ERES 2018
DOIS VIZINHOS - PR

SEMANA DE CIÊNCIA E TECNOLOGIA DE DOIS VIZINHOS 2018

Realização

SBC Sociedade Brasileira de Computação

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

II Escola Regional de Engenharia de Software

ANAIS

Graduação
Pós-Graduação
Indústria

Apoio

PALMARIUM GARDEN CENTER

SUDOTEC desenvolvimento e tecnologia

SEBRAE

Unisep
UNIÃO DE ENSINO DO SUDOESTE DO PARANÁ

Prefeitura Municipal de Dois Vizinhos

Dois Vizinhos

22 a 26 de Outubro

@eres2018oficial

@EscolaERES

@eres2018sbc

coens-dv@utfpr.edu.br

www.eres.net.br

Patrocinadores

COPEL Pura Energia

KEPHA Digital Business Experts

DEZ TELECOM

CRESOL

servDATA SERVIÇOS CONTÁBEIS

Visual SOFTWARE

zallpy group

CISS Gestão para o Varejo

CAMALEON marketing

SS SISTEMAS

DATACOPER

CONCREVALLE

CodePlus

II ERES
Anais da 2ª Escola Regional de Engenharia de Software (ERES 2018)

Universidade Tecnológica Federal do Paraná – Dois Vizinhos-PR

20 de Outubro de 2018

E79a Escola Regional de Engenharia de Software – ERES (2.: 22-24 out. 2018.: Dois vizinhos, PR)
Anais da 2ª Escola Regional de Engenharia de Software [recurso eletrônico] / Organização: Universidade Tecnológica do Federal do Paraná – UTFPR – Campus Dois Vizinhos. – Dois Vizinhos: UTFPR, 2018.
189p.:il.

Anual
Realização: Sociedade Brasileira de Computação – SBC
ISSN: 2594-9896

1. Engenharia de Software. 2. Ensino Superior. 3. Computação. I. Universidade Tecnológica Federal do Paraná – Dois Vizinhos.

CDD: 004.41

Catalogação na fonte: Keli Rodrigues do Amaral Benin - CRB: 9/1559
Biblioteca da UTFPR-Dois Vizinhos

II ERES 2018

2ª Escola Regional de Engenharia de Software

COMISSÃO ORGANIZADORA

Coordenação Geral - Rafael Alves Paes de Oliveira (UTFPR-DV)
Coordenação Local - Newton Carlos Will (UTFPR-DV)
Coordenador Financeiro - André Roberto Ortoncelli (UTFPR-DV)
Coordenador do Fórum de Pós-graduação - Elder de Macedo Rodrigues (UNIPAMPA)
Coordenador do Fórum de Graduação - Maicon Bernardino da Silva (UNIPAMPA)
Coordenador da Trilha Indústria - Francisco Carlos Monteiro Souza (UTFPR-DV)

COMISSÃO EXECUTIVA DA ERES - CEERES

Rafael Alves Paes de Oliveira (UTFPR-DV)
Elder de Macedo Rodrigues (UNIPAMPA)
Maicon Bernardino da Silva (UNIPAMPA)
Edson Oliveira Júnior (UEM)
Marcelo Hideki Yamaguti (PUC-RS)
Ingrid Nunes (UFRGS)
Fernando dos Santos (UDESC)

COMITÊ DE CIENTÍFICO

Comitê do Fórum de Pós-graduação

André Endo (UTFPR)
Edson Oliveira Jr (UEM)
Fábio Basso (UNIPAMPA)
Maicon Bernardino (UNIPAMPA)
Marco Aurélio Graciotto (UTFPR)
Silvia Vergilio (UFPR)

Comitê do Fórum de Graduação

Adolfo Neto Federal (UTFPR)
Alessandra Dutra (PUCRS)
Aline Mello (UNIPAMPA)
Ana Paula Bacelo (PUCRS)
Andre Ortoncelli (UTFPR)
Andrea Bordin (UNIPAMPA)
Érica Souza (UTFPR)
Elder Rodrigues (UNIPAMPA)
Evandro Kuszera (UTFPR)
Fabiane Benitti (UFSC)
Fabrício Pretto (UNIVATES)
Fábio Basso (UNIPAMPA)
Fernando Santos (UDESC)
Gilleanes Thorwald Araujo Guedes (UNIPAMPA)

Gustavo Jansen de Souza Santos (USP)
Ingrid Nunes (UFRGS)
Lisandra Fontoura (UFSM)
Marcelo de Souza (UDESC)
Marcelo Yamaguti (PUCRS)
Maria Claudia Emer (UTFPR)
Marisângela Pacheco Brittes (UTFPR)
Michel Albonico (UTFPR)
Paulo Varela (UTFPR)
Pedro Machado (UTFPR)
Rafael Alves Paes de Oliveira (UTFPR)
Ricardo Pereira e Silva (UFSC)
Sabrina Marczak (PUCRS)
Simone S. Borges (UTFPR)
Thais Webber (UNISC)
Victor Santander (UNIOESTE)
Willian Watanabe (UTFPR)

Comitê da Trilha Indústria

Alinne C. Souza (UTFPR)
Pedro Machado (UTFPR)
Rafael de Oliveira (UTFPR)
Rodrigo Pagno (UTFPR)
Diógenes Dias (USP)
Livia Degrossi (USP)
Lina Garcés (USP)
Ricardo Vilela (USP)
Victor Santiago (UFLA)
Carlos Soares de Souza (IFSul)
Rafael Daron (Unisep)
Andre Abe Vicente (UOL)

REALIZAÇÃO

Sociedade Brasileira de Computação - SBC

ORGANIZAÇÃO

Universidade Tecnológica Federal do Paraná - Câmpus Dois Vizinhos - UTFPR-DV

PATROCÍNIO

Camaleon Marketing
CISS Gestão para o Varejo
Codeplus
Concrevalle
Copel
Cresol Baser
Datacoper
DEZ Telecom
Kepha Digital Business Experts
Servdata Serviços Contábeis

SS Sistemas
Visual Software
Zallpy Group

APOIO

PALMARIUM Garden Center
Prefeitura Municipal de Dois Vizinhos
SEBRAE
SUDOTEC Dois Vizinhos
Unisep

Editorial

É com grande satisfação que apresentamos os artigos aceitos para a segunda edição da Escola Regional de Engenharia de Software (ERES), a qual compõem os anais do evento. A II ERES ocorreu de 22 a 24 de outubro de 2018, na cidade de Dois Vizinhos - PR, sob a organização da Universidade Tecnológica Federal do Paraná, câmpus Dois Vizinhos.

A ERES é promovida anualmente pela Sociedade Brasileira de Computação (SBC) com o objetivo disseminar o conhecimento e boas práticas em Engenharia de Software (ES), tanto do ponto de vista profissional quanto acadêmico. A ERES 2018 é um espaço regional para que possam ser apresentados os resultados de pesquisas de graduação e pós-graduação e relatos de experiência na indústria. Além disso, possibilitará um ambiente natural para a discussão de abordagens no ensino-aprendizagem na ES nas instituições de ensino superior da Região Sul do Brasil.

Mantendo a tradição de outras escolas regionais, foram aceitas submissões de artigos em duas categorias: Fórum de Graduação e Fórum de Pós-graduação. Além destes, esta edição também contou com uma terceira categoria, a Trilha da Indústria, sendo um espaço para apresentação de trabalhos de profissionais do setor produtivo de T.I, com o objetivo de incentivar a troca de experiências entre profissionais da academia e insdústria. Todos os artigos foram avaliados por pelo menos 3 membros do Comitê Científico. Além das sessões técnicas, a programação do evento contou com palestras e minicursos proferidas por pesquisadores de renome e profissionais da indústria da área de Engenharia de Software.

O Fórum de Graduação recebeu 25 submissões, das quais 17 foram aceitas para apresentação oral, o que representa 68% de taxa de aceitação. Por outro lado, o Fórum de Pós-graduação obteve 2 submissões, sendo todas aceitas, o que representa 100% de taxa de aceitação. Já a Trilha da Indústria obteve 5 submissões, das quais 3 foram aceitas para apresentação, o que representa 60% de taxa de aceitação.

Os Anais da II ERES representam o resultado do esforço coletivo de um grande número de pessoas. Agradecemos aos docentes, discentes e técnicos-administrativos da UTFPR-DV que trabalharam arduamente para garantir o bom andamento do evento. Gostaríamos de agradecer também aos membros do Comitê Científico que fizeram revisões de excelente qualidade. Um agradecimento especial aos professores Elder de Macedo Rodrigues (UNIPAMPA), Maicon Bernardino da Silva (UNIPAMPA) e Francisco Carlos Monteiro Souza (UTFPR-DV) na dedicação e empenho no gerenciamento e organização dos fóruns científicos. Por fim, agradecemos aos autores que submeteram seus trabalhos para a ERES 2018.

Prof. Dr. Rafael Alves Paes de Oliveira – Coordenador Geral
Prof. Me. Newton Carlos Will – Coordenador Local

Índice/Programa em Sessões Técnicas

Análise do Framework HoneyComb sob a perspectiva da Engenharia de Requisitos

Dirlene Kosvoski, Lis Ângela De Bortoli

Instituto Federal do Rio Grande do Sul - Campus Sertão - Rodovia RS 135, Km 25 - Distrito Eng. Luiz Englert - CEP: 99170-000 - Sertão-RS

dirlenekosvoski123@gmail.com,lis.debortoli@sertao.ifrs.edu.br

Abstract. *The objective of this article is to present an analysis of the HoneyComb framework from the point of view of requirements engineering in order to verify the characteristics necessary to a communication tool to support requirements engineering based on social networks. The results of the comparison demonstrate that there is a 78% adequacy of the Honeycomb elements with the characteristics of requirements engineering.*

Resumo. *O objetivo deste artigo é apresentar uma análise do framework HoneyComb sob a perspectiva da engenharia de requisitos, a fim de verificar as características necessárias a uma ferramenta de comunicação, para apoiar a engenharia de requisitos, baseada em redes sociais. Os resultados do comparativo demonstram que há uma adequação de 78% dos elementos do Honeycomb com as características da engenharia de requisitos.*

1. Introdução

A Engenharia de Requisitos (ER) é o primeiro passo no desenvolvimento de software, independente do modelo de desenvolvimento adotado. Esta atividade envolve vários grupos de pessoas, como analistas de sistemas, programadores, clientes, usuários e gerentes de projetos, conhecidos como *stakeholders*. Segundo Sommerville (2011), ER é o processo de descobrir, analisar, documentar e verificar requisitos e restrições do sistema.

Por envolver pessoas com diferentes conhecimentos e culturas, o processo da ER é visto como uma estrutura social, onde diferentes grupos de pessoas se comunicam em torno de um único objetivo: definir requisitos. Várias são as técnicas adotadas na descoberta ou elicitação de requisitos, tais como: entrevistas, questionários, reuniões, observação, cenários, casos de uso e abordagens baseadas em etnografia. É um processo de estreita colaboração entre desenvolvedores e usuários/clientes e existe muita comunicação envolvida. Esta comunicação pode ser permeada do que se chama de ruídos, ou seja, o que um quer dizer não é exatamente o que os outros vão entender.

Desta forma, entende-se que uma ferramenta a ser desenvolvida para apoiar esta atividade se caracteriza como um software social ou colaborativo. Numa tentativa de entender melhor a natureza e a estrutura dos softwares sociais, Smith (2007) criou o *framework HoneyComb*, um conjunto de sete elementos que representam os aspectos fundamentais deste tipo de software.

Sendo assim, o objetivo principal deste artigo é apresentar uma análise do *framework HoneyComb,* sob a perspectiva da engenharia de requisitos, a fim de verificar as características necessárias a uma ferramenta de comunicação, para apoiar a

engenharia de requisitos, baseada em redes sociais.

A seção 2 deste artigo aborda a engenharia de requisitos e seus problemas. A seção 3 apresenta o *framework HoneyComb* e, na seção 4, a sua relação com a engenharia de requisitos. Por fim, apresentam-se as considerações finais e as referências utilizadas.

2. Engenharia de requisitos

Segundo Sommerville (2011), os requisitos de um sistema são as descrições do que o sistema deve fazer, os serviços que oferece e as restrições ao seu funcionamento. O glossário de engenharia de software do IEEE (1990), define requisito como: (1). uma condição ou capacidade necessitada por um usuário para resolver um problema ou alcançar um objetivo; (2) a condição ou capacidade que deve ser satisfeita ou possuída por um sistema ou componente de sistema para satisfazer um contrato, um padrão, especificação, ou outros documentos impostos formalmente; uma representação documentada de uma condição ou capacidade como em (1) e (2). Para Pressman (2016), entender os requisitos de um sistema está entre as tarefas mais difíceis enfrentadas por um engenheiro de software. É complicado estabelecer os requisitos principalmente porque existem várias pessoas envolvidas direta ou indiretamente com essa atividade, os *stakeholders*. Estes, possuem culturas diferentes e falam linguagens diferentes; cada um é especialista em sua área, causando problemas de comunicação.

A engenharia de requisitos é a subárea da engenharia de software que procura sistematizar o processo de definição de requisitos. A primeira atividade da engenharia de requisitos é a descoberta, também chamada de levantamento, elicitação e aquisição. Esta atividade é colaborativa e realizada em grupo, onde os envolvidos trabalham juntos para conhecer o problema e definir uma solução. Segundo Sommerville (1997), na elicitação de requisitos a equipe de desenvolvimento de software trabalha junto com os clientes e usuários finais a fim de descobrir como o domínio de aplicação e os serviços do sistema estão organizados, qual o desempenho esperado, quais as restrições de hardware e assim por diante. Ainda conforme o autor, este processo não envolve simplesmente perguntar o que se quer do software; é necessário uma análise criteriosa da organização, do domínio da aplicação e de como realmente o sistema deve ser utilizado.

Além disso, normalmente o funcionamento das organizações é complexo, baseado em funções pouco estruturadas, com fluxos de informação pouco claros. Outro fator que dificulta a atividade é que o comportamento humano é dinâmico, difícil de definir e entender. As pessoas muitas vezes apresentam resistência a mudanças e acabam omitindo informações importantes. Ainda, os problemas são dinâmicos, ou seja, mudam com frequência; o que hoje é uma regra, amanhã pode não ser mais. Várias técnicas têm sido utilizadas para levantamento de requisitos como reuniões, entrevistas, questionários, observações, etnografia, cenários, etc. Todas elas possuem vantagens e desvantagens e são adequadas à determinadas situações.

3. Software Social e o Framework HoneyComb

Shirky (2003) define softwares sociais como sendo aqueles que suportam a interação em grupo. Klamma et al (2007) entendem software social como ferramentas e ambientes que suportam atividades em redes sociais digitais. Chatti et al (2007) compreendem

software social como ferramentas que aumentam as habilidades sociais e colaborativa das pessoas, facilitando a conexão social e a troca de informações.

Smith (2007), por sua vez, entende software social como software que permite às pessoas se conectarem por meio de uma comunicação mediada por computador. Um exemplo de software social muito utilizado atualmente são as redes sociais virtuais. Para Wassermann e Faust (apud Recuero, 2009) rede social é definida como um conjunto de dois elementos: atores ou nós da rede, que são as pessoas, instituições ou grupos; conexões, que são as interações ou laços sociais. Para Recuero (2009) uma rede, é uma metáfora para observar os padrões de conexão de um grupo social, a partir das conexões estabelecidas entre os diversos atores. A abordagem de rede tem, assim, seu foco na estrutura social, onde não é possível isolar os atores sociais nem sequer suas conexões.

O *framework HoneyComb* (Smith, 2007) é um conjunto de sete elementos que representam os aspectos fundamentais de softwares sociais. Conforme Figura 1, o elemento identidade refere-se a uma identificação pessoal única dentro do sistema, ou seja, a expressão de características de personalidade de uma pessoa (quem é a pessoa no espaço e no tempo).

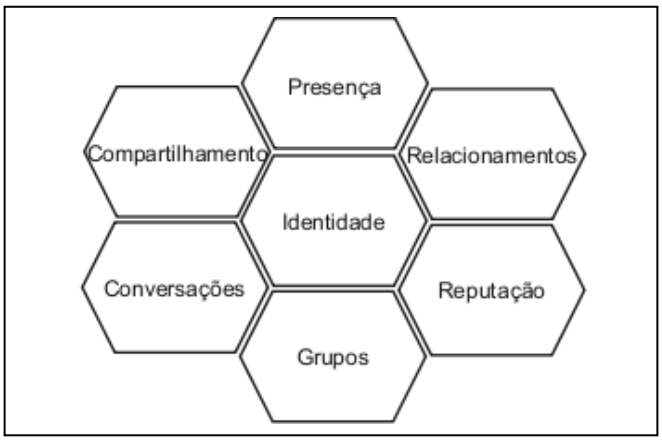


Figura 1. Conjunto de elementos do *HoneyComb Framework*

Fonte: Smith, 2007

A ideia principal é ter o elemento identidade como central em qualquer software social combinando com a implantação dos demais elementos. Assim como a presença e os relacionamentos, que referem-se a recursos que permitam saber se um usuário está disponível no sistema, compartilhando o mesmo tempo e espaço, e representar como os usuários estão conectados entre si. A reputação significa algum meio ou conjunto de recursos para se conhecer o status de outras pessoas no sistema, saber o que as pessoas pensam sobre outras.

Grupos são os recursos que promovem a formação de comunidades que compartilham interesses, preferências, ideias, opiniões, e concentram as conversações, recursos pelos quais as pessoas podem se comunicar no sistema. Além disso, os compartilhamentos são funcionalidades que permitem que os participantes compartilhem algo.

4. O *Framework HoneyComb* e a Engenharia de Requisitos

Vários autores embasaram o estudo na área de engenharia de requisitos, dentre eles destacam-se Sommerville (1997 e 2011), Pressman (2016), Leite (2003) e Thayer (1997). Após estudo dos conceitos, as autoras definiram dez características que representam essa atividade:

- Trabalho Coletivo: na engenharia de requisitos o trabalho é coletivo, ou seja, é necessário buscar informações de várias fontes de informação. As principais fontes de informação são as pessoas (*stakeholders*).
- Negociação: é preciso negociar para definir requisitos. O objetivo é que os *stakeholders* alcancem um consenso.
- Conflito: é possível que a negociação gere conflito, pois diferentes pontos de vista são expostos.
- Colaboração: os *stakeholders* trabalham numa base de igualdade e de ajuda mútua, de modo a aprofundarem reciprocamente o seu conhecimento.
- Comunicação: exposição de ideias e pontos de vista.
- Registro: todos os requisitos devem ser documentados.
- Diferença cultural e de linguagem: cada *stakeholder* tem uma bagagem cultural diferente. Cada domínio de aplicação usa termos próprios.
- Geração de ideias: produção de ideias para o novo sistema.
- Compartilhamento: os *stakeholders* compartilham informações, como documentos, leis, procedimentos, etc.
- Moderação: necessário para gerenciar conflitos entre *stakeholders* na negociação.

Tabela 1. Relação das características da ER e do *HoneyComb Framework*

		HoneyComb Framework						
		Identidade	Presença	Relacionamento	Reputação	Grupos	Conversação	Compartilhamento
Características ER	Trabalho coletivo	MI	I	NA	MI	MI	MI	MI
	Negociação	MI	MI	NA	MI	MI	MI	MI
	Colaboração	MI	PI	NA	MI	MI	MI	MI
	Comunicação	I	I	NA	MI	MI	MI	MI
	Registro	NA	NA	NA	NA	MI	I	MI
	Diferença cultural	MI	NA	NA	MI	MI	MI	MI
	Geração de ideias	I	NA	NA	MI	MI	MI	MI
	Conflito	I	I	NA	I	PI	MI	I
	Compartilhamento	I	PI	NA	MI	MI	MI	MI
	Moderação	I	I	NA	MI	MI	I	MI

Com o intuito de analisar a adequação do *framework Honeycomb* com a engenharia de requisitos foi elaborado um comparativo, conforme Tabela 1. Foram consideradas as características da engenharia de requisitos citadas acima e, após buscou-se analisar o grau de importância dos elementos do *framework Honeycomb* para cada uma delas.

Utilizou-se a seguinte avaliação: MI (Muito Importante): quando a relação do

elemento com a característica é essencial; I (Importante): quando a relação do elemento com a característica é relevante; PI (Pouco Importante): quando a relação do elemento com a característica é pouco relevante; NA (Não se aplica): quando não há relação do elemento com a característica.

Analisando em termos gerais, pode-se observar, conforme o gráfico (Figura 2), a compatibilidade das características do *HoneyComb framework* com as características da engenharia de requisitos. É possível observar que 55,7% das características do *framework* são consideradas muito importantes na adequação com a engenharia de requisitos, em sua maior parte relacionadas ao compartilhamento, conversação, grupos e reputação. Já 18,6% das características foram consideradas como importantes, relacionadas principalmente à identidade e presença. Somente 4,3% das características são de pouca importância, relacionadas à presença e aos grupos. As características não aplicáveis à engenharia de requisitos (21,4%), estão relacionadas praticamente ao relacionamento, com algumas exceções relacionadas à presença e identidade.

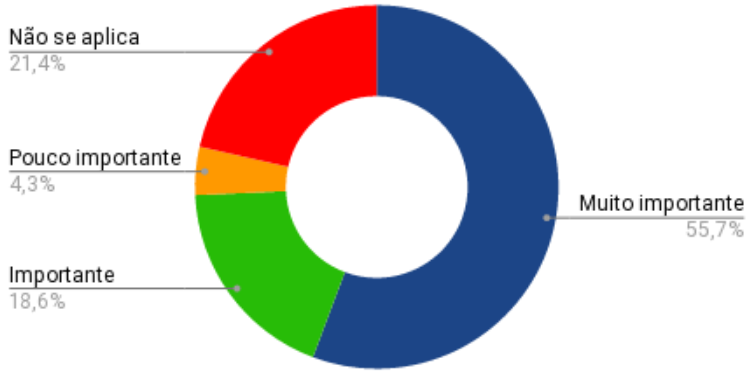


Figura 2. Compatibilidade das características do *HoneyComb Framework* com a engenharia de requisitos

Analisando individualmente cada elemento do *framework*, obteve-se os seguintes resultados:

- Identidade: aparece como importante em 50% das características da ER, 40% como muito importante e apenas 10% não se aplica.
- Presença: aparece como importante em 40% das características da ER, 10% como muito importante, 20% como pouco importante e 30% não se aplica.
- Relacionamento: não se aplica nas características da ER.
- Reputação: aparece como muito importante em 80% das características da ER, 10% como importante e em 10% não se aplica.
- Grupos: aparece como muito importante em 90% das características da ER e apenas 10% têm pouca importância.
- Conversação: aparece como muito importante em 80% das características da ER e em 20% como importante.
- Compartilhamento: aparece como muito importante em 90% das características da ER e 10% como importante.

A seguir uma análise das característica da ER com relação ao *framework.*

4.1. Trabalho Coletivo e Negociação

A definição de requisitos é um processo que essencialmente necessita de comunicação, onde cada um julga a melhor forma de se comunicar, e através da qual se dá a compreensão entre os *stakeholders*. O trabalho coletivo se baseia nisso com todos os processos realizados coletivamente entre os envolvidos, buscando diferentes fontes de informação. No trabalho coletivo é muito importante saber as características das pessoas envolvidas para que todos se conheçam, e isso será facilitado pela existência de um grupo virtual. É importante a presença dos envolvidos para que seja possível realizar discussões sobre o tema e também é muito importante no momento da tomada de decisões saber o conhecimento dos envolvidos sobre determinado assunto.

A conversação é muito importante porque quanto mais canais de comunicação existirem entre os envolvidos melhor será a troca de informações. O compartilhamento é muito importante porque no trabalho coletivo será necessário compartilhar informações sobre o tema, e essas informações, muitas vezes, estarão na forma de arquivos, o que contribui também para o registro das informações. O elemento identidade é muito importante porque, em uma negociação, é necessário conhecer as pessoas com quem se está trabalhando para evitar que ocorram problemas futuros. Da mesma forma, é muito importante saber se a pessoa com quem se está negociando está presente.

Ter um grupo é muito importante pois será possível negociar os requisitos, visando consenso. É muito importante que em uma negociação tenham-se vários meios de conversação. Por fim, é muito importante que em uma negociação haja bastante compartilhamento de informações.

4.2. Colaboração e Comunicação

O elemento identidade, assim como a reputação, são muito importantes porque é preciso saber se a colaboração é de fonte confiável. Considerou-se a presença como pouco importante, pois a colaboração pode ser feita em momentos em que o grupo não esteja *online*. O fato de ter um grupo é muito importante pois, desta forma, será possível atingir todos os *stakeholders* ao mesmo tempo. É muito importante ter vários meios de conversação e que haja compartilhamento.

Para uma boa comunicação é importante saber a identificação dos envolvidos e também, é relevante para uma comunicação razoável, saber da presença dos *stakeholders*, a fim de agilizar o processo de definição de requisitos e não necessitar fazê-lo em momentos diferentes. O elemento reputação é muito importante pois na comunicação é necessário saber se a informação fornecida é de fonte confiável. É muito importante o elemento grupo, para que a comunicação flua da melhor forma possível, abrangendo ideias de todos os *stakeholders* e, ter várias formas diferenciadas e eficientes de conversação, facilita o processo. É muito importante que os *stakeholders* possam compartilhar seus arquivos de registro para facilitar a comunicação.

4.3. Registro e Diferença cultural

O elemento grupo é muito importante, pois os requisitos são definidos em um grupo e o registro é necessário para que todos possam ter acesso ao que foi definido. Já o elemento conversação é importante pois pode ser necessário trocar informações antes ou

durante o registro dos requisitos, bem como na atualização. É muito importante o compartilhamento pois pode ser necessário compartilhar arquivos antes ou durante o registro dos requisitos, bem como na atualização.

O elemento identidade é muito importante pois é preciso conhecer as características de cada *stakeholders,* para facilitar a compreensão dos envolvidos. A reputação também é muito importante, pois é preciso saber qual o conhecimento de cada *stakeholder* que está fornecendo a informação, para que seja possível tirar dúvidas sobre os termos próprios utilizados. Da mesma forma, é muito importante ter um grupo, pois a participação de vários *stakeholders* pode amenizar as diferenças de conhecimento e linguagem. Para melhor compreender os requisitos é muito importante ter várias formas de conversação disponíveis e compartilhar informações entre os *stakeholders*.

4.4. Geração de ideias e Conflito

É importante a identidade na geração de ideias pois é preciso conhecer a origem de cada *stakeholder*. A reputação é muito importante para saber qual o conhecimento do *stakeholder* que está gerando a ideia. Ter um grupo é muito importante, pois há possibilidade de surgirem diversas ideias. É muito importante ter várias formas de conversação e ter compartilhamento de informações, para facilitar a geração e troca de ideias.

O elemento identidade é importante para que seja possível identificar os envolvidos nos conflitos. A presença e a reputação são relevantes para que o conflito possa ser resolvido da melhor forma e o mais breve possível. A conversação é importante porque diversos meios podem auxiliar e agilizar a moderação, e o compartilhamento é importante porque diferentes fontes de informação podem colaborar na resolução de um conflito.

4.5. Compartilhamento e Moderação

É importante poder identificar quem está compartilhando e, é pouco importante, a necessidade de estar presente. O elemento reputação é considerado muito importante, para se saber se a fonte da informação é confiável. É muito importante compartilhar informações com todos os *stakeholders* e também, ter a conversação para que se possa discutir as informações compartilhadas. Por fim, a comunicação é relevante, pois no compartilhamento, necessita-se de diversos meios de comunicação para discutir as ideias do grupo.

É importante, para identificar os envolvidos nos conflitos, assim como a presença e a reputação, para que o conflito possa ser resolvido da melhor forma e o mais breve possível. O elemento grupo é muito importante, pois como são várias pessoas envolvidas, muitas ideias podem surgir, o que pode causar conflitos que precisam de moderação. A conversação é importante, porque diversos meios de comunicação podem auxiliar e agilizar a moderação. O compartilhamento, por sua vez, tem sua relevância, porque diferentes fontes de informação podem colaborar na resolução de conflitos.

5. Considerações finais

A engenharia de requisitos é uma etapa essencial no desenvolvimento de software,

porém, por envolver muita comunicação entre diferentes pessoas, torna-se uma atividade de difícil condução. Ferramentas de redes sociais estão em constante crescimento, sendo utilizadas em diferentes áreas. As redes sociais apresentam um espaço para comunicação e possibilitam o desenvolvimento de trabalho coletivo. Sendo assim, apresentou-se neste artigo, parte de uma pesquisa sobre a utilização de redes sociais como forma de minimizar os problemas de comunicação desta área.

Verificou-se que o *Framework Honeycomb* se adequa em 78% de seus elementos, com as características da engenharia de requisitos, sendo 55% consideradas como muito importantes, 18% como importantes, 4% como pouco importantes e 21% não se aplicam. Estes resultados mostram que os elementos do *framework* (exceto o relacionamento), devem ser considerados no desenvolvimento do protótipo. Porém, outros elementos deverão ser acrescentados, em virtude das particularidades da engenharia de requisitos como, por exemplo, o consenso, a resolução de conflitos, a moderação e a negociação.

Na continuidade, pretende-se desenvolver um protótipo, contemplando as características da engenharia de requisitos com base nos conceitos de redes sociais, software social e colaborativo, bem como os elementos do *Framework Honeycomb*.

6. Referências

Chatti M. A.; Jarke, M.; Wilke D. F. (2007), "The future of e-learning: a shift to knowledge networking and social software. In: *Journal of Knowledge and Learning*, v.3, n. 4-5, p. 404-420.

IEEE Standard Glossary of Software Engeneering Terminology (1990) http://www.mit.jyu.fi/ope/kurssit/TIES462/Materiaalit/IEEE_SoftwareEngGlossary.pdf, Setembro de 2016.

Klamma, R et al. (2007), "Social Software for life-long learning. Educational Technology and Society!, v-10, n-3, p 72-93.

Leite, J. C. P. , Doorn, J. H. (2003). Perspectives on Software Requirements. Springer.

Pressman, R. (2016). Engenharia de Software - uma abordagem profissional. McGraw-Hill.

Recuero, R. (2009). Redes sociais na internet. Sulina.

Shirky, C. (2003). "A group is its own worst enemy: social structure and social software". In: O´Really Emergin Technology Conference, Santa Clara. Proceedings… Santa Clara: O´Really.

Smith, G. (2007) "Social Software Building Blocks", http://nform.com/publications/social-software-building-block, Acesso em Setembro de 2016.

Sommerville, I. (2011). Engenharia de Software. Addison Wesley.

Sommerville, I., Sawyer, P. (1997). Requirements Engineering - A good practice guide. John Wiley & Sons.

Thayer, R., Dorfman, M. (1997). Software Requirements Engineering. IEEE Computer Society Press.

Levantamento de tecnologias para identificação animais domésticos acoplado ao ciclo de vida de um Sistema *Web*

Tatiana Tozzi, Daniel Fernando Anderle, Rodrigo Ramos Nogueira

Instituto Federal Catarinense – Campus Camboriú (IFC) – Camboriú, SC – Brasil

tatitozzi@hotmail.com, {daniel.anderle, rodrigo.nogueira}@ifc.edu.br

Abstract. *This article addresses the steps of identifying existing technologies that can be used to identify lost or abandoned animals. A questionnaire was carried out to identify which technologies are used to identify, locate, rescue and adopt domestic animals. From the results obtained in the research, the technologies were identified, so the research reports the main technologies used so far and is moving towards the development of a proposal for a system to improve and expand the dissemination of lost or adoption.*

Resumo. *Este artigo aborda as etapas da identificação de tecnologias existentes atualmente que possam ser utilizadas na identificação de animais perdidos ou abandonados. Para isso foi realizado um questionário buscando identificar quais as tecnologias são utilizadas para identificação, localização, resgate e adoção de animais domésticos. A partir dos resultados obtidos na pesquisa foram identificadas as tecnologias, deste modo a pesquisa reporta as principais tecnologias utilizadas até o momento e caminha para o desenvolvimento de uma proposta de um sistema para melhorar e ampliar a divulgação de animais domésticos que se encontrem perdidos ou para adoção.*

Palavras-chave*: Identificação de tecnologias. Sistema Web. Levantamento de tecnologias. Proteção animal.*

1. Introdução

Os animais domésticos fazem parte do dia a dia dos seres humanos desde os primórdios, e são representados em maioria pelas espécies felina e canina, gatos e cachorros respectivamente. Segundo o IBGE [2010], o Brasil possui a quarta maior população mundial de animais domésticos, com 132,4 milhões de animais domésticos, sendo eles cães, gatos, aves e peixes. A Organização Mundial da Saúde (OMS), conforme pesquisa realizada em 2014 estima-se que o Brasil possui 30 milhões de animais abandonados, sendo 10 milhões de gatos e 20 milhões de cachorros [ANDA, 2014].

Atualmente à internet e as tecnologias são a principal fonte para divulgar eventos de adoção de animais e para anunciar animais encontrados e perdidos. Esta pesquisa vem sendo construída com o objetivo responder a seguinte pergunta de pesquisa: "Por meio de um sistema *web* é possível diminuir o abandono animal e melhorar a divulgação de animais para adoção na região da AMFRI[1]". Para atingir o objetivo esta pesquisa identifica as tecnologias utilizadas atualmente que possam ser usadas para

[1] Associação dos Municípios da Foz do Rio Itajaí – Santa Catarina [AMFRI, 2018].

auxiliar na identificação e divulgação de animais domésticos, tendo como base o seguinte roteiro:

- Apresentar os resultados através de uma Pesquisa de opinião;
- Descrever as tecnologias encontradas;
- Desenvolver um projeto para o desenvolvimento de um Sistema *Web*;
- Testar a viabilidade do modelo proposto junto às ONGs e protetores independentes.

2. Materiais e Métodos

Esta pesquisa se classifica quanto à natureza aplicada e tecnológica, em questão aos objetivos, como exploratória e quanto aos procedimentos, bibliográfica [Figura 1 – A]. Este trabalho foi dividido em seis fases, sendo que as mesmas estão relacionadas com os objetivos propostos desta pesquisa [Figura 1 – B].

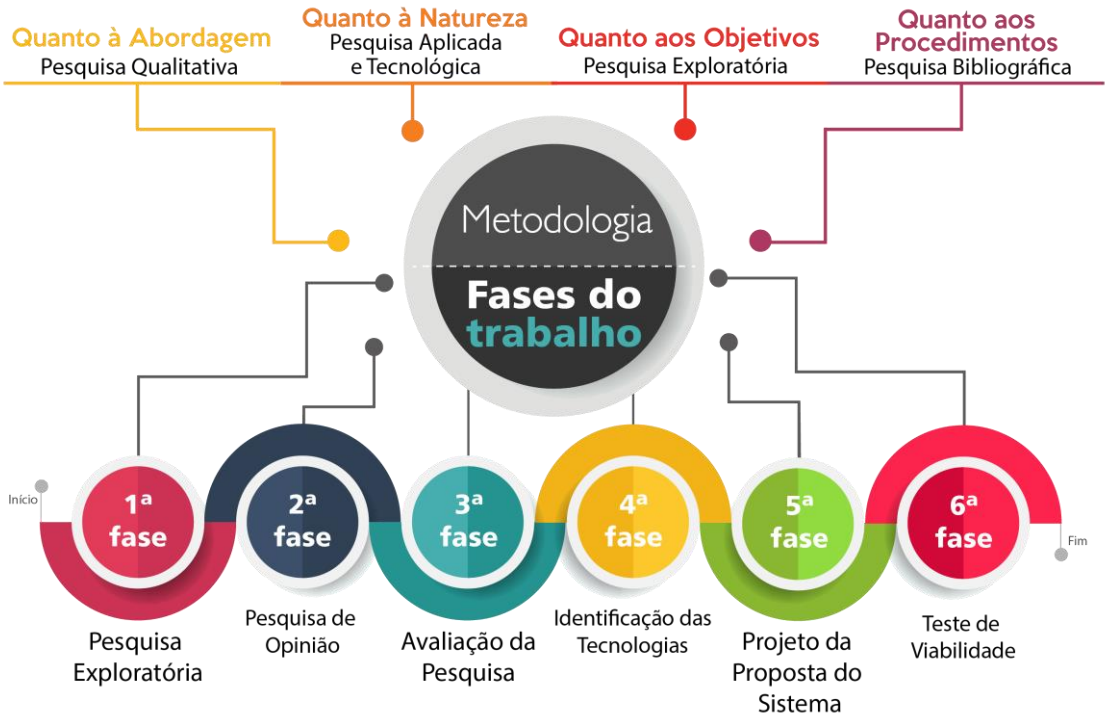


Figura 1 – Fases de desenvolvimento do trabalho

3. Resultados Parciais

Até o presente momento foram desenvolvidas quatro fases do trabalho, as quais serão apresentadas resumidamente a seguir:

A primeira fase – **Pesquisa exploratória** buscou-se identificar os principais conceitos abordados durante o decorrer da pesquisa, os quais são: animais domésticos, posse responsável, protetores independentes, ONG, centro de zoonoses, resgate de animais, abandono de animais. Como parte desta fase foram selecionados seis[2] trabalhos

[2] O trabalho/tecnologia identificada por Coimbra [2017], é apresentado na 4ª fase, pois, o mesmo também foi identificado na pesquisa de opinião.

científicos que tem como objetivo auxiliar os animais domésticos.

Carpanezi, et al. [2016] desenvolveram um aplicativo, onde foram cadastrados os animais para adoção e quem tivesse interesse em adotar algum dos animais cadastrados poderia conhecê-lo através de suas características e fotos. Já Silva, et al. [2017] teve como proposta à implementação de um sistema informatizado para registro de controle dos animais acolhidos pelo Centro de Controle de Zoonoses. O Projeto Adoção Animal de Evangelista et al. [2015] teve como pretensão ampliar a divulgação dos animais abandonados que vivem em abrigos utilizando mídias sociais [*Facebook* e *Blogger*] incentivando a adoção de animais.

Menezes Filho e Souza [2017] tiveram como objetivo o desenvolver uma ferramenta para registro e identificação de animais de companhia. Para que fosse atingido o objetivo foi utilizado tecnologias para o desenvolvimento web, com a intenção de criar uma base de dados para as prefeituras armazenarem informações sobre a população de animais. Donatti [2017] desenvolveu dois sistemas utilizando um protocolo baseado no *ZigBee* e a tecnologia *Global Positioning System* – GPS para a coleta de dados referentes ao posicionamento global dos animais (bovinos), identificando a longitude e latitude do animal.

A segunda fase – **Pesquisa de Opinião** constitui na realização de uma pesquisa com os moradores da região da AMFRI, através de um questionário composto por 24 perguntas utilizando o *Google Forms*. As perguntas tinham como objetivo identificar à faixa etária, o sexo, a quantidade de animais e a espécie de animais possuem, se os participantes da pesquisa fazem parte de alguma ONG de proteção animal ou se atua como protetor independente, quais tecnologias foram utilizadas para divulgar animais abandonados ou para adoção entre outras questões. Porém o principal objetivo desta fase foi testar a viabilidade para desenvolver as próximas fases deste trabalho, e posteriormente o desenvolvimento do sistema.

Figura 2 - Infográfico com os dados principais da pesquisa

A terceira fase – **Avaliação da pesquisa** foram apresentados os resultados da

fase anterior, através de um infográfico[3] (figura 2), são expostos os resultados da pesquisa de opinião.

Após duas semanas de divulgação e aplicação da pesquisa de opinião, 100 pessoas responderam à pesquisa. As cidades que mais participaram da pesquisa foram as cidades: Camboriú com 45% e Balneário Camboriú com 32%. A maioria dos participantes tem entre 26 a 31 anos (24%), sendo 69 % das participações foram de mulheres e 31 % homens. Destes participantes 89% afirmaram ter animais domésticos.
Dos participantes, 71% já utilizaram alguma tecnologia para auxiliar no resgate e divulgação de animais domésticos. A pergunta em seguida continha várias tecnologias, a qual os participantes poderiam selecionar quais já usaram, sendo a mais identificada as Redes Sociais. Questionados se gostariam de conhecer as tecnologias citadas[4] na pesquisa 93% responderam que sim, uma vez que 61% dos participantes não conheciam as tecnologias citadas.

Questionados se já realizaram algum resgate de animais, 79% dos participantes responderam que si, sendo a maioria dos resgates realizados de cães e em seguida gatos. Por meio da 19ª pergunta, descobrimos que 77% dos participantes utilizam redes sociais para auxiliar na divulgação de animais perdidos, abandonados ou para adoção. Sendo o *Facebook*, a rede social mais utilizada para este fim e as publicações desses anúncios são em maioria feitas na própria linha do tempo dos participantes. 98% dos participantes acham que um Sistema *Web* poderia melhorar e ampliar a divulgação de animais perdidos ou para adoção.

A quarta fase (**Identificação de tecnologias**) foram realizadas a identificação das tecnologias abordadas na pesquisa de opinião, sendo elas conforme apresentadas pela tabela 1, sendo que na coluna 1 são identificados os tipos de tecnologia e na coluna 2 a descrição das mesmas.

Tabela 1 - Tecnologias identificadas

Tecnologia	Descrição
Microchip RFID (*Radio-Frequency IDentification*)	É um método de identificação automática por meio de sinais de rádio, onde são recuperados e armazenados dados remotamente através de um dispositivo de *tags* RFID, tal dispositivo é implantado sobre a pele do animal;
Microchip NFC (*Near Fiel Communication*)	É uma tecnologia que possibilita a troca de informações e dados entre dispositivos assim como o RFID, porém, para acessar as informações no *microchip* basta possuir um *smartphone* compatível com essa tecnologia e se aproximar 10 centímetros do animal [COIMBRA, 2017], já o RFID necessita de um leitor específico para este fim;
Coleira com *qrCode* (*Quick Response Code*)	Consiste em uma coleira com uma medalha de identificação com *qrCode*, através da leitura do *qrCode* é possível acessar a página do animal, a qual contém informações de contato do tutor, telefone do médico veterinário, fotos e informações médicas;

[3] Questionário da pesquisa da opinião e resultados. Disponível em: <https://tatitozzi.github.io/resultadopesquisa.github.io/>.
[4] *Microchip* - RFID, *Microchip* - NFC, Coleira com *qrCode*, Coleira com *Tag*, Aplicativo de busca, Aplicativo de Identificação, Redes sociais.

Tecnologia	**Descrição**
Coleira com *tag*	É uma alternativa para utilização do *microchip* sem que este seja implantado no animal, a *tag* contendo os dados do animal (o código de identificação) é colocada na coleira do animal;
Aplicativos de busca	Podem ser utilizados para cadastrar informações do animal e dados de contato do tutor;
Aplicativos de identificação	Promovem a identificação do animal através de reconhecimento facial, utilizando a tecnologia de comparação de imagens (visão computacional e inteligência artificial).
Redes sociais	São grandes aliadas na procura e divulgação de animais, em fevereiro de 2018 foi criada da rede social *Puppyfi*, com o principal objetivo auxiliar os animais, assim auxiliando tutores a encontrarem seus animais desaparecidos.

Atualmente estamos na 5º fase (**Projeto da Proposta do Sistema),** como resultado parcial desta fase, foi realizado o levantamento dos requisitos do sistema (funcional e não-funcional).

Segundo Wazlawick [2011], "[...] levantamento de requisitos é o processo de descobrir quais são as funções que o sistema deve realizar e quais são as restrições que existem sobre essas funções". Através do levantamento de requisitos são identificadas as funções que o sistema deverá realizar e quais serão as restrições que devem existir em conjunto com as funções identificadas, essa fase é uma descoberta, na qual busca-se listar a maior quantidade de funções e restrições [WAZLAWICK, 2011].

Os requisitos funcionais tratam das funcionalidades que o sistema deverá possuir ou serviços que se espera que o sistema venha a realizar. Para Sommerville [2007], os requisitos funcionais são:

> Os requisitos funcionais de um sistema descrevem o que o sistema deve fazer. Esses requisitos dependem do tipo de software que está sendo desenvolvido, dos usuários a que o software se destina e da abordagem geral considerada pela organização ao redigir os requisitos. Quando expressos como requisitos de usuários eles são geralmente descritos de forma bastante abstrata.

Através do quadro 1 demostra-se os requisitos elencados para o sistema.

Quadro 1 - Requisitos Funcionais

RF01	O sistema deve permitir o cadastro de usuário utilizando informações básicas (nome, telefone, e-mail, senha);
RF02	O sistema deve permitir que o usuário faça *login* utilizando o e-mail e senha fornecidos no ato do cadastro, passando a ter acesso a funcionalidades de acesso restrito;
RF03	O sistema deve permitir a edição da conta do usuário (troca de senha e telefone);
RF04	O administrador do sistema pode adicionar e remover usuários ou trocar o nível de acesso do usuário;
RF05	O usuário pode publicar anúncios com a características do animal que venha ter encontrado e inserir fotos do mesmo;
RF06	O usuário pode publicar anúncios com a características do animal que venha ter perdido e inserir fotos do mesmo;

RF07	Usuários não identificados (sem cadastro) podem visualizar as publicações do sistema, tendo acesso as características do animal e fotos;
RF08	O usuário pode gerar um panfleto do seu anúncio;
RF09	Usuários não identificados (sem cadastro) podem visualizar apenas o nome do autor do anúncio;
RF10	O contato entre anunciante e visitante ocorrerá através de um *chat*, assim o mesmo só poderá ser contatado pelo sistema;
RF11	O sistema notificará o usuário quando o mesmo tiver uma mensagem de um visitante;
RF12	O usuário (anunciante) pode compartilhar o anúncio, juntamente com as fotos, nas redes sociais *Facebook*, *Twitter*, *Instagram*, ampliando a propagação do anúncio;
RF13	O anúncio pode expirar (30 dias) ou pode ser cancelado ou publicado novamente;
RF14	O administrador poderá cadastrar novos usuários;
RF15	O sistema deve inserir uma marca d'agua nas fotos submetida para os anúncios;
RF16	O usuário poderá ter acesso ao sistema utilizando as credencias do *Google* e *Facebook* (*oauth2*);
RF17	O usuário poderá identificar em um mapa a localização do animal (desaparecido/localizado);
RF18	O administrador terá acesso a uma lista de usuários e seu nível de acesso;
RF19	O usuário caso se esqueça de sua senha, poderá solicitar a recuperação através de uma página própria, a qual o mesmo deverá fornecer seu e-mail. Em seguida será enviado para o e-mail do usuário um link para criar outra senha de acesso;
RF20	O *login* e cadastro serão validados através de uma ferramenta como o *reCAPTCHA*;
RF21	O usuário pode a qualquer momento excluir sua conta do sistema;
RF22	Os visitantes terão acesso aos anúncios na página principal do sistema;
RF23	O sistema deve permitir a conexão simultânea de vários usuários.

Os requisitos não-funcionais, não fazem parte das características do sistema, assim esses requisitos não estão ligados as funções que o sistema deverá realizar.

Segundo Engholm [2010],

> Requisitos não-funcionais especificam requisitos não abrangidos pelos requisitos funcionais. Eles especificam critérios que avaliam o funcionamento do sistema, ao invés das funcionalidades disponibilizadas aos usuários. Típicos requisitos não-funcionais incluem disponibilidade, desempenho, tempo de resposta e *throughput*.

No quadro 2 apresenta-se os requisitos não-funcionais elencados para o sistema.

Quadro 2 - Requisitos Não-funcionais

RNF01	O sistema será construído para rodar em ambiente *web*;
RNF02	Para acessar o sistema será necessário estar conectado à internet;
RNF03	O sistema terá uma interface simples e amigável;
RNF04	O sistema poderá ser acessado em *smartphones*, *tablets* e computadores;
RNF05	O sistema deverá ser responsivo;

RNF06	As mensagens enviadas via chat do sistema devem ser entregues ao destinatário;
RNF07	O sistema deve garantir que os dados sejam protegidos de acesso não autorizado;
RNF08	O sistema deve ser seguro contendo senha criptografada e *reCAPTCHA* (*login* e cadastro);
RNF09	O sistema deverá ser confiável;
RNF10	O sistema deve ter uma elevada taxa de disponibilidade, estando operacional 99,7% do tempo;
RNF11	O SGBD utilizado será o *PostgreSQL* 8.2.4 ou superior;
RNF12	O Sistema será feito HTML, CSS e PHP.

O ator do sistema segundo Pfleeger [2004] é "[...] uma entidade que fornece ou recebe dados". Um ator pode representar um usuário real, um dispositivo ou outro sistema, o qual venha interagir com o sistema. Os atores que fazem parte do sistema são identificados através do quadro 3.

Quadro 3 - Atores

Nome	**Descrição**
Administrador	O administrador do sistema pode inserir e remover usuários, assim como anúncios do sistema sem solicitar permissão para tal ação.
Usuário (anunciante)	O usuário pode inserir anúncios com foto do animal desaparecido/encontrado.
Visitante	O visitante pode acessar a página do sistema, a qual contará com os anúncios publicados pelos anunciantes.

4. Considerações Finais e Trabalhos Futuros

Neste trabalho foram apresentadas as fases desenvolvidas na pesquisa, a qual trata-se como parte do Trabalho de Conclusão de Curso (TCC). Sendo que atualmente estamos desenvolvendo a quinta fase (projeto de sistema *Web*) e será desenvolvido os casos de uso, a modelagem do banco de dados e a prototipação das telas do sistema.

Como trabalhos futuros pretende-se dar seguimento à última fase (6ª) a qual visa em testar a viabilidade da aplicação proposta junto às ONGs, Centros de Zoonoses e Protetores Independentes, identificando se nossa proposta vem atender necessidades para a divulgação, identificação, localização e adoção de animais domésticos.

Referências

AMFRI. **Associação dos Municípios da Foz do Rio Itajaí**. Disponível em: <www.amfri.org.br/>. Acesso em: 18 mai. 2018

ANDA. **Brasil tem 30 milhões de animais abandonados**. 2014. Disponível em: <https://goo.gl/sjojsT>. Acesso em: 14 set. 2017.

CARPANEZI, et al. **Desenvolvimento de um aplicativo mobile para adoção de animais de estimação**. 2016. Disponível em: <https://goo.gl/vkRKJr>. Acesso em: 28 fev. 2018.

COIMBRA, D. da S. **O Uso da Tecnologia NFC na Identificação PET**. 2016. Disponível em: <https://goo.gl/gHSDKa>. Acesso em 28 fev. 2018.

DONATTI. R. N. **Desenvolvimento de um sistema de monitoramento de animais, utilizando rede de sensores sem fio, baseado no protocolo ZigBee e tecnologia GPS**. 2017. Disponível em: <https://goo.gl/fTTdQZ>. Acesso em: 10 mai. 2018.

ENGHOLM JUNIOR, Hélio. Engenharia de Software na prática. São Paulo: Novatec Editora, 2010.

EVANGELISTA, et al. **Projeto adoção animal**: Incentivando a prática da adoção de Cães e Gatos Abandonados – Resultados preliminares. 2015. Acesso em: <https://goo.gl/7v87U9>. Acesso em 10 mai. 2018.

IBGE. **Um panorama da saúde no Brasil**: acesso e utilização dos serviços, condições de saúde e fatores de risco e proteção à saúde 2008. Rio de Janeiro: IBGE; 2010. Disponível em: <https://goo.gl/uPfejo>. Acesso em: 13 nov. 2017.

MENEZES FILHO, et al. **Registro geral de Animais** (RGA): um sistema para o registro e identificação de animais de companhia. 2017. Disponível em: <https://goo.gl/QYmf1w>. Acesso em: 12 abr. 2018.

PFEEGER, Shari Lawrence. Engenharia de Software: teoria e prática. 2 ed. São Paulo: Prentice Hall, 2004.

SILVA, et al. **Sistema Informatizado para o Centro de Controle de Zoonoses**. 2017. Disponível em: <https://goo.gl/Zpm7f9>. Acesso em: 10 mar. 2018.

SOMMERVILLE, Ian. Engenharia de Software. 8ª ed. São Paulo: Pearson Addison-Wesley, 2007.

WAZLAWICK, Raul Sidnei. Análise e projeto de sistemas de informação orientados a objetos. 2. ed. Rio de Janiero: Elsevier, 2011.

Aplicação de uma Survey para Identificação de Problemas no Processo de Coleta e Análise de Requisitos de Sistema

Luana Belusso, Pedro H. A. Machado, Felipe B. Ribeiro, Jackson G. Schimit

Coordenadoria do Curso de Engenharia de Software
Universidade Tecnológica Federal do Paraná (UTFPR) – Dois Vizinhos, PR – Brazil

luanabelusso@alunos.utfpr.edu.br, pedrohmachado@utfpr.edu.br,
felrib@alunos.utfpr.edu.br, jacksonschimit@gmail.com

Abstract. *The article addresses an identification of problems in the data collection process and an analysis of software requirements, through a research done by two times consisting of the application of a questionnaire and the conduct of a research session. The discussion comprehends how the process of requirements analysis is being constructed in real scenarios and thus impact on the developed systems.*

Resumo. *Este artigo aborda a identificação de problemas no processo de coleta e análise de requisitos em empresas desenvolvedoras de software, através da realização de uma Survey composta por duas etapas que consistem na aplicação de um questionário e na realização de entrevistas com os profissionais atuantes. A discussão compreende a forma como o processo de análise de requisitos está sendo desempenhado em cenários reais e de que forma impacta nos sistemas desenvolvidos.*

1. Introdução

Tendo em vista a necessidade de desenvolver sistemas que atendam às solicitações dos clientes e proporcionem às empresas o acompanhamento do constante avanço tecnológico e das demandas do mercado, os processos de desenvolvimento de *software* evoluíram para se adaptar a essa realidade.

Embora as metodologias ágeis proponham inserir o cliente durante todo o processo de desenvolvimento de *software*, a abordagem ágil tem sido questionada quanto à definição de papéis e práticas que incluam atividades relacionadas à verificação e validação de requisitos de usabilidade e experiência de usuário, uma vez que são comumente separadas do processo de desenvolvimento de *software* (SALAH, 2011). Essas atividades são fundamentais, pois à medida que o uso de sistemas evolui, há um aumento de usuários de variados perfis e o *software* deve ser capaz de atender as suas necessidades, uma vez que a qualidade de um produto é também resultado da experiência que o usuário obtém com a sua utilização (PERES; MEIRA, 2015).

