2019a). Extraction of configurable and reusable microservices from legacy systems: An exploratory study. In *23rd International Systems and Software Product Line Conference - Volume A*, pages 26–31, New York, NY, USA. ACM.

Carvalho, L., Garcia, A., Assunção, W. K. G., de Mello, R., and de Lima, M. J. (2019b). Analysis of the criteria adopted in industry to extract microservices. In *7th International Workshop on Conducting Empirical Studies in Industry and 6th International Workshop on Software Engineering Research and Industrial Practice*, pages 22–29. IEEE.

Casey, A., Beeler, R., Fry, C., Mauvais, J., Pernice, E., Shor, M., Spears, J., Speck, D., and West, S. (2017). Strategies for migrating to a new experiment setup tool at the national ignition facility. *JACoW Publishing*, pages 1–4.

Chen, R., Li, S., and Li, Z. (2018). From Monolith to Microservices: A Dataflow-Driven Approach. In *Asia-Pacific Software Engineering Conference (APSEC)*, pages 466–475.

[9] `https://martinfowler.com/articles/microservices.html`

Di Francesco, P., Lago, P., and Malavolta, I. (2018). Migrating towards microservice architectures: an industrial survey. In *IEEE international conference on software architecture (ICSA)*, pages 29–2909. IEEE.

Di Francesco, P., Lago, P., and Malavolta, I. (2019). Architecting with microservices: A systematic mapping study. *Journal of Systems and Software*, 150:77–97.

Francesco, P. D., Malavolta, I., and Lago, P. (2017). Research on architecting microservices: Trends, focus, and potential for industrial adoption. In *IEEE International Conference on Software Architecture (ICSA)*, pages 21–30.

Fritzsch, J., Bogner, J., Wagner, S., and Zimmermann, A. (2019). Microservices migration in industry: Intentions, strategies, and challenges. In *IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 481–490. IEEE.

Fritzsch, J., Bogner, J., Zimmermann, A., and Wagner, S. (2018). From monolith to microservices: a classification of refactoring approaches. In *International Workshop on Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment*, pages 128–141. Springer.

Hassan, S., Ali, N., and Bahsoon, R. (2017). Microservice Ambients: An Architectural Meta-Modelling Approach for Microservice Granularity. *IEEE International Conference on Software Architecture (ICSA)*, pages 1–10.

Knoche, H. and Hasselbring, W. (2018). Using microservices for legacy software modernization. *IEEE Software*, 35(3):44–49.

Lewis, G., Plakosh, D., and Seacord, R. (2003). *Modernizing Legacy Systems: Software Technologies, Engineering Processes, and Business Practices*. Addison-Wesley Professional.

Mayer, B. and Weinreich, R. (2018). An approach to extract the architecture of microservice-based software systems. In *IEEE Symposium on Service-Oriented System Engineering (SOSE)*, pages 21–30.

Newman, S. (2015). *Building Microservices*. O'Reilly Media, 1st edition.

Ponce, F., Márquez, G., and Astudillo, H. (2019). Migrating from monolithic architecture to microservices: A rapid review. In *38th International Conference of the Chilean Computer Science Society (SCCC)*, pages 1–7. IEEE.

Rahman, M. T., Querel, L.-P., Rigby, P. C., and Adams, B. (2016). Feature toggles: Practitioner practices and a case study. In *13th International Conference on Mining Software Repositories (MSR)*, pages 201–211, New York, NY, USA. ACM.

Taibi, D., Lenarduzzi, V., and Pahl, C. (2017). Processes, motivations, and issues for migrating to microservices architectures: An empirical investigation. *IEEE Cloud Computing*, 4(5):22–32.

Algoritmo Baseado na Meta-heurística Particle Swarm Optimization para Configuração de Pools de Threads em Motores de Execução de Plataformas de Integração

Maira S. Brigo, Rafael Z. Frantz, Daniela L. Freire

Universidade Regional do Noroeste do Estado do Rio Grande do Sul
Departamento de Ciências Exatas e Engenharias
Rua Lulu Ilgenfritz, 480 – 98700-000 – Ijuí/RS – Brasil

mairа.brigo@sou.unijui.edu.br, {rzfrantz,dsellaro}@unijui.edu.br

Abstract. *This paper reports on the experience of using an optimization algorithm to find the optimal configuration of thread pools to run integration processes that receive a large volume of data. Thread pools are computational resources in the runtime system of integration platforms. Execution models based on a single pool of threads have performance problems when used in contexts in which there is a large volume of data. An execution model based on multiple local pools of threads can avoid performance problems, as long as these pools are configured with the ideal number of threads. However, finding that ideal number of threads is not a trivial task, as there is no automatic way to perform this calculation. In this paper we explore the use of the Particle Swarm Optimisation metaheuristic to find the ideal number of threads for each local pool.*

Resumo. *Este artigo relata a experiência de usar um algoritmo de otimização para encontrar a configuração ideal de pools de threads para executar processos de integração que recebem um grande volume de dados. Threads são recursos computacionais no sistema de tempo de execução de plataformas de integração. Os modelos de execução baseados em um único pool de threads apresentam problemas de desempenho quando usados em contextos nos quais há um grande volume de dados. Um modelo de execução baseado em vários pools locais de threads pode evitar problemas de desempenho, desde que esses pools sejam configurados com o número ideal de threads. No entanto, encontrar esse número ideal de threads não é uma tarefa trivial, pois não há uma maneira automática de realizar esse cálculo. Neste artigo, exploramos o uso da meta-heurística Particle Swarm Optimization para encontrar o número ideal de threads para cada pool local.*

1. Introdução

Os processos de negócio das empresas estão em constante transformação, requerendo a atualização e/ou inclusão de novas aplicações para dar suporte à execução dos processos. O conjunto destas aplicações formam o ecossistema de *software* da empresa [Manikas 2016]. Geralmente, as aplicações foram desenvolvidas em distintas linguagens, são executadas em diferentes sistemas operacionais e possuem modelos de dados que geralmente, não é o mesmo, tornando o ecossistema de *software* heterogêneo. Esta heterogeneidade faz a troca de dados e o compartilhamento de funcionalidades das

aplicações ser um desafio nos setores de Tecnologia de Informação (TI) das empresas. A área de Integração de Aplicações Empresariais *(Enterprise Application Integration - EAI)* proporciona metodologias, técnicas e ferramentas para desenvolver processos de integração [Romero and Vernadat 2016]. Definimos processo de integração como um programa computacional que integra aplicações envolvidas em um processo de negócio.

Plataformas de integração são ferramentas que oferecem recursos para modelar, implementar e executar processos de integração [Freire et al. 2019a]. A modelagem costuma ser realizada utilizando uma linguagem de domínio específico (DSL) e o resultado é um modelo conceitual que descreve o *workflow* com as tarefas e as aplicações integradas, isso permite as aplicações compartilharem dados e funcionalidades, com baixo nível de acoplamento. A implementação é feita com um conjunto de APIs que permitem transformar o modelo conceitual em código executável. A execução do código é responsabilidade do motor de execução, onde as *threads* são os recursos computacionais que executam as tarefas dos processos de integração e estão organizadas de duas diferentes formas: *pool* global ou *pool* local. A estratégia global consiste em *threads* armazenadas e disponibilizadas em um único *pool* associado àquele processo de integração, enquanto a estratégia local consiste em utilizar diversos *pools* de *threads* associados diretamente às tarefas.

Na literatura há dois modelos que podem ser seguidos para implementação dos motores de execução; *process-based* e *task-based* [Blythe et al. 2005, Alkhanak et al. 2016, Boehm et al. 2011, Frantz et al. 2012]. No modelo *process-based*, uma mesma *thread* é responsável pela execução de todas as tarefas sobre uma mesma mensagem que percorre o processo de integração, e no modelo *task-based*, distintas *threads* podem ser usadas para executar tarefas que processam uma mesma mensagem ao longo do processo. Este artigo centra-se em motores de execução que adotam o modelo *task-based*. Neste modelo, quando o motor de execução possui um *pool* global e a taxa de chegada de mensagens é muito alta, ocorre priorização da execução das tarefas do início do processo de integração, prejudicando a execução das demais tarefas [Freire et al. 2019b]. Esta priorização degrada a execução dos processos de integração, pois o tempo de espera das tarefas finais por *threads* tende a ficar elevado. Quando o motor de execução possui *pools* locais, esta degradação é minimizada, porque cada tarefa possui um *pool* de *threads* dedicado a execução de suas instâncias e isso faz com que as mensagens sejam executadas com celeridade, melhorando seu tempo de processamento, desde que seja uma configuração ótima ou próxima de ótima.

Uma medida usual de desempenho da execução de processos de integração é o *makespan*, que é o tempo médio de processamento de uma mensagem desde sua entrada até sua saída do processo [Chirkin et al. 2017]. Um *makespan* baixo é indicativo de mais mensagens processadas por unidade de tempo. Enquanto um *makespan* elevado é indicativo de menos mensagens processadas por unidade de tempo. Geralmente os engenheiros de *software* aumentam o número de *threads* para aumentar o desempenho do motor de execução, especialmente em contextos de grande volume de dados. Porém, um grande número de *threads* aumenta a concorrência entre elas pelos recursos computacionais que compartilham, levando a degradação de desempenho na execução [Pusukuri et al. 2011]. Portanto, o desafio encontrado é dimensionar a quantidade ideal de *threads* em cada *pool* local, independente do número de *threads* disponíveis, para que essa configuração de *pools* locais possa melhorar o desempenho dos motores de execução. Este artigo explora

uma alternativa para obter a configuração ideal para *pools* de *threads* locais por meio de um algoritmo de otimização que minimiza o *makespan* do processamento de mensagens, melhorando o desempenho dos motores de execução. Foi realizado um experimento com o algoritmo baseado na meta-heurística *Particle Swarm Optimization* (PSO) em um processo de integração, onde se variam o número de *threads* distribuídos em *pools* de *threads* locais. As principais contribuições com este artigo são de que é possível encontrar uma configuração ideal para os *pools* locais de *threads*, é possível encontrá-la por meio do algoritmo e obtém um *makespan* otimizado dado a um número total de *threads* fixo. O experimento pode ser ampliado para outras taxas de entrada de mensagens, processos de integração, números de *threads* e números de configuração testados. O restante deste artigo está organizado da seguinte forma: a Seção 2 apresenta um embasamento teórico. A Seção 3 discute trabalhos relacionados. A Seção 4 formula o problema e apresenta a proposta. A Seção 5 apresenta o experimento realizado e analisa os resultados alcançados. A Seção 6 apresenta as conclusões do trabalho.

2. Fundamentação Teórica

Um processo de integração consiste em um *workflow* de estrutura *pipes-and-filters* [Alexander 1977], na qual os *pipes* são os canais e os *filters* são tarefas que processam os dados que são encapsulados no formato de mensagens. [Hohpe and Woolf 2003] documentaram um conjunto de padrões de integração que inspiram a concepção dessas tarefas proporcionadas pelas linguagens de domínio específico das plataformas de integração. Em um processo de integração, as tarefas estão dispostas em uma determinada ordem, de maneira que uma mensagem só pode ser processada por uma tarefa depois de já ter sido processada por todas as tarefas que a antecedem. Uma mensagem é completamente processada quando executada por cada uma das tarefas que compõem o *workflow* que ela percorre desde que entra no processo até quando ela sai.

O processo de integração é realizado pelo motor de execução. Os principais elementos de um motor de execução são: planificador, filas de instâncias, e *pools* de *threads*. O planificador gerencia a fila de instâncias, e as *threads*. Os agendamentos de cada uma das instâncias são feitos em uma fila de instâncias relacionada a sua tarefa e consequentemente ao seu *pool*, onde permanecem enquanto aguardam *threads* disponíveis para serem executadas. A fila de instâncias mantém o agendamento da execução das tarefas prontas para serem executadas, sendo que, uma tarefa é considerada apta, quando há mensagens aguardando em todas as suas entradas. Uma tarefa possui obrigatoriamente entradas e saídas. Para a tarefa ser executada, é necessário que haja *thread* disponível para sua execução. As *threads* estão recorrentemente verificando a sua respectiva fila de instâncias, na qual as primeiras instâncias anotadas na fila serão as primeiras a serem executadas. As *threads* do *pool*, quando estão disponíveis, buscam na sua fila as instâncias para executar. Chamamos de uma configuração de um *pool* local um vetor, no qual cada elemento corresponde ao número de *threads* de um *pool* local.

O *task-based* é um modelo que a literatura define como tendo uma granularidade mais baixa, e portanto, é um modelo que busca incrementar o processo paralelo de tarefas e o compartilhamento de *threads* entre tarefas de processos. A literatura também acrescenta que esse modelo costuma ser mais eficiente que o *process-based* [Blythe et al. 2005, Frantz et al. 2012]. A meta-heurística PSO foi utilizada já que segundo seu criador, cada partícula do enxame controla melhor sua posição no hiperespaço,

sendo uma meta-heurística eficaz em diversos tipos de problemas, além de ter um conceito simples e ser de fácil implementação [Eberhart and Kennedy 1995].

3. Trabalhos Relacionados

Na revisão de literatura, identificamos um conjunto de trabalhos que analisam e buscam reduzir o *makespan* utilizando técnicas de otimização. Nos trabalhos analisados, as propostas são algoritmos ou aperfeiçoamento de algoritmos tendo como base o PSO, ou outras meta-heurísticas. Em concordância, este artigo busca a minimização do *makespan*, mas através de um algoritmo baseado apenas no PSO, não são feitas comparações com outras meta-heurísticas. [Irman et al. 2019] propuseram um algoritmo para reduzir o *makespan* envolvido no tempo de produção de uma indústria na Indonésia. Os autores compararam a eficiência de seu algoritmo na solução do problema também com o uso dos métodos heurísticos *Camppbell Dudek Smith* (CDS) e PSO. Neste trabalho eles demonstraram que o *makespan* utilizando o CDS foi o mais alto, o com o PSO foi o de desempenho médio, e com o algoritmo proposto no artigo se obteve o menor *makespan*. O trabalho de [Zarrouk and Jemai 2018] teve o intuito de minimizar a produção e carga de trabalho total. Para isso, foram avaliados e comparados o desempenho de duas variantes de PSO com diferentes representações de partículas, uma delas foi o PSO com esquema de codificação *Job-Manchine* (PSO-JMS) e a outra o PSO com o esquema de codificação *Only-Manchine* (PSO-OMS). Como resultado, o PSO-OMS ofereceu o melhor desempenho.

[Bozorgnezhad et al. 2019] propuseram minimizar o *makespan* para solucionar dois problemas simultâneos: a) encontrar a melhor atribuição do trabalhador; e, b) resolver o problema de agendamento correspondente. Para isso, foram utilizadas duas abordagens de aproximação baseadas no PSO: o PSO e o SPSO. Os resultados mostraram que os dois algoritmos minimizaram eficientemente, mas o SPSO teve soluções melhores, principalmente para problemas maiores. [Teekeng et al. 2016] tinham como função objetivo minimizar o *makespan* para resolver problemas de agendamento. Para isso, propuseram um novo algoritmo, denominado EPSO, que foi baseado no PSO. No EPSO foram incluídos dois conjuntos de recursos, com o intuito de expandir o espaço de solução e evitar a convergência prematura. [Zhang and Wu 2014] investigaram um problema de agendamento de permutas de produção, com o objetivo de minimizar o tempo total de fluxo. Para isso, propõem uma meta-heurística híbrida baseada no PSO. Os resultados mostram que a proposta pode competir com outras meta-heurísticas poderosas da literatura.

[Khosiawan et al. 2018] tinham objetivo de minimizar o *makespan* e o consumo total de bateria em um cronograma de sistema de agendamento. Fizeram a comparação da otimização diferencial do enxame de partículas fundidas na evolução com a evolução diferencial e a otimização do enxame de partículas. A primeira mostrou resultados eficazes. Também para um problema de agendamento de tarefas, [Sarathambekai and Umamaheswari 2017] apresentaram este em sistemas distribuídos, e utilizaram a otimização de enxame de partículas discretas (DPSO) com várias topologias de vizinhança. Esta é uma meta-heurística recente para algoritmo populacional.

Estes trabalhos relacionados buscam otimizar o *makespan* em processos de integração, mas nenhum deles faz esta otimização utilizando o modelo de execução *task-based* e seguindo a estratégia de *pools* locais, sendo este o diferencial do nosso trabalho.

4. Formulação do problema

Um processo de integração é composto por um fluxo de trabalho que pode ser representado por um Grafo Acíclico Direcionado, representado por $DAG = (Z, H)$, onde Z é o conjunto de *n* tarefas e H são as arestas que representam a relação de dependência entre as tarefas, ou seja, as tarefas dependem umas das outras e se relacionam entre si. Assumimos que um conjunto de recursos utilizados para o processamento é representado por *r* e o seu armazenamento é indicado por *Dr*, sendo que todos estes recursos se localizam em *P* locais diferentes. Em um DAG existem nós que são conectados por arestas. Em nosso contexto, os nós são tarefas, as arestas são canais de comunicação em que são organizadas mensagens, e as instâncias são ocorrências concretas das tarefas que são executadas por *threads*. As *threads* estão disponíveis em *pools* locais, sendo que cada tarefa tem um *pool* de *threads*. Um caminho é um conjunto sequencial de tarefas que estão organizadas em um fluxo de trabalho, e o caminho mais longo no processo de integração, se chama *caminho crítico*. Em cada nó há um tempo de processamento associado, e dos nós pode haver um conjunto de instâncias que permitem uma execução paralela do nó.

Se busca encontrar o número correto de instâncias para cada nó, para que o tempo de processamento seja o menor possível. Para resolver o problema da degradação, consideramos que existe uma restrição que limita o número total de *threads* a serem distribuídas nos *pools*, e que é necessário ter pelo menos uma *thread* para que a tarefa seja executada. O tempo de processamento de uma tarefa é a soma do tempo de execução e do tempo de espera na fila de instâncias. O *makespan* é obtido realizando a soma de tempo de execução de cada tarefa e a soma do tempo de transferência de dados de uma tarefa para sua sucessora [Bilgaiyan et al. 2014]. Assim, o *makespan* de todo o fluxo do processo T_t (r_i) é o somatório do tempo incorrido da execução de cada tarefa T_i para o r_i de recursos e o somatório dos tempos de transmissão (T_m) entre um conjunto de tarefas que sejam dependentes entre si. O tempo total de processamento (TT_p) é calculado pela Equação 1:

$$TT_p = \sum T_i + \sum T_m \tag{1}$$

Queremos que o tempo de processamento que uma mensagem leva para atravessar todo o grafo seja o mínimo possível, então, a função objetivo é: *Minimizar o makespan do processamento de mensagens pelo caminho crítico de um processo de integração, encontrando a melhor configuração dos pools de threads locais, utilizando uma quantidade pré-definida de threads para ser distribuída entre os pools locais*. Para isso, podemos utilizar a função objetivo em uma meta-heurística, que é uma técnica que tem o intuito de buscar a otimização. Se baseia em uma população, em que a função objetivo é testada para os indivíduos dessa população inicial, e são feitas operações para melhorar a convergência do resultado, buscando minimizar a função objetivo. Como proposta, utilizamos a técnica meta-heurística PSO, que tem motivação no comportamento social [Shi and Eberhart 1999]. Inicialmente é gerada uma população com possíveis configurações de *threads* para os *pools* locais, que é a população inicial. Então, é calculado o tempo total médio de processamento das mensagens no fluxo de trabalho, o *makespan*, para cada configuração. Sempre que o *makespan* atual for menor que o anterior, é atualizado o *makespan* mínimo acompanhado de sua configuração. Ao fim, é encontrada a melhor configuração, que é a que possui o *makespan* mínimo de todo o processo.

5. Experimento

Esta seção descreve o experimento realizado com uma adaptação do PSO, para encontrar a configuração ideal dos *pools* de *threads* local para *makespan* mínimo em um processo de integração. O protocolo aplicado para conduzir o estudo experimental tem bases nos trabalhos de [Jedlitschka and Pfahl 2005, Perry et al. 2000] e [Wohlin et al. 2012].

5.1. Questão de pesquisa e hipótese

Foi selecionada uma questão de pesquisa para ser respondida a partir deste estudo experimental: *É possível encontrar uma configuração ideal de pools de threads para que motores de execução de plataformas de integração possam executar um processo de integração no menor makespan possível no contexto de grandes volumes de dados?*

Para esta questão de pesquisa, foi fornecida uma hipótese, que deve ser confirmada ou refutada pelo experimento: *Podemos encontrar a configuração ideal dos pools de threads locais por meio de um algoritmo baseado na técnica de otimização PSO, cujo objetivo é minimizar o makespan da execução no processo de integração*.

5.2. Ambiente do experimento

O experimento foi realizado em uma máquina equipada com 32 processadores *Intel Xeon CPU* E5-4610 1,80 *GHz*, 32 *GB* de *RAM* e sistema operacional *Microsoft Windows 10 Education* 64 *bits*. O *software Matlab* [Leonard and Levine 1995], versão R2018, foi usado executar os algoritmos. O código fonte do algoritmo de otimização está disponível para *download*[1].

5.3. Variáveis

Nesta seção, descrevem-se as variáveis dependentes e independentes que consideramos em nosso experimento. As variáveis independentes são:

Número de soluções: a população de configurações iniciais dos *pools* de *threads*. O valor desta variável foi de 20 soluções.

Número de *threads*: número total de *threads* que são distribuídos nos *pools* locais. O valor desta variável foi de 100 *threads*.

Número de mensagens: carga de trabalho do processo de integração, em que o valor testado foi 1.000.000 mensagens.

A variável dependente é:

***Makespan*:** o tempo médio total de processamento que uma mensagem leva para ser processada por todas as tarefas que fazem parte do *caminho crítico* do processo de integração.

5.4. Execução e coleta de dados

Nesta seção, descreve-se o estudo de caso e a coleta de dados. O estudo de caso adotado é o processo de integração para processamento de pedidos proposto por [Hohpe and Woolf 2003], conforme a Figura 1. O processo de integração tem cinco aplicações: *Ordering System*, *Widget Inventory*, *Gadget Inventory*, *Invalid Items Log* e *Inventory System*. O *Ordering System* representa o aplicativo de origem entregando os

[1] `www.gca.unijui.edu.br/publication/data/eres-pso-sources.zip`

dados dos novos pedidos ao processo de integração. Os dados de um pedido compõem uma mensagem que flui pelo processo de integração. Cada mensagem com um novo pedido é dividida em distintas mensagens, cada uma contendo apenas um item. O destino da mensagem para a aplicação *Widget Inventory* ou para a aplicação *Gadget Inventory*, está contido no conteúdo da mesma. Mensagens com itens não pertencentes a nenhum desses inventários são roteadas para *Invalid Items Log*. O aplicativo *Inventory System* representa a aplicação de destino final, que responde registrando a disponibilidade dos itens.

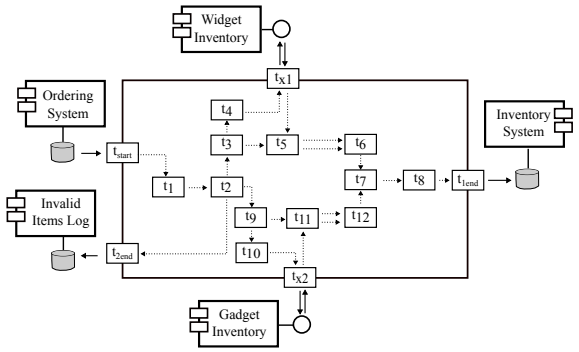


Figura 1. Modelo conceitual do processo de integração.

Este processo de integração possui três caminhos diferentes para mensagens que são representados no DAG da Figura 2.O *caminho crítico* é formado pelas tarefas t_{start}, t_1, t_2, t_3, t_4, t_{x1}, t_5, t_6, t_7, t_8, t_{1end}. As tarefas do *caminho crítico* consomem os seguintes tempos de processamento tomados do trabalho de [Haugg et al. 2019], em milissegundos: 2.005ms, 2.005ms, 3.005ms, 3.005ms, 2.005ms, 2.005ms, 4.553ms, 2.005ms, 4.553ms, 2.005ms, 2.005ms. A simulação da execução deste processo foi realizada na máquina virtual descrita anteriormente, com o *software Matlab*, utilizando o caminho crítico com um processamento de 1.000.000 de mensagens. Foram experimentadas de maneira aleatória pelo algoritmo, 20 configurações diferentes para os *pools* de *threads* locais, sendo um *pool* por tarefa, ou seja, 11 *pools*, com a restrição de usar no total 50 *threads*. A simulação resultou em 20 configurações, e cada uma delas gerou um *makespan*. Cada configuração e seu *makespan* foram coletados e armazenados para posterior análise.

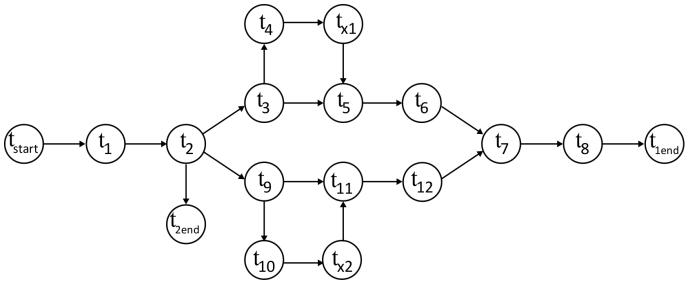


Figura 2. Grafo Acíclico Direcionado que representa o processo de integração.

5.5. Resultados

O menor *makespan* encontrado foi de 1002528,15ms com as configurações: [16,7,2,8,3,3,3,1,3,3,1] e [7,5,2,3,5,8,5,1,7,1,6]. A Figura 3 apresenta o gráfico dos *makespan* de todas as configurações de *threads* geradas pelo algoritmo. Neste estudo de

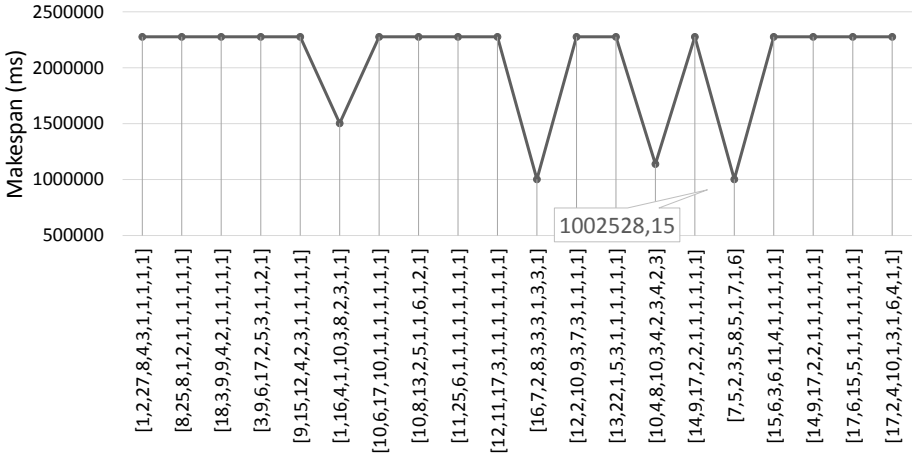


Figura 3. Makespan para diferentes configurações de pools de threads locais.

caso foi possível encontrar as configurações que obtiveram o menor tempo de processamento. Estas configurações contém o número ideal de *threads* nos *pools* locais do motor de execução, e podemos nos referir a elas como as configurações ideais, pois são as que encontraram o menor *makespan* no processo de integração.

5.6. Ameaças à validade

Instrumentação e fonte de ruído são possíveis ameaças. Para mitigá-las, antes da realização da simulação, definiu-se a questão de pesquisa, formulou-se a hipótese e definiram-se as variáveis independentes e dependentes. Depois, foram fornecidas informações sobre o ambiente de execução, ferramentas de suporte, execução e coleta de dados. Para minimizar a interferência no tempo de execução do algoritmo, toda a simulação foi realizada na mesma máquina, utilizando recursos mínimos e desconectada da *Internet* durante as execuções.

6. Conclusão

Neste artigo foi apresentada uma proposta para encontrar uma configuração ideal de *threads* para cada *pool* local, com o objetivo de reduzir o *makespan* em um processo de integração. Como o sistema de execução é o componente da plataforma responsável pela execução dos processos de integração, ele se torna o elemento mais importante quando o objetivo é o desempenho, pois ele está diretamente relacionado aos recursos computacionais. Propomos um algoritmo de otimização baseado na meta-heurística PSO, ele foi implementado e experimentado, encontramos múltiplas configurações dentre as quais uma configuração que minimizou o *makespan*, que denominamos de ideal no experimento do estudo de caso proposto. O experimento permitiu confirmar a hipótese de que era possível encontrar a configuração ideal dos *pools* de *threads* locais por meio de um algoritmo baseado em PSO, cuja função objetivo era minimizar o *makespan* do processo de integração. O algoritmo ainda, pode ser utilizado em diferentes contextos, com diferentes números de mensagens, outros processos de integração, números de *threads* e das configurações testadas. Como trabalho futuro, propomos de estender os resultados desta pesquisa, analisar o experimento estatisticamente e, utilizar o mesmo estudo de caso e protocolo de experimentação, utilizando outra meta-heurística. Também realizar uma comparação en-

tre os dois experimentos para posterior análise, para entender qual meta-heurística é mais adequada para otimizar o desempenho de processos de integração.

Agradecimentos

Este trabalho tem o apoio da Fundação de Amparo à Pesquisa do Estado do Rio Grande do Sul (FAPERGS) no contexto do Programa Pesquisador Gaúcho, com termo de outorga número 17/2551-0001206-2 e da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pela bolsa de pós-graduação. Gostaríamos de agradecer ao colega Fernando Parahyba por seus comentários em versões anteriores deste artigo.

Referências

Alexander, C. (1977). *A pattern language: towns, buildings, construction*. Oxford University Press.

Alkhanak, E. N., Lee, S. P., Rezaei, R., and Parizi, R. M. (2016). Cost optimization approaches for scientific workflow scheduling in cloud and grid computing: A review, classifications, and open issues. *Journal of Systems and Software*, 113:1–26.

Bilgaiyan, S., Sagnika, S., and Das, M. (2014). Workflow scheduling in cloud computing environment using cat swarm optimization. In *2014 IEEE International Advance Computing Conference (IACC)*, pages 680–685. IEEE.

Blythe, J., Jain, S., Deelman, E., Gil, Y., Vahi, K., Mandal, A., and Kennedy, K. (2005). Task scheduling strategies for workflow-based applications in grids. In *IEEE International Symposium on Cluster Computing and the Grid, 2005.*, volume 2, pages 759–767.

Boehm, M., Habich, D., Preissler, S., Lehner, W., and Wloka, U. (2011). Cost-based vectorization of instance-based integration processes. *Information Syst.*, 36(1):3–29.

Bozorgnezhad, F., Asadi-Gangraj, E., and Mahdi, M. (2019). A simultaneous worker assignment and scheduling problem for minimizing makespan in flexible flow shop via metaheuristic approaches.

Chirkin, A. M., Belloum, A. S., Kovalchuk, S. V., Makkes, M. X., Melnik, M. A., Visheratin, A. A., and Nasonov, D. A. (2017). Execution time estimation for workflow scheduling. *Future Generation Computer Systems*, 75:376–387.

Eberhart, R. and Kennedy, J. (1995). Particle swarm optimization. In *Proceedings of the IEEE intl. conference on neural networks*, volume 4, pages 1942–1948. Citeseer.

Frantz, R. Z., Corchuelo, R., and Molina-Jiménez, C. (2012). A proposal to detect errors in enterprise application integration solutions. *Journ. of Syst. and Soft.*, 85(3):480–497.

Freire, D. L., Frantz, R. Z., and Roos-Frantz, F. (2019a). Ranking enterprise application integration platforms from a performance perspective: An experience report. *Software: Practice and Experience*, 49(5):921–941.

Freire, D. L., Frantz, R. Z., Roos-Frantz, F., and Sawicki, S. (2019b). Survey on the runtime systems of enterprise application integration platforms focusing on performance. *Software: Practice and Experience*, 49(3):341–360.

Haugg, I. G., Frantz, R. Z., Roos-Frantz, F., Sawicki, S., and Zucolotto, B. (2019). Towards optimisation of the number of threads in the integration platform engines using simulation models based on queueing theory. *Rv. Br. de Cmp. Ap.*, 11(1):48–58.

Hohpe, G. and Woolf, B. (2003). *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. Addison-Wesley.

Irman, A., Febianti, E., and Khasanah, U. (2019). Minimizing makespan on flow shop scheduling using campbel dudek and smith, particle swarm optimization, and proposed heuristic algorithm. In *IOP Conference Series: Materials Science and Engineering*, volume 673, page 012099. IOP Publishing.

Jedlitschka, A. and Pfahl, D. (2005). Reporting guidelines for controlled experiments in software engineering. In *Intl. Symposium on Empirical Software Eng.*, pages 95–104.

Khosiawan, Y., Khalfay, A., and Nielsen, I. (2018). Scheduling unmanned aerial vehicle and automated guided vehicle operations in an indoor manufacturing environment using differential evolution-fused particle swarm optimization. *International Journal of Advanced Robotic Systems*, 15(1):1729881417754145.

Leonard, N. E. and Levine, W. S. (1995). *Using MATLAB to analyze and design Control Systems*. Benjamin-Cummings Publishing Company.

Manikas, K. (2016). Revisiting software ecosystems research: A longitudinal literature study. *Journal of Systems and Software*, 117:84–103.

Perry, D. E., Porter, A. A., and Votta, L. G. (2000). Empirical studies of software engineering: A roadmap. In *Proc. of the Conf. on The Future of Soft. Eng.*, pages 345–355.

Pusukuri, K. K., Gupta, R., and Bhuyan, L. N. (2011). Thread reinforcer: Dynamically determining number of threads via os level monitoring. In *IEEE Iinternational Simposium on Workload Characterization (IISWC)*, pages 116–125.

Romero, D. and Vernadat, F. (2016). Enterprise information systems state of the art: Past, present and future trends. *Computers in Industry*, 79:3–13.

Sarathambekai, S. and Umamaheswari, K. (2017). Task scheduling in distributed systems using heap intelligent discrete particle swarm optimization. *Computational Intelligence*, 33(4):737–770.

Shi, Y. and Eberhart, R. C. (1999). Empirical study of particle swarm optimization. In *Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406)*, volume 3, pages 1945–1950. IEEE.

Teekeng, W., Thammano, A., Unkaw, P., and Kiatwuthiamorn, J. (2016). A new algorithm for flexible job-shop scheduling problem based on particle swarm optimization. *Artificial Life and Robotics*, 21(1):18–23.

Wohlin, C., Runeson, P., Hst, M., Ohlsson, M. C., Regnell, B., and Wessln, A. (2012). *Experimentation in Software Engineering*. Springer.

Zarrouk, R. and Jemai, A. (2018). Performance evaluation of particles coding in particle swarm optimization with self-adaptive parameters for flexible job shop scheduling problem. In *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, pages 396–407. Springer.

Zhang, L. and Wu, J. (2014). A pso-based hybrid metaheuristic for permutation flowshop scheduling problems. *The Scientific World Journal*, 2014.

Políticas de Gerência de Configuração de Software para Grupos de Pesquisa

Felipe Fernandes da Silva[1],
Aline Maria Malachini Miotto Amaral[1], **Thelma Elita Colanzi**[1]

[1]Universidade Estadual de Maringá (UEM)
Maringá – PR – Brazil

felipe_ffs6@hotmail.com, ammmamaral@uem.br, thelma@din.uem.br

Abstract. *Considering an increasingly dynamic scenario that requires constant change, today's software development requires people and organizations to control the quality of artifacts generated throughout their process. Changes in software artifacts can occur for a variety of factors, whether for correcting functionality or adapting to new business demand. Besides, such artifacts are generally elaborated collaboratively what makes that in order to ensure their quality it is necessary the adoption of Software Configuration Management (SCM) practices. Computing research group environments resemble in many aspects software development environments, in which the main artifact produced is software. These software, elaborated over several researches and by many researchers, need to have their quality controlled. In this sense, the objective of this work is to propose a set of SCM policies for research group environments. It is expected that with the implementation of policies such as the proposals the quality of software developed in academic environments can be improved.*

Resumo. *Considerando um cenário cada vez mais dinâmico que requer mudanças constantes, o desenvolvimento de software atual exige que pessoas e organizações controlem a qualidade dos artefatos gerados ao longo de seu processo. Alterações nos artefatos de software podem ocorrer por vários fatores, seja para corrigir a funcionalidade ou adaptar-se à nova demanda de negócios. Além disso, esses artefatos geralmente são elaborados de forma colaborativa, o que faz com que, para garantir sua qualidade, seja necessária a adoção de práticas de Gerenciamento de Configuração de Software (GCS). Os ambientes dos grupos de pesquisa em computação se assemelham em muitos aspectos aos ambientes de desenvolvimento de software, nos quais o principal artefato produzido é o software. Esses softwares, elaborados em várias pesquisas e por muitos pesquisadores, precisam ter sua qualidade controlada. Nesse sentido, o objetivo deste trabalho é propor um conjunto de políticas de GCS para ambientes de grupos de pesquisa. Espera-se que, com a implementação de políticas como as propostas, a qualidade do software desenvolvido em ambientes acadêmicos possa ser melhorada.*

1. Introdução

Ao longo do ciclo de vida de um projeto de software uma grande quantidade de itens de informação é produzido. É possível criar documentos, código-fonte, manuais e estes podem ser alterados por correções, adaptações do funcionamento e novas implementações.

Com a finalidade de não perder o controle do projeto em consequência das mudanças, é necessário que estas mudanças sejam devidamente controladas e gerenciadas [Figueiredo et al. 2004]. Para lidar com estas complexidades um processo denominado Gerência de Configuração de Software (GCS) foi estabelecido [Crnkovic et al. 2003].

A GCS pode ser definida como o controle da evolução de sistemas complexos ou, de forma mais pragmática, como a disciplina que permite manter produtos de software em evolução sob controle, e, portanto, contribuindo para satisfazer restrições de qualidade e de cronograma [Estublier 2000].

Em um ambiente de desenvolvimento de projetos de pesquisa na área de Computação, a dificuldade de gerenciar artefatos e mudanças é grande. A falta de recursos financeiros, humanos e principalmente de tempo para as atividades de gerenciamento impactam fortemente o dia a dia dos desenvolvedores de sua equipe. Sempre com prazos curtos para suas entregas, pesquisadores de grupos de pesquisa muitas vezes negligenciam as atividades conferidas pela gerência de configuração. Cabe ressaltar que os artefatos produzidos por trabalhos de grupos de pesquisa, normalmente tem continuidade em diversas pesquisas e diferentes pesquisadores, o que faz com que a falta da adoção de práticas de GCS seja um risco para a continuidade de muitos projetos.

Dentro do contexto apresentado, o presente trabalho teve como objetivo a definição Políticas de Gerência de Configuração para ambientes de desenvolvimento de grupos de pesquisa. Além disso, também foi realizado um estudo de caso aplicando as políticas definidas em um ambiente real de grupos de pesquisa.

Este trabalho está organizado da seguinte forma: a Seção 2 apresenta os fundamentos que guiram o desnvolvimento deste trabalho. Na Seção 3 as políticas de GCS propostas são descritas e a Seção 4 demonstra a aplicação das proposta deste trabalho no contexto de um grupo de pesquisa. Finalmente na Seção 5 são apresentadas as conclusões e trabalhos futuros.

2. Contextualização

Esta seção descreve os principais conceitos e características inerentes à Gerência de Configuração (GCS) necessários para a definição das politicas de GCS apresentadas neste trabalho.

2.1. Gerência de Configuração

A gerência de configuração é a arte de identificar, organizar e controlar modificações em um software que está sendo construído por uma equipe de desenvolvimento, com o objetivo de maximizar a produtividade minimizando erros e coordenando o desenvolvimento de software para gerir a confusão produzida pelas muitas mudanças que ocorrem ao longo do desenvolvimento de um software [Leon 2015].

De acordo com [Scott and Nisse 2001], a implementação de um processo de gerência de configuração de sucesso requer planejamento, entendimento do contexto e estrutura organizacional e as relações entre os elementos organizacionais.

A GCS atua ao longo de todo o ciclo de desenvolvimento do software, desde a solicitação de uma alteração até a auditoria dos itens entregues ao final do desenvolvimento [Pressman 2011]. Nesse sentido, pode-se identificar cinco principais ati-

vidades: Identificação dos itens de configuração; Controle de versão; Controle de alteração/mudança; Auditoria; e Relatos das alterações.

Estes itens podem ser documentos, casos de teste, código-fonte, seção de especificações de requisitos criados durante a análise, ferramentas de desenvolvimento, bibliotecas de código de terceiros, documentação da instalação, manutenção, operação ou uma parte de modelo de projeto.

Na GCS, o controle de versões é um sistema que registra as mudanças feitas em um arquivo ou um conjunto de arquivos ao longo do tempo, de forma que se possa recuperar versões específicas, podendo ser utilizado para versionar praticamente qualquer tipo de arquivo em um computador. Para oferecer suporte automatizado a esta atividade foram desenvovidos softwares de controle de versão, que são também conhecidos como CVS (*Concurrent Version System*) [Chacon and Straub 2014].

De acordo com [Sink 2011], existem operações e conceitos que foram implementados em todos os sistemas de controle de versão, porém cada sistema implementa à sua maneira dentro de sua estrutura, são eles:

- *Commit*: um *commit* corresponde a formalização de uma ação no banco de dados do sistema de controle de versão.
- *Pull/Update*: é responsável por realizar a operação de atualização de conteúdo local, aplicando as alterações no repositório e realizando *merging* com as alterações realizadas, quando existem alterações.
- *Checkout/Clone*: trata-se de *checkout* um processo para obter uma cópia de um projeto a partir do repositório onde ele se encontra, a cópia criada é chamada de cópia de trabalho.
- *Push*:A ação de *push* corresponde ao repasse de alterações do repositório local, onde as ações de *commits* foram realizados para um repositório principal.
- *Merge*: Trata-se da junção de trechos que estão divergentes, ou seja, é a fusão de dois *branches* de desenvolvimento.
- *Add*: corresponde a operação de adição que é utilizada quando se tem um arquivo ou diretório que ainda não está sob o controle de versão e deseja adicionar o repositório.
- *Diff*: A operação *diff* corresponde as alterações realizadas na cópia de trabalho.

Além do controle de versões, a GCS também é caracterizada pelo sistema de controle de mudanças. Este sistema é encarregado de executar a função de controle da configuração de forma sistemática, armazenando todas as informações geradas durante o andamento das solicitações.

Segundo Sommerville [Sommerville 2011], o processo de gerenciamento de mudanças inicia quando um cliente submete uma solicitação de mudança (CR *change request*) que descreve a mudança requerida. Este cliente é um *Stakeholder*, que não faz parte da equipe de desenvolvimento, podendo até ser um cliente interno, como a equipe responsável pelos testes do software ou a equipe de marketing, por exemplo.

Por fim, a Auditoria de gerência de configuração visa complementar possíveis revisões técnicas que foram feitas, de modo a garantir a consistência das alterações e do versionamento.

2.2. Git

O Git [Spinellis 2012] é um software livre, liberado sob a licença GNU GPL [GPL 2017], para controle de versão distribuído. Criado por Linus Torvalds, o Git nasceu em 2005 e surgiu de uma necessidade específica de armazenamento do núcleo do Linux (Sistema operacional), que é um projeto de código aberto e razoavelmente grande.

Segundo [Loeliger and McCullough 2012], os objetivos iniciais da implementação do Git eram: desempenho rápido e eficiente, manter integridade e confiança, imutabilidade, transações atômicas, suporte robusto a desenvolvimento não linear (milhares de braços paralelos), facilitar o desenvolvimento distribuído, *Design* interno organizado, repositórios completos e reforçar a responsabilidade.

Por conta dessas características, o Git ganhou um grande espaço no mercado, tendo, segundo [Duck 2017], 47% de usuários, superando, assim, o SubVersion que conta com 40% de usuários. Porém, muito do sucesso do Git se deve ao GitHub [Loeliger and McCullough 2012], pois este permitiu que grandes corporações colocassem seus projetos em repositórios públicos para que desenvolvedores de todas as partes do mundo pudessem contribuir para os projetos, criando uma comunidade de mútua colaboração e desenvolvimento.

GitHub: conforme [Loeliger and McCullough 2012], já existem diversas plataformas para o Git, mas o GitHub se destaca entre elas. GitHub é uma ferramenta de hospedagem de código-fonte com controle de versão utilizando o Git. O desenvolvimento da plataforma começou em 2007. Os projetos no GitHub podem ser acessados e manipulados usando a interface de linha de comando Git padrão e todos os comandos Git padrão funcionam com ele.

3. Políticas de GCS para ambiente de grupos de pesquisa

Esta seção descreve as políticas de GCS criadas para ambientes de grupos de pesquisa, por meio da utilização das ferramentas Git e GitHub.

3.1. Descrição do cenário

As políticas propostaso terão como base a utilização de duas ferramentas, o Git e o GitHub.

Considerando um ambiente em que as ferramentas já estejam configuradas e instaladas, é necessário primeiramente realizar a definição e criação do repositório para o projeto de acordo com a estratégia definida pela organização.

Após a criação do repositório, o mesmo deverá ser configurado de acordo com a estratégia de desenvolvimento que se deseja abordar. No processo aqui definido, sempre que um *merge* for solicitado, tal solicitação deverá ser feita por meio de um *pull request*. A funcionalidade *pull request* é disponibilizada pelo GitHub e nada mais é do que uma solicitação de *merge* realizada pelo desenvolvedor quando finalizou o desenvolvimento de sua alteração e deseja integrar estas alterações ao *branch* principal, para que este esteja disponível na versão de liberação para os clientes.

3.2. Papéis do Processo

Segundo Dart [Dart 1990], um cenário operacional para aplicação da gerência de configuração em uma empresa envolve os papéis de gerente de projeto, gerente de

configuração, engenheiros de software e usuários. No contexto de um projeto de pesquisa, esses papéis não existem. Sendo assim, outros devem ser criados para que possa haver a distribuição das responsabilidades do desenvolvimento.

Este processo conterá apenas três papéis para que seja realizado em sua completude, quais sejam:

Professor orientador: responsável pela criação das solicitações de mudanças, criação do repositório para cada nova versão do software que estará disponível; e definição de cronograma de entregas.
Pesquisador responsável: este papel pode ser realizado por um aluno de mestrado, doutorado ou por um professor; responsável por efetuar revisões técnicas nas alterações realizadas pelo desenvolvedor e caso a alteração seja aprovada, realizar o *merge* da mesma, resolvendo possíveis conflitos.
Pesquisador desenvolvedor: reposável por realizar as alterações e correções conforme solicitado pelo professor orientador e pesquisador responsável após a revisão do artefato modificado.

A Figura 1 demonstra estes papéis, suas relações e as tarefas envolvidas no processo de gerência de configuração definido para grupos de pesquisa conforme descritos anteriormente.

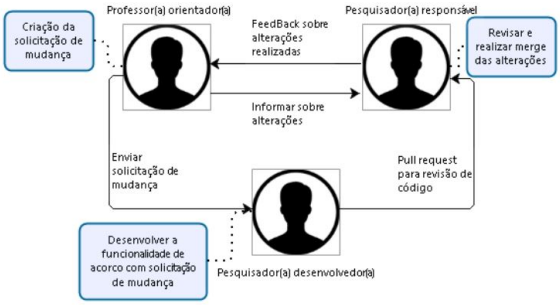


Figura 1. Papéis e tarefas envolvidos no processo de Gerência de Configuração

3.3. Identificação dos Itens de Configuração

A primeira tarefa no processo de gerência de configuração consistem em identificar os itens que devem ser gerenciados. Inicialmente, é necessário identificar quais itens são produzidos pelo processo de desenvolvimento do grupo de pesquisa. Código-fonte, requisitos e testes são exemplos.

Alguns aspectos devem ser considerados para a determinação dos itens: os itens são utilizados por um ou mais grupos de trabalho?; os itens sofram alterações ao longo do tempo? Os itens são críticos ou de importância elevada para o projeto? Os itens são dependentes uns dos outros?

Os itens de configuração identificados são os artefatos que serão passíveis de gerência de configuração, logo, são os arquivos que deverão ser armazenados no repositório remoto. Cada item armazenado no sistema de controle de versão passa a possuir um número de revisão e a cada alteração persistida no sistema de controle de versão este número é alterado. Além da revisão, outros atributos devem ser armazenados, tais como: data e hora, o usuário que realizou a ação, o tipo de ação realizada (inclusão, alteração ou

remoção de um arquivo no repositório), além de uma mensagem que pode ser adicionada pelo usuário que realizou a ação.

3.4. Gerenciamento de Solicitação de Mudanças

O controle de modificação consiste em gerenciar o ciclo de vida de uma mudança desde a sua criação até a sua conclusão. O GitHub possui um mecanismo de registro de solicitação de mudanças denominado *Issues*, e, por meio deste mecanismo, os requisitos e problemas devem ser cadastrados e acompanhados através dos estados de seu ciclo de vida.

Uma solicitação de mudança deve conter: título; descrição da solicitação de mudança; tipo - classificada de acordo com um padrão definido pela organização (por exemplos: Correção, Melhoria, Implementação, etc.); e versão.

Cada *Issue* criada no GitHub possui um número, gerado automaticamente e este pode ser utilizado como referência para o nome do *branch* que será criado para desenvolver determinada funcionalidade.

3.5. Desenvolvimento das Funcionalidades

Após o recebimento da solicitação de alteração, o pesquisador desenvolvedor realizará a implementação das funcionalidades requisitadas.

Para desenvolver a funcionalidade solicitada, o pesquisador desenvolvedor deverá criar localmente um *branch*, gerando assim uma linha de desenvolvimento separada da versão principal, que após a finalização da implementação da funcionalidade conterá a versão atual do software, adicionada a alteração realizada.

O grupo de pesquisa deve estabelecer um padrão para a criação de nomes de *branches*, a fim de que tais nomes sejam reconhecidos nas revisões. Para tanto, deverá ser utilizado o número da *Issue* como identificador para que possa ser possível identificar a origem da mudança e o que a mudança solicita.

Conforme a implementação da funcionalidade evolui, é possível efetivar as alterações localmente atravéspor meio do comando *commit*. A realização de *commits* frequentes é importante para manter as alterações efetivadas no repositório de modo que não se reverta alguma alteração incorretamente, perdendo-se uma implementação importante.

Após a finalização do desenvolvimento da funcionalidade, as alterações deverão ser sincronizadas com o repositório remoto e posteriormente deverá ser aberto o *pull request*, finalizando assim todo o processo de desenvolvimento.

A Figura 2 apresenta o processo em um contexto geral para o desenvolvimento das funcionalidades na visão do pesquisador desenvolvedor.

3.6. Revisão de Código e *Merge* das Mudanças

Com o processo de desenvolvimento finalizado, é imprescindível que se realize uma revisão de código para identificar possíveis falhas ou dúvidas do pesquisador responsável que surgiram em decorrência da revisão. Caso o código falhe na revisão, o pesquisador desenvolvedor deve ser avisado sobre as falhas e como corrigi-las. Este cadastro deverá ser realizado no *pull request* do código alterado.

Após o código passar na revisão de código, será necessário realizar o *merge* das alterações realizadas pelo desenvolvedor e, neste ponto, podem haver conflitos entre o

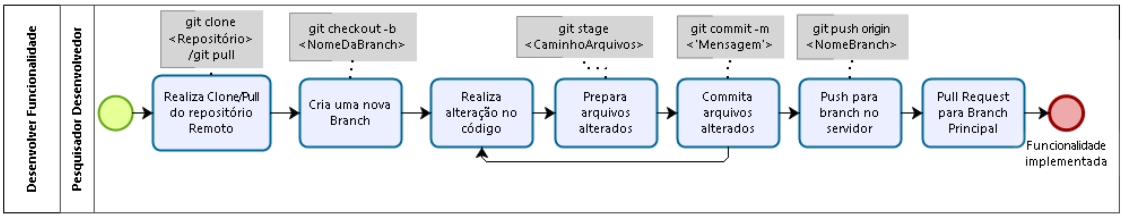


Figura 2. Fluxo do desenvolvimento de novas funcionalidades na ferramenta na visão do pesquisador desenvolvedor

código presente no *branch* de desenvolvimento e as alterações solicitadas pelo desenvolvedor. O pesquisador responsável deverá resolver estes conflitos, de maneira que o código fique coerente e funcional para a ferramenta.

Após a realização da revisão, o *branch* criado para o desenvolvimento deverá ser excluído, pois o mesmo não se faz mais necessário. A Figura 3 demonstra o fluxo do processo de revisão de código e merge das funcionalidades enquanto que a Figura 4 apresenta o processo de desenvolvimento de funcionalidades adaptado para contemplar a revisão de código.

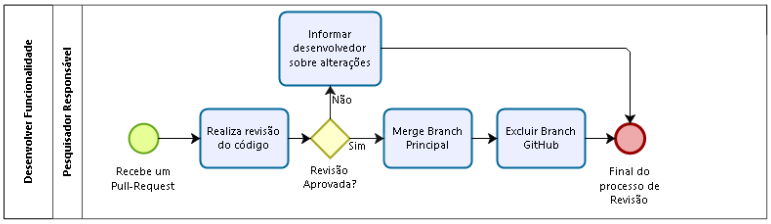


Figura 3. Fluxo do revisão de código e merge de novas funcionalidades na ferramenta na visão do pesquisador responsável

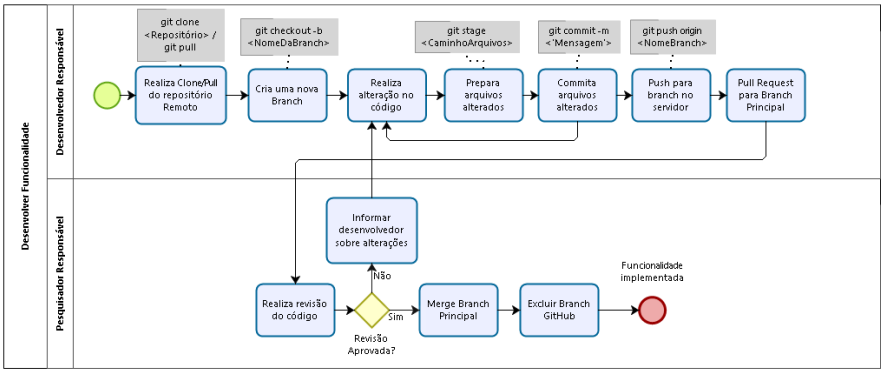


Figura 4. Fluxo de desenvolvimento, revisão de código e merge de novas funcionalidades na ferramenta no contexto geral

3.7. Novas Versões

Considerando o ambiente de grupos de pesquisa, uma nova versão deverá ser disponibilizada após seis meses de desenvolvimento. Será utilizado o seguinte padrão: No fechamento da primeira versão do software no ano, após seis meses de desenvolvimento, será atribuído a versão *Major* o valor do ano corrente, e a versão *Minor* o valor 1. Para abertura da segunda versão no ano, após mais seis meses de desenvolvimento, o valor para a versão *Major* irá se manter e somente a versão *Minor* será incrementada, passando

a assumir o valor 2. Este padrão irá refletir não só na versão do software como também no nome dos *branchs* que devem ser criados para cada versão, para que esta possa ser considerada como finalizada.

Para isso, quando finalizado o processo de desenvolvimento e testes de todas as funcionalidades de uma versão, deverá ser aberto um novo *branch* no qual será retirada a versão oficial a partir do *branch* de desenvolvimento, com o número da versão sendo referenciada. Este processo garante que uma versão antiga não será perdida por sobrescrita de *branches* no GitHub. Essa abertura de versões deverá ser realizada pelo professor orientador ou pelo pesquisador responsável.

A Figura 5 demonstra a como é definida a abertura de versões, e os nomes a serem utilizados para cada versão e cada *branch*.

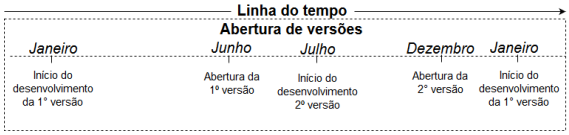


Figura 5. Período de início do desenvolvimento e abertura de versões

Ao realizar uma correção em uma versão, a mesma deverá ser replicada para as versões seguintes e posteriormente, no *branch* de desenvolvimento, para que essas correções mantenha a integridade nas próximas versões que serão abertas. A figura 6 demonstra a maneira correta de propagar as correções realizadas em versões que o desenvolvimento já foi finalizado.

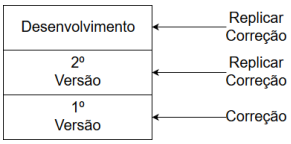


Figura 6. Esquema de propagação das correções realizadas em versões

4. Relato de Experiência: Implantação das Políticas de GCS

Considerando as políticas definidas, este relato de experiência apresenta a implantação destas políticas de GCS em um ambiente de grupo de pesquisa que tem como foco a realização de pesquisas em SBSE (Engenharia de Software Baseada em Busca).