Para Nielsen (2012) diversos fatores definem a usabilidade de um sistema, dentre eles, a facilidade de aprendizado e de memorização com que o usuário interage com uma interface e a satisfação e eficiência durante a realização de uma atividade. Diante disso, tem se estudado formas de incorporar práticas que auxiliem o processo de descoberta e análise de requisitos de *software*, de forma que o sistema desenvolvido proporcione uma melhor experiência ao usuário.

Uma forma abordada na literatura para auxiliar o processo de descoberta e análise de requisitos de *software* é a utilização da técnica de prototipagem. Reconhecida pela sua eficiência, a prototipagem através dos seus diversos níveis de abrangência, se adapta ao contexto dos processos em que é aplicada. Além disso, sua utilização possibilita explorar soluções alternativas, realizar testes de usabilidade e obter *feedbacks* dos usuários antes do início do desenvolvimento do *software* (SURANTO, 2015).

As atividades de levantamento e análise de requisitos são fundamentais para a criação de recursos que atendam as necessidades e expectativas dos usuários e evitem o desenvolvimento de recursos não utilizados, uma vez que há um investimento que não agrega valor ao produto final (VAZQUEZ; SIMÕES, 2016).

O objetivo deste trabalho é verificar a ocorrência de requisitos mal especificados ou mal interpretados e evidenciar os problemas relatados por profissionais atuantes em empresas de desenvolvimento de *software* da região Sudoeste do Paraná, de forma a avaliar como o processo de análise de requisitos desempenhado em cenários reais impacta nos sistemas desenvolvidos.

2. Fundamentação Teórica

A engenharia de requisitos é composta por um processo sistemático que envolve etapas para documentação, que serve de insumo para o desenvolvimento de sistemas. Assim, o *software* deve estar em conformidade com essas especificações, as quais podem definir desde uma funcionalidade, comportamento ou propriedade do sistema até uma restrição técnica ou de negócio (MACHADO, 2016).

O processo da engenharia de requisitos inclui quatro principais etapas que são desempenhadas de forma iterativa e se caracterizam por: (i) avaliar se o desenvolvimento do sistema é rentável na perspectiva de negócio e orçamento, e viável acerca da tecnologia empregada; (ii) realizar o levantamento e a análise dos requisitos do contexto ao qual o sistema será desenvolvido; (iii) converter os requisitos analisados em documentações formais; (iv) verificar os requisitos quanto à aderência, completude e consistência em comparação aos processos que se visam especificar (SOMMERVILLE, 2011).

As etapas de coleta e análise de requisitos se concentram em descobrir os requisitos que o sistema deve possuir para atender aos processos realizados pelos usuários. Há diversos desafios nessa etapa, como a dificuldade do *stakeholder* em demonstrar o que deseja e necessita que o sistema atenda, a compreensão do analista sob os requisitos fornecidos e a diversidade de *stakeholders* existentes (SOMMERVILLE, 211).

Assim, a etapa de validação de requisitos se concentra em verificar se os requisitos especificados atendem ao que o cliente realmente deseja. Essa etapa é fundamental porque inclui atividades que visam evitar que recursos e funcionalidades que não atendam aos usuários, sigam para o processo de desenvolvimento de *software* e gerem manutenções com custos maiores em comparação à mudança de requisitos realizada antes do desenvolvimento (SOMMERVILLE, 2011).

Na engenharia de requisitos, os protótipos auxiliam através do levantamento e da validação dos requisitos com usuários, o que possibilita a identificação de problemas e o

esclarecimento de dúvidas nas fases iniciais de desenvolvimento, gerando especificações de requisitos mais assertivas o que implica na redução de tempo e custo com manutenção de sistemas que não atendem às necessidades dos usuários (FU; BASTANI; YEN, 2008).

Os protótipos também são utilizados com foco na investigação de soluções e apoio ao desenvolvimento de interfaces de sistemas. Um estudo realizado por Suranto (2015) descreve a utilização de prototipagem de *software* de alta fidelidade para explorar funcionalidades de forma interativa e para ter uma percepção do produto final. A utilização de prototipagem de baixa fidelidade é utilizada no estudo para propor soluções de forma rápida para validações com as partes interessadas. O estudo de Laskowska et al. (2014) apresenta uma abordagem sistemática para utilização de prototipagem. A conclusão do estudo aponta que avaliações antecipadas combinadas ao desenvolvimento ágil auxiliam a entrega de produtos que atendem as expectativas dos usuários.

3. Metodologia

O presente trabalho visa avaliar a ocorrência de problemas presentes no processo de análise de requisitos de *software*, relacionados a requisitos mal especificados ou mal interpretados. Para isso, foi realizada uma *Survey* (Figura 1) utilizando questionário e entrevista como instrumentos de pesquisa. Essa pesquisa foi divulgada para 15 empresas de desenvolvimento de *software* da região Sudoeste do Paraná, de pequeno, médio e grande porte.

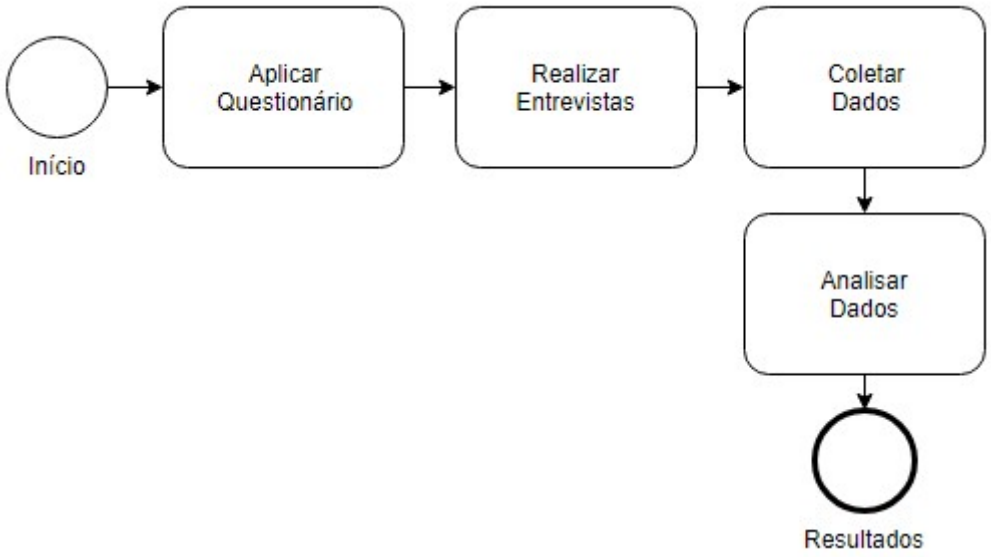


Figura 1. Esquematização da pesquisa de trabalho.
Fonte: Autoria própria (2018).

A primeira etapa da pesquisa consistiu na aplicação de um questionário, o qual foi criado em uma plataforma *online,* contendo 19 perguntas elaboradas em conjunto à especialistas na área de engenharia de requisitos e destinadas à profissionais no cargo de analista de requisitos ou que desempenham esse papel nas empresas. Dentre elas, foi adicionada uma pergunta para que os participantes pudessem registrar um contato

pessoal, caso tivessem interesse em participar da segunda etapa da pesquisa por meio da realização de entrevistas. O questionário foi divulgado através de plataformas *online* para as empresas de desenvolvimento de *software* durante o período de 30 dias. Após o período de divulgação, os dados coletados passaram por uma análise, os quais correspondem a respostas de vinte e oito pessoas, que representam seis empresas desenvolvedoras de *software*, sendo 61% analistas de requisitos, 14% desenvolvedores e os demais em cargos diversos.

Assim, a segunda etapa da pesquisa consistiu na realização de entrevistas em plataformas *online* com os profissionais que demonstraram interesse em participar da pesquisa através das respostas coletas pela aplicação do questionário. O modelo de entrevistas realizado foi semi-estruturado, assim, foram definidas perguntas essenciais para todas as empresas e no decorrer da entrevista, exploradas novas perguntas que o entrevistador julgou necessário.

O objetivo da realização das entrevistas é definido pela coleta de informações complementares as perguntas do questionário. Assim, as perguntas descritivas presentes no questionário foram aplicadas nas entrevistas, as quais duraram em média trinta minutos, foram gravadas e analisadas para a síntese dos resultados. Seis profissionais foram entrevistados, os quais representam a participação de apenas três empresas. Cinco profissionais atuam como analista de requisitos e um como testador de *software.*

4. Resultados e Discussões

Os resultados compreendem as respostas de profissionais que atuam em empresas que desenvolvem sistemas para versões *desktop*, *web* e aplicativos *móbile*, sendo três empresas de pequeno porte, uma de médio e duas de grande porte. Todas as empresas relatam a divisão dos profissionais por equipes, cada equipe possui um analista de requisitos responsável pelas demandas de desenvolvimento de *software*, as quais e caracterizam por uma manutenção ou melhoria de um sistema.

Os principais meios de coleta e análise de requisitos é por entrevistas com os clientes, protótipos de *software* e observação. Dentre os participantes, cinco possuem menos de um ano de experiência na área, doze entre um a cinco anos, oito entre seis a dez anos e três mais de dez anos.

Em análise dos resultados, 93% dos participantes relatam a ocorrência de problemas com requisitos mal especificados ou mal interpretados, os quais são evidenciados na Tabela 1. Os problemas apresentados são relatos efetivos, não houve sugestão de problemas aos participantes. As colunas “Questionário” e “Entrevista” informam se houve a ocorrência do problema entre as duas etapas da pesquisa.

A maioria dos participantes relata problemas em relação ao desenvolvimento de funcionalidades que não atendem as solicitações dos clientes, assim, essas empresas investem na produção de sistemas que possuem funcionalidades que não são utilizadas. Isso gera a necessidade de encaminhar uma nova solicitação para o desenvolvimento adaptando a funcionalidade ou criando uma nova solução. Esse retrabalho em ajustar e desenvolver funcionalidades impacta em custos e prazos de entregas, pois tem que ser absorvido pela empresa na grande maioria dos casos em que não há renegociação com o cliente.

Código	Descrição do problema	Questio-nário	Entre-vista	Quantidade evidências questionário	Quantidade evidências entrevista	Quantidade total de evidências
1	Funcionalidade desenvolvida não atende as solicitações dos clientes	X	X	10	4	14
2	Retrabalho para desenvolver a solução correta	X	X	4	4	8
3	Aumento de custos que não podem ser renegociados com os clientes / Prejuízos financeiros	X	X	4	1	5
4	Prazos não atendidos devido a especificações incorretas, notadas durante o desenvolvimento	X	X	3	2	5
5	Falha de comunicação	X		3	0	3
6	Má compreensão da necessidade por parte da equipe de desenvolvimento	X		3	0	3
7	Analista de requisitos não possui experiência para coleta e análise dos requisitos	X		2	0	2
8	Credibilidade da empresa em relação ao cliente é afetada negativamente	X	X	1	1	2

9	Implementação de recursos não utilizados pelos clientes	X		1	0	1
10	Solução desenvolvida é complexa e por isso não atende as necessidades dos clientes	X		1	0	1
11	Impacto na estimativa de tempo de desenvolvimento do recurso	X		1	0	1
12	Pouco tempo destinado à análise de requisitos	X		1	0	1
13	Troca de analista de requisitos que acompanha a solicitação antes e durante o desenvolvimento		X	0	1	1
14	Desenvolvimento de funcionalidades não aceitas por clientes diferentes		X	0	1	1
15	Má usabilidade da funcionalidade desenvolvida		X	0	1	1

Tabela 1. Evidências de problemas relacionados ao processo de análise de requisitos de *software*.
Fonte: Autoria própria (2018).

A Figura 2 apresenta as evidências de problemas coletados através da aplicação do questionário e da realização das entrevistas, a legenda do gráfico é associada ao código do problema da Tabela 1. O gráfico evidência a disparidade na ocorrência do problema 1 em comparação aos demais, nota-se o relato de problemas através das entrevistas que não foram evidenciados pelo questionário.

Além dos principais problemas abordados no questionário e nas entrevistas que se referem ao desenvolvimento de funcionalidades que não atendem a solicitação dos clientes, o retrabalho para adaptar a funcionalidade desenvolvida para que possa atender

ao que o cliente necessita, aumento de custos e o não cumprimento dos prazos estabelecidos (Figura 2).

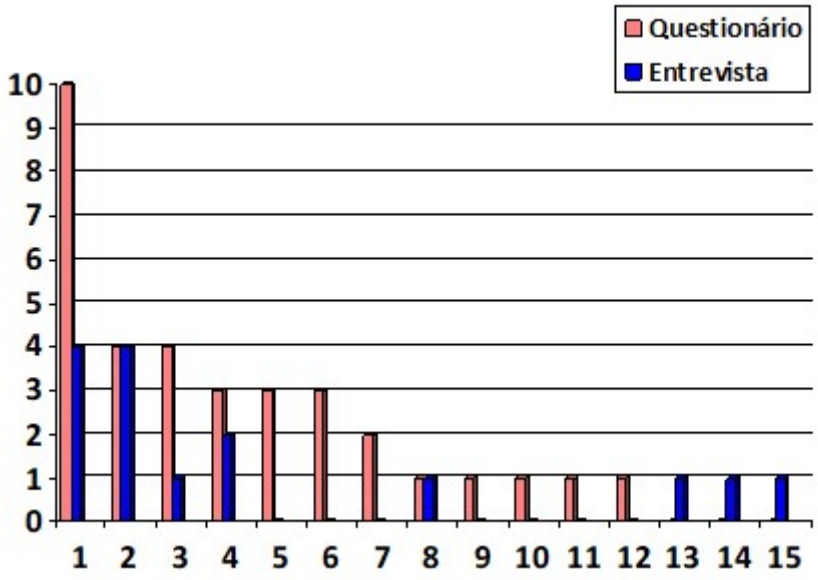


Figura 2. Evidências de problemas relacionados ao processo de análise de requisitos de *software*.
Fonte: Autoria própria (2018).

A falha na comunicação e a má compreensão da solicitação pela equipe de desenvolvimento são relatos que evidenciam uma falha no processo que envolve desde a comunicação com o cliente até a equipe responsável pelo desenvolvimento. Um entrevistado relatou problemas relacionados à usabilidade da funcionalidade desenvolvida: "os clientes não sabiam utilizar o sistema e entravam em contato com o suporte da empresa para obter um entendimento". O produto que era facilmente operado pela equipe de desenvolvimento, não tendo passado por testes de usabilidade, ao ser entregue em sua versão final, não era operado por diversos clientes, por não saberem como utilizá-lo.

Os participantes também relataram clientes que não informam o problema e somente apresentam a solução desejada. O analista encaminha a solicitação para o desenvolvimento, sem investigar o real problema enfrentado pelo cliente. Entretanto, após ser desenvolvida e entregue, a solução não atende ao que o cliente necessita, ou seja, o real problema não é solucionado.

7. Considerações finais

Este trabalho apresenta a forma como o processo de análise de requisitos realizado nas empresas de desenvolvimento de *software* impacta no produto final, tanto na perspectiva da empresa em relação ao retrabalho, como na perspectiva do cliente que não tem a necessidade atendida pelo sistema.

Os resultados aproximam a realidade das empresas à conceitos acadêmicos, assim, pretende-se a extensão desse trabalho para a verificação de técnicas utilizadas nas

etapas de elicitação e análise de requisitos, bem como a integração de técnicas de prototipagem de *software* para validação de requisitos com usuários. A fim de identificar como a utilização de técnicas de prototipagem de *software* auxilia o desenvolvimento de funcionalidades mais assertivas e que atendam as solicitações dos usuários, empregadas em contextos reais das empresas da região.

Como trabalhos futuros pretendem-se a realização de uma análise comparativa entre os resultados aqui apresentados e a literatura, bem como expandir os objetos de estudo do trabalho, por questionário e entrevistas. Também a realização de pesquisas para identificar a associação de técnicas de prototipagem às etapas de coleta e análise de requisitos, na busca em aprimorar as especificações de requisitos para o desenvolvimento de produtos mais assertivos.

8. Referências

Fu, J.; Bastani, F. B.; Yen, I. L. (2008). Model-driven prototyping based requirements In: PAECH, B.; MARTELL, C. (Ed.). Innovations for Requirement Analysis. Stakeholders Needs to Formal Designs. Berlin, Heidelberg: Springer Berlin Heidelberg, pages 43–61.

Laskowska, A. et al. (2014) Best practices for validating research software prototypes – markos case study. In: eChallenges e-2014 Conference Proceedings, pages 1–9. ISSN 2166-1650.

Machado, F. N. R. (2016). Análise e Gestão de Requisitos de Software. São Paulo, SP: Érica.

Nielsen, J. (2012). "Human Computer Interaction", https://www.nngroup.com/articles/usability-101-introduction-to-usability, September.

Peres, A. L.; Meira, S. L. Towards a framework that promotes integration between the ux design and scrum, aligned to cmmi. In: 10th Iberian Conference on Information Systems and Technologies. 2015.

Salah, D. (2011). A framework for the integration of user centered design and agile software development processes. In: 33rd International Conference on Software Engineering, pages 1132–1133.

Sommerville, I. (2011). Engenharia de Software. 9. ed. Sao Paulo, SP: Pearson.

Suranto, B. (2015). Software prototypes: Enhancing the quality of requirements engineering process. In: International Symposium on Technology Management and Emerging Technologies, pages 148–153.

Vazquez, C. E.; Simões, G. S. (2016). Engenharia de Requisitos. 1. ed. Brasport.

Wiegers, K. E. (2003). Software Requirements: Practical techniques for gathering and managing requirements throughout the product development cycle. 2. ed. Washington, EUA: Microsoft Press.

Análise Comparativa de Metodologias de Priorização de Demandas com Foco em Valor de Negócio na Produção de Software

Jackson G. Schimit[1], Pedro Henrique de Alencar Machado[1], Felipe Brena Ribeiro[1], Luana Belusso[1], Rafael Alves Paes de Oliveira[1]

[1]Coordenadoria do Curso de Engenharia de Software – Universidade Tecnológica Federal do Paraná (UTFPR)
Estr. p/ Boa Esperança, S/n - Zona Rural - 85.660-000 - Dois Vizinhos - PR - Brasil

jackson.schimit@gmail.com, pedrohmachado@utfpr.edu.br,

felipebrenaribeiro@gmail.com, luanabelusso@alunos.utfpr.edu.br,

raoliveira@utfpr.edu.br

Abstract. *The need of development and delivering software solutions that clients uses makes more necessary when the clients don't use a high number of functionalities delivered in applications. Based on this problem, this study focus on a comparative analysis about methods of priorization of demands which considers business value as a criteria. A survey of methodologies who considers the criteria's was made, and after, the survey and the comparative analysis were compared and the results showed.*

Resumo. *A necessidade do desenvolvimento e entrega de soluções de software que os clientes usam, se faz cada vez mais necessário quando evidenciado um alto número de funcionalidades entregues em aplicações que os clientes pouco utilizam. Com base nessa problemática, este estudo, objetiva à realização de uma análise comparativa sobre metodologias de priorização de demandas que consideram valor de negócio em seus critérios. Foi realizado um levantamento de metodologias que consideram tais aspectos como relevantes, após isso, as respectivas foram comparadas e os resultados apresentados.*

1. Introdução

Existem muitos processos de desenvolvimento de software disponíveis, mas é fundamental que todos eles possuam em seu conteúdo as atividades de especificação, projeto e implementação, validação e evolução, para que, quem desenvolve possa garantir a qualidade não só do processo, mas também do produto [Sommerville 2010]. A falta de padronização ou dificuldade na escolha de uma métrica para priorizar demandas durante o processo faz com as empresas optem pelo mais conveniente para o produto, cliente ou momento, induzindo as equipes a realizar incorretamente a priorização na maioria dos casos [Machado 2017].

Este trabalho tem como objetivo, realizar uma análise comparativa entre metodologias de priorização de demandas que considerem valor de negócio.

2. Fundamentação Teórica

2.1. Valor de Negócio

Entregar software que gere valor agregado se tornou objetivo, mas valor pode ser agregado, monetário ou de negócio, depende de quem analisa e da perspectiva. A maioria dos métodos que medem "valor agregado", apontam rastreamento de custos em cronograma de projeto como valor agregado, mas não tratam em nenhum momento o valor de negócio [Biffl et al. 2006]. Toda implementação de software parte de uma necessidade de negócio, desta forma, entendemos que gerar "valor de negócio"seja atender as necessidades dos clientes. Em práticas ágeis a definição de "valor de negócio"fica a cargo e responsabilidade de algum profissional envolvido no processo, que no caso do *Scrum* é o *Product Owner*. Porém a interpretação sobre o que é valor pode divergir de uma pessoa para outra, fazendo com que membros e partes de uma equipe ou organização tenham visões diferentes sobre o termo [Sommerville 2010].

No *Scrum* o *Product Owner* atua como um representante do cliente dentro da equipe, e uma de suas atividades consiste em priorizar o que deverá ser desenvolvido [Schwaber and Sutherland 2011].

2.2. Priorização de Demandas

Mesmo com a utilização das metodologias ágeis ainda são encontradas dificuldades durante o processo de fabricação de um software, e dentre elas, priorizar demandas ganha destaque, pois, saber em que momento cada item deve ser feito é tarefa complexa [Pressman 2011].

A incorreta priorização de demandas prejudica o produto e seus usuários, entregar funcionalidades que não serão utilizadas e não agregam valor causa prejuízo, insatisfação e aversão ao produto. Um estudo citado por Salinas(2015), analisou cerca de 1000 organizações e 2000 projetos de software que utilizam metologias ágeis, e divulgou um resultado onde ilustra que aproximadamente 64% das funcionalidades desenvolvidas nunca ou raramente são utilizadas, a figura 1 ilustra os dados do referido estudo [Torrecilla-Salinas et al. 2015].

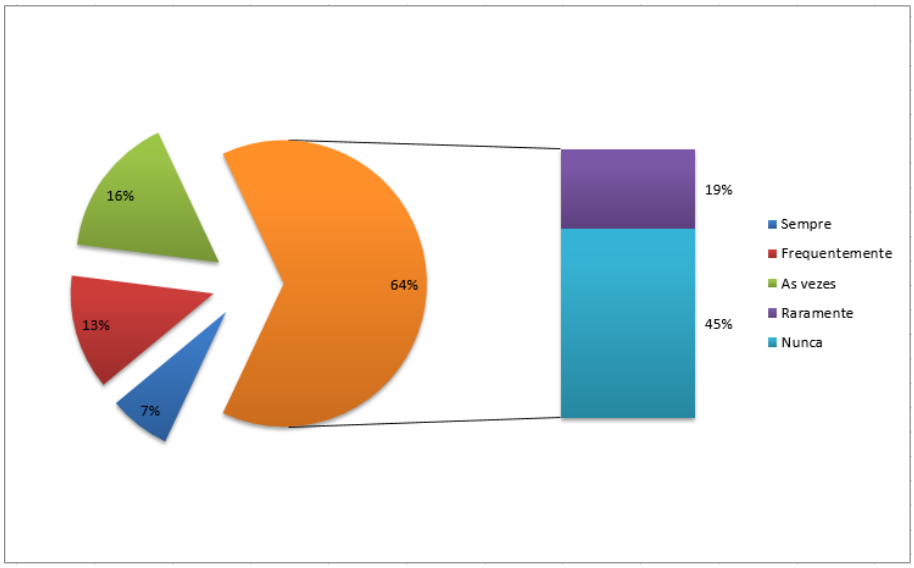


Figura 1. Adaptado de [Torrecilla-Salinas et al. 2015].

3. Metodologia

O estudo apresentado trata de uma pesquisa de caraterística exploratória. Para execução da metodologia utilizamos as etapas descritas na Figura 2.

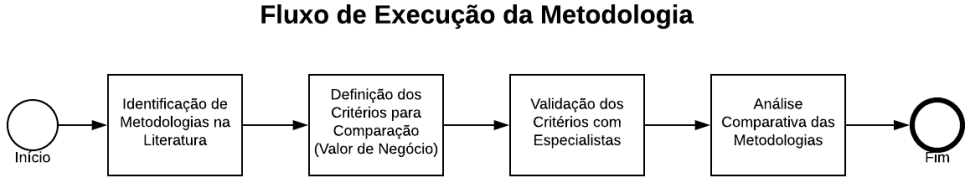


Figura 2. Etapas a serem executadas na metodologia.

3.1. Identificação das Metodologias na Literatura

Foram analisados trabalhos relevantes relativos a área de processos de produção de software que continham metodologias e dados sobre priorização de demandas. Para a realização da análise comparativa do trabalho em questão, foram elencadas três metodologias candidatas, cujas serão apresentadas nas sessões seguintes ao estudo.

3.2. Definição dos Critérios para Comparação (Valor de Negócio)

Buscando identificar qual das metodologias possui uma maior convergência com o foco em valor de negócio, foram elencados oito critérios que consistem em aspectos relacionados à definição de valor de negócio. Tais critérios foram escolhidos com base estudos presentes na literatura e opinião dos especialistas.

3.3. Validação dos Critérios com Especialistas

O trabalho contou com a colaboração de quatro profissionais da área de processo de produção de software, com o propósito de refinar e adequar os critérios de comparação definidos no item anterior. Foram realizadas entrevistas semi-estruturadas com os profissionais, cujos realizaram a revisão dos critérios e escolha dos mesmos. A Figura 3 ilustra o grau de instrução, cargos e experiência dos especialistas entrevistados.

Lista de Profissionais Colaboradores			
Colaborador	**Cargo**	**Instrução**	**Experiência**
Analista A	Gerente de Projetos	Pós Graduado	10 Anos
Analista B	Product Owner	Pós Graduado	8 Anos
Analista C	Diretor de Produto	Pós Graduado	15 Anos
Analista D	Arquiteto de Soluções	Pós Graduado	12 Anos

Figura 3. Profissionais colaboradores do estudo.

3.4. Análise Comparativa das Metodologias e Resultados

Depois que as metodologias e critérios foram identificados, o próximo passo foi realizar uma análise comparativa entre as respectivas.

4. Resultados e Discussões

Resultante das análises de estudos literários da área de processos de produção de software, as seguintes metodologias de priorização de demandas foram identificadas como aquelas que aderem ao termo de Valor de Negócio:

4.1. Moscow

Esta metodologia classifica as demandas de software mediante o seu valor para o negócio. A técnica é representada pelo acrônimo MoSCoW e possui quatro categorias, descritas na Figura 4, as quais as demandas podem ser classificadas [Achimugu et al. 2014], sendo elas:

***Must Have* (Essencial):** As demandas classificadas como essenciais são aquelas que são vistas tanto pelo cliente quanto pelo negócio como indispensáveis, e devem estar obrigatoriamente presentes nas primeiras versões de um produto.

***Should Have* (Necessário):** Nesta categoria se enquadram as demandas que que são vistas como necessárias, porém, não são imprescindíveis. Aqui são enquadrados itens importantes para o cliente, mas não impactam diretamente no sucesso do negócio.

***Could Have* (Desejável):** As demandas desejáveis nem sempre são vistas como relevantes pelo cliente, porém, são importantes para o negócio. Essas demandas tendem a ficar em segundo plano e não serem entregues nas primeiras versões devido a baixa visibilidade.

***Won't Have* (Não Necessário):** Aqui são elencadas as demandas que não são relevantes ou importantes tanto para o negócio quando para o cliente, estas não serão implementadas tão cedo, mas o seu mapeamento permite análises e ações futuras.

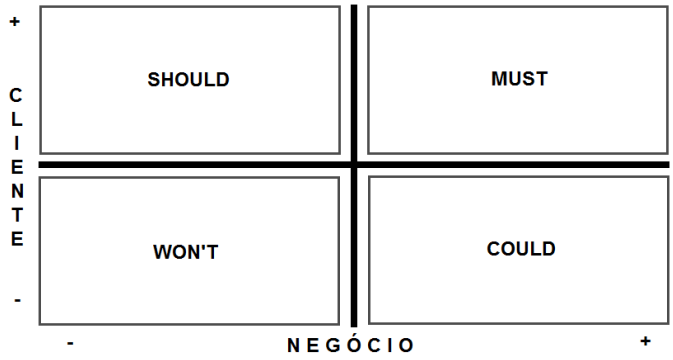


Figura 4. Adaptado de [Massari and Vidal 2018].

4.2. *Theme Screening*

É uma metodologia que se destaca por ser mutável e possuir uma aplicação imparcial e de fácil entendimento. Consiste em ordenar as funcionalidades com base em temas de negócio utilizados como fatores de comparação, conforme ilustrado na Figura 5 [Massari and Vidal 2018].

Para aplicar é preciso definir os temas de comparação e garantir que todos são válidos para todas as demandas em questão. Devem ser definidos no mínimo cinco e no máximo nove temas, onde, um deles deve classificado como tema base e servir de referência para a pontuação dos outros. O tema base recebe pontuação zero enquanto os mais importantes recebem um sinal e positivo e o menos importantes um sinal de negativo.

O valor de cada demanda priorizada é obtido através da soma dos sinais negativos e positivos, ao final, ordenando do maior para o menor temos o resultado da priorização obtido de forma simples e não viciada.

	Critérios						
	Complexidade	Esforço	ROI	Integração	Orçamento	Total	Prioridade
Acesso diferenciado para assinantes	-	-	+	+	+	+1	02
Disponibilizar vídeos sobre o produto	-	+	+	-	-	-1	04
Pagar com cartão de crédito	0	0	0	0	0	0	03
Integração com mídias sociais	+	+	+	-	+	+2	01
Disponibilizar acesso gratuito	-	-	+	-	+	-1	04

Figura 5. Adaptado de [Massari and Vidal 2018].

4.3. GUT

Consiste em uma metodologia que atribuí as demandas uma pontuação baseada em três variáveis relacionadas ao valor de negócio, onde cada variável possui sua escala de pontuação, que pode variar de vinte a cem pontos [Massari and Vidal 2018], conforme descrito abaixo:

Gravidade: Leva em consideração a importância que determinada demanda tem sobre o produto, avaliando o dano causado pela sua implementação (ou não) para o projeto como um todo, a escala de pontuação é definida conforme mostra Figura 6.

Gravidade	Pontuação
Extemamente Importante	100
Muito Importante	80
Importante	60
Relativamente Importante	40
Pouco Importante	20

Figura 6. Adaptado de [Massari and Vidal 2018].

Urgência: Avalia a urgência com que determinada demanda deve ser tratada, baseada no valor de negócio agregado. Possui a escala de pontuação conforme ilustração da Figura 7.

Urgência	Pontuação
Muito Alta	100
Alta	80
Média	60
Baixa	40
Muito Baixa	20

Figura 7. Adaptado de [Massari and Vidal 2018].

Tendência: Diz respeito a tendência de melhora ou piora da situação do produto com relação ao negócio, caso a demanda tratada seja desenvolvida (ou não) e sempre baseado no estado atual da demanda. A escala de pontuação desta variável pode ser visualizada na Figura 8.

Tendência	Pontuação
Melhorar / Piorar Muito	100
Melhorar / Piorar	80
Inalterado	60
Desaparecer Parcialmente	40
Desaparaecer Completamente	20

Figura 8. Adaptado de [Massari and Vidal 2018].

Após pontuadas as três variáveis o valor de negócio se dá pelo seguinte cálculo, (Gravidade + Urgência + Tendência) / Três. Onde a demanda com valor mais elevado é a mais importante e a com menor valor é a menos importante, conforme ilustrado na Figura 9.

	GUT			Total	Prioridade
	Gravidade	Urgência	Tendência		
Acesso diferenciado para assinantes	100	80	100	280	02
Disponibilizar vídeos sobre o produto	80	100	20	200	04
Pagar com cartão de crédito	100	100	40	240	03
Integração com mídias sociais	100	100	100	300	01
Disponibilizar acesso gratuito	40	40	20	100	05

Figura 9. Adaptado de [Massari and Vidal 2018].

4.4. Critérios de Comparação

Buscando obter resultados focados na geração de valor de negócio nas demandas priorizadas, e contando com o auxílio dos profissionais colaboradores, foram definidos os critérios de comparação ilustrados na figura 10.

Critérios de Comparação		
Identificador	**Título**	**Descrição**
C1	Importância para o negócio	Esse critério faz referência ao quão importante tal demanda é para o produto final.
C2	Visão do cliente	Entende como visão do cliente, o seu ponto de vista sobre um determinado recurso.
C3	Material teórico publicado	Material teórico publicado ajuda no propósito de entendimento e aplicação de uma respectiva
C4	Estudo pratico publicado	Além do material teórico, estudos práticos contribuem para a identificação de boas práticas e como obter os melhores resultados com tal metodologia.
C5	Pesos diferentes sobre os decisores	Dependendo do envolvido em tal priorização, a sua decisão pode ser mais relevante que de outros perfis, dessa forma, se faz importante tal fator.
C6	Complexidade técnica atribuída	O aspecto técnico é essencial para a priorização de determinado recurso, dependendo dos fatores de complexidade, uma determinada funcionalidade, pode não ser viável de ser implementada.
C7	Flexível a mudanças	Mudanças ocorrem, práticas maleáveis às mudanças se tornam mais relevantes que as demais.
C8	Aplicação adequada a equipes ágeis	Equipes ágeis tendem à serem mais adeptas a mudanças e se portarem melhor com cenários complexos, cujos estão relacionados ao contexto de priorização de demandas considerando valor de negócio.

Figura 10. Critérios de comparação definidos.

4.5. Análise Comparativa

Resultante da análise comparativa realizada aplicando os critérios de comparação explicitados no item anterior, temos os dados ilustrados na figura 11.

Metodologias	Critérios								Total
	C1	C2	C3	C4	C5	C6	C7	C8	
MOSCOW	SIM	SIM	SIM	SIM	NÃO	SIM	NÃO	SIM	60
Theme Screening	SIM	SIM	SIM	NÃO	NÃO	SIM	SIM	SIM	60
GUT	SIM	SIM	SIM	NÃO	NÃO	NÃO	NÃO	SIM	40

Figura 11. Comparativo entre metodologias.

A análise dos dados e resultados apresentados dão destaque para a metodologia *Theme Screening*, principalmente, por ser a única das três que atendeu C7, critério esse que segundo opinião dos especialistas, permite as equipes que a utilizarem poder adequar a metodologia da melhor forma a sua realidade.

5. Trabalhos Futuros

Para continuidade da pesquisa, será feita uma revisão da literatura visando encontrar novas metodologias e critérios, cujos, serão confrontados com os elencados no trabalho atual. As novas metodologias e critérios poderão ser reavaliadas pela mesma equipe de especialistas, gerando assim uma maior relevância da pesquisa.

6. Conclusões

Baseado na análise comparativa realizada entre as três metodologias de priorização de demandas e levando em consideração os oito critérios de comparação podemos concluir que, no cenário abordado por este trabalho, levando em consideração o valor de negócio para a priorização de demandas a metodologia *Theme Screening* se mostra mais eficiente e flexível em seus critérios, e ainda, possui um entendimento fácil, fazendo com que seja melhor aceita e aplicada pelas equipes de desenvolvimento.

Referências

Achimugu, P., Selamat, A., Ibrahim, R., and Mahrin, M. N. (2014). A systematic literature review of software requirements prioritization research. *Information and software technology*, 56(6):568–585.

Biffl, S., Aurum, A., Boehm, B., Erdogmus, H., and Grünbacher, P. (2006). *Value-based software engineering*. Springer Science & Business Media.

Machado, P. H. d. A. (2017). Redução de desperdÍcios no desenvolvimento de software de grande porte por meio de ferramentas lean.

Massari, V. L. and Vidal, A. (2018). *Gestão Ágil de Produtos com Agile Think Business Framework: Guia para certificação EXIN Agile Scrum Product Owner*. Brasport.

Pressman, R. S. (2011). Engenharia de software: uma abordagem profissional. 7ª edição. *Ed: McGraw Hill*.

Schwaber, K. and Sutherland, J. (2011). The scrum guide. *Scrum Alliance*, 21.

Sommerville, I. (2010). *Software engineering*. New York: Addison-Wesley.

Torrecilla-Salinas, C., Sedeño, J., Escalona, M., and Mejías, M. (2015). Estimating, planning and managing agile web development projects under a value-based perspective. *Information and Software Technology*, 61:124–144.

Analyzing the Impact of the Search Phase in a Systematic Mapping Study

João Carbonell[1], Luciano Marchezan[1], Aníbal Neto[1], Elder Rodrigues[1], Maicon Bernardino[1], Yury Alencar Lima[1]

[1]Federal University of Pampa (UNIPAMPA) – Alegrete, RS – Brazil.

{joaocarbonellpc, lucianomarchp, netoiiung, yuryalencar19}@gmail.com
elderrodrigues@unipampa.edu.br,bernardino@acm.org

Abstract. *Collect data from a research field is a crucial task in academic works to understand a scientific area. Systematic Mapping Studies (SMS) provide a formal and well designed protocol to select, classify and present relevant data. To select a digital library database or database set and to define the search strings used in database engines on digital libraries are search phase activities aimed to delimit the retrieved works quantity. This work analyzes the impact of definition of the digital libraries that may be used in the search phase on systematic mapping studies conduction, as well as the definition of the search strings. The results related with exclusive studies from each base were compared and presented. In addition, the difference in the results of the retrieved studies according to changes in the search string are presented. Based on these results, five activities that may give support to inexperienced researches when conducting SMS.*

Resumo. *Coletar dados de um campo de pesquisa é uma tarefa crucial em trabalhos acadêmicos para compreender uma área científica. Mapeamentos Sistemáticos da Literatura fornecem uma maneira formal e bem definida de se selecionar, classificar e apresentar dados relevantes. Selecionar bases de dados e definir as strings de buscas para estas são atividades importantes da fase de busca em um mapeamento sistemático. Estas atividades são utilizadas para se delimitar a quantidade de trabalhos pesquisados. Neste trabalho, o impacto da escolha das bases das bibliotecas digitais nas fases de busca de um Mapeamento Sistemático é analisado, bem como a definição das strings. Foram comparadas e demonstrados os resultados relacionados aos estudos exclusivos de cada base, além das diferenças nos resultados com a troca de termos e caracteres especiais em strings de buscas. Também foram mapeadas cinco atividades que podem ajudar pesquisadores com pouca experiência conduzindo mapeamentos sistemáticos.*

1. Introduction

Systematic Mapping Studies (SMS) are important in the scientific research field to collect overall information about a topic of interest. In addition, SMS may be used to identify contributions for an area and give support to other researchers to find primary studies of the specific area.

To perform a SMS, there is a well-defined protocol proposed by [Petersen et al. 2008] that may be followed, increasing the chance of obtaining

relevant results. In the search phase of this protocol, databases must be selected, and search strings have to be formulated. These decisions have impact in the search phase, because they are directly related to the possible set of retrieved studies. However some researchers often uses scientific databases which are known to index minor databases works.

The protocol executed when performing a SMS is extremely important to achieve satisfactory results. Thus, several decisions made during the preparation of the protocol may impact in the final set of selected studies. The objective of this work is to investigate, analyze and provide evidence about the impact of some decisions related with the search phase of the SMS protocol. The impact of these decisions may be summarized as two Research Questions (RQ):

RQ1. What is the impact of the database selection?
RQ2. What is the impact of a poor formulated search string?

To answer these RQs we observed and documented the information directly related with these decisions impact: (i) how researches made these decisions; (ii) which information and reasons were used to make the decisions, and; (iii) what was the impact of these decisions in the final result of the SMS. The result of this analysis indicate the problems that may appear if some aspects of the SMS are not carefully planed.

The rest of this work is structured as follows. Section 2 gives a description of systematic studies and their protocols. Section 3 details our motivation and the research methodology used to achieve our results. Section 4 presents and discuss the results of our research. Finally, Section 5 concludes this work.

2. Background

2.1. Systematic Studies

Systematic studies are a research methodology commonly used in scientific field [Connolly et al. 2012, Stuck et al. 1999, Šmite et al. 2010]. These studies follow guidelines and well-defined protocols [Kitchenham 2004], executing process activities to improve the analysis of studies and obtained results, thus reducing the bias and mitigating the loss of data and threats to the research. They aggregate the experience, date and information from different studies in order to answer a specific research question defined by the researchers [Budgen and Brereton 2006].

2.2. Systematic Literature Review and Systematic Mapping Study

The Systematic Literature Reviews (SLR) are inspired on medical research field activities. Their main goal is to perform a deep analysis of study methodologies and results to collect and evaluate evidence pertaining to a focused topic [Biolchini et al. 2005]. According to [Budgen and Brereton 2006], a systematic review is a means of identifying, evaluating and interpreting research relevant data from a topic area, or phenomenon of interest.

Similar to a SLR, a Systematic Mapping Study (SMS) uses guidelines and protocols to find relevant studies. In the SMS, the results are categorized, summarized and presented in a map [Petersen et al. 2008] which shows that data in form of graphs and tables.

2.3. Protocols of Systematic Studies

The protocols of systematic studies are processes in which every step aims to ensure the relevance of the data collected and to avoid rework. An example of mapping protocol is presented by [Petersen et al. 2008] (see Figure 1). Some protocol activities, such as definition of exclusion and inclusion criteria, improve the depth and relevance of primary studies. Others activities, such as definition of scope, research questions and search strings, helps to delimit the boundaries of the review.

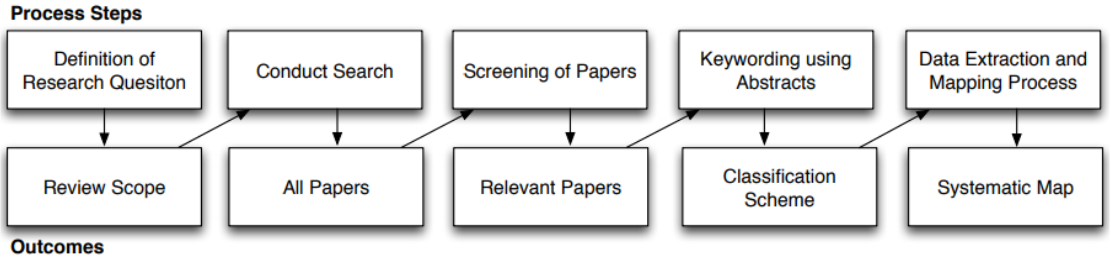


Figure 1. Systematic Mapping Protocol [Petersen et al. 2008]

The definition of the search strings must be in accordance with the search engine limitations of each database. In the structure of a search string, terms and logical operators are inserted to be read according to the syntax of each search engine. According to [Petersen et al. 2008] a good way to construct the search string is to structure them with PICO (population, intervention, comparison and outcome) [Kitchenham 2004].

3. Motivation and Methodology

In the Software Engineering context, both SLR and SMS have great importance in a proposal research methodology. Although there are several protocols for both, more commonly used for SLR is presented in [Kitchenham 2004] and for SMS was propose in [Petersen et al. 2008]. These protocols detailed the whole processes to perform these systematic reviews/mappings, however, there is still some decisions that have to be made by who is executing those protocols. These decisions are related with the search phase, which is similar for both SLRs and SMSs. We could summarize these decisions with two questions:

1. Which databases should be used for the search of studies?
2. Which search string should be used in each database?

Answer these questions is not a trivial task. To collect evidence that may help us obtain those answers we observed and registered the reasons and results related with those questions during the search phase of a real SMS conducted between July and September 2017. The objective of this SMS was to present and overview of DSL supporting tools. During the search phase of this SMS we registered: (i) the number of studies return from each database, unique studies of each database and the number of studies from each database included in the final selection; (ii) generic string, strings for each database and impact of minor changes on them; The results of the collected data are presented in the following sections.

4. Results and Discussion

By analyzing the data collected during the execution of the SMS we could answer the following RQ.

4.1. RQ1. What is the impact of the database selection?

To decide which databases should be used to perform the search some characteristics related with those databases have to be considered:

1. The capacity of the search engine related with the number of terms, synonyms and operators (AND, OR, NOR) that can be included in the search string;
2. If the databases indexes exclusive studies;

Although some researchers may argue that some databases such as SCOPUS and Compendex index works from other databases, that are still works which are exclusive to a specific database. During the conduction of the SMS used to answer the RQ of this work, the databases used were IEEEXplore, ACM Digital Library, Compendex (Engineering Village), Science Direct, SCOPUS and SpringerLink. We documented the total number of studies from each database, the number of studies exclusive to a database and also how many of the exclusive studies were part of the final selection.

Table 1. Number of relevant studies retrieved from each database

Database	Retrieved Studies		
	Total	**Exclusive**	**Final Selection**
ACM Digital Library	401	89	3
Compendex	424	29	1
IEEE Xplore	355	33	3
Science Direct	101	23	1
SCOPUS	960	352	4
SpringerLink	154	107	0

Table 1 presents these numbers, where SCOPUS was the database with higher number of retrieved studies, 960. Science Direct and SpringerLink had the lower number of retrieved works, 101 and 154 respectively. Considering the exclusive studies, SCOPUS was also the database with the higher number, 352. Compendex and Science direct on other hand had the lower number, 29 and 23. Even with the higher number of exclusive retrieved studies, SCOPUS, should not be used and the only database during a systematic study. The reason is the exclusion of a possible high number of exclusive studies from other database. In this case, the total of 281 would not be considered because they were not indexed in SCOPUS. Considering the works for the final selection, ACM and SCOPUS were the databases with more exclusive studies selected, 3 and 4 respectively. These results indicated that even if only one database, such as IEEE, would not be included in the SMS, some studies would not be mapped in the SMS.

In addition, some of the retrieved studies contained insufficient data related to our topic of interest, which was tools to develop DSLs. In this case, we have to further search in the grey literature about additional information of the tools extracted from the studies.

4.2. RQ2. What is the impact of a poor formulated search string?

The first step was to define a generic search string that would be used to find studies about domain specific languages development tools. Figure 2 presents the generic search string.

The string have terms and synonyms used on related researches of the Domain Specific Languages area. The term "DSML" is related to Domain Specific Modeling Languages, which are languages that have graphical notations. The terms "Little Language" and "Small Language" are both less used synonyms, but still would be used because in SMSs we need to cover the research area as widely as possible.

```
(DSL OR DSML OR Domain Specific Language OR Domain-Specific
Language OR Domain-Specific-Language OR Domain-Specific
Modeling Language OR Domain Specific Modeling Language OR
Domain-Specific-Modeling-Language OR Small Language OR
Small-Language OR Little Language OR Little-Language AND
Tool OR Language Workbench)
```

Figure 2. Search String

However, six databases were used during this research, then it was necessary to derive each search string for each specific database as shown in Table 2. The IEEEXplore database have a specific differences about the search engine, this database limits the search to 15 terms in each string. Therefore the composite search strategy was to use two search strings, then we split the generic search string into two compound terms, which are: DSL and DSML. Keeping the other common terms in each compound search.

For others databases the string were derived accord to each search engine. It is important to know that for some databases the use of plural it makes a difference about the retrieved works.

These derived strings, however, are not free from problems, as some databases have specific rules that may have great impact in the results. For instance, if we include quotation marks in the terms *language workbench* in the SpringerLink string, the results would change from 154 to 1,257 returned studies. This small change in the string, could cost a lot of meaningless effort during the SMS because of these additional 1,103 studies, none would be included in the final selection.

A similar problem occurred with ACM Digital Library, where by removing some of the DSL synonyms, *Little Language* and *Small Language*, in this case, we obtained a total of 273 studies, 128 less than the original 401.

4.3. Preliminary Activities Before Conducting a SMS

As final result of our analysis during the conduction of a SMS, we documented five activities to be performed before conducting the SMS. This preliminary activities may help inexperienced researches when preparing to conduct both SMS or SLR. As illustrated in Figure 3, the first activity is to select the databases that will be used during the SMS or SLR. This activity is really important because as showed in previous sections, most databases index exclusive works from conferences and journals. Thus, the not addition of a database will exclude any of these exclusive works.

Table 2. Search Strings Derived for Each Database

Database	Search String
ACM Digital Library	((acmdlTitle:("DSL" "DSML" "Domain Specific Language" "Domain-Specific Language" "Domain-Specific-Language" "Domain Specific-Language" "Domain-Specific Modeling Language" "Domain Specific Modeling Language" "Little-Language" "Little Language" "Small-Language" "Small Language"))) OR (recordAbstract:("DSL" "DSML" "Domain Specific Language" "Domain-Specific Language" "Domain-Specific-Language" "Domain Specific-Language" "Domain-Specific Modeling Language" "Domain Specific Modeling Language" "Little-Language" "Little Language" "Small-Language" "Small Language")) AND ((acmdlTitle:("Tool" "Tools" "Language Workbench")) OR (recordAbstract:("Tool" "Language Workbench")))
Compendex (Engineering Village)	(((DSL) OR (DSML) OR ("Domain Specific Language") OR ("Domain-Specific Language") OR ("Domain-Specific-Language") OR ("Small Language") OR ("Small-Language") OR ("Little-Language") OR ("Little Language") OR ("Domain Specific Modeling Language") OR ("Domain-Specific Modeling Language") OR ("Domain-Specific-Modeling-Language") OR ("Domain-Specific Modeling-Language" OR "Domain Specific Modeling-Language") WN KY) AND (tool) OR ("Language Workbench") WN KY) AND ((English) WN LA)
IEEE Xplore - DSL	(((((DSL OR "Domain Specific Language" OR "Domain-Specific Language" OR "Domain-Specific-Language" OR "Domain Specific-Language" OR "Small Language" OR "Small-Language" OR "Little-Language" OR "Little Language") AND (Tool OR "Language Workbench")))))
IEEE Xplore - DSML	(((DSML OR "Domain-Specific Modeling Language" OR "Domain Specific Modeling Language" OR "Domain-Specific Modeling-Language" OR "Domain Specific Modeling-Language") AND (Tool OR "Language Workbench")))
Science Direct	TITLE-ABSTR-KEY(DSL or DSML or "Domain Specific Language" or "Domain-Specific Language" or "Domain-Specific-Language" or "Domain Specific-Language" or "Little-Language" or "Little Language" or "Small-Language" or "Small Language" or "Domain Specific Modeling Language" or "Domain-Specific Modeling Language" or "Domain-Specific-Modeling-Language" or "Domain-Specific Modeling-Language") and TITLE-ABSTR-KEY(tool or "Language Workbench")
SCOPUS	(TITLE-ABS-KEY (dsl OR dsml OR "domain specific language" OR "domain-specific language" OR "domain-specific-language" OR "domain specific language" OR "domain specific modeling language" OR "domain-specific modeling language" OR "domain specific modeling-language" OR "domain-specific-modeling-language") AND TITLE-ABS-KEY (tool OR "language workbench")) AND (LIMIT-TO (LANGUAGE , "English"))
Springer Link	((DSL OR DSML "domain specific language" OR "domain-specific language" OR "domain-specific-language" OR ("Small Language") OR ("Small-Language") OR ("Little-Language") OR ("Little Language") OR ("Domain Specific Modeling Language") OR ("Domain-Specific Modeling Language") OR ("Domain-Specific-Modeling-Language") OR ("Domain-Specific Modeling-Language" OR "Domain Specific Modeling-Language") AND (tool or language workbench))

The second activity is the definition of the search string based on the terms and synonyms related with the research area. A crucial point here is the difference between SMS and SLR search strings. In SMS generic strings should be less restrictive to achieve more wider results while SLR aims to specific areas or topics.

During the third activity, the generic string is derived into specific strings for each database search engine. Then, a preliminary search with the variations of the search string is performed in the search engines. If problems are found with the strings, there is a possibility that the generic strings have problems as well, so it should be re-defined. It

is difficult to say if a search string have problems or not. A possible hint to identify this problem is a huge or tiny number of retrieved studies.

During the last activity the returned studies are briefly analyzed. This analysis may be performed only in the titles and abstracts of the studies. If there is a great number of studies outside the scope of the review, there is a possibility that there are still problems with the search strings. If the number of studies outside scope are small or zero, then researchers may continue to the other phases of their review.

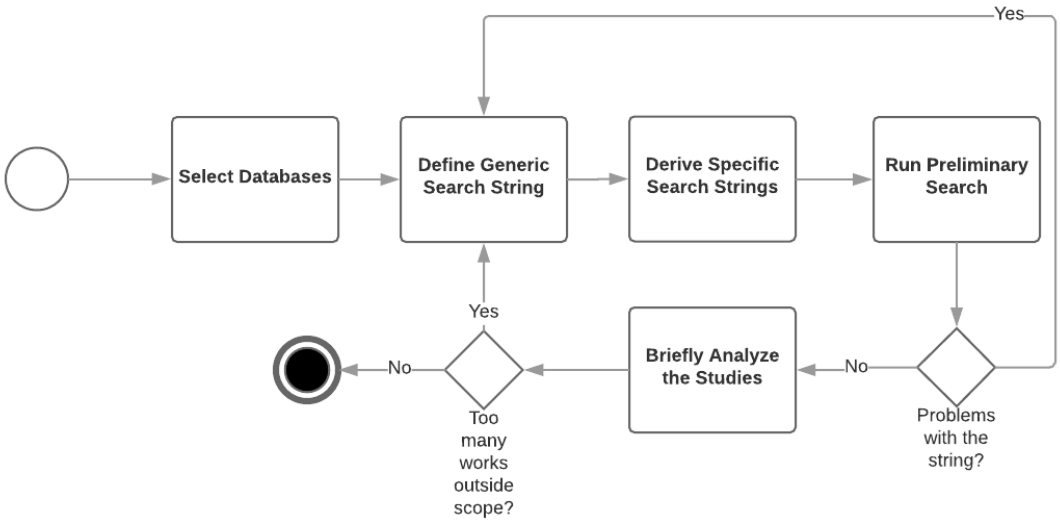


Figure 3. Activities conducted before the DSLs SMS

5. Conclusion

This work presented evidences related to the impact of the search phase in a SMS conducted in the DSL area. This evidence could be used to analyze issues that may have negative impact in a SMS such as: researchers decided to use only one digital library (*e.g.* SCOPUS), thus involuntary excluding exclusive works from other databases; and researchers used a poor formulated search string which did not cover the area as expected.

The evidence was collected by observing and documenting data during the conduction of the search phase of the SMS in the DSL area. We registered the different numbers of studies for each database, as well as exclusive works of a database that were included in the final SMS selection. Based on this numbers we have made some decisions, such as reformulating a search string.