Para atuar neste campo foi criado dentro do Departamento de Informática da Universidade Estadual de Maringá um grupo de pesquisa que tem como objetivo o desenvolvimento de ferramentas de suporte a problemas de SBSE. No referido grupo de pesquisa atuam professores, alunos de graduação e alunos de mestrado da Universidade Estadual de Maringá. A principal ferramenta desenvolvida por este grupo é denominada OPLA-Tool [Féderle et al. 2015].

Durante o desenvolvimento da OPLA-Tool não houve uma preocupação com processos de gerência de configuração. O processo de desenvolvimento de uma nova funcionalidade passava por um processo não estruturado e não padronizado, permitindo que cada aluno desenvolvedor atuasse da maneira que achasse conveniente. Ao final do desenvolvimento todos as alterações de código eram enviados a um aluno de mestrado que era responsável por realizar *merges* manuais para que todas as funcionalidades desenvolvidas fossem integradas a uma única versão do software, o que gerava um grande esforço e aumentava o risco de problemas no software.

Ao realizar a identificação dos itens de configuração, foi constatado apenas um tipo de artefato que seria relevante para ser gerenciado pelo controle de versão, sendo este o código-fonte. Assim sendo, o código fonte do projeto foi versionado em um repositório público no GitHub, e este repositório foi configurado para que desenvolvedores possam contribuir com o projeto de maneira ordenada por meio do processo definido neste trabalho.

Os papéis foram definidos exatamente como proposto na seção 3.2, ficando sobre responsabilidade dos professores a criação das solicitações de mudança que devem ser abertas no GitHub, bem como a abertura das versões para liberações oficiais. As revisões de código e *merges* ficaram sob responsabilidade de um aluno de mestrado (mais experiente e designado pelos professores orientadores) que possui conhecimento prévio da ferramenta. O desenvolvimento foi alocado a alunos que desenvolveram os trabalhos de conclusão de curso com foco na OPLA-Tool, bem como alunos de iniciação científica e mestrado que fazem parte do projeto. Estes fizeram uso do Git para realizar o versionamento dos artefatos. A abertura de novas versões foi realizada como descrito na Figura 7 e a a propagação de correções foi definida conforme apresentado na Figura 8.

Figura 7. Período de início do desenvolvimento e abertura de versões definidos para OPLA-Tool

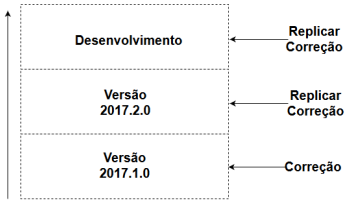


Figura 8. Esquema de propagação das correções realizadas em versões anteriores definidos para OPLA-Tool

5. Considerações Finais

O presente trabalho teve como objetivo propor políticas de GCS para ambientes de grupos de pesquisa. Para tanto foi realizada uma pesquisa sobre o que é e como ocorre o processo de gerência de configuração, bem como identificado, como funciona, em geral, o ambiente de desenvolvimento de grupos de pesquisa, em especial o grupo de pesquisa em SBSE do Departamento de Informática da Universidade Estadual de Maringá.

Dentro deste contexto, foi definido um processo de gerência de configuração segundo o que propõe a literatura. Esse processo propõe mudança nos papéis necessários para implantação de GCS em relação aos especificados na literatura e modifica as tarefas para que possam ser executadas por estes papéis. Também especifica os fluxos de desenvolvimento que devem ser aplicados, propondo revisões técnicas como parte da garantia da qualidade do software e como deve ser a abertura de novas versões do software bem como o fluxo de correção de problemas.

Como trabalhos futuros pretende-se aplicar o processo de GCS definido em ambientes de grupo de pesquisa diferente do grupo de SBSE, para permitir uma melhor

adequação das políticas de GCS já definidas, bem como a definição de novas práticas, relevantes para o contexto de grupos de pesquisa. Além disso, pretende-se elaborar um processo de *deploy* automatizado com entrega contínua para a ferramenta OPLA-Tool e implantar testes automatizados como pré-requisito para realização do *Merge* no GitHub.

Referências

Chacon, S. and Straub, B. (2014). *Pro git*. Apress, New York, NY, USA.

Crnkovic, I., Asklund, U., and Dahlqvist, A. P. (2003). *Implementing and integrating product data management and software configuration management*. Artech House, Norwood.

Dart, S. (1990). Spectrum of functionality in configuration management systems. Technical Report CMU/SEI-90-TR-011, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA.

Duck, B. (2017). Compare repositories.

Estublier, J. (2000). Software configuration management: A roadmap. *In ICSE '00 Conference on The Future of Software Engineering*, pages 279–289.

Féderle, E. L., do Nascimento Ferreira, T., Colanzi, T. E., and Vergilio, S. R. (2015). Oplatool: A support tool for search-based product line architecture design. In *Proceedings of the 19th International Conference on Software Product Line*, SPLC '15, pages 370–373, New York, NY, USA. ACM.

Figueiredo, S., Santos, G., and Rocha, A. R. (2004). Gerência de configuração em ambientes de desenvolvimento de software orientados a organização. *Simpósio Brasileiro de Qualidade de Software, Brasília*.

GPL, G. (2017). General public licence.

Leon, A. (2015). *Software configuration management handbook*. Artech House, Norwood.

Loeliger, J. and McCullough, M. (2012). *Version Control with Git: Powerful tools and techniques for collaborative software development*. O'Reilly Media, Sebastopol, 2 edition.

Pressman, R. S. (2011). *Engenharia de software: uma abordagem profissional*. Amgh Editora, Porto Alegre, 7 edition.

Scott, J. A. and Nisse, D. (2001). *Guide to Software Engineering Body of Knowledge*. IEEE Computer Society Press, Livermore.

Sink, E. (2011). *Version Control by Example*. Pyrenean Gold Press Champaign, IL, 1 edition.

Sommerville, I. (2011). *Engenharia de software*. Pearson Prentice Hall, São Paulo, 9 edition.

Spinellis, D. (2012). Git. *IEEE Software*, 29(3):100–101.

Systematic Literature Review on Web Performance Testing

Guilherme Legramante[1], Maicon Bernardino[1], Elder Rodrigues[1], Fábio Basso[1]

[1] Laboratory of Empirical Studies in Software Engineering
Universidade Federal do Pampa (UNIPAMPA)
Caixa Postal 15.064 – 97.546-550 – Alegrete – RS – Brazil

guilhermelegramante@gmail.com,

{bernardino,elderrodrigues,fabiobasso}@unipampa.edu.br

Abstract. *Performance Testing is essential to ensure the quality and scalability of Web applications. A well-defined process may guide Performance Testing Engineer in conducting this task. We intended to enlighten some major inputs related to web performance testing. For this, we have formulated and executed a given protocol, according to the Systematic Literature Review (SLR) protocol in Software Engineering. So, 37 papers were selected/analyzed and we have extracted their most relevant contribution in order to answer our research questions. This analysis enabled us discovering preeminent performance testing profiles/roles, approaches, artifacts, methods, stages or phases and activity flows that have been reported in the literature. We believe that, despite those several studies that mapping performance test context, there are a few remarks in which a clarification might be needed, once there is no well-established process that comprises the whole activities mapped as well as established a relation with other studies. Therefore, this study intends to provide relevant input that one may establish a novel web performance testing process.*

Resumo. *Teste de desempenho é essencial para garantir a qualidade e a escalabilidade de aplicações web. Um processo bem definido pode orientar o Engenheiro de Teste de Desempenho na condução desses testes. Pretendemos investigar os principais conceitos relacionados a Teste de Desempenho. Para isso, executamos uma Revisão Sistemática da Literatura. Assim, 37 artigos foram analisados e suas principais contribuições foram extraídas visando responder às nossas questões de pesquisa. Essa análise nos permitiu descobrir perfis / funções de teste de desempenho proeminentes, abordagens, artefatos, métodos, estágios ou fases e fluxos de atividades que estão relatados na literatura. Apesar de alguns estudos elencarem as entradas e saídas relacionadas ao processo de teste de desempenho, não há um processo bem estabelecido que abranja todas as atividades mapeadas e estabeleça uma relação com outros estudos. Desta forma, este estudo fornece informações relevantes para estabelecer um novo processo de teste de desempenho da web.*

1. Introduction

Web applications need to respond quickly to user actions, once that user engagement is conditioned by the speed in which the system responds to their actions. For instance, a few

seconds of waiting in a determinate task may deflect a purchase in a virtual store, since this delay might make a possible client changing his mind. So, knowing the breaking point of a given system may ensure proper functioning, which allows designing safety mechanisms to support the expected load. Considering the crescent number of systems and Web applications, with a large demand for infrastructure and scalability, we consider that it is necessary to further research that foresees activities related to this demand.

Based on assumptions presented before, it seems a matter of huge importance accessing mechanisms for software performance assessment. So that, by employing a Performance Testing it is possible to plan, execute, monitor, and analyze the results of a system under certain conditions, thereby obtaining the possible expected behaviors of a given software when subjected to these conditions [Bernardino et al. 2016]. In this context, Software Performance Engineering (SPE) [Woodside et al. 2007] can be divided into two general approaches. The former one focuses on early-cycle, by a predictive model-based, *i.e.* performance evaluation and modeling. The latter one adopts a measurement-based approach that involves its late-cycle, *i.e.* performance testing. Considering these assumptions, this research addresses the latter approach, since it enables us to investigate all phases, stages, and activities of performance testing.

For the sake of automating some performance testing tasks, some techniques have been developed. The Capture and Replay (CR) is one of the most used and widespread techniques in performance test automation tools. This technique consists of writing scripts automatically through some system execution functionality. Then, the generated script is executed and the test is performed. Another technique broadly utilized is Model-Based Testing (MBT), in which a model is created, using a specific notation, to generate a test according to planning in the model. MBT may use formal methods to validate the system under test and can be applied either at the hardware or software level. Based on system requirements, test cases are generated based on the models generated [Rodrigues et al. 2015].

Although there are already numerous tools and approaches to performance testing, we assume that information could be better summarized. Discrepancies among defined content by approaches and techniques might hinder the flow of performance testing activities. Based on this, we conducted an SLR on the performance testing area, seeking out comply with to meet this demand. Thus, we selected 37 studies with the purpose of explains the main concepts related to web performance testing. We summarize SLR results in a feature model, followed by a brief explanation of each located input. This provided us an area overview, according to our research questions.

The present paper is organized as follows. In Section 2, we present paper background. Section 3 SLR protocol/execution is presented, followed by SLR results in Section 4. In Section 5, conclusion and future works are discussed.

2. Background

Performance testing is a possibility to plan, execute, monitor, and analyze the results of a system under certain conditions, thereby obtaining the possible expected behaviors of determining software, when subjected to these conditions [Bernardino et al. 2016]. According to Freitas and Vieira [Freitas and Vieira 2014], performance testing is a test that aims to evaluate the performance of the system at a given load scenario. In summary, per-

formance testing provides a load simulation and measurement to detect bottlenecks and the breaking point in which a system crashes under a certain workload.

Woodside [Woodside et al. 2007] defines Software Performance Engineering (SPE) as representing the entire collection of software engineering activities and also related analyses throughout the software development cycle, which are directed to meeting performance requirements. Revealing bottlenecks and achieving improvements in scalability and software performance are some of the main objectives of SPE. In that sense, SPE is classified into two general approaches: predictive model-based and measurement-based. The former one concentrate on the early-cycle and the latter one in the late-cycle of the software development life cycle. Hence, performance testing is associated with a measurement-based approach.

According to Meier *et al* [Meier et al. 2007], a performance testing may be divided into two categories; Load Testing and Stress Testing. A load testing aims to determine a System Under Test (SUT) behavior, which in turn, depicts an application subject a workload. It should be noticed that the load test is conducted to assess if the given system meets specified non-functional requirements. Stress testing places a system under higher-than-expected traffic loads so developers can see how well it works above its expected capacity limits. Moreover, Molyneaux [Molyneaux 2009] includes soak, or scalability, testing, in a way that a soak testing may subject the SUT to a load for a long period, in which some problems dismissed in other categories may become noticeable.

3. Systematic Literature Review Protocol

SLR scope is conducting Performance Testing study area, seeking out for guidelines, taxonomy, process, or frameworks that support activities related to planning, execution, monitoring, and reporting of test results. In this research, we endorsed the protocol proposed by Kitchenham [A. Kitchenham 2007] in SLR. The GQM (Goal, Question, Metric) paradigm [Caldiera and Rombach 1994] usage means to resume the review scope: *For the purpose of* **identify / characterize**, *with respect to* **performance testing processes**, *from the viewpoint of* **performance test engineers and researchers**, *in the context of* **software engineering environment**.

3.1. Research Questions

We assigned the following Research Questions (RQ): **RQ1.** What are the performance testing profiles/roles, artifacts, methods or approaches? **RQ2.** What are the performance testing stages and phases? **RQ3.** What are the performance testing activities, steps or tasks? **RQ4.** What are activity or task flows performed in performance testing?

3.2. Question Structure

Research questions (RQs) are design by means of (PICOC) [Wohlin et al. 2012] criteria, that takes into consideration as follows: **P**opulation: published research on software; **I**ntervention: performance testing; **C**omparison: general comparison of the retrieved processes; **O**utcome: published papers on Performance Testing; **C**ontext: software testing practice and research.

3.3. Search Strategy

To perform the proposed search, we selected the following databases: ACM Digital Library; Engineering Village; IEEE Xplore Digital Library; ScienceDirect; Scopus. These databases were chosen because they stored the main publications in the computer science field and they also offered a web-based search engine. Hence, we elaborated search strings according to each database particularity. The generic string that was used to derive the other strings is shown in Table 1.

Table 1. Generic Search String.

`(process OR framework OR method OR approach OR guideline OR taxonomy OR ontology) AND (web) AND ((performance OR load OR stress OR workload) AND (test OR testing)) AND (stage OR phase OR activity)`

3.4. Selection Criteria and Quality Assessment Criteria

Inclusion criteria and exclusion criteria were defined and applied in order to filter in the initial search. Besides, for a study to be included, it must satisfy at least one inclusion criterion and at least one exclusion criterion as a means to excludes the study from our analysis. To evaluate selected studies' relevance and also answering some research questions, we used quality assessment criteria. The quality assessment criteria are featured which may be exploited in two stages: the first stage being the individual evaluation of each researcher, to reduce bias probability; the second stage is where researchers should reach a consensual note about publications in a "divergent state" in a quality measurement grade. Due to space limitation, we provide a set of documents in an online repository that including the list of inclusion and exclusion criteria; quality assessment criteria, and; data extraction strategy (`http://bit.ly/2wKwUPp`).

3.5. Selection Process

(1) **Pilot Search Strategy:** In order to verify the quality of the proposed search string, the approach called Search-Based String Generation (SBSG) [Souza et al. 2018] was applied. The approach is based on precision and sensitivity indexes calculation. The precision is the ability to identify the amount of *irrelevant* studies, while the sensibility is a measure to identify all of the *relevant* studies. When precision is zero, no irrelevant study is detected. This approach applies an Artificial Intelligence technique through the Hill-Climbing algorithm suggested by Russell [Russell and Norvig 2016], which allows the measurement of precision and sensitivity indexes for a set of keywords and an initial set of selected papers. The proposed *string* was submitted on the basis of 8 (eight) pre-selected studies. Thus, the achieved results were 11.27% precision and 79.49% sensitivity. (2) **Search Databases:** The strings were generated using selected terms and synonyms and were run in the selected databases, resulting in an initial aggregation of studies; (3) **Removal of Duplicates:** The results of initial selection were filtered-out for duplicated entries; (4) **Selection Studies:** In this step, we read separately the title and the abstract (reading the introduction and conclusion when necessary) of each study. Here, we decided to select or reject an article following defined inclusion and exclusion criteria; (5) **Quality Assessment:** The selected studies from inclusion and exclusion criteria application were

submitted to quality assessment criteria; (6) **Data Extraction:** To answer to RQs, the selected/classified studies were obtained and relevant data were extracted using a form.

Our initial selection was conducted in May 2019, on ACM Digital Library, Engineering Village, IEEE Explore, Science Direct, and Scopus and provided close to 1328 results. After filtering out duplicate entries, the number of results was reduced to 1081. The number of duplicate entries was quite large and this might be attributed to papers being revised from conferences publications into journal articles, being extended and submitted in later conferences, and overlapping results from databases. After separately applying inclusion and exclusion criteria fifty-two (52) studies remained. Finally, the quality assessment reduced the number of results was reduced to thirty-seven (37) papers.

4. Results and Discussion

This section presents the SLR results, in which thirty-seven (37) studies are discussed to respond to defined research questions. Figure 1 provides us an overview of obtained results in the form of a feature model. Nodes Test Plan, Model, Planning and Analysis are of optional nature, *e.g.*, an approach that does not use a model as an artifact, this is not required.

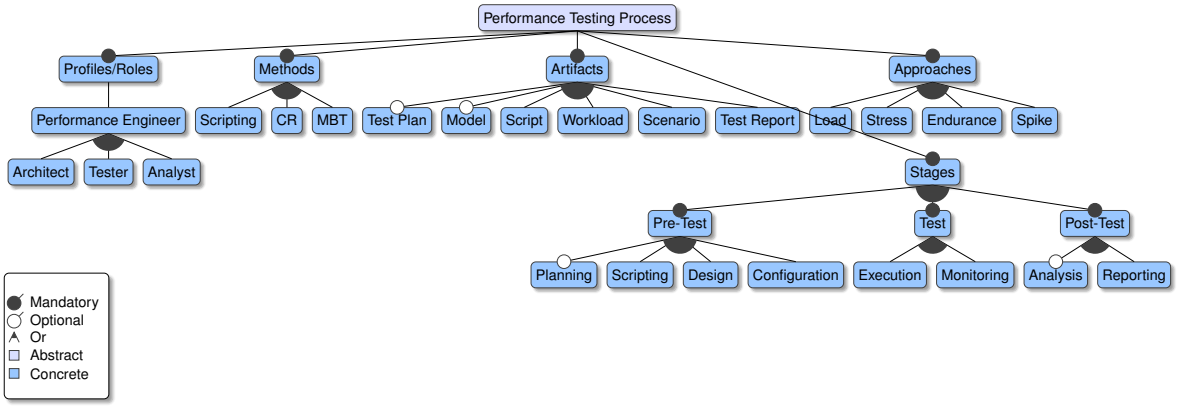


Figure 1. Overview of web performance testing process.

4.1. RQ1. What are the performance testing profiles/roles, artifacts, methods or approaches?

4.1.1. Profiles/Roles

After to analyze selected studies, we identified the following four (4) profiles/roles: (1) **Performance Engineer:** The performance engineer must have the knowledge to support all stages, phases, and activities of the performance test. This role can be specialized in other roles (Performance Architect, Performance Tester, and Performance Analyst). Some papers make reference directly or indirectly to this role [Subraya 2006] [Xu et al. 2014] [Van Der Ster et al. 2011]. (2) **Performance Architect:** This role is involved within Design and Configuration Phases and it must make a connecting bridge between early phases and testing execution. A Performance Architect must-have skills to make design and configuration activities. The term ”Performance Architect” is reported in Subraya [Subraya 2006] paper. (3) **Performance Tester:** This role is directly related

to the testing execution phase. A Performance Tester is the one who should "operate" performance testing, making use of available tools for performing this activity. A few papers bring this role within another nomenclature as User and Developer [Tselikis et al. 2007] [Pfau et al. 2017]. We merged these terms in Performance Tester, once we believe to be more suitable for this context. (4) **Performance Analyst:** Performance Analyst has participation in early and late performance testing phases. He is responsible for initial testing planning and documentation, providing input to subsequent phases, design, and configuration. This role is also present after testing execution, on this account, it is employed in the analysis and reporting phases. This role is not directly reported in the chosen papers. However, Subraya [Subraya 2006] refers to their activities, without specific nomenclature.

4.1.2. Artifacts

The most relevant artifacts are presented to support performance testing activities: **Performance Test Plan:** is a document elaborated by a Performance Analyst as a means to, provide support and guiding the team in the whole test activities. In this document general testing features, such as testing type, scope, approach, and the steps to achieve performance testing goals are explained. This artifact is generated in the planning phase and it is reported in some papers that focuses on this phase [Meier et al. 2007] [Freitas and Vieira 2014] [Yin et al. 2008] [Huang et al. 2011]; **Model:** is used as input in technique known as Model-Based Testing. A model is an abstraction of software behavior that enables reuse and facilitates the understanding of the flow of activities performed by the test [Yin et al. 2008]; **Performance Script:** is the main input artifact for running the test. Through it, the test execution flow is defined, since a script is represented by a set of instructions and may be obtained in an automatic or manual manner. In the former, scripts are generated through tools that use a capture and replay mechanisms [Subraya 2006]. On the latter one, in manual form scripts are generated through a programming language code; **Workload:** is responsible for modifying the SUT situation through its different configurations. A workload may vary based on the test approach and it includes a number of users, concurrent active users, data volumes, and transaction volumes, along with the transaction mix. For performance modeling, a workload is associated with an individual scenario [Pfau et al. 2017]; **Performance Scenario:** defines as a set of steps in an application [Meier et al. 2007]. Moreover, a scenario may map a given application context, within a determinate workload for a user profile, it should be modeled on the basis in usage patterns and log files. **Performance Test Report:** reports the data retrieved by test execution. This report must contain test results, organized in a way that allows their interpretation by stakeholders. Meier *et al* [Meier et al. 2007] lists six key components, which are not mandatory, of a technical report: a results graph, a table with single-instance measurements, a workload model, test environment, general observations and a references section.

4.1.3. Methods

Three methods are related to performance testing conducting [Rodrigues et al. 2015]: **Scripting:** This method involves technique support by the manual script where the performance tester writes a set of code statements, which are going to be inputted to a load

generator to providing a workload in a given scenario; **Model-Based Testing:** In this method, a software behavior under test is verified according to their model. It has some advantages such as enabling the application of models for appropriate testing models creation, as well its use in performance testing; **Capture and Replay:** This method consist of recording the execution of the application's functionalities for the generation of test scripts for the later execution of these scripts simulating the execution of the application's functionalities.

4.1.4. Approaches

Subraya [Subraya 2006] presents a set of four (4) performance testing approaches called LESS (Load, Endurance, Stress, Spike): (1) **Load Testing:** A load is a quantitative of users which compete to increase traffic of application. It is useful for determining the break-point and checking when bottlenecks begin to emerge; (2) **Endurance Testing:** An endurance testing is directly related to the reliability of the application. Different test execution times can be set to check the behavior of the application in different scenarios based on the duration of the test. Endurance testing may be to performed on a normal load or on a stress load, but the main focus of this approach is the test duration; (3) **Stress Testing:** A stress testing is similar to the load testing. However, the stress testing aims to check how the application handles in its limit. Therefore, it helps researches to identify the load that the system can handle before breaking down or degrading quickly; (4) **Spike Testing:** A spike testing is conducted to verify application behavior under a surge in a short duration. The application is subjected to a sudden load increase.

4.2. RQ2. What are the performance testing stages and phases?

For our purpose, stages were mapped as being the activities group in the high level, which may have one or more phases. We identified three (3) stages and nine (9) phases in the performance testing context. The stages and phases are as follows:

4.2.1. Pre-Test

This stage comprises previous phases to test execution. The test definition and preparation are prepared in this stage. The Pre-Test stage has four (4) phases: (1) **Planning:** In this phase occurs test definition. Major requirements related to the test are mapped and some factors should be analyzed, such as network and infrastructure environment, business functions related with the performance requirements and everything that may be really relevant to the test; (2) **Scripting:** This phase involves activities that focus on script elaboration, which can be obtained by different means. For instance, supported by models and uses the MBT or CR for a automatic generation or also by means of coding, where scripts are made from specific programming language; (3) **Design:** Using the test specifications defined in the planning phase, performance testing is designed taking into account environments particularities and performance testing goals; (4) **Configuration:** It is the last phase before test execution. In this state, adjusts and setting performance testing are made. Issues like workload type, performance testing type and tool functionalities should be considered as well as infrastructure issues.

4.2.2. Test

After the Pre-Test stage, in this stage occurs the Test stage, moment in which test execution is realized. **Execution:** In this phase workload is generated and the SUT is monitored to obtain inputs that indicate the main bottlenecks and the behavior of the system under this load. In addition to this monitoring, test should provide mechanisms to collect necessary metrics, which were defined in the Pre-Test stage. **Monitoring:** Defined metrics as throughput, response time, hits per second must be monitored during test execution. This monitoring allows performance testing roles to obtain outputs for subsequent phases, analysis and reporting in post-test stage.

4.2.3. Post-Test

This stage encompasses the phases that postdate Execution, Analysis and Reporting respectively. 1. **Analysis:** In this phase result analysis is conducted, according to the metrics that were collected during test execution. The support by a specific tool is very important in this phase, once manual execution is impracticable. However, this analysis is directly related to the collection of metrics results exposure during the test, not the analysis by the performance engineer in a decision making process; 2. **Reporting:** This phase is the sequence of the analysis phase. In this phase, test results are reported. This report might vary between a detailed and automated report, depending on the tool used, or a report with minor information, so that the performance engineer/architect has the task of interpreting performance testing report results.

It was possible to verify that most of studies addresses test execution. Other issue demonstrated in this table is an evaluation type, achieved by selected studies. Case studies are more recurring in this context, once eleven (11) empirical studies this type were founded, followed by experiments with seven (7) studies that used this approach to assessing the study. Another relevant question to be highlighted is that the majority of studies are not focused on all phases and stages of a performance testing, on the grounds that they focus on some specific phases.

4.3. RQ3. What are the performance testing activities, steps or tasks?

Based on the selected papers, we found one hundred thirty eight (138) performance testing activities, steps or tasks, so it is unfeasible to detail them in this paper In Figure 2 it is possible to identify that assumption, as well as the trend evidenced in previous research questions. There is a greater concentration of activities in the test execution phase, where we identified forty seven (47) activities related to this phase. The other phases have a similar commensurate of activities, ranging from fifteen (15) to twenty two (22) activities, except Scripting and Reporting phase where twelve (12) and five (5) related activities, respectively, were found.

It is relevant to allude that some activities can have differences only in their nomenclature, for the sake of having same objective in practice. It is also worth emphasizing that due to the varied possibilities for a performance test, not necessarily all the mapped activities must be used, as a result of the particularity of each test, a certain group of activities will be executed.

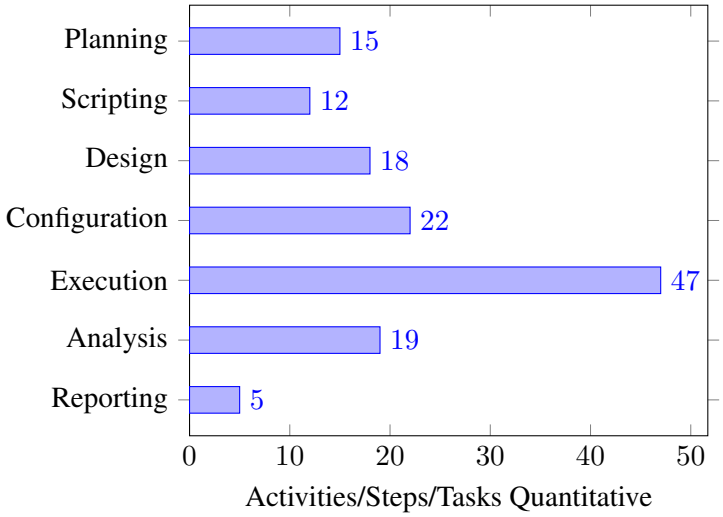


Figure 2. Relation Activities/Steps/Tasks Quantitative X Phases.

4.4. RQ4. What are activities/tasks flow performed in performance testing?

The mapping of the stages, phases and subsequent activities reported in the selected studies allowed us allowed us to understand that the phases can be organized in a circular manner. Performance testing may be thought of as a sequential activity and may be instantiated as many times as necessary. In this flow the sequence starts in the Planning, following the six (6) next phases until completing the cycle. Another reason that motivated us to model the flow in this manner is due to the fact that is the large variety of activities and tasks that do not include all mapped phases, making it possible to understand the sequence of the test independent of the activity described to contemplate the phases in their totality or not.

5. Conclusion and Future Work

Performance Testing is an important area that has a direct influence on application reliability and scalability. Based on this, mapping the main concepts of the area is important when trying to define a generic process in the context of performance testing for Web applications. In this paper, we present the protocol, execution, and results of an SLR for an overview of performance testing for Web applications. Hence, thirty-seven (37) studies were selected and analyzed to obtain subsidies that answered our research questions. We assume that our main contribution was obtained through SLR results, which allowed us to map the main concepts related to the performance testing area, encompassing all its stages and phases. Our results were reported in a textual description of them and by a feature model that encompasses the whole SLR results.