The results of the data collected during our SMS gave us evidence to achieve two conclusions:

1. **The researches should include as many databases as possible**: this selection may have small but important impact in the final selection. For instance, if during our SMS we did not search works in SCOPUS, 4 studies would not be included in the final selection. Although this is a small number, the not inclusion of these works would have a huge impact in the SMS analysis, such as, not including emerging DSL tools;

2. **Minor changes in the search string may change the retrieved studies numbers to more than 1000**: in addition, the IEEE Xplore string had to be split because of the limitation of the database terms. When deriving the generic string for each database we also had to be careful, as some databases had returned a complete different number of studies with minor changes in the strings.

In addition, we documented five activities that may give support to researches with low experience conducting a SMS or SLR. This activities should be performed before the conduction of the systematic review and can be considered preliminary activities.

For the future, we are currently working to expand this analysis into the additional phases of the SMS protocol. We plan to construct a set of guidelines based on this research and develop a tool to give support to SMS and SLR conduction with the implementation of such guidelines.

References

[Biolchini et al. 2005] Biolchini, J., Mian, P. G., Natali, A. C. C., and Travassos, G. H. (2005). Systematic review in software engineering. *System Engineering and Computer Science Department COPPE/UFRJ, Technical Report ES*, 679(05):45.

[Budgen and Brereton 2006] Budgen, D. and Brereton, P. (2006). Performing systematic literature reviews in software engineering. In *Proceedings of the 28th international conference on Software engineering*, pages 1051–1052. ACM.

[Connolly et al. 2012] Connolly, T. M., Boyle, E. A., MacArthur, E., Hainey, T., and Boyle, J. M. (2012). A systematic literature review of empirical evidence on computer games and serious games. *Computers & Education*, 59(2):661–686.

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

[Petersen et al. 2008] Petersen, K., Feldt, R., Mujtaba, S., and Mattsson, M. (2008). Systematic mapping studies in software engineering. In *EASE*, volume 8, pages 68–77.

[Šmite et al. 2010] Šmite, D., Wohlin, C., Gorschek, T., and Feldt, R. (2010). Empirical evidence in global software engineering: a systematic review. *Empirical software engineering*, 15(1):91–118.

[Stuck et al. 1999] Stuck, A. E., Walthert, J. M., Nikolaus, T., Büla, C. J., Hohmann, C., and Beck, J. C. (1999). Risk factors for functional status decline in community-living elderly people: a systematic literature review. *Social science & medicine*, 48(4):445–469.

MAS-ML - Uma Linguagem para Modelagem de Sistemas Multi-Agentes: Uma Análise do Estado da Arte por Meio de uma Revisão Sistemática

Lukas Felipe Gaedicke, Gilleanes Thorwald Araujo Guedes, João Pablo Silva da Silva

[1]Universidade Federal do Pampa - Campus Alegrete (UNIPAMPA)
Av. Tiarajú, 810 - Ibirapuitã, Alegrete – RS – Brazil

{lukasfgaedicke, gilleanesguedes,joaosilva}@unipampa.edu.br

Abstract. *In this paper we present and analyze MAS-ML, a language for modeling multiagent systems. We show the language evolution from its initial proposal to the present day and determine its strong and weak aspects. This study has as its goal to establish the state-of-art of this language and determine how it can be improved. To achieve this goal we conducted a systematic review of this language, retrieving all the papers considered relevant about MAS-ML, describing the language evolution and analyzing its features.*

Resumo. *Neste artigo apresentamos e analisamos a MAS-ML, uma linguagem para modelagem de sistemas multiagentes. Nós apresentamos a evolução da linguagem desde sua proposta inicial até os dias atuais e determinamos seus aspectos fortes e fracos. Este estudo tem como objetivo estabelecer o estado-da-arte dessa linguagem e determinar como ela pode ser melhorada. Para atingir este objetivo, conduzimos uma revisão sistemática da literatura sobre esta linguagem, recuperando todos os artigos considerados relevantes sobre a MAS-ML, descrevendo as evoluções da linguagem e analisando suas características.*

1. Introdução

O interesse em utilizar agentes como auxiliares nas mais diversas aplicações vem crescendo ao longo dos anos. Todavia, o desenvolvimento de sistemas que suportem agentes apresenta novos desafios para a engenharia de software, o que demonstrou a necessidade da criação de novas metodologias e processos de desenvolvimento para este tipo de sistema, bem como da criação de linguagens de modelagem que os suportem. Levando isto em consideração, várias linguagens foram derivadas a partir da UML (Unified Modeling Language), uma linguagem amplamente utilizada na área de engenharia de software, para o projeto de sistemas multiagentes (SMAs).

Dentre as linguagens que estendem a UML, a MAS-ML (MultiAgent Systems Modeling Language) em seu estado atual, está entre as que contemplam a maior quantidade de recursos para a modelagem das características estruturais e comportamentais dos agentes. Isto é corroborado por [Gonçalves et al. 2015], que a compara com as linguagens AUML de [Odell et al. 2000] e AML de [Cervenka and Trencansky 2007]. Isso posto, consideramos a MAS-ML como possuidora de um interesse particular de estudo.

Segundo [Silva et al. 2008b], a MAS-ML é uma linguagem de modelagem que tem como principal objetivo facilitar a abstração no desenvolvimento de SMAs. Esta linguagem foi estendida a partir da superestrutura da UML, no qual foram inseridos novos

conceitos de modelagem, representados por novas metaclasses e estereótipos, consideradas apropriadas para capturar as características típicas de SMAs.

Com o objetivo de estabelecer o estado-da-arte dessa linguagem e determinar como ela pode ser melhorada, realizamos uma revisão sistemática sobre MAS-ML, recuperando os artigos considerados relevantes. A partir disso, nós analisamos as características acrescidas à linguagem a cada evolução, procurando identificar as vantagens e limitações da MAS-ML. Esse trabalho se justifica pelo fato de não termos encontrado outra revisão sistemática sobre o tema.

A estrutura deste artigo está organizada da seguinte forma: na Seção 2, é apresentado uma breve fundamentação teórica, na Seção 3 apresentamos a metodologia e o protocolo utilizado para a revisão sistemática. Na Seção 4 apresentamos os resultados obtidos através da revisão sistemática e, após isso, as considerações finais desse estudo.

2. Fundamentação Teórica

2.1. UML (Unified Modeling Language ou Linguagem de Modelagem Unificada)

Segundo [Guedes 2018] a UML é uma linguagem padrão de modelagem adotada internacionalmente pela indústria de engenharia de software. Esta linguagem é utilizada principalmente para modelar software durante a sua especificação de requisitos e seu projeto.

A UML oferece diversos diagramas com objetivos e visões distintas, divididos em estruturais e comportamentais. Dentre estes podemos citar o diagrama de casos de uso, utilizado para identificar os requisitos do software; o diagrama de classes que permite representar as classes que compõem o sistema e como elas se relacionam; o diagrama de sequência que demonstra a ordem em que as mensagens são disparados entre os objetos envolvidos em um processo; e o diagrama de atividade utilizado para representar o fluxo de controle e de objetos de uma atividade, podendo esta ser um método, um algoritmo, ou mesmo um processo completo.

2.2. Noções Básicas de Agência

De acordo com [Vicari and Gluz 2007], um agente é um processo computacional, situado em um ambiente, projetado para atingir um propósito através de um comportamento autônomo e flexível. O propósito de um agente pode ser especificado pela definição de suas crenças e desejos e o comportamento deste agente é fortemente influenciado por suas intenções. Agentes podem perceber eventos que ocorrem sobre o ambiente e realizar ações sobre o mesmo. Podem ser classificados em reativos ou cognitivos, os reativos apenas reagem a eventos, enquanto os cognitivos possuem crenças, desejos e intenções a respeito do ambiente em que se encontram. Um sistema composto por um conjunto de agentes é chamado sistema multiagente.

3. Protocolo da Revisão Sistemática

Para realização deste estudo, realizamos uma revisão sistemática da literatura. De acordo com [Kitchenham 2004], uma Revisão Sistemática (RS) tem como objetivo identificar, avaliar e interpretar os resultados do estudo que estão relacionados com as questões, área temática ou fenômeno que se deseja coletar evidências para servir como fundamentação para conclusões.

De acordo com [Biolchini et al. 2005], uma RS deve basear-se em um protocolo previamente definido, que deve formalizar a execução da RS desde a estipulação das questões de pesquisa, passando pelo estabelecimento dos critérios de seleção (inclusão e exclusão de trabalhos), das bases digitais utilizadas para a extração dos estudos relacionados e das palavras chaves aplicadas durante a busca nestas bases, até a definição de como será o relatório final.

Dessa forma, elaboramos um protocolo[1], no qual definimos duas questões de pesquisa *("QP1. Quão abrangente é a MAS-ML para a modelagem de sistemas multiagentes?")* e *"QP2. Quais são as lacunas da MAS-ML para a modelagem de sistemas multiagentes?").* Neste protocolo também definiu-se a estratégia de seleção dos trabalhos e a forma de extração das informações nos estudos encontrados. A tabela 1 apresenta os critérios de inclusão e exclusão utilizados durante a revisão sistemática.

Tabela 1. Critérios de inclusão e exclusão

Tipo	Critério
Inclusão	O trabalho deve reportar explicitamente estudos sobre a linguagem MAS-ML.
Exclusão	Estudo não escrito em inglês.
Exclusão	Estudo não disponíveis para download.
Exclusão	Estudo com menos de 6 páginas.
Exclusão	Estudo não relacionado à modelagem de sistemas multiagentes.

Para encontrar os trabalhos, realizamos um processo de busca em seis bases de dados digitais (ACM Digital Library, IEEE Xplore, Engineering Village, ScienceDirect, SpringerLink e Scopus), todas elas amplamente utilizadas e conhecidas no meio acadêmico. Também definiram-se 3 palavras chave para a busca de estudos relacionados: MAS-ML, Modeling Language e Multi-Agent Systems. Com o intuito de aumentar a abrangência desse estudo, buscando identificar a maior quantidade de artigos sobre a MAS-ML, foram definidos mais 4 sinônimos derivados das palavras chave, sendo elas Multi-Agent System Modeling Language, Modeling Agents e MAS.

Após o levantamento de termos centrais e seus sinônimos foi construída uma *string* de busca genérica, que pode ser observada na figura 1. Contudo é importante ressaltar que para cada uma das bases de dados pesquisadas a string padrão sofreu modificações com o objetivo de adequá-la às particulariedades de cada um dos mecanismos de busca adotados pelas bases online.

Para selecionar os estudos, definimos 4 etapas: (1) executar as strings de busca nas bases digitais; (2) remover os estudos duplicados; (3) aplicar os critérios de exclusão nos estudos restantes da etapa 2; (4) aplicar os critérios de inclusão nos estudos remanescentes da etapa 3; (5) ler e extrair as informações dos estudos remanescentes da etapa 4.

```
(MAS-ML OR Multi-Agent System Modeling Language) AND (Modeling Language OR
                Modeling Agents AND Multi-Agent Systems OR MAS)
```

Figura 1. Ilustração da *String* genérica utilizada.

[1]Artefatos gerandos durante a execução do protocolo `https://github.com/LukasGaedicke/MAS-ML-SR-Artifacts/`

4. Resultados

A figura 2 ilustra as etapas do processo de pesquisa durante o qual houve uma redução na quantidade dos estudos relacionados com nosso objetivo a cada etapa do processo. Assim, na Etapa 1, a busca retornou 186 resultados e após isso, na execução do passo 2 restaram 151 estudos, remanescentes da eliminação de trabalhos duplicados. Já na aplicação dos critérios de exclusão, na Etapa 3, restaram 53 estudos. Na etapa 4 do processo, em que foram aplicados os critérios de inclusão, restaram apenas 8 estudos, os quais foram lidos e analisados. As informações extraídas destes trabalhos são apresentadas e discutidas na seção 4.1.

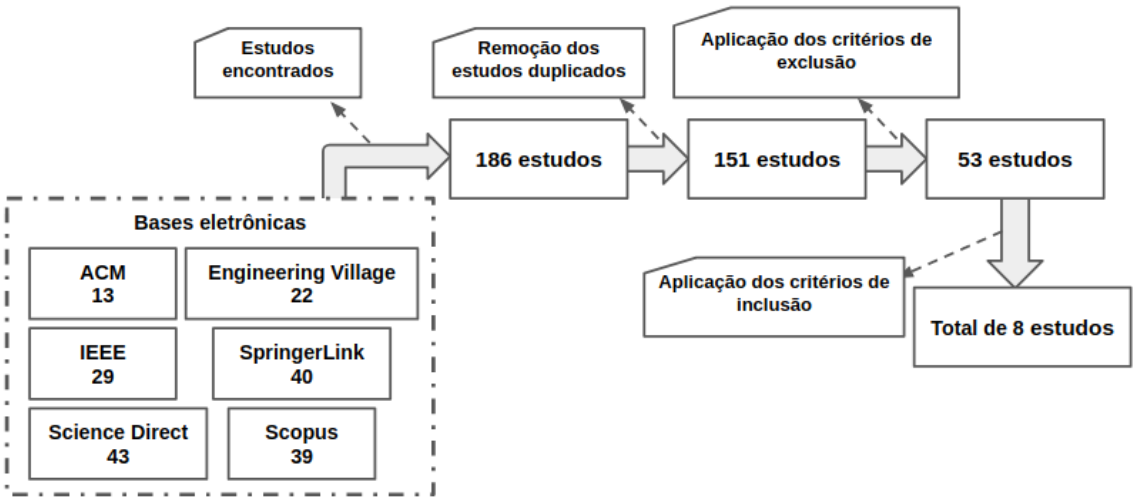


Figura 2. Ilustração do processo de seleção dos estudos.

4.1. Visão Geral dos Estudos

O estudo de [Silva and Lucena 2004], contém a primeira proposta da linguagem MAS-ML propriamente dita. Esta linguagem estendeu a UML baseada na descrição de propriedades estruturais e dinâmicas apresentadas em um *framework* conceitual chamado Taming Agents and Objects (TAO). Esta proposta inicial procurou identificar as características particulares de sistemas multi-agentes (SMA) que não eram cobertas pela UML. Assim, para sanar essa limitação, o estudo apresenta como solução um metamodelo UML estendido, descrevendo as novas metaclasses e estereótipos que podem ser usados para modelar as características dos agentes em um SMA.

A modelagem na MAS-ML foi dividida em dois aspectos, estruturais e dinâmicos. Os aspectos estruturais, propostos em [Silva and Lucena 2004], englobam os diagramas estáticos (diagramas de classes, organização e papéis). Enquanto os aspectos dinâmicos, propostos originalmente em [Silva and Lucena 2003] - um trabalho precursor da MAS-ML - envolvem os diagramas comportamentais (diagrama de sequência) e foram incorporados à MAS-ML para representar as interações entre os agentes.

A partir desses recursos, segundo [Silva and Lucena 2004], a MAS-ML seria capaz de abordar as características particulares de um SMA que não são suportadas pela UML, tornando-se possível representar abstrações associadas a um SMA e descrever as relações estáticas e dinâmicas entre essas abstrações. Mas apesar dos benefícios da

utilização da MAS-ML para modelagem de agentes, o estudo de [Silva and Lucena 2004], apenas se preocupava com os aspectos estruturais e parcialmente com os aspectos dinâmicos do sistema (apenas diagramas de sequência). Dessa forma, essa versão da linguagem não suportava a modelagem de planos associados aos objetivos dos agentes.

Em seu trabalho seguinte [Silva et al. 2004], estendeu o diagrama de sequência da MAS-ML, de maneira que a linguagem passasse a suportar a representação de envio e recebimento de mensagens FIPA [FIPA 2018] entre agentes. Posteriormente, no estudo de [Silva et al. 2005], o diagrama de atividade da UML foi estendido e incorporando aos diagramas dinâmicos da MAS-ML, para permitir a representação dos planos e ações dos agentes, além das funções desempenhadas pelos agentes durante a execução de um plano.

Após isso, no trabalho de [Silva et al. 2008a], foram definidas uma série de diretrizes para ajudar os projetistas a modelar orientação, interação, adaptação, distribuição, mobilidade e execução simultânea de objetivos relativas aos agentes. Em seguida [Silva et al. 2008b] apresenta com um maior grau de detalhamento, os elementos e mecanismos da MAS-ML descritos anteriormente nos estudos citados. Esse trabalho passou a ser considerado o modelo de referência padrão da linguagem, sendo utilizado por todas as extensões posteriores.

Em outro trabalho, [Freire et al. 2012], estendeu a MAS-ML de [Silva et al. 2008b] de forma a viabiliazer sua integração com a NormML [Figueiredo and da Silva 2010], uma linguagem de modelagem para a especificação de normas. Segundo [Figueiredo and da Silva 2010], a grande maioria dos agentes possuem total autonomia para realizar a execução de tarefas que lhe são determinadas. porém em alguns cenários é necessário restringir o comportamento dos agentes por meio de normas, definindo sanções e aplicando-as quando normas são violadas ou cumpridas. Dessa forma, visando possibilitar a modelagem de elementos estáticos de normas de agentes ainda em fase de projeto, [Freire et al. 2012] produziu uma nova versão da MAS-ML com suporte a modelagem de normas de agentes, chamada de NorMAS-ML. O estudo apresenta como principal limitação a realização apenas da extensão dos aspectos estruturais da MAS-ML, deixando em aberto a extensão dos aspectos dinâmicos de normas na MAS-ML.

Seguindo uma outra perspectiva, o estudo de [Adamzadeh et al. 2014], estendeu a MAS-ML a partir da versão de [Silva et al. 2008b], definindo conceitos relacionados à mitigação de danos em ambientes de resposta a emergências (EREs). Todavia, a extensão ainda necessita de melhorias porque apenas alguns dos conceitos do metamodelo para gerenciamento de desastres de [Othman et al. 2014], foram considerados na extensão.

Finalmente, em [Gonçalves et al. 2015] a linguagem MAS-ML foi evoluída para a MAS-ML 2.0, o que envolveu a extensão dos aspectos estáticos e dinâmicos propostos por [Silva et al. 2008b], de forma que, além das características particulares de SMAs, a linguagem também contemple a modelagem das arquiteturas internas dos agentes propostas por [Russell and Norvig 2016]. Considerando que um agente faz uso de suas percepções para obter informações sobre meio ambiente e sobre outros agentes, além de planejar uma sequência de ações para atingir um objetivo. Assim, tomando como base que a MAS-ML não contemplava essas caracteristicas dos agentes, foi realizada a definição de duas novas metaclasses, sendo elas: *AgentPlanningStrategy*, para representar a funcionalidade

de planejamento dos agentes, e *AgentPerceptionFunction*, responsável por representar a percepção dos agentes. Também foram definidos novos estereótipos para a representação do comportamento dos agentes nos diagramas da MAS-ML.

Após a extensão das metaclasses, os diagramas estáticos das MAS-ML sofreram modificações, sendo possível agora a representação de cinco estruturas de agentes baseados em arquiteturas, sendo eles: Agentes de Reflexo Simples, Agentes Baseados em Modelo, Agentes baseados em metas com estrutura de planejamento, Agentes baseados em metas com estrutura de plano e Agentes Baseados em Utilidade.

4.2. Respondendo às Questões

No que concerne à primeira pergunta "Quão abrangente é a MAS-ML para a modelagem de sistemas multiagentes?"), tomando como base os estudos encontrados, observamos que a MAS-ML original de [Silva et al. 2008b] suporta a modelagem e representação (I) das principais entidades de um SMA, apresentadas como propriedades estáticas como agentes, papéis de agente, organizações, sub-organizações e ambientes; (II) das propriedades dinâmicas, como a interação dos agentes por meio de troca de mensagens, o fluxo de execução dos planos e ações dos agentes. A MAS-ML 2.0 de [Gonçalves et al. 2015] estende a MAS-ML original com suporte a (III) modelagem das arquiteturas internas de agentes propostas tanto nos diagramas estáticos como dinâmicos. A NorMAS-ML de [Freire et al. 2012] estende a MAS-ML original com suporte a (IV) modelagem dos elementos estáticos das normas para agentes. A extensão de [Adamzadeh et al. 2014] estende a MAS-ML original com suporte a (V) modelagem dos aspectos estáticos dos conceitos relacionados à mitigação de danos em EREs.

Considerando que a modelagem de SMAs requer todas as características citadas e existem três extensões distintas, não existe hoje uma versão que suporte todos as propriedades necessárias para a modelagem de um SMA. Percebe-se portanto a necessidade de unir todas as versões em uma linguagem única.

Com relação à segunda pergunta, "Quais são as lacunas da MAS-ML para a modelagem de sistemas multiagentes?"), observamos que (I) a MAS-ML 2.0 ainda precisa sofrer outras extensões para fornecer suporte à modelagem de outras arquiteturas internas propostas pela literatura; (II) a MAS-ML ainda não possui recursos para a representação do aprendizado e a autonomia dos agentes em seus recursos dinâmicos; (III) as extensões de [Adamzadeh et al. 2014] e [Freire et al. 2012] apenas estenderam os aspectos estruturais da linguagem, sem incluir os aspectos dinâmicos.

Além disso, a MAS-ML e suas extensões não contém nenhum recurso específico para a modelagem de requisitos particulares para SMAs, tais como percepções, ações, objetivos e planos. Acreditamos que, como alternativa, se utilizaria para isto o diagrama de casos de uso da UML padrão sem qualquer extensão, mas não encontramos nenhum exemplo dessa aplicação. Finalmente, não encontramos nenhuma proposta de um processo de desenvolvimento para sistemas multiagentes que utilize a MAS-ML como notação.

5. Considerações Finais

Considerando o objetivo principal desse estudo, de identificar a abrangência e lacunas da MAS-ML, destacamos que a MAS-ML atualmente suporta parcialmente a modelagem

das principais entidades e arquiteturas internas de agentes de um SMA por meio de diagramas estaticos e dinâmicos, além de possibilitar a modelagem dos aspectos estaticos de normas e EREs.

Mas apesar dos benefícios, a linguagem ainda possui oportunidades de melhoria. Com base nos estudos dessa revisão, identificamos que atualmente a MAS-ML possui 3 versões disjuntas. De um ponto de vista de engenharia de software, isso é um grande problema, porque se torna difícil realizar a identificação de quais adaptações cada uma das linguagens implementou em sua versão. E mesmo que [Adamzadeh et al. 2014], em seu estudo utilize duas versões diferentes da linguagem MAS-ML em um mesmo exemplo de uso, ainda não é possível afirmar qual é a compatibilidade das linguagens entre si.

A partir destas conclusões, como trabalhos futuros, passamos à segunda etapa de nossa pesquisa (que já se encontra em andamento) durante a qual estamos aplicando a linguagem MAS-ML na modelagem de um sistema multiagente real com o objetivo de determinar se realmente a linguagem é adequada para a modelagem deste tipo de sistema e se ela possui mais limitações. Com isso almejamos determinar o que precisaria ser estendido nesta linguagem. Podemos destacar neste momento a falta de mecanismos para a modelagem de requisitos que esta linguagem apresenta, além da falta de um processo de desenvolvimento que suporte sua utilização.

Com relação à deficiência em modelar requisitos específicos para este tipo de sistema, poderíamos aplicar o metamodelo de Guedes [Guedes and Vicari 2012] com este objetivo, que foi projetado especificamente para a modelagem de requisitos de sistemas multiagentes e verificar sua compatibilidade com a MAS-ML. No que tange à falta de um processo, consideramos a sua elaboração muito importante, posto que é necessário estabelecer regras de como utilizar a notação da linguagem. Esta elaboração deverá também constituir um trabalho futuro.

Referências

Adamzadeh, T., Zamani, B., and Fatemi, A. (2014). A modeling language to model mitigation in emergency response environments. In *Computer e Knowledge Engineering (ICCKE), 2014 4th International eConference on*, pages 302–307. IEEE.

Biolchini, J., Mian, P. G., Natali, A. C. C., and Travassos, G. H. (2005). Systematic review in software engineering. *System Engineering and Computer Science Department COPPE/UFRJ, Technical Report ES*, 679(05):45.

Cervenka, R. and Trencansky, I. (2007). *The Agent Modeling Language-AML: A Comprehensive Approach to Modeling Multi-Agent Systems*. Springer Science & Business Media.

Figueiredo, K. and da Silva, V. T. (2010). Normml: a modeling language to model norms. In *1st Workshop on Autonomous Software Systems. Salvador, Brazil*.

FIPA (2018). Fipa agent management specification. `http://www.fipa.org/specs/fipa00023/SC00023K.html`, Acessado em 2018-09-26.

Freire, E. S. S., Cortés, Mariela Inés e Gonçalves, E. J. T., and Lopes, Y. S. (2012). Normas-ml-a modeling language to model normative multi-agent systems. In *ICEIS (2)*, pages 113–119.

Gonçalves, E. J. T., Cortés, M. I., Campos, G. A. L., Lopes, Y. S., Freire, E. S., da Silva, V. T., de Oliveira, K. S. F., and de Oliveira, M. A. (2015). Mas-ml 2.0: Supporting the modelling of multi-agent systems with different agent architectures. *Journal of Systems and Software*, 108:77–109.

Guedes, G. T. A. (2018). *UML 2 - Uma Abordagem Prática - 3a. Edição*. Novatec Editora.

Guedes, G. T. A. and Vicari, R. (2012). *Um metamodelo UML para a modelagem de requisitos em projetos de sistemas multiagentes*. PhD thesis, Universidade Federal do Rio Grande do Sul - UFRGS.

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

Odell, J., Parunak, H. V. D., and Bauer, B. (2000). Extending uml for agents. *Ann Arbor*, 1001:48103.

Othman, S. H., Beydoun, G., and Sugumaran, V. (2014). Development e validation of a disaster management metamodel (dmm). *Information Processing & Management*, 50(2):235–271.

Russell, S. J. and Norvig, P. (2016). *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,.

Silva, V. T., Choren, R., and De Lucena, C. J. (2004). A uml based approach for modeling and implementing multi-agent systems. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 914–921. IEEE Computer Society.

Silva, V. T., Choren, R., and De Lucena, C. J. (2008a). Modeling mas properties with mas-ml dynamic diagrams. In *Agent-Oriented Information Systems IV*, pages 1–18. Springer.

Silva, V. T. and Lucena, C. J. (2004). From a conceptual framework for agents and objects to a multi-agent system modeling language. *Autonomous Agents and Multi-Agent Systems*, 9(1-2):145–189.

Silva, V. T. and Lucena, C. J. P. (2003). *Extending the UML Sequence Diagram to model the dynamic aspects of Multi-Agent Systems*. PUC.

Silva, V. T., Noya, R. C., and de Lucena, C. J. (2005). Using the uml 2.0 activity diagram to model agent plans and actions. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 594–600. ACM.

Silva, V. T. D., Choren, R., and De Lucena, C. J. (2008b). Mas-ml: a multiagent system modelling language. *International Journal of Agent-Oriented Software Engineering*, 2(4):382–421.

Vicari, R. and Gluz, J. C. (2007). An intelligent tutoring system (its) view on aose. *International Journal of Agent-Oriented Software Engineering*, 1(3-4).

Estudo Exploratório no Refinamento de uma DSL para Versões Baseadas em EMF, EMF Forms e Angular

Esther Salgado Favero, Igor Ademilson de Oliveira,
Pedro Sebastian Zanella Nuñez, Fábio Paulo Basso

[1]Universidade Federal do Pampa (UNIPAMPA)
Engenharia de Software
Alegrete, Rio Grande do Sul, Brasil
{estherfavero97@gmail.com, igornnt@gmail.com ,
pedrosebastian90@gmail.com, fabiobasso@unipampa.edu.br}

Abstract. *This work presents an exploratory study of technologies for building Domain Specific Languages (DSLs) including EMF, EMF Forms and Angular. Our goal is to explore three versions of a DSL designed to represent business models created for the validation of startup models. Our domain model is conceptually directed to represent business plans for all areas of the labor market. This research performs a "black box"analysis, investigating in practice these technologies as a way of finding the best option for generation of a business plan management information system. Our conclusions are: 1) The development of systems based on DSL concepts requires elevated knowledge for domain analysis; 2) the tools investigated bring a facility for the integration of each technology, and; 3) Visual results are not pleasing to an end user, thus opening window for new research for usability.*

Resumo. *Este trabalho apresenta um estudo exploratório de tecnologias para a construção de Linguagens Específicas de Domónio (DSL), incluindo EMF, EMF Forms e Angular. Nosso objetivo é explorar três versões de uma DSL concebida para representação de modelos de negócio criados para a validação de modelos de startups. Nosso modelo de domínio é conceitualmente direcionado para representação de planos de negócios para todas as áreas do mercado de trabalho. Esta pesquisa realiza uma análise "black box", investigando na prática estas tecnologias como forma de encontrar a melhor opção para um sistema de informação gerencial de planos de negócio. Nossas conclusões são: 1) O desenvolvimento de sistemas com base em conceitos de DSL requer um nível elevado de conhecimento para análise de domínio; 2) as ferramentas investigadas trazem uma facilidade para a integração de cada tecnologia, e; 3) os resultados visuais não são agradáveis para um usuário final, portanto abrindo espaço para novas pesquisas em usabilidade.*

1. Introdução

Sabe-se que “startup” é o termo utilizado para definir uma empresa recém criada, que apresenta um modelo de negócios considerado inovador e que ainda não fora validado totalmente junto ao mercado (Giardino et al. 2014). Empresas novas no mercado tendem a ter inseguranças a respeito de produtos, serviços ou afins que a mesma desenvolve. Este

cenário de incerteza significa dizer que não há uma garantia de que a ideia ou o projeto da empresa dará certo.

Modelos de planos de negócio servem para guiar uma startup no desenvolvimento e gerenciamento de seu negócio. Voltados para a tomada de decisão, tais modelos são caracterizados por Sistemas de Informação Gerenciais (SIG) (Giardino et al. 2014), que são munidos de bancos de dados e estruturas para cadastro e busca de informações. Eles também são caracterizados por cenários dinâmicos cujos dados tendem à sofrer mudanças conforme a inovação é confrontada perante o tempo de mercado (Basso et al. 2015). Tendo isto em vista, é recomendada a utilização de SIGs para a elaboração e gestão de modelos para plano de negócios.

Usando-se de geração automática de código e refinamentos (Basso et al. 2016), SIGs podem ser produzidos por meio de técnicas e ferramentas para Prototipação Rápida de Aplicações (RAP). Este trabalho discute sobre a geração e refinamento de três diferentes versões de um SIG, cujo modelo de domínio é focado em negócios. PAra este domínio, é recomendável o uso de características de usabilidade como padrões de design (Welie 2018). Ou seja, levando-se em conta que o SIG deve oferecer aos empreendedores a possibilidade de criação de um modelo de negócio genérico que abranja todas as áreas de mercado de trabalho, investiga-se as características de usabilidade oferecidas por três ferramentas de RAP.

Nossa contribuição é uma descrição de benefícios e limitações observadas dessas ferramentas de RAP e que suportam a Engenharia Dirigida a Modelos (MDE) (Brambilla and Fraternali 2014). Tais ferramentas tem o intuito de utilizar modelos em diferentes níveis de abstração para a construção de Linguagens Especificas de Domínio (DSL) (Voelter 2009). Estes modelos são gerados automaticamente com transformadores de modelos (Basso et al. 2017) e refinados manualmente para aderir com alguns padrões de design (Welie 2018).

Este artigo é organizado da seguinte forma: na Seção 2 está disposta a Metodologia, discutindo a abordagem utilizada para o problema proposto. Na seção 3 apresenta-se as principais caracteríistias da DSL proposta. A Seção 4 relata a execução do estudo. A discussão que sumariza nossos achados é apresentada na Seção 5 e a Seção 6 expõe as lições aprendidas e respostas às questões de pesquisa.

2. Metodologia

Como parte de uma disciplina optativa de nosso curso de Engenharia de Software, denominada "Engenharia Dirigida por Modelos", explorou-se o desenvolvimento deste sistema sob a forma de uma DSL construída sobre o Eclipse Modelling Framework (EMF) (Steinberg et al. 2008). DSLs são comumente conhecidas na literatura como formas de se representar uma arquitetura de software usando-se de diagramas (Kelly and Tolvanen 2008). No entanto, para o domínio do problema proposto, a DSL deve ser construída sob a forma de um sistema de informaçao web (Brambilla and Fraternali 2014), como por exemplo, composta de um conjunto de telas que permitam a realização de tarefas do tipo CRUD (Create, Read, Update e Delete) e outras estruturas disponíveis em SIGs (Welie 2018).

Neste sentido, investigou-se alguns atributos de qualidade associados com algumas tecnologias de MDE que afetam a criação deste sistema como uma DSL. Buscamos

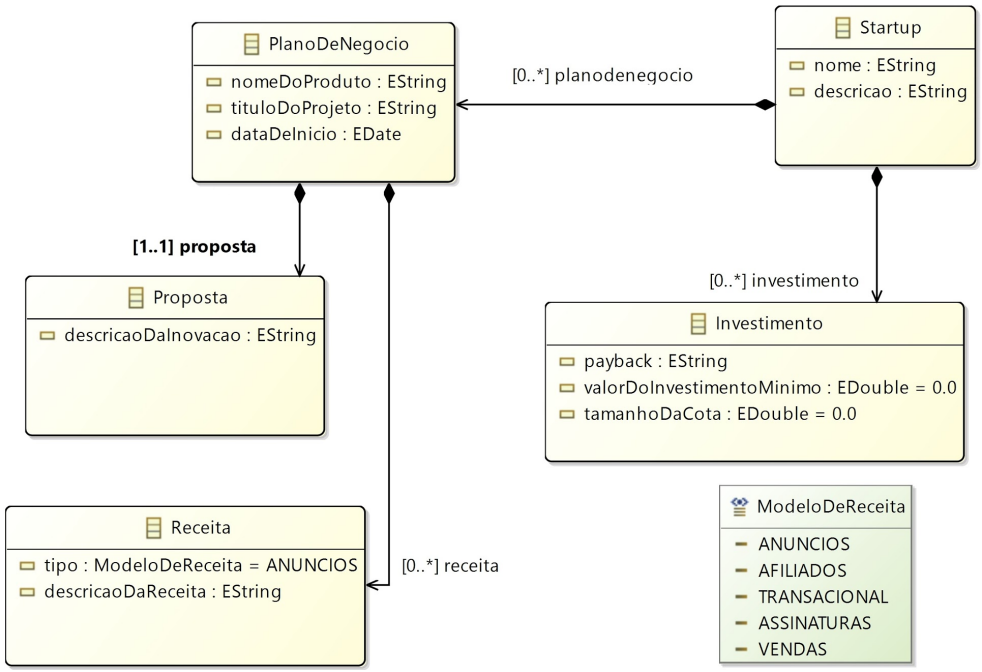


Figura 1. Primeira parte do diagrama de classes da DSL proposta

assim conduzir uma pesquisa exploratória do tipo "black-box"para responder às seguintes questões: 1) Como usuários inexperientes de tenologias para MDE, qual é a nossa percepção sobre as dificuldades na geração de uma DSL com a tecnologia EMF? 2) Uma vez que a DSL precisa ter o front-end semelhane aos sistemas de informações gerenciais, qual é a nossa percepção sobre as dificuldades na adaptação desta DSL com a tecnologia EMF Forms? e 3) Já que a DSL precisa ter a sua execução na web, atendendo um critério definido internamente como requisito não funcional desejável desse sistema, qual é a nossa percepção sobre as dificuldades na adaptação desta DSL com tecnologias para javascript como Angular 5?

3. Principais Características da DSL Proposta

O modelo de negócio proposto faz o uso do MDE com sua respectiva DSL, à qual é responsável por especificar o modelo de domínio através do EMF (Steinberg et al. 2008), uma ferramenta para metamodelagem. Neste mapeamento, foram selecionados artigos que tratavam de assuntos relacionados ao modelo de negócio. A partir deles, se embasou a construção do metamodelo parcialmente apresentado na Figura 1.

Visto que o trabalho trata da área “business”, a representação do modelo de domínio (metamodelo) também foi inspirado em uma ferramenta já existente, o MakeMoney <makemoney.starta.com.br/>. O MakeMoney é um software que permite uma empresa testar suas idéias de produto, analisar a possibilidade de inovação e aumentar sua competitividade no mercado.

4. Relato do Estudo Exploratório

Com base nestas questões de pesquisa, a construção da DSL foi feita utilizando-se de técnicas para MDE disponíveis com suporte ferramental na distribuição Eclipse Modelling

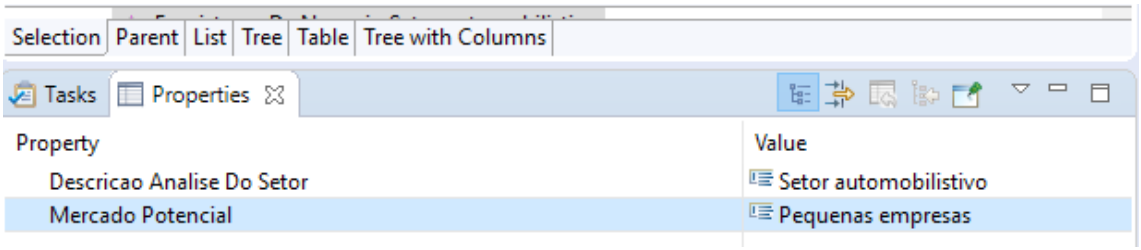


Figura 2. DSL Versão 1 como um plugin gerado pelo EMF

Tools (EMT) (Steinberg et al. 2008). O EMT é uma ferramenta que dispõe do modelador gráfico Ecore, isto é, um diagrama de classes utilizado na construção de metamodelos. Além disso, este é um ecosistma de ferramentas integradas de MDE, cujos plugins são usados para a geração de código Java para aplicativos construídos sobre o guarda-chuva do EMF.

Usando um ecosistema maduro para a criação de DSLs, nosso desafio de pesquisa resumiu-se à explorar diferentes tecnologias/plugins para a geração de três diferentes versões da DSL proposta, buscando aquela que melhor atende aos requisitos do sistema proposto. Além disso, no MDE, o aumento ou redução do nível de abstração é realizado através de transformações dos modelos que representam as versões da nossa DSL. Quer dizer, o modelo da DSL proposta, originalmente em nível independente de uma plataforma web, deve ser transformado em modelos de nível dependente de plataforma de execução web.

4.1. Geração da Versão 1

Utilizando o EMF, gerou-se o modelo de domínio mostrado na Figura 1. Então, transformou-se esta representação para uma DSL ilustrada na Figura 2. Esta figura apresenta o editor disponível no EMF para especificar os dados de um modelo de negócios. Além deste formulário, o EMF também disponibiliza uma árvore de elementos para o usuário final interagir sobre elementos do plano de negócio. Claramente, o resultado final não é amigável e não nos agradou, o que nos impulsionou à procurar por alternativas para agregar à esta DSL uma melhor usabilidade.

4.2. Transformações de Refinamentos da Versão 2

Em uma perspectiva de melhorar os aspectos visuais da DSL na Versão 1, foi utilizado o EMF Forms <www.eclipse.org/ecp/emfforms/> para a geração de uma Versão 2. O plugin do eclipse EMF Forms é uma ferramenta desenvolvida para a criação de interfaces de usuário para modelos EMF, mas cuja representação é baseada em formulários. Nesta pesquisa, encontramos um plugin do EMT que possibilita a geração do modelo conforme o metamodelo EMF Forms de maneira automática. Assim, o passo seguinte foi a utilização da ferramenta de editoração do EMF Forms, que permite que as interfaces sejam customizadas de uma maneira simples, sem codificação. Além disso, este aparato ferramental proporciona aparência e comportamento uniformes para operações do tipo CRUD. Portanto, o resultado final da versão 2 de nossa DSL é mostrado na Figura 3.

4.3. Estudo Black-box de Plugins Eclipse Para Web Front-End

Para se chegar até a tecnologia para criação dos formulários para a web foi realizada uma simples busca na internet utilizando o Google com ferramenta de pesquisa. Os cinco

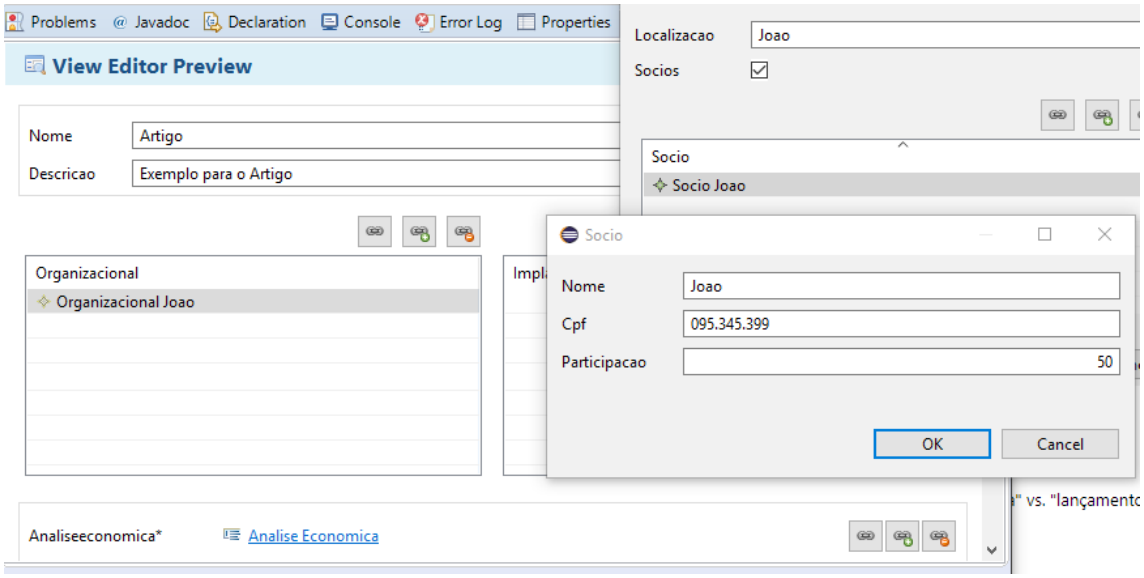


Figura 3. DSL Versão 2 como formulários gerados com base no EMF Forms

Tabela 1. Plugins Eclipe para a Execução de uma DSL em Ambiente Web

Plugin	URL do Plugin	Tipo
JSON Forms	https://eclipsesource.com/blogs/tutorials/emf-forms-and-json-forms-integration-guide/	SI & CRUD
Web Modeling Framework	https://projects.eclipse.org/proposals/web-modeling-framework	Diagrama
EMF REST	https://modeling-languages.com/emf-rest-restful-apis-from-models/	Diagrama
EMF Parsley	http://www.rcp-vision.com/emf-parsley-a-web-application-is-a-few-steps/	Árvore EMF
MPS da jetbrains	https://www.jetbrains.com/mps/	DSL Textual

resultados encontrados são apresentados na Tabela 1, que caracteriza cada opção para diferentes tipos de interação do usuário final com a DSL.

Através dessa análise black-box, chegou-se até a ferramenta JSON Forms, que é um framework que possibilita a geração de interfaces para Sistemas de Informação (SI) web baseadas em formulários. Tal framework utiliza arquivos em formato JSON para representar o esquema da interface e a estrutura da visão (view). Através da pesquisa foi descoberta a possibilidade de renderizar esses formulários utilizando frameworks e bibliotecas web de uma forma mais simples através de tags html personalizadas (web components).

Assim, selecionou-se o framework JSON Forms para o desenvolvimento da terceira versão da DSL proposta. Esses componentes web processam arquivos do JSON Forms e renderizam um formulário com estilo CSS, executando algumas validações. Por fim, as tecnologias encontradas na execução da DSL na web são: AngularJs - utilizando a biblioteca, Angular Schema Form; Angular 5 - com a biblioteca Angular JSON Schema Form; e o React.js - com a biblioteca react-jsonschema-form.

4.4. Transformações e Refinamentos da Versão 3

Por meio de um plugin disponível no EMT, foi possível gerar arquivos JSON (JSON Schemas), juntamente com as Views (UI Schemas) respectivas. Também gerou-se arqui-

vos com schemas JSON tendo como entrada os modelos conforme o EMF Forms. Como resultado, obteve-se os esquemas de formulário e de estrutura de view com base em JSON. Posteriormente, através desses arquivos JSON gerados à partir das View Forms da aplicação, foi realizado o refinamento manual dos artefatos gerados, criando a aplicação final com o framework de desenvolvimento de aplicações e interfaces para web Angular, como ilustrado na Figura 4.

Com o intuito de se realizar uma avaliação prática para se verificar a viabilidade de se construir interfaces web para o usuário, optamos por refinar o resultado final com o Angular 5 e suas tecnologias relacionadas. O motivo é que o mesmo apresenta-se como um framework robusto para a criação de aplicações corporativas e com tecnologias atuais. Assim, o AngularJs foi desconsiderado, pois foi descontinuado pelos mantenedores em detrimento do novo Angular. A biblioteca React é também uma biblioteca moderna, sendo hoje a mais utilizada do mercado para dentre as demais concorrentes, porém seu foco não é para criação de aplicações comerciais, mas sim para criação de interfaces.

4.5. Resultado Final

Como lição aprendida, observamos que essas tecnologias integradas trazem a possibilidade de se criar de forma rápida formulários sem que o desenvolvedor invista muito tempo na tarefa de programação. Com uma simples tag html e alguns atributos é possível renderizar na tela formulários razoavelmente complexos. Um exemplo da utilização do componente web para renderização é mostrado na Figura 4.

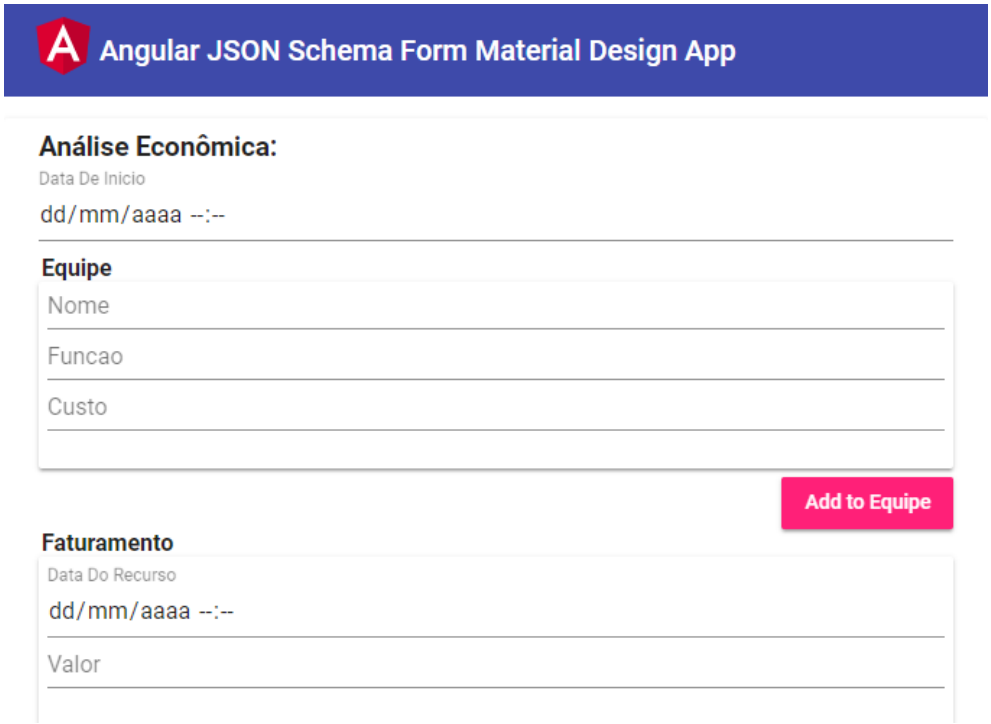


Figura 4. DSL Versão 3 como um formulário web gerado com base em Angular e JSON Forms

5. Discussões

Com base no trabalho realizado, observa-se a facilidade com que se pode construir uma DSL. Entretanto, várias questões no que tange a qualidade de software ainda não foram bem abordadas, como a questão da experiência do usuário e, talvez, questões de arquitetura e integração entre sistemas diversos. As aplicações geradas através do EMF Forms são difíceis de serem utilizadas em cenários reais e a solução para gerar formulários web, nesse momento, apenas pode dar uma produtividade maior na construção de parcelas de um determinado sistema.

Porém, mesmo essa questão precisa ser bem avaliada. A tarefa de criação de simples formulários web normalmente é delegada a desenvolvedores front-end, que geralmente não tem conhecimento sobre criação e modelagem de domínio, essencial para a criação de DSLs. Em um contexto empresarial, a tarefa de criação de formulários com base no EMF Forms exigiria algum treinamento, ou que exista um profissional capacitado e especializado, o que talvez não seja viável para projetos pequenos.

Esperava-se, a princípio, que para a execução de nosso estudo, já existissem bibliotecas, frameworks ou softwares capazes de gerar sistemas web com base em EMF, com pouca ou nenhuma programação. Para levar nossa DSL da versão 2 para a versão 3, muito esforço manual foi necessário. Assim, existe um gap de pesquisa que merece ser investigado na transformação automática de modelos EMF Forms para ambiente web. Também, é preciso uma forma de representar os formulários mais ricos do que os formatos existentes no EMF Forms, como os padrões de projeto de desenho e interação (Welie 2018).

Essa pilha de tecnologias necessárias para a execuçãoda DSL na web também é um desafio de pesquisa. É importante que se gere o código completo para que as tarefas de refinamento não sejam tão demoradas. Esta pilha de tecnologias do EMT auxilia times na realização de tarefas mais triviais, poupando esforços e recursos para funcionalidades chave da DSL. No entanto, é preciso a geração de interfaces com maior usabilidade.

Outra questão importante é que realmente não existem ferramentas de fácil utilização para modelagem de negócios e validação de startups no mercado. Porém, por ser um problema recorrente para a maioria dos empreendedores, a criação de tal sistema seria um bom investimento. Por ser uma área bem conhecida e estruturada, criá-la como uma DSL torna-se completamente viável, necessitando de um time de bons engenheiros e de especialistas de domínio, além de melhorias nas pesquisas para a transformação para diferentes versões de DSLs rodando em diferentes plataformas.

6. Conclusão

Concluímos, com base num estudo exploratório, que a geração e refinamento de sistemas de informações, baseados em DSLs para formulários, pode ser um grande desafio em termos de engenharia. O sucesso para desenvolvimento das diferentes versões da DSL explorada exigem profissionais treinados e uma grande proximidade com especialistas de domínio. No entanto, isto caracteriza um grande potencial de pesquisa e inovação para formentar o crescimento neste mercado. O motivo é que as tecnologias estudadas trazem a possibilidade de se resolver problemas específicos e com real foco no domínio.

As tecnologias investigadas são devotadas para um dos grandes desafios das tecnologias utilizadas atualmente no mercado: o desenvolvimento rápido de sistemas orien-

tados à objetos. Assim, respondendo as questões de pesquisa voltadas para este desafio, pode-se destacar que: 1) Apesar das facilidades trazidas pelo EMT e pelo EMF, ainda não há um bom suporte para a construção de soluções baseadas na diversidade de padrões de design e interação em (Welie 2018); 2) Neste sentido, as interfaces de usuário geradas e refinadas com EMF Forms não são visualmente agradáveis e tem pouco foco em questões como usabilidade e experiência de uso e, o mais importante; 3) Por fim, as tecnologias disponíveis não possibilitam de forma eficiente e robusta a geração de interfaces web para javascript, o que em caso contrário motivaria uma maior utilização para tais recursos.

Como um trabalho futuro, pretende-se desenvolver uma alternativa para as transformações de modelos EMF para ambientes web. Para tanto, no lugar de EMF Forms, pretende-se utilizar a ferramenta MockupToME (Basso et al. 2016), que permite elaborar formulários com uma interação mais rica, e o desenvolvimento de transformações customaizáveis para múltiplas combinações de tecnologias (Basso et al. 2017).

Por fim, acredita-se que a execução deste trabalho, além de nos proporcinar uma visão panorâmica do estado da arte disponível no ecossitema do Eclipse para geração de DSLs para SIGs, também abre novos estudos visando migrar automaticamente as DSLs desenvolvidas em nosso grupo de pesquisa que possuem esta mesma característica. Destaca-se o RAS++, uma DSL para a representação de ativos para negócios envolvendo a transferência de tecnologia em cenários de coopetição na Engenharia de Software.

Referências

[Basso et al. 2017] Basso, F. P., Oliveira, T. C., Werner, C. M., and Becker, L. B. (2017). Building the foundations for 'mde as service'. *IET Software*, 11:195–206.

[Basso et al. 2016] Basso, F. P., Pillat, R. M., Oliveira, T. C., Roos-Frantz, F., and Frantz, R. Z. (2016). Automated design of multi-layered web information systems. *Journal of Systems and Software*, 117:612 – 637.

[Basso et al. 2015] Basso, F. P., Pillat, R. M., Roos-Frantz, F., and Frantz, R. Z. (2015). Combining mde and scrum on the rapid prototyping of web information systems. *International Journal of Web Engineering and Technology*, 10(3):214–244.

[Brambilla and Fraternali 2014] Brambilla, M. and Fraternali, P. (2014). Large-scale model-driven engineering of web user interaction: The webml and webratio experience. *Science of Computer Programming*, 89, Part B:71 – 87. Special issue on Success Stories in Model Driven Engineering.

[Giardino et al. 2014] Giardino, C., Unterkalmsteiner, M., Paternoster, N., Gorschek, T., and Abrahamsson, P. (2014). What do we know about software development in startups? *Software, IEEE*, 31(5):28–32.

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

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

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

[Welie 2018] Welie (2018). Patterns and interaction design <http://www.welie.com/patterns/>. Acessado em: 10 de Julho de 2018.