Currently, we are working on the development of a broad and generic web performance testing process, supporting those involved in performance testing activities with a well-defined process that presents roles, phases, and activities with their respective inputs and outputs. This SLR is the initial point of this process, once despite highlighting the main concepts, it is still necessary that they are refined and confronted with the state of practice in the industrial environment. Thus, we have also employed efforts to conduct a survey to this end.

References

A. Kitchenham, B. (2007). *Guidelines for performing Systematic Literature Reviews in software engineering. EBSE Technical Report EBSE-2007-01*.

Bernardino, M., Zorzo, A. F., and Rodrigues, E. M. (2016). Canopus: A domain-specific language for modeling performance testing. In *IEEE International Conference on Software Testing, Verification and Validation (ICST)*, pages 157–167. IEEE.

Caldiera, V. R. B.-G. and Rombach, H. D. (1994). Goal question metric paradigm. *Encyclopedia of software engineering*, 1:528–532.

Freitas, A. and Vieira, R. (2014). An ontology for guiding performance testing. In *2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, volume 1, pages 400–407. IEEE.

Huang, X., Wang, W., Zhang, W., Wei, J., and Huang, T. (2011). An adaptive performance modeling approach to performance profiling of multi-service web applications. In *Proc. International Computer Software and Applications Conference*, pages 4–13.

Meier, J., Farre, C., Bansode, P., Barber, S., and Rea, D. (2007). *Performance testing guidance for web applications: patterns & practices*. Microsoft press.

Molyneaux, I. (2009). *The art of application performance testing: Help for programmers and quality assurance*. " O'Reilly Media, Inc.".

Pfau, J., Smeddinck, J. D., and Malaka, R. (2017). Automated Game Testing with ICARUS: Intelligent Completion of Adventure Riddles via Unsupervised Solving. In *Extended Abstracts Publication of the Annual Symposium on Computer-Human Interaction in Play*, pages 153–164, New York, NY, USA. ACM.

Rodrigues, E., Bernardino, M., Costa, L., Zorzo, A., and Oliveira, F. (2015). Pletsperf-a model-based performance testing tool. In *2015 IEEE 8th International Conference on Software Testing, Verification and Validation (ICST)*, pages 1–8. IEEE.

Russell, S. J. and Norvig, P. (2016). *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,.

Souza, F. C., Santos, A., Andrade, S., Durelli, R., Durelli, V., and Oliveira, R. (2018). *Automating Search Strings for Secondary Studies*, chapter 558, pages 839–848. Springer International Publishing.

Subraya, B. M. (2006). *Integrated approach to web performance testing: A practitioner's guide*.

Tselikis, C., Mitropoulos, S., and Douligeris, C. (2007). An evaluation of the middleware's impact on the performance of object oriented distributed systems. *Journal of Systems and Software*, 80(7):1169–1181.

Van Der Ster, D. C., Elmsheuser, J., Garcia, M. U., and Paladin, M. (2011). HammerCloud: A stress testing system for distributed analysis. In *Journal of Physics: Conference Series*, volume 331, Taipei, Taiwan.

Wohlin, C., Runeson, P., Hst, M., Ohlsson, M. C., Regnell, B., and Wessln, A. (2012). *Experimentation in Software Engineering*. Springer Publishing Company, Incorporated.

Woodside, M., Franks, G., and Petriu, D. C. (2007). The future of software performance engineering. In *Future of Software Engineering*, pages 171–187. IEEE.

Xu, X., Jin, H., Wu, S., Tang, L., and Wang, Y. (2014). URMG: Enhanced CBMG-based method for automatically testing web applications in the cloud. *Tsinghua Science and Technology*, 19(1):65–75.

Yin, J., Ming, Z., Xiao, Z., and Wang, H. (2008). A web performance modeling process based on the methodology of learning from data. In *Proceedings of the 9th International Conference for Young Computer Scientists, ICYCS 2008*, pages 1285–1291, Zhang Jia Jie, Hunan, China.

A Unified Feature Model for Scrum Artifacts from a Literature and Practice Perspective

Luciano A. Garcia[1], **Edson OliveiraJr**[1], **Gislaine Camila L. Leal**[1], **Marcelo Morandini**[2]

[1]Informatics Department, State University of Maringá (UEM)
Maringá, Brazil.
[2]School of Arts, Sciences and Humanities,University of São Paulo (USP),
São Paulo, Brazil.

lucianogarcia11@hotmail.com, edson@din.uem.br, gclleal@uem.br, m.morandini@usp.br

Abstract. *Scrum has become one of the most popular Agile methods. Among its main elements are its artifacts. These artifacts are related to the requirements required for the software and how they will be worked on during a Scrum interaction called Sprint. Given the importance of artifacts in the Scrum structure, evidence of the adaptations of these artifacts was collected with the aid of a systematic mapping study and a survey literature with practitioners of the method. Later, we systematized the evidence of adaptations found and built models of features in order to register them and enable users of the methods to have a broader understanding of the features that Scrum artifacts can assume.*

1. Introduction

There has always been a search for productivity and quality in software development, which is particularly evident in the agile manifesto[1] [Schwaber and Sutherland 2017], as opposed to traditional software development processes oriented to documentation that until then were the most accepted. Therefore, methods that were already known became popular, among them Scrum.

Scrum is an interactive and incremental approach that replaces the phases of the traditional software development process with the delivery of a high-value suite, which provides an early return of successes and errors in the development of the respective software [Schwaber and Sutherland 2017]. According to the authors of Scrum [Schwaber and Sutherland 2017], the main components of Scrum are the roles, events, artifacts and rules that unite and predict the interaction among them.

In this paper, we show the Scrum artifacts. Scrum artifacts are related to the requirements needed for the software to be developed, which and how these requirements will be developed in Scrum interactions and finally a functional version of these requirements.

In view of the importance of artifacts for Scrum and, consequently, for software development, we seek evidence of adoption and adaptation in the literature and in the experience of practitioners in the use of Scrum, thus we try to answer the following research question "***How have been Scrum artifacts adapted throughout actual software***

[1]https://agilemanifesto.org

development projects?". To this end, we use information about the artifacts that are part of a Systematic Mapping Study (SMS) of Scrum practices and complement it with information obtained through a Survey with Scrum practitioners. To organize and relate the evidence found for the Scrum artifacts in the SMS and the Survey, we used features models [Czarnecki et al. 2005].

2. Background and Related Work

In this section we present the main theoretical concepts to support our proposal.

2.1. The Scrum Framework and Artifacts

According to the authors of Scrum in [Schwaber and Sutherland 2017], Scrum aims to develop, deliver and maintain complex products. It is defined as a framework in which people approach complex and adaptive problems in a productive and creative way to deliver products with the highest possible value.

Scrum uses an iterative and incremental approach to improve predictability and risk control. Scrum is made up of Scrum teams linked to roles, events, artifacts and rules. Each component has a specific purpose and is essential for the use and success of Scrum. Although there are many elements involved in the Scrum dynamics, we emphasize the Scrum artifacts in this paper.

The Scrum artifacts are designed to maximize the transparency of information, thus everyone has the same understanding of what is actually done. Therefore, Scrum is composed of three main artifacts, namely: Product Backlog, Sprint Backlog and Increment.

The **Product Backlog (PB)** is an ordered list of everything needed in the product, in the case of software that will be developed. It is the only source of requirements and software changes. The PO is responsible for the PB, including, updating and ordering its content. The PB and its items must be visible to all stakeholders, assimilating the pillar of transparency preached by Scrum.

The Sprint Backlog (SB) consists of a subset of PB items that were selected by the Dev. Team to be developed, taking into account their priority and the Dev. Team's development capacity. This activity should take place at the Sprint Planning event. Seeking to comply with the Scrum transparency principle, the SB should always be visible to stakeholders, identifying which of its items are ready, then in progress, and which have not yet started. The SB is a source of work to be performed by the Dev. Team in a sprint cycle. As the sprint is being executed, new tasks can be identified to complete the PB items that were selected for the SB, and they must be included in the SB. It is also necessary to add at least one improvement item identified in the Sprint Retrospective event, in order to have a continuous improvement of the process.

The **Increment** is the result of the PB items that were selected for Sprint and that have become a functional version of the product to be delivered. They were inspected in the Sprint Review and released by the PO.

2.2. Feature Modeling

According to [Czarnecki et al. 2005] feature is a system property that is relevant to some stakeholders (customers, analysts, architects, developers, system administrators, etc.) and

is used to capture similarities or differences between systems in a family. The features are organized in diagrams, in the shape of a tree, where the root represents a concept (such as a software system). The diagrams added with descriptions of resources, relationships, priorities, stakeholders, etc., form what is defined as feature models. A feature model is a relationship among a parent feature and its daughter features [Czarnecki et al. 2005].

A feature model is composed of some basic elements, which are: feature diagram, composition rules and relational analysis. In addition, the feature models follow the Czarnecki-Eisenecker notation [Czarnecki and Eisenecker 1999] and the FeatureIDE Tool[2], to elaborate the feature models of this paper.

Figure 1 presents the main elements involved in a feature model and which will be detailed below.

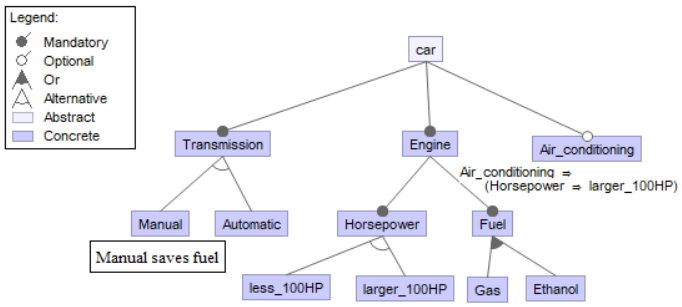


Figure 1. Example of a feature model based on the [Kang et al. 1990] notation

Feature Diagram. In general, the features are organized in the feature diagrams in the form of trees [Sochos et al. 2004] as shown in Figure 2. The features are represented by the tree nodes described in the feature diagram, and in this hierarchy child features can be classified as: *mandatory* - the daughter characteristic must be selected; *optional* - the child feature may or may not be selected; alternative (OR) - at least one of the child features must be selected; and Exclusive OR (XOR) - only one of the child features must be selected.

A feature can also be defined as a concept, and in this case called abstract, this can be seen in Figure 1, for the node where the *car* is written and the others observed in the figure are concrete features.

Composition Rules. They define the relationship between features that cannot be expressed in the features diagram, indicating which combinations of features are valid. In Figure 1, a composition rule is required under the feature *Ar_conditioning* that requires the car to have an engine with a power greater than 1000 in order to support the air conditioning.

Relational Analysis. It is a recommendation specifying when a particular feature should or should not be selected. In the example contained in Figure 1 for the definition of a car, below the *manual* feature there is information that recommends that the choice of a car with the manual transmission tends to be more economical in terms of fuel.

[2]http://www.featureide.com

2.3. Related Work

To the best of our knowledge and based on a non-systematic search there is no related work aimed at modeling the adaptations of Scrum artifacts using feature models.

However, Diebold et al. [Diebold et al. 2015] present how practitioners have adapted Scrum in 10 German software projects. They claim such adaptations occur in the Sprint length, events, team size, and requirements engineering. Practitioners also varied the roles, effort estimations and quality assurance. Certain adaptations come from a previous hierarchical non-agile organization, thus many of them are for good reasons. We corroborate a significant part of the results regarding artifacts adaptations.

3. Adaptations for Scrum Artifacts

To find the evidence regarding the adaptations of the Scrum artifacts in the literature and in the experience of the practitioners, we used clippings from a Systematic Mapping Study (SMS) and also part of a Survey conducted with Scrum practitioners. In the next topics we show the main information related to the SMS and the Survey regarding the Scrum artifacts.

3.1. Adaptations from the Literature

The literature reports different studies on the use of Scrum in the most varied domains and situations. We sought to carry out a Systematic Mapping Study (MSL) by primary studies that revealed elements or features of Scrum adjusted when an organization decides to adopt it.

SMS planning followed the recommendations of Petersen et al. [Petersen et al. 2015]. The SMS had a wide search without date restriction in 5 electronic databases and a manual search in 11 journals and 17 conferences with a date restriction from 07/2007 to 07/2017, in other words, 10 years. Through this process, 281 primary studies were related, in which the inclusion and exclusion criteria defined in [Garcia 2019] were applied, and which resulted in 50 studies selected for the extraction of information. More information about SMS is available at [Garcia 2019] and at `https://doi.org/10.5281/zenodo.3357803`.

Our SMS focuses on the Scrum artifacts, thus the research questions defined are: **RQ1** - What were the Scrum artifacts adapted?; **RQ2** - Do the artifacts follow the recommendations in the Scrum guide?; and **RQ3** - Which techniques were used to prioritize and organize and estimate items in Scrum artifacts?

All studies answering the survey are shown in Table 1.

Table 1. Studies with adaptations in Scrum artifacts.

Study ID	PB	SB	INC
S2, S4, S13, S14, S28, S32, S36, S37, S38, S41, S43, S44, S50, S52, S58	X		
S5, S6, S15	X	X	X
S9, S11, S12, S26, S29, S33, S48, S49	X	X	
Total	**26**	**11**	**03**

Table 1, which answers the research question **RQ1**, shows that of the 50 selected studies, 26 presented information about Product Backlog (PB), 11 about Sprint Backlog (SB) and only 3 about Increment (INC).

Regarding the question **RQ2**, it can be seen that the recommendations of the Scrum guide were only partially followed in the studies. This happened in 19 of the 26 studies, including the following: S4, S5, S6, S9, S11, S13, S12, S14, S15, S28, S29, S33, S36, S37, S38, S41, S44, S48, S50.

To answer the **RQ3** question, only studies that partially met the Scrum recommendations were considered, since the others do not. However, not all studies in partial compliance brought the necessary information to answer this research question. With regard to the **INC** artifact, which is shown in Table 1, the 3 related studies only mentioned the artifact and did not bring any relevant information to MSL. We then analyzed only the other two artifacts **PB** and **SB** for this research question, that are in Table 2.

With regard to PB the information found in the studies was systematized in Table 2. In it one can see a column called Feature ID, which represents an identifier for the information / feature found in the study. The identifier has the following notation: ***XXxnX***. Where: ***XX*** represents the artifact (PB, SB); ***x*** what kind of feature (e - how to estimate, p - how to prioritize and r - how to represent); ***n*** is a sequential number for the feature; ***X*** identifies the origin of the feature (M-SMS, S-Survey)

Table 2. SMS Information for PB and SB.

Study ID	Information	Feature ID	Study ID	Information	Feature ID
Product Backlog			**Sprint Backlog**		
S4	Prioritize: PO and Dev Team worked together Representation: User Stories	PBp1M PBr1M	S6	Selection: Items selected for SB by Dev. Team	SBs1M
S5, S28	Estimate: Planning Poker	PBe1M	S11	Status: SB had the tasks framed in the following status: Not Started, In Progress, Completed, Blocked	 SBt1M SBt2M SBt3M SBt4M
S6	Prioritize:Screening meeting	PBp2M	S33	Progress:features to identify task progress: PB identifier to which the task belongs, Completion time	 SBp1M SBp2M
S9	Prioritize: PO with SM and Dev. Team help Representation: User Stories	PBp3M PBr2M			
S11	Prioritize:PO didn't prioritize Estimate: SM and Dev Team	PBp4M PBe2M			
S12	Prioritize:Only the PO Representation: User Stories	PBp5M PBr3M			
S13, S14	Prioritize: PO and Dev Team worked together	PBp6M			
S29	Prioritize: Only the PO Estimate: PO and Dev Team Representation: User Stories	PBp7M PBe3M PBr4M			
S33	Estimate:Experienced professionals	PBe4M			
S36, S44	Estimate:Planning Poker Representation: User Stories	PBe5M PBr5M			
S37	Estimate: Planning Poker Representation:Use case	PBe6M PBr6M			
S38	Representation: User Stories	PBr7M			
S41	Estimate:Story Points and Value Points Representation: User Stories	PBe7M PBr8M			
S48	Prioritize: LOEs visual representation of software priorities of the customer and what they want in the final state.	PBp8M			
S50	Prioritize:Refinery requirements.	PBp9M			

Looking at Table 2 , the information regarding the **RQ3** question was systematized in the who prioritized, how was estimated and how was represented PB.

Still responding to **RQ3**, now with respect to SB, the 11 studies presented in Table 1 were analyzed, but only 3 studies returned information capable of indicating adaptations for this artifact (see Table 2).

As can be seen in Table 2, the information was systematized for the SB as follows: (i) who selected the items of the PB to be worked on in the SB; (ii) which status the tasks within the SB assumed during its development and (iii) how to monitor the progress of the tasks. The Feature ID column in Table 2, also followed the notation defined for the

features in Table 2 (***XXxnX***), with a small difference for ***x***, where it assumed the following values: s -selection, t- state, p - progress.

With the information listed in Tables 1 and 2, the feature model illustrated in Figure 2 was elaborated.

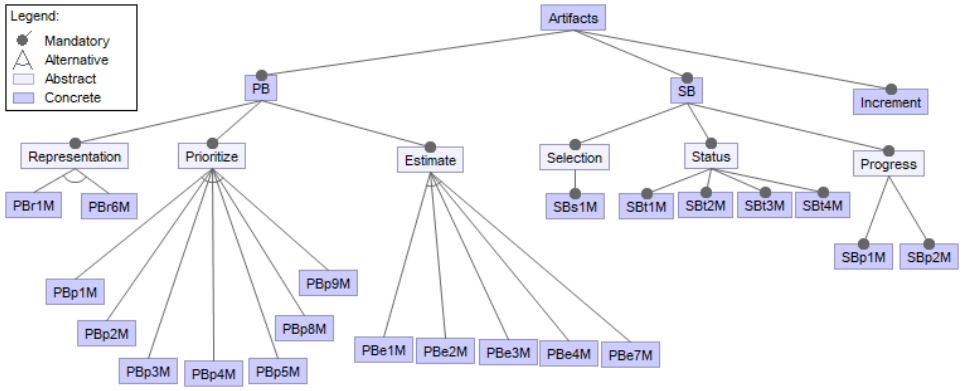


Figure 2. Feature Model for Scrum Artifacts based on the SMS

In the elaboration of the feature model in Figure 2 , some features were removed because they represent the same information for the parent features: Representation, Prioritize and Estimate. For the PB representation form, only the *PBr1M* and *PBr6M* features were considered. The others were the same as the *PBr1M* feature. For the way of prioritizing PB, features *PBp6M* and *PBp7M* were dispensed because they are equal to features *PBp1M* and *PBp5M* respectively. Regarding the way to estimate the PB, the *PBe5M* and *PBe6M* features were discarded because they are the same as the *PBe1M* feature. Regarding the SB, no treatment needed to be done.

3.2. Adaptations from the Practitioners Survey

The evidence of adaptations presented for the Scrum artifacts, in relation to the practitioners, is part of a broader Survey that was conducted, which includes the following steps: Planning, Pilot Test, Data Collection and Analysis of Results.

The survey in question had an international scope in relation to practitioners. A total of 14 respondents were obtained, who demonstrated having a profile with good knowledge, when asked how much experience they had with the use of Scrum in software development. Of these respondents, 21.4% reported having more than 6 years of experience with Scrum, 35.7% between 3 and 6 years of experience, 35.7% between 1 and 3 years of experience and only 7.1% reported having less 1 year of experience with Scrum in software development.

The information found with the SMS helped us to elaborate the research instrument (questionnaire) that was applied to the respondents. More information about the Survey conducted can be verified in [Garcia 2019], we will focus here in relation to the data obtained from the respondents for the Scrum artifacts, due to space limitations.

The answers to the questions investigated in the Survey regarding the Scrum artifacts are presented here in a grouped form, regardless of the percentage, as it is enough once mentioned to characterize it as being used, and also due to space limitations. The

following questions for the PB were investigated: **RQ1** - What form of representation is used in PB?; **RQ2** - How was PB prioritized?; **RQ3** - What form of estimate is used for BP?; **RQ4** - What is the way to measure work progress in PB?; and **RQ5** - What software was used for the management of PB?.

Table 3 presents the systematized responses obtained by the Survey for the PB. It used the same notation used to identify features in SMS ***(XXxnX)***. The differences are in the ***x***, where *g* has been added for *Progress* and *w* for *Software*. Also the ***X*** that has now assumed the *S* to represent the origin of the information as a Survey.

For SB in the Survey, the following questions were investigated: **RQ6** - Who did select the items for the SB?; **RQ7** - In what states were the tasks classified?; **RQ8** -How was the progress of tasks in the SB monitored?; and **RQ9** - What softwares were used to manage the SB?.

Table 3. Survey information for PB and SB.

RQ	Information	Feature	RQ	Information	Feature	RQ	Information	Feature
Product Backlog			**Sprint Backlog**			**Increment**		
RQ1	Representation: User History	PBr1S	RQ6	Selection: PO	SBs1S	RQ10	Test: Test Driven Development (TDD)	INCe2S
RQ1	Representation: Use Cases	PBr2S	RQ6	Selection: PO and Dev.Team	SBs2S	RQ10	Test:Integration Test	INCe1S
RQ2	Prioritize: PO and Dev. Team	PBp1S	RQ6	Selection: Dev.Team	SBs3S	RQ10	Test:Ad-hoc (defined in Ticket Jira)	INCe3S
RQ2	Prioritize:PO alone	PBp2S	RQ6	Selection: SM	SBs4S	RQ10	Test:Unit Testing	INCe4S
RQ2	Prioritize: PO and SM	PBp3S	RQ7	Task - Status: Concluded, In progress, Blocked, New, Open, Classifying, Draft, Verified, To do, To test Testing	 SBt1S SBt2S SBt3S SBt4S SBt5S SBt6S SBt7S SBt8S SBt9S SBt10S SBt11S	RQ10	Test:User Note	INCe5S
RQ3	Estimate:Planning Poker	PBe1S	RQ8	Task - Progress:Monitoring task board	SBg1S	RQ10	Test:Manual with some automation	INCe6S
RQ3	Estimate:Story Points and Value Points	PBe2S	RQ8	Task - Progress:Burndown Sprint Chart	SBg2S	RQ10	Test:Team Demonstration	INCe7S
RQ3	Estimate:Story Points	PBe3S	RQ8	Task - Progress:Weekly report	SBg3S	RQ10	Test:User Acceptance Test	INCe8S
RQ3	Estimate: Story Points and Value Points with Planning Poker	PBe4S	RQ8	Task - Progress: Board Jira	SBg4S	RQ11	Version:Used field in Jira and Jira Releases	INCv1S
RQ4	Progress: Performed in Sprint Planning or pre-sizing	PBg1S	RQ8	Task - Progress: Visual Board	SBg5S	RQ11	Version:ServiceNow Update Sets	INCv2S
RQ4	Progress: Risk and Impact Techniques	PBg2S	RQ9	Software:ServiceNow	SBw1S	RQ11	Version:From Visual Studio	INCv3S
RQ4	Progress: Burndown Release Chart	PBg3S	RQ9	Software:Readmine	SBw2S	RQ11	Version:Crashlytics	INCv4S
RQ5	Software: Azure Devops	PBw1S	RQ9	Software:Jira	SBw3S	RQ11	Version:Version Control by itself XX.YY.ZZZ	INCv5S
RQ5	Software: TFS	PBw2S	RQ9	Software: Azure Devops	SBw4S	RQ11	Version:GitHub	INCv6S
RQ5	Software: Trello	PBw3S	RQ9	Software: RTC	SBw5S	RQ12	Quality:Sprint Review	INCq1S
RQ5	Software: Jira	PBw4S	RQ9	Software: Microsoft Team Foundation Server	SBw6S	RQ12	Quality:Evaluated by the PO	INCq2S
RQ5	Software: ServiceNow	PBw5S	RQ9	Software: Version One	SBw7S	RQ12	Quality:Acceptance test by PO	INCq3S
RQ5	Software: Product Roadmap	PBw6S				RQ12	Quality:By the QC testing team	INCq4S
RQ5	Software: RTC	PBw7S				RQ12	Quality:Functional and unit tests	INCq5S
RQ5	Software: Microsoft Foundation Server	PBw8S				RQ12	Quality:Search with users	INCq6S
RQ5	Software: MS	PBw9S				RQ12	Quality:QA tested and verified problems	INCq7S
RQ5	Software: Version One	PBw10S				RQ12	Quality:By defect density	INCq8S
RQ5	Software: Rally	PBw11S				RQ12	Quality:Definition of Done, static code, others	INCq9S

Regarding the Increment (INC), the following questions were investigated: **RQ10** - What testing techniques are used for the INC?; **RQ11** - Which Version control is used for the INC?; and **RQ12** - How was the quality of the INC assessed?

Table 3 systematizes the answers found for the questions investigated for the INC. The notation used to represent the features for the INC follows the same idea as the one used for PB and SB, with some differences, which are: ***XXX*** to represent the INC artifact, ***x*** that assumes (e -test, v-version and q -quality).

Based on Table 3, the characteristics model was elaborated with the information from the Survey for Scrum artifacts. The model is shown in Figure 3.

3.3. Threats to Validity

Our main threats to this study are: small samples in the SMS and the survey, which might jeopardizes statistical significance; lack of quality evaluation of the primary studies of the SMS; and the interpretation of the adaptations might be biased in a certain way due to the expected results from the researchers point of view.

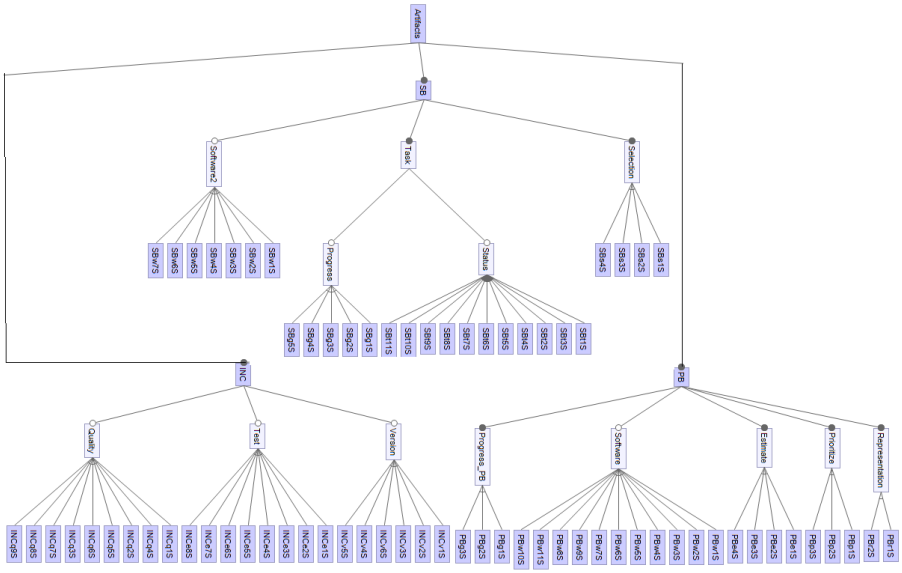


Figure 3. Feature Model for Scrum Artifacts based on the Survey

4. Feature Unification Process

In this section we describe the process of unifying the models of features obtained for the SMS and the Survey.

4.1. Scrum Guide Compliance Check

Product Backlog (PB). The information regarding the PB in the SMS and that appears in Table 2 was compared with the Scrum guide and no inconsistencies were found in them. Regarding the PB data in the Survey, which are shown in Table 3, they were compared with the Scrum guide and there were also no inconsistencies.

Sprint Backlog (PB). Regarding the SMS, the information found for the SB and shown in Table 2, there was no inconsistency with the Scrum guide. For the SB information in the Survey and shown in Table 3, we found inconsistencies in the following features for the *Selection* of items for SB , in relation to the Scrum guide:*SBs1S*, *SBs2S* and *SBs4S*. The Scrum guide states that it is the responsibility of the Dev. Team to select the items to compose the SB. Soon, inconsistent features will be eliminated for the unified feature model.

Increment (INC). In the SMS there was no return of information for the Increment so there are no inconsistencies regarding the Scrum guide to be verified. Regarding the Survey, although more information was returned to the INC (see Table 3), no inconsistencies were found in relation to the Scrum guide.

4.2. Elimination of Redundant Features

We eliminated the features common in the SMS and in the Survey for artifacts. We wanted to maintain the features found in the Survey, for cases of repeated/similar features, with no prejudice to information.