Anais da 2ª Escola Regional de Engenharia de Software (ERES 2018)
em Dois Vizinhos, PR, Brasil, 22 a 24 de outubro de 2018

A Summary of Toolboxes Scoping MDWE Front-Ends

Jean T. Piagetti, Maurício El Uri, Fábio Paulo Basso

[1]Universidade Federal do Pampa
(UNIPAMPA) – Alegrete, RS – Brazil

{mauricioeluri,jeanpiagetti}@gmail.com, fabiobasso@unipampa.edu.br

Abstract. *This paper characterizes 27 studies, published between 2005 and 2017, scoping the development of web front-ends through toolboxes for Model-Driven Web Engineering (MDWE). It addresses Domain Specific Languages (DSLs), embedded with some design techniques, devoted to help in the various phases of software development. Moreover, it also contextualizes model-based tools for rapid prototyping, with evolutive prototyping and source code generation, and highligths those that speed-up the analysis process through automated design. Our conclusion is that only six of these proposals have been reported to be used in industry, thus a limitation found in the literature of the area that hinders the understanding of the acceptance criteria of these tools in contexts of software factories.*

Resumo. *Este artigo caracteriza 27 estudos, publicados entre 2005 e 2017, endereçando o desenvolvimento de web front-ends por meio de ferramentas para Model-Driven Web Engineering (MDWE). Ele aborda Linguagens Específicas de Domínio (DSLs), com o uso de técnicas de design, propostas para auxiliar em muitas fases de desenvolvimento de software. Também aborda quais destas ferramentas são caracterizadas para prototipagem rápida de aplicações, com prototipação evolutiva e geração de código fonte, que tem como objetivo acelerar o processo de de análise por meio de design automático. Nossa conclusão é que apenas seis destas propostas têm sido relatadas com aplicação na indústria, assim uma limitação encontrada na literatura da área que atrapalha a compreensão dos critérios de aceitação destas ferramentas em contextos de fábricas de software.*

1. Introdução

Temos visto cada vez mais esforços para tornar o conteúdo presente na internet acessível aos usuários finais. Porém, mesmo com todos estes esforços, poucos programadores e equipes de desenvolvimento realmente se preocupam em tornar seus websites acessíveis em múltiplas plataformas [Brambilla and Fraternali 2014, Basso et al. 2017]. Por exemplo, além do conhecimento técnico necessário para desenvolvimento de sistemas de informação [Basso et al. 2016], desafios para incorporar em aplicativos as novidades do mercado incluem, principalmente, a dificuldade das equipes de desenvolvimento de cumprir com o cronograma em iterações de um processo de desenvolvimento cada vez mais curtas [Basso et al. 2015]. Por conta disso, importantes características de um sistema web como acessibilidade, facilidade de uso e navegabilidade acabam ficando de lado em prol de cumprir com o escasso tempo de mercado.

Neste sentido, ferramentas para Prototipação Rápida de Aplicações (RAP) [Elkoutbi et al. 2006, Planas et al. 2009] podem contribuir para superar este desafio. Algumas destas ferramentas permitem, por meio de modelos, a geração de sistemas conforme múltiplas plataformas [Basso et al. 2016]. Para tanto, Model-Driven Engineering (MDE) [Kent 2002] é um paradigma para o desenvolvimento de software baseado em transformações de modelos, e vem sendo implementado com algumas técnicas de design que contribuem para a RAP.

Em processos típicos baseados em MDE, as transformações de modelo devem receber um modelo altamente detalhado para gerar partes funcionais de aplicativos [Schmidt 2006]. Para gerar o código-fonte completo [Kelly and Tolvanen 2008], várias partes de um design de aplicativo são detalhadas em DSLs (Domain Specific Languages) [Voelter 2009] e/ ou decoradas com anotações adicionadas aos elementos de modelo representados com a UML (Unified Modeling Language) [Elkoutbi et al. 2006], uma linguagem de modelagem de propósito geral usada em projetos arquiteturais de alguns sistemas web. Em qualquer caso, isso torna a construção do software dependente de tarefas de design.

Neste contexto, nosso objetivo é descrever 13 anos de pesquisa no design de front-end web focando em Model-Driven Web-Engineering (MDWE). MDWE inclui desde abordagens para arquitetura de redes de computadores ao desenvolvimento de aplicativos web [Brambilla and Fraternali 2014]. Assim, focamos este estudo nas DSLs propostas para design de front-ends web e nas abordagens que automatizam o design por meio de transformações de modelos [Basso et al. 2017]. Para tal, um protocolo de pesquisa foi elaborado, e leva em consideração os aspectos que englobam o conjunto dos elementos do MDWE. No entanto, por motivos de espaço, o protocolo do estudo foi omitido deste relato.

A seguir apresenta-se o resultados do nosso mapeamento. Inicialmente, a Seção 2 apresenta um embasamento teórico. A Seção 3 apresenta nossos achados deste mapeamento. Finalmente, a Seção 4 apresenta nossas conclusões e trabalhos futuros.

2. Embasamento Teórico

No desenvolvimento de sistemas de informações da web, os front-ends da web como o layout são compostos pelos componentes para GUI (Graphical User Interface) e por diagramas comportamentais [Nunes and Schwabe 2006]. Para permitir a geração de código-fonte completo com uma abordagem MDWE, esses modelos são decorados com a semântica para ações do usuário, fluxos de tela e lógica de negócios. Assim, usando-se uma DSL apropriada, é possível abstrair detalhes de implementação, focando na especificação de semântica em modelos que formalizam o conhecimento sobre requisitos de software.

O MDWE permite o desenvolvimento de sistemas de informação para múltiplas plataformas por meio de transformações de modelos [Basso et al. 2016]. Outro emprego comum, por exemplo, é no desenvolvimento de serviços que integram múltiplos sistemas (mashups), usando uma arquitetura orientada a serviço (SOA) e também micro-serviços [de Vrieze et al. 2011]. Por fim, outro bom exemplo está na automação de processos de negócios, realizada por meio de modelos representados com BPMN [Pillat et al. 2015].

Uma ampla variedade de abordagens para MDWE foram propostas como produtos derivados de pesquisa desde 2000 [Ceri et al. 2000]. Atualmente, a falta de um mapeamento das propostas existentes gera uma incerteza sobre o que cada ferramenta oferece para a indústria de software, bem como quais são as lacunas de pesquisa. Assim, este estudo apresenta um mapeamento de propostas para MDWE que introduzem elementos representacionais para o design de GUIs que levam à geração de protótipos. Este mapeamento é novo, já que estudos existentes focam em plataformas específicas como AJAX [Mesbah and Van Deursen 2008], com análise de propostas e desafios sobre frameworks mais utilizados. Diferentemente, nosso estudo foca em tecnologia existente para gerar código específico de plataforma por meio de modelos independentes delas.

3. O Estado da Arte em Design de Web Front-Ends

A Tabela 1 apresenta os artigos encontrados em ordem cronológica. A seguir, descrevemos algumas propostas de design e prototipação consideradas nas abordagens para o MDWE publicadas entre 2013 e 2017.

3.1. Propostas Entre 2005 e 2012

Algumas propostas para o MDWE usam a UML como linguagem de especificação (S01, S03-S05) permitindo que Designers especifiquem modelos em várias camadas (também conhecido como *multiview design*). Este modelos representam componentes para Interfaces Gráficas de Usuário (GUI), fluxos, lógica de negócios, informações estruturais para bancos de dados. Exemplos de elementos projetados dentro de modelos UML são diagramas de casos de uso, diagramas de classes, diagramas de sequência e atividade, todos designados com tags de Perfis UML simples. Em S02 e S07 é apresentado o refinamento de modelos independentes de plataforma (como diagramas de classes anotados, casos de uso, diagramas de colaboração e diagramas de atividades) para específicos de tecnologias web. Além disso, é possível usar esses modelos para gerar sistemas de informações da Web de várias camadas e transformá-lo para abstrações de nível inferior que mapeiam elementos anteriores para o código-fonte.

Em termos representacionais, pode-se citar o estudo S06, que apresenta uma proposta de arquitetura para aplicações AJAX, chamada SPIAR. Esta arquitetura foi baseada na análise de vários frameworks AJAX e configurações levando em conta os problemas de design e limitações do mesmo. Detalhes como propriedades arquiteturais, interatividade com o usuário, latência percebia pelo usuário e desempenho da rede também foram levados em conta, assim como vários outros pontos referentes à qualidade do software.

Outra característica das propostas publicadas entre 2005 e 2012 é o foco na geração de código. Ou seja, nas pesquisas deste intervalo valorizava-se a geração de código por meio de modelos, sendo esta uma novidade. Por exemplo, tendo percebido que lojas que vendem produtos on-line geralmente precisam manter blogs para que seus consumidores possam discutir sobre os itens que fazem parte de seu catálogo, ou mesmo para manter uma facilitada comunicação com o consumidor, autores em S14 desenvolveram uma DSL para suprir esta demanda. Utilizando o Blojsom para geração dos blogs, esta DSL é composta por vários modelos, cada um com sua responsabilidade: estilização, conteúdo, navegação, etc. Apesar de ser capaz de gerar blogs inteiros, necessitando de poucos acabamentos pela equipe de desenvolvimento, esta DSL foi proposta em 2010, em

Tabela 1. Propostas para design e geração de front-end web no MDWE

Id	Título	Abordagem	RAP Approach
S01	Vanderdonckt, J. A mda-compliant environment for developing user interfaces of information systems. In 17th international conference on Advanced Information Systems Engineering	Manual Design	Mockup Model to Code
S02	Nunes, D. A. and Schwabe, D.Rapid prototyping of web applications combiningdomain specific languages and model driven design. In 6th international conferenceon Web engineering	Manual Design	Mockup Model to Code
S03	Elkoutbi, M., Khriss, I., and Keller, R. K. Automated prototyping of user interfaces based on uml scenarios. Automated Software Eng.	Manual Design	Use Case Model to Mockup Model
S04	Kavaldjian, S. A model-driven approach to generating user interfaces. In The 6th Joint Meeting on European software engineering conference and the ACM SIGSOFT symposium on the foundations of software engineering: companion papers	Manual Design	Mockup Model to Code
S05	Souza, V. E. S., Falbo, R. D. A., and Guizzardi, G. A uml profile for modeling framework-based web information systems. In 12th International Workshop on Exploring Modelling Methods in Systems Analysis and Design EMMSAD2007	Manual Design	Multi-layered Model to Code
S06	Mesbah, A. and Van Deursen, A. A component-and push-based architectural style for ajax applications. Journal of Systems and Software	Manual Design	Runtime MDWE
S07	Melia, S., Gomez, J., Perez, S., and Diaz, O. A model-driven development for gwt-based rich internet applications with ooh4ria. In Web Engineering, 2008.	Manual Design	Mockup Model to Code
S08	Sadat-Mohtasham, S. H. and Ghorbani, A. A. A language for high-level description of adaptive web systems. Journal of Systems and Software	Automated Design	Multi-Layered Model to Code
S09	Yang, F., Gupta, N., Botev, C., Churchill, E., Levchenko, G., and Shanmugasundaram, J. Wysiwyg development of data driven web applications. Proc. VLDB Endowment.	Manual Design	Mockup Design
S10	Ginzburg, J., Rossi, D. D. G., and Urbieta, M. Oblivious integration of volatile functionality in web application interfaces. Journal of Web Engineering	Manual Design	Mockup to Code
S11	Chavarriaga, E. and Macías, J A. A model-driven approach to building modern semantic web-based user interfaces. Advances in Engineering Software	Manual Design	Mockup to Code
S12	De Vrieze, P., Xu, L., Bouguettaya, A., Yang, J., and Chen, J. Building enterprise mashups. Future Generation Computer Systems	Manual Design	Multi-Layered Model to Code
S13	Aquino, N., Vanderdonckt, J., and Pastor, O. Transformation templates: adding flexibility to model-driven engineering of user interfaces. In 2010 ACM Symposium on Applied Computing	Manual Design	Mockup to Code
S14	Díaz, O. and Villoria, F. M. Generating blogs out of product catalogues: An mde approach. Journal of Systems and Software	Manual Design	Mockup to Code
S15	Busch, M., Koch, N., Masi, M., Pugliese, R., and Tiezzi, F. Towards model-driven development of access control policies for web applications. In Workshop on Model-Driven Security	Manual Design	Mockup to Code
S16	Vara, J. M. and Esperanza, M. A framework for model-driven development of information systems: Technical decisions and lessons learned. Journal of Systems and Software	Manual Design	Multi-layered Model to Code
S17	Rivero, J. M., Grigera, J., Rossi, G., Luna, E. R., and Koch, N. Towards agile model-driven web engineering. In IS Olympics: Information Systems in a Diverse World	Manual Design	Mockup to Code
S18	Molina, A. I., Giraldo, W. J., Gallardo, J., Redondo, M. A., Ortega, M., and Garc. Ciat-gui: A mde-compliant environment for developing graphical user interfaces of information systems. Advances in Engineering Software	Manual Design	Multi-Layered Model to Code
S19	Grigera, J., Rivero, J., Luna, E. R., Giacosa, F., and Rossi, G. From requirements to web applications in an agile model-driven approach. In Web Engineering	Manual Design	Mockup to Code
S20	Deufemia, V., D'Souza, C., and Ginige, A. Visually modelling data intensive web applications to assist end-user development. In 6th International Symposium on Visual Information Communication and Interaction	Automated Design	Runtime MDWE
S21	Batory, D., Latimer, E., and Azanza, M. (2013). Teaching model driven engineering from a relational database perspective. In 16th International Conferenceon Model Driven Engineering Languages and Systems	Manual Design	Physical Model to Mockup to Code
S22	Basso, F. P., Pillat, R. M., Frantz, R. Z., and Rooz-Frantz, F. Assisted tasks to generate pre-prototypes for web information systems. In 16th International Conferenceon Enterprise Information Systems.	Automated Design	Domain Model to Mockup to Multi-Layered MVC to Code
S23	Brambilla, M. and Fraternali, P. Large-scale model-driven engineering of web user interaction: The webml and webratio experience. Science of Computer Programming	Manual Design	Multi-layered Model to Code
S24	Basso, F. P., Pillat, R. M., Roos-Frantz, F., and Frantz, R. Z. Combining mde and scrum on the rapid prototyping of web information systems. International Journal of Web Engineering and Technology	Automated Design	Domain Model to Mockup to Multi-Layered MVC to Code
S25	Antonelli, H. L., da Silva, E. A. N., and Fortes, R. P. M. (2015). A model-driven development for creating accessible web menus. Procedia Computer Science	Automated Design	Mockup to Code
S26	Basso, F. P., Pillat, R. M., Oliveira, T. C., Roos-Frantz, F., and Frantz, R. Z. Automated design of multi-layered web information systems. Journal of Systems and Software	Automated Design	Domain Model to Mockup to Multi-Layered MVC to Code
S27	Chavarriaga, E., Jurado, F., and Díez, F. (2017). An approach to build xml-based domain specific languages solutions for client-side web applications. Computer Languages, Systems & Structures	Automated Design	Mockup to Code

um momento que a tecnologia de blogs ainda era imatura, porém Blojsom já está defasado, o que acaba dificultando a utilização da DSL pela comunidade atualmente. Portanto, para terem longevidade, DSLs devem ser construídas de modo agnóstico às plataformas.

3.2. Prototipação Rápida de Aplicações

Trabalhos S02, S05, S13, S14, S17, S22-S24 e S26 permitem especificar a semântica para lógica, habilitando a geração de código fonte completo para todas as camadas de aplicações. Essas contribuições são importantes porque geram protótipos funcionais de sistemas de informações da web que são testáveis pelo desenvolvedor.

Nesse sentido, estudos S17 e S25 afirmam que não é possível garantir que os protótipos gerados através da UML atendam às necessidades do cliente. Eles sugerem o desenvolvimento orientado a mockup, com transformações iniciadas a partir de mockups como o ponto chave para melhorar o feedback do cliente em fases preliminares de um

processo de software. Uma vez que se verifica a aceitação de um determinado protótipo funcional, isto dá uma confiança maior do que validar requisitos usando protótipos de papel. Isto não significa que o protótipo apresentado é o produto final de uma Sprint. Logo, o mesmo deve ser refinado ou em código ou em modelo ao longo da execução de uma Sprint.

Apesar de muitas propostas permitirem a geração de protótipos funcionais por meio de modelos, poucas são adequadas para a prototipação rápida de aplicações. Então há uma diferença entre ferramentas aptas à prototipação e outras aptas para a prototipação rápida. Por exemplo, no estudo S17 os autores anotam manualmente os mockups enquanto propostas recentes usam técnicas de design automatizadas para acelerar a produção de modelos. Da mesma forma, o estudo S11 propôs a geração de front-ends da web em modelos usando algumas técnicas de design automatizado, mas como front-ends da web e não são anotados com a semântica que permite a geração de protótipos totalmente implementados. O estudo S20 também gera modelos MVC através de mockups anotados. Apesar de ser caracterizado como rápido, os modelos são projetados em uma abordagem descartável do MDWE (chamada de throwaway prototyping), e portanto não usado em todo o processo de desenvolvimento de software por meio de refinamentos.

A seguir, apresenta-se de que modo os trabalhos mais recentes resolvem esta limitação.

3.3. Propostas Recentes

Essas propostas publicadas entre 2005 e 2008 aplicam uma abordagem simples de design. Ou seja, elas repassam a responsabilidade para detalhamento do modelo para o usuário final. Adotando uma abordagem de detalhamento de cima para baixo (detalha-se um modelo abstrato e transforma-se este num modelo específico de plataforma), é possível modelar semântica de ações em componentes GUI de sistemas de informação na web. Por exemplo, modelos conceituais são altamente detalhados com semântica de ações antes de gerar um protótipo testável. A mesma abordagem manual para refinamento de modelos também é encontrada em abordagens mais recentes como S17 e S21.

Tal abordagem é ruim para execução em metodologias ágeis. Assim, somente quando o modelo de entrada é validado para transformações, é possível transformá-lo em código-fonte. Propostas mais recentes pós 2007 são tentativas de usar as práticas que requerem tetes mais imediatos, como técnicas de design automatizado (S08, S18, S25) e combinações entre MDWE e Agile (S17, S19, S22 e S24). Estas últimas sugerirem que o Engenheiro comece gerando Mockups (modelos de GUI anotados projetados com um DSM em vez de UML).

O estudo S08, publicado em 2008, apresenta um exemplo de suporte ferramental para apoio ao design automatizado de GUI. Trata-se de uma proposta para sistemas adaptativos voltados para web numa abordagem interativa, que permite montar as páginas web na medida que o usuário vai navegando na aplicação. Isso é conhecido na literatura como prototipação evolutiva de software e permite ao desenvolvedor direcionar o contexto da aplicação conforme os requisitos são descobertos. Além disso, a DSL foi projetada com base na análise de sistemas web-adaptativos com ênfase em engenharia de software baseada em linhas de produto. Trata-se de uma contribuição que permite ao designer especificar um mockup e definir um contexto, através de um UML Profile, fazendo

o mapeamento da interação num contexto adaptativo.

Para visualizar e modificar protótipos gerados a partir do MDWE, o estudo S18 apresenta a ferramenta denominada CIAT-GUI. O diferencial do suporte ferramental é que isto permite testar modelos de sistemas de informação em diferentes níveis de abstração de projeto. Esses níveis exigem o uso de técnicas de design automatizadas, portanto auxiliando na execução de prototipação evolutiva do software.

A esse respeito, podemos destacar também uma recente pesquisa experimental S21. Com o objetivo de permitir que alunos não experientes automatizem o design de aplicativos web, os autores consideraram uma perspectiva diferente para o design: perspectiva do banco de dados em que os modelos são, de fato, tabelas e relacionamentos. Esta pesquisa mostrou que os alunos produzem melhores resultados se usando ferramentas simples (por exemplo, transformações de modelos codificados em Java). Isso permite que pessoas não experientes automatizem tarefas de design. Além disso, a pesquisa sugere que automatizar o design de protótipos é viável no contexto de fábricas de software, já que faltam especialistas no MDE.

O desenvolvimento de DSLs para modelagem de elementos específicos para acessibilidade ainda desperta o interesse na pesquisa. Por exemplo, o estudo S25 propõe uma DSL que utiliza os pontos de referência do WCAG 2.0 (Web Content Acessibility Guidelines). Chamada de AMenu - Acessible Menu, esta DSL é capaz de gerar toda a estrutura de menus acessíveis para a web, facilitando assim o refinamento do modelo e a manutenção da aplicação por meio de geração de protótipos. A DSL apresentada é muito simples, é textual e apresenta integração com muitas das IDE's utilizadas atualmente pelos desenvolvedores, inclusive o Eclipse.

Enquanto as demais ferramentas foram construídas utilizando Eclipse-EMF ou UML Profiles, o trabalho mais recente (S27) trata de uma DSL textual programada utilizando Javascript. Ela foi criada para o desenvolvimento de sistemas web, sendo dividida em vários modelos, onde cada um é responsável por determinada camada de aplicação. Baseada em XML, ela é capaz de gerar todo código HTML3, CSS3 e Javascript necessário, sendo a geração de Javascript um dos diferenciais da ferramenta. Portanto, exceto pela novidade ser o desenvolvimento da própria DSL, ela permite ao modelador o uso da mesma semântica de ações já introduzida pelos trabalhos anteriores.

3.4. Aplicação em Contextos da Indústria

Outro ponto que investigamos foi como estas tecnologias vem sendo utilizadas na indústria. Encontramos que, dentre 27 artigos, apenas seis das pesquisas apresentam estudos de viabilidade das ferramentas em contextos industriais: S09, S20, S22-S24, S26. Cabe ressaltar que os estudos 22-24 e 26 apresentam relatos de aplicação em mais de um contexto industrial. Ou seja, são relatados em projetos de software de vários domínios, cujas tecnologias utilizadas para implementação também variam em termos de APIs, padrões arquiteturais e de codificação. Portanto, são poucos os casos em que o objetivo de se obter independência de plataformas é comprovadamente observado em projetos de software executados por meio de MDWE.

Essa observação é um tanto paradoxal para este tipo de pesquisa. Em geral, as abordagens de MDWE são motivadas sobre o guarda-chuva de desenvolvimento multiplataforma (independente delas), mas a maioria dos relatos (24) não corroboram com

sua comprovada efetividade na prática como uma solução deste problema. Um dos motivos por esta pequena amostra, i.e., de três relatos de contextos envolvendo desenvolvimento multi-plataforma, pode ser o fato de que artigos tratando de DSLs oferecem uma contribuição focada em termos de abstração, e não de prática. Para remover esta limitação, que nos impede de realizar uma conclusão definitiva sobre a efetividade das DSLs na prática, é preciso realizar uma segunda revisão do tipo snowballing. Por meio dela, espera-se encontrar os trabalhos dos mesmos autores dos 27 artigos relatados, mas que apresentem estudos observacionais complementares.

4. Conclusões e Trabalho Futuros

Este artigo apresentou um mapeamento de literatura em MDWE front-ends, caracterizando de forma resumida o total de 27 propostas. Por meio deste mapeamento, encontramos que a pesquisa dirigida para características estruturais de DSLs, como design de componentes que embutem semântica de ações neste contexto, é madura. Ou seja, em termos de representatividade, as DSLs existentes cobrem, se não todas, grande parte das necessidades para design de componentes de web front-ends das plataformas de desenvolvimento atuais. No entanto, as abordagens para design automatizado ainda estão no seu início. Portanto, pretende-se ampliar este estudo para os seguintes complementos:

1) Um consenso observado nos estudos é que DSLs para representação de front-ends web são essenciais para desenvolvimento rápido de aplicações em multi-plataformas. Além disso, DSLs construídas com base em design automático permitem capturar as necessidades do cliente nas fases preliminares de desenvolvimento de software, o que é importante para a execução da disciplina da Engenharia de Software denominada Verificação e Validação (V&V). Assim, em trabalhos futuros pretende-se explorar técnicas de V&V alinhadas com design automático, avaliando também se ferramentas de nosso grupo de pesquisa como a MockupToME [Basso et al. 2016], contribuem para a execução de atividades nessa disciplina;

2) A literatura carece de estudos de viabilidade do MDWE conduzidos em contextos de Fábricas de Software [Basso et al. 2017]. Ao longo de nossa investigação, percebeu-se que poucos trabalhos discutiram como a indústria recebeu estas tecnologias. Isso é um tanto decepcionante, já que tais pesquisas são motivadas em problemas da indústria de software. Assim, como seguimento dos estudos empíricos discutidos em [Basso et al. 2015], pretende-se estudar como diferentes atributos de qualidade, na transferência de tecnologia de MDWE, afetam a aceitação destes produtos no mercado, e;

3) Por fim, deve-se ampliar o relato deste mapeamento. Por motivos de formato do relato, e como forma de caracterização da área de pesquisa, pôde-se apenas conceituar as propostas existentes. Assim, ficou faltando elementos importantes de um mapeamento sistemático como o protocolo de pesquisa, questões de pesquisa e as suas respostas, bem como uma análise comparativa das características estruturais das 27 DSLs encontradas. Portanto, como próximo trabalho, pretende-se apresentar esta parte complementar de nosso estudo.

Referências

[Basso et al. 2017] Basso, F. P., Oliveira, T. C., Werner, C. M., and Becker, L. B. (2017). Building the foundations for 'mde as service'. *IET Software*, 11:195–206.

[Basso et al. 2016] Basso, F. P., Pillat, R. M., Oliveira, T. C., Roos-Frantz, F., and Frantz, R. Z. (2016). Automated design of multi-layered web information systems. *Journal of Systems and Software*, 117:612 – 637.

[Basso et al. 2015] Basso, F. P., Pillat, R. M., Roos-Frantz, F., and Frantz, R. Z. (2015). Combining mde and scrum on the rapid prototyping of web information systems. *International Journal of Web Engineering and Technology*, 10(3):214–244.

[Brambilla and Fraternali 2014] Brambilla, M. and Fraternali, P. (2014). Large-scale model-driven engineering of web user interaction: The webml and webratio experience. *Science of Computer Programming*, 89, Part B:71 – 87. Special issue on Success Stories in Model Driven Engineering.

[Ceri et al. 2000] Ceri, S., Fraternali, P., and Bongio, A. (2000). Web modeling language (webml): a modeling language for designing web sites. *Computer Networks*, 33(1-6):137–157.

[de Vrieze et al. 2011] de Vrieze, P., Xu, L., Bouguettaya, A., Yang, J., and Chen, J. (2011). Building enterprise mashups. *Future Generation Computer Systems*, 27(5):637 – 642.

[Elkoutbi et al. 2006] Elkoutbi, M., Khriss, I., and Keller, R. K. (2006). Automated prototyping of user interfaces based on uml scenarios. *Automated Software Engg.*, 13(1):5–40.

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

[Kent 2002] Kent, S. (2002). Model driven engineering. In *Integrated Formal Methods*, pages 286–298.

[Mesbah and Van Deursen 2008] Mesbah, A. and Van Deursen, A. (2008). A component- and push-based architectural style for ajax applications. *Journal of Systems and Software*, 81(12):2194–2209.

[Nunes and Schwabe 2006] Nunes, D. A. and Schwabe, D. (2006). Rapid prototyping of web applications combining domain specific languages and model driven design. In *6th international conference on Web engineering*, pages 153–160.

[Pillat et al. 2015] Pillat, R. M., Oliveira, T. C., Alencar, P. S., and Cowan, D. D. (2015). BPMNt: A BPMN extension for specifying software process tailoring. *Information and Software Technology*, 57(0):95 – 115.

[Planas et al. 2009] Planas, E., Cabot, J., and Gómez, C. (2009). Verifying action semantics specifications in uml behavioral models. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 5565 LNCS:125–140.

[Schmidt 2006] Schmidt, D. C. (2006). Guest editor's introduction: Model-driven engineering. *IEEE Computer*, 39(2):25–31.

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

Anais da 2ª Escola Regional de Engenharia de Software (ERES 2018)
em Dois Vizinhos, PR, Brasil, 22 a 24 de outubro de 2018

Um Mapeamento Sistemático Preliminar sobre Frameworks de Avaliação de Sistemas Legados

Jonnathan R. Lopes, Lukas F. Gaedicke, Andréa S. Bordin

[1]Universidade Federal do Pampa (UNIPAMPA)
Av. Tiarajú, 810, Ibirapuitã – Alegrete, RS – Brasil

{jonnathan.riquelmo, lukasfgaedicke}@gmail.com, andreabordin@unipampa.edu.br

Abstract. *Legacy systems need to be evaluated in order to know the best evolution strategy, it's not an easy task due to the inherent complexity that this type of system can have. In this way, it's important to know what aspects should be observed in this evaluation process. This paper presents a preliminary systematic mapping of literature which presents the dimensions and criteria for the evaluation of this type of system. The data collected in 17 studies show take into account technical and business aspects comprised of several criteria.*

Resumo. *Sistemas legados necessitam ser avaliados com o propósito de se saber qual a melhor estratégia de evolução, o que não é uma tarefa fácil devido à complexidade inerente que esse tipo de sistema pode possuir. Assim, é importante saber quais os aspectos que devem ser observados nesse tipo de avaliação. Esse estudo apresenta um mapeamento sistemático preliminar da literatura, no qual são apresentadas as dimensões e os critérios para a avaliação desse tipo de sistema. Os dados coletados em 17 estudos mostram que a avaliação deve levar em conta aspectos técnicos e de negócio compostos por diversos critérios.*

1. Introdução

O termo sistema legado possui definições na literatura que descrevem seus diversos aspectos. Para [Bennett 1995] são sistemas que foram desenvolvidos em algum momento do passado, mas que ainda mantêm um alto valor agregado e que muitas vezes são considerados críticos para as organizações que os possuem. Esses sistemas habitualmente foram desenvolvidas sem os métodos de engenharia de software adequados e frequentemente são mantidos para acomodar novas funcionalidades com o passar do tempo [Ransom et al. 1998]. Mais recentemente, [Johann 2016] corroborou com a ideia de que sistemas legados são importantes para o negócio porque geram receitas e que, em muitos casos, são o principal sistema de suporte para o negócio.

Por muitas vezes possuírem um alto valor de negócio para as organizações, os sistemas legados podem ser um grande problema. Isso acontece devido às constantes manutenções realizadas ao longo dos anos que acabam deteriorando o sistema como um todo, tornando a manutenibilidade um processo cada vez mais oneroso para as organizações.

Exposto isso, é notável que evoluir sistemas legados não é uma tarefa trivial. A tomada de decisão em relação ao que fazer com sistemas desse tipo exige uma análise ponderada e criteriosa de vários aspectos. Dessa forma, se faz necessário saber quais as

dimensões e critérios que devem ser analisados para uma correta avaliação do sistema e também quais as possíveis decisões a serem tomadas em relação a esses sistemas.

Na literatura existem algumas proposições de *frameworks* para o auxílio na tomada de decisão. A maioria propõe dimensões e conjuntos de critérios que devem ser observados quando um procedimento de avaliação de sistemas legados é realizado.

Assim, o objetivo deste artigo é realizar um mapeamento sistemático (MS) preliminar da literatura buscando investigar as propostas para avaliação de sistemas legados e as respectivas características que especificam as dimensões e critérios que devem ser levados em conta nesse tipo de processo. O trabalho relacionado que mais se aproxima desta pesquisa é o de [Agilar et al. 2016], um MS sobre processos, técnicas e ferramentas para apoiar a modernização de sistemas legados. Contudo, esse trabalho não aborda especificamente questões relacionadas às dimensões e critérios de avaliação de sistemas, as quais antecedem o processo de modernização em sí.

A estrutura do artigo segue conforme especificado: na Seção 2 é descrita a metologia do protocolo executado, a Seção 3 apresenta os resultados obtidos através do mapeamento e, finalmente, a Seção 4 expõe as considerações finais do estudo elaborado.

2. Metodologia

2.1. Protocolo do Mapeamento Sistemático

Para a obtenção de resultados confiáveis e reproduzíveis é vital que um processo bem definido e estruturado seja seguido. O protocolo de MS utilizado para este estudo foi o de Petersen *et al.* 2008 [Petersen et al. 2008], cujas etapas são exibidas na Figura 1.

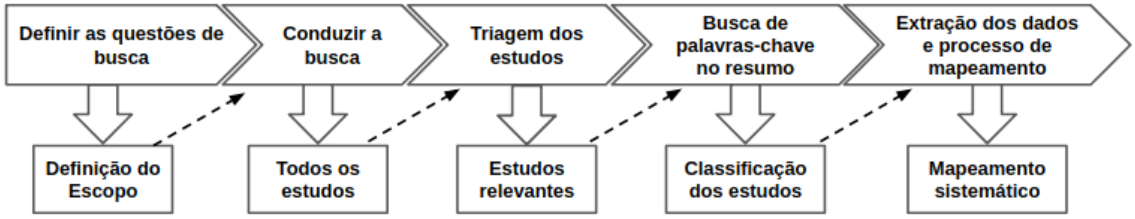


Figura 1. Processo de Mapeamento Sistemático [Petersen et al. 2008]

Em linha gerais, o protocolo visa definir todas as etapas e atividades necessárias para conduzir o MS, como as questões de pesquisa (QP), critérios de seleção, bases a serem utilizadas, a estratégia de leitura dos estudos recuperados e a síntese dos resultados.

2.2. Bases de Busca

Para a realização desse MS utilizou-se bases de dados que possuem: **(I)** mecanismo de pesquisa baseado na web; **(II)** mecanismo de busca capaz de usar palavras-chave; e **(III)** documentos da grande área da Ciência da Computação. As bases selecionadas para a realização do trabalho foram: *ACM Digital library*, *Engineering Village (Compendex)*, *IEEE Xplore* e *Scopus*. Todas as bases são amplamente reconhecidas no meio acadêmico.

2.3. String de Busca

Foi definido um conjunto de palavras referentes ao tema de pesquisa, bem como sinônimos considerados expressivos. Para a construção da *string* foram investigados os termos

que representam o tema investigado, sendo eles, *e.g.* Legacy System, Legacy Application, Framework, Assessment, Management, entre outros.

```
(legacy system OR legacy software OR legacy information system OR
legacy software system OR software modernization OR Legacy Code OR
Legacy Application) AND (framework OR decision making OR
decision-making OR assessment OR evaluation OR management OR model)
```

Figura 2. String Genérica de Busca

Após o levantamento de termos centrais e seus sinônimos houve a construção de uma *string* padrão, exibida na Figura 2. Contudo, é importante ressaltar que, para cada uma das bases de dados utilizadas, a *string* padrão sofreu modificações. Isso se fez necessário para ter a adequação quanto à parâmetros exigidos pelos mecanismos de busca. Para a validação da *string* foram realizadas buscas-piloto. Com um conjunto de artigos predefinidos que deveriam retornar, foi efetuado a pesquisa e então verificado quantos destes trabalhos eram recuperados. Para tanto, mediante os resultados finais desse processo, as *strings* foram consideradas satisfatórias para o prosseguimento da MS.

2.4. Questões de Pesquisa

Após a definição do contexto e objetivo foram elaboradas três questões de pesquisa (QP) que guiaram o restante do MS: **QP1 -** *Quando e onde os estudos têm sido publicados?* ; **QP2 -** *Quais dimensões são abordadas no processo de avaliação de sistemas legados?* ; **QP3 -** *Em cada dimensão quantos critérios são utilizados no processo de avaliação de sistemas legados?*.

2.5. Critérios de Seleção de Estudos

Os critérios de inclusão indicam por qual ou quais critérios um estudo é incluído no MS, ou seja, considerado relevante. Da mesma forma, os de exclusão indicam por qual ou quais critérios um estudo é excluído, ou seja, considerado não relevante [Nakagawa et al. 2017]. Nesse MS os critérios da Tabela 1 foram aplicados.

Tipo	Critério
Inclusão	CI1. *O estudo deve abordar um framework ou processo para avaliação de sistemas legados.*
Exclusão	CE1. *O estudo está duplicado.*
Exclusão	CE2. *O estudo não está escrito em inglês.*
Exclusão	CE3. *O estudo não fornece acesso completo ao seu conteúdo.*
Exclusão	CE4. *O estudo não atende ao CI1.*

Tabela 1: Critérios de Seleção

2.6. Condução do Mapeamento Sistemático

A condução prática do MS iniciou-se com as buscas sendo realizadas nas bases de dados selecionadas. Na Tabela 2 são apresentados os resultados em relação a cada base, bem como o total de estudos primários recuperados. Após, deu-se início a etapa de seleção.

A avaliação dos trabalhos foi realizada em *peer review* (revisão por pares), por meio de quatro iterações, no qual diferentes atividades foram realizadas para alcançar o

Base de Dados Pesquisada	Estudos	% do Total
ACM Digital library	101	9%
Engineering Village (Compendex)	335	31%
IEEE Xplore	136	12%
Scopus	562	48%
TOTAL	**1134**	**100%**

Tabela 2: Estudos Recuperados por Bases de Dados

subconjunto final de trabalhos que foram analisados neste MS. A Figura 3 apresenta os ciclos e os artigos restantes em cada iteração.

É importante salientar que houve a inclusão de quatro trabalhos sob a orientação de uma especialista com experiência na área de sistemas legados. Os estudos incluídos somaram-se aos 13 trabalhos resultantes do último ciclo de seleção, o que implicou em um total de 17 trabalhos base para este estudo.

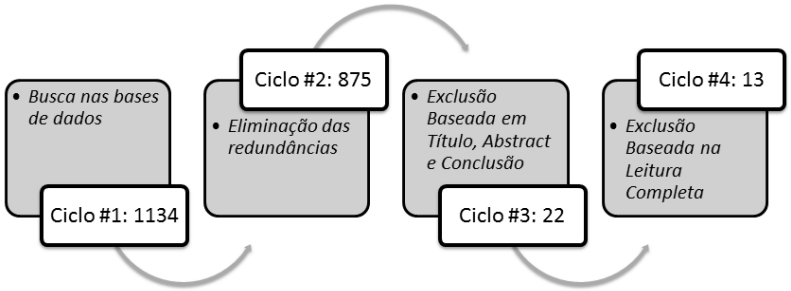


Figura 3. Ciclos de Avaliações dos Trabalhos

2.7. Estratégia de Extração dos Dados

Com o objetivo de registrar as informações necessárias para responder as QPs estabelecidas, foram definidos os dados pertinentes relacionados, sendo eles: ***Ano***; ***Autor(es)***; ***Filiação***; ***Dimensões***, ou espectro de análise de *sistemas legados*; ***Critérios***, ou subconjunto de variáveis observadas nas dimensões de análise.

Esse conjunto de informações foi extraído novamente com uma revisão em par e então tabulado. O resultado auxiliou na formulação da síntese dos dados. Após, foi realizada a discussão dos resultados frente às QPs do estudo.

3. Resultados e Discussão

Após o fim do processo de seleção, extração e tabulação dos dados, as QPs foram discutidas. Em relação a **QP1** *("Quando e onde os estudos têm sido publicados?")*, foi identificado que os estudos selecionados foram publicados a partir do ano de 1995 até o ano de 2013, estando distribuídos de maneira uniforme no período. No entanto, observa-se que nos últimos 4.5 anos não foram publicados estudos que atendessem os critérios de inclusão do protocolo estabelecido. Destaca-se ainda que o estudo de [Sneed 1995] é o primeiro e mais relevante, sendo citado 296 vezes.

Ainda em relação a **QP1**, 5 dos 17 estudos foram publicados na *Euromicro Conference on Software Maintenance and Reengineering [CSMR]*, como pode ser observado

Referência	Evento [Sigla]	h-index
[Sneed 1995]	IEEE Software	96
[Bennett et al. 1999]	IEE Proceedings - Software (Atual IET Software)	96
[Brooke and Ramage 2001]	International Journal of Information Management [IJIM]	82
[De Lucia et al. 2001]	IEEE International Conference on Software Maintenance [ICSM]	50
[Battaglia et al. 1998] [Ransom et al. 1998] [Koskinen et al. 2005] [Aversano et al. 2005] [Bergmayr et al. 2013]	Euromicro Conference on Software Maintenance and Reengineering [CSMR]	41
[Aversano and Tortorella 2004]	Journal of Software Maintenance and Evolution [JSME]	39
[Rajavat and Tokekar 2011]	Communications in Computer and Information Science [CCIS]	35
[O'Byrne and Wu 2000]	International Workshop on Principles of Software Evolution [IWPSE]	28
[Alkazemi 2014]	Journal of Software [JSW]	24
[Jain and Chana 2015]	International Conference on Communication Technology Proceedings [ICCT]	21
[Kankaanpää et al. 2007]	International Conference on Enterprise Information Systems [ICEIS]	11
[Fronckowiakand and Zandoli 2013]	Proceedings of Student/Faculty Research Day, [CSIS] (Pace University - United States)	0
[Alkazemi et al. 2013]	IEEE International Conference on Systems, Man, and Cybernetics [SMC]	N/A

Tabela 3: Estudos Analisados

na Tabela 3. Para atestar a qualidade dos estudos encontrados, foi pesquisado o *h-index*[1] para todos os trabalhos selecionados. Este fator foi proposto em 2005 por Jorge E. Hirsch e serve como indicativo de qualidade e notoriedade de autores, revistas ciêntificas e eventos. Desta forma, os estudos foram agrupados por evento e publicação, e então ordenados de maneira crescente utilizando por base o ano de publicação.

Para a **QP2** *("Quais dimensões são abordadas no processo de avaliação de sistemas legados?")*, após a análise dos estudos foram constatadas evidências que colaboraram para o arranjo dos dados, conforme a Tabela 4. A grande maioria dos estudos apontam que os sistemas legados são avaliados segundo duas ou, em alguns casos, até três dimensões, sendo a de Negócio e a Técnica as dimensões mais expressivas.

Existem ainda algumas propostas que abordam a dimensão técnica subdividindo-a em dois tópicos: um que aborda a infraestrutura de software e hardware necessária para o sistema legado, e outra que se refere ao próprio sistema em avaliação. Essa categorização foi adotada na síntese dos dados e pode ser observado na Tabela 4, onde os critérios técnicos foram divididos em dois tipos denotados por *CR-Tec/Sof* e *CR-Tec/Apoio*.

Apesar de alguns estudos apontarem uma quantidade maior de dimensões, foi inferido que as mesmas podem ser classificadas nas duas dimensões já mencionadas. Ainda assim, cabe destacar o estudo detalhado de [Koskinen et al. 2005] que propôs 13 dimensões, as quais foram categorizadas neste MS nessas duas dimensões citadas, *e.g. 11.Processos de Negócio* é equivalemente a dimensão de negócio.

Finalmente para a **QP3** *("Em cada dimensão quantos critérios são utilizados no processo de avaliação de sistemas legados?")*, observa-se na Tabela 4 que a maioria dos estudos sugeriram critérios de avaliação, com exceção de [Battaglia et al. 1998]. Por outro lado, o estudo de [Koskinen et al. 2005] novamente destaca-se como o que mais cita critérios de avaliação.

É importante ressaltar que o maior número de critérios situa-se na dimensão técnica relacionada ao próprio sistema em avaliação, perfazendo 52% de todos os critérios.

[1] `https://www.scimagojr.com/`

REFERÊNCIA	DIMENSÃO	CR Negócio	CR-Tec/Soft	CR-Tec/Apoio	TOTAL
[Sneed 1995]	1.Qualidade Técnica; 2.Valor de negócio	3	25	0	28
[Battaglia et al. 1998]	1.Qualidade Técnica; 2.Valor de negócio	0	0	0	0
[Ransom et al. 1998]	1.Avaliação de valor de negócios; 2.Avaliação de Ambiente Externo; 3.Avaliação de Aplicação	4	9	20	33
[Bennett et al. 1999]	Organização; Técnico	0	6	1	7
[O'Byrne and Wu 2000]	1.Adequação do sistema; 2.Adequação da Plataforma Subjacente; 3.Qualidade do sistema	3	3	5	11
[Brooke and Ramage 2001]	Valor de negócio; Valor técnico	3	6	0	9
[De Lucia et al. 2001]	1.Valor de negócio; 2.Valor técnico	9	12	3	24
[Aversano and Tortorella 2004]	1.Tecnologias 2.Organização 3.Processo 4.Sistemas Legados	7	8	12	27
[Koskinen et al. 2005]	1.Caracterisiticas gerais da aplicação; 2.Requisitos do usuário; 3.Fatores de manutenção; 4.Mantenedores; 5.Processo de manutenção; 6.Qualidades técnicas do sistema; 7.Ferramentas básicas de desenvolvimento; 8.Documentação do sistema; 9.Projeto do sistema; 10.Manutenibilidade do sistema; 11.Processo de negócio; 12.Mudanças de tecnologia; 13.Qualidade do sistema;	2	35	12	49
[Aversano et al. 2005]	1.Valor de negócio; 2.Valor técnico; 3.Sistema legado	3	7	0	10
[Kankaanpää et al. 2007]	1.Valor técnico; 2.Valor de negócio	10	6	0	16
[Rajavat and Tokekar 2011]	1.Domínio do sistema; 2.Domínio gerencial; 3.Domínio técnico	2	2	2	6
[Bergmayr et al. 2013]	1.Negócio; 2.Técnico	1	1	0	2
[Fronckowiakand and Zandoli 2013]	1.Negócio; 2.Técnico; 3.Processo de decisão	9	7	4	20
[Alkazemi et al. 2013]	1.Suporte; 2.Negócio; 3.Arquitetura; 4.Tecnologia	3	6	8	17
[Alkazemi 2014]	1.Suporte; 2.Negócio; 3.Arquitetura; 4.Tecnologia	3	6	8	17
[Jain and Chana 2015]	1.Avaliação da aplicação; 2.Qualidade de Serviço; 3. Avaliação da nuvem de destino	0	6	3	9
TOTAL		**62**	**149**	**74**	**285**

Legenda - ***CR***: Critérios, ***Tec/Soft***: Técnico/Software, ***Tec/Apoio***: Técnico/Apoio.

Tabela 4: Dados Extraídos

Por outro lado, a soma com os 26% que representam os critérios técnicos de apoio ao software totalizam 78%, fornecendo um número expressivo de proposições na dimensão técnica. Os critérios relacionados à dimensão de negócio representam apenas 22%.

4. Considerações Finais

Por meio da análise dos dados do MS realizado foi possível evidenciar que existem diversas proposições de *frameworks* de avaliação de sistemas legados na literatura. Os estudos, em geral, descrevem diversas dimensões e critérios que precisam ser considerados e analisados cautelosamente antes da tomada de decisão por conta do alto valor que os sistemas legados podem representar para as organizações que mantêm seu funcionamento.

A partir dos resultados obtidos é possível observar uma maior frequência de critérios relacionados à dimensão técnica, visto que os dados expostos na seção 3, demonstram um total de 293 critérios técnicos contra apenas 62 critérios de negócio. Um dos motivos dessa tendência pode ser atribuída ao fato dos pesquisadores serem especialistas da área da Computação, e por este fato, acabarem não caracterizando alguns aspectos mais específicos e de difícil identificação na dimensão de negócio. Os resultados obtidos através desse estudo poderão ser utilizados por mantenedores e pesquisadores da área como base para a geração de um arcabouço de conhecimento das dimensões e critérios indispensáveis na análise de sistemas legados.

Como trabalhos futuros, será realizado o *snowballing* dos trabalhos já incluídos nesse estudo e, posteriomente o mapeamento será complementado com a identificação dos critérios mais utilizados nas dimensões, assim como a identificação das alternativas de evolução decorrentes dos processos de avaliação, descritas nos *frameworks* dos estudos analisados. Essa atividade complementará a qualidade do presente trabalho, uma vez que será possível realizar associações entre as dimensões e critérios com as decisões de evolução mais adequadas a serem tomadas, *e.g.* reengenharia, migração ou substituição, em relação a sistemas legados nas organizações.

Referências

Agilar, E., Almeida, R., and Canedo, E. (2016). A systematic mapping study on legacy system modernization. In *SEKE*, pages 345–350.

Alkazemi, B. Y. (2014). A framework to assess legacy software systems. *Journal of Software*, 9(1):111–115.

Alkazemi, B. Y., Nour, M. K., and Meelud, A. Q. (2013). Towards a Framework to Assess Legacy Systems. *2013 IEEE International Conference on Systems, Man, and Cybernetics*, pages 924–928.

Aversano, L., Esposito, R., Mallardo, T., and Tortorella, M. (2005). Evolving legacy system toward elegacy system in ebusiness context. In *CSMR*, CSMR '07, pages 201–, Washington, DC, USA. IEEE Computer Society.

Aversano, L. and Tortorella, M. (2004). An assessment strategy for identifying legacy system evolution requirements in eBusiness context. *Journal Of Software Maintenance And Evolution*, pages 255–276.

Battaglia, M., Savoia, G., and Favaro, J. (1998). Renaissance: A method to migrate from legacy to immortal software systems. *CSMR*, 1998-March(D):197–200.

Bennett, K. (1995). Legacy systems: coping with success. *IEEE Software*, 12(1):19–23.

Bennett, K. H., Ramage, M., and Munro, M. (1999). Decision model for legacy systems. *IEE Proceedings - Software*, 146(3):153–159.

Bergmayr, A., Brunelière, H., Izquierdo, J. L. C., Gorroñogoitia, J., Kousiouris, G., Kyriazis, D., Langer, P., Menychtas, A., Orue-Echevarria, L., Pezuela, C., and Wimmer, M. (2013). Migrating legacy software to the cloud with ARTIST. *CSMR*, pages 465–468.

Brooke, C. and Ramage, M. (2001). Organisational scenarios and legacy systems. *International Journal of Information Management*, 21(5):365–384.

De Lucia, A., Fasolino, A. R., and Pompella, E. (2001). A decisional framework for legacy system management. *IEEE International Conference on Software Maintenance, ICSM*, pages 642–653.

Fronckowiakand, D. and Zandoli, R. (2013). A new algorithmic approach to the it modernization problem. *Proceedings of Student/Faculty Research Day, CSIS, Pace University*, pages 280–287.

Jain, S. and Chana, I. (2015). Modernization of Legacy Systems: A Generalised Roadmap. *Proceedings of the Sixth International Conference on Computer and Communication Technology 2015*, pages 62–67.

Johann, S. (2016). Dave thomas on innovating legacy systems. *IEEE Software*, (2):105–108.

Kankaanpää, I., Tiihonen, P., Ahonen, J., Koskinen, J., Tilus, T., and & Sivula, H. (2007). Legacy system evolution - a comparative study of modernisation and replacement initiation factors. *ICEIS 2007*, pages 280–287.

Koskinen, J., Ahonen, J. J., Sivula, H., Tilus, T., Lintinen, H., and Kankaanpaa, I. (2005). Software modernization decision criteria: an empirical study. In *CSMR*, pages 324–331.

Nakagawa, E., Scannavino, K., Fabbri, S., and Ferrari, F. (2017). *Revisão Sistemática da Literatura em Engenharia de Software: Teoria e Prática*. Elsevier Editora Ltda.

O'Byrne, P. and Wu, B. (2000). Lace frameworks and technique-identifying the legacy status of a business information system from the perspectives of its causes and effects. In *Proceedings International Symposium on Principles of Software Evolution*, pages 170–174.

Petersen, K., Feldt, R., Mujtaba, S., and Mattsson, M. (2008). Systematic mapping studies in software engineering. In *EASE*, volume 8, pages 68–77.

Rajavat, A. and Tokekar, V. (2011). ReeRisk - A decisional risk engineering framework for legacy system rejuvenation through reengineering. *CCIS*, pages 152–158.

Ransom, J., Somerville, I., and Warren, I. (1998). A method for assessing legacy systems for evolution. *CSMR*, pages 128–134.

Sneed, H. M. (1995). Planning the Reengineering of Legacy Systems. *IEEE Software*, 12(1):24–34.

Anais da 2ª Escola Regional de Engenharia de Software (ERES 2018)
em Dois Vizinhos, PR, Brasil, 22 a 24 de outubro de 2018

Avaliação de Algoritmos de Análise de Sentimentos em Tweets no Domínio da Copa do Mundo FIFA 2018

Patricia Feliciano[1], **Paulo Roberto Farah**[1]

[1]Departamento de Engenharia de Software
Universidade do Estado de Santa Catarina (UDESC)
Rua Dr. Getúlio Vargas, 2822 – 89.140-000 – Ibirama – SC – Brasil

patthyfeliciano@gmail.com, paulo.farah@udesc.br

Abstract. *Social networks have generated a large amount of data on the Web, so various methods and techniques have been proposed in the form of applications for monitoring and analysis of repercussion of brands, products relevant events such as the FIFA World Cup. This paper analyzes appropriate classification methods for the FIFA World Cup 2018 event domain on Twitter. It was collected more than 1 million of tweets, 4000 were labeled manually by a human and 400 thousand were classified. The results show that Opinion Finder algorithm had an almost perfect fit with kappa coeficient of 0,95 for the domain of the event.*

Resumo. *Redes sociais têm gerado um grande volume de dados na Web e vários métodos e técnicas vêm sendo propostos em forma de aplicações para monitoramento e análise de repercussão de marcas, produtos e eventos relevantes como a Copa do Mundo FIFA. Este artigo analisa métodos de classificação adequados para o domínio da Copa do Mundo FIFA 2018 no Twitter. Foram coletados mais de 1 milhão de tweets, 4000 foram rotulados manualmente por um humano e 400 foram classificados. Os resultados mostram que o algoritmo Opinion Finder obteve predição quase perfeita baseado no coeficiente kappa de 0,95 para o domínio do evento estudado.*

1. Introdução

A popularização e o aumento do uso das redes sociais na Internet transformaram o modo de interação entre pessoas e organizações. Por meio delas, diversos indivíduos possuem a liberdade de expressar suas opiniões sobre marcas, eventos, produtos, acontecimentos dentre vários assuntos que desejam publicar.

A expressividade dos usuários pode ser compartilhada com aspectos positivos, neutros ou até mesmo negativos como críticas, por exemplo. Logo, a rapidez de propagação e abrangência pode repercutir em impactos diretamente ligados a imagem de marcas e organizações empresariais, as quais geralmente não conseguem acompanhar a demanda de análise que todo conteúdo gerado requer ao mesmo tempo em que são postados.

Como nem sempre as repercussões das opiniões são positivas, estas por sua vez, podem comprometer e influenciar opiniões de demais indivíduos de forma negativa. Assim sendo, a análise destas informações pode servir vantajosamente como forma de extração de opiniões do público alvo acerca de seu produto ou serviço ofertado [Gomes 2013].

Nesse contexto, a mineração de textos disponibiliza um conjunto de técnicas capazes de automatizar o processo de extração e análise de sentimentos, de modo que as

organizações aproveitem os resultados obtidos para elaborar suas táticas de marketing em produtos e eventos, além de aperfeiçoar estratégias dentre outras possíveis aplicações [Tan 1999].