Product Backlog (PB). Observing Tables 2 and 3, the following features were found to be the same for PB: *PBr1M = PBr1S, PBr6M = PBr2S, PBp1M = PBp1S,*

PBp5M = PBp2S, PBe1M = PBe1S and *PBe7M = PBe2S*.Therefore, in the unified features model, only the Survey's original features will be present, which are: *PBr1S, PBr2S, PBp1S, PBp2S, PBe1S* and *PBe2S*.

Sprint Backlog (SB). Looking at Tables 2 and 3, it was found that the following features for the SB are the same: *SBs1M=SBs3S, SBt2M=SBt2S, SBt3M=SBt1S* and *SBt4M=SBt3S*. The features that will be part of the unified model will be those that have the origin of the Survey, that is, those that end with the letter *S*.

Increment (INC). With regard to the Increment artifact, repeated features were not identified since the SMS did not return information for these artifacts, so those that were returned by the Survey were assumed.

After the consistency of the information from the SMS and the Survey was made with the Scrum guide and verification of the redundant features in both models, in this section we show the resulting model with the remaining features. This can be seen in Figure 4.

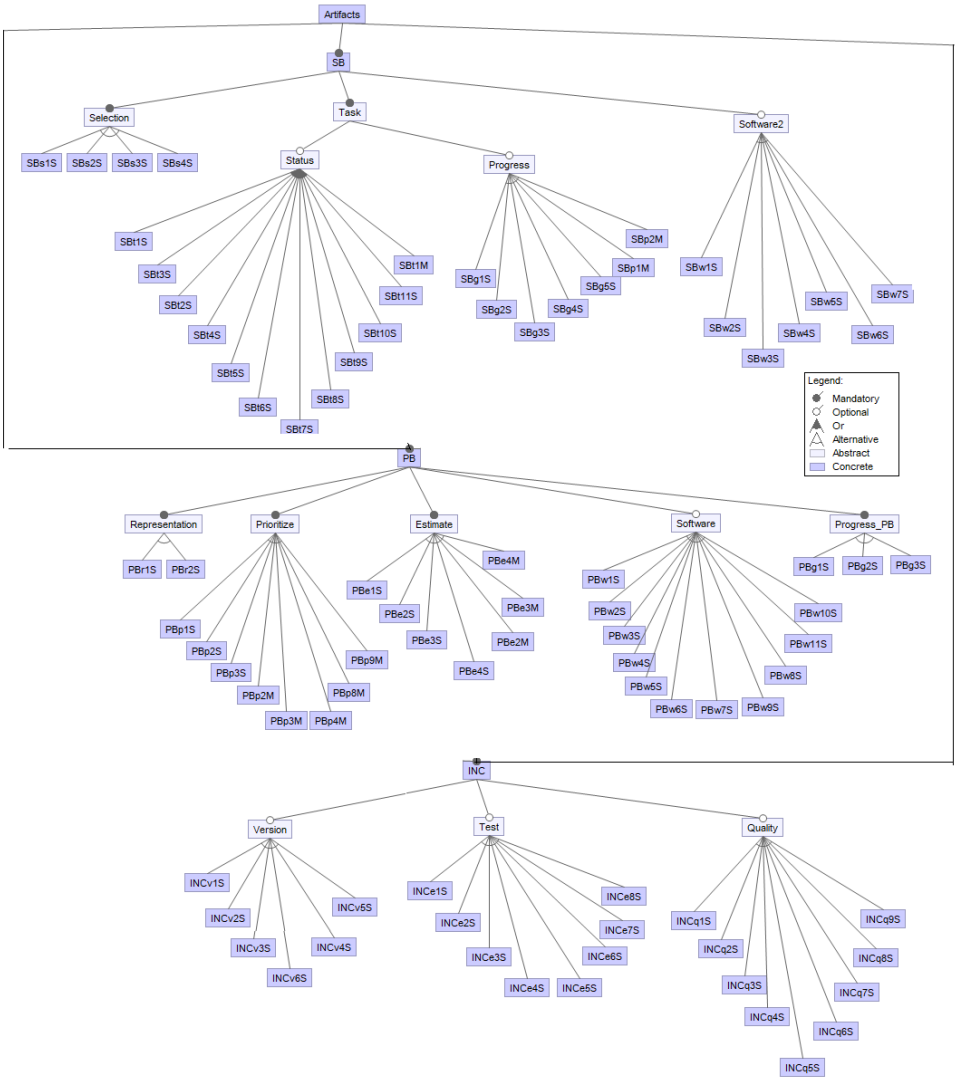


Figure 4. A Feature Model for Scrum Artifacts Adaptations

5. Conclusion

In the information sets we analyzed for the SMS and the Survey, we were able to build models of feature that would meet the recommendations of the Scrum guide in relation to its artifacts. We observed that the analyzed literature did not bring significant information about the Increment artifact, which can be demonstrated in a certain way were a shortage on the subject. But practitioners, when provoked through the Survey, provided interesting information about this artifact such as: tools used to manage PB and SB, version control for Increment, etc.

Through the feature models elaborated in this work, we store and systematize the knowledge about the adaptations of the Scrum artifacts found. Feature models facilitate the identification of relationships between features and also show which are mandatory in the use of artifacts. We do not intend in this work that the unified feature models for Scrum artifacts provide all possible solutions for the use of this component, through its derivation, but that it allows users of the method to have it as a starting point to create their own model or who knows how to improve it. Although we checked the conformity of the features found with the Scrum guide, we did not carry out a practical validation of the proposed models, which appears as an opportunity for future work and which we believe to be important.

References

Czarnecki, K. and Eisenecker, U. W. (1999). Components and generative programming. In *ACM SIGSOFT Software Engineering Notes*, volume 24, pages 2–19. Springer-Verlag.

Czarnecki, K., Helsen, S., and Eisenecker, U. (2005). Formalizing cardinality-based feature models and their specialization. *Software process: Improvement and practice*, 10(1):7–29.

Diebold, P., Ostberg, J.-P., Wagner, S., and Zendler, U. (2015). What do practitioners vary in using scrum? In Lassenius, C., Dingsøyr, T., and Paasivaara, M., editors, *Agile Processes in Software Engineering and Extreme Programming*, pages 40–51. Springer International Publishing.

Garcia, L. (2019). Adaptations of the scrum framework for software development projects. Master's thesis, Universidade Estadual de Maringá.

Kang, K. C., Cohen, S. G., Hess, J. A., Novak, W. E., and Peterson, A. S. (1990). Feature-oriented domain analysis (foda) feasibility study. Technical report, Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst.

Petersen, K., Vakkalanka, S., and Kuzniarz, L. (2015). Guidelines for conducting systematic mapping studies in software engineering: An update. *Information and Software Technology*, 64:1–18.

Schwaber, K. and Sutherland, J. (2017). *The Definitive Guide to SCRUM: The rules of the Game*. Scrum.org. https://www.mitchlacey.com/resources/official-scrum-guide-current-and-past-versions.

Sochos, P., Philippow, I., and Riebisch, M. (2004). Feature-oriented development of software product lines: mapping feature models to the architecture. In *Net. ObjectDays*, pages 138–152. Springer.

Avaliação Preliminar de uma Linguagem para a Representação Textual de Modelos Conceituais de Bancos de Dados Relacionais

Jonnathan Riquelmo[1], **Maicon Bernardino**[1],
Fábio Paulo Basso[1], **Elder Macedo Rodrigues**[1]

[1]Laboratory of Empirical Studies in Software Engineering (LESSE),
Engenharia de Software, Universidade Federal do Pampa (UNIPAMPA)
Av. Tiarajú, 810 - Bairro Ibirapuitã - Alegrete, RS

{jonnathan.riquelmo, fabiopbasso, eldermr}@gmail.com, bernardino@acm.org

Abstract. *The selection of a conceptual database modeling approach (DBs), among other things, depends on the domain of the problem, knowledge and preference of the developer. To evaluate a textual language with such scope, this article reports an evaluation study of two initial grammars using the focus group technique. This preliminary assessment was carried out with thirteen (13) participants, providing important feedbacks for the evolution of grammar. Subsequently, information about the developed Eclipse plugin is presented, including modeling and model transformation resources.*

Resumo. *A seleção de uma abordagem de modelagem conceitual de bancos de dados (BDs), entre outras coisas, depende do domínio do problema, do conhecimento e da preferência do desenvolvedor. Para avaliar uma linguagem textual com tal escopo, este artigo relata um estudo de avaliação de duas gramáticas iniciais usando a técnica de grupo focal. Essa avaliação preliminar foi realizada com treze (13) participantes, fornecendo feedbacks importantes para a evolução da gramática. Posteriormente, são apresentadas informações sobre o plugin Eclipse desenvolvido, incluindo recursos de modelagem e transformação de modelo.*

1. Introdução

Nas últimas décadas surgiram várias abordagens de desenvolvimento de Banco de Dados (BD) para sistemas de informação. Alguns dos métodos em uso, entre eles o de modelo relacional, tem como base a proposta de modelagem conceitual de dados com abordagem gráfica concebida por [Chen 1976]. Essa é uma das abordagens mais bem aceitas pela comunidade de desenvolvedores ainda nos dias de hoje, chamada de Modelo Entidade-Relacionamento (ER).

Na modelagem ER a construção de um BD é estruturada em um modelo de BD, o qual é uma descrição detalhada dos tipos de informações que devem ser armazenadas. O projeto de BD acontece em três fases distintas de modelagem, em que são gerados o modelo conceitual, lógico e o físico [Heuser 2009]. A modelagem ER tem foco no modelo conceitual, cujo os conceitos principais são o de entidades, que são uma representação de um conjunto de objetos do domínio modelado, e suas relações.

Posto isso, esse estudo apresenta a continuação da pesquisa evidenciada no trabalho de [Riquelmo et al. 2019], o qual propôs a definição de uma linguagem específica de domínio (*Domain Specific Language* - DSL), com abordagem textual, para a representação de modelos conceituais de BDs relacionais, uma vez que a grande maioria das soluções existentes tem abordagem puramente gráfica. Partindo da premissa de que desenvolvedores e projetistas de software possuem diferentes preferências por abordagens representacionais de modelos de BDs, este estudo contribui para a concepção de uma nova DSL mapeada para necessidades de ensino, para alunos com maior afinidade com linguagens de programação num perfil de desenvolvedor.

Para [van Deursen et al. 2000] e [Fowler 2010] uma DSL é uma linguagem de programação ou linguagem de especificação executável que oferece, por meio de notações e abstrações apropriadas, poder expressivo focado e, geralmente, restrito a um domínio de problema específico. Assim como outras linguagens, as DSLs devem apresentar um conjunto de sentenças bem definidas por uma sintaxe e semântica própria. Entre exemplos conhecidos de DSLs estão: **(i)** SQL, para bancos de dados; **(ii)** CSS, para *layout* de páginas *Web*; **(iii)** XML, para codificação de dados; **(iv)** UML, para projeto de software; e **(v)** LaTeX, para tipografia de documentos.

Este estudo é o relato de uma avaliação preliminar com a utilização de um grupo focal para as duas variações da gramática proposta, exibidas na Figura 1, bem como exposição da estrutura e operação da ferramenta implementada posteriormente.

Figura 1. Exemplos de Uso das duas Variações da Gramática da DSL.

```
Domain University;
Entities{
        Person{
                PID: int isIdentifier,
                Name: string
        }
        Teacher isA Person{
                Phone: int,
                Salary: money
        }
        Student isA Person{
                Course: string
        }
        OutsrcEmployee isA Person{
            OutsourcedEID: int,
            Company: string
        }
        Class{
                ClassID: int isIdentifier,
                Course: string,
                Semester: string
        }
        ClassRoom {
                ClassRoomID: int isIdentifier,
                Capacity: int
        }
};
Relationships{
        [many Student isRelatedWith many Class]
        TeacherClass [many Teacher isRelatedWith many Class]
        ClassSchedule [many TeacherClass isRelatedWith many ClassRoom]
                {ClassScheduleID: int, DayOfWeek: datetime, Discipline: string}
        Supervisor [zero one OutsrcEmployee isRelatedWith many OutsrcEmployee]
};
```

```
Domain University
Entities{
        Person{
                * PID int,
                Name string
        }
        Teacher is Person{
                Phone int,
                Salary money
        }
        Student is Person{
                Course string
        }
        OutsrcEmployee is Person{
            OutsourcedEID int,
            Company string
        }
        Class{
                * ClassID int,
                Course string,
                Semester string
        }
        ClassRoom {
                * ClassRoomID int,
                Capacity int
        }
}
Relationships{
        [(1,N) Student relates (1,N) Class]
        TeacherClass [(1,N) Teacher relates (1,N) Class]
        ClassSchedule [(1,N) TeacherClass relates (1,N) ClassRoom]
                {ClassScheduleID int, DayOfWeek datetime, Discipline string}
        Supervisor [(0,1) OutsrcEmployee relates (1,N) OutsrcEmployee]
}
```

Este artigo está organizado da seguinte maneira. A Seção 3 apresenta o processo de planejamento, preparação e execução do grupo focal, bem como a análise dos resultados e suas ameaças à validade. A seguir, ocorre a demonstração da ferramenta construída a partir da proposta de DSL, na Seção 4. Finalmente, as considerações preliminares são discutidas na Seção 5, em que ainda são indicados os planos de trabalhos futuros.

2. Trabalhos Relacionados

Anteriormente, após um extenso estudo composto por um Mapeamento Multivocal de Literatura (MLM, do inglês *Multivocal Literature Mapping*)[1], foram selecionadas ferramen-

[1]DOI: https://doi.org/10.5281/zenodo.3950586

tas que mais se aproximam da DSL proposta, totalizando quatro (4) soluções. Contudo, este estudo tem também como cerne uma avaliação utilizando um grupo focal. Sendo assim, realizou-se uma investigação *ad-hoc* buscando estudos que relatassem avaliações com grupos focais no contexto de DSLs.

O trabalho de Poltronieri [Poltronieri et al. 2018] apresenta um *framework* de avaliação de usabilidade para DSLs. Este estudo relata um grupo focal para avaliação da proposta feita. Mesmo não sendo a avaliação de uma DSL propriamente dita, mas sim uma abordagem para avaliação de DSLs, o uso da mesma metodologia de avaliação o torna um trabalho interessante para análise. Esta avaliação contou com sete (7) participantes, entre eles especialistas em Engenharia de Software, Interação Humano-Computador e Teste de Desempenho.

Houve ainda previamente um teste piloto com apenas dois participantes para verificar a adequação do protocolo criado. Todo o processo é descrito dividido nas fases de planejamento, preparação, moderação e análise de dados. O resultado final confirmou que o *framework* proposto ajuda na avaliação de usabilidade de DSLs.

3. Grupo Focal

Esta seção descreve a avaliação preliminar conduzida para analisar as duas alternativas de DSLs propostas, visando assim o seu aperfeiçoamento em uma versão final que seja aderente ao processo ensino-aprendizagem no ambiente acadêmico. Para tanto, foi estabelecido o uso de um grupo focal, um método de pesquisa qualitativa para gerar *feedback* de um conjunto de pessoas em relação a um tema específico. Essa abordagem é muito utilizada como uma atividade para pesquisa de mercado em diversas áreas, uma vez que pode cumprir papel importante apoiando pesquisas exploratórias.

O processo executado nessa etapa, expressado na Figura 2, teve como base as diretrizes estabelecidas no trabalho de [Kontio et al. 2008], as quais cobrem a aplicação desse método no contexto da Engenharia de Software.

Figura 2. Processo do Grupo Focal.

3.1. Planejamento

Durante o planejamento foi definido um protocolo que deveria ser seguido. Nesse protocolo, motivado pelo problema que era gerar uma versão definitiva da gramática da DSL proposta, foram criados os documentos necessários para sua execução: **(i)** Termo de Consentimento Livre e Esclarecido (TCLE); **(ii)** Glossário de Termos; **(iii)** Questionário de Perfil; **(iv)** Instrumentos da Discussão 1, 2 e 3; **(v)** Modelos das Gramáticas Avaliadas; **(vi)** Roteiro de Apresentação. Todos os modelos dos documentos produzidos estão disponíveis em um repositório[2] público.

[2]https://github.com/JonnathanRiquelmo/Focus-Group-Protocol

3.2. Preparação

Tipicamente, as avaliações que utilizam grupos focais devem ser constituídas de quatro (4) a seis (6) grupos focais individuais para que o rigor científico seja considerado verdadeiramente alto. O tamanho de cada grupo focal pode variar de três (3) a até doze (12) elementos, sendo mais comum esse número ficar entre quatro (4) e oito (8) participantes [Kontio et al. 2008].

Por questões de viabilidade de tempo e recursos humanos, para o presente estudo foi possível ser executado um (1) grupo focal. Após a realização do convite, treze (13) participantes colaboraram, todos da área da Engenharia de Software. Desse total, três (3) participantes eram alunos de graduação, nove (9) eram mestrandos e um (1) doutorando.

Foi então aplicado o Questionário de Perfil, em que foi possível identificar que havia um nível equilibrado de conhecimento entre os participantes. Isso foi constatado pois todos já possuíam contato com DSL, tendo utilizado esse tipo de linguagem, ao menos, durante a graduação. Ainda foi informado que todo o processo seria gravado em áudio, fato com o qual todos concordaram.

3.3. Execução

O grupo focal foi realizado no segundo semestre de 2019, nas dependências de uma Instituição de Ensino Superior (IES), e teve duração duas (2) horas. Iniciando com a exposição do roteiro que deveria ser seguido, onde houve a apresentação do objetivo do grupo focal e os conceitos básicos envolvidos, aconteceu o pedido para que os participantes realizassem a leitura e assinatura do TCLE. Com o documento preenchido, foi dado continuidade ao roteiro previsto,

A cada Instrumento de Discussão disponibilizado esperou-se até que os participantes respondessem de forma individual. Após, aconteceu uma discussão em grupo sobre o tema levantado. Todo o processo foi gravado em áudio e teve o suporte dos três (3) pesquisadores envolvidos neste estudo. Foi realizada também a transcrição das observações levantadas pelo debate que se seguiu, caracterizando assim as práticas de *brainstorming*[3] previstas em grupos focais.

3.4. Análise dos Resultados

Segundo [Kontio et al. 2008], a fase de análise e interpretação dos dados gerados constitui parte importante da pesquisa qualitativa, considerando o contexto, o comportamento e a percepção dos sujeitos. Para a fase de análise dos dados, o áudio foi analisado paralelamente as anotações realizadas. De posse desses materiais e das respostas dos participantes para cada instrumento de discussão, foi possível avaliar os resultados do grupo focal executado.

Após a apresentação do roteiro preparado para o grupo focal, deu-se início as discussões dos três (3) instrumentos criados para a dinâmica. O primeiro instrumento continha a seguinte afirmação associada a uma escala Likert composta de níveis de concordância dispostos de um (1) a cinco (5), sendo um (1) indica discordância total e cinco (5) concordância total:

[3] *Brainstorming* é uma técnica utilizada para propor soluções a um problema específico. Consiste em uma reunião, também chamada de tempestade de ideias, na qual os participantes devem ter liberdade de expor suas sugestões e debater sobre as contribuições dos colegas.

"Linguagens específicas de domínio com abordagem textual podem ser aplicadas na modelagem conceitual posto que conseguem descrever de forma rápida e concisa determinadas propriedades. Sendo assim, essas soluções podem ser utilizadas ou mesmo adaptadas de uma forma efetiva no que diz respeito a representação do domínio que modelam."

Após todos os participantes responderem o instrumento, foi aberto um momento de discussão entre todos. Por não terem visto o modelo proposto de DSL, algumas dúvidas surgiram e os pesquisadores envolvidos procuraram sanar todas de forma a não influenciar as discussões seguintes.

O debate prosseguiu com os participantes levantando possíveis vantagens de um modelo textual. Alguns citaram acreditar que essa abordagem poderia ser de mais fácil entendimento, porém que isso dependeria do usuário. Esta suposição incluiu dois prováveis tipos de perfis: analistas e desenvolvedores. O grupo chegou a conclusão que a abordagem poderia ser vista como mais produtiva por usuários de perfil desenvolvedor, mas menos proveitosa por aqueles que tivessem um viés mais analista em razão do seu nível de abstração em relação às abordagens gráficas. A Figura 3a exibe a distribuição das respostas dos participantes para a primeira discussão. Após, passou-se para a execução da discussão do segundo instrumento. A atividade era composta da seguinte pergunta:

"Considerando que um modelo conceitual de banco de dados deve definir ao menos as entidades de domínio, seus atributos e o número de ocorrências (cardinalidade) possíveis de associações (relacionamento), como você definiria uma gramática básica (DSL) para a sua representação?"

Foi informado que os participantes poderiam conversar livremente durante toda a realização deste instrumento. Após cada um sugerir a sua sintaxe, houve debate e troca de informações sobre como estruturar melhor as informações. Neste momento foi possível avaliar que o perfil de alguns acabaria por influenciar no terceiro elemento, pois alguns fizeram sugestões para tornar a linguagem mais sucinta, enquanto outros defenderam que a gramática poderia utilizar mais elementos. A discussão que se seguiu foi focada principalmente em como representar as relações do modelo ER em uma sintaxe textual.

A maior dificuldade se mostrou em como definir uma ordem. Outro ponto que merece destaque foi quanto à cardinalidade, onde no geral seguiu-se a nomenclatura utilizada no diagrama original de Chen (*e.g.* 0,1). Entretanto, ao fim do instrumento houveram opiniões muito divergentes em relação à representação ideal já que alguns participantes acabaram optando por incluir as relações dentro das entidades e outros fora.

Aconteceram também sugestões quanto às palavras-chave possíveis, como `Element`, `ElementFather`, `ElementSource`, `Type`, e `Referential`. Ainda, seis (6) participantes sugeriram o uso de ponto e vírgula (;) para separação das declarações de elementos, e todos os treze (13) preconizaram a utilização de símbolos como parênteses, colchetes e/ou chaves para agrupar conjuntos similares de elementos.

Finalizada a dinâmica proposta, chegou-se ao último instrumento do grupo focal. Este artefato era composto de um exemplo de cada versão da gramática produzida preliminarmente para este trabalho [**?**]. De posse das versões, foi pedido que os participantes realizassem a escolha entre as opções, apontando assim qual avaliavam como mais viável para modelagem ER. Também foi solicitado que fossem indicados os pontos positivos e

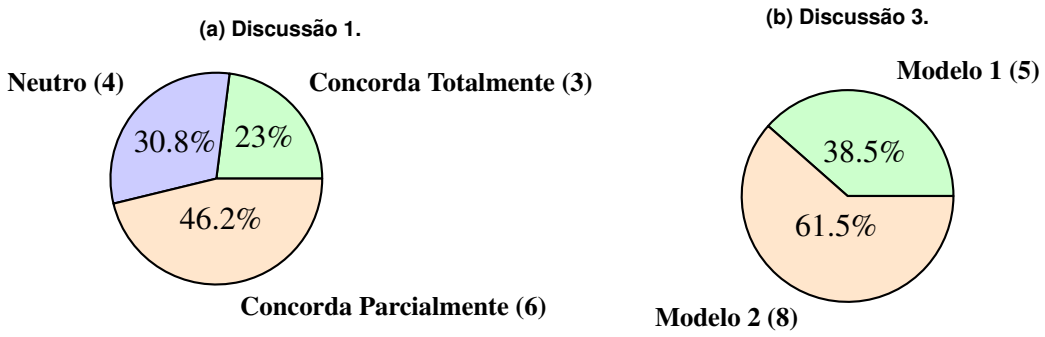


Figura 3. Resultados das Discussões - Grupo Focal.

negativos observados em cada modelo.

O segundo modelo acabou sendo escolhido pela maioria, como pode ser observado na Figura 3b. Porém, ao final das discussões obteve-se um consenso de que a forma de definição de entidades do primeiro modelo e a disposição dos relacionamentos do segundo, em especial as cardinalidades, eram os mais adequados para a aplicação no ensino, indicando assim a necessidade de uma fusão de ambas as versões.

3.5. Ameaças à Validade

Para a listagem das possíveis ameaças tiveram como base os tópicos levantados no trabalho de Wohlin [Wohlin et al. 2012].

Validade do Construto - (i) ***Explicação pré-operacional inadequada***: Esta ameaça está relacionada com o fato do grupo focal não ter o objetivo dos artefatos suficientemente definidos antes de serem traduzidos em medidas ou tratamentos. (ii) ***Interação de diferentes tratamentos***: Se os sujeitos estiverem envolvidos em mais de um estudo, os tratamentos dos diferentes estudos poderão interagir e reverberar nos resultados finais. Todos os sujeitos fizeram apenas esse grupo focal na época de sua realização.

Validade Interna - (i) ***História***: Há o risco de algum período temporal específico ter influência na realização do experimento. Para mitigar esta ameaça, e em razão do grupo focal ser realizado em ambiente acadêmico, todo o processo foi executado mediante aviso dos participantes quanto a um período em que todos não estivessem necessariamente sobrecarregados com atividades acadêmicas *e.g.* provas e trabalhos. (ii) ***Testes***: Se os testes forem repetidos, os sujeitos podem responder de maneira diferente em momentos diferentes, pois sabem como o teste é realizado. Se houver necessidade de familiarização com os testes, é importante que os resultados do teste não sejam devolvidos ao sujeito, para assim não apoiar o aprendizado não intencional. Não houve necessidade de repetição das atividades, uma vez que as mesmas foram executadas uma vez por cada participante. (iii) ***Instrumentação***: Essa ameaça está relacionada aos artefatos usados para a execução da experiência, como formulários de coleta de dados, etc. Se estes foram mal projetados, a experiência é afetada negativamente. Para combater essa ameaça, todos os artefatos foram verificados e validados previamente em reuniões entre os pesquisadores envolvidos neste trabalho.

Validade Externa - (i) ***Sujeitos***: Os sujeitos selecionados para o grupo focal

podem não representar um grupo significativo para a área de estudo. Buscando tentar mitigar essa ameaça foi realizado com participantes da área de Engenharia de Software e Ciência da Computação, e logo, inseridos no contexto de uso da modelagem conceitual de BD relacionais. Porém, o fato da amostra ser de apenas um grupo focal é uma ameaça indicada na literatura. Não foi possível mitigar este fato. (ii) ***Interação dos sujeitos com os artefatos de avaliação***: Essa é a ameaça relacionada a aplicação dos artefatos de avaliação do grupo focal com os sujeitos. Dependendo do momento isto pode afetar os resultados. Se, por exemplo, um questionário for realizado alguns dias após a execução do grupo focal, as pessoas tendem a responder de maneira diferente do que fariam momentos após as atividades. Todos os instrumentos foram realizados na mesma sessão.

Validade da Conclusão - (i) ***Confiabilidade dos resultados***: A confiabilidade dos resultados obtidos tem impacto direto na validade do grupo focal como um todo. Por ser uma avaliação com maior foco qualitativo, esta ameaça não pôde ser mitigada em razão da subjetividade inerente às resposta dos sujeitos em tais avaliações.

4. A Ferramenta

Esta seção apresenta a ferramenta Eclipse com o *plugin* final da solução proposta. Acabou se optando inicialmente por um *plugin* por conta de sua integração com a plataforma Eclipse, bem como pela vantagem de poder ser insumo para um aplicativo independente (*standalone application*).

A principal diferença é que, quando usado como um *plugin* do Eclipse, o editor pode fornecer, além das funcionalidades da gramática, suporte a outras linguagens *e.g.* Java, PHP. Um produto independente, por outro lado, fornece toda a infraestrutura voltada unicamente a linguagem desenvolvida. Em ambos os casos é possível que haja a distribuição como uma ferramenta livre desde que seguidas as diretrizes da licença de software EPL-2.0[4].

A arquitetura da solução é exposta na Figura 4 e foi construída com o auxílio do Xtext, um *framework* de código aberto para o desenvolvimento linguagens de programação textuais. Nele, a partir da gramática resultante do grupo focal conduzido, gerou-se a maior parte da infraestrutura do editor, do *parser* e modelo Ecore. Para entendimento, o Ecore é uma representação na memória em tempo de execução do modelo criado utilizando a DSL. Após, ocorreram então alguns ajustes manuais necessários e o acréscimo da escrita de um gerador para a conversão dos modelos conceituais para os modelos lógicos.

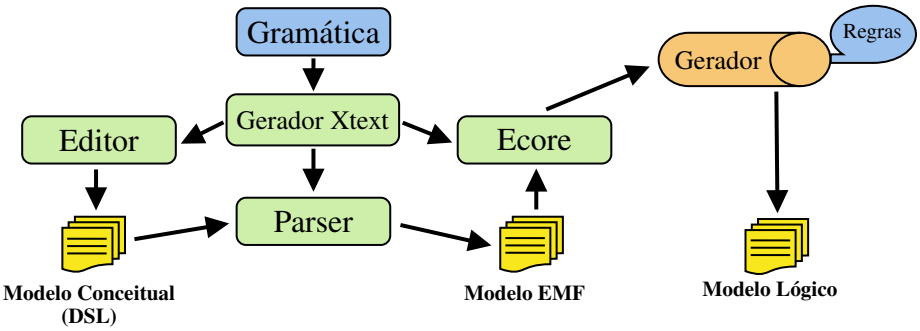
4.1. Operação

Como dito anteriormente, a versão final da gramática resultou da fusão entre as duas versões previamente definidas e avaliadas com a execução do grupo focal. As principais mudanças foram as modificações das palavras reservadas `isA` e `isRelatedWith` por `is` e `relates`, respectivamente, além da adoção da convenção `(0:1) (1:1) (0:N) (1:N)` para as cardinalidades.

A Figura 5 apresenta a ferramenta em funcionamento, após o *plugin* ser integrado no Eclipse, onde é possível ser feita a criação de arquivos de modelagem utilizando a

[4]https://www.eclipse.org/legal/epl-2.0/

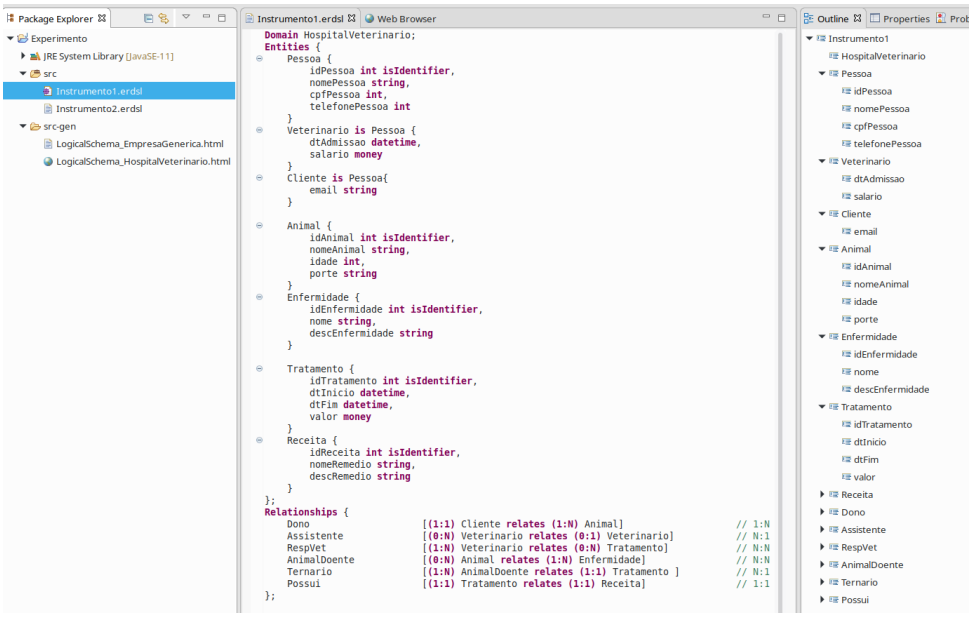
Figura 4. Arquitetura geral da ferramenta.



DSL. Nesse exemplo há a modelagem de sete (7) entidades e seis (6) relacionamentos, incluindo um autorelacionamento e um relacionamento ternário.

A modelagem na ferramenta ganha validação em tempo real com base nos dois blocos previstos na gramática (entidades e relacionamentos), *syntax highlighting*, que indica erros de sintaxe em tempo de escrita, autocomplemento de código e *hovering*, uma funcionalidade que exibe informações sobre um item quando o cursor do *mouse* é colocado sobre ele. Sabe-se que a definição dos tipos de dados não é prevista no modelo conceitual clássico mas, por questões relacionadas a pretensão futura de realizar a geração de instruções SQL, foi decidido pela manutenção dessa escolha de projeto.

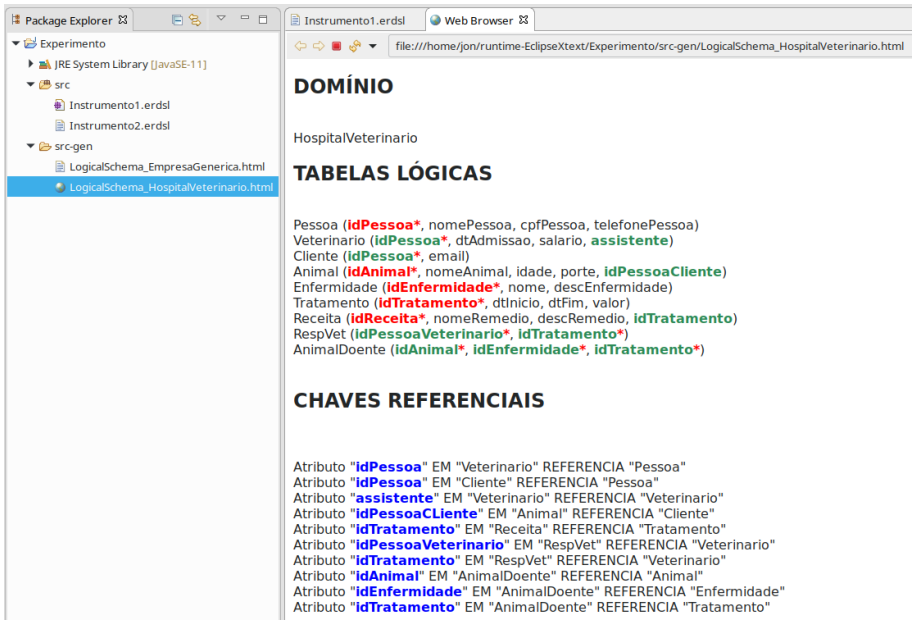
Figura 5. Fragmento da solução sendo utilizada.



A função para o mapeamento e geração do modelo lógico é executada automaticamente toda a vez que uma modelo é salvo. Na Figura 5 pode-se observar os arquivos *.html* na estrutura de diretórios na guia da esquerda, dentro da pasta *src-gen*. Foram gerados arquivos com esse formato para uma melhor visualização a partir de marcações no texto. Estas marcações podem ser renderizadas por qualquer navegador de *Internet*, ou mesmo dentro do próprio ambiente, aumentando o poder de compreensão por parte do usuário.

O modelo lógico derivado pelo gerador que mapeia o modelo conceitual é apresentado na Figura 6.

Figura 6. Exemplo de modelo lógico gerado.



Mediante as premissas assumidas para a realização da transformação, o modelo original resulta em um novo modelo composto de nove (9) entidades, e também já possuindo suas integridade referenciais inferidas, ou seja, os registros que apontam para outros registros já são estabelecidos.

É importante salientar que as premissas foram escolhidas previamente com base no livro referência de [Heuser 2009] para o processo de mapeamento e transformação de um modelo conceitual em uma representação lógica do banco de dados. As premissas implementadas até agora podem ser resumidas em: (i) adição de coluna para relacionamentos 1:1; (ii) adição de coluna para relacionamentos 1:N, e; (iii) criação de tabela própria para relacionamentos N:N. Em relação ao conceito de generalização/especialização, também foi necessário assumir uma ideia inicial que teria que ser tomada como verdade.

Segundo [Heuser 2009], para esses casos existem duas alternativas passíveis de serem derivadas. A primeira recomenda o uso de uma única tabela para toda a hierarquia de entidades, ou seja, recomenda a fusão das tabelas. A segunda recomenda o uso de uma tabela por entidade modelada, desde que respeitado a integridade referencial, ou seja, as chaves primárias das entidades filhas devem apontar necessariamente para a chave primária da entidade pai. No caso da ferramenta resultante neste trabalho, optou-se pela segunda alternativa. O gerador do modelo lógico foi desenvolvido com a linguagem de propósito geral (*General-purpose Language* - GPL) Xtend, uma linguagem que tem suas raízes no Java.

5. Considerações Preliminares

Com a alegação anterior em que afirmou-se que profissionais que trabalham com software possuem diferentes predileções nas abordagens para representação de BDs, este estudo

colaborou para a formulação de uma DSL que possa atender atividades de ensino, visando estudantes com perfis direcionados para as atividades de desenvolvimento.

Os resultados preliminares apontaram para uma aceitação da proposta, sendo que, a partir de dois modelos de gramática analisados, houveram recomendações para a adoção de expressões mais concisas no bloco de entidades e relações, assim como de convenções no que diz respeito as cardinalidades. Dessarte, a construção de um *plugin* capaz de realizar a modelagem e transformação entre modelos conceituais e lógicos foi alcançado e então disponibilizado como software de código aberto [5].

Este trabalho tem por objetivo contribuir com uma ferramenta que auxilie no processo de projeto e modelagem de BDs utilizando uma DSL textual de fácil uso e compreensão. Pretende-se que a proposta final não apenas realize modelagem, mas sim a transformação dos modelos conceituais gerados em *scripts* SQL para diferentes tecnologias SGBDs, *e.g.* PostgreSQL, MySQL e SQL Server.

Já estão sendo desenvolvidos os geradores para *scripts* SQL. Pretende-se ainda que seja realizada uma avaliação empírica do produto desenvolvido. Esse experimento, por exemplo, poderá servir para a avaliação da abordagem textual e sua viabilidade frente a abordagem gráfica. Outra possibilidade de evolução futura é a criação de uma versão gráfica desta DSL e sua integração com a atual, tornando a solução final uma ferramenta de abordagem bidirecional (textual e gráfica).

Referências

Chen, P. P.-S. (1976). The entity-relationship model—toward a unified view of data. *ACM Trans. Database Syst.*, 1(1):9–36.

Fowler, M. (2010). *Domain Specific Languages*. Addison-Wesley Professional, London,England, 1st edition.

Heuser, C. A. (2009). *Projeto de Banco de Dados*. Bookman, Porto Alegre, BR.

Kontio, J., Bragge, J., and Lehtola, L. (2008). *The Focus Group Method as an Empirical Tool in Software Engineering*, pages 93–116. Springer London, London.

Poltronieri, I., Zorzo, A. F., Bernardino, M., and de Borba Campos, M. (2018). Usa-dsl: usability evaluation framework for domain-specific languages. In *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, pages 2013–2021.

Riquelmo, J., Bernardino, M., Basso, F. P., and Rodrigues, E. M. (2019). Uma linguagem específica de domínio para a representação de modelos conceituais de bancos de dados relacionais. In *Anais da III Escola Regional de Engenharia de Software*, pages 89–96, Porto Alegre, RS, Brasil. SBC.

van Deursen, A., Klint, P., and Visser, J. (2000). Domain-specific languages: An annotated bibliography. *SIGPLAN Not.*, 35(6):26–36.

Wohlin, C., Runeson, P., Hst, M., Ohlsson, M. C., Regnell, B., and Wessln, A. (2012). *Experimentation in Software Engineering*. Springer Publishing Company, Incorporated, London, England.

[5]Repositório: https://github.com/ProjetoDSL/ERDSL

Do Harpia ao Mosaicode a Evolução de um Ambiente de Programação Visual

Luan Luiz Gonçalves[1], **Flávio Luiz Schiavoni**[1]

[1] Arts Lab in Interfaces, Computers, and Everything Else - ALICE
Federal University of São João del-Rei - UFSJ
São João del-Rei - MG

fls@ufsj.edu.br, luanlg.cco@gmail.com

Abstract. *The Harpia programming environment was an important tool to develop Computer Vision applications and it became obsolete since its dependencies became deprecated. This paper presents the code refactoring of this tool and the evolution of it to a new programming environment called Mosaicode.*

Resumo. *O ambiente de programação Harpia era uma importante ferramenta para o desenvolvimento de aplicações de visão computacional que tornou-se obsoleto devido a suas dependências terem se tornado depreciadas. Este artigo apresenta a refatoração do código desta ferramenta e a sua evolução para um novo ambiente de programação, chamado Mosaicode.*

1. Introdução

O Harpia é um ambiente de programação visual que funciona como um **Gerador de código** para aplicações no domínio de processamento de imagens voltado para auxílio na educação, treinamento, implementação e gerenciamento de sistemas de Visão Computacional. Esta ferramenta foi criada em 2003 dentro do edital CT-INFO 2003 - Software Livre da FINEP pelo grupo de pesquisa multidisciplinar S2i da Universidade Federal de Santa Catarina (UFSC) e esteve presente nos repositórios oficiais do Debian e Ubuntu até aproximadamente 2009 quando seu projeto foi descontinuado. Apesar de não existir uma necessidade evidente de manutenção em seu código, a mesma possuía dependências que tiveram seu desenvolvimento descontinuado e por este motivo encontrava-se inoperante para os sistemas atuais. O presente artigo apresenta a evolução do Harpia e sua transformação na ferramenta Mosaicode a partir de uma iniciativa de torná-la operante novamente.

Como outros ambientes de programação voltados para a área de Processamento de Sinais, o Harpia utiliza o paradigma de programação visual onde a programação é feita criando **Diagramas** por meio da combinação de **Blocos** e suas **Conexões**. Cada Bloco representa uma funcionalidade do sistema e é capaz de gerar o trecho de código relacionado a esta funcionalidade onde o código gerado pode ser configurado por meio de Propriedades que definem seu comportamento. As propriedades configuradas desta maneira são chamadas de propriedades estáticas e são definidas em tempo de programação. As Conexões entre os Blocos são feitas por meio de suas Portas e servem para trocar dados entre os trechos de códigos distintos, o que permite uma configuração do código gerado em tempo de execução. Um Diagrama desta ferramenta pode ser visto na Figura 1.

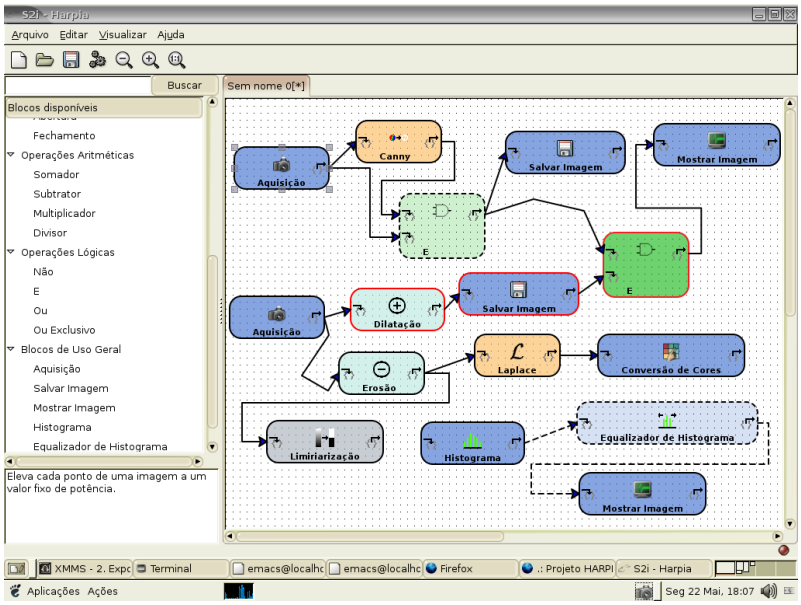


Figura 1. Diagrama feito no ambiente de programação Harpia.

Esta ferramenta possui um grande potencial para o ensino e programação de Processamento Digitais de Imagens além de possuir características claramente atrativas para outras áreas da computação como Ensino de Computação, Linguagens de Programação, Realidade Virtual, Visão computacional, Computação Musical e Arte Digital. Por esta razão, em 2014, pesquisadores do Departamento de Computação da Universidade Federal de São João del-Rei (UFSJ), entraram em contato com o grupo de desenvolvimento inicial, e de posse do código-fonte, iniciaram uma refatoração de código do Harpia com o objetivo de retomar o projeto e torná-lo novamente operante. Esta refatoração do código visava resolver as dependências quebradas e também projetar uma infra-estrutura arquitetural para a ferramenta por meio de um modelo formal para garantir uma melhor integração entre seus componentes [Garlan and Shaw 1994].

2. Evolução da ferramenta

A evolução da ferramenta partiu de, inicialmente, torná-la operante. Além disto, a modificação para corrigir a dependência quebrada serviu para aprimorar o conhecimento sobre o código-fonte e sobre o funcionamento da mesma.

2.1. Mudança na persistência

O Harpia é feito em Python e utiliza para a persistência em XML a biblioteca Amara, que está descontinuada. A modificação desta implementação envolveu uma modificação brusca no código pois a dependência de arquivos XML estava distribuída em diversas classes do projeto, o que tornava o acoplamento desta dependência bastante forte. A refatoração do código e a adoção da nova biblioteca para esta funcionalidade iniciou-se pelo isolamento de todos os métodos de persistência em uma única classe *proxy* responsável pela persistência XML. Esta classe traz para o sistema todas as funcionalidades da nova dependência e todas as classes que trabalham com persistência chamam métodos desta classe do sistema [Gamma et al. 1994].

Além da adoção de uma classe *proxy* como um único ponto de acoplamento com a persistência XML, foi adotada outra biblioteca para substituir a Amara. Neste ponto da refatoração optou-se por escolher a biblioteca Beautiful Soup para este fim por ser uma biblioteca bastante madura e cujo adoção ocorre por diversos outros projetos.

2.2. Refatoração da GUI

A GUI do Harpia é baseada na ferramenta Glade que traz diversas vantagens como o desenvolvimento rápido de componentes e telas por definir as mesmas em arquivos XML. Porém, tal estratégia de desenvolvimento implicava na distribuição dos arquivos XML com as definições de GUI e impedia que telas criadas dinamicamente fossem criadas usando o Glade. Cada Bloco da ferramenta necessitava de um arquivo Glade com a definição de sua janela de Propriedades e a alteração nas propriedades de um Bloco ou a criação de um novo Bloco implicava na utilização de diversas ferramentas. Isto tornava o desenvolvimento de novos blocos trabalhoso e não aproveitava a característica principal do Glade para o desenvolvimento de GUI que é a prototipação rápida. Além disto, o sistema dependia de uma versão antiga e descontinuada da ferramenta Glade.

Decidiu-se neste ponto alterar toda a interface gráfica do sistema para algo mais flexível que o Glade e que permitisse a criação de telas em tempo de execução para a configuração das propriedades dos Blocos. Tinha-se também uma preocupação em a) diminuir a dependência do sistema de uma biblioteca de GUI, b) isolar as classes de GUI de maneira a diminuir o impacto sobre alterações futuras nas mesmas e c) separar a definição dos Blocos e processamentos de sua representação gráfica.

Diante desta decisão, optou-se por utilizar a API Gtk em sua versão 3.22 para a refatoração da interface gráfica do sistema. Para isolar esta dependência analisou-se que a estratégia utilizada para a persistência XML de criar uma classe *proxy* não seria eficiente pois significaria a reescrita de todos os componentes Gtk. Por esta razão, a utilização da API Gtk foi feita por meio de classes que estendem a API Gtk e fornecem uma interface simples para os demais componentes do sistema. Isto tornou as dependências do projeto mais fracas onde uma modificação de um componente implica na modificação de um trecho reduzido de código. Portanto o impacto de futuras modificações como alterações na versão do Gtk foi reduzido. A nova GUI pode ser vista na Figura 2.

2.3. Refatoração das extensões e Mudança na Geração de código

O conjunto inicial de Blocos da ferramenta Harpia era focados no domínio de visão computacional e era baseado na biblioteca OpenCV em sua versão 2.4. Esta versão de biblioteca também estava descontinuada sendo que o padrão atual da mesma é a versão 3.4 e os códigos são incompatíveis entre estas versões. Para permitir que o código gerado fosse atualizado para esta versão da biblioteca, todos os Blocos definidos na ferramenta tiveram que ser reescritos para a versão mais nova do OpenCV.

A definição de um Bloco neste momento dependia de: a) um arquivo Glade para sua tela de propriedades, b) um arquivo XML para definir a ajuda, c) um arquivo de imagem para seu ícone, d) um arquivo Python para fazer a conexão entre Glade, XML e a ferramenta e e) a alteração do gerador de código da ferramenta. Os Bloco da ferramenta estavam definidos internamente no código da mesma e a criação de novos Blocos com novas funcionalidades dependia de uma alteração no código fonte do próprio Harpia.

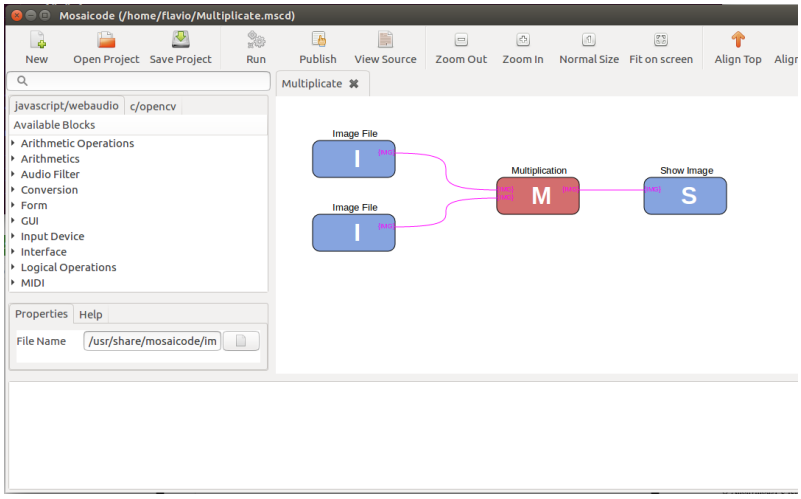


Figura 2. Diagrama feito no ambiente de programação Mosaicode.

Para simplificar a evolução da ferramenta e garantir uma maior flexibilidade e adequação do código gerado, a ferramenta foi alterada de maneira a desassociar o código dos blocos de programação do código da ferramenta.

Como estratégia de simplificar a alteração e criação de novos Blocos, neste momento optou-se por não mais ter ícones nos Blocos mas apenas uma identificação visual por cor e uma letra. Também optou-se por definir no próprio Bloco sua Ajuda, Propriedades, Portas e Código a ser gerado por este bloco. Isto reduziu a definição de um Bloco para uma única classe Python contendo campos definidos por dicionários e nenhuma método em especial. As Portas de um Bloco também foram definidas em uma classe Python contendo tanto sua definição quanto sua geração de código. Por fim, definiu-se um Padrão de código que significa a estrutura do código-fonte a ser gerado por um conjunto de blocos.

O modelo inicial de geração de código utilizava a concatenação de Textos para a montagem do código final. Por esta razão, a mesma dependia de códigos específicos para concatenar as propriedades dos Blocos. Nesta mudança, adotou-se um modelo de geração de código baseado em substituição de palavras-chaves, como no Framework Jakarta Velocity [Harrop 2004], permitindo com isto que o modelo de código gerado fosse desassociado do código da ferramenta e pudesse ser alterado sem a modificação do código-fonte da mesma. Isto permitiu que a ferramenta sofresse alterações em seus pontos de variação e que tornou possível desassociar o Harpia da geração de código para o domínio específico de Visão Computacional ou mesmo para a linguagem C.

2.4. Criação de um Meta-Modelo

Com a mudança no modo de representar os Blocos, a Geração de código na ferramenta passou a ser definidos de maneira abstrata por um conjunto de **Classes de Modelo**. Diagrama, Bloco, Conexão, Porta, Propriedade e Padrão de código passaram a compor um conjunto de classes com o Meta-Modelo do Sistema, o que permitiu modificar o sistema em tempo de execução adicionando ou removendo instâncias destas classes. Com isto, o sistema passou a trabalhar com extensões de seu Meta-modelo.

Tendo um meta-modelo comum para a representação da GUI ou mesmo para a

persistência, foi possível definir extensões para o ambiente como um conjunto novo de Blocos, Portas e Padrões de Código partindo de um meta-modelo genérico que pode ser representado tanto como classes Python quanto como arquivos XML. Foi adicionado ao sistema a capacidade de carregar extensões em tempo de execução permitindo assim desassociar a ferramenta de suas extensões e torná-la uma ferramenta genérica para geração de código baseada em modelos. O carregamento da ferramenta se dá com o carregamento das extensões na seguinte ordem: Classes Python instaladas com o sistema, Arquivos XML instalados com o sistema e Arquivos XML no espaço de usuário. Assim, se o usuário quiser alterar e personalizar Blocos em sua instância da ferramenta, o mesmo consegue fazer de maneira a alterar a instalação da ferramenta apenas para seu usuário e sem depender de acesso especial para isto.

3. Mudança arquitetural e o surgimento do Mosaicode

Para completar a modificação do sistema na refatoração da GUI, foram criadas também classes de controle para evitar que o sistema dependa totalmente das GUIs para seu funcionamento permitindo que seja possível realizar diversas tarefas passando argumentos ao programa em linha de comando. Tanto GUI quanto Persistência passaram a ter dependência mais fraca de bibliotecas externas. Para isto, foi utilizado o padrão MVC (*model, view and control*) para a refatoração do código, separando o desenvolvimento do software em camadas e facilitando a alteração da interface gráfica sem alterar a parte funcional da aplicação [Krasner et al. 1988], conforme ilustrado pela Figura 3.

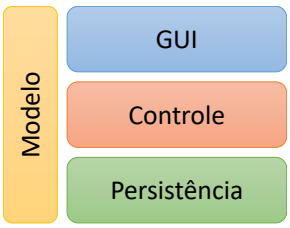


Figura 3. Arquitetura da ferramenta inspirada no modelo em camadas e na arquitetura MVC.

A **Camada de Persistência** é responsável por salvar e carregar os Diagramas feitos na ferramenta e também por carregar e persistir as extensões da ferramenta (Blocos, Propriedades, Portas e Padrões de código). Com isto, toda a persistência XML da ferramenta está isolada nesta camada, o que permite alterar facilmente esta dependência futuramente. A camada de persistência utiliza a camada de Modelo e seus métodos são chamados exclusivamente pela camada de Controle.

A **Camada de Controle** é responsável pelas ações do ambiente. É nesta camada que ocorrem a ligação entre as demais classes do ambiente, garantindo um baixo acoplamento do código. Todas as ações do ambiente estão definidas na camada de controle que toma decisões sobre quais classes devem tomar parte de quais ações. Uma classe especial do controle é o Gerador de Código (CodeGenerator) que possui a missão de validar diagramas e gerar código.

A **Camada de GUI** define a interface de usuário e é baseada no GTK versão 3. Todos os componentes gráficos do ambiente estendem uma classe Gtk e especializa esta

classe para as necessidades da ferramenta. Assim, temos classes Menu, Janela Principal, Barra de Ferramentas e assim por diante. Os componentes gráficos como Blocos, Conexões e Diagramas baseiam-se na biblioteca Goocanvas. A configuração de propriedades de um Bloco é genérica e configurável a partir da descrição de Propriedades de cada Bloco.

A **Camada de Modelo** representa os componentes do sistema como Diagrama, Porta, Conexões, Blocos e Padrão de Código. Um Bloco pode possuir Propriedades, Portas de Entrada e de Saída e uma Conexão será obrigatoriamente uma relação entre um Bloco origem (Source) e um Bloco destino (Sink) e suas respectivas portas.

Neste ponto da refatoração pudemos notar que esta versão da ferramenta seguia as premissas do Harpia mas já não tinha mais código em comum com a ferramenta inicial e também não se destinava mais ao domínio exclusivo da Visão computacional. Por esta razão, a mesma foi rebatizada com o nome de **Mosaicode**, apresentada na Figura 2.

Além de redesenhar a ferramenta e torná-la novamente operante, a evolução do Harpia para o Mosaicode contou também com a adição de novas funcionalidades como a capacidade de alinhar os blocos, novo layout de Blocos e Conexões e a documentação, ajuda de blocos por diagramas de exemplo e extensão do Meta-modelo e adição de comentários no Diagrama.

Outra funcionalidade adicionada foi a incorporação de um servidor Web que permite publicar o código gerado localmente para simplificar a colaboração e cooperação em criações colaborativas.

3.1. Extensões do Mosaicode

Uma vez que o ambiente de programação Mosaicode permite a utilização de novas extensões, alguns projetos de desenvolvimento em paralelo passaram a ocorrer para criar neste ambiente um ferramental adequado para o desenvolvimento de aplicações voltadas para o domínio de Arte Digital, Realidade Virtual e Visão Computacional.