A Copa do Mundo FIFA é um evento esportivo muito popular. O campeonato foi disputado entre os dias 14 de junho e 15 de julho de 2018, por 32 seleções de diversos países [FIFA]. Atualmente é comum ver o crescimento exponencial da repercussão em torno de eventos de grande porte como esse.

A partir deste cenário, [Rezende 2005] afirma que: "Devido à incapacidade do ser humano de interpretar tamanha quantidade de dados, muita informação e conhecimento, possivelmente úteis, podem estar sendo desperdiçados, ficando ocultos dentro das Bases de Dados espalhadas pelo mundo". Assim sendo, a pergunta chave para definição do problema é: como obter e classificar sentimentos e opiniões expressados pelos usuários do Twitter a respeito da Copa do Mundo de 2018 utilizando o método mais adequado para este domínio?

O objetivo geral deste estudo é avaliar algoritmos na predição de sentimentos em tweets sobre a Copa do Mundo de 2018, de modo a descobrir o(s) método(s) de classificação com maior índice de concordância em predição sobre este evento. Para isto, este trabalho tem como objetivos específicos: (1) coletar dados textuais relacionados ao evento Copa do Mundo FIFA 2018; (2) rotular humanamente uma amostragem de tweets coletados; (3) categorizar sentimentos e opiniões extraídas da base de dados coletada; (4) validar o nível de acurácia dos métodos classificadores com a amostragem rotulada pelo ser humano.

Vistos os objetivos específicos, justifica-se o tema escolhido pela repercussão que eventos de grande abrangência vem a apresentar, como por exemplo a geração de opiniões e sentimentos, muitas vezes polêmicos em torno de assuntos relacionados. No esporte não é diferente, a popularidade do futebol no Brasil, país onde foi sediado o último Campeonato Mundial de Futebol FIFA 2014, aliado ao crescimento exponencial de adeptos e simpatizantes, foram fatores influentes na escolha da abordagem deste evento atual e popular.

O presente trabalho foi estruturado em quatro seções, sendo esta a primeira. A seção 2 contém alguns trabalhos relacionados a esse estudo. A seção 3 descreve a arquitetura e as etapas inerentes ao trabalho. A seção 4 apresenta os resultados obtidos. Por fim, a seção 5 possui a conclusão e os trabalhos futuros.

2. Trabalhos Relacionados

Esta seção apresenta trabalhos relacionados à mineração textual e análise de sentimentos em mídias sociais.

2.1. iFeel

Através de uma pesquisa em torno de técnicas e ferramentas existentes, um estudo relacionado apresenta um *benchmarking* entre 18 métodos de classificação de sentimentos [Araújo et al. 2016]. O projeto iFeel identifica e classifica sentimentos e opiniões em textos estruturados e não estruturados. Os autores explicam que um método que apresentou desempenho eficaz em um contexto de 'política', não era tão eficaz quando aplicado a um

contexto de 'esportes', exemplo meramente ilustrativo. Isso se dá em virtude de contextos de treinamento realizados, dicionários léxicos distintos, presença de expressões populares em torno de um assunto específico, ou seja, vários fatores podem refletir no desempenho de um método classificador.

Os autores comparam os seguintes algoritmos: Opinion Lexicon, Sentistrength, Socal, Happiness Index, Sann, EmoticonsDS, Sentiment, Stanford, Afinn, Mpqa, Nr-chashtag, Emolex, Emoticons, Sasa, Panast, Vader e Umigon. De acordo com as métricas aplicadas pelos autores, os cinco métodos que mais se destacaram foram: Sentistrength, Afinn, Opinion Lexicon, Umigon e Vader [Benevenuto et al. 2015].

2.2. Copa do Mundo FIFA 2014

Este trabalho analisa e compara o comportamento das torcidas fora dos campos. Além disso, toma como consideração final a concepção do impacto resultante da análise textual para analisar e auxiliar no entendimento da dinâmica desse comportamento [Kim et al. 2015].

Nessa pesquisa 790.744 usuários do Twitter tiveram seu comportamento acompanhado e suas mensagens registradas e analisadas tanto antes quanto durante o evento. Nesse sentido, foram analisados padrões temporais de volumes de mensagens, tópicos repercutidos, compartilhamento de postagens (retweets), dentre outros.

Os autores pesquisaram três questões principais, sendo que a primeira analisou se o comportamento dos participantes apresentou variação nos momentos de transmissão comparado à momentos normais em que não estava sendo transmitido o evento. A segunda questão tratou de como a diversidade de tópicos muda durante o evento. Por fim, o terceiro tópico analisado relacionou como os tweets variaram de acordo com a proximidade dos países com o evento FIFA 2014, bem como o papel dos usuários multilíngues sobre as diferenças geográficas e culturais dentro das comunidades online.

Neste contexto, relata-se a abordagem de busca utilizada, onde traduziu-se o termo "worldcup2014" para vários idiomas para filtrar os tweets coletados, bem como a utilização de abreviaturas de nomes de seleções que estariam disputando os jogos, como por exemplo: (#BRAvsGER).

3. SAMS

Esta seção apresenta o projeto Sistema de Análise e Monitoramento de Sentimentos (SAMS). Inicialmente é descrita a arquitetura dos processos incorporados ao presente trabalho, seguido das etapas de desenvolvimento.

3.1. Arquitetura Geral

A arquitetura geral do SAMS é apresentada na Figura 1 a qual contém os módulos e suas respectivas ferramentas adotados na pesquisa.

Inicialmente, a arquitetura possui um módulo coletor (1) que tem como objetivo coletar tweets em tempo real com base nas palavras-chaves inicialmente especificadas. É responsável pela persistência na base de dados dos tweets coletados (2) para, em seguida, serem exportados com auxílio da ferramenta de exportação (3). Logo acontece a retroalimentação, onde identifica-se a frequência das palavras que mais aparecem na coleta e

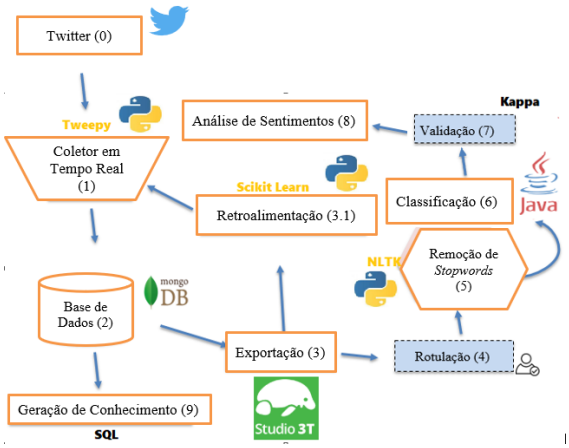


Figura 1. Arquitetura geral do SAMS.

assim, estas serem analisadas e inseridas no filtro para robustez do coletor. Deste modo, a ferramenta Robo3T, a qual basicamente permitiu a elaboração de scripts para geração de conhecimento, bem como a exportação dos dados em formato CSV contribui para dois fluxos desta arquitetura.

A partir das planilhas CSV exportadas foi possível efetuar os tratamentos necessários; como a remoção de *stopwords* (5) para em seguida submeter estes dados aos métodos de classificação (6). Feito isto, acontece a validação dos resultados rotulados versus os resultados provenientes da classificação automatizada (7). Nesta fase de validação foram geradas as matrizes de confusão e o coeficiente Kappa, ambos úteis na identificação dos métodos com a predição mais assertiva para a Copa do Mundo FIFA 2018. Na sequência, acontece a análise de sentimentos (8). Ao final da análise de sentimentos foi realizado a geração de conhecimento (9), esta fase apesar de ser exposta como a última não possui relação direta da fase(8), pois trata-se de uma exploração da base de dados coletada a qual envolve análises via SQL representadas graficamente.

Neste contexto, [Landis and Koch 1977] classificam a força da concordância através de faixas analisadas sobre o índice calculado. A Tabela 1 apresenta a classificação do índice Kappa. Pode-se avaliar o nível de concordância que um coeficiente Kappa Cohens pode assumir, partindo da insignificância representada pelo valor zero passando por várias faixas até o valor 1, o qual caracteriza uma excelente concordância em um cenário quase perfeito.

Tabela 1. Classificação do Índice Kappa

Valor	Interpretação		Valor	Interpretação
< 0	Ausência de concordância		0,41 – 0,60	Moderada
0,00 – 0,20	Mínima		0,61 – 0,80	Substancial
0,21 – 0,40	Razoável		0,81 – 1,00	Quase perfeita

3.2. Etapas

Nesta seção serão apresentadas as etapas de desenvolvimento de estudo, deste a coleta até os resultados obtidos. Para a elaboração do presente trabalho foram realizadas as seguintes etapas: coleta, rotulação, categorização, validação e análise de resultados.

3.2.1. Coleta

O Twitter foi escolhido por ser uma plataforma com um grande número de usuários ativos. Além disto, a abrangência mundial, tanto desta rede social quanto do evento FIFA 2018 são fatores que podem instigar usuários a compartilhar ideias constantemente. O Twitter disponibiliza uma API que possibilita de maneira prática a exploração de informações textuais e atributos públicos contidos nos tweets postados.

Diante deste cenário, a metodologia utilizada foi a coleta de dados, a validação de classificadores, comparados ao treinamento humano, a qual preocupa-se em analisar a acurácia dos métodos estudados e sua eficácia na resolução dos problemas propostos diante de características do evento estudado. A extração de dados textuais iniciou-se em abril de 2018, sendo executada até início de junho em dias variados com intervalos no período de extração. Foram coletados atributos públicos como: tweet, coordenadas, localização, país, idioma, data de criação da conta do usuário, data da postagem, quantidade de amigos e quantidade de seguidores, cor de fundo da conta do usuário, tipo de dispositivo que originou a postagem coletada e persistidos em uma coleção no MongoDB.

3.2.2. Rotulação

Nesta etapa de desenvolvimento foram rotulados manualmente 2 mil tweets aleatórios em português e 2 mil também aleatórios em inglês. No momento de rotulação, foram consideradas as intenções que o usuário teve ao expressar-se no Twitter. De modo que a opinião particular do rotulador em polêmicas presentes no texto não foi considerada.

Os tweets com apenas links, datas, marcações de outros usuários, ambiguidade de interpretação, ironias, propagandas, ou que não expressavam nenhum sentimento plausível bem como opiniões sobre assuntos relacionados ao evento foram considerados neutros (0). Em contrapartida, tweets que claramente expressavam sentimentos bons, opiniões favoráveis, foram categorizados com positivo (1). De forma que os demais tweets foram vistos como negativos (-1), sendo eles portadores de características ruins, triste, adversos ou mesmo vocabulários obscenos, tanto no idioma inglês quanto no português.

3.2.3. Validação de Classificadores

Após a rotulação humana em uma significativa amostragem da base de dados, tornou-se viável a validação dos métodos, no propósito de identificar quais os melhores métodos dentre os 18 disponibilizados em [Benevenuto et al. 2015, Araújo et al. 2016]. Os resultados obtidos com esta avaliação, não são capazes de declarar uma verdade única ou prover um *benchmarking* completo, mas sim, indicar através de métricas estatísticas quais os melhores métodos para o evento específico idealizado no presente trabalho. Visto que, conforme [Benevenuto et al. 2015], ainda não se encontrou um método classificador com desempenho acima de todos os demais existentes. Por isto a importância de uma validação como esta antes de aplicar qualquer método em qualquer contexto e analisar sentimentos às cegas.

Antes de submeter os arquivos para o iFeel realizou-se um pré-processamento do

texto, o qual consistiu na remoção de caracteres especiais, exceto o "#" que caracteriza uma *hashtag* na maioria de suas aparições. Nesse sentido, também foram removidas palavras sem relevância, conhecidas como *stopwords*, sendo para o descarte destas palavras utilizou-se a plataforma NLTK (Natural Language Toolkitk), empregada no Python para trabalhar com a linguagem humana. Tudo isto para melhorar o desempenho dos algoritmos de classificação textual a serem utilizados na sequência.

Foram submetidos os mesmos 4 mil tweets rotulados na subseção anterior aos cinco melhores métodos do iFeel (Sentistrength, AFINN, OpinionLexion, Umigon e Vader) conforme elencados pelo *benchmarking* de [Araújo et al. 2016]. Visto isto, para geração das matrizes de confusão de todos os métodos analisados utilizou-se as tecnologias já existentes provenientes da biblioteca de aprendizado de máquina Scikit Learn que consequentemente deram acesso à funções nativas desta API: confusion_matrix e cohen_kappa_score.

4. Resultados

Esta seção apresenta os resultados obtidos com o presente trabalho. A começar pelos principais resultados provenientes das matrizes de confusão, as quais viabilizaram o cálculo do coeficiente Kappa (κ) e o índice de concordância desenvolvido para validação dos métodos de classificação que foram utilizados.

Inicia-se a apresentação dos resultados pela abordagem das seis principais matrizes de confusão geradas para a base de dados em inglês sobre os métodos de classificação utilizados neste estudo que mais se destacaram perante aos demais, vide Tabela 2.

Tabela 2. Matrizes de Confusão

	OpinionFinder				Umigon				SentiStrength		
	+	N	-		+	N	-		+	N	-
+	467	16	0		468	15	0		471	12	0
N	7	1478	3		58	1413	17		85	1337	66
-	0	8	21		0	22	7		0	8	21
	OpinionLexicon				Vader				Affin		
	+	N	-		+	N	-		+	N	-
+	471	12	0		468	15	0		471	10	2
N	268	1198	22		230	1241	17		522	911	5
-	0	16	13		1	24	4		3	18	8

Observa-se, a partir da Tabela 2, a quantidade de verdadeiros e falsos acertos que cada um dos seis métodos exibidos acima apresentou perante a rotulação humana sobre a amostragem em inglês. Deste modo, caracteriza-se a diagonal principal como os verdadeiros acertos, ou seja, o conjunto de dados foi rotulado positivo tanto pelo método o quanto pelo rotulador, o mesmo aplica-se às duas outras categorias negativo e neutro.

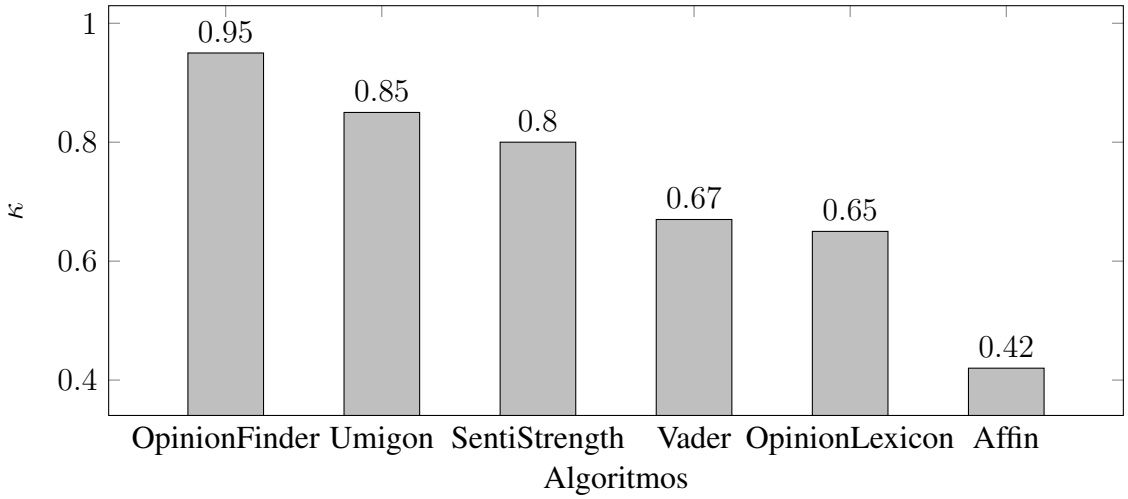
Além disso, nota-se que a quantidade de acertos verídica, diagonal principal é maior para o método *Opinion Finder* do que para os demais, principalmente o *Affin* que resultou em muitas divergências nas demais posições da matriz.

Na sequência, ao analisar o índice de concordância, e observou-se que, para a base

de dados em português, nenhum dos 18 métodos do iFeel apresentou classificação do índice Kappa (κ) igual ou superior a *0,40* o que indica que nenhum método de classificação resultou em predições no mínimo moderadas em relação a este contexto e idioma.

Portanto, optou-se por aplicar as demais etapas de desenvolvimento somente aos tweets coletados em inglês, visto que para este idioma submetido aos mesmos métodos obteve-se resultados muito mais satisfatórios, cujo Kappa dos seis melhores métodos variou entre *0,42* e *0,95*, onde 1 corresponde a predição excelente. Em seguida, a Figura 2 representa graficamente os principais resultados da validação através do índice de concordância Kappa.

Figura 2. Resultado da comparação do índice κ entre os algoritmos avaliados.



Pode-se observar que, dentre os seis métodos com maior índice de concordância estão presentes os mesmos cinco métodos que mais se destacaram no *benchmarking* de [Araújo et al. 2016]. Entretanto, o segundo ponto observado é que *Opinion Finder* foi o método que melhor se destacou com um coeficiente de *0,95* o que caracteriza uma predição quase perfeita.

5. Conclusão e Trabalhos Futuros

O presente trabalho teve como objetivo principal a classificação de algoritmos para análise de sentimentos no domínio da Copa do Mundo FIFA 2018 no Twitter. Logo, através de etapas de desenvolvimento e validações efetuadas conclui-se que o objetivo foi alcançado. Inicialmente foram expostos cinco objetivos específicos, os quais também foram contemplados. Este estudo de caso coletou dados textuais relacionados ao evento almejado. Além disto, rotulou-se uma amostragem significativa da base de dados coletada do Twitter. Na sequência foram categorizados os sentimentos e opiniões presentes na base de dados extraída para amostragem rotulada de modo a utilizar os classificadores do iFeel.

Foram validados os níveis de acurácia dos classificadores comparado a rotulação humana. E, por fim, foi classificada a base de dados coletada no idioma inglês, visto que durante a etapa de validações este foi o idioma com melhores resultados apresentados.

Aliado às funcionalidades, desenvolvimento e validações estatísticas efetuadas, SAMS é um estudo de caso capaz de identificar dentre vários métodos, qual o melhor

método para um contexto específico. Isto significa que as predições dos classificadores utilizados não acertaram por um simples acaso, mas sim, porque cientificamente apresentaram-se na faixa de classificação mais próxima a predição perfeita, caracterizando assim um diferencial de suma importância deste trabalho perante aos estudos correlatos explanados.

Em resumo, mais de 1 milhão de tweets foram coletados, juntamente com os 4 mil tweets rotulados humanamente, além de, 400 mil tweets classificados, bem como as matrizes de confusão e acurácia geradas para cada um dos vários métodos estudados. Por fim, o presente trabalho também possibilitou a geração do índice de concordância dos classificadores e validações efetuadas sobre o estudo e caracterizaram os principais resultados obtidos sobre os objetivos previamente estipulados.

Em trabalhos futuros pretende-se caracterizar as postagens coletadas a respeito da Copa do Mundo FIFA 2018. Além disso, comparar os métodos estudados em outros eventos relevantes de tamanha abrangência.

Referências

Araújo, M., Diniz, J. P., Bastos, L.and Soares, E., Júnior, M., Ferreira, M., Ribeiro, F., and Benevenuto, F. (2016). ifeel 2.0: A multilingual benchmarking system for sentence-level sentiment analysis. In *Proceedings of the International AAAI Conference on Web-Blogs and Social Media*, Cologne, Germany.

Benevenuto, F., Ribeiro, F., and Araújo, M. (2015). Métodos para análise de sentimentos em mídias sociais. In *Brazilian Symposium on Multimedia and the Web (Webmedia)*, Manaus, Brasil.

FIFA. Site oficial do evento copa do mundo fifa 2018. `https://www.fifa.com/worldcup`. Acesso em junho de 2018.

Gomes, H. J. C. (2013). Análise de sentimentos na classificação de notícias. In *Proceedings of the 8th Iberian Conference on Information Systems and Technologies (CISTI)*, Lisboa, Portugal.

Kim, J. W., Kim, D., Keegan, B., Kim, J., Kim, S., and Oh, A. (2015). Social media dynamics of global co-presence during the 2014 fifa world cup. In *CHI'15 Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, Seoul, Republic of Korea. ACM New York, NY, USA ©2015.

Landis, J. and Koch, G. (1977). An application of hierarchical kappa type statistics in the assessment of majority agreement among multiple observers. *Biometrics*, 33(2):363–374.

Rezende, S. O. (2005). Mineração de dados. In *Encontro Nacional de Inteligência Artificial e Computacional (ENIAC)*, São Leopoldo, Brasil.

Tan, A. H. (1999). Text mining: The state of the art and the challenges. In *Proceedings of the PAKDD Workshop on Knowledge Disocovery from Advanced Databases*, Beijing, China.

Seu Sangue, Minha Vida (SSMV): Um Projeto de Responsabilidade Social

Gustavo Satheler, Jéssica Ribeiro, Judson Moreira, Michael Martins, Rodrigo Barbosa, Sabrina Winckler, Maicon Bernardino, Elder Rodrigues

Universidade Federal do Pampa (UNIPAMPA)
Av. Tiarajú, 810, Ibirapuitã – Alegrete, RS – Brasil
Laboratory of Empirical Studies in Software Engineering (LESSE)

{gustavosatheler, jessicafaveror, judson.henrique01}@gmail.com
{michaelmartins096, roliveira.loki, sabrinacarlew}@gmail.com
bernardino@acm.org, eldermr@gmail.com

Abstract. *The donation rate in Brazil is below the ideal average. Based on this problem it was proposed the tool Seu Sangue, Minha Vida (SSMV), a web-based application that helps in the process of blood donation, focused on social aspects. Besides that, a survey of related jobs and similar applications was conducted. To evaluate the proposed tool, a quiz was developed to to obtain opinions and critics from the community. Based on the results more than half of the people has had need of donation and most believe that the tool could help the process. Therefore, despite the existence of similar systems, the proposed tool presents new differentiated features.*

Resumo. *O índice de doação no Brasil está abaixo da média ideal. Com base neste problema foi proposta a ferramenta Seu Sangue, Minha Vida (SSMV), focado em uma aplicação Web que auxilie no processo de doação de sangue, voltado à aspectos sociais. Além disso, uma pesquisa de trabalhos relacionados e aplicações semelhantes foi realizada. Para avaliar a ferramenta proposta, um questionário para obter opiniões e críticas dos usuários foi desenvolvido. Com base em seus resultados mais da metade das pessoas já precisaram de doação e a maior parte considera que a ferramenta poderia auxiliar o processo. Assim, apesar de já existirem sistemas similares, a ferramenta proposta apresenta novas funcionalidades diferenciadas.*

1. Introdução

Atualmente no Brasil, apenas 1,6% da população é doadora de sangue, sendo consenso que este índice está abaixo da média considerada ideal pela OMS (Organização Mundial da Saúde), o qual varia de 3% a 5% [Beraldo 2018]. Um dos fatores que colabora para este baixo índice é a falta de informações específicas, como a necessidade de determinados tipos sanguíneos em hemocentros ou a disponibilidade de doadores aptos.

O objetivo desse artigo é apresentar uma proposta de um sistema que possa auxiliar neste problema, onde será desenvolvida uma aplicação Web que auxilie no processo de doação de sangue, buscando doadores e divulgando a necessidade de doação de hemocentros.

Foi desenvolvida uma versão do sistema inicialmente como um trabalho acadêmico em 2017. Essa versão não foi disponibilizada e será utilizada somente como protótipo para a implementação do sistema proposto por este artigo.

Para melhor aplicação do sistema foi desenvolvido um questionário online com uma demonstração em vídeo de um novo protótipo da aplicação Web e algumas de suas funcionalidades. Neste questionário buscou-se ouvir a opinião de possíveis usuários sobre o funcionamento do sistema, assim como sugestões e críticas sobre o mesmo. A partir dos resultados obtidos por meio desta pesquisa, foram revisados os requisitos de software e decisões de projeto, assim como o funcionamento dos processos de utilização do sistema.

Este artigo está organizado conforme segue. A Seção 2 aborda a descrição e processos do sistema em si. A Seção 3 mostra os aspectos sociais considerados e metodologias utilizadas para validar os requisitos. A Seção 4 apresenta os requisitos de software revisados, mostrando o que cada ponto da aplicação permite o usuário fazer. A Seção 5 aborda os resultados da pesquisa por questionário online, assim como a validação de possíveis ameaças do projeto. A Seção 6 discute sobre os trabalhos relacionados, revisando plataformas existentes que possuem o mesmo objetivo de aplicação. A Seção 7 apresenta a conclusão das análises dos dados recolhidos, as considerações acerca do desenvolvimento do sistema e mostra a intenção de trabalhos futuros relacionados a aplicação.

2. SSMV - Uma Aplicação Web para Doação de Sangue

O Seu Sangue, Minha Vida (SSMV) é uma proposta com objetivo de apoiar o processo de doação de sangue. Servirá como uma aplicação Web, sem fins lucrativos. A proposta é uma plataforma que promove a conexão entre doadores, receptores e hemocentros, com o intuito de que assim ambos tenham acesso às informações sobre doações, carências e disponibilidade de tipos sanguíneos. Assim, hemocentros terão a possibilidade de manter suas reservas de sangue sempre abastecidas. Esta é uma iniciativa que visa utilizar tecnologia para ajudar a salvar vidas, conectando pessoas por todo território nacional.

2.1. Processo de Doação de Sangue

A proposta de processo de doação de sangue é apresentada na Figura 1. O processo é divido em 3 perfis: solicitante, doador, hemocentro. O processo é composto por 10 atividades, sendo cada uma delas descritas em detalhes:

Solicitar Doação de Sangue: Esta atividade é realizada usando a aplicação Web por um solicitante. Para a solicitação é necessário preencher um questionário, informando o nome do beneficiário que receberá o sangue, o tipo sanguíneo, a data limite e a urgência;
Divulgar Solicitação de Doação de Sangue: Uma vez realizada a solicitação, são encaminhadas notificações por meio da aplicação Web e por e-mail, além de permitir a divulgação em redes sociais *e.g.* Facebook e Twitter. Nesta atividade, as notificações só serão encaminhadas para usuários com a opção de doação de sangue habilitada e que possuam sangue compatível com o solicitado;
Receber Solicitação de Sangue: Esta atividade é realizada após o recebimento da notificação, na qual o sistema permite que o doador opte por fazer a doação. Caso ele não responda a solicitação dentro do prazo estipulado, o processo será finalizado;
Registrar Intenção de Doação: Nesta atividade, o doador informa que tem intenção de doar, a partir da notificação recebida, informando o local que irá realizar a doação.

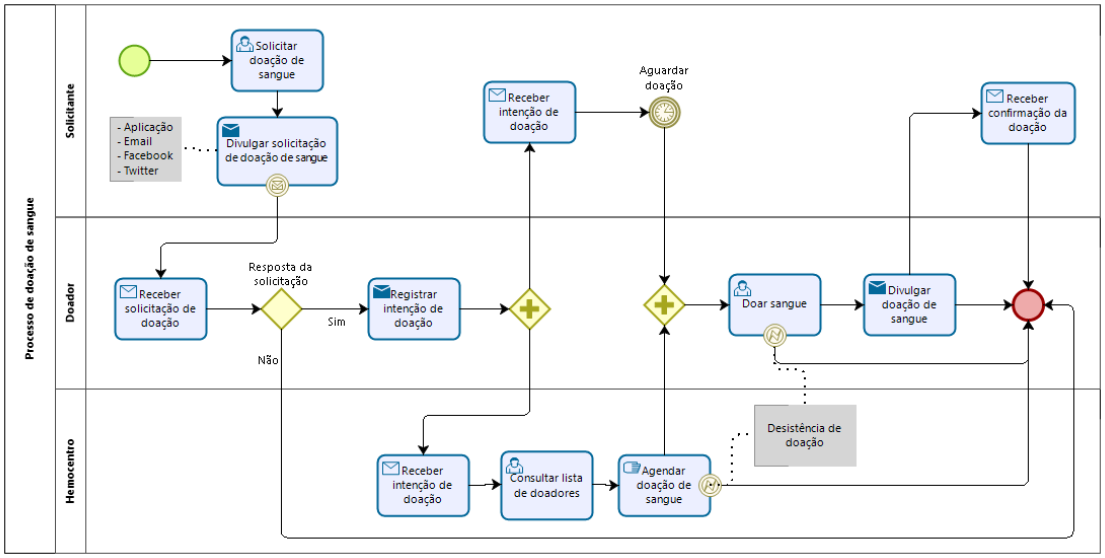


Figura 1. Processo de doação de sangue

E então são encaminhadas notificações por meio da aplicação Web e por e-mail para o solicitante e o hemocentro informado;
Receber Intenção de Doação: Esta atividade é realizada após o recebimento da notificação, na qual o solicitante aguarda até a efetivação da doação de sangue;
Consultar Lista de Doadores: Esta atividade é realizada por um funcionário do hemocentro, em que ele tem acesso a uma lista de doadores compatíveis através do sistema;
Agendar Doação de Sangue: Esta atividade é realizada por um funcionário do hemocentro, em que é possível contatar doadores compatíveis;
Doar Sangue: Essa atividade é realizada presencialmente pelo doador no hemocentro, em que ocorre a coleta de sangue;
Divulgar Doação de Sangue: Após doar sangue, o doador recebe uma declaração de doação, a qual possui um código que poderá ser protocolado no sistema, para que assim fique registrado que a doação foi de fato realizada;
Receber Confirmação de Doação: Finalmente, na última atividade do processo, o solicitante recebe uma notificação de que uma doação foi efetivada.

3. Decisões de Projeto

A ideia central para o desenvolvimento desta ferramenta se baseia no intuito de cobrir necessidades relacionadas a aspectos sociais. Nesta seção será discutido cada uma das decisões de projeto (*Design Decision* - DD) tomadas com o objetivo de validar os requisitos descritos na Seção 4.

DD1) *Realizar pesquisa de interesse acerca da viabilidade de implementação da ferramenta.* Foi realizada uma pesquisa de interesse, por meio de um questionário online, visando descobrir qual a real aplicabilidade da ferramenta e a opinião das pessoas acerca da mesma. Para a aplicação, foi realizada uma demonstração de um protótipo navegável de alta fidelidade do processo de solicitação de doação de sangue. Também foi efetuada uma entrevista exclusiva com o hemocentro da cidade de Alegrete, o qual cobre

5 hospitais da região da fronteira oeste do Rio Grande do Sul, na qual teve como objetivo descobrir as limitações e interesses do hemocentro em relação a proposta do sistema.

DD2) *Realizar pesquisa de sistemas que possuam a mesma proposta que o SSMV.* Foram pesquisados sistemas semelhantes ao que se planeja desenvolver e selecionados os três que mais possuem reconhecimento na área. Os critérios levados em consideração para a seleção foram: visibilidade, número de usuários, divulgações em fontes renomadas e principalmente a proposta e as funcionalidades do sistema.

DD3) *Realizar análise de publicações relacionadas.* Foram realizadas buscas de artigos relacionados no Google Acadêmico que tratam do mesmo assunto deste estudo. Com base nos resultados da busca foram selecionados os três principais artigos que possuíam características mais semelhantes.

DD4) *Realizar brainstorming para a aplicação Web SSMV.* Foi realizado um *brainstorming* buscando os possíveis diferenciais da ferramenta que se pretende desenvolver.

4. Requisitos

Com o objetivo de definir os processos elementares de negócio associados ao sistema, foi realizado um levantamento de requisitos com base nas necessidades essenciais do projeto a fim de demonstrar o processo de doação de sangue. Além disso, foi utilizado os resultados do questionário para revisar e adicionar requisitos. Como resultado dessa atividade, mapeou-se diversas aplicações similares, analisando suas funcionalidades em comum.

RQ1) *A aplicação Web deve permitir que qualquer tipo de usuário realize solicitação de doação de sangue.* Essa funcionalidade visa informar aos potenciais doadores quando alguma pessoa estiver precisando do seu tipo sanguíneo, definindo na própria solicitação o tipo sanguíneo desejado, bem como a urgência e o beneficiado. O usuário pode divulgar a solicitação nas redes sociais se desejar.

RQ2) *A aplicação Web deve permitir que uma requisição de doação seja enviada para outros usuários próximos a localização atual do solicitante.* Essa função faz com que as requisições feita por um determinado usuário seja enviada para doadores cadastrados mais próximos a localização atual do requisitor.

RQ3) *A aplicação Web deve permitir que qualquer usuário gerencie suas notificações.* Deste modo os usuários poderão retornar respostas às solicitações de doação recebidas e/ou simplesmente gerenciar os avisos que o sistema disponibiliza.

RQ4) *A aplicação Web deve permitir que os hemocentros possam atualizar seus horários de funcionamento.* Essa funcionalidade tem como objetivo manter informações atualizadas sobre o hemocentro.

RQ5) *A aplicação Web deve permitir que os usuários registrem a intenção de doar*. Facilita o processo de agendamento ao doador e ao hemocentro. O registro de intenção serve como um aviso prévio ao hemocentro de que alguém possivelmente irá doar, além de informar o doador sobre as informações que facilitem o agendamento da doação no hemocentro mais próximo.

RQ6) *A aplicação Web deve disponibilizar informações confiáveis ao usuário.*

Este requisito disponibiliza informações confiáveis coletadas de fontes consistentes, visando suprir toda e qualquer dúvida acerca da doação de sangue.

RQ7) *A aplicação Web deve permitir que doadores registrem suas doações e as divulguem.* Mantém um controle de doações realizadas. Para certificar registros confiáveis, é possível a divulgação somente com o código da declaração de doação emitido após realização da doação.

RQ8) *A aplicação Web deve informar ao doador quando doar novamente.* Tem como finalidade auxiliar os usuários a manter a regularidade da doação por meio de uma notificação. Este requisito foi adicionado após a revisão dos resultados de respostas do questionário.

RQ9) *A aplicação Web deve disponibilizar ao hemocentro sua lista individual de doadores.* Auxilia o hemocentro possibilitando um acesso virtual a sua lista de doadores, para que quando precisar, possa entrar em contato utilizando o SSMV.

RQ10) *A aplicação Web deve calcular um ranking de mais frequentes doadores* Esse requisito utiliza técnicas de *gamification*, que classifica doadores em um *ranking* e a cada doação o usuário acumulará pontos que podem ser convertidos em benefícios e aumenta sua colocação.

5. Avaliação: *Survey*

Nesta seção serão apresentados os resultados referentes a pesquisa realizada utilizando um questionário (*survey*) para coleta dos dados, bem como considerações e questões levantadas na entrevista com o hemocentro de Alegrete.

5.1. Resultados Preliminares

O questionário[1] possui 6 sessões, sendo compostas respectivamente por informações do projeto e acesso ao vídeo no formato de um tutorial mostrando o processo de doação de sangue pelo sistema, termo de consentimento livre e esclarecido, identificação (se é doador, não doador ou hemocentro), uma sessão especial voltada exclusivamente a hemocentros, dados do perfil do entrevistado e por fim 8 questões escalares e descritivas, sobre avaliação da viabilidade da proposta [Sanches 2010].

Esta pesquisa foi encaminhada aos e-mails dos hemocentros do Brasil, a algumas ONGs de doação de sangue, a comunidade acadêmica da Universidade Federal do Pampa, bem como foi divulgado em redes sociais, tais como: *Instagram, WhatsApp* e *Facebook* dos autores deste projeto. Vale ressaltar que a coleta de dados foi realizada entre os dias 3 e 20 de setembro de 2018.

Foi realizado um teste piloto antes de disponibilizar o questionário ao público, no qual cada autor responsável pelo projeto e ao menos uma pessoa não vinculada ao projeto responderam às questões, a fim de validar o instrumento de coleta de dados, bem como a consistência e relevância das questões propostas.

5.2. Análise dos Resultados

Obteve-se um total de 109 respostas, sendo estes 78 doadores, 23 não doadores, 3 candidatos a doadores, 3 que não podem ser doadores e 2 hemocentros. Desta amostra de

[1]O questionário na íntegra está disponível em: `http://bit.ly/SSMV_ERES2018`

participantes da pesquisa, pelo menos 52,8% já precisou requisitar alguma doação de sangue.

Conforme ilustrado no gráfico da Figura 2, a investigação apontou que 61,4% (67) das pessoas que responderam a pesquisa concordam totalmente que a ferramenta ajudaria a reduzir o tempo de espera e a falta de doação de sangue nos hemocentros, ao passo em que 23,9% (26) somente concordam, 13,8% (15) estão indecisos e 0,9% (1) discordam.

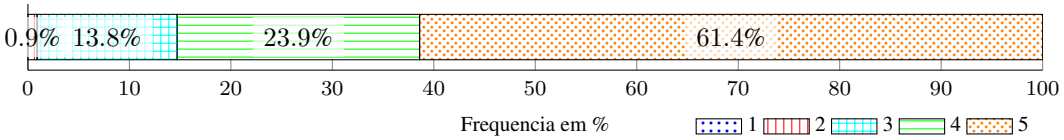


Figura 2. Nível de concordância dos respondentes em relação a Questão 3

Em relação ao impacto positivo no uso de mídias sociais para a divulgação de campanhas de doação de sangue, a Figura 3 mostra que 87% (95) dos respondentes concordam totalmente que seria, de fato, positivo, enquanto 11,1% (12) só concordam e 1,9% (2) estão indecisos.

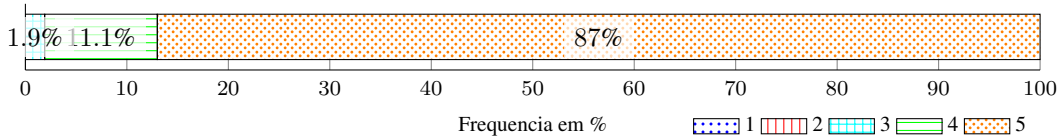


Figura 3. Nível de concordância dos respondentes em relação a Questão 4

Além destes dados, foram sondadas opiniões e sugestões dos entrevistados acerca da aplicação Web. Em geral o ponto de vista em relação a proposta foi positivo. No que se refere as sugestões recebidas, boa parte do sistema já cobria, só não haviam sido demonstradas no tutorial. Ademais, algumas das sugestões propostas não são viáveis de serem incorporadas ao sistema, por conta das limitações apontadas pelo hemocentro.

Segundo a assistente social e o analista de sistemas do hemocentro de Alegrete, seria viável a utilização da aplicação Web por parte deles, pois como foi informado, eles já utilizaram um sistema semelhante, porém desistiram pois o mesmo exigia uma carga de trabalho além do normal. Foi discutido na entrevista quais são as maiores limitações deles em relação à utilização de sistemas deste tipo. Dentre as principais estão: fornecimento de dados acerca dos estoques de sangue, bem como informações de doadores, à quem será destinado o sangue, dentre outras questões burocráticas que estão sempre sendo atualizadas e os impedem de fazer algumas coisas.

5.3. Ameaças ao Estudo

Nesta subseção, discute-se as ameaças a este estudo e as soluções encontradas para mitigar estas [Wohlin et al. 2012] [Lima et al. 2012].

Para mitigar problemas relacionados ao entendimento das questões contidas no questionário foi disponibilizado um vídeo no qual são explicadas as funcionalidades mais relevantes relacionadas ao processo que deseja-se validar no sistema. Essas questões foram elaboradas com base no que foi exposto. Ademais foi feito um teste piloto, que

permitiu validar a consistência e entendimento das questões de acordo com o que foi explicado, prevenindo possíveis problemas futuros.

Na elaboração do questionário uma das preocupações mais relevantes foi com o tamanho do mesmo, visando elaborar-lo com poucas questões e que estas fossem objetivas. Considerando que uma alta probabilidade do público destinado não ter disposição de tempo para responder o questionário, optou-se por separar as questões em seções bem definidas e não requisitar muitas respostas dissertativas.Ressalta-se que na computação dos dados não foi considerada a categoria de identificação selecionada pelo entrevistado.

Buscando um resultado significativo para a pesquisa, esta foi encaminhada para sessenta e três (63) hemocentros do Brasil, quatro (4) ONGs de doação de sangue, para toda a comunidade acadêmica da Universidade Federal do Pampa e divulgada nas redes sociais dos autores. Isso gerou repostas significativas em relação a viabilidade da implementação do sistema, apesar da baixa quantidade.

Uma das ameaças recorrentes em pesquisas deste tipo é o ponto de vista de cada pessoa. Para mitigar esta buscou-se atingir o público mais amplo possível. Vale salientar que os resultados obtidos foram suficientes para validar o que se desejava com o questionário.

6. Trabalhos Relacionados

Existem inúmeras plataformas e publicações no Brasil com o desígnio de facilitar a doação de sangue. Nesta seção serão abordados três das que mais se destacaram na pesquisa, sendo comparado as funcionalidades de cada uma em relação ao SSMV, conforme apresentado na Tabela 1.

Funcionalidades	**Hemoliga**	**#PartiuDoarSangue**	**Hemogram**	**SSMV**
Aplicação Web	x	x		x
Aplicação *Mobile*	x	x	x	
Visualizar registros de doações	x	x	x	x
Solicitar doação de sangue			x	x
Fornecer localização de hemocentro	x	x	x	x
Gerenciamento de solicitação		x	x	x
Controle de estoque de sangue	x			
Promover campanhas de doação	x			
Acompanhar doação				x
Acompanhar campanha	x			
Registro da intenção de doar				x
Divulgação de doação				x
Ranking de doadores				x
Disponibilizar lista de doadores		x	x	x
Registro de doação	x	x	x	x

Tabela 1. Tabela de Comparações de Funcionalidades

6.1. Plataformas Analisadas

As plataformas que correspondem aos critérios citados na Seção 3 analisadas na Tabela 1 foram: `HemoLiga`[2], `#PartiuDoarSangue`[3] e `Hemogram`[4].

6.2. Publicações Relacionadas

Com base na problemática da área, optou-se pelo estudo de alguns artigos relacionados [Moraes 2015] [Monteiro 2013] [Silva 2017] ao desenvolvimento de aplicações com propostas semelhantes a do SSMV.

A relação encontrada entre eles foi o foco em aplicações Web integradas com dispositivos móveis. Em geral, os requisitos propostos por cada artigo não são tão distintos dos propostos neste artigo. Entretanto, o SSMV se sobressairia sendo mais eficaz em termos de sua usabilidade, portabilidade e a aceitação da comunidade.

7. Considerações Finais

Este trabalho apresentou requisitos essenciais, funcionalidades, decisões de projeto e processos que foram levantados e definidos em colaboração com o Hemocentro de Alegrete e possíveis usuários, no contexto de uma proposta de desenvolvimento de uma ferramenta que auxilie no processo de doação de sangue. Com base nisso, notou-se que as ferramentas existentes não atendem a todos os requisitos identificados. Por este motivo foram apresentadas as decisões de projeto no desenvolvimento de uma aplicação Web que atende a todos esses requisitos.

Como trabalhos futuros, pretende-se dar continuidade ao desenvolvimento da ferramenta e cobrir outros tipos de doações, como de medula óssea e cabelo, com o intuito de ajudar mais pessoas na sociedade.

Referências

Beraldo, N. (2018). Jovens entre 18 e 29 anos são os maiores doadores de sangue no país. Disponível em `https://goo.gl/8XvYtT`. Acessado em 21/08/2018.

Lima, V. C. M., Neto, A. G. S. S., and Emer, M. C. F. P. (2012). Investigação experimental e práticas ágeis: ameaças à validade de experimentos envolvendo a prática ágil programação em par. In *3º Workshop Brasileiro de Métodos Ágeis*.

Monteiro, J. T. (2013). SADBS – Sistemas de Agendamentos e Doações Web e Mobile de Banco de Sangue. Universidade Tecnológica Federal do Paraná – UTFPR.

Moraes, E. J.; Moreira, R. F. (2015). Ferramenta para Gestão de Hemocentros com aplicativo para divulgação de doadores de sangue no Facebook. *ALTEC Brasil*.

Sanches, S. (2010). Instrumentos da pesquisa qualitativa. Acessado em 01/09/2018.

Silva, R. G.; de Araujo, D. R. B. (2017). Proposta de um sistema de apoio à doação sanguínea baseado em gamificação. *Journal of Engineering and Applied Research*, 2(2).

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., and Regnell, B. (2012). *Experimentation in Software Engineering*. Springer.

[2]HemoLiga: `http://hemoliga.com.br/`
[3]#PartiuDoarSangue: `https://partiudoarsangue.com.br/`
[4]Hemogram: `http://bit.ly/hemogramApp`

Anais da 2ª Escola Regional de Engenharia de Software (ERES 2018)
em Dois Vizinhos, PR, Brasil, 22 a 24 de outubro de 2018

Em Direção à Engenharia Reversa Adaptativa de Binários e Códigos para Modelo

Giliardi Schmidt[1], Guilherme Bolfe[1], Fábio Paulo Basso[1], Elder Rodrigues[1], Maicon Bernardino[1]

[1]Universidade Federal do Pampa (UNIPAMPA)
Av. Tiarajú, 810, Ibirapuitã – Alegrete, RS – Brasil
Laboratory of Empirical Studies in Software Engineering (LESSE)

gili.schmidt@hotmail.com
guilhermebolfe11@gmail.com, fabiopbasso@gmail.com,
eldermr@gmail.com, bernardino@acm.org

Abstract. *Through code-to-model reverse engenireering techniques it is possible to extract structural information from source code to a level independent of the programming language adopted. Some benefits assossiated with these techniques include ease of understanding of poorly documented systems as well as the (semi-) automatic migration of applications from one technology to another. This paper investigates the possibility for the adaptative extraction of structural characteristics from source codes written in the C++ laguague to a UML model, presenting a proof concept and discussions about future works.*

Resumo. *Por meio de técnicas de engenharia reversa de código-para-modelos é possível extrair informações estruturais de códigos-fonte para um nível independente da linguagem de programação adotada. Alguns benefícios associados com essas técnicas incluem facilidade para o entendimento sobre sistemas mal documentados e também a migração (semi-)automática de aplicações de uma tecnologia para outra. Este artigo investiga a possibilidade para a extração adaptativa de características estruturais de códigos-fonte escritos em C++ para modelos UML, apresentando uma prova de conceito e discussões sobre trabalhos futuros.*

1. Introdução

Muitos produtos de software com o passar do tempo ficam obsoletos, devido à constante evolução das necessidades dos usuários e da evolução natural das tecnologias disponíveis, necessitando assim manutenções e/ou adições de novas funcionalidades. Para realizar estas alterações no software, faz-se necessário entender o código-fonte do sistema: sua estrutura, algoritmos e fluxos de execução. Porém, há situações em que este código-fonte está documentado de maneira inconsistente ou superficial, ou até mesmo não possua nenhum documento de apoio, tornando o entendimento do sistema uma tarefa custosa.

Além da necessidade de alterar o código-fonte do software, há casos em que deseja-se efetuar as tarefas de *Verificação e Validação* (V&V) no sistema. Estas duas tarefas podem necessitar de uma documentação mais abrangente sobre o código-fonte, como a estrutura estática do mesmo [Korshunova et al. 2006]. O processo de verificação e validação analisa exaustivamente e testa o software para determinar se ele executa

suas funções, para garantir que ele realiza suas funções de forma correta e mensura sua qualidade e confiabilidade [D. R. Wallace 1989].

Para obter esta documentação faltante, é possível utilizar a engenharia reversa, onde parte-se de um código-fonte pronto e eleva-se para um nível mais abstrato, um modelo. O processo de obtenção de representações úteis de alto nível que se da a partir de código-fonte é chamado engenharia reversa [Brunelière et al. 2014]. Ferramentas de modelagem UML - *Unified Modeling Language* como Astah e EA, peritem ao engenheiro de software aplicar a engenharia reversa de código para diagramas de classes. No entanto, o modelo revertido é poluído com detalhes de APIs, o que dificulta sua legibilidade em atividades de V&V. Além disso, para ser utilizado com propósitos de migração, o modelo revertido precisa ser constantemente e manualmente ser refinado para um outro livre de detalhes de implementação de plataformas alvo.

Neste artigo apresenta-se uma nova abordagem para engenharia reversa no refinamento automatizado de modelos em níveis independentes de plataforma. O refinamento deve-se dar de modo adaptativo para contextos inter-organizacionais [Basso et al. 2017]. Como resultado, espera-se gerar modelos em níveis de abstração úteis para a execução de atividades de V&V e de migração de aplicações. Portanto, apresenta-se uma nova perspectiva de investigação em relação à outras abordagens para engenharia reversa existentes [Korshunova et al. 2006, Brunelière et al. 2014, Méndez-Acuña et al. 2017].

O artigo é organizado conforme segue: A Seção 2 discute nossa proposta e a Seção 3 apresenta um resumo dos principais trabalhos relacionados. A abordagem atual para engenharia reversa é demonstrada na prova de conceito apresentada na Seção 4. Na Seção 5 apresenta-se uma discussão sobre a viabilidade para aplicação da abordagem FOMDA para engenharia reversa adaptativa (FOMDA-reverse) e os próximos passos para estendê-la e viabilizá-la em contextos inter-organizacionais de reutilização. For fim, a Seção 6 apresenta as conclusões.

2. Visão Geral da Abordagem FOMDA-Reverse

A contribuição deste trabalho é um estudo analítico-prático discutindo sobre a viabilidade de aplicação da engenharia reversa adaptativa. Inicialmente, extraímos de um código na linguagem C++ as características estruturais estáticas, como classes, atributos de classes e assinaturas de métodos, para um diagrama de classes UML. Este diagrama UML, por fim, é salvo em um arquivo no formato XMI.

Para conseguir elevar o código a um nível mais alto é necessário criar uma árvore sintática abstrata - AST (*Abstract Syntax Tree*), esta que por sua vez possui diversas formas de ser construída, utilizando diferentes ferramentas. Neste trabalho optou-se por utilizar a ferramenta ANTLr [Parr 2013], o qual se adequou mais à necessidade da elaboração de uma prova de conceito devido o fato de possuir gramaticas prontas. Após sua construção, a AST pode ser utilizada para analise de softwares, desenvolvimento de softwares e construção de compiladores [G. Fischer 2007].

Para a implementação de uma nova abordagem para engenharia reversa adaptativa, parte-se de um trabalho anterior já consolidado. Em experiencias anteriores executadas por meio da abordagem FOMDA [Basso et al. 2013], modelos UML foram adotados em processos adaptativos de transformações de modelo-para-código (M2C), portanto permitindo a customização na geração de código para contextos inter-organizacionais.

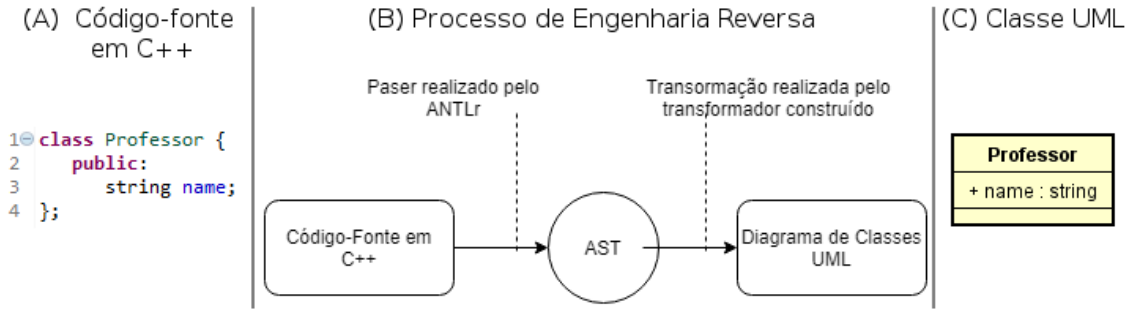


Figura 1. Um simples processo de transformação do código-fonte para modelo

Neste artigo apresenta-se uma prova de conceito que investiga a viabilidade para se aplicar o processo contrário, em transformações de código-para-modelo (C2M). Como resultado, espera-se o desenvolvimento de suporte ferramental que permita reverter código para modelos em níveis independentes de características de plataformas. Portanto, nossa proposta é chamada de FOMDA-reverse e permitirá executar atividades automáticas de refinamento de modelos independentes de plataforma baseadas em contextos.

De momento, esta pesquisa encontra-se em estágio inicial de caracterização do problema. Assim, este estudo investigou os elementos necessários para o desenvolvimento de componentes que aplicam transformação de código-para-modelo seguindo uma técnica comum aos trabalhos relacionados. Ou seja, adotou-se uma abordagem simples de engenharia reversa, utilizando-se de duas etapas para a transformação do código-fonte para o modelo UML. Portanto, no estágio atual não foram considerados diferentes contextos e componentes adaptativos para reverter modelos de código.

A Figura 1 apresenta uma visão geral deste simples processo de transformação como segue:

Passo 1: Levantamento de características estruturais para uma AST. Nesse passo é gerada uma AST utilizando a ferramenta ANTLr. Focou-se no desenvolvimento de um componente para transformação construído para extrair informações de estrutura estática de um código-fonte escrito na linguagem C++ para um diagrama de classes UML, utilizando a ferramenta ANTLr.

Passo 2: Transformação dos dados em conformidade com a AST para uma representação comum a plataformas/linguagens cruzadas em programação orientada à objetos (C++, Java, C#, PHP, etc). Nesse sentido, o transformador construído navega por esta AST, extraindo informações sobre as classes, atributos e métodos destas classes, gerando um modelo composto de um diagrama de classes UML. Adotou-se a UML como linguagem de representação comum devido à sua grande expressividade para representar sistemas orientados à objetos.

3. Trabalhos Relacionados

No trabalho apresentado por [Korshunova et al. 2006] é feita a engenharia reversa de código-fonte para modelos UML. A ferramenta CPP2XMI possibilita gerar diversos diagramas, tais como: classes, sequência e atividades. Estes diagramas são salvos no formato XMI.

No trabalho apresentado por [Fleurey et al. 2007] foi realizada a migração de um conjunto de aplicações de uma instituição financeira. Parte do trabalho de migração consistiu em extrair informações sobre o código-fonte das aplicações existentes para modelos utilizando ASTs.

O projeto *open-source* MoDisco [Brunelière et al. 2014] provê um *framework* genérico e extensível para engenharia reversa. Diferentemente de outras soluções de engenharia reversa que apenas geram modelos UML a partir de tecnologias específicas, e vice-versa, este *framework* tem como objetivo prover suporte para a geração de modelos a partir de diferentes metamodelos, portanto tendo como entrada diversos tipos de artefatos como código-fonte, banco de dados, etc.

Estes trabalhos aplicam engenharia reversa utilizando-se de processos de extração de informações do código sem considerar o contexto. MoDisco é um suporte ferramental rico e flexível para permitir customização no processo de engenharia reversa, mas ele ainda é limitado para a execução adaptativa dos componentes de transformação reversa. Isso acarreta em modelos revertidos com informações desnecessárias, que atrapalham a execução de atividades de V&V e de migração de aplicações.

Para resolver esse problema, é necessário tornar o processo de engenharia reversa adaptativo. Essa necessidade é diferente de se reverter um modelo para arquiteturas de linhas de produto, por exemplo, como o realizado em [Méndez-Acuña et al. 2017]. Neste trabalho, apesar de tratar do termo "contexto", o processo de extração de detalhes de implementação continua sendo estático. Diferentemente, FOMDA-reverse tornará este processo dinâmico, permitindo a inclusão de componentes adaptativos que extraem dados independentemente de se tratar de um código para linha de produtos ou um software simples.

4. Implementação do Componente para Transformação C2M

Com a utilização da ferramenta ANTLr, foi gerado um *parser*, para a linguagem de programação Java. Para a geração deste *parser*, utilizou-se uma gramática pronta [Sanchez]. A partir do *parser* gerado pelo ANTLr, estendeu-se a classe abstrata *BaseVisitor*, gerada automaticamente pelo ANTLr, e criou-se diversos *Visitors* para extrair as informações necessárias da AST gerada. A Figura 2 apresenta um dos *Visitors* criados, sendo o responsável por extrair informações sobre uma classe.

Assim, o transformador construído neste trabalho aplica os *Visitors* gerados na AST e cria os elementos do diagrama UML com base no tipo de nó visitado.

O transformador gerado neste trabalho tem a capacidade de extrair as informações estruturais de classes: o nome da mesma, quais são seus atributos e funções, a visibilidade destes atributos e funções e a assinatura (parâmetros de entrada e tipo de retorno) destas funções.

Para exemplificar o seu funcionamento, considere o que é apresentado nas Figuras 1(A) e 2. Ao fornecer como entrada o código-fonte da Figura 1 (A) ao *parser* criado pelo ANTLr, é gerada de forma automatizada uma AST.

Ao aplicar o *Visitor ClassSpecifierVisitor* ao nó raiz de uma AST, sempre que houver um nó do tipo *classspecifier* (que corresponde à uma classe), o método *visitClassSpecifier(ClassSpecifierContext ctx)* será invocado. Para extrair o nome desta classe,

```
9  public class ClassSpecifierVisitor extends BaseVisitor<Class> {
10
11     public ClassSpecifierVisitor(Model model) {
12         super(model);
13     }
14
15     @Override
16     public Class visitClassspecifier(ClassspecifierContext ctx) {
17         Class c = null;
18
19         String name = ctx.classhead().classheadname().getText();
20
21         try {
22             c = super.createClass(name);
23
24             if (ctx.classhead().baseclause() != null) {
25                 ClassNameVisitor classNameVisitor = new ClassNameVisitor(model, c);
26                 ctx.classhead().baseclause().accept(classNameVisitor);
27             }
28
29             MemberSpecificationVisitor memberSpecificationVisitor = new MemberSpecificationVisitor(model, c);
30             ctx.accept(memberSpecificationVisitor);
31
32         } catch (Exception e) {
33             e.printStackTrace();
34         }
35
36         return c;
37     }
```

Figura 2. Visitor gerado para visitar nós do tipo *classspecifier*

navega-se pelos filhos do nó em questão, como exemplificado na linha 19 da Figura 2. Após isso, são aplicados novos *Visitors* conforme necessário. Por exemplo, é aplicado um novo *Visitor*, que é responsável por extrair as informações de atributos e funções da classe (linhas 29 e 30 da Figura).

Como resultado, nossas transformações resultam em um modelo UML que é independente da plataforma de desenvolvimento e da própria linguagem C++. Para este modelo se dá a definição de: Modelo Independente de Plataforma (PIM) [Basso et al. 2013]. Por fim, o transformador construído chamado de "visitor" ainda não é adaptativo. Para tal, a seção 4.1 apresenta o planejamento para incremento de pesquisa existente que já traz suporte ferramental para a execução de transformações adaptativas.

4.1. Um Paralelo com a Abordagem FOMDA

Uma vez que o modelo UML é extraído a partir de um código-fonte em C++, ele é refinado para a geração de código em outra plataforma alvo, como ilustrado na Figura 3. Assim, o modelo é a entrada "PIM do Sistema A" para um processo de transformação, também chamado de *assembly*, configurado em FOMDA DSL [Basso et al. 2013].

De modo à permitir a execução adaptativa de componentes de transformação de modelos, a abordagem FOMDA oferece uma DSL para a representação de um Modelo de Domínio de Plataformas (PDM) [Basso et al. 2017], que é integrado em uma cadeia de transformação e com o modelo de entrada da seguinte forma:

1) O projetista do plano de migração utiliza as configurações (PDM + transformadores localizados em "Transformações M2M"), para refinar um modelo do sistema (PIM) para modelos em nível intermediário [Basso et al. 2016]. Para tanto, o projetista de aplicação precisa: A) selecionar as características não funcionais do PDM, gerando assim uma instância do PDM; B) indicar qual é o PIM que representa os requisitos funcionais do sistema; C) indicar os transformadores que ele vai utilizar para converter o PIM em

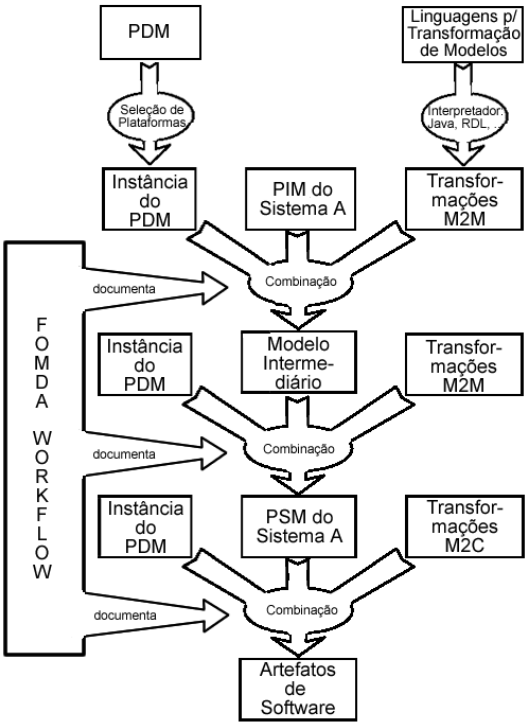


Figura 3. Modelos e Transformações Utilizadas pela Abordagem FOMDA

um novo modelo. Estas três tarefas do projetista de aplicação são identificadas na Figura 3 pela seta que combina a saída dos três modelos (a instância do PDM, o PIM e uma transformação M2M).

2) Tais configurações definem uma receita que organiza e documenta o processo de transformações de modelos que o projetista de aplicações deve seguir. Essa ordem precisa organizar as características do PDM, os elementos do PIM e as transformações.

3) Para organizar os refinamentos intermediários de modelos entre um PIM e um PSM, o projetista de transformações especifica e configura um Domínio de Cadeia de Transformações (TCDM). Ele é utilizado para fazer a combinação entre as características selecionadas no PDM, os elementos do PIM e as transformações.

4) O TCDM oferece ao projetista de aplicação uma receita para gerar cadeias de transformações para contextos intra-organizacionais. As cadeias, por fim, permitem transformar PIMs em PSMs por meio do mapeamento de características de plataformas que ditam como a composição de transformações adaptativas acontece.

5) O resultado final é ilustrado pela geração dos artefatos do sistema para uma nova plataforma de execução. Para mais informações, as contribuições em [Basso et al. 2013, Basso et al. 2014] demonstram elementos da FOMDA DSL que permitem execuções adaptativas de componentes de transformação.

5. Discussões

A abordagem FOMDA vem se mostrando efetiva para flexibilizar a geração de modelos e código no desenvolvimento de sistemas para múltiplos domínios [Basso et al. 2017].

A mesma foi utilizada em contextos reais de fábricas de software e se mostrou útil para reutilização de componentes de transformação de modelos [Basso et al. 2013]. Logo, para o próximo trabalho, planeja-se o desenvolvimento de geradores de código para diferentes plataformas, complementando assim o *framework* disponível nesta abordagem.

Além disso, como a abordagem foi concebida para executar processos de transformações adaptativos, a pergunta comum de pesquisa que pretende-se investigar é: Quais são as vantagens e desvantagens ao se introduzir os conceitos de variabilidade de comunalidade também no processo de engenharia reversa de um processo de migração de aplicações motivado neste artigo?

Como exemplificado em [Basso et al. 2016], realizar a engenharia reversa de código Java para modelos é uma tarefa crítica para o sucesso de abordagens para *Rountrip Engineering* [Kelly and Tolvanen 2008]. *Rountrip* permite a execução de uma abordagem de MDE que, tanto pode gerar o código a partir de um modelo de entrada, quanto reverter o código gerado para modelos. No entanto, a abordagem FOMDA nunca foi aplicada para processos de migração de aplicações.

Assim, propõem-se como objetivo de pesquisa derivado deste artigo, que a ordem da Figura 3 seja executada em ordem reversa. A meta para trabalhos futuros é estender a API FOMDA [Basso et al. 2014] para tornar os transformadores de engenharia reversa, como o demonstrado na Figura 2, adaptativos para o contexto do código. Isso terá implicações positivas para se executar *Rountrip Engineering* no futuro, sendo este um tópico de investigação bastante recente na literatura [Méndez-Acuña et al. 2017]. Assim, planeja-se esta meta de acordo com três pesquisas dirigidas:

- Engenharia reversa adaptativa de diretivas de APIs misturadas com sintaxe de linguagens de programação. A prova de conceito desenvolvida neste trabalho levou em conta um código puramente basado em C++. No entanto, os códigos reais são repletos de bibliotecas adicionadas ao conjunto de comandos padrão da linguagem. Logo, estes detalhes específicos de APIs devem ser recuperados também numa abordagem adaptativa para engenharia reversa;
- Engenharia reversa adaptativa de diretivas de *tag-libraries* misturadas com sintaxe de código de interfaces gráficas de usuário Web (JSP, JSF, etc.). Aqui o problema se caracteriza do mesmo modo que na proposta anterior, porém as informações estão sob a forma de *tags* XML, Javascript e arquivos semi-estruturados, e;
- Engenharia reversa adaptativa de binários. Também ruma-se nessa implementação para o desenvolvimento de casos de testes adaptativos [Basso et al. 2014], com a utilização de Java *Reflection* e *Annotations*.

6. Conclusão

O transformador gerado tem a capacidade de extrair as informações estruturais estáticas de códigos fonte escritos em C++. Porém, sua efetividade não foi testada e avaliada. Ou seja, pode haver situações em que o transformador não consiga extrair as informações corretas, ou até mesmo não funcione.

É possível citar como vantagem do trabalho realizado que, após abstrair as informações do código-fonte para um modelo, é possível exportá-lo para outras linguagens de programação. Assim, facilitando a migração de aplicações. Além de exportações e

migrações, após o processo de engenharia reversa estar aplicado, pode-se utilizar técnicas de verificação e validação nos modelos gerados, garantindo uma melhor qualidade das aplicações.

Referências

Basso, F. P., Oliveira, T. C., and Farias, K. (2014). Extending junit 4 with java annotations and reflection to test variant model transformation assets. In *29th Symposium On Applied Computing*, SAC'14, pages 1601–1608.

Basso, F. P., Oliveira, T. C., Werner, C. M., and Becker, L. B. (2017). Building the foundations for 'mde as service'. *IET Software*, 11:195–206(11).

Basso, F. P., Pillat, R. M., Oliveira, T. C., and Becker, L. B. (2013). Supporting large scale model transformation reuse. In *12th International Conference on Generative Programming: Concepts & Experiences.*, GPCE'13, pages 169–178.

Basso, F. P., Pillat, R. M., Oliveira, T. C., Roos-Frantz, F., and Frantz, R. Z. (2016). Automated design of multi-layered web information systems. *Journal of Systems and Software*, 117:612 – 637.

Brunelière, H., Cabot, J., Dupé, G., and Madiot, F. (2014). Modisco: A model driven reverse engineering framework. *Information and Software Technology*, 56(8):1012–1032. cited By 65.

D. R. Wallace, R. U. F. (1989). Software verification and validation: an overview. 6(3):10–17.

Fleurey, F., Breton, E., Baudry, B., Nicolas, A., and Jézéquel, J.-M. (2007). Model-driven engineering for software migration in a large industrial context. In *International Conference on Model Driven Engineering Languages and Systems*, pages 482–497. Springer.

G. Fischer, J. Lusiardi, J. v. G. (2007). Abstract syntax trees – and their role in model driven software development. pages 1–6.

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

Korshunova, E., Petkovic, M., g. j. V. Den Brand, M., and Mousavi, M. R. (2006). Cpp2xmi: Reverse engineering of uml class, sequence, and activity diagrams from c++ source code. In *2006 13th Working Conference on Reverse Engineering*, pages 297–298.

Méndez-Acuña, D., Galindo, J. A., Combemale, B., Blouin, A., and Baudry, B. (2017). Reverse engineering language product lines from existing dsl variants. *Journal of Systems and Software*, 133:145 – 158.

Parr, T. (2013). *The Definitive Antlr 4 Reference*. Pragmatic Bookshelf; Edição: 2 (25 de janeiro de 2013).

Sanchez, C. Cpp14 grammar. https://github.com/antlr/grammars-v4/tree/master/cpp. Acessado em: 03/07/2018.

Sistemas de Reputação: Uma Análise em Plataformas de Crowdtesting

Rafael Kogler, Fernando Costela, Alexandre Lazaretti Zanatta

Ciência da Computação – Universidade de Passo Fundo (UPF)
Bairro São José – Passo Fundo/RS

rafakogler23@gmail.com, nandocostella@gmail.com, zanatta@upf.br

***Abstract**: Reputation systems play an important role in the crowdtesting platforms. We analized five crowdtesting plataform focus on reputation systems. Differents mechanisms of reputation systems was presented. Crowdtesting platforms strongly use competitive reputation as an incentive for testers to perform their tasks. The negative reputation decreases the probability of crowdtester receiving new projects. We presented some suggestions to crowdtester improve their reputation.*

***Resumo**. Os sistemas de reputação possuem um papel relevante nas plataformas de crowdtesting. Realizou-se um estudo sobre os sistema de reputação das principais plataformas de crowdtesting e apresentou-se um comparativo dos diferentes mecanismos de sistemas de reputação. As plataformas de crowdtesting utilizam fortemente a reputação competitiva como forma incentivo para que os testadores realizem suas tarefas. Percebeu-se que um testador com baixa reputação provavelmente terá dificuldades em submeter sua tarefa. Ao final, são apresentadas algumas sugestões para que os testadores possam melhorar sua reputação.*

1. Introdução

A inteligência coletiva é um tema interdisciplinar e tem sido explorada pelas mais diversas áreas do conhecimento, entre elas, no processo de desenvolvimento de software. O conhecimento ou inteligência coletiva, é uma prática comum de troca de conhecimento por novas formas de organização e de coordenação flexível e em tempo real (Castells 2002). A partir desse conhecimento coletivo "as estruturas de hierarquização nas empresas estão sendo remodeladas", favorecendo a participação de atores externos em processos de criação em massa, alterando, inclusive, as relações de trabalho entre empregador e empregados (Tapscott e Williams 2007), (Deloitte 2017).

Crowdsourcing surge nesse contexto. Conforme Howe (Howe 2006), crowdsourcing "é o ato de pegar um trabalho tradicionalmente designado à um empregado e externá-lo para um grupo indefinido, e geralmente grande, de pessoas através da internet". Por meio de indivíduos independentes, distribuídos geograficamente, com conhecimentos e habilidades específicas, a criatividade advinda de uma grande quantidade de pessoas interagindo para executar diversas atividades e resolver problemas, cria oportunidades, mas também potencializa as dificuldades.

Exemplos de aplicação do crowdsourcing são abordados por (Cooper, et al. 2010), (Brabham 2008) e Xiao e Paik (Xiao e Paik 2014), incluindo a área de Engenharia de Software, e, por conseguinte, o desenvolvimento de software (Doan, Ramakrishnan e Halevy 2011), (Hosseini, et al. 2014), (LaToza, et al. 2013), (Mao, Yang, et al. 2013),

(Wu, Li e Tsa 2013). Pode-se dizer que crowdsourcing é um caso extremo de desenvolvimento distribuído de software. Crowdsourcing para desenvolvimento de software, ou Software Crowdsourcing (SWCS), tem se tornado uma área emergente e ainda pouco explorada na engenharia de software, inclusive nos testes de software (Wu, Li e Tsa 2013).

Crowdtesting é o termo utilizado para o desenvolvimento de testes de software pelo modelo crowdsourcing. O testador ao ingressar em plataformas de crowdtesting independente de seu conhecimento, participa de um sistema classificatório, baseado em reputação, pautado na qualidade e no desenvolvimento da realização de atividades propostas. O crowdtesting utiliza o conceito de crowdsourcing, pois os usuários realizam testes remotamente em seu ambiente. Os desenvolvedores de software ao contratar esse tipo de serviço, permitem que seus produtos sejam testados em diversas plataformas, dispositivos, versões, idiomas, pois os testadores utilizam seus dispositivos pessoais para realizar as tarefas.

Em comparação com o teste de software tradicional, o crowdtesting, possui a vantagem de recrutar, não apenas os testadores profissionais, mas também os usuários finais para participar dos ciclos de testes (Mao, Licia, et al. 2016). Os testes submetidos e concluídos passam pela análise de um avaliador. Em caso de aprovação, o testador adquire uma pontuação elevando os níveis de reputação, ampliando, com isso, suas chances em receber convites para novos projetos. Em caso de rejeição, há uma influência negativa na reputação do testador, fato que pode gerar uma desmotivação, por obter uma desvantagem na ausência de convites para novos projetos frente a outros testadores, ocasionando, por vezes, em sua desistência.

Uma questão enfrentada pela maioria das comunidades virtuais é que os participantes "têm uma alta tendência a interromper sua participação depois de algum tempo" (Lu, Phang e Yu 2011). O modelo possui desafios, entre eles: a baixa adesão de participantes; a ausência de incentivo remunerado; a pouca qualidade nos resultados das tarefas conclusivas; e, ainda, a exposição de dados confidenciais que são disponibilizados à multidão (Yang, et al. 2016). Assim, este trabalho tem como objetivo, realizar um estudo comparativo do funcionamento dos mecanismos de sistemas de reputação das principais plataformas de crowdtesting para a elaboração de recomendações para que futuros testadores possam aumentar sua reputação, ampliando, com isso, suas possiblidades de participação.

2. Metodologia

O trabalho apresentado caracteriza-se por uma pesquisa exploratória com abordagem qualitativa, buscando entender e interpretar o funcionamento, métodos, regras e mecanismos de reputações que determinam ou influenciam no perfil do indivíduo que está participando das plataformas de crowdtesting. Torna-se qualitativa pelo fato de buscar um melhor entendimento de compreensão sobre os assuntos citados através de diálogos com outros testadores que participam das plataformas abordadas neste estudo.

Para esse estudo, foram selecionadas apenas as plataformas de crowdtesting uTest, Testbirds, Passbrains, Bugfinders e 99Tests sugeridas por Mao *et al* (Mao, Licia, et al. 2016) do tipo de chamada "on-demand matching". O desenvolvimento deste trabalho foi organizado em quatro etapas. Inicialmente fez uma revisão da literatura sobre sistemas de reputação e como estes sistemas apoiam a participação de usuários em comunidades virtuais. Após, para a coleta de dados foram realizadas tarefas disponibilizadas pelas plataformas de crowdtesting, para entender o seu funcionamento,

regras, diretrizes e características. Efetuou-se o detalhamento do sistema de reputação das plataformas e fez-se a uma análise comparativa, de natureza narrativa, destas plataformas utilizando observações coleta de documentos e artefatos (que também representam narrativas organizacionais). Por fim, foram sugeridas recomendações de como os testadores podem utilizar os sistemas de reputação para participarem ativamente em plataformas de crowdtesting. A questão de pesquisa é: *"Quais são e como funcionam os mecanismos de reputação no crowdtesting?.*

3. Reputação no Crowdtesting

Reputação é "uma medida do nível desejável é uma entidade, conforme esse valor seja estabelecido por uma pessoa ou grupo de pessoas externas à entidade em questão" (Allahbakhsh, et al. 2012). A reputação em plataformas de crowdtesting é um importante mecanismo na medida que são submetidos os defeitos encontrados, o avaliador responsável pelo projeto, avalia os testes submetidos e enviados pelos testadores, verificando se está de acordo com os requisitos propostos, se possui duplicações, e por fim, se é considerado um defeito. Caso o ciclo de teste seja aprovado, o testador além de receber as bonificações propostas pelo ciclo de teste adquire uma pontuação que influencia em sua reputação e classificação, obtendo, com isso, pontos de reconhecimento pela qualidade do seu serviço na plataforma. Caso contrário, impactará negativamente, ocasionando uma provável ausência de convites de projetos e privilégios diante de outros testadores.

A seguir são apresentados os mecanismos de reputação das plataformas de crowdtesting uTest, Testbirds, Passbrains, Bugfinders e 99Tests.

3.1.1 Reputação na plataforma Applause / uTest

A plataforma uTest, disponibiliza dois sistemas de reputação entre os testadores. O "uPoints" e o "Badges". O testador obtém pontos por meio de contribuições; compartilhamentos e publicações de conteúdos envolvendo teste de software; leituras; escritas; participações em tarefas e, além disso, adiciona e segue outros testadores. Entretanto, este modelo não possui remuneração e não faz parte de um teste real, sendo importante o testador participar dessas tarefas disponibilizadas para melhorar a credibilidade em termos de contribuições na plataforma. Esse sistema de reputação, é conhecido como Pontos do uTest 'uPoints'. A reputação do testador contribuinte é indicada em um painel, sendo possível visualizar a quantidade de pontos adquiridos; a classificação diante dos membros cadastrados; a quantidade de amigos/seguidores ao finalizar as tarefas com sucesso.

Outra forma de reputação é conhecida na plataforma como 'Badges', onde o testador possui a oportunidade de participar de projetos remunerados e competir com outros testadores, em busca de encontrar e submeter o maior número de defeitos encontrados, de acordo com os requisitos propostos na tarefa. A cada teste disponibilizado, é informado um requisito que deve ser realizado, como por exemplo, os tipos e técnicas de testes, bem como, os dispositivos, versões, navegadores, dentre outros detalhes técnicos, que devem ser respeitados ao submeter um defeito encontrado, assim o avaliador responsável pelo projeto, analisa se o que foi enviado está de acordo com todos esses quesitos. Em caso de avaliação positiva é atribuído pontos de reputação ao testador, que ao atingir determinada pontuação adquire os 'Badges' relacionados aos testes aprovados realizados na plataforma, impactando positivamente na credibilidade e qualidade no envio da tarefa.

A figura 1 mostra o fluxograma da plataforma com destaque (indicado pelas letras "A" e "B") as atividades da reputação.

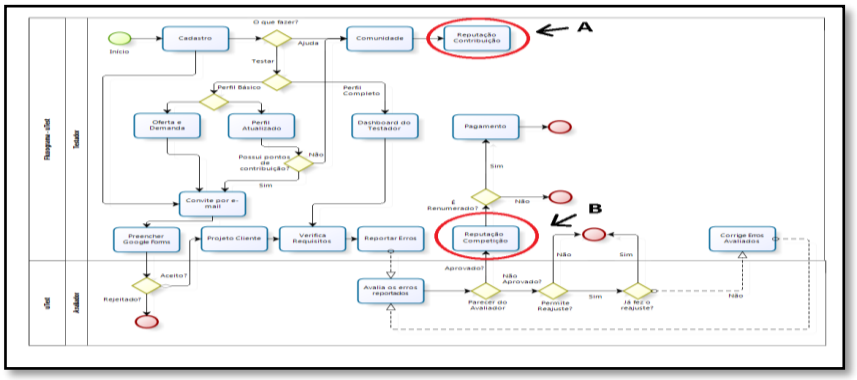


Figura 1. Fluxograma da Plataforma uTest

3.1.2 Reputação na plataforma TestBirds

A plataforma Testbirds disponibiliza dois sistemas de reputação entre os testadores: Pontos de Comunidade 'CPs' e os Pontos de Experiência 'EPs'. Participar ativamente da comunidade TestBirds é uma forma de aumentar a reputação na plataforma conhecida como Pontos de Comunidade CPs, ou seja, o testador deve, por exemplo, convidar membros novos e externos a plataforma para se cadastrarem e realizarem testes. Se este novo membro tiver testes aceitos, quem indicou "ganha" reputação. Outra forma é participar de mecanismos síncronos e assíncronos de trocas de mensagens entre os testadores. Quanto mais o testador for ativo na comunidade, mais recebe boa reputação. A plataforma possui um segundo modelo de reputação, conhecido com Pontos de Experiência 'EPs', no qual é necessário que o testador participe de testes remunerados; encontre defeitos e faça os relatórios; tenha o seu perfil atualizado com experiências pessoais e profissionais; e, por fim, registre seus dispositivos que serão colocados a disposição para a realização dos testes. A figura 2 mostra o fluxograma da plataforma com destaque (indicado pelas letras "A" e "B") as atividades da reputação.

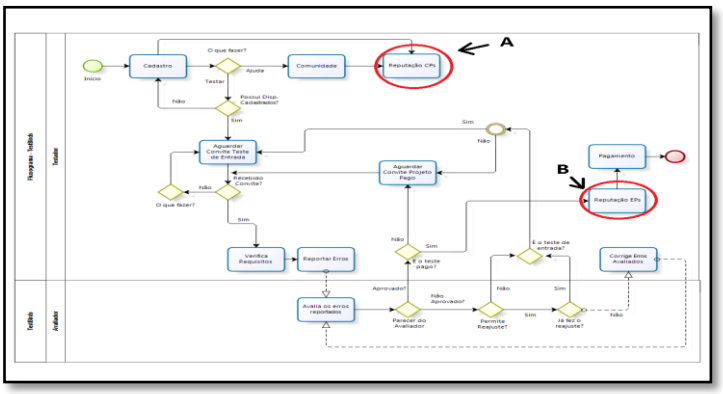


Figura 2. Fluxograma da Plataforma Testbirds

3.1.3 Reputação na plataforma BugFinders

A plataforma BugFinders utiliza o sistema de reputação conhecido como 'BugFinders Rank', que são adquiridos por meio da participação no ciclo de testes, encontrando e submetendo os defeitos de acordo com os requisitos especificados. O sistema de reputação possui seis nivelamentos e quanto mais alto o nível, mais o testador adquire

vantagens. Neste sistema, o nível de reputação não possui impacto ao testador quanto ao pagamento final, pois a plataforma, atualmente, está apenas coletando informações para que, posteriormente, seja analisada tal possibilidade remuneratória. A plataforma considera dois itens fundamentais para conquistar uma reputação positiva no perfil do testador: a quantidade de erros que foram submetidos e a taxa de aprovação. A figura 3 mostra o fluxograma da plataforma com destaque (indicado pela letra "A") a atividade da reputação.

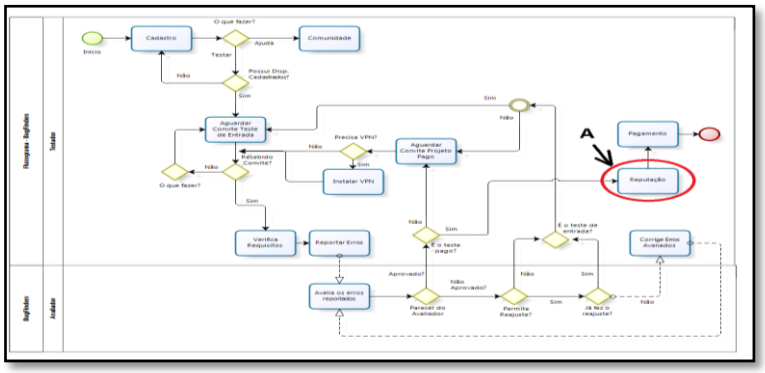


Figura 3. Fluxograma da Plataforma BugFinders

3.1.4 Reputação na plataforma Passbrains

A plataforma Passbrains utiliza o modelo de reputação conhecido como Pontos de Crédito de Carreira 'CCP', tais pontos virtuais são adquiridos a partir de tarefas concluídas e aprovadas e, à medida que são conquistados podem elevar as chances do testador em receber convites para novos projetos remunerados. A Passbrains oferece um fórum, o qual disponibiliza algumas informações de como o testador pode adquirir as CCPs para elevar a sua reputação. Caso seja realizado uma tarefa é possível visualizar a quantidade de CCPs que será adquirida. Para participar de projetos remunerados, primeiramente, é necessário realizar o teste de entrada. Ao submeter os defeitos encontrados, é primordial que o testador seja aprovado nos testes realizados, adquirindo um somatório de pontos para elevar a reputação e os CCPs. À medida que o testador conquiste trinta CCPs, a plataforma inicia o envio de convites de acordo com os dispositivos cadastrados. Ao adquirir os CCPs, a plataforma oferece algumas qualificações de carreira possuindo "maiores" chances de recompensas financeiras, bem como, convites de projetos futuros. Ao realizar atividades incompletas, submeter defeitos que não estejam de acordo com os requisitos ou defeitos duplicados, o testador sofrerá impactos negativos em sua reputação, sendo descontados os CCPs adquiridos durante a sua permanência na plataforma. A figura 4 mostra o fluxograma da plataforma com destaque (indicado pela letra "A") a atividade da reputação.

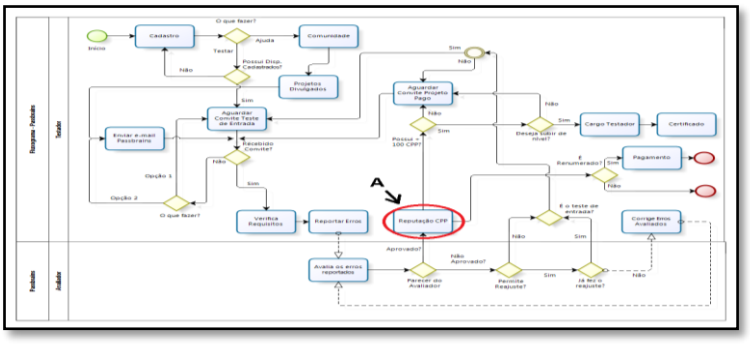


Figura 4. Fluxograma da Plataforma Passbrains

3.1.5 Reputação na plataforma 99tests

A plataforma 99tests utiliza um sistema de reputação onde se faz necessário que o testador participe de ciclo de testes disponibilizados na plataforma, submetendo defeitos que foram encontrados conforme os requisitos propostos. Se o avaliador responsável pelo projeto aprovar o defeito submetido, o testador adquire uma pontuação. Ao finalizar o ciclo de teste, o pagamento é dividido proporcionalmente, conforme a reputação de cada testador.

A 99tests disponibiliza um painel informativo de cada testador em seu perfil, sendo possível visualizar a quantidade de erros e de ciclos submetidos e aprovados; a quantidade de reputação adquirida, com base em todos os testes participados e o nível de experiência de cada testador. A plataforma disponibiliza este painel para, de certar forma, incentivar o testador a melhorar as suas habilidades com teste. A figura 5 mostra o fluxograma da plataforma com destaque (indicado pela letra "A") a atividade da reputação.

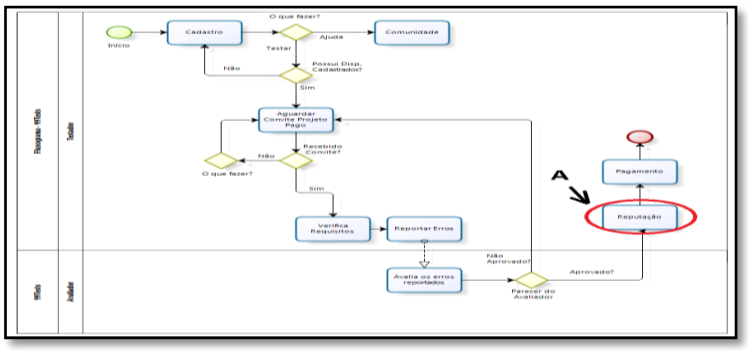


Figura 5. Fluxograma da Plataforma 99Tests

4. Discussão e Resultados

Os sistemas de reputação são ferramentas que possibilitam aos usuários ter uma classificação diante de algum serviço on-line, a fim de transmitir confiança aos indivíduos que requisitam um serviço e necessitam negociá-lo, tornando-se uma medida aproximada do quanto o usuário possui credibilidade e conhecimento diante da plataforma em que está participando. Para tanto, a reputação possui um papel descritivo, fornecendo informações sobre o perfil e a qualidade de determinado usuário. É comum o uso de mecanismos de reputação nas plataformas de crowdtesting, como pode ser observado na Tabela 1.

Algumas plataformas como a uTest e o Testbirds disponibilizam ambientes de estudo, os quais incentivam o testador a compartilhar conteúdos relacionados a testes de software e a participar de tarefas pré-definidas entre as plataformas, em troca da aquisição de uma pontuação na reputação por contribuição. Outro sistema de reputação que as plataformas utilizam é o competitivo, que possui como característica em comum, o registro de um dispositivo pessoal, pelo qual a plataforma envia convites de determinados projetos de acordo com os dispositivos cadastrados, facilitando a participação do testador, pois com isso, há a disponibilização de diversas ferramentas necessárias à realização dos testes. Outra característica é a aprovação do conteúdo submetido à participação de testes, pois através desta aprovação o testador pode obter mais pontuações para participar do sistema classificatório de reputação com outros testadores ativos.

Tabela 1. Reputação das Plataformas Estudadas

Plataformas	Applause/uTest	Passbrains	TestBirds	99Tests	BugFinders
Reputação por Contribuição	**Pontos do uTest (uPoints)** Convidar amigos Obter Seguidores Publicar Artigos/Fóruns Comentar Quiz uTest 101 Outros	Não tem	**(CPs) - Pontos da Comunidade** Convidar amigos Postar no blog Participar do 1º teste Ser ativo/reconhecido Outros	Não tem	Não tem
Reputação Competitiva	**Emblemas (Badges)** Ouro - 10% adicional Prata - 5,0%adicional Bronze - 2,5% adicional Aprovado - sem bônus Avaliado - sem bônus - Aprovação em testes	**CPP – Pontos de Crédito de Carreira** 500 CPP – Gerente Teste 300 CPP – Eng. QA 200 CPP – Testador Sênior 100 CPP – Testador Junior - Aprovação em testes - Aceitar convites - Participar de pesquisas	**(EPs) – Pontos de Experiência** - Aprovação em testes - Contribuições - Completar o Perfil - Registar o 1º Dispositivo	**Reputação** - Aprovação em testes	**BugFinders Rank** Diamante Platina Ouro Prata Bronze Aço - Aprovação em testes

A seguir são apresentadas algumas recomendações para que os testadores possam melhorar a sua reputação. Vale destacar que estas recomendações emergiram a partir das observações das análises realizadas nas plataformas, ou seja, são apenas percepções dos autores no sentido de iniciar um processo de apoio aos testadores receberem novos convites para participarem de atividades de testes de software.

1. Compartilhar e publicar conteúdos relacionados a testes de software;
2. Cadastrar os dispositivos pessoais na plataforma é fundamental para receber convites de projetos de acordo com o dispositivo que o testador possui;
3. Participar ativamente do ambiente de estudo disponível em cada plataforma;
4. Obter aprovação em testes de entrada;
5. Submeter defeitos encontrados, de acordo com os requisitos propostos no projeto;
6. Evitar envio de defeitos já identificados, assim, duplicados;
7. Respeitar as regras de cada plataforma.

5. Conclusão

As plataformas utilizam mecanismos de classificação de reputação (reputação por contribuição e competitiva) para atribuição de pontuação aos testadores. Testador com baixa reputação provavelmente terá dificuldades em submeter sua tarefa.

Percebeu-se que todas as plataformas disponibilizam uma classificação de indivíduos baseadas em reputação, perfazendo um fator motivacional para incentivá-los na participação de atividades propostas na plataforma. Cada indivíduo, ao preencher seu cadastro em uma das plataformas, inicia com a reputação zerada e à medida que se envolve na participação de testes reais e contribui com seu conhecimento à comunidade, adquire a possibilidade de elevar sua credibilidade na plataforma, caso contrário, poderá impactar negativamente em seu perfil do indivíduo. A reputação é calculada de acordo com a qualidade e a quantidade de contribuições. A seleção do testador é baseada na reputação fornecida pela plataforma. Cada testador possui uma reputação determinada pelo número e qualidade dos defeitos encontrados em seus testes anteriores.

Uma limitação do trabalho é a análise em apenas cinco plataformas, por isso, não é possível generalizar os resultados. Como trabalhos futuros espera-se avaliar cada uma das recomendações por meio de um experimento.

Referências

Allahbakhsh, M., A. Ignjatovic, B. Benatallah, and E. Bertino. "Reputation management in crowdsourcing systems." *Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 8th International Conference on. IEEE.* 2012 . 664-671.

Brabham, D. C. "Crowdsourcing as a model for problem solving an introduction and cases." *Convergence: the international journal of research into new media technologies*, Fev 2008, 1 ed.: 75–90.

Castells, Manuel. *A sociedade em Rede – a era da informação: economia, sociedade e cultura.* São Paulo: Paz & Terra – Volume 1, 2002.

Cooper, S, F Khatib, A Treuille, J. Barbero, J. Lee, and M. Beenen. "Predicting protein structures with a multiplayer online game." *Nature*, Ago 2010: 756-760.

Deloitte. *Navigating the future of crowd.* Relatório Técnico, Deloitte Review, Delloite Press, 2017.

Doan, A, R Ramakrishnan, and A. Y. Halevy. "Crowdsourcing systems on the world-wide web." *Communications of the ACM* 54, no. 4 (Abr 2011): 86-96.

Hosseini, M., K.T. Phalp, J. Taylor, and R. Ali. "Towards crowdsourcing for requirements engineering." *International Working Conference on Requirements Engineering: Foundation for Software Quality.* 2014. 43-69.

Howe, J. "The rise of crowdsourcing." *Wired magazine* 14, no. 6 (Jan 2006): 1-4.

LaToza, T., W. Towne, A. van der Hoek, and J. Herbsleb. "Crowd development." *6th International Workshop on Cooperative and Human Aspects of Software Engineering.* 2013. 85-88.

Lu, X., W. Phang, and J. Yu. "Encouraging participation in virtual communities through usability and sociability development: An empirical investigation." *Special Interest Group on Management Information Systems*, Ago 2011, 3 ed.: 96-114.

Mao, K., C. Licia, M. Harman, and J. Yue. "A Survey of the Use of Crowdsourcing in Software Engineering." *Journal of Systems and Software*, Abr 2016: 57-84.

Mao, K., Y. Yang, M. Li, and M. Harman. "Pricing crowdsourcing-based software development tasks." *International Conference on Software Engineering.* 2013. 1205-1208.

Tapscott, D., and A. D. Williams. *Wikinomics: How Mass Collaboration Changes Everything.* Penguin, 2007.

Wu, W., W. Li, and W. T. Tsa. "An evaluation framework for software crowdsourcing." *Frontiers of Computer Science* 7, no. 5 (Out 2013): 694-709.

Xiao, L., and H.Y. Paik. "Supporting Complex Work in Crowdsourcing Platforms: A View from Service-Oriented Computing." *23rd Australian IEEE Software Engineering Conference.* 2014. 11-14.

Yang, Y., M. R. Karim, R. Saremi, and G. Ruhe. "Who Should Take This Task?: Dynamic Decision Support for Crowd Workers." *10th International Symposium on Empirical Software Engineering and Measurement.* 2016. 8.

Anais da 2ª Escola Regional de Engenharia de Software (ERES 2018)
em Dois Vizinhos, PR, Brasil, 22 a 24 de outubro de 2018

ProDOC: Uma Proposta de Processo de Desenvolvimento Orientado a Comportamento

Yury Alencar Lima[1], Juliana Mareco Medeiros[1], Luciano Marchezan[1], Elder Rodrigues[1], Maicon Bernardino[1]

[1]Universidade Federal do Pampa (UNIPAMPA)
Código Postal 97.546-550 – Alegrete – RS – Brasil

{yuryalencar19,julianamareco18,lucianomarchezan94,eldermr}@gmail.com
bernardino@acm.org

Abstract. *Due to the difficulties of understanding the requirements in the Software Engineering scenario, this work addresses a proposed development process, which aims to improve the understanding of the system requirements, impacting the implementation and creation of the test cases, according to behavior of the software. The Behavior Oriented Development Process (ProDOC) uses specifications for the elicitation and elaboration of use scenarios for each functionality, in order to increase understanding of the requirements and improve the validation of the final system. To evaluate ProDOC, a questionnaire was carried out with researchers and / or professionals of the area, thus showing the acceptance and relevance of the proposal.*

Resumo. *Devido as dificuldades de compreensão dos requisitos no cenário da Engenharia de Software, este trabalho aborda uma proposta de processo de desenvolvimento, o qual tem por objetivo melhorar o entendimento dos requisitos do sistema, impactando na implementação e na criação dos casos de testes, de acordo com o comportamento do software. O Processo de Desenvolvimento Orientado a Comportamento (ProDOC), utiliza especificações para a elicitação e elaboração de cenários de uso para cada funcionalidade. A fim de aumentar a compreensão dos requisitos e melhorar a validação do sistema final. Com o intuito de avaliar o processo ProDoc foi enviado um questionário para pesquisadores e/ou profissionais da área. Os resultados do questionário mostraram a aceitação e relevância da proposta.*

1. Introdução

Com o objetivo de melhorar a produção e satisfazer as necessidades dos clientes são desenvolvidos processos, focando na agilidade no desenvolvimento e qualidade do produto final [Beck et al. 2001]. Dentre estes processos é possível citar as metodologias ágeis, que possuem um grande crescimento decorrente do surgimento de várias *startups* [Souza et al. 2015]. Isto acontece, devido a estas empresas disporem de grande concorrência e uma alta demanda, que resulta na necessidade de realizarem entregas rápidas e funcionais para o cliente, garantindo uma alta qualidade e confiabilidade [Souza et al. 2015]. Assim, com o uso destas metodologias, a empresa consegue desenvolver o sistema de forma rápida, adaptando-se as necessidades encontradas durante o desenvolvimento, além de obter um *feedback* frequente do cliente. Entretanto, este processo

tende a diminuir o formalismo e a documentação em comparação aos demais processos tradicionais.

Existem várias metodologias ágeis, onde muitas vezes são instanciadas do *framework* SCRUM que, por sua vez, tem como objetivo a gestão e o planejamento de processos de software [Schwaber and Sutherland 2011]. Este *framework* ganhou bastante espaço entre as *startups* devido a constante comunicação com o cliente e as entregas rápidas e iterativas, que possibilitam um constante *feedback* das partes interessadas, além da facilidade de adaptação dos requisitos do sistema em desenvolvimento. A utilização deste *framework* também não possui limitação relacionada ao uso de novas práticas de um projeto para outro [Schwaber and Sutherland 2011]. Sabendo disto, foram desenvolvidas tecnologias de apoio ao desenvolvimento, as quais podem ser utilizadas de acordo com o projeto.

Uma destas técnicas que podem ser utilizadas em conjunto com a metodologia ágil é o *Behavior-Driven Development* (BDD) [North 2006]. Esta técnica tem a finalidade de aumentar o entendimento da equipe em relação ao comportamento do sistema que será desenvolvido. A mesma também possibilita práticas como o *Test-Driven Development* (TDD) [Astels 2003], em que os testes serão criados antes e durante a implementação do software, neste caso, focando em métodos e classes. No caso de BDD estes testes são implementados baseados em contextos de uso descritos pelo próprio cliente. Para relatar estes cenários comumente é utilizado o Gherkin, que é uma linguagem descritiva de funcionalidades, cenários e exemplos [Cucumber 2018b]. Esta linguagem também pode ser usada em conjunto com algum *framework* possibilitando a automatização dos testes de sistema.

Tendo em vista que a extração e compreensão dos requisitos do cliente é um ponto essencial no desenvolvimento de qualquer sistema este estudo propõe um novo processo de desenvolvimento, focando na elicitação de requisitos e implementação das reais necessidades do cliente. O presente trabalho foi estruturado conforme segue. A Seção 2 apresenta o referencial teórico utilizado para a criação do Processo de Desenvolvimento Orientado a Comportamento (ProDOC). A Seção 3 apresenta a proposta de processo e exemplificação do mesmo. A Seção 4 apresenta uma avaliação da proposta de processo. A Seção 5 apresenta as considerações finais e trabalhos futuros.

2. Fundamentação Teórica

2.1. Scrum

O Scrum é definido como um *framework* para o planejamento e gerência de projetos complexos [Schwaber and Beedle 2002]. Pode ser considerado uma alternativa para a utilização da metodologia ágil no desenvolvimento de sistemas. Desta maneira, é possível resolver problemas e se adequar a novos requisitos, enquanto são realizadas entregas de partes do produto final até a conclusão do software. As responsabilidades do gerenciamento e planejamento do projeto são divididas em três papéis:

- ***Product Owner (PO)*:** Responsável por fornecer os requisitos do sistema, definir seus níveis de prioridade e ordem de implementação;
- ***Scrum Team:*** Responsável pelo desenvolvimento do sistema, ou seja, modelagem, codificação, testes e documentação;

- ***Scrum Master*:** Responsável pelo apoio técnico a equipe de desenvolvimento. Realiza a organização do grupo e participar efetivamente da proteção da equipe assegurando que a mesma não se comprometa demais. Ele certifica que cada pessoa envolvida no projeto está seguindo seu papel e seguindo as regras do Scrum.

O Scrum pode ser separado em três fases: Planejamento, *Sprints* e Encerramento. Na fase de planejamento tem-se a criação do *Product Backlog*, o qual é um conjunto de funcionalidades que deverão ser implementadas durante todo o processo de desenvolvimento e são definidos os papéis de cada membro da equipe. Após esta etapa é definido o *Sprint Backlog* que é composto de um subconjunto de funcionalidades inseridas no *Product Backlog*. Devido ao Scrum realizar entregas incrementais, o *Sprint Backlog* tem o objetivo de definir quais requisitos serão implementados naquele respectivo *Sprint*. Os *Sprints* são ciclos de desenvolvimento dentro do projeto que possuem duração máxima e fixa. O *Sprint Backlog* é criado pelo PO e o *Scrum Master* e é realizado a priorização das funcionalidades críticas para o sistema. Cada integrante do *Scrum Team* escolhe uma das atividades necessárias para a conclusão do *Sprint* e inicia-se todo o desenvolvimento.

Durante o decorrer do *Sprint* são realizadas reuniões diárias denominadas de *Scrum Daily Meeting*. Estas reuniões diárias tem por objetivo disseminar o conhecimento do que foi feito no dia anterior, identificar os impedimento encontrados para a concretização das atividades, além de enfatizar os próximos passos de cada integrante da equipe. Após o tempo determinado de um *Sprint* é gerado um incremento funcional do produto final baseado no *Sprint Backlog*. Ao final desta etapa, é realizado um *Sprint Retrospective*, onde é realizado uma análise do desenvolvimento no *Sprint*. O encerramento do Scrum acontece depois de um ou vários *Sprints* e é marcado pela conclusão do projeto, em que são realizados os testes de integração, testes de sistema, documentação do usuário e preparação do material de treinamento e marketing.

2.2. Gherkin

O Gherkin [Cucumber 2018b] é uma linguagem representativa que tem como finalidade a criação de especificações que podem ser executáveis ou não. Esta forma de caracterização possui um conjunto de palavras-chave e utiliza a indentação com o objetivo de realizar uma definição da sua estrutura e significado. As palavras-chave possuem suporte em diversos idiomas e possuem uma fácil compreensão tanto do cliente quanto do desenvolvedor, possibilitando sua utilização como artefato de documentação de requisitos, além da descoberta de vários cenários de uso de uma respectiva funcionalidade, um exemplo de uso é apresentado no trecho de código da especificação Gherkin na Figura 1.

```
Funcionalidade: Autenticar no site "X"
    Cenario: Realizar login no site "X".
        Dado o usuario estar com o site aberto
        Quando o mesmo inserir login
        E inserir a senha
        E clicar no botao "login"
        Entao o site redireciona a tela "home"
```

Figura 1. Exemplo de cenário de uso Gherkin

A primeira linha de um arquivo Gherkin é definida a linguagem da especificação seguindo é possível encontrar a palavra-chave primária "*Funcionalidade*" que representa

o nome de um requisito do sistema, que pode ser seguido ou não de uma breve descrição. No arquivo é descrito em alto nível um ou vários cenários de uso da respectiva funcionalidade que são ilustrados na linguagem por meio da palavra-chave "*Cenário*". A Figura 1 apresenta a palavra-chave "*Dado*" que é responsável por descrever alguma pré-condição para que seja executada a funcionalidade. Adicionalmente, outra denominação é o "*Quando*", o qual tem como função explicitar uma respectiva ação, caso necessite realizar mais do que uma é possível realizar o agrupamento utilizando o conectivo "*E*", o mesmo também pode ser usado em conjunto com outras palavras-chave. Por fim, um cenário é composto da última especificação que é o "*Então*", o qual representa os quais resultados das ações realizadas.

2.3. *Behavior Driven Development*

O *Behavior Driven Development* (BDD) [North 2006] é um conjunto de práticas que visam reduzir algumas atividades que acarretam o desperdício no desenvolvimento, como é o caso do retrabalho devido a requisitos mal compreendidos ou vagos, resistência em refatorar o código ou os ciclos de *feedback* lentos devido as transições. Além disso, é considerado uma técnica para desenvolvimento ágil que encoraja a colaboração entre os desenvolvedores, setores de qualidade e pessoas não-técnicas ou de negócios em um projeto de software. O objetivo do BDD é a integração das regras de negócio com a linguagem de programação, mantendo seu foco no comportamento do software com a finalidade de desenvolver e realizar a entrega do produto no melhor tempo possível com alta qualidade. Esta técnica pode ajudar a mitigar problemas como ambiguidade nas descrições do funcionamento do sistema. Isto é possível devido a descoberta deliberativa que acontece na utilização do BDD, que visa descrever o uso do software por meio de cenários ou exemplos de uso juntamente com os clientes [North 2006]. Assim, descobrindo novos requisitos e minimizando a possibilidade de por algum motivo estes serem ignorados pela equipe.

Com a finalidade de descrever o sistema e descobrir novas funcionalidades, são realizadas interações entre os principais interessados no projeto do sistema, como, por exemplo, os proprietários do produto, analistas de negócio, especialistas de domínio, programadores, testadores, entre outros. Isto é necessário devido os *stakeholders* possuírem papéis diferentes dentro do mesmo domínio além de possuírem visões distintas da mesma funcionalidade. Sabendo disto, a definição de exemplos concretos, muitas vezes, acarreta na descoberta de novos problemas e colabora com a equipe de desenvolvimento para a compreensão do domínio. Quando esses exemplos são criados colaborativamente são denominados de Mapeamento de Exemplo e podem se tornar testes de aceitação automatizados ou até mesmo documentação [Cucumber 2018a].

3. Processo de Desenvolvimento Orientado a Comportamento

O Processo de Desenvolvimento Orientado a Comportamento é baseado no *framework* Scrum e utiliza os mesmos papéis envolvidos, sendo eles: *Scrum Master*, *Product Owner* (PO) e o *Scrum Team*. Entretanto, aumenta o foco no quesito comunicação com o PO, compreensão e elicitação dos requisitos do produto, isso acontece devido ao uso de práticas do BDD. Além de melhorar na documentação do sistema, o mesmo mantém as três fases do Scrum: Planejamento, *Sprints* e Encerramento como pode ser visualizado na Figura 2.

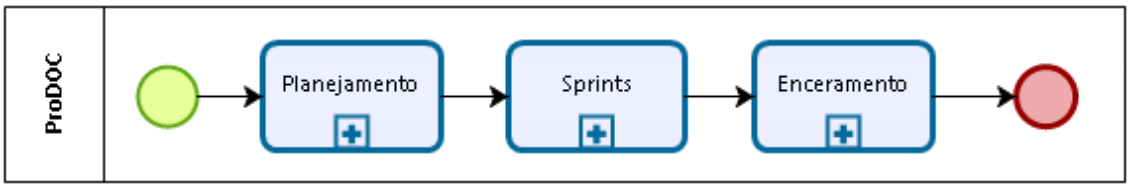


Figura 2. Modelo alto nível do ProDOC

A etapa de Planejamento que pode ser visualizada na Figura 3, é adaptada no ProDOC é onde acontece a reunião entre os interessados e os envolvidos no desenvolvimento. Nesta reunião, o PO tem como um dos seus objetivos elicitar requisitos para a equipe (Figura 3), e o mesmo é responsável pela estruturação e criação de arquivos de funcionalidades, e formatar requisitos usando a sintaxe Gherkin. As perguntas do PO terão como intuito extrair cenários de uso para cada funcionalidade descrita. Após isto, é recomendado priorizar os requisitos, com a finalidade de descobrir quais funcionalidades são mais importantes. Ao final desta etapa, é gerado o *Product Backlog* ordenado e estruturado em arquivos no formato Gherkin.