- **Web Art:** Foi criada uma extensão para a criação de WebArt e Computação Musical baseada na linguagem Javascript e na biblioteca webaudio do HTML5. Esta extensão envolve Blocos de HTML reativos conectados por meio do padrão de projeto Observador. Estes Blocos possuem diversas funcionalidades para síntese musical, criação de sons, Criação de formulários HTML, Criação de elementos gráficos para a web, captura de sensores por páginas em dispositivos móveis, síntese de imagem com SVG e Canvas, entre outras [Gomes et al. 2019].
- **Síntese de imagens:** Baseado na biblioteca open source OpenGL na linguagem C, criou-se uma extensão para a síntese de imagem em 2D e 3D. Esta extensão permite a definição de elementos básicos da síntese de imagens, como Quadrado, Cubo, Círculo, Reta, Esfera, e também a transformação destes elementos como a escala, rotação e translação [Gomes 2019].
- **Síntese de som:** Baseado nas bibliotecas PortAudio e LibSoundFile foi criada uma extensão para processamento e síntese de som. Esta extensão possui Blocos como ruído branco, osciladores, operadores aritméticos de sinal, Ganho, entre outros e utiliza a biblioteca PortAudio para a captura e reprodução de áudio. Além disto, utiliza a biblioteca Libsoundfile para a leitura e escrita de arquivos de som [Luiz Gonçalves 2017].

- **Processamento de Imagens e Visão computacional:** Como relatado anteriormente, foi criado uma extensão para processamento de Imagem e Visão computacional baseado na biblioteca OpenCV 3.4 na linguagem C++. Esta biblioteca possui Blocos para captura de imagens por câmera ou por arquivos e também diversas operações aritméticas ou morfológicas de imagens [Resende 2019].
- **Outras extensões:** Encontra-se em desenvolvimento uma extensão para a criação de GUIs baseada na linguagem C++ e na biblioteca Gtk+3. Também está sendo desenvolvido uma extensão para a comunicação em rede e uma extensão para a implementação de controladores físicos como controladores MIDI e joysticks.

3.2. Estendendo o ambiente por plugins

Além de permitir que o ambiente seja modificado por seus usuários por meio da modificação e criação de novas extensões, foi criado também um modelo de extensão do próprio ambiente por meio de *plugins*[Syeed et al. 2015]. Um *plugin* do Mosaicode é um trecho de código distribuído a parte do ambiente de programação que altera o comportamento do mesmo adicionando novas funcionalidades. A API de *plugin* do Mosaicode permite que um *plugin* crie novos Menus, instale-se na Barra de Ferramentas do sistema e tenha acesso a todas as classes da camada de controle do ambiente.

Para isto, as classes de controle foram modificadas e diversos métodos de notificação de alteração do ambiente foram transformados para funcionar como o padrão de projeto Observador de maneira que os controles do Mosaicode consigam notificar qualquer classe interessada em saber de qualquer modificação do ambiente. Também foi criado um Carregador de novos *plugins* e um primeiro *plugin* do ambiente que permite ao usuário adicionar, alterar e remover Blocos, Portas e Padrões de código.

3.3. Adoção de ferramentas e tecnologias

Para garantir a manutenção e evolução desta ferramenta, algumas ferramentas e tecnologias foram adotadas e incorporadas no desenvolvimento da mesma.

- **Código-fonte disponível:** Desde o início deste trabalho, todas as modificações no projeto são mantidas no repositório Github e as Extensões e *Plugins* são mantidos em repositórios separados do código-fonte do ambiente.
- **Testes Unitários:** Foi adotado o pytest para a criação de testes unitários para todo o código da ferramenta, seus plugins e extensões.
- **Documentação online:** A ferramenta, plugins e extensões estão com seus códigos documentados utilizando a ferramenta sphinx e o padrão docstring.
- **Padrão de código:** O desenvolvimento também segue o padrão de código PEP8 e utiliza a ferramenta coverage para verificar se o sistema está íntegro, documentado e testado antes do lançamento de uma nova versão.
- **Empacotamento e distribuição:** a ferramenta Mosaicode, suas extensões e plugins estão empacotadas disponíveis no repositório pypi.
- **Internacionalização**: o módulo gettext foi utilizado para definir as strings contidas na ferramenta Mosaicode, oferecendo o suporte a mais de um idioma.

4. Lições aprendidas

Quando procurou-se a ferramenta Harpia no repositório Ubuntu / Debian esperava-se ter encontrado-a disponível e operante e foi uma certa surpresa ter descoberto que a mesma

estava obsoleta e que esta obsolescência tinha sido causada pela obsolescência de uma de suas dependências. Esta talvez tenha sido a primeira lição aprendida. Muitas vezes evita-se "reinventar a roda" e reescrever um código que já existe e está disponível em alguma biblioteca, framework ou componente. No entanto, nem sempre atenta-se que adotar uma biblioteca significa acoplar um código de terceiro e depender da existência desta biblioteca para garantir a longevidade do próprio *software*. Certamente esta lição não indica que sempre deve-se desenvolver novo código que já existe mas que deve-se sempre 1) avaliar a longevidade da biblioteca antes de criar a dependência e 2) tentar garantir um baixo acoplamento para esta dependência de maneira que a modificação da dependência e alteração do código não resulte em outra ferramenta, como aconteceu neste nosso exemplo. Certamente é necessário avaliar o **impacto que uma dependência** pode causar na evolução do Sistema antes de adotar a biblioteca como dependência.

Outra lição aprendida é que a manutenção de um sistema nem sempre ocorre por falha de programação, erro de código ou obsolescência do sistema. Manutenções ocorrem pois um software é um sistema complexo que costuma ter diversos componentes cuja manutenção depende da manutenção de todos os componentes do mesmo. Além disto, as necessidades dos usuários podem mudar ao longo do tempo e isto pode criar novos requisitos no sistema, requisitos estes que demandam manutenção. Por esta razão, pensar o desenvolvimento já tendo como certeza que o sistema irá precisar de manutenção constante implica em adotar diversas estratégias, de documentação à testes, para simplificar manutenções e garantir a longevidade do sistema.

5. Conclusão

Este artigo apresentou a refatoração do código e evolução do ambiente de programação visual Harpia. Na refatoração do código, a ferramenta foi reescrita e sua arquitetura foi modificada de maneira a trabalhar com extensões e *plugins*. Após isto, novas extensões foram desenvolvidas para atender outros domínios de aplicação além da Visão Computacional e se estendendo para o domínio da Arte Digital, Computação Musical, Computação Gráfica, Processamento Digital de Imagens, Realidade Virtual e Aumentada[Luiz Gonçalves 2017, Gonçalves and Schiavoni 2019, Gomes et al. 2019]. Isto permitiu isolar a geração de código da ferramenta inicial e adicionar à mesma outros domínios de aplicação e outras linguagens de programação. Após a ramificação, a ferramenta foi renomeada passando a se chamar Mosaicode.

No momento, o desenvolvimento deste ambiente tem se mostrado bastante frutífero e motivador para fomentar o desenvolvimento tecnológico em nosso grupo de pesquisa. Os primeiros testes feitos com usuários [Schiavoni and Gonçalves 2017] apontam que este ambiente auxilia na criação de aplicações para estes domínios de aplicação. Ultrapassando a barreira do desenvolvimento tecnológico, este projeto tem servido como objeto para investigação acadêmica e científica nas áreas interdisciplinares em que o projeto está inserido. Ao todo, em seis anos de projeto, já soma-se mais de 20 artigos publicados em conferências e em revistas acadêmicas / científicas além de relatórios, monografias e dissertações. Este trabalho tem envolvido alunos de graduação e mestrado em projetos de Iniciação Científica e trabalhos de conclusão de curso na área de Computação e mais recentemente na área de criação artística em cursos como Artes Aplicadas, Música e Artes Cênicas.

Os autores gostariam de agradecer ao CNPq (151975/2019-1), a FAPEMIG (APQ-02148-18), a Universidade Federal de São João del-Rei e aos membros do laboratório ALICE – Arts Lab in Interfaces, Computers, Education and Else – pelo apoio a esta pesquisa.

O repositório com o código do ambiente Mosaicode e mais informações sobre o projeto podem ser encontrados em `https://mosaicode.github.io/`.

Referências

Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1994). Design patterns: elements of.

Garlan, D. and Shaw, M. (1994). An introduction to software architecture. Technical report, Carnegie Mellon University, Pittsburgh, PA, USA.

Gomes, A., Resende, F., Gonçalves, L., and Schiavoni, F. (2019). Prototyping web instruments with mosaicode. In Schiavoni, F., Tavares, T., Constante, R., and Rossi, R., editors, *Proceedings of the 17th Brazilian Symposium on Computer Music*, pages 114–120, São João del-Rei - MG - Brazil. Sociedade Brasileira de Computação.

Gomes, A. L. N. (2019). *Extensão de Síntese de Imagens no Mosaicode para Arte Digital*. Monografia (bacharelado em ciência da computação), Universidade Federal de São João del-Rei.

Gonçalves, L. and Schiavoni, F. (2019). The development of libmosaic-sound: a library for sound design and an extension for the mosaicode programming environment. In Schiavoni, F., Tavares, T., Constante, R., and Rossi, R., editors, *Proceedings of the 17th Brazilian Symposium on Computer Music*, pages 99–105, São João del-Rei - MG - Brazil. Sociedade Brasileira de Computação.

Harrop, R. (2004). *Pro Jakarta Velocity: From Professional to Expert*. Apress.

Krasner, G. E., Pope, S. T., et al. (1988). A description of the model-view-controller user interface paradigm in the smalltalk-80 system. *Journal of object oriented programming*, 1(3):26–49.

Luiz Gonçalves, L. (2017). Sound design com o mosaicode. Monografia (bacharelado em ciência da computação), Universidade Federal de São João del-Rei.

Resende, F. R. (2019). *Processamento Digital de Imagens e Visão Computacional no ambiente Mosaicode*. Monografia (bacharelado em ciência da computação), Universidade Federal de São João del-Rei.

Schiavoni, F. L. and Gonçalves, L. L. (2017). From virtual reality to digital arts with mosaicode. In *2017 19th Symposium on Virtual and Augmented Reality (SVR)*, pages 200–206, Curitiba - PR - Brazil.

Syeed, M. M. M., Lokhman, A., Mikkonen, T., and Hammouda, I. (2015). Pluggable systems as architectural pattern: An ecosystemability perspective. In *Proceedings of the 2015 European Conference on Software Architecture Workshops*, ECSAW '15, pages 42:1–42:6, New York, NY, USA. ACM.

Modeling and Configuring UML-based Software Product Lines with SMartyModeling

Leandro F. Silva[1], **Edson OliveiraJr**[1]

[1]Informatics Department, State University of Maringá (UEM)
Maringá, Brazil.

leandroflores7@gmail.com, edson@din.uem.br

Abstract. *Variability modeling in UML-based Software Product Lines (SPL) has been carried out mostly using the UML Profiling mechanism. However, there is no UML-based SPL life cycle supporting tool, which takes advantages of UML standard diagrams in a controlled environment exclusively for it. In this scenario, we developed SMartyModeling, which allows SPL modeling on UML models, use of different visualization techniques to SPL/variability information, traceability, and configuration of products. The architecture of SMartyModeling was instantiated based on VMTools-RA, a Reference Architecture for software variability tools. This paper presents the SMartyModeling in an architectural viewpoint, describes its requirements, views, and elements selected from VMTools-RA and the decisions made during the instantiation process. We also present examples of using the environment, modeling an adaptation of the Mobile Media SPL and generating a product. We also discuss lessons learned and performed evaluations.*

1. Introduction

The requirement to achieve high levels of quality in software products has concentrated efforts from academia and industry, demanding quality in the development process and in the products created. Software engineering has evolved to create and improve methods for software development in a faster and more quality way, allowing to reach the established deadlines and goals [Long et al. 2017].

The reuse of requirements, architectures and other artifacts in a high level of abstraction is efficient in software development traditional techniques, focused on the source code. In this context, the Software Product Line (SPL) approach has been consolidated as a technique for systematic reuse in many companies around the world [Linden et al. 2007]. The SPL approach comprises a set of essential activities such as variability management, which is a key issue for success in adopting SPL. The adoption of the SPL approach aims at increasing the reuse of requirements and artifacts, thus reusing documents, source code and artifacts and ensuring better quality control to software production in large-scale [de Almeida 2019].

The industry has increasingly required the support of tools for the SPL approach [Meinicke et al. 2014]. It has been developing its own solutions to support the SPL life cycle. However, the current support tools are mainly restricted to the problem space based on feature modeling [Bashroush et al. 2017], such as FeatureIDE, AspectJ, AHEAD, Captain Feature, and DOPLER [K. U. and Nandhini 2017]. To provide support to the

solution space in SPL, we developed SMartyModeling, an environment for SPL modeling, with support to UML stereotype-based approaches such as PLUS [Gomaa 2006], Ziadi [Ziadi et al. 2003], and SMarty [OliveiraJr et al. 2010]. SMartyModeling was built based on the architectural requirements, views and elements defined by VMTools-RA [Allian 2016], a Reference Architecture (RA) for software variability tools.

2. Background and Related Work

2.1. Software Product Lines and Variability Management

The SPL approach has been consolidated as a technique for systematic reuse in many companies around the world [SPLC 2018]. SPL engineering has proven to be the methodology for developing a diversity of software products and software-intensive systems at lower costs, in shorter time, and with higher quality [de Almeida 2019]. SPL is based on the principle of systematic reuse of requirements and artifacts, including documents, source code and artifacts, ensuring better quality control to software production in large-scale. SPL corresponds to a set of software systems that share common and manageable features that contain the needs of a particular segment or mission [Linden et al. 2007].

SPL has consolidated Variability Management (VM) as one of its essential activities for a successful non-opportunistic reuse. VM encompasses a number of activities, such as identification and modeling of system variants, implementation (realization), and selection and configuration (product derivation). Software variability represents the product features in terms of variants and variation points [Capilla et al. 2013]. Many different approaches to variability representation have been discussed over the years. Thus, variability can be formally defined by four fundamental elements [Linden et al. 2007]:

- **Variation Point:** describes where differences exist in the SPL artifacts;
- **Variant:** the different possibilities that exist to satisfy a variation point;
- **Variability Dependencies:** this is used as a basis to denote the different variants that are possible to fill a variation point. The notation includes a cardinality which determines how many variants can be selected simultaneously; and
- **Constraint Dependencies:** they describe dependencies among certain variant selections. There are two forms:
  - *Requires*: the selection of a specific variant may require the selection of another variant (perhaps for a different variation point).
  - *Excludes*: the selection of a specific variant may prohibit the selection of another variant (perhaps for a different variation point).

2.2. The SMarty Approach for UML-based SPLs

SMarty-based Management of Variability (SMarty) is a VM approach based on UML stereotypes. SMarty consists of an UML profile, named *SMartyProfile*, which represents variability through stereotypes and tagged values, and a process, named *SMartyProcess*, which guides the user in identification, representation and tracking of variabilities in SPL models based on UML [OliveiraJr et al. 2010].

SMarty supports VM in use case, class, activity, component and sequence diagrams. The SMartyProfile is based on the inter-relationship of the main concepts of SPL, as Variabilities, Variation Points and Variants (Section 2.1). SMartyProfile defines the following stereotypes [OliveiraJr et al. 2010]:

- **variability:** represents variabilities in an UML note. It has the following attributes: name: name used to refer to a variability; minSelection: minimum number of variants selected to solve a variation point or variability; maxSelection: maximum number of variants selected to solve a variation point or variability; bindingTime: moment of variability resolution; allowsAddingVar: indicates whether new variants can be included after resolution of a variability; variants: collection of instances associated with variability; realizes: collection of variability of lower level models that realizes variability.
- **variationPoint:** stereotype of variation point;
- **mandatory:** represents this variant must necessarily be present in any product;
- **optional:** represents an optional variant;
- **alternative_OR:** indicates the existence of a group of inclusive variants. Different combinations of inclusive variants can be selected for the resolution of a variation point or variability;
- **alternative_XOR:** indicates the existence of a group of exclusive variants. Only one variant of this group can be selected for the resolution of a variation point or variability;
- **mutex:** represents the mutually exclusive relationship between variants;
- **requires:** represents a complement relationship between two variants.

The full description of the SMartyProfile and application examples are available in [OliveiraJr et al. 2010].

2.3. VMTools-RA: a Reference Architecture for Software Variability Tools

VMTools-RA stands for Variability Management Tools - Reference Architecture. Reference Architectures (RA) are considered a predefined standard designed for a specific business context [Nakagawa et al. 2014]. In this case, VMTools-RA encompasses knowledge and practice for developing and evolving variability tools [Allian 2016].

VMTools was created from information organized from three sources: applicable standards in the context of VM, a systematic mapping study on software variability tools, and secondary studies on software variability tools. The information analyzed served as basis for establishing 21 architectural requirements for VMTools-RA. Requirements identified for VMTools-RA were divided into two groups: (i) nine architectural requirements related to variability management implementation and (ii) 12 architectural requirements addressed to organizational support and market analysis for the maintenance of quality and evolution of variability management [Allian 2016].

VMTools-RA defines the Module View, represented in an UML class diagram, responsible for describing specific functionalities through modules (packages), data flow, and interface. VMTools-RA consists of four modules encapsulate functionalities [Allian 2016]:

- Support: provides technologies for storage of system data (e.g., information, models, and domain assets) and importing and exporting information;
- Organizational Management: is responsible for the organizational process for maintaining quality of software products. It supports an environment of communication and sharing of variability management information through different stakeholders;

- Domain Analysis: is responsible for domain asset acquisition and management and is designed with a middleware for integration to different domain analyses and requirement tools; and
- Variability Management: involves four sub-modules: Variability Modeling: identifies and models variations, traceability, documentation and composition rules related to constraints dependencies; Variability Validation: validates variability models by consistency check using arithmetical checkers or logic solvers; Variability Decision: realizes variability in different binding times; and Variability Evolution: manages variability evolution.

Figure 1 presents the Module View of VMTools-RA. Figure 1 presents the modules described, including their respective sub-modules and associations. The modules are presented emphasizing the flow of information and how the modules communicate.

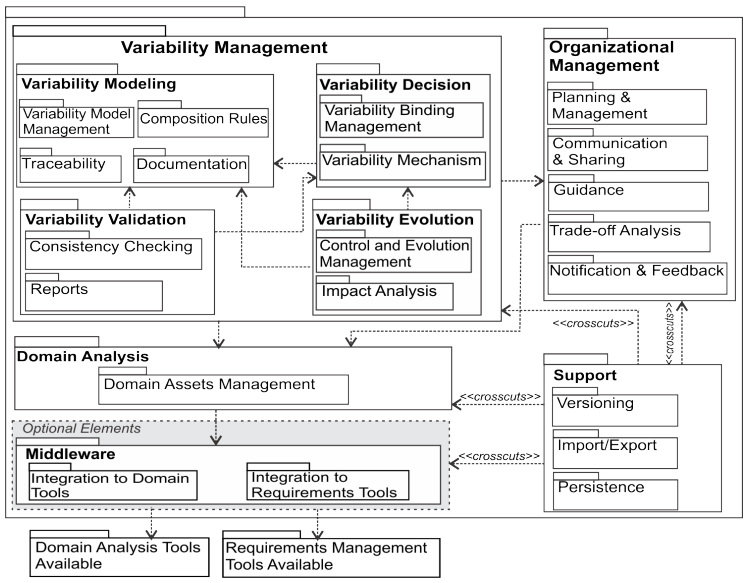


Figure 1. Module View of VMTools-RA [Allian 2016].

3. SMartyModeling: an Environment for UML-based SPLs

SMartyModeling[1] is an environment for engineering UML-based SPLs in which variabilities are modeled as stereotypes using any compliant UML profile. We are currently adopting the SMarty approach (Section 2.2) for VM. The environment supports use case, class, component, sequence, and activity diagrams. It has as main features in its current version: variability modeling and constraining, matrix-based support to traceability among SPL elements, and specific product configuration.

The creation of a RA as VMTools-RA (Section 2.3), intended for software variability tools, allows the use of consistent models, including efficient and tested solutions to reduce the complexity for the development of a new environment for variability on SPL context. In this scenario, the SMartyModeling architecture was designed and instantiated from VMTools-RA.

[1]SMartyModeling is available at `https://github.com/leandrofloress/demo_SMartyModeling_tool` for free using according to Creative Commons 4.0 licenses.

We designed the SMartyModeling architecture according to the VMTools-RA description of elements and views. As VMTools-RA is widely designed for software variability, we decided to develop an environment for SPL variability. Thus, the first step was to select the architectural requirements described by VMTools-RA for the construction of the new environment. We consider the following architectural requirements: Manage domain assets (models, UML diagrams, features) [AR.1.1]; Build variability models [AR.1.2]; Consider composition rules (e.g., cardinalities and dependencies) [AR.1.3]; Document variability models in detail [AR.1.5]; Validate conformance among variability models by using automatic consistency check [AR.1.6]; Implement impact analysis to determine what changes are required [AR.2.2]; Support import/export of variability assets and relevant information [AR.2.9]; and Offer an efficiency scalability of feature models [AR.2.11]. We adapted certain VMTools-RA views and elements for the context of SPL specifically aiming at: identifying, constraining, representing, and tracing variabilities on SPL context. Figure 2 presents the Module View of VMTools-RA instantiated for SMartyModeling architecture.

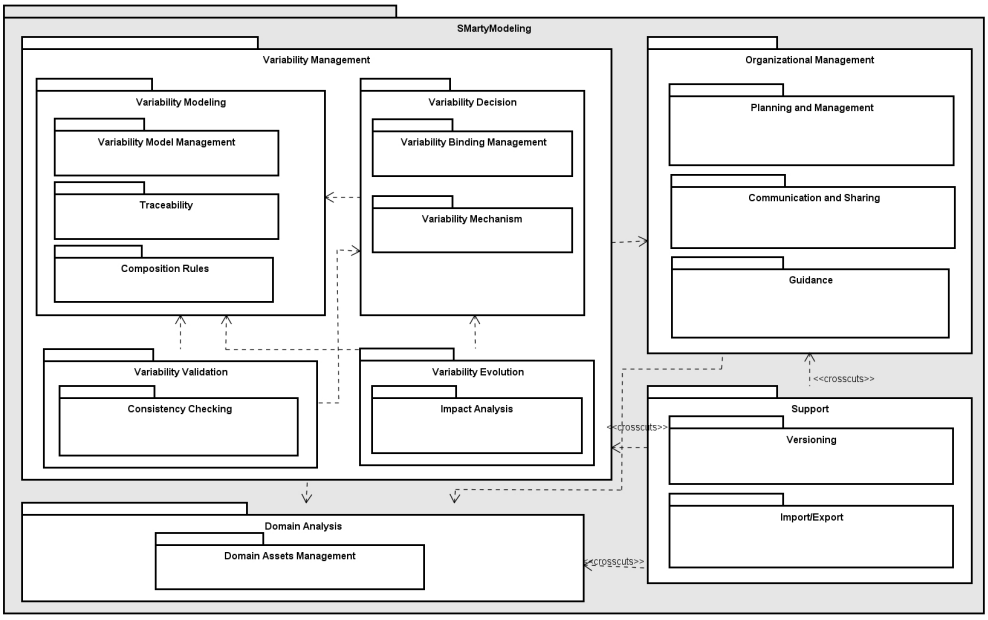


Figure 2. Module View Instance for SMartyModeling.

3.1. Why do We Need SMartyModeling?

The industry has increasingly required the support of tools for the SPL approach. However, the current support tools are mainly restricted to the problem space based on feature modeling [Bashroush et al. 2017]. In this scenario, we developed SMartyModeling, an environment for SPL modeling, with support to most UML stereotype-based approaches. The main benefit of SMartyModeling is provide support for the main activities related to SPL Management. SMartyModeling provides users control over the data and the integration of new functionalities with existing tools.

3.2. SMartyModeling Component-based Architecture

VMTools-RA does not specify an specific architectural pattern or style. Therefore, we built SMartyModeling under the Model-View-Controller (MVC) pattern. We used Java

SE for desktop. From the information considered from VMTools-RA, such as requirements, elements and modules, the functionalities were organized, designing the solutions proposed in classes and separating packages and classes according to the modules. SMartyModeling architecture is composed of classes and interfaces organized in four main packages: `Model`, `Controller`, `View`, and `File`. The latter has two sub packages: `Export` and `Import`. Package `Model` comprises the main classes and interfaces of the environment: `Project`, which is related to several other essential classes and interfaces, such as: `Profile`, `Traceability`, *`Diagram`*, `Stereotype`, and `Product`. Figure 3 depicts the internal organization of the `model` package aiming at the separation of concerns, increasing cohesion, and decreasing coupling.

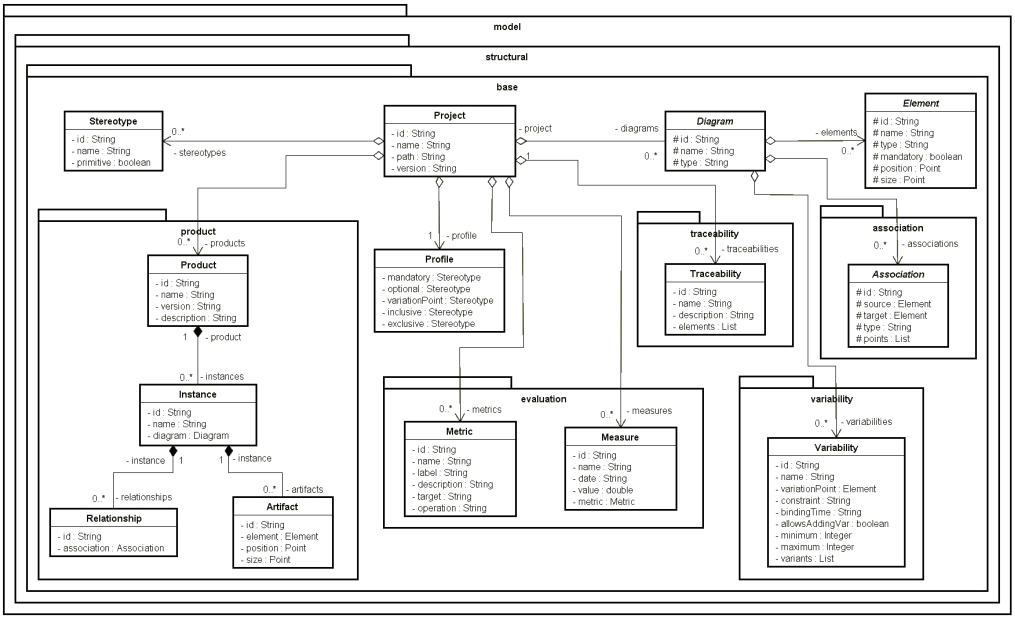


Figure 3. Classes and Interfaces of the Package `model`.

The class `Project` contains a reference to a set of *`Diagram`*, which allows variability modeling in use case, class, component, sequence, and activity diagrams. Each *`Diagram`* is a composition of *`Element`*, *`Association`*, and `Variability`. We defined the structure of a `Variability` taking into consideration tagged-values of a variability definition in the SMarty approach. Therefore, a `Variability` is related to a variation point and a set of variants to resolve such variation point.

The class `Project` is related to a `Profile`, which defines the role of each `Stereotype` for modeling variabilities. For instance, user might use the Gomaa's profile [Gomaa 2006], the Ziadi et al.'s [Ziadi et al. 2003], or the SMartyProfile [OliveiraJr et al. 2010]. We set the latter as default. One or more SPL specific `Product` might be configured for a `Project`. Each `Product` is composed of `Instance`, which is a composition of `Relationship` and `Artifact`. An `Instance` class refers to a `Diagram`, thus a `Product` is composed of a set of `Diagram`.

3.3. SMartyModeling Current Features

SMartyModeling can be divided into features. For each feature, architectural decisions were made taking into account the description of the requirements presented by VMTools-

RA. Therefore, currently, SMartyModeling consists of four main features:

- **Modeling of Diagrams:** feature responsible for defining the classes responsible for allowing the modeling of UML diagrams, including their elements and associations. The diagrams supported by the environment are: features, use case, class, component, activity and sequence;
- **Elements Traceability:** feature responsible for allowing the traceability of the elements modeled in different diagrams in the environment, allowing to associate a specific interest to a description and a set of elements;
- **Variability Management:** feature responsible for allowing variability management, with the class structure defined according to the description of the SPL concepts. By default, the environment uses the stereotypes described by the SMarty approach (Section 2.2); and
- **Product Instantiation:** feature responsible for guiding the user in the instantiation process, respecting the selection of optional elements, resolving the variability constraints and configuring the products;

3.4. How was SMartyModeling Evaluated?

Initially, we analyzed the feasibility of SMartyModeling in two studies: one qualitative and one experiment. All instrumentation and data are available at `https://zenodo.org/record/3336227`. The qualitative study allowed identifying the points to be improved, in particular, the limitation of the interface in manipulating the elements and defining the associations of SMartyModeling elements. We then, raised hypotheses mainly in relation to the extent SMartyModeling improves the application of UML-based SPL concepts and how limitations in the interface interfere SPL modeling.