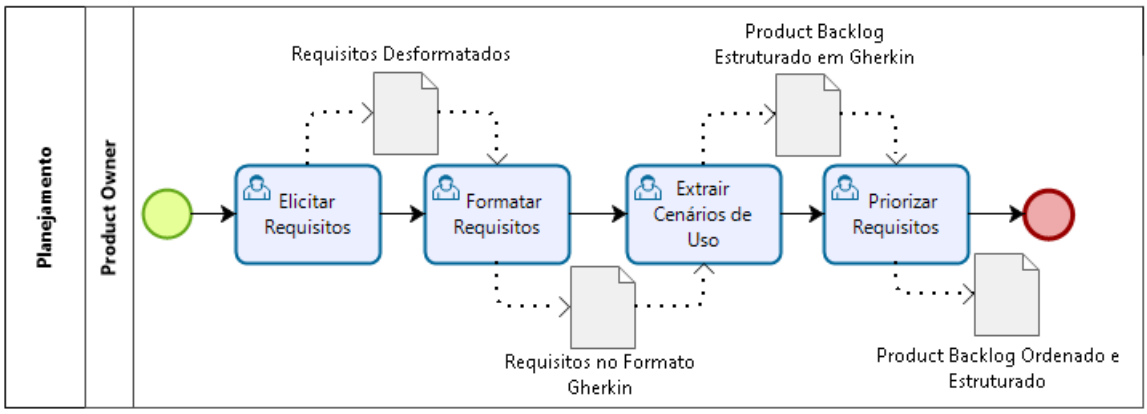


Figura 3. Fluxo do subprocesso Planejamento utilizando ProDOC

Concluída a etapa de planejamento, é dado início aos *Sprints* (Figura 4), tendo como primeira atividade a escolha dos requisitos que serão desenvolvidos, para isso, deve ser considerada a priorização e dependência entre as funcionalidades. O *Scrum Master* supervisiona a equipe em relação ao comprometimento com as funcionalidades. Como no Scrum, a escolha das atividades é por responsabilidade do integrante do *Scrum Team*. A atividade de implementação é orientada aos cenários descritos nos arquivos Gherkin, focando no comportamento da funcionalidade. Com o objetivo de verificar o andamento do projeto são realizadas diariamente as *Scrum Daily Meeting*, que são as reuniões em que cada membro da equipe descreve as atividades exercidas no dia anterior, dificuldades, impedimentos e atividades que serão exercidas.

Em virtude da especificação Gherkin conter o comportamento que o sistema deve exercer é possível realizar a implementação dos testes em paralelo com a do sistema. Estes testes são utilizados para verificar a aceitação da funcionalidade, tendo em vista que

representam o padrão comportamental especificado pelo PO como pode ser visualizado na Figura 4. Seguindo o processo, são realizadas as integrações necessárias e a disponibilização das funcionalidades implementadas. Considerando que os *Sprints* são iterativos e incrementais, após cada integração são executados os testes de regressão, com o propósito de verificar as funcionalidades já implementadas no sistema. Os *Sprints* são realizados até que todas as funcionalidades estejam implementadas com o comportamento que foram descritas nas especificações Gherkin. Durante qualquer fase do *Sprint* podem ser encontrados alguns erros e devem ser realizadas suas refatorações.

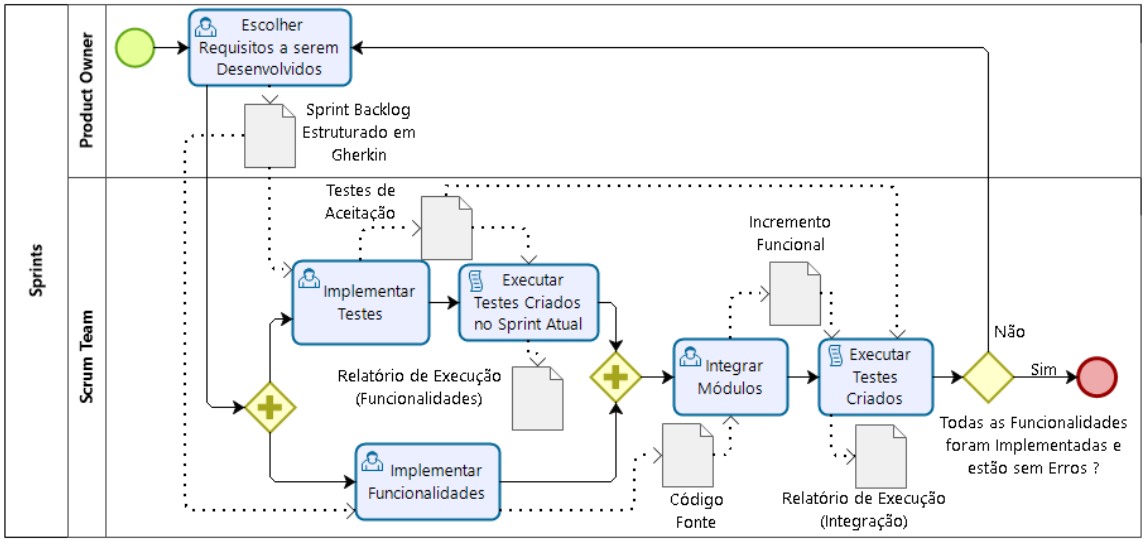


Figura 4. Fluxo do subprocesso *Sprints* utilizando ProDOC

Após a execução de todos os *Sprints* necessários, o ProDOC possui a fase de Encerramento, em que toda a documentação de usuário é entregue no formato Gherkin, inserindo apenas informações consideradas relevantes ao sistema. Como garantia, antes da entrega são executados novamente a suíte de testes relacionadas aos cenários cruciais do sistema. Nesta etapa é preparado o material de marketing e treinamento que pode também ser baseado na documentação Gherkin. Este subprocesso é representado na Figura 5. Ao final destas atividades, o produto final é entregue para o cliente. Com o intuito de analisar o débito técnico da equipe e projetar melhorias são analisados os relatórios de teste gerados a cada *Sprint*.

4. Avaliando a Proposta

Para realizar uma avaliação preliminar do ProDOC foi aplicado um questionário[1] em dez pesquisadores e/ou profissionais que já trabalharam com metodologias ágeis, principalmente Scrum. Além de coletar dados de caracterização de perfil dos participantes, o questionário coletou informações referentes ao nível de conhecimento relacionados aos tópicos (metodologias ágeis e desenvolvimento dirigido a comportamento) associados ao ProDOC. Resultando que dentre os participantes envolvidos 40% trabalharam ou trabalham na indústria e/ou graduação com Scrum por mais de três anos, 50% possuem até três

[1]Questionário disponível em: `https://tinyurl.com/prodoceres`

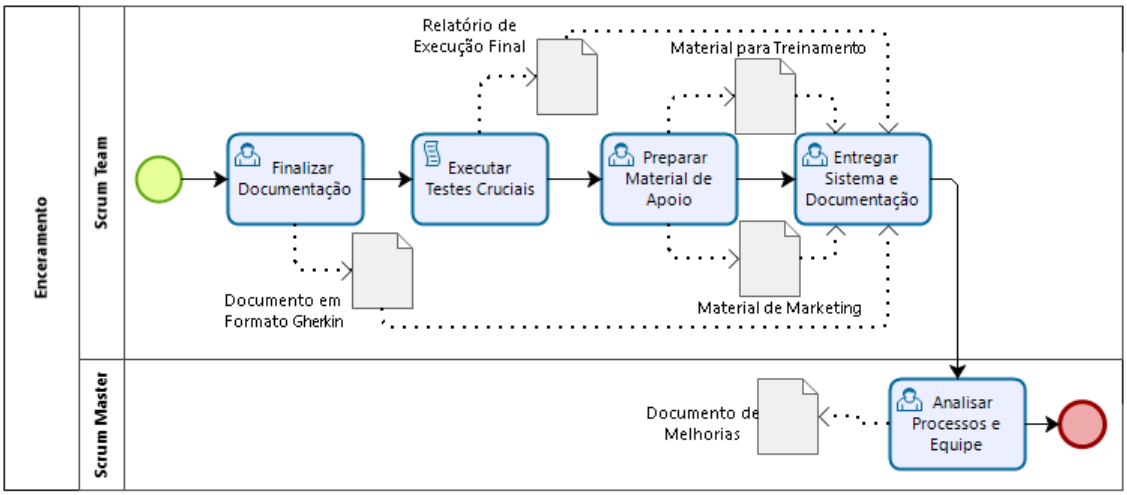
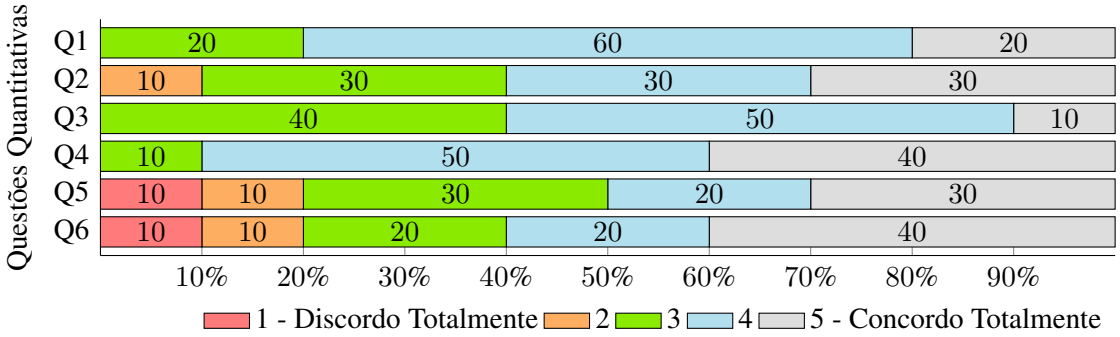


Figura 5. Fluxo do subprocesso Encerramento utilizando ProDOC

anos de experiência com metodologias ágeis e Scrum na graduação e/ou indústria e 10% não possuíam nenhuma experiência.

O questionário era composto por uma seção de questões (**Q**) quantitativas que tinham como objetivo medir o impacto das vantagens de uso do Gherkin em relação a: levantamento de requisitos (**Q1**), entendimento relacionado ao sistema (**Q2**), entendimento do comportamento e maior facilidade no desenvolvimento (**Q3**), melhoria da documentação do sistema (**Q4**), possibilidade de implementar testes em paralelo (**Q5**), e auxiliar na criação de tutoriais e treinamentos para o usuário final (**Q6**). As respostas referentes ás questões quantitativas podem ser visualizadas através da Figura 6.



PS: Algumas porcentagens geram um decimal recorrente que não resulta em um total de 100%.

Figura 6. Diagrama de frequência das questões quantitativas

Resultados da Análise: Ao realizar uma análise dos resultados das questões quantitativas, é possível demonstrar que o processo está na direção correta, devido a maioria dos participantes concordarem com os princípios do processo. Pode-se notar que ainda há alguns problemas relacionados à implementação de testes em paralelo e criação de tutoriais para auxílio devido as respostas das questões **Q5** e **Q6**. Para estas problemáticas serão planejadas melhorias com o objetivo de suprir estas necessidades. Apesar desses problemas, obteve-se um alto grau de confiança relacionado ao impacto e relevância da proposta. A primeira afirmação, é interpretada a partir dos resultados das perguntas de

que o ProDOC teria impacto no desenvolvimento. Isso indica que a criação do processo é uma pesquisa significativa. Quanto à relevância da proposta, foram alcançados resultados satisfatórios, mostrando que o processo está sendo desenvolvido na direção correta. Dentre as últimas perguntas do questionário, foram inseridas quatro questões abertas, com o objetivo de proporcionar aos participantes a oportunidade de contribuir com comentários, sugestões e verificar se usariam o processo para o desenvolvimento. A maioria das sugestões estavam relacionadas a tornar o processo mais iterativo e incremental. Alguns participantes acham que a proposta tem potencial de proporcionar outras melhorias a fim de demonstrar maiores benefícios para o desenvolvimento. A maior parte dos participantes afirmaram que utilizariam o processo de desenvolvimento e outros que dependeria do contexto e do produto a ser implementado.

5. Considerações Finais

Com base nas respostas quantitativas da avaliação, o ProDOC se mostrou eficaz em diversas partes do desenvolvimento, mesmo que ainda possua melhorias a serem implementadas. O reaproveitamento de artefatos e entendimento do produto a ser desenvolvido são seus pontos fortes, isto acontece decorrente ao uso da sintaxe Gherkin para as especificações. Devido a documentação possuir um maior papel no desenvolvimento do sistema a mesma terá uma maior prioridade. Proporcionando, assim, uma melhor manutenção do sistema posteriormente. Além disso, o ProDOC permite a implementação dos testes em paralelo com o sistema.

Com o propósito de que o processo tenha êxito, é necessário uma equipe que consiga extrair os reais cenários de uso. Tendo em vista que é um ponto crucial para o sucesso do projeto. Como trabalho futuro serão realizadas melhorias no processo com base na avaliação apresentada e será realizada a aplicação do ProDOC em um cenário real, com o objetivo de medir o real impacto na equipe com o seu uso.

Referências

Astels, D. (2003). *Test driven development: A practical guide*. Prentice Hall Professional Technical Reference.

Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., et al. (2001). Manifesto for agile software development.

Cucumber (2018a). BDD Overview. Disponível em: `https://docs.cucumber.io/bdd/overview/`. Acesso em: 07.07.2018.

Cucumber (2018b). Gherkin Reference. Disponível em: `https://docs.cucumber.io/gherkin/reference/`, addendum. Acesso em: 07.07.2018.

North, D. (2006). Introducing behaviour driven development. *Better Software Magazine*.

Schwaber, K. and Beedle, M. (2002). *Agile software development with Scrum*, volume 1. Prentice Hall Upper Saddle River.

Schwaber, K. and Sutherland, J. (2011). The scrum guide. *Scrum Alliance*, 21.

Souza, G., Nunes, J., Oliveira, J., Araújo, N., Gorgônio, F., and Vale, K. (2015). Diretrizes para uma metodologia de desenvolvimento de software aplicada a startups de tecnologia da informação.

Anais da 2ª Escola Regional de Engenharia de Software (ERES 2018)
em Dois Vizinhos, PR, Brasil, 22 a 24 de outubro de 2018

Modelagem organizacional e processos de negócio em metodologias ágeis: uma revisão sistemática da literatura

Camila Tiemi Outa, Victor F. A. Santander

UNIOESTE – Universidade Estadual do Oeste do Paraná, Cascavel/PR – Brasil

camila-tiemi@hotmail.com, victor.santander@unioeste.br

***Abstract**: The proper understanding of the organization and business processes is an important aspect in software processes. In this sense, we need to understand how practices, techniques and tools of organizational and business processes modelling are used by agile methodologies in the software engineering. This is a relevant aspect considering the using and matter of agile methodologies in the last years. In this paper, to obtain works related to organizational modelling and agile methodologies we present a systematic review of the literature.*

***Resumo**: A necessidade de compreender o ambiente organizacional e processos de negócio é fundamental em qualquer processo de software. Neste sentido, é importante conhecer como práticas, técnicas e ferramentas de modelagem organizacional e processos de negócio são utilizadas ou percebidas em contextos de desenvolvimento ágil na engenharia de software. Este é um aspecto importante e pouco explorado que deve ser estudado considerando a importância de metodologias ágeis nos últimos anos. Para realizar este estudo apresenta-se no presente artigo uma Revisão Sistemática da Literatura.*

1. Introdução

O desenvolvimento de software pode ser levado a cabo utilizando diversas metodologias. Independente da abordagem utilizada deve existir a preocupação em compreender adequadamente as necessidades da organização em relação a uma potencial solução computacional. Essa compreensão passa inicialmente pelo entendimento dos processos de negócio e ambiente organizacional no qual o software deverá operar. Isto implica que engenheiros de software devem se preocupar em utilizar técnicas, práticas e ferramentas que possam auxiliar nesse entendimento. Umas das alternativas é utilizar técnicas específicas para esse fim tais como i*(lê-se istar) [Yu 1995], BPMN (*Business Process Modeling Notation*) [Muehlen e Recher 2008], Casos de Uso de Negócio [Molina et al. 2007], entre outras. Contudo, estas técnicas são mais utilizadas em metodologias tradicionais as quais possuem uma fase específica para esse fim, com metas e papéis bem definidos diferentemente de metodologias ágeis.

Metodologias ágeis têm como essência considerar mais importante a satisfação do cliente ao invés de processos mais rígidos ou documentos formais. Como consequência nem sempre é dada a importância adequada ao entendimento dos processos de negócio e ambiente organizacional. Esta atividade deve ser apoiada pelo engenheiro de requisitos e faz parte do processo de elicitação, análise e negociação, documentação e validação de requisitos funcionais, não-funcionais e organizacionais. Por outro lado, é inegável o crescimento do uso de metodologias ágeis nos dias atuais tanto por organizações privadas quanto públicas. Isto nos faz refletir sobre como essas metodologias estão lidando com o uso de abordagens para compreender e modelar os processos de negócio e ambiente organizacional no desenvolvimento de software. É

importante saber como essas metodologias tratam esses aspectos já que essa compreensão influencia diretamente na qualidade da solução apresentada bem como na elicitação e documentação de requisitos alinhados com as metas organizacionais.

Observando este contexto, neste trabalho apresenta-se uma revisão sistemática (RS) da literatura cujo objetivo principal é encontrar trabalhos que apresentem técnicas, práticas e/ou ferramentas para apoiar o processo de entendimento de processos de negócio e ambientes organizacionais no âmbito do uso de metodologias ágeis. Na seção 2 apresenta-se brevemente a fundamentação teórica do trabalho. Nas seções 3.1, 3.2 e 3.3 são apresentadas respectivamente, o planejamento, execução e análise dos resultados da revisão sistemática da literatura. Finalmente, na seção 4 são realizadas as considerações finais do trabalho.

2. Fundamentação Teórica

A preocupação de adaptar-se rapidamente a mudanças nos requisitos, escopo e tecnologia num contexto de priorização de satisfação do cliente é o foco no uso de métodos ágeis [Cockburn e Highsmth 2000]. Contudo, independente da metodologia utilizada, o engenheiro de requisitos não deve somente atender ao que o cliente expressa, mas também observar os processos de negócio e ambiente organizacional procurando por melhorias não percebidas ou detectadas pelo cliente. Para esse fim, o engenheiro de requisitos necessita elicitar e modelar esses processos e/ou ambientes.

A modelagem de processos de negócios é utilizada para mostrar as atividades de uma área de negócios ou da empresa como um todo, e a sua sequência de execução, permitindo assim o entendimento de seu funcionamento [Costa 2009] [Pizza 2012]. Entretanto, o uso de técnicas de modelagem de processos e ambientes ainda é vista com desconfiança por utilizadores de métodos ágeis. Isto faz com que esta atividade seja negligenciada em alguns contextos. Assim, a presente revisão sistemática procura obter uma visão dos trabalhos realizados unindo métodos ágeis às práticas associadas ao entendimento de processos de negócio e ambientes organizacionais.

3. Revisão Sistemática

Esta revisão sistemática segue as diretrizes propostas em [Kitchenham e Charters 2007], dividindo a revisão em três fases: (3.1) planejamento da RS; (3.2) execução da RS, e (3.3) análise dos estudos da RS. Cabe ressaltar que para apoiar esta revisão foi utilizada a ferramenta StArt (*State of the Art through Systematic Review*) [LAPES 2015].

3.1. Planejamento

Definição das questões de pesquisa: As questões de pesquisa devem ser definidas visando orientar a seleção de estudos. As questões de pesquisa definidas para nosso estudo são:

Q1. Quais são as práticas, técnicas e ferramentas de modelagem organizacional e processos de negócio utilizadas em metodologias ágeis?

Q2. Quais as vantagens e desvantagens do uso de modelos de processos de negócio/organizacionais adotados com metodologias ágeis?

Q3. Como modelos de processo de negócio/organizacionais estão sendo aplicados com metodologias ágeis?

Q4. O processo de software em metodologias ágeis torna-se mais eficaz incluindo modelos de processo de negócio/organizacionais?

Definição das *Strings* de Busca: Após a definição das questões de pesquisa o próximo passo envolve a definição das *strings* de busca. Assim como sugerido em [Kitchenham e Charters 2007], utilizamos a técnica PICO (*Population*, *Intervention*, *Comparison*, *Outcome*) para enquadrar as questões de pesquisa e facilitar o processo de definição de

strings. Essas *strings* são aplicadas nas bases de dados eletrônicas escolhidas. Por questões de espaço as mesmas não são apresentadas. Contudo, podem ser consultadas em [LES 2018].

Definição das fontes de Busca: Em [McColl e Granville 2014] ressalta-se que pesquisar apenas um banco de dados não é suficiente para uma revisão sistemática, pois nenhuma base de dados cobre todos os resultados possíveis de pesquisa. Assim, no presente trabalho foram consideradas as bases de dados eletrônicas conforme segue: ISI Web of Science, Scopus, Compendex (Engineering Village), Science Direct, Springer Link e ACM Digital Library. Em [Brereton et al. 2007] destaca-se que essas bases de dados estão entre as mais relevantes para uma revisão sistemática em engenharia de software.

Definição e Seleção de Estudos: Os critérios de inclusão e exclusão de estudos [Siddaway 2014] devem ser baseados nas questões sugeridas no começo do protocolo da revisão sistemática. A tabela 1 apresenta os critérios adotados para a presente RS.

Tabela 1 – Critérios de inclusão e exclusão de estudos

	Critérios de Inclusão
1	Estudos que relatem problemas, desafios, praticas, técnicas identificados em modelos de processo de negócio/organizacionais aplicados em metodologias ágeis
2	Estudos que descrevam contribuições de modelos organizacionais ou modelos de negócios em metodologias ágeis, questionários envolvendo questões de negócio/organizacionais no processo ágil, modelos adaptados a um negócio/organização.
3	Estudos publicados em revistas, eventos, livros na área de computação que estejam indexados nas bases de estudos.
4	Estudos publicados até dezembro de 2017.
5	Estudos escritos em inglês.

	Critérios de Exclusão
1	Estudos que não relatem problemas, desafios, sugestões, alterações identificadas na aplicação de metodologias ágeis.
2	Estudos que não descrevam contribuições de modelos organizacionais ou modelos de negócios em metodologias ágeis, questionários envolvendo questões de negócio/organizacionais no processo ágil, modelos adaptados a um negócio/organização.
3	Trabalhos de conclusão de curso, dissertações e/ou monografias de mestrado/doutorado.
4	Estudos publicados após dezembro de 2017.
5	Estudos não escritos em inglês.
6	Estudos duplicados (Apenas uma cópia de cada trabalho será considerada).

O procedimento de seleção de estudos é baseado nas etapas adaptadas de [Silva 2015] conforme segue. A **1ª Etapa** tem como objetivo importar todos os estudos que foram retirados das bases de dados eletrônicas na ferramenta StArt. Na **2ª Etapa** utiliza-se a ferramenta StArt para remover os estudos duplicados. Na **3ª Etapa** é feita a leitura e análise dos títulos, *keywords* (palavras-chave) e abstracts (resumos) dos estudos. Na **4ª Etapa** leva-se a cabo a leitura e análise completa de cada estudo não excluído. Os estudos incluídos para a análise na terceira fase da RS devem atender todos os critérios de inclusão.

3.2. Execução

A fase de execução da RS foi realizada conforme passos descritos a seguir.

Busca e Organização: Neste passo foi realizada a aplicação das *strings* de busca definidas no planejamento da RS. Este passo também trata da obtenção dos arquivos BibText [Silva 2015] que foram gerados nas bases de dados e importados na ferramenta StArt. No total foram obtidos 794 estudos. A tabela 2 resume a quantidade obtida por cada base de dados e a figura 1 mostra um gráfico exibindo a porcentagem de cada base.

Tabela 2 - Resultado por base de dados

Base de dados	Resultados
Web Of Science	10
Scopus	521
Science Direct	75
Compendex	25
Springer Link	163
Total	**794**

Tabela 3 - Resultado duplicados por base de dados

Base de dados	Total Resultados	Total Duplicados
Web Of Science	10	1
Scopus	521	26
Science Direct	75	1
Compendex	25	15
Springer Link	163	0
Total	**794**	**43**

Tabela 4 - Total de excluídos

Base de dados	Total Resultados	Total excluídos
Web Of Science	9	9
Scopus	495	492
Science Direct	74	73
Compendex	10	8
Springer Link	163	163
Total	**751**	**745**

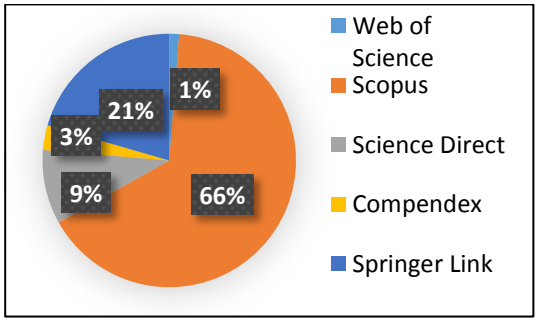


Figura 1 - Gráfico de pizza com porcentagens de cada base de dados

Processo de Seleção: Este passo foi realizado conforme as 4 etapas definidas no planejamento. No final foram identificados 43 estudos duplicados (tabela 3). A base de dados Web of Science e Science Direct apresentaram apenas um trabalho duplicado, a base de dados Scopus apresentou 26 trabalhos duplicados, a base de dados Compendex apresentou 15 artigos duplicados, e a base de dados Springer Link não apresentou nenhum trabalho duplicado.

Em seguida foram aplicados os outros critérios de exclusão aos artigos obtidos. Os artigos foram excluídos com base no título, palavras-chave e *abstract*. A tabela 4 mostra a quantidade de artigos excluídos de acordo com o total de resultados de cada base.

Tabela 5. Artigos Selecionados para a análise de resultados.

ID	TITULO	Tipo	Ano	Base de Dados
8706	Assessing the adoption level of scaled agile development: a maturity model for Scaled Agile Framework	*Journal*	2017	Scopus
8862	Climbing the stairway to heaven: Evolving from agile development to continuous deployment of software	*Book*	2014	Scopus
9050	A Hybrid Model for IT project with Scrum	*Conference*	2011	Scopus
15085	Method for Adaptation and Implementation of Agile Project Management Methodology	*Conference*	2016	Science Direct
23193	An agile strategy for implementing CMMI project management practices in software organizations	*Conference*	2015	Compendex
23200	Mapping agile practices to CMMI-DEV level 3 in web development environments	*Conference*	2014	Compendex

Após aplicar os critérios de exclusão, foram aplicados os critérios de inclusão aos artigos selecionados. Foi realizada a leitura e análise dos *abstracts* e leitura na íntegra dos artigos. Após a análise de cada artigo, foram selecionados 6 estudos para a interpretação final os quais atenderam todos os critérios de inclusão. A tabela 5 apresenta um resumo dos artigos selecionados.

3.3. Análise de Resultados

Nesta seção é realizada a análise de resultados de acordo com as questões de pesquisa definidas na seção 3.1.

Q1: Quais são as práticas, técnicas, ferramentas de modelagem organizacional e processos de negócio utilizadas em metodologias ágeis?

Dois artigos mencionaram o uso de práticas que podem ser consideradas no contexto do entendimento da organização e seus processos. Mais especificamente no artigo ID 15085 [Rasnacisa e Berzisaa 2016] propõe-se um método para apoiar o processo de adaptação e implementação do gerenciamento de projetos utilizando metodologias ágeis. Defende-se no artigo que metodologias ágeis precisam ser adaptadas dependendo do tipo de projeto, organização e pessoas envolvidas. Nesse contexto, utiliza a metodologia SCRUM como base do método proposto acrescentando a esta metodologia o artefato *"Business Requirements*" e o papel "*Business Owner*". Estas mudanças segundo o artigo deixam mais claros aspectos do negócio a serem suportados por um sistema computacional. Contudo, no artigo não há exemplos desse artefato nem menção à ferramentas de apoio utilizadas. Outro artigo que faz referência à necessidade de compreender de forma adequada os processos de negócio da organização e associação de soluções computacionais que melhorem esses processos é o artigo ID 8706 [Turetken et al. 2016]. Este artigo propõe o SAFe MM (*Scaled Agile Framework – Maturity Model*) o qual é um modelo de maturidade que introduz um roteiro para adoção dos métodos ágeis em empresas tradicionais. O modelo de maturidade proposto é construído visando apoiar o uso do método SAFe o qual orienta organizações no processo de adoção e adaptação de métodos ágeis em projetos maiores, de maior complexidade e maior quantidade de pessoas envolvidas. A este desafio os autores denominam de escalabilidade de métodos ágeis. No método SAFe, propõe-se um artefato denominado de "*Business Epic*" que são *containers* construídos na interação com *stakeholders* do negócio para definir elementos do sistema computacional que agreguem valor aos processos de negócio. Há um *template* que auxilia na construção deste artefato e que pode ser considerado como uma prática de modelagem organizacional e processos de negócio já que procura explicitar de que forma o negócio atual poderá ser beneficiado de uma solução computacional. Elementos do processo de negócio atual são descritos nesse artefato.

Já o artigo ID 23200 [Torrrecilla-Salinas et al. 2014] foca na satisfação do nível 3 do modelo de maturidade CMMI-DEV [CMMI 2014] através do mapeamento de práticas ágeis para atingir o que é solicitado nesse nível de maturidade. Esse modelo de maturidade é amplamente utilizado por organizações em nível mundial que desejam melhorar seus processos de desenvolvimento. Nos estágios do mapeamento proposto no artigo podemos observar dois objetivos específicos (*Specific Goals*) de áreas de processo desse nível do CMMI-DEV. O objetivo específico "*Establish Organizational Process Assets"* relacionado à área de processo "*Organizational Process Definition* (OPD)" e os objetivos específicos "*Determine Process Improvement Opportunities"* e "*Plan and Implement Process Actions"* relacionados à área de processo "*Engineering Organizational Process Focus* (OPF)". Em relação ao primeiro objetivo o artigo relata que o método proposto em [Schawaber 2007] possui práticas e artefatos que permite satisfazê-lo. Nesse caso específico, o artefato proposto é o "*Enterprise Product Backlog"* [Schawaber 2007]. Para o segundo e terceiro objetivos o artigo defende que as práticas e propostas apresentadas nos trabalhos [Derby e Larsen 2006] e [Poppendieck e Poppendieck 2003] satisfazem essas metas. Contudo, não apresenta exemplos dessas práticas.

Finalmente o artigo ID 23193 [Soares e Meira 2015] não responde diretamente a questão da pesquisa mas a leitura do mesmo permite apontar algumas lacunas e comentários associados à questão da pesquisa. Mais especificamente esse artigo

referencia o modelo de maturidade *Capability Maturity Model Integration* (CMMI). O artigo propõe o uso de estratégias de metodologias ágeis em organizações que desejam alcançar os níveis de maturidade do CMMI. Para este fim, o artigo realiza uma revisão sistemática para encontrar trabalhos relacionados ao tema e posteriormente realiza um estudo de campo aplicando entrevistas em organizações que desejam aderir ao CMMI com práticas advindas do Scrum [Carvalho e Mello 2012], XP [Souza 2007] and Kanban [Ribeiro 1999]. Constata-se que é possível utilizar metodologias ágeis para alcançar níveis de maturidade do CMMI 2, CMMI 3 e CMMI 5. No entanto, relata-se no artigo que as práticas ágeis sozinhas não são suficientes para alcançar esses níveis de maturidade e práticas adicionais devem ser adotadas. Observando o CMMI e suas áreas de processos chave é possível constatar que haveria necessidade de encontrar estratégias para satisfazer as seguintes áreas de processo: Definição do Processo Organizacional e Foco no Processo Organizacional pertencentes ao nível 2 do CMMI, Desempenho do Processo Organizacional do nível 4 e Inovação e Implantação Organizacional do nível 5.

Q2: Quais as vantagens e desvantagens do uso de modelos de processos de negócio/organizacionais adotados com metodologias ágeis?

Nos artigos ID 15085 [Rasnacisa e Berzisaa 2016], 8706 [Turetken et al. 2016] e 23200 [Torrecilla-Salinas et al. 2014] defende-se que os artefatos propostos para melhorar o entendimento dos processos e ambientes organizacionais facilitam o trabalho de implantação de métodos ágeis ou satisfação de modelos de maturidade como CMMI e CMMI-DEV. O artigo de ID 23193 [Soares e Meira 2015] apesar de não propor práticas, técnicas ou ferramentas específicas para modelagem de processos de negócio descreve que o maior desafio na escalabilidade de métodos ágeis não está no uso de práticas ágeis existentes ou adoção de novas práticas e sim na interface e coerência entre essas práticas e os processos organizacionais existentes. Isto nos leva a crer que modelar esses processos pode ser uma forma de melhorar as chances de sucesso no uso de metodologias ágeis em um contexto de uso em projetos maiores, geograficamente distribuídos, mais complexos e com maior quantidade de pessoas envolvidas.

Q3: Como modelos de processo de negócio/organizacionais estão sendo aplicados com metodologias ágeis?

Nos artigos ID 8706 [Turetken et al. 2016] e 23200 [Torrecilla-Salinas et al. 2014] a proposta de artefatos e/ou práticas está associado à necessidade de satisfazer áreas de processos chaves bem como objetivos específicos dessas áreas nos níveis de maturidade desejados nos modelos CMMI e CMMI-DEV. Já no artigo de ID 15085 [Rasnacisa e Berzisaa 2016] a proposta de artefato que é agregado ao método SCRUM surge como uma necessidade se adaptar métodos ágeis à organizações cujos contextos são diferentes daqueles inicialmente considerados no surgimento dessas metodologias. Organizações com projetos maiores e com maior número de *stakeholders* envolvidos precisam adaptar as metodologias ágeis às suas realidades. No artigo, a ideia do artefato proposto é preencher essa necessidade verificada pelos autores.

Q4: O processo de software em metodologias ágeis seria mais eficaz incluindo modelos de processo de negócio/organizacionais?

O artigo de ID 23193 [Soares e Meira 2015] relata que para alcançar níveis de maturidade do CMMI é necessário que as metodologias ágeis sejam adotadas/adaptadas considerando práticas sugeridas pelo CMMI mas que não firam seus princípios. Mais especificamente, seriam necessárias estratégias para satisfazer as áreas de processos

chave e objetivos específicos exigidos nos níveis 2, 4 e 5 do CMMI conforme expresso na Q1. No artigo de ID 8706 [Turetken et al. 2016] o autor se refere à escassez em trabalhos que discutam os desafios e orientações sobre a escalabilidade de práticas ágeis nas organizações e em particular organizações que estão fazendo transições de uma abordagem tradicional. Nesse artigo destaca-se o artefato denominado "*Business Epic*" associado ao método SAFe (*Scaled Agile Framework*) o qual na opinião dos autores torna o processo de escalabilidade de métodos ágeis mais maduro. No artigo ID 8862 [Olsson e Bosch 2014] há uma referência ao modelo ESAO (*Ecosystem, Strategy, Architecture and Organizing*) [Bosch e Bosch 2014], o qual é utilizado para avaliar as dimensões de ponta a ponta dos negócios, tecnologia e organização com considerações especialmente tomadas para as partes interessadas e externas no ecossistema de negócios do qual uma empresa faz parte. Um dos maiores desafios apontados pelos autores do artigo está em alinhar estratégias internas de negócios com os ecossistemas nos quais organizações estão inseridas. Isto implica em que os processos de negócio internos devem ser muito bem entendidos para avaliar melhorias na transição de processos tradicionais para metodologias ágeis. Isto sugere que o uso de modelos de processos de negócio ou organizacionais pode ser uma parte fundamental para o sucesso dessas transições. Finalmente o artigo ID 9050 [Hayata e Han 2011] propõe um modelo hibrido para aplicar SCRUM em processos de software tradicionais e em gerenciamento de projetos de TI (tecnologia de Informação). Não há menção explicita à modelagem de processos de negócio e ambientes organizacionais, mas no método há a atividade de "Análise de Requisitos". Se considerarmos essa atividade no contexto da engenharia de requisitos tradicional, podemos considerar que a mesma abrange o entendimento dos processos de negócio e posterior uso para elicitar, validar e documentar requisitos. Isto poderia auxiliar no processo de adoção de metodologias ágeis junto às organizações acostumadas a processos tradicionais.

4. Considerações finais

O propósito desta revisão sistemática foi investigar trabalhos que nos permitissem encontrar estudos relacionados à modelagem organizacional e processos de negócio no contexto de metodologias ágeis. Foram obtidos ao final 6 artigos os quais foram analisados e utilizados para responder cada uma das questões de pesquisa. Os resultados mostram que não há muitos trabalhos relacionados ao tema e observa-se que é um campo passível de futuras pesquisas. Por exemplo, como trabalhos futuros poderíamos averiguar como técnicas tradicionais de modelagem organizacional e de processos de negócio tais como BPMN, casos de Uso de Negócio e i* entre outras, poderiam ser utilizadas em metodologias ágeis considerando principalmente a satisfação de modelos de maturidade como CMMI, CMMI-DEV ou MPS-Br [MPSBR 2011].

Referências

[Kichenham e Charters 2007] Kitchenham, B., Charters, S. (2007). Guidelines for performing Systematic Literature Reviews in Software Engineering.

[Mccoll e Glanville 2014] Mccoll, R., Glanville, J. (2014). What is a systematic review? Evidence Based Health care.

[Brereton et al. 2007] Brereton, P., Kitchenham, B. A., Budgen, D., Turner, M., Khalil, M. (2007). Lessons from applying the systematic literature review process within the software engineering domain. Journal of Systems and Software, v. 80, n. 4, p.571 – 583.

[Siddaway 2014] Siddaway, A. (2014).What is a systematic literature review and how do II do one? https://www.stir.ac.uk/media/schools/management/documents/centregradresearch/How%20to%20do%20a%20systematic%20literature%20review%20and%20meta-analysis.pdf.

[LAPES 2015] LAPES - Laboratório de pesquisa em Engenharia de Software. (2015). http://lapes.dc.ufscar.br/tools/start_tool.

[Silva 2015] Silva, A. (2015). Obtendo Casos de Uso a Partir de Modelos de Processos de Negócio: Uma Revisão Sistemática, http://www.inf.unioeste.br/~tcc/2015/TCC%20-%20Alexandre%20Silva.pdf.

[Cookburn e Highsmth 2000] Cockburn, A. and Highsmth, J. (2000). Agile software development: The business of innovation, https://ieeexplore.ieee.org/document/947100

[Schwaber 2007] Schwaber, K. (2007). The Enterprise and Scrum. Microsoft Press, Redmond.

[Derby e Larsen 2006] Derby, E., Larsen, D. (2006). Agile Retrospectives. Making Good Teams Great. The Pragmatic Bookshelf, Dallas.

[Poppendieck e Poppendieck 2013] Poppendieck, M., Poppendieck, T. (2003). Lean Software Development. An Agile Toolkit. Addison-Wesley, Boston.

[Souza 2007] Souza, L.M. (2007) Método Ágil XP (Extreme Programming), In: Revista Eletronica da FIA.

[Carvalho e Mello 2012] Carvalho, V.C., Mello, C.H.P. (2012). Aplicação do método ágil scrum no desenvolvimento de produtos de software em uma pequena empresa de base tecnológica. In: Revista Gestão e Produção.

[Pizza 2012] Pizza, W. (2012). A metodologia Business Process Management (BPM) e sua importância para as organizações. In: Faculdade de Tecnologia de São Paulo-FATEC SP.

[Muehlen e Recher 20008] Muehlen, M.Z., Recher, J. (2008). How Much Language Is Enough? Theoretical and Practical Use of the Business Process Modeling Notation. In: Seminal Contributions to Information Systems Engineering, P. 429-443; 25 Years of CAiSE – Springer.

[Costa 2009] Costa, L. (2009). Formulação de uma metodologia de modelagem de processos de negócio para implementação de Workflow. In: PPGEP-UTFPR.

[Ribeiro 1999] Ribeiro, P. D. (1999). Kanban – resultados de uma implantação bem-sucedida. 3. ed. Rio de Janeiro: COP Editora.

[Bosch e Bosch-Sijtsema 2014] Bosch, J., Bosch-Sijtsema, P. (2014) ESAO: A holistic ecosystem-driven analysis model. In: Proceedings of the 5th International Conference on Software Business (ICSOB), Cyprus.

[Yu 1995] Yu, E. S. (1995). Modelling Strategic Relationships for Process Reengineering. In: Ph.D. Thesis. Dept. of Computer Science, University of Toronto.

[Molina et al. 2007] Molina, J. G., Ortín, M. J., Moros, B., Nícolás, J., Toval, A. (2007). De los Procesos del Negocio a los Casos de Uso. In: Universidade de Murcia, Espanha.

[Turetken et al. 2016] Turetken, O., Stojanov, I., Trienekens, J. J. M. (2016). Assessing the adoption level of scaled agile development: a maturity model for Scaled Agile Framework. In: Journal of Software: Evolution and process.

[Olson e Bosch 2014] Olsson, H. H., Bosch, J. (2014). Climbing the "Stairway to Heaven": Evolving From Agile Development to Continuous Deployment of Software. In: Book: Continuous Software Engineering, p. 15-27.

[Hayata e Han 2011] Hayata, T., Han, J. (2011). A Hybrid Model for IT Project with Scrum. In: Proceedings of 2011 IEEE International Conference on Service Operations, Logistics and Informatics.

[Rasnacisa e Berzisaa 2016] Rasnacisa, A., Berzisaa, S. (2016). Method for Adaptation and Implementation of Agile Project Management Methodology. In: ICTE 2016 - Procedia Computer Science, Elsevier.

[Soares e Meira 2015] Soares, F. S. F., Meira, S. R. L. (2015). An agile strategy for implementing CMMI project management practices in software organizations. In: 10th Iberian Conference on Information Systems and Technologies (CISTI).

[Torrecilla-Salinas et al. 2014] Torrecilla-Salinas, C.J., Sedeño, J., Escalona, M.J., Mejias, M. (2014). Mapping Agile Practices to CMMI-DEV Level 3 in Web Development Environments. In: 23ª International Conference On Information Systems Development (ISD).

[CMMI 2014] CMMI Product Team: CMMI for Development, Version 1.3. (2014). Carnegie Mellon. In:University (2010), http://www.sei.cmu.edu/reports/10tr033.pdf.

[MPSBR 2011] MPSBR. (2011). Mps. br–melhoria de processo do software brasileiro.

[LES 2011] LES Laboratório de Engenharia de Software. (2018). www.inf.unioeste.br/~les.

Aplicação do Mapeamento de Fluxo de Valor para Identificação de Desperdícios Operacionais em uma Empresa de Software de Grande Porte

Felipe B. Ribeiro[1], Pedro Henrique de A. Machado[1], Jackson Gaspar Schimit, Luana Belusso[1], Rafael A. Paes de Oliveira[1]

[1]Coordenadoria do Curso de Bacharelado em Engenharia de Software - Universidade Tecnológica Federal do Paraná (UTFPR)
Estr. p/ Boa Esperança, S/n - Zona Rural - 85.660-000 - Dois Vizinhos - PR - Brasil

felipebrenaribeiro@gmail.com, pedrohmachado@utfpr.edu.br

jacksonschimit@gmail.com, luanabelusso@alunos.utfpr.edu.br

raoliveira@utfpr.edu.br

Abstract. *Through the application of value stream mapping in a real large-scale software maintenance environment, information will be picked from the requests that the clients open with this company, and the teams and directions will be analyzed to identify if there is any operational waste in the resolution of the requests. After the analysis the results will be showed putting the operational wastes in evidence.*

Resumo. *Através da aplicação do mapeamento de fluxo de valor em um ambiente real de manutenção de um software de grande porte, será colhido informações das solicitações que os clientes abrem com a empresa, e analisado as equipes e direcionamentos para identificar se há algum desperdício operacional na resolução das solicitações. Após a análise os resultados serão apresentados evidenciando os desperdícios operacionais encontrados.*

1. Introdução

O Processo de Desenvolvimento de Software é uma área da Tecnologia da Informação (TI) que ao longo dos anos foi e continua se desenvolvendo de forma benéfica para as empresas, fornecendo regras, padrões e diretrizes para facilitar o desenvolvimento do software e todos os aspectos a ele relacionado. Hoje as empresas de software, chamadas de *Software Houses* (SH), em sua maior parte atuam no mercado com um software robusto e pronto para atender as necessidades dos clientes, mas também estão abertos a receber solicitações de melhoria do mesmo, com o objetivo de aperfeiçoá-lo continuamente. Caso haja algum processo interno dos clientes que o sistema não consegue atender, é solicitado a empresa do sistema que faça uma melhoria ou uma customização para que se adapte e tenha essa funcionalidade, beneficiando a todos. Esse processo de customização/melhoria recebe o nome de manutenção de software.

Ao aplicar um processo de desenvolvimento em uma fábrica de software, é comum que dificuldades e falhas sejam encontradas até que formas sejam levantadas para corrigi-las. Com a sociedade se tornando cada dia mais imediatista e carente de

softwares de qualidade, faz-se necessário técnicas para que todo tempo desperdiçado seja extinto do processo de produção para um maior destaque no mercado de trabalho. [Machado and Tretin 2017]

O Mapeamento de fluxo de valor (MFV) é um método de modelagem baseado na observação do fluxo de processos, materiais e equipes dentro da empresa, que são transformados em um mapa. Esse mapa sintetiza os fluxos atuais dos processos. Por fluxo de valor, entende-se o conjunto de todas as atividades que ocorrem desde a obtenção da matéria prima até a entrega do produto ao consumidor final. Assim sendo, surge a problemática deste artigo: Identificar as falhas e desperdícios no processo de manutenção de um software de grande porte por meio da aplicação de MFV.

2. Fundamentação Teórica

2.1. Processo de Desenvolvimento de Software

No contexto da Crise do Software de 1970 devido a falta de mão de obra qualificada, surge um conceito de padronização do desenvolvimento de software, os chamados Processos de Desenvolvimento de software, que tornam o desenvolvimento mais compreensível. A padronização de modelos de processo começou com os modelos genéricos, que compreendem 5 atividades: Comunicação, Planejamento, Modelagem, Construção e Entrega. Com esses passos estruturados, é possível orientar ao time de desenvolvimento a situação atual do projeto e qual direção deve ser seguida [Pressman 2011].

Em 1970, Winston W. Royce propôs o modelo de desenvolvimento cascata (Waterfall) para que os processos passassem a ser bem definidos. Esse modelo de desenvolvimento é rígido em relação aos seus processos, ou seja, só é possível avançar para o próximo passo do projeto se os anteriores foram concluídos com sucesso. Para a época foi um modelo que revolucionou a forma de desenvolvimento desde os seus passos iniciais até a entrega do software. As fases são: Engenharia de sistemas, análise, projeto, codificação, teste, implementação e manutenção [Hirama 2011].

Porém, como os requisitos da construção de um software são voláteis, o modelo cascata começou se tornar rígido demais para o resultado que os clientes desejavam. Logo, um novo modelo de processo de desenvolvimento foi apresentado ao mercado produtor de software, o modelo Espiral. Em 1988 Barry Boehm apresentou o modelo espiral, que consiste em um modelo mais flexível em relação as fases do projeto e que permitiam iteração para melhorias ou consertos de erros no escopo [Pressman 2011].

Mesmo considerando os benefícios que os modelos sequenciais e evolutivos trouxeram com o objetivo de amenizar o caos do desenvolvimento de software, eles ainda não focavam na flexibilidade de serem adaptados a todos os processos e formas de desenvolvimento que as empresas aplicam, e com essas lacunas e brechas nos processos anteriores, surgiram as metodologias de desenvolvimento ágil de software.

2.2. Desenvolvimento Ágil

O desenvolvimento ágil surgiu da necessidade de adaptar os modelos sequenciais ao processo de desenvolvimento de softwares complexos, que precisavam de modelos mais rápidos, que tivessem toda a documentação associada e que as entregas aos clientes fossem feitas de forma consistente. Assim, surgiram os modelos de desenvolvimento ágil, e os mais famosos foram o *Extreme Programming* e o *Scrum* [Beck and Gamma 2000].

Os processos ágeis ganharam espaço e credibilidade por revolucionar o foco da equipe de desenvolvimento, passando a focar muito mais na satisfação do cliente e no bem-estar da equipe ao invés de produção demasiada. O Manifesto Ágil que foi publicado em 2001 mostrou quatro pilares da forma ágil de desenvolvimento: Indivíduos e interações em vez de processos e ferramentas; Software executando em vez de documentação; Colaboração do cliente em vez de negociação de contratos; Respostas rápidas a mudanças em vez de seguir planos. [Poppendieck and Poppendieck 2003]

Embora ótimos resultados tenham sido obtidos a partir da aplicação dos modelos ágeis de desenvolvimento nos projetos de pequeno e médio porte, há um grande desafio ao aplicar essa metodologia para projetos de grande porte, pois há uma série de fatores que se alteram ao se tratar de um software maior. Dentre eles, podemos citar: Desafios para manter o teste contínuo; A manutenção do sistema aumentará, pois, várias versões serão disponibilizadas ao fim das *sprints*; Sobrecarga nas gerências que precisam coordenar várias equipes e suas comunicações ao mesmo tempo; Priorização de demanda é difícil de ser criada e mantida; Tempos ociosos no processo de desenvolvimento. Tais problemas tendem a ser minimizados a partir da aplicação de práticas enxutas de desenvolvimento de software.

O *Lean* surgiu a partir das premissas do *Toyota Production System* (TPS), idealizado durante a Segunda Guerra Mundial, em 1940, com foco em reduzir desperdícios nas linhas de produção da montadora que sofria grandes impactos financeiros por baixa demanda de venda dos veículos, que eram produzidos demasiadamente. Taiichi Ohno foi o idealizador do TPS, pois ele viu que o modelo tradicional não estava gerando lucros para a empresa, devido ao alto preço dos carros comparado com o que a população ganhava de dinheiro. Sendo assim, Shigeo Shingo, outro idealizador do TPS definiu os sete desperdícios da manufatura, que são: Inventário, super processamento, superprodução, transporte, espera, movimentação e defeitos.

Esses sete desperdícios da manufatura foram convertidos para a área de software, que se tornaram: Trabalho parcialmente feito, processos extras, funcionalidades extras, mudança de tarefas, espera, movimentação e defeitos [Poppendieck and Poppendieck 2003].

Uma das formas de sintetizar o processo de desenvolvimento de software de uma empresa e constatar se há desperdícios operacionais é aplicando o mapeamento de fluxo de valor.

2.3. Mapeamento de Fluxo de Valor

O Mapeamento de Fluxo de Valor é uma ferramenta desenvolvida pelo *Operations Management Consulting Division* (OMCD), e tem um papel importante para sintetizar os fluxos atuais dos processos. Suas características o tornam um diagrama amplamente utilizado para sintetização dos processos, pois ele fornece uma linguagem comum, visual e simbólica, tem fácil visualização e compreensão, ajuda a visualizar os processos individuais e os departamentais, auxilia na melhoria como um todo e não apenas de algumas equipes, agrega técnicas e conceitos da manufatura enxuta [Rother and Shook 2003].

Após o MFV ser difundido dentro da filosofia *Lean*, ele se tornou um método adequado para empresas que desejam implementar o conceito *Lean*, pois ele indica quais as ações a serem tomadas após a identificação dos desperdícios praticados durante o de-

senvolvimento de software. O MFV possibilita medir e a avaliar os processos a baixo custo, já que é uma simples visualização dos processos realizados, e favorece a mudança cultural da empresa, já que fornece a compreensão dos benefícios obtidos pela melhoria dos processos [Machado and Tretin 2017].

3. Metodologia

O Fluxograma da Figura 1 representa o fluxo utilizado para a coleta dos dados, análise dos dados até o encontro de evidências de desperdícios operacionais.

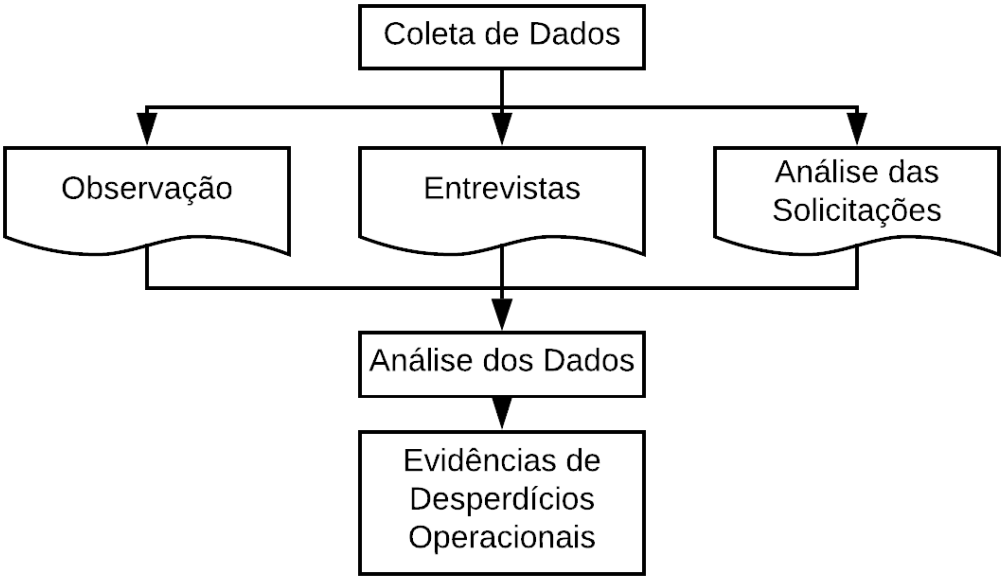


Figura 1. Fluxograma da Metodologia a ser Utilizada
Fonte: Dados da Pesquisa

3.1. Coleta de Dados

Para a coleta de dados, objetivando a identificação e redução dos desperdícios operacionais em uma fábrica de produção de software, foi utilizado a abordagem triangular, envolvendo três métodos de coleta dos dados, sendo eles: Observação, entrevistas e análise das solicitações. Essas técnicas de coleta de dados foram escolhidas devido a sua compatibilidade com o estudo, possibilitando uma abrangência maior e uma coleta melhor dos dados do objeto de estudo.

3.1.1. Observação

O primeiro método aplicado foi a observação do atual fluxo de trabalho da empresa, acerca dos processos utilizados na solicitação de customização e melhoria que os clientes requerem. Essas solicitações chegam através do centro de suporte da empresa, onde são categorizadas conforme a sua demanda: Customização de Sistema ou Melhoria de Sistema.

- Customização de Sistema: É identificada quando já existe uma funcionalidade no sistema e o cliente deseja alterá-la para melhor se encaixar nos processos operacionais do cliente;
- Melhoria de Sistema: É caracterizada quando o cliente deseja uma nova funcionalidade no sistema.