Then, we performed an experiment to identify efficiency and effectiveness of SMartyModeling and Astah, and to provide initial evidence on the feasibility of SMartyModeling and its further development. Overall results support higher efficiency of Astah in relation to SMartyModeling. On the other hand, SMartyModeling had advantage in effectiveness for the sample of invited participants, based on a hypothesis test and discussed threats to validity.

4. Modeling SPLs and Configuring SPL Products

To exemplify SPL modeling with SMartyModeling, we chose Mobile Media (MM). MM is an SPL that implements mobile applications to manipulate media (photos, music and video) on mobile devices. In our example, we adapted a MM use case model. Figure 4 presents the adaptation made for the MM use case model. `User` actor and the `Delete Album`, `View Album` and `Add Album` use cases are mandatory, while the `Receive Photo` use case is optional. Two variabilities are defined, both having the `View Album` use case as a variation point. The first variability is inclusive and has four variants: `Add Photo`, `Delete Photo`, `View Photo` and `Play Music`. The second variability is exclusive and has two variants: `Set Favorite` and `View Favorites`.

The next step is to create a new Product. As presented in Figure 3, an specific product is composed of a set of Instances, directly associated with a specific Diagram. SMartyModeling guides the instantiation process of a Diagram, in a process consisting of three stages:

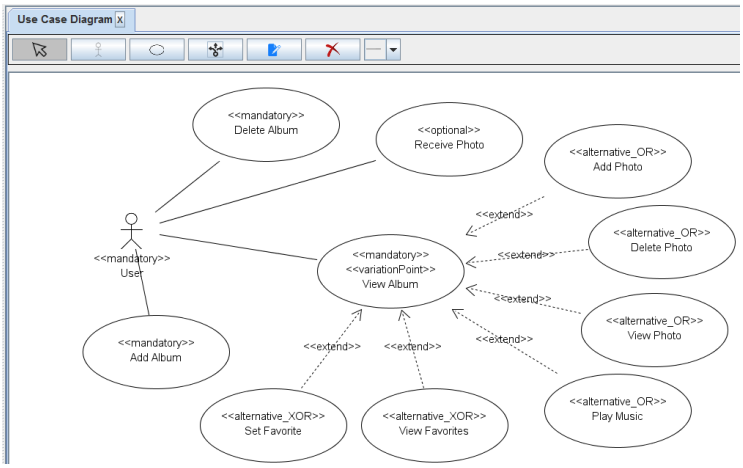


Figure 4. Mobile Media Use Case Diagram (Adapted).

- **Initial Information:** an initial panel is presented to the user, requesting information about which Product the Instance will be part of and selecting the Diagram that will be used as the basis for instantiation;
- **Selection of Optional Elements:** a panel is presented with all the elements of the diagram, with a checkbox associated with each element, and the mandatory elements of the diagram are fixed as marked and the optional elements are marked according to the user's option;
- **Resolution of Variability:** a panel is presented, according to the mandatory and optional elements selected in the previous step, listing the elements that are marked as variation point. Thus, for the diagram's variability, the following are presented:
  - **Inclusive Variability:** with the name of Variability, followed by its respective set of variants, associating each variant with a checkbox, thus allowing the user to resolve the variability, choosing the variants for the new Instance; and
  - **Exclusive Variability:** with the name of Variability, followed by a combobox with the variants defined for the variability, thus allowing the user to solve the variability, choosing a single variant for the new Instance.

Following the example, the Figure 5 shows the sequence of steps for instantiating the diagram shown in Figure 4. In the first panel of the Figure 5, we selected the base Diagram for instantiation and the Instance name. The second panel presents the mandatory elements and we selected the optional `Receive Photo` use case. And in the third panel, we solved the exclusive variability, choosing the `Set Favorite` use case and we solved the inclusive variability, selecting the `Add Photo`, `Delete Photo` and `View Photo` use cases as variants.

5. Lessons Learned on Developing SMartyModeling

The development of the SMartyModeling environment involved a series of decisions, including the views and elements described by VMTools-RA to the planning, definition, architecture instantiation and implementation of the environment. Therefore, the first lesson

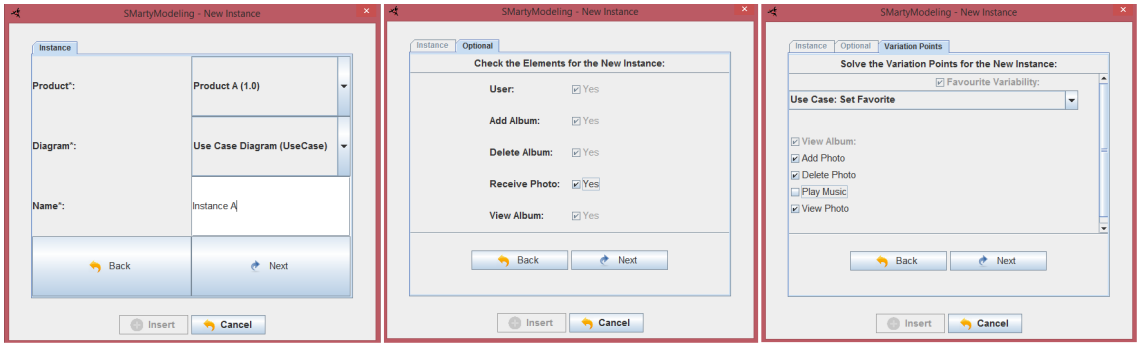


Figure 5. Interface with the 3 stages for instantiating the Moblie Media Diagram.

was that although VMTools-RA presents a complete and safe view of the concept of variability, we need to adapt these definitions to the context of SPL. During the architecture instantiation process, it was necessary to design classes and interfaces before implementation. Therefore, we design the classes in a generic way considering the requirements and views described by VMTools-RA, however, leaving the architecture flexible for changes. As a main contribution, we can mention the instantiation and implementation of an architecture from VMTools-RA, resulting in a practical application, which is part of an maturity process of an AR. The initial evaluations (Section 3.4) indicated good results, mainly in relation to the application of the SPL concepts. As a limitation, we can consider, from the point of view of instantiation, the restriction in developing solutions for the modules related to the most complete documentation, reports, and evolution of variability, and activities of organizational management. From a practical point of view, the evaluations also indicated limitations reported by the participants, in particular, regarding the definition of associations and event handling in the modeling panel.

6. Final Remarks

We presented SMartyModeling, a tool that supports the main activities on SPL management. We described the main steps of the development process, starting from the selection of VMTools-RA architectural requirements, views and elements, the adaptation of the instantiated architecture to the SPL context, the lessons learned in the implementation and the initial evaluations performed. VMTools-RA considers software variability tools in a broader context. Therefore, the first decision was to restrict the representation of variability for the context of SPL. However, regardless of this restriction, the concepts and elements described mainly in terms of Variability Management were taken into account, being naturally adapted to the SPL domain.

The instantiation and implementation of an architecture from VMTools-RA collaborates mainly with a practical application and is part of a maturity process for the RA. Among the points to improve the environment it could include more elements described by VMTools-RA. In particular, regarding the evolution of variability, planning a solution that included a complete control with information and management of the evolution of variability during the project.

We have designed two more evaluations for SMartyModeling. The first aims to evaluate the SMartyModeling instantiation process according to the guidelines presented by VMTools-RA. For this evaluation, SPL experts will be invited, presenting the require-

ments, views and guidelines of VMTools-RA,as well as the documents with the instantiation decisions, architecture description and documentation of SMartyModeling. The second one aims at evaluating usability, without the objective of comparing it with another tool. Such evaluation will be carried out with potential users of SMartyModeling by extending the Technology Acceptance Model with the System Usability Scale.

References

Allian, A. P. (2016). VMTools-RA: a Reference Architecture for Software Variability Tools. Master's thesis, State University of Maringá. in portuguese.

Bashroush, R., Garba, M., Rabiser, R., Groher, I., and Botterweck, G. (2017). Case tool support for variability management in software product lines. *ACM Comput. Surv.*, 50(1):14:1–14:45.

Capilla, R., Bosch, J., and Kang, K. C. (2013). *Systems and Software Variability Management: Concepts, Tools and Experiences*. Springer.

de Almeida, E. S. (2019). Software Reuse and Product Line Engineering. In Cha S., Taylor R., K. K., editor, *Handbook of Software Engineering*, pages 321–348. Springer, Cham, Switzerland.

Gomaa, H. (2006). *Designing software product lines with uml 2.0: From use cases to pattern-based software architectures*. Springer-Verlag.

K. U., K. and Nandhini, M. (2017). Classification of Tools For Feature-Oriented Software Development A Comprehensive Review. *International Journal of Computer Sciences and Engineering*, 5:329–337.

Linden, F. J. V. D., Schmid, K., and Rommes, E. (2007). *Software product lines in action: The best industrial practice in product line engineering*, volume 20. Springer-Verlag New York, Inc.

Long, F., Amidon, P., and Rinard, M. (2017). Automatic inference of code transforms for patch generation systems. In *ACM SIGSOFT FSE*, pages 727–739.

Meinicke, J., Thum, T., Schroter, R., Benduhn, F., and Saake, G. (2014). An Overview on Analysis Tools for Software Product Lines. In *Proceedings of the 18th International Software Product Line Conference: Companion Volume for Workshops, Demonstrations and Tools - Volume 2*, page 94–101, New York, NY, USA. Association for Computing Machinery.

Nakagawa, E. Y., Guessi, M., Maldonado, J. C., Feitosa, D., and Oquendo, F. (2014). Consolidating a process for the design, representation, and evaluation of reference architectures. In *WICSA*, pages 143–152, Washington, DC, USA.

OliveiraJr, E., Maldonado, J. C., and Gimenes, I. M. S. (2010). Systematic Management of Variability in UML-based Software Product Lines. *Journal of Universal Computer Science (JUCS)*, pages 2374–2393.

SPLC (2018). *Software Product Line Conference - Hall of Fame*. URL: https://splc.net/fame.html.

Ziadi, T., Hélouët, L., and Jézéquel, J.-M. (2003). Towards a uml profile for software product lines. In *PFE*, pages 129–139. Springer.

Práticas e Ferramentas Utilizadas em Startups de Software Paranaenses: Um Estudo Exploratório

Bruno Henrique Cavalcante[1], Liandra Dos Santos Jesus[2], Gislaine Camila Lapasini Leal[2], Renato Balancieri[3], Ivaldir De Farias Junior[4]

[1]Departamento de Informática – Universidade Estadual de Maringá (UEM)
CEP 87.020-900 – Maringá – PR – Brasil

[2]Departamento de Engenharia de Produção – Universidade Estadual de Maringá (UEM)
CEP 87.020-900 – Maringá – PR – Brasil

[3]Universidade Estadual do Paraná (UNESPAR) – Campus Apucarana
CEP 86.813-250 – Apucarana – PR – Brasil

[4]Universidade de Pernambuco (UPE) – Campus Garanhuns
CEP 50.100-010 – Recife – Pernambuco – Brasil

bruno.h.cavalcante@gmail.com, {ra102374,gclleal}@uem.br,
renato.balancieri@unespar.edu.br, ivaldir.junior@upe.br

Abstract. *This survey seeks to identify techniques and tools applied on software development in startups located in the Brazilian state of Paraná. Expecting to be able to contribute to the academic and industrial field, by entering an area that is still relatively unknow and presenting relevant information for them, especially for companies that are just starting out.*

Resumo. *Este estudo exploratório busca identificar as práticas e ferramentas utilizadas no desenvolvimento de software em startups do Paraná. Buscando contribuir tanto para a academia quanto para a indústria, por explorar uma área ainda pouco estudada e apresentar informações importantes para essas startups, principalmente aquelas que estão iniciando.*

1. Introdução

Startups são pequenas organizações que estão buscando um modelo de negócio que se expanda exponencialmente, por meio da oferta de um produto ou serviço [Paternoster *et al.* 2014]. É comum que as mesmas acabem por criar novos mercados, entregando ao público produtos ou serviços que consequentemente geram novas necessidades.

Por meio de mapeamentos sistemáticos Paternoster *et al*. (2014) e Klotins *et al*. (2015) apontaram a necessidade de aumentar as pesquisas em relação as práticas de desenvolvimento de *software*, com o intuito de replicá-las e reutiliza-las em outros setores da indústria de tecnologia. Porém, apesar do desenvolvimento de *software* ser o centro das atividades de uma *startup*, o mesmo não possui uma base de conhecimento científico bem definido. Neste contexto, o objetivo deste estudo é apresentar as práticas e as ferramentas adotadas no desenvolvimento de *software* em *startups* paranaenses.

2. Método de Pesquisa

Este trabalho foi conduzido por meio de um *survey* exploratório, um método de pesquisa quantitativo frequentemente utilizado em pesquisas empíricas na Engenharia

de *Software* como forma de obter dados de diversas fontes e explorar áreas pouco estudadas [Wohlin *and* Aurum 2015].

E a partir de *sites* de associações e agrupamentos de *startups* foi possível obter a lista das empresas participantes da pesquisa. Sendo que o procedimento de coleta consistiu do envio de e-mails para as *startups* encontradas, contendo a carta de apresentação e o *link* de acesso ao utilizando a ferramenta de Formulários do Google[1]. Na qual, o mesmo ficou disponível por um período de 120 dias.

2. Resultados e Discussões

Os dados considerados para essa pesquisa foram 28 registros. Após a exclusão dos *outliers* por respostas incompletas, o total da amostra foi reduzido para 26.

Em relação aos 26 respondentes da pesquisa, 65% possuem mais do que 5 anos de experiência, indicando que a grande maioria dos envolvidos nas *startups* possuem uma quantidade considerável de experiência com desenvolvimento de *software*. Do restante, 15% possuem entre 3 e 5 anos, 12% possuem entre 1 e 3 anos e 8% possuem menos do que 1 ano de experiência. Sendo que 42% é especialista, 30% são graduando ou graduado, 12% mestrando, 8% Mestre, 8% doutorando ou Doutor. Isto demonstra que boa parte das pessoas que estão envolvidas com *startups* são recém-formadas ou acabaram de terminar uma pós-graduação, ou então são pessoas que decidiram não investir em uma carreira acadêmica além da graduação e da especialização. Porém, apenas 4% dos respondentes têm mais do que 5 anos de experiência com *startups,* 31% entre 3 e 5 anos, 27% entre 1 e 3 anos e 38% menos de um ano.

2.1 Caracterização das *Startups*

A Figura 1 apresenta a distribuição das *startups* de acordo com o seu ano de fundação. E por meio da linha de tendência é possível verificar o crescimento exponencial do número de *startups* fundadas entre 2010 e 2017. É importante salientar que o questionário foi disponibilizado em meados de julho de 2017, ficando disponível até meados de outubro de 2017, logo, é possível observar que a quantidade de *startups* fundadas no ano 2017 está incompleta.

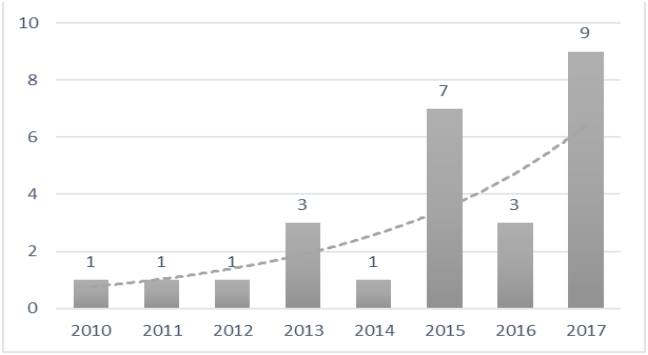


Figura 1. Ano de fundação das startups.

A classificação do nível de maturidade das *startups* participantes desta pesquisa, foi realizada segundo Crowne (2002), que descreve a maturidade das *startups* em três

[1] https://goo.gl/forms/QqksndXyuNCSRPFw1

fases: a inicial, de estabilização e de crescimento. Assim, foi identificado que 46% encontram-se em fase inicial, e que de acordo com mesmo autor estão entre a concepção do produto e a realização da primeira venda. 19% em fase de estabilização, ou seja, entre o momento em que o primeiro consumidor recebe o produto até o momento em que o produto está estável o suficiente para ser vendido a um novo consumidor sem causar sobrecarga no desenvolvimento. E 34% em fase de crescimento, ou seja, entre o fim da fase anterior e acaba quando as taxas de crescimento e fatia de mercado estão estáveis e todos os processos necessários para o desenvolvimento e venda do produto estão estabilizados [Duc *et al.* 2016].

Os dados permitem concluir que quase metade das *startups* estão em fase inicial, a minoria encontra-se na fase de estabilização, e o restante das *startups* estão na fase de crescimento sendo capazes de entregar seus produtos para mais de um cliente e em breve irão se estabilizar, tornando-se organizações maduras. Entretanto, chegar a esta fase de maturidade é um grande desafio para essas organizações, visto que a maioria destas não conseguem atingir dois anos de vida.

Independentemente do nível de maturidade da *startup*, os dados demonstram que o investimento externo não é um dos principais fatores para o sucesso. Porém, a participação das mesmas em grupo ou rede de colaboração é alta nas fases inicial e de estabilização. Na fase de estabilização o número de *startups* que fazem parte de um grupo ou rede de colaboração é maior, mas com uma diferença de apenas 4% das que não fazem parte. E segundo Santos (2016) a participação traz benefícios como, maior facilidade de comunicação entre as organizações, além de facilitar a troca de conhecimento etc., o que não faz sentido neste trabalho devido que 75% das empresas que estão na fase crescimento não fazem parte de alguma rede ou grupo.

Obviamente que somente com esses dados não é possível chegar a nenhuma conclusão, mas é possível gerar perguntas a serem respondidas em uma pesquisa mais direcionada, para compreender esse comportamento e identificar se as *startups* que chegam a essa fase não utilizam-se de parcerias ou deixam de utilizá-las; e se não utilizam-se de investidor pelo fator de não conseguirem um ou não necessitarem.

2.2 Caracterização do Desenvolvimento de Software nas *Startups*

O estudo analisou a utilização das metodologias ágeis e tradicionais, práticas e técnicas de desenvolvimento, suítes/plataformas de desenvolvimento, ferramentas de controle de versão e configuração, de gerenciamento de projeto e de gestão do conhecimento.

Em relação as metodologias ágeis, 69% das *startups* indicaram utilizá-las, e 31% não, os dados demonstram uma maior tendência de adoção aos métodos ágeis, dividido entre (*Scrum*, *Lean/Lean Startup* e *Test Driven Development* - TDD). Além disso, 1 *startup* confundiu metodologia e técnica, e indicando o *Kanban* como metodologia. Estes métodos permitem que o produto seja desenvolvido em pequenas iterações e entregas frequentes, possibilitando que sejam obtidos *feedbacks* constantes dos usuários. Isso ajuda a visualizar se o desenvolvimento do produto está indo na direção correta, permitindo que sejam realizados ajustes quando este não é o caso.

Observando os dados da Tabela 1, que apresenta um painel das práticas adotadas pelas *startups* paranaenses de *software*. Assim, na fase inicial, tem-se que *Backlog* é a prática mais utilizada seguida de padrões de código, enquanto o uso de *frameworks*

destaca-se na fase de estabilização, seguido do *Kanban*, e por fim na fase de crescimento a capacitação dos membros da equipe, o MVP e o uso de *frameworks* são as práticas mais utilizadas.

Tabela 1. Painel de práticas.

Práticas	Fase Inicial (12)	Fase de Estabilização (5)	Fase de crescimento (9)
Backlog	10 (83,33%)	1 (20,00%)	6 (66,67%)
Bug-tracking	1 (8,33%)	1 (20,00%)	3 (33,33%)
Capacitação dos membros da equipe	5 (41,67%)	1 (20,00%)	8 (88,88%)
Cliente no local de desenvolvimento	1 (8,33%)	1 (20,00%)	2 (22,22%)
Code review	7 (58,33%)	1 (20,00%)	2 (22,22%)
Empoderamento da equipe de desenvolvimento	2 (16,67%)	2 (40,00%)	5 (55,56%)
Entregas frequentes/contínuas	6 (50,00%)	2 (40,00%)	7 (77,78%)
Gerenciamento de requisitos	5 (41,67%)	1 (20,00%)	4 (44,44%)
Integração contínua	5 (41,67%)	2 (40,00%)	4 (44,44%)
Kanban	4 (33,33%)	3 (60,00%)	4 (44,44%)
MVP (*Minimum Viable Product*)	7 (58,33%)	1 (20,00%)	8 (88,88%)
Padrões de código	8 (66,67%)	2 (40,00%)	7 (77,78%)
Planning Game	2 (16,67%)	1 (20,00%)	3 (33,33%)
Programação em pares	1 (8,33%)	1 (20,00%)	2 (22,22%)
Prototipagem evolucionária	3 (25,00%)	0 (0,00%)	3 (33,33%)
Releases curtos	5 (41,67%)	0 (0,00%)	4 (44,44%)
Reuso de Software	0 (0,00%)	1 (20,00%)	7 (77,78%)
Uso de *frameworks*	7 (58,33%)	4 (80,00%)	8 (88,88%)
Uso de indicadores de desempenho	1 (8,33%)	0 (0,00%)	6 (66,67%)
Uso de logs e estatísticas	1 (8,33%)	0 (0,00%)	7 (77,78%)
Uso de soluções COTS ou *open source*	1 (8,33%)	2 (40,00%)	5 (55,56%)

O uso de *frameworks* agiliza o processo de desenvolvimento, por retirar a necessidade de implementar algo que já está pronto [Giardino *et al*. 2014]. Outra prática popular é o uso de *backlog*, sendo uma das principais práticas da metodologia *Scrum*, a metodologia mais utilizada entre os respondentes. O mesmo permite que a equipe de desenvolvimento tenha um registro sempre atualizado das tarefas que ainda devem ser realizadas.

O MVP cumpre um papel importante nos processos de uma *startup*, servindo como um artefato de *design* e como artefato recusável, além de ser o ponto chave da metodologia *Lean/Lean Startup* [Ries 2011], permitindo testar o produto com o menor investimento possível e assim, acelerar o seu lançamento. O *Kanban* é uma técnica proporciona velocidade na produção e um ciclo rápido e contínuo de *feedback* com o cliente.

A Tabela 2 apresenta um painel das ferramentas de gerenciamento de projeto, controle de versão e configuração, suítes/plataformas de desenvolvimento e de gestão de conhecimento utilizadas pelas *startups* participantes. Contudo, em relação as suítes/plataformas de desenvolvimento, é difícil argumentar quanto a sua utilização, visto que a escolha das mesmas geralmente depende de fatores particulares da

organização, de desenvolvimento ou do desenvolvedor, restrições de orçamento e público alvo também podem influenciar na decisão.

Em termos de ferramentas de gerenciamento de projeto, na fase inicial e de crescimento o *Trello* aparece como a mais popular provavelmente pelo fato de o mesmo ser organizado por meio de cartões de transitam entre painéis, além de ser uma ferramenta gratuita no plano mais básico.

Tabela 2. Apresentação das ferramentas em relação ao nível de maturidade.

Nível de Maturidade	**Número de *startups***	**Ferramentas**			
		Gerenciamento de Projeto	**Controle de versão e configuração**	**Suítes/plataformas de desenvolvimento**	**Gestão do conhecimento**
Fase Inicial	12 (46%)	Trello (7) Redmine (2) Atlassian Jira (1) Asana (1) Basecamp (1) Taiga (1) Nenhuma (1)	Github (7) Gitlab (4) Bitbucket (2)	Visual Studio (6) Eclipse (5) NetBeans (2) Atom (3) XCode (1) RadStudio Deplhi (1)	Freshdesk (1) Atlassian Confluence (1) Open KM (2) Collectiva Knowledge (2) Slack (1) Drive (1) Nenhuma (6) Google Hangouts (1)
Fase de estabilização	5 (19%)	Trello (1) Redmine (1) IBM Jazz (1) Asana (1) Nuenhuma (1)	Git (1) Github (2) Gitlab (1) Bitbucket (2) Subversion (2)	Visual Studio (2) Eclipse (1) NetBeans (2) JetBrains (2) XCode (1)	Freshdesk (1) Atlassian Confluence (1) World (1) Drive (1) Nenhuma (1)
Fase de crescimento	9 (35%)	Trello (8) Redmine (2) Atlassian Jira (3) Google Docs (1)	Github (3) Gitlab (1) Bitbucket (3) Subversion (4) Nuenhuma (1)	Visual Studio (2) Deplhi (1) Atom (2) Netbeans (2) VIM (2) JetBrains (2) PyCharm (1) PHPStorm (1)	Atlassian Confluence (3) Jira (2) Collectiva Knowledge (1) zendesk (1) Vtiger (1) Drive (1) Nenhuma (2)

As ferramentas de controle de versão e configuração facilitam uma análise do histórico de mudanças são, portanto, poderosas ferramentas de *backup*. Além disto, por meio das versões é possível verificar quais trechos de código foram modificados, quem fez as mudanças e também métricas como, número de mudanças dentro de um período de tempo, número de erros e indicar inanição. Assim, na fase inicial e de estabilização, o *Github* foi a ferramenta mais utilizada nesse contexto, já na fase de crescimento foi o *Subversion*.

Por fim, as ferramentas de gestão do conhecimento mais utilizadas na fase inicial foram *Open KM e Collectiva Knowledge* e na fase de crescimento foi *Atlassian Confluence*. Já na fase de estabilização, quatros ferramentas tiveram o mesmo número indicações (*Freshdesk, Atlassian Confluence, World e Drive*). Sendo que, uma das

startups indicou que está estudando a respeito de qual ferramenta utilizar e 8 não utilizavam nenhuma ferramenta nesse contexto.

No caso da escolha de qual ferramenta utilizar, ocorre restrições ligadas ao perfil da organização e de seus desenvolvedores. A condução de concursos de inovação tem potencial para trazer ideias de fora da organização ao observar como outras *startups* gerenciam o seu conhecimento de *software*, a partir disto é possível traçar uma estratégia de gerenciamento mais adequada.

3. Considerações Finais

Este estudo permitiu verificar que nos últimos anos o número de *startups* fundadas no Paraná vem crescendo de forma acelerada, e também é possível verificar que o fato de a *startup* possuir investidor externo ou fazer parte de um grupo ou rede de colaboração tem pouca relação com o nível de maturidade da mesma.

No que se refere as práticas relacionadas ao desenvolvimento de *software*, os resultados demonstram que a maioria das *startups* paranaenses utilizam metodologias ágeis, com destaque para o *Scrum* e o *Lean Startup*. Ainda assim, vale notar que, por razões desconhecidas, algumas utilizam metodologias tradicionais, sendo essa uma oportunidade de pesquisa, uma vez que *startups* estão fortemente ligadas ao uso de métodos ágeis e, portanto, seria interessante entender as motivações ligadas a escolha oposta. Adotar noções e práticas mais abertas poderiam auxiliá-las com os desafios provenientes da limitação de recursos e competências.

Estes resultados são importantes pois permitem monitorar o estado de desenvolvimento das *startups* paranaenses e possibilitam que novos estudos sejam conduzidos com base nos dados adquiridos, permitindo que novas hipóteses sejam formuladas a partir do que foi encontrado. Do ponto de vista da indústria, estes resultados servem como um ponto inicial para as *startups* que estão iniciando suas operações e ainda não possuem metodologias definidas, ferramentas, técnicas ou práticas que irão utilizar em seu processo de desenvolvimento.

References

Crowne, M. (2002), Why software product startups fail and what to do about it? Evolution of software product development in startup companies. In Engineering Management Conference, 2002. IEMC'02. 2002 IEEE International, 1, pp.338-343.

Duc, A. N., Shah, S. M. A. and Ambrahamsson, P. (2016), Towards an early stage software startups evolution model. In 42th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), 2016, pp. 120–127.

Giardino, C., Unterkalmsteiner, M., Paternoster, N., Gorschek, T. and Abrahamsson, P. (2014), What do we know about software development in startups? IEEE software, 31, 5, pp. 28–32.

Klotins, E., Unterkalmsteiner, M. and Gorschek, T. (2015), Software engineering knowledge areas in startup companies: a mapping study. In International Conference of Software Business, pp. 245–257.

Paternoster, N., Giardino, C., Unterkalmsteiner, M., Gorschek, T. and Abrahamsson, P. (2014), Software development in startup companies: A systematic mapping

study. The 29th International Conference on Software Engineering and Knowledge Engineering, 56, 10, pp. 1200–1218.

Ries, E. (2011), The Lean Startup. New York: Crown Business.

Santos, M. C. F. R. (2016), O ecossistema de startups de software da cidade de São Paulo. PhD thesis, Universidade de São Paulo.

Wohlin, C. and Aurum, A. (2015) Towards a decision-making structure for selecting a research design in empirical software engineering. Empirical Software Engineering, 20, 6, pp. 1427–1455.