Depois de receber a solicitação de melhoria ou customização, a equipe de Gestão de Mudanças é acionada e encaminha a solicitação para a estimativa de horas da equipe responsável pela situação. Essa estimativa de horas compreende três aspectos: Análise, Desenvolvimento e Testes.

Após a estimativa da solicitação ser concluída, a solicitação é encaminhada para a equipe dos representantes comerciais, responsáveis de entrar em contato com o cliente, avisando-o a respeito da estimativa de custo que é gerada em decorrência da solicitação, caso ela seja realizada. Com o aceite de investimento por parte do cliente, a solicitação retorna para a equipe responsável para que seja alocada nas iterações de desenvolvimento, onde, inclusive, será estimado a sua data de entrega.

Após a finalização dos processos da fábrica, os analistas de qualidade entram em contato com o cliente para estar reportando que a melhoria/customização solicitada está pronta para ser utilizada.

A observação permitiu analisar o fluxo atual de forma minuciosa, com muitos detalhes nos documentos gerados.

3.1.2. Entrevista

Outra forma de coleta dos dados necessários para este estudo foi o método de entrevistas, com o propósito de validar se o modelo utilizado atualmente pela empresa nos processos operacionais, comerciais e nos processos de fábrica de desenvolvimento, supre as necessidades dos clientes, satisfazendo-os a respeito de suas solicitações. Dois métodos de entrevista foram utilizados: Face a Face e Mediada.

A entrevista Face a Face é realizada pessoalmente com o entrevistado, por outro lado, a mediada utiliza-se de uma ou mais ferramentas tecnológicas para proceder com a entrevista, tais como vídeo conferência ou ligação.

A estrutura da entrevista é semiestruturada, ou seja, foi elaborado um roteiro de questões a serem feitas para o entrevistado para que ele a responda, porém a entrevista não é limitada apenas a essas questões, é permitido que o entrevistado contribua com outras informações relevantes ao projeto.

3.1.3. Análise das Solicitações

A análise das solicitações foi mais um passo crucial ao estudo, e ocorre por meio da análise das conversas que foram realizadas com o cliente por meio da ferramenta de HelpDesk, atualmente utilizada no Centro de Suporte e na Fábrica de Desenvolvimento de Software da empresa.

Essas conversas são armazenadas em Logs que, por sua vez, contém os dados

necessários para o levantamento do MFV atual de cada solicitação, dados como: Data de recebimento, data de encaminhamento, data de contato com o cliente, se o solicitado é o que foi entregue, entre outros.

3.2. Análise dos Dados

A análise dos dados foi feita de forma específica, solicitação por solicitação. Os critérios para a seleção das solicitações foram os seguintes: Já encaminhada para desenvolvimento; Já entregue para o cliente; Com investimento por parte do cliente. Tais critérios foram selecionados com o propósito de ter uma base de solicitações com data de recebimento e data de entrega, e que tenha passado por todos os processos e equipes.

Após ter as solicitações categorizadas por demanda, elas foram submetidas ao MFV para que se possa ter uma visão clara do *lead time* de cada solicitação, gerando dados concretos de qual equipe tem se delongado mais em cumprir seus prazos com as solicitações, evidenciando a problemática levantada por essa proposta de trabalho.

4. Resultados

4.1. Evidências de Desperdícios Operacionais

A aplicação dos métodos de coleta de dados foi concluída e os seguintes resultados obtidos: O processo de observação retornou todo o caminho que uma solicitação percorre dentro da empresa, desde a sua solicitação até a entrega ao cliente. Com o método das entrevistas foi possível analisar as ações tomadas com situações de clientes críticos e que precisam de mais atenção para evitar um possível cancelamento, e a análise das solicitações retornou o tempo utilizado em cada equipe para a resolução da solicitação. Ao final, os dados coletados foram agrupados e foram encontradas 14 solicitações de melhoria / customização do software da empresa, cujas solicitações se encaixam nos critérios da 3º etapa da coleta dos dados. O mapeamento de fluxo de valor foi aplicado nas 14 solicitações previamente selecionadas. A seguir, as Figuras 2 e 3 ilustram exemplos dos mapas que comprovam o tempo de ociosidade nos respectivos processos.

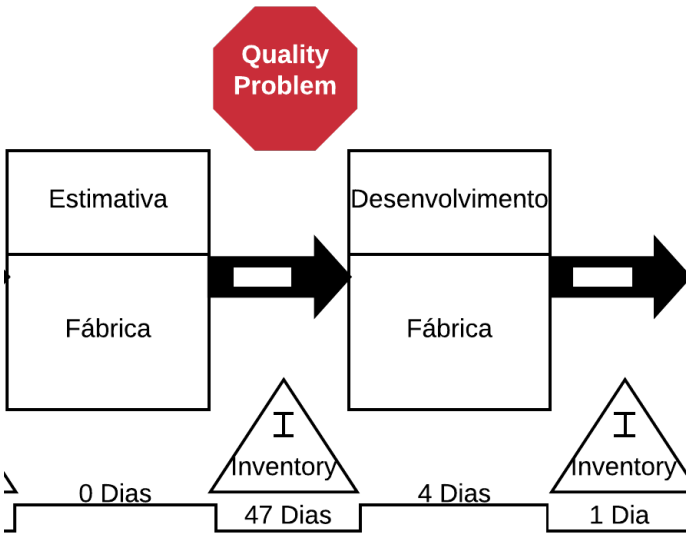


Figura 2. Solicitação 1
Fonte: Dados da Pesquisa

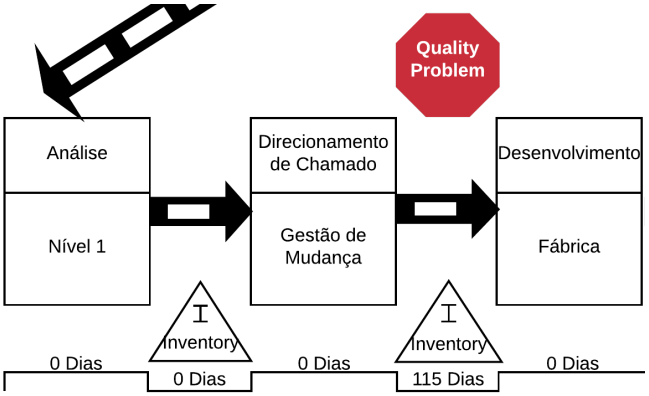


Figura 3. Solicitação 2
Fonte: Dados da Pesquisa

Na linha inferior do mapeamento é mostrado o lead-time da solicitação, ou seja, todo o tempo que ela esteve dentro da empresa. Os dias na parte superior representam o tempo trabalhado e o dias na parte inferior é o tempo de ociosidade da solicitação. As figuras apresentam apenas o problema identificado e não a solicitação completa, e, ao final da linha, é mostrado o *lead-time* total da solicitação. A solicitação da Figura 2 tem um *lead-time* de 78 dias, sendo 15 trabalhados e 63 dias de ociosidade. Na Figura 3 há um *lead-time* de 147 dias, sendo 27 trabalhados e 120 dias de ociosidade.

Dentro do contexto do objeto de estudo, alguns processos se repetem em cada uma das 14 solicitações. Após a análise dos 14 mapas, alguns destes processos apresentaram um padrão de desperdícios operacionais, que são apresentadas na Figura 4.

	Média de Dias de Ociosidade	Falhas Encontradas
Pendente de Estimativa	78,75	8
Pendente de Comercial	63,88	6
Pendente de Desenvolvimento	45,12	8
Desenvolvimento	31	2
Pendente de Gestão de Mudanças	27	1
Aceite do Cliente	18,12	8
Negociação com o cliente	18	1
Análise do Nível 1	7	1

Figura 4. Média de Dias de Ociosidade
Fonte: Dados da Pesquisa

O desperdício operacional que ficou mais evidenciado foi o Pendente de Estimativa, onde, nas 14 solicitações, ele foi evidenciado em 8 casos, conforme destacado na coluna: ”Falhas Encontradas”. Para o cálculo da média de dias de ociosidade, foi realizado uma média aritmética dos tempos de ociosidade de tal situação, onde, dos 8 casos de ociosidade, as solicitações ficaram em média 78,75 dias em situação de desperdício. O tempo ocioso que uma solicitação fica ”Pendente de Estimativa”é identificado geralmente na transição entre a equipe de Gestão de Mudanças e a equipe de Estimativas. Vários fatores podem estar elevando o tempo de ociosidade, podendo ser eles: Alta demanda de estimativas a serem realizadas; Poucas pessoas aptas a fazer a estimativa de uma solicitação;

Priorização de outras demandas ao invés de seguir o fluxo correto por ordem de chegada e prioridade principal das solicitações. Esses fatores podem estar transformando esse tempo de ociosidade em um desperdício operacional considerável, conforme evidenciado. Outro fator que merece destaque é a situação, destacada na Figura 4 como: ”Pendente de Desenvolvimento ”. Tal circunstância foi evidenciada em 8 casos dos 14 mapas analisados. A média de tempo em ociosidade das falhas encontradas foi de 31 dias. Diferentes fatores podem estar impactando em tais problemáticas, como: Má priorização de demandas, falta de comunicação entre a equipe operacional e a equipe de desenvolvimento, entre outros.

5. Conclusão

A coleta de dados obteve sucesso em seus três métodos, mostrando resultados relevantes para o estudo e evidenciando que o tempo de ociosidade das solicitações está muito alto ao ser comparado com o tempo trabalhado em cada solicitação. Esse desperdício operacional encontrado nesta análise possibilita uma visão clara dos processos que atualmente estão demandando mais tempo para serem executados e consequentemente sugere a mudança de culturas e processos visando melhor satisfação dos solicitantes.

A proposta de solução para o problema identificado é a aplicação de melhorias nos processos internos da empresa, tais como: Aumento de equipes; Capacitação das equipes atuais; Melhor categorização das solicitações no centro de suporte. A aplicação destas melhorias e coleta de resultados caracteriza-se como um trabalho futuro, comparando as coletas de dados pré-melhorias e pós-melhorias para certificar-se que estas foram benéficas e aumentaram a satistação dos solicitantes.

Referências

Beck, K. and Gamma, E. (2000). *Extreme programming explained: embrace change*. addison-wesley professional.

Hirama, K. (2011). *Engenharia de Software, Qualidade & Produtividade com Tecnologia*. Elsevier, Rio de Janeiro.

Machado, P. and Tretin, M. G. (2017). Análise dos impactos na utilização de melhoria contínua em processos Ágeis de desenvolvimento de software: Um estudo de caso.

Poppendieck, M. and Poppendieck, T. (2003). *Lean software development: an agile toolkit*. Addison-Wesley.

Pressman, R. S. (2011). *Engenharia de Software: Uma Abordagem Profissional*. AMGH Editora.

Rother, M. and Shook, J. (2003). *Learning to see: value stream mapping to add value and eliminate muda*. Lean Enterprise Institute.

Anais da 2ª Escola Regional de Engenharia de Software (ERES 2018)
em Dois Vizinhos, PR, Brasil, 22 a 24 de outubro de 2018

Detectando e propondo melhorias em cenário DevOps de uma empresa de desenvolvimento de software

Diogo Reis Pavan[1], **Érica F. de Souza**[2], **Rafael A. P. Oliveira**[1]

[1]COENS – Universidade Tecnológica Federal do Paraná
Campus Dois Vizinhos (UTFPR-DV)
Estr. p/ Boa Esperança, S/n - Zona Rural, Dois Vizinhos – PR – Brasil

[2]COENS – Universidade Tecnológica Federal do Paraná
Cornélio Procópio (UTFPR-CP)
Avenida Alberto Carazzai, 1640 – Centro – Cornélio Procópio, PR – Brasil

diogopavan@alunos.utfpr.edu.br, {ericasouza, raoliveira}@utfpr.edu.br

Abstract. *DevOps (Development and Operations) refers to a culture of collaborative software development in which two distinct teams are defined: (i) development team, responsible for creating new functionalities and products; (ii) operation team, responsible for maintenance and adjustments. DevOps automates processes in order to promote deliveries in less time, increase team collaboration, and less cost to solve problems. However, the deployment of the DevOps culture is delicate and complex due to several factors, such as a lack of cooperation among the team, the need to change habits, the implementation of information/knowledge sharing mechanisms, etc. The present study presents a strategy for the detection of critical points for the improvement of DevOps scenarios in companies that are starting with the philosophy. From a survey that was designed and executed in a real scenario of a software development company, this study presents the following contributions: (i) proposal of a generic method to detect problems in DevOps scenarios; and (ii) an empirical validation that may be a reference for similar studies.*

Resumo. *DevOps (Desenvolvimento e Operações) é o termo que se refere a uma cultura de desenvolvimento colaborativo de software no qual duas equipes distintas são designidas: (i) equipe de desenvolvimento, responsável por criar novas funcionalidades e produtos e; (ii) equipe de operações, responsável por manutenções e ajustes. A adoção de DevOps automatiza processos de modo a promover entregas em menor tempo, mais colaboração entre as equipes e menos custo para resolução de problemas. Entretanto, a implantação da cultura DevOps é delicada e complexa devido a diversos fatores como, por exemplo, falta de cooperação entre a equipe, necessidade de mudança de hábitos, implementação de mecanismos de compartilhamento de informações/conhecimento, etc. O presente estudo apresenta uma estratégia para a detecção de pontos críticos para a melhora de cenários DevOps em empresas que estão iniciando com a filosofia. A partir de uma survey que foi modelada e executada em um cenário real de uma empresa de desenvolvimento de software, este estudo apresenta as seguintes contribuições: (i) proposta de um método genérico para detecção de problemas em cenários DevOps; e (ii) uma validação empírica que pode ser base para estudos similares.*

1. Introdução

Os Métodos Ágeis são uma alternativa aos processos tradicionais de desenvolvimento de software [Pressman 2010]. Os processos tradicionais apresentam alta burocracia e inflexi-

bilidade, o que não os torna aptos para satisfazer as dinâmicas do mercado [Cruz 2015] e conduzir um desenvolvimento rápido com capacidade de mudanças [Sommerville 2010]. O advento e impulsão da filosofia ágil e metodologias ágeis ocorreu devido ao Manifesto Ágil do ano de 2001, no qual está exposto a filosofia por trás dos métodos ágeis [Sommerville 2010], listando seus valores e princípios.

A adoção das metodologias ágeis para desenvolvimento de software possibilitou a entrega de produtos mais relevantes e coerentes aos clientes [Machado 2017] e resolveram os problemas encontrados nos métodos tradicionais, provendo eficiência no desenvolvimento, satisfação das partes interessadas e percepção de desempenho do projeto [Serrador and Pinto 2015], além de diminuir o tempo necessário para o desenvolvimento [Kim et al. 2016]. Entretanto, a etapa referente à entrega e implantação do produto para o cliente exige tempo e maior atenção, devido aos frequentes problemas e complicações durante a execução dessa etapa [Kim et al. 2016].

Uma extensão dos métodos ágeis, no qual auxilia a implantação do sistema em produção é o movimento cultural DevOps [Davis and Daniels 2016, Jabbari et al. 2016, Kim et al. 2016]. A cultura DevOps (*Desenvolvimento e Operações*) busca aproximar as equipes de desenvolvimento e operações de uma organização [Davis and Daniels 2016, Wettinger et al. 2015], incentivando a adoção de práticas que levem à entrega contínua e integração contínua com qualidade [Jabbari et al. 2016]. Adicionalmente, visa à aproximação e criação de um canal de comunicação e colaboração mútuo, levando a uma maior transparência e confiança entre as equipes, trabalhando em direção às metas organizacionais compartilhadas [Davis and Daniels 2016].

Os benefícios do DevOps, conforme pesquisas [Puppet 2017, CATechnologies 2015], estão atrelados à melhoria no tempo de entrega do sistema, aumento na frequência de implantação do código, diminuição da taxa de falhas decorrentes de mudanças, rapidez no tempo de recuperação após falhas, fidelização e conquista de clientes, além de refletir positivamente no aspecto financeiro da organização. A cultura DevOps se torna fundamental no cenário atual do mercado de software, uma vez que as atualizações frequentes e contínuas dos sistemas tornaram-se prioridades em razão dos usuários e clientes esperarem novas *features* e correções de defeitos o mais rápido possível [Wettinger et al. 2015].

Por se tratar de um movimento cultural, é necessário que a organização esteja alinhada às práticas e princípios de tal cultura. A adoção efetiva do DevOps é afetada pela ausência de estratégias advindas do meio acadêmico para o profissional [Kamuto and Langerman 2017], ausência da gestão do conhecimento [Wettinger et al. 2015] e a falta de consciência das organizações de que o DevOps se trata de um movimento cultural [Samarawickrama and Perera 2017].

O objetivo do presente estudo é realizar a avaliação do cenário DevOps dentro de uma organização desenvolvedora de software utilizando-se de uma *survey*.

As principais contribuições do estudo são: (i) propor uma estratégica genérica para a detecção de problemas e lacunas em cenários DevOps; e (ii) validação empírica que pode ser base para estudos similares.

Além da seção introdutória, o presente trabalho está organizado da seguinte forma: a Seção 2 abrange os tópicos relacionados ao trabalho; a Seção 3 apresenta os materiais e métodos propostos para a realização do trabalho e êxito nos respectivos objetivos; a Seção 4 que compreende a exposição dos resultados obtidos; e, por fim, as considerações finais do trabalho estão na Seção 5 do documento.

2. Background

A presente seção apresenta os principais conceitos técnicos essenciais para o entendimento completo do trabalho apresentado.

2.1. Processos de software

Um processo de desenvolvimento de software é um conjunto de atividades e tarefas associadas ou inter-relacionadas que transformam produtos de trabalho de entrada em produtos de saída [Society et al. 2014]. A qualidade de um produto final de software está intimamente relacionada com a qualidade de um processo de software. Por se tratar de um fator de extrema importância, processos podem ser alvo de melhoria de qualidade, medições e ter apoio ferramental definido em nível de projeto [Sommerville 2010].

A agilidade no desenvolvimento de software é necessária, principalmente, devido a três fatores: *(i)* necessidades de entregas rápidas para satisfação de clientes; *(ii)* aquisição de vantagem competitiva e fatia de mercado pela empresa; *(iii)* alinhamento entre mercado e novas tecnologias [Kim et al. 2016, Samarawickrama and Perera 2017, Wettinger et al. 2015].

2.2. DevOps

O DevOps é visto como um movimento cultural que tem por intuito diminuir a distância entre o time de desenvolvimento e o time de operações [Davis and Daniels 2016, Wettinger et al. 2015]. O time de desenvolvimento responsável por criar as aplicações, adicionar funcionalidades e corrigir defeitos, enquanto o time de operações é responsável por cuidar das aplicações em produção, zelando pela estabilidade [Sato 2017]. O termo foi criado em 2008 por Patrick Debois, gerente de projetos, que propôs a discussão de métodos para solucionar os conflitos entre as áreas de desenvolvimento e operações e se popularizou por meio de eventos chamados *"DevOps Days"* [Davis and Daniels 2016].

O objetivo do DevOps é remover as barreiras existentes entre esses times, aproximando-os e criando um canal de comunicação e colaboração mútuo, levando à uma maior transparência e confiança entre as equipes e trabalhando em direção às metas organizacionais compartilhadas [Davis and Daniels 2016]. Além disso, melhorar as entregas de software, agregando qualidade, segurança e capacidade de obter *feedback* rápido sobre os produtos.

O DevOps possui práticas que dão suporte à adoção da cultura, sendo tais práticas fundamentais para enviar mudanças o quanto antes para a produção, reduzir ou eliminar erros durante a implantação e encontrar e reparar falhas com rapidez no sistema [Bass 2018]. As principais práticas DevOps são: controle de versão, testes automatizados, entrega contínua, implantação contínua e monitoramento. As práticas do DevOps, por vezes, são sobrepostas e uma dá suporte à outra [Vadapalli 2017], sendo possível suas realizações por conta da automação das atividades que compõem tais práticas.

Na próxima seção é apresentado o modo como o presente estudo sugere a avaliação e diagnóstico de cenários DevOps, visando a sua melhoria e aderência tecnológica.

3. Materiais e Métodos

Nesta seção são descritas as atividades executadas para alcançar o objetivo do presente trabalho, o qual está pautado em avaliar o cenário de DevOps em uma empresa desenvolvedora de software que está iniciando na adoção da cultura. Deste modo, este trabalho é motivado por três questões de pesquisa (QP):

- **QP1**: Qual o nível de familiaridade do time de desenvolvimento e operações com a cultura DevOps?
- **QP2**: Qual a impressão dos times de desenvolvimento e operações acerca da cultura DevOps adotada?
- **QP3**: Como as práticas são vistas pela equipe?

3.1. Estrutura da *survey*

Pelo fato de DevOps se tratar de uma metologia nova, a estratégia empírica *survey* foi escolhida pelos pesquisadores. A *survey* é um meio de coleta de informações de determinado grupo de pessoas sobre suas ações, comportamentos ou opiniões por meio de um instrumento de pesquisa, geralmente questionários [Freitas et al. 2000, Wohlin et al. 2012]. O intuito da *survey* é ajudar a responder as questões de pesquisa propostas e foi elaborada com o auxílio de uma ferramenta online para a criação de questionários[1].

As questões da *survey* foram definidas com o propósito de atender e responder as questões de pesquisa. Dessa maneira, para a QP1 são formuladas as seguintes perguntas: ”Você já ouviu falar do termo DevOps?”(Q1), ”Como você definiria o significado de DevOps?”(Q2), ”Você já trabalhou com práticas da cultura DevOps?”(Q3); para a QP2, a seguinte pergunta foi incluída: ”Na sua opinião, a equipe/produto que atualmente você trabalha utiliza práticas da cultura DevOps?”(Q4); e referente a QP3, que visa descobrir quais são as principais práticas utilizadas naquele cenário de DevOps, a seguinte pergunta foi formulada: ”Na sua opinião, quais dessas práticas, ferramentas ou tecnologias são encontradas no ambiente e equipe de trabalho?”(Q5). Adicionalmente, serão acrescentadas algumas questões sobre o perfil do profissional: ”Há quanto tempo você trabalha na área de TI?”(Q6) e ”Há quanto tempo você trabalha na empresa atual?”(Q7).

3.2. Aplicação da *survey*

Para validar a efetividade da *survey* em detectar problemas em ambientes iniciais de DevOps, foi planejada uma aplicação da *survey* em um ambiente real de desenvolvimento. Uma empresa real de software, que chamaremos de Empresa A, por motivos de sigilo, foi o alvo da pesquisa. A Empresa A está localizada no Paraná, é formada por mais de 400 colaboradores, possui atuação nacional, fornecendo soluções para supermercados, lojas de materiais de construção e restaurantes, contando com mais de 4 mil clientes.

A pesquisa foi executada especificadamente no departamento de Novos Produtos, que é o responsável por desenvolver as novas soluções da empresa. Foram envolvidas na atividade três equipes que desenvolvem um produto, tais equipes - totalizando 25 colaboradores - trabalham com metodologias ágeis e estão iniciando o trabalho com o DevOps.

O fluxo planejado para a execução das atividades será: coletar os dados por meio da *survey*, analisar os dados obtidos e, por fim, identificar as lacunas e surgerir melhorias para o cenário de DevOps da Empresa A.

4. Resultados e Discussões

Nesta seção serão apresentados os resultados obtidos da *survey*, uma análise das respostas e as ameaças à validade da pesquisa.

4.1. Resultados da *survey* e análise dos dados

A *survey* teve aderência de 18 participantes e as primeiras perguntas da *survey* estavam relacionados ao perfil do profissional. Na Figura 1 é possível visualizar as respostas obtidas acerca desse tópico.

[1]Google Formulários (https://www.google.com/intl/pt-BR/forms/about/)

Figura 1. Os gráficos representam, respectivamente, as respostas para ”Há quanto tempo você trabalha na área de TI?” (Q6) e ”Há quanto tempo você trabalha na empresa atual?” (Q7)

As respostas das três perguntas referentes a QP1 são vistas nas Figuras 2 e 3. As respostas da pergunta QP2 estão na Figura 4. Por fim, a Figura 5 exibe as respostas para QP3.

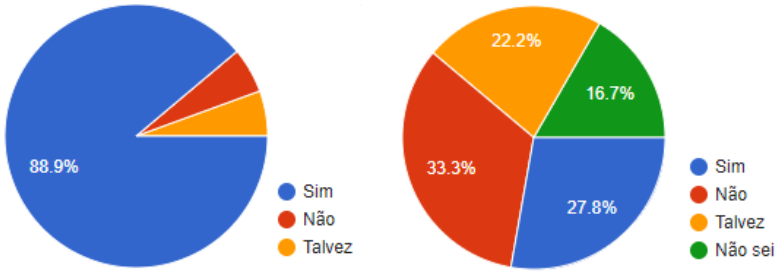


Figura 2. Os gráficos representam, respectivamente, as respostas para ”Você já ouviu falar do termo DevOps?” (Q1) e ”Você já trabalhou com práticas da cultura DevOps?” (Q3)

Referente ao perfil dos profissinais integrantes das equipes, é possível verificar, em sua maioria (94,4%), profissionais experientes que possuem mais de 6 anos trabalhando na área de TI e desses profissionais, apenas 33,3% trabalham há menos de um ano na Empresa A.

As perguntas Q1, Q2 e Q3, referentes à QP1, demonstram que grande maioria (88,9%) dos colaboradores já ouviram falar do termo DevOps. Entretanto, apenas 8 dentre os 18 participantes da pesquisa tentaram definir o termo DevOps, apesar de já estarem inseridos em um ambiente DevOps em estágio inicial. Em relação à definição ou significado do termo, muitas respostas são rasas, não abrangendo uma definição completa. Adicionalmente, apesar dos profissionais experientes e a grande maioria ter ouvido falar do termo, poucos afirmam (27,7%) terem trabalhado com práticas da cultura DevOps. Portanto, a familiaridade dessas equipes com a cultura DevOps é baixa, configurando-se como um ponto a ser melhorado.

Das respostas obtidas para Q4, referente à QP2, 38,8% dos colaboradores acreditam que sua equipe ou as equipes que desenvolvem o produto, utilizam práticas da cultura DevOps. Essa visão dos colaboradores é significativamente baixa e a impressão é de que poucos aspectos e práticas da cultura são utilizados.

A pergunta Q5, referente à QP3, faz oposição à pergunta anterior, pois muitas práticas que fazem parte da cultura DevOps tiveram um bom número de respostas e na visão dos colaboradores, estão presentes nas equipes. Entretanto, alguns aspectos tiveram pontuações baixas, tal como Melhorias e reflexões contínuas e Monitoramento de

Framework de Integração entre as áreas de Desenvolvimento e Operação
Entrega contínua
Operações de suporte ao ambiente de desenvolvimento
Conjunto de processos e ferramentas que garantam uma sincronia harmoniosa entre desenvolvimento e operação visando alcançar melhores resultados de execução de trabalho e entrega de valor.
Basicamente, seria uma metodologia/prática utilizada no desenvolvimento de software para que a equipe de desenvolvimento e infra estrutura/deploy consigam ter uma maior integração, trabalhando em conjunto e de forma mais sincronizada, com o objetivo de diminuir o tempo das entregas para produção.
Capacidade de produzir e gerenciar o desenvolvimento de software em todas as suas fases de forma automática e sincronizada, cobrindo todos os processos
Em poucas palavras, um conjunto de ferramentas para auxiliar/automatizar processos de desenvolvimento de software.

Figura 3. Lista das respostas para ”Como você definiria o significado de DevOps?” (Q2).

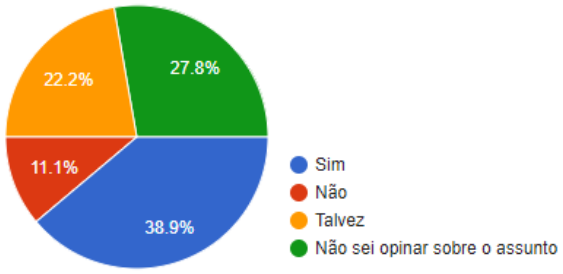


Figura 4. O gráfico representa as respostas para ”Na sua opinião, a equipe/produto que atualmente você trabalha utiliza práticas da cultura DevOps?” (Q4).

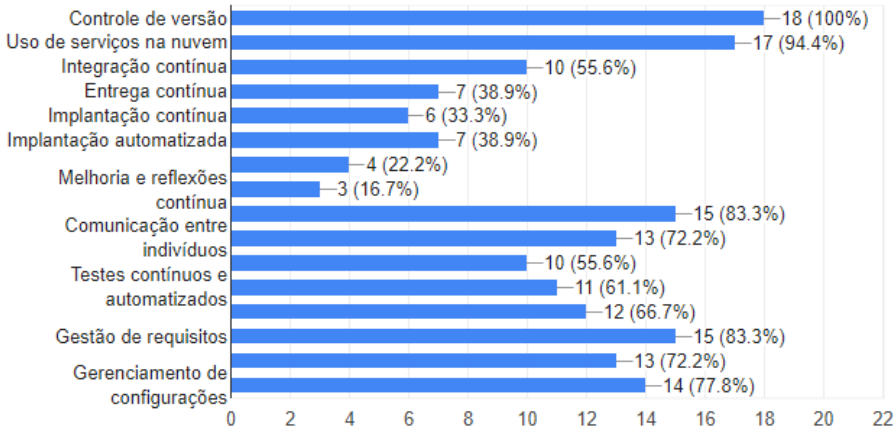


Figura 5. O gráfico representa as respostas para ”Na sua opinião, quais dessas práticas, ferramentas ou tecnologias são encontradas no ambiente e equipe de trabalho?” (Q5).

desempenho do sistema.

Após a análise dos dados, percebe-se a necessidade de um aculturamento das equipes em relação ao DevOps. A divergência do número de pessoas que ouviram o termo e sabem definí-lo e a visão de que a cultura DevOps adotada na empresa é baixa, porém muitas práticas são realizadas, mostram que o problema está atrelado ao conhecimento

sobre a cultura, práticas e princípios. Adicionalmente, é preciso melhorar nas questões referentes à melhoria contínua e monitoramento do sistema.

Perante os resultados obtidos, as sugestões para melhoria da cultura DevOps na empresa são: (i) fortalecer a cultura DevOps através do compartilhamento do conhecimento: essa melhoria pode estar atrelada à Gestão do Conhecimento, que consiste de um conjunto de princípios e técnicas que visam criar e gerenciar conhecimento de modo efetivo e eficiente para as pessoas terem acesso ao conhecimento [Vasanthapriyan et al. 2015]. Treinamentos, palestras e ferramentas, como Wikis ou blogs, são exemplos de práticas que podem colaborar com o conhecimento das práticas e princípios da cultura DevOps, auxiliando na sua adoção; (ii) prover um melhor monitoramento do sistema: a apresentação dinâmica das informações relevantes acerca do desempenho do sistema, por meio de monitores ou televisão, além de contribuir com o compartilhamento de conhecimento, auxilia em um dos aspectos mais importantes da cultura DevOps; e (iii) desenvolver a melhoria contínua e reflexões: esse aspecto está atrelado às metodologias ágeis, portanto deve sempre ser executada e fortificada dentro das equipes.

4.2. Ameaças à validade

Por meio de uma análise prévia do cenário, as possíveis limitações da pesquisa levantadas foram:

- Somente uma empresa: a pesquisa foi realizada apenas em uma empresa real de desenvolvimento de software, limitando a pesquisa ao contexto de tal empresa;
- Quantidade de entrevistados: pelo questionário não ter um teor de obrigatoriedade, não houve aderência total dos colaboradores.

5. Considerações Finais

A cultura DevOps tem por objetivo melhorar a entrega de software, provendo práticas e princípios, entretanto sua adoção efetiva é prejudicada devido ao aculturamento necessário da organização ou equipes de desenvolvimento e operações. Este estudo, o qual se insere em um trabalho que visa a cooperar com a indústria, apresenta uma *survey* que pode ser aplicada em times que estejam iniciando na cultura DevOps. A *survey* se mostrou efetiva em revelar pontos para que gestores dediquem esforços em melhorias. Como trabalho futuro, espera-se a ampliação da *survey* e sua aplicação sistemática em times de desenvolvimento para sugestões de melhoria do ambiente DevOps.

Referências

Bass, L. (2018). The software architect and devops. *IEEE Software*, 35(1):8–10.

CATechnologies (2015). Assembling the devops jigsaw. https://www.ca.com/us/modern-software-factory/content/assembling-the-devops-jigsaw.html, Setembro.

Cruz, F. (2015). *Scrum e Agile em Projetos: Guia completo*. Brasport, Rio de Janeiro.

Davis, J. and Daniels, K. (2016). *Effective DevOps: Building a Culture of Collaboration, Affinity, and Tooling at Scale*. O'Reilly Media, Inc., Sebastopol.

Freitas, H., Oliveira, M., Saccol, A. Z., and Moscarola, J. (2000). O método de pesquisa survey. *Revista de Administração da Universidade de São Paulo*, 35(3):105–112.

Jabbari, R., bin Ali, N., Petersen, K., and Tanveer, B. (2016). What is devops?: A systematic mapping study on definitions and practices. In *Proceedings of the Scientific Workshop Proceedings of XP2016 (XP 2016)*, pages 1–11, New York. ACM.

Kamuto, M. B. and Langerman, J. J. (2017). Factors inhibiting the adoption of devops in large organisations: South african context. In *Proceedings of the 2nd International Conference on Recent Trends in Electronics, Information Communication Technology (RTEICT 2017)*, pages 48–51, Bangalore. IEEE.

Kim, G., Debois, P., Willis, J., and Humble, J. (2016). *The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations*. IT Revolution Press, Portland.

Machado, P. H. A. (2017). Redução de desperdícios no desenvolvimento de software de grande porte por meio de ferramentas lean. Dissertação (mestrado em engenharia de produção e sistemas), Universidade Tecnológica Federal do Paraná, Pato Branco.

Pressman, R. (2010). *Engenharia de Software: Uma abordagem profissional*. McGraw-Hill Education, New York, 7^{th} edition.

Puppet (2017). 2017 state of devops report. https://puppet.com/resources/whitepaper/2017-state-of-devops-report, Setembro.

Samarawickrama, S. S. and Perera, I. (2017). Continuous scrum: A framework to enhance scrum with devops. In *Proceedings of the 17th International Conference on Advances in ICT for Emerging Regions (ICTer)*, pages 19–15, Colombo. IEEE.

Sato, D. (2017). *DevOps na prática - entrega de software confiável e automatizada*. Casa do Código, São Paulo.

Serrador, P. and Pinto, J. (2015). Does agile work? — a quantitative analysis of agile project success. *International Journal of Project Management*, 33(5):1040–1051.

Society, I. C., Bourque, P., and Fairley, R. E. (2014). *Guide to the Software Engineering Body of Knowledge (SWEBOK(R)): Version 3.0*. IEEE Computer Society Press, Los Alamitos, CA, USA, 3^{rd} edition.

Sommerville, I. (2010). *Software Engineering*. Addison-Wesley Publishing Company, USA, 9^{th} edition.

Vadapalli, S. (2017). *Hands-on DevOps: Explore the concept of continuous delivery and integrate it with data science concepts*. Packt Publishing, Birmingham.

Vasanthapriyan, S., Tian, J., and Xiang, J. (2015). A survey on knowledge management in software engineering. In *Proceedings of the International Conference on Software Quality, Reliability and Security (QRS 2015)*, pages 237–244, Washington. IEEE Computer Society.

Wettinger, J., Andrikopoulos, V., and Leymann, F. (2015). Automated capturing and systematic usage of devops knowledge for cloud applications. In *Proceedings of the 3rd International Conference on Cloud Engineering (IC2E 2015)*, pages 60–65, Tempe. IEEE Computer Society.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., and Wesslén, A. (2012). *Experimentation in Software Engineering*. Springer, New York.

Anais da 2ª Escola Regional de Engenharia de Software (ERES 2018)
em Dois Vizinhos, PR, Brasil, 22 a 24 de outubro de 2018

Achievements, Challenges and Opportunities on Mutation Testing of Concurrent Programs

Rodolfo Adamshuk Silva[1], **Simone do Rocio Senger de Souza**[2]

[1]Software Engineering Course Coordination – Federal University of Technology - PR
85660-000 – Dois Vizinhos – PR – Brazil

[2]Institute of Mathematics and Computer Sciences – University of São Paulo
13566-590 – São Carlos – SP – Brazil

rodolfoa@utfpr.edu.br, srocio@icmc.usp.br

Abstract. *With the increasing advances in the hardware technology and the massive presence of multicore processors in personal computers, concurrent programming has been becoming more popular. Therefore, new challenges have been emerged due to the communication and synchronization aspects inherent in concurrent programs. The testing activity in this scenario is considered essential for the delivery of reliable programs. Mutation testing is a fault-based testing criterion that uses mistakes done by developers during the software development. The application of this criterion in concurrent programs is challenging due to the non-determinism and concurrency problems, such as starvation and deadlock. Studies have been conducted for the application of the mutation test in concurrent programs, however, some aspects still need attention and improvements, leading researchers to face a collection of challenges. This paper presents a roadmap of this field identifying the achievements, challenges, and opportunities to study in the context of mutation testing of concurrent programs. As a result, researchers may find a guide for the proposal of new researches in the field.*

1. Introduction

Concurrent programming paradigm has become more popular nowadays. It happens because the hardware supports parallel programming, increasing the performance of concurrent programs. Concurrent programming adds new concepts, models and primitives to sequential programming, supporting the development of algorithms that, when executed, create processes that can perform concurrently and that interact in problem-solving. In this way, concurrent programming allows the split of a task into smaller portions to increase application performance, improve fault tolerance, or optimize the use of resources such as processors, memory, and I/O devices.

A concurrent program is composed of two or more processes or threads that work together to perform a task (Andrews, 2001). This programming style differs from sequential programming in which the Von Neumman's architecture is used that is composed of statements executed sequentially, conditional and unconditional deviations, calls to procedures and repetitive structures. Concurrent programming permits a performance optimization of applications, exploiting the concurrency in different architectures, allowing better use of available resources.

The basic idea of concurrent programming is the partition of an application into concurrent processes, each one responsible for solving a piece of the problem. The partition is done through additional software features not available for sequential programs, such as the activation and termination of concurrent processes. According to Almasi and Gottlieb (1989), concurrent processes are processes that began its execution and, at a specific point in time, have not finalized it yet. These processes compete for resources such as processors, memory, and I/O devices. Parallel processes represent a special type of concurrent processes, as there is a guarantee that they are running on different processors at the same time.

Mutation testing is a testing criterion that uses information about typical errors that can be made in the software development process to derive test cases. Hence, the typical errors in a domain or paradigm of development are characterized and implemented as mutation operators that during the test activity, generate modified versions (mutants) of the product under test. The objective of the mutation test is to create a test set that can identify the behavior difference between the original program and the mutated program that has undergone a minor modification. When this difference in behavior is found, then it can be said that the mutant is dead (killed). Consider a program P that runs with a set of T test cases. The creation of mutants is based on a model of faults F. Each fault f present in F it is introduced in P one by one producing a set of mutants M. Each element present in F is a mutation operator. When M is executed with T and has a behavior other than P, mutant M is killed by the test case t. The goal is to kill all M with at least one t. If a mutant cannot be killed, the tester needs to show that M is equivalent to P or to add new t that kills M. If in the attempt to kill a mutant, a new t is generated and when executed in P, makes it to fail, the program need to be corrected and the test must reinitialize.

In the context of concurrent programs, apply software testing becomes a challenge because new issues must be explored such as nondeterminism and race conditions. When a test input is executed in a sequential program, only one correct output is given. The statements in a sequential program are always executed in the same order and do not change from one execution to another. This is false for concurrent programs, in which different executions of the same input may exercise different statements and may result in different outputs depending on the order in which the processes were executed. This can be a problem in mutation testing. When a mutant M is executed by a test case t, it will be killed if it exhibits an incorrect output according to a given specification. In the case of concurrent programs, an alternative would be the comparison of the set of all possible results of M with t with all possible results of P with t and then kill M if any output is different.

This paper addresses a roadmap, defining the achievements, challenges, and opportunities for mutation testing of concurrent programs. The remainder of the paper is organized as follows: Section 2 presents an overview of the roadmap. Section 3 provides the achievements in mutation testing of concurrent programs; Section 4 addresses the challenges identified in the mutation testing procedure in concurrent programs; Section 5 describes some opportunities in the mutation testing of concurrent programs; the conclusions are summarized in Section 6.

2. Mutation testing research roadmap

A roadmap provides directions to reach the desired destination starting from the body of knowledge of the field and going until an ideal scenario. The mutation testing for concurrent programs is organized as follows:

- Achievements: The first section constitutes in the theoretical background of mutation testing of concurrent programs. The main research in the area is the definition of mutation operators. Each programming language, library, and API has its own set of mutation operators. Another point is how the mutants will be executed and evaluated. This characterizes the definition of testing strategies that basically uses deterministic and non-deterministic approaches. Finally, some tools have been defined to support the testing application.
- Challenges: In this section, we discuss the main problems faced in the definition of new techniques, approaches, and tools in the context of mutation testing of concurrent programs. Example of challenges is the program selection, mutant generation, and the definition of deterministic or non-deterministic execution. These challenges must guide researches to propose solutions to these problems.
- Opportunities: The last section presents the opportunities in the research field, with examples of researches that must be addressed to improve the application of the mutation test to concurrent programs.

Fig. 1 illustrates the roadmap. The main activities of the mutation testing process are at the top of the figure, namely initial execution, mutant generation, mutant execution, and equivalent mutant identification. In the horizontal, there are the roadmap sections, namely achievements, challenges, and opportunities. In the white boxes, there are the aspects identified.

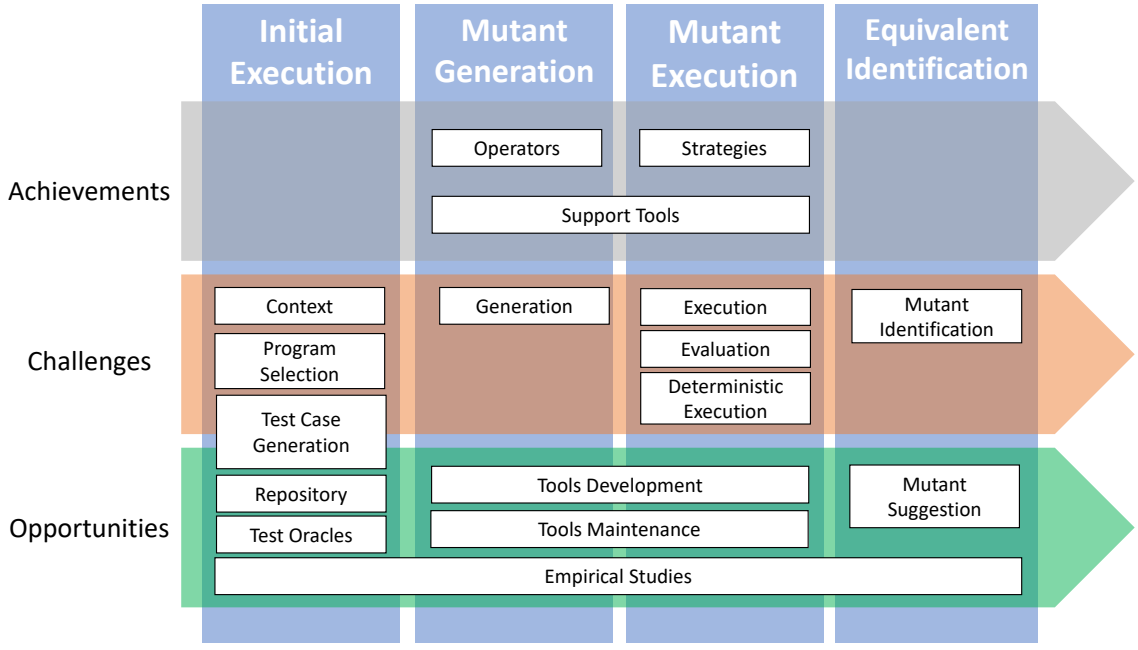


Figure 1. Mutation testing for concurrent programs roadmap.

3. Achievements

Besides the knowloge aquired during the years in the researching of mutation testing of concurrent programs, this paper is based on some empiral studies defined in the literature. Souza et al. (2011) presented a systematic review related to concurrent testing approaches, bug classification and testing tools. Melo et al. (2015) developed a catalog containing 116 tools for testing concurrent programs. **Mutation operators.** In the literature, several approaches have been defined for the definition of mutation operators for concurrent programs. Offutt et al. (1996) propose the definition of mutation operators for Ada programs. Considering the operators defined in Offutt et al. (1996), Silva-Barradas (1998) presents the application of the mutation test criterion for concurrent programs in Ada. The author classifies the mutation operators into Mutation operators for declarations and mutation operators for conditional expressions. M. Delamaro et al. (2001) defined a set of 20 mutation operators to test concurrency and synchronization aspects of the Java language. The authors state that mutation operators cover most of the aspects related to synchronization and concurrency and are also low cost (generating a small number of mutants). Ghosh (2002) presents a proposal of mutation operators for concurrent object-oriented programs in Java. The focus of the approach is focused on cases where access to shared data needs to be protected, but it is not. The failure model assumes that programmers may not use the *synchronized* construct when needed.

Giacometti et al. (2003) present an initial proposal for the application of the mutation test for programs in message passing environment, considering the PVM (Parallel Virtual Machine) environment. Unlike the other works, this one presents the definition of mutation operators considering a predefined concurrent programming pattern, while the others define mutation operators only for some functions that allow concurrent programming in such languages. Bradbury et al. (2006) defined mutation operators for Java programs (J2SE 5.0). The authors used the fault taxonomy defined by Farchi et al. (2003) as a basis for the creation of the operators. Jagannath et al. (2010) define mutation operators for programs developed using the actor-oriented programming model. A program developed with this approach consists of a set of concurrent objects that communicate through message exchange.

Wu and Kaiser (2011) present an approach where second-order mutation operators are applied which generate competition errors not represented by first-order operators. The authors present 6 second-order mutation operators for Java. Silva et al. (2012b) defined 26 mutation operators for concurrent programs in MPI. The operators are classified into 3 categories organized according to the function-target where the mutation operators are applied. The "Collective" category presents mutation operators that are applied in collective-communication functions, the "Point-to-point" category presents mutation operators that are applied in point-to-point communication functions subdivided in operators applied on send, receive or others point-to-point functions and the "All" category presents operators that can be applied in both collective and point-to-point functions. Gligoric et al. (2013) explored selective mutation for concurrent mutation operators for Java programs. The authors used the mutation operators implemented by Bradbury et al. (2006) and created three operators.

Testing strategies. To deal with the non-determinism in the testing of concurrent programs, some strategies were defined. The MET (**Multiple Execution Testing**) ap-

proach consists of executing a program P with an input x several times and examining the result of each execution. If one of the executions presents an output not expected, an error was identified in the program. This technique does not ensure that all possible synchronization sequences were executed. Therefore, a different synchronization sequence can lead to an error in the program. In the DET (**Deterministic Execution Testing**) approach presented in Tai et al. (1989), each test case is defined by an input x and a synchronization sequence s. For each test case, the execution of the x entry with the sequence s is forced, and the result is observed. Carver (1993) defined the **Deterministic Execution Mutation Testing** (DEMT) in which the mutant is run deterministically. MET is used to generate different synchronization sequences of the original program. A pair $\{$input, synchronization$\}$ is considered as test input and is applied to both original and mutant programs.

Offutt et al. (1996) present an approach in which the original program is executed n times with the same test case T to create a subset of n possible executable outputs. After that, each mutant is freely executed, and the output generated is compared with the subset. The mutant is killed if the output is not present in the subset and do not consider the synchronization sequence followed. Silva-Barradas (1998) defined the **Behavior Mutation Analysis** approach based on behaviors that a program can show. A behavior is composed of a synchronization sequence and an output. Initially, the approach works as DEMT, in which the mutant is forced to execute a synchronization sequence and is killed if the output obtained was different or if the mutant was incapable of following the sequence. M. E. Delamaro (2004) developed an approach for reproducing the run of a concurrent Java program using instrumentation. It is based on the technique of **Record and Playback**, in which the synchronization sequence occurred during the run of synchronized methods and objects is recorded in the recording phase. In the playback phase, the synchronization sequence guides the next event to be run when a thread enters a synchronized point.

Lei and Carver (2010) proposed the **Reachability Test** in which all the feasible synchronization sequences are obtained, reducing the number of redundant ones. Through the identification of "dispute conditions" between pairs of synchronizations, the approach determines during execution which synchronizations are possible to occur in a new run. The prefix-based testing technique is employed to run the program deterministically until a certain part and, after that point, it allows non-deterministic execution. Silva (2013) and Silva et al. (2012a) developed two approaches to support mutation testing. the **DEMT Adapted** approach runs the mutant deterministically, and the output is compared with the set of all possible outputs generated by reachability testing. The mutant is killed if it presents a different output or could not follow the synchronization sequence. The **MET Adapted** approach runs the mutant freely several times with the objective to perform different synchronization sequences, and the mutant is killed if it shows an output not presented by the original program. Souza et al. (2015) developed a **Composite Approach** that uses reachability testing to guide the selection of the synchronization sequences for covering a specific structural testing criterion. The information about synchronization coverage is used for the selection of a test case to be part of the test case set. The objective is to reduce the test activity costs by reducing the number of tests necessary for code coverage.

Despite the benefits of techniques to deal with the non-determinism, the computational costs are high for the execution of different synchronization sequences for each input. This cost increases in the mutation testing, once each mutant has different synchronization sequences that must be considered during the test.

Support tools. For mutation testing, a few tools have been developed to support the mutation testing of concurrent programs. **Javalanche** (Schuler & Zeller, 2009) is an open source framework for mutation testing that applies a subset of method-level mutation operators. A characteristic of this tool is the possibility of ranking mutations by their impact on the behavior of program functions. Javalanche implements a subset of method-level mutant operators composed of 7 operators. **MutMut** (Gligoric, Jagannath, & Marinov, 2010) proposes an approach for an efficient execution of mutants in multithreaded programs. It uses a technique for the selection of mutants to be executed. When the original program is executed, the technique selects points in the code for mutation considering relevant aspects of the concurrent programs. The approach also enables the tester to select a *thread* to be executed, forcing the mutation introduced to be executed. **ConMan** (Bradbury et al., 2006) implements a set of mutation operators for concurrent programs in Java (J2SE 5.0). The mutation operators are classified into operators that modify critical regions, keywords, and calls for concurrent methods and operators that replace concurrent objects.

CCmutator (Kusano & Wang, 2013) implements those operators as well as new specific mutation operators for concurrent programs in *PThreads*. It utilizes the High Order Mutation technique, in which two or more mutations are inserted in the program for the creation of strong mutants and improvements in the quality of the testing case set. **Comutation** (Gligoric et al., 2013) uses selective mutation based on the mutation operators for concurrent Java programs. Selective mutation selects a subset of mutation operators in which test cases that have a high mutation score for this subset also feature for the other operators. The objective is to reduce the mutation testing cost. **ValiMut** (Silva, 2013) tool supports the use of mutation testing in competing programs developed in MPI. The tool allows the application of mutation operators defined for MPI in Silva et al. (2012b). This tool uses some modules of the ValiMPI tool (Hausen, Vergílio, Souza, Souza, & Simão, 2007) and applies the adapted MET approach (previously described) for the deterministic execution of the mutants. **BeMutation** (Behavior Mutation) (Silva, 2018) is a tool to support the application of mutation testing in Java multithread programs. BeMutation has implemented 27 mutation operators defined in Bradbury et al. (2006) and extended in Gligoric et al. (2013). The tool uses JPF-Inspector tool to generate synchronizations sequences.

4. Challenges

Context. The context means the kind of concurrent programs the approach will be applied to. There are several libraries and APIs that support concurrent programming, such as MPI (Message Passing Interface), PVM (Parallel Virtual Machine), PThreads, Java multithreading, etc. The challenge related to the context is the definition of a procedure to apply mutation testing for a specific programming language or libraries. For each language, a new procedure must be defined, considering the fault model and the mutation operators that are strictly related to the nuances of the programming language syntax.

Program selection. In the attempt to develop new supporting tools, researchers and developers must have a set of programs to evaluate characteristics of the tool, such as scalability and completeness related to the target programming language. Therefore, a challenge in this scenario is to find a set of programs classified using some software metrics as size (or Lines of Code), complexity, toy or real program, etc. The Software-artifact Infrastructure Repository (SIR, 2018) is a repository in which a set of programs can be found, however, when it comes to concurrent programs, the original version of the program is not available, only the faulty version. This was a problem, once it was impractical to correct the programs before the use in the experiment due to the complexity of the programs.

Mutant generation. The mutant generation is another issue in the testing of concurrent programs. Even with mutation operators defined for a given programming language, for some programming languages (e.g. PVM) there is not a tool to support the generation of the mutants. It is necessary a tool to support the generation of the mutants to guarantee the application of the mutation testing criterion. **Mutant execution.** As happens in the generation, it is crucial the use of a tool for the supporting of the mutant execution. In the execution task of mutation testing of concurrent programs, it is necessary to identify the synchronization sequence executed in the execution, especially when the Deterministic Testing approach is used. **Mutant evaluation.** After the execution of the mutant, it is necessary to evaluate whether output of the mutant is valid or not. It is necessary to evaluate the output, since different executions of a concurrent program with a single input may generate different and correct outputs. If the mutant presents an output that is different from the execution of the original program, it is necessary to evaluate if the different output was generate by the mutation or a different synchronization sequence. If the output would never be presented by the original program, the mutant is considered as dead. Therefore, to evaluate this, it is mandatory to know all possible outputs of the original program. This is impractical due to the size and complexity of concurrent programs. An alternative is to use the deterministic execution and use a synchronization sequence as support for this evaluation. The challenge is that there is no tool to support the evaluation of mutants. Some testing strategies were defined, but empirical evaluations are necessary to compare and identify the benefits of each one.

Test case generation. In the testing of programs, it is necessary to test different inputs and evaluate the outputs. In the testing of concurrent programs, besides the input, it is necessary to select synchronization sequences to execute and evaluate. A challenge in this context is the use of tools and/or techniques to support this activity. **Equivalent mutants identification.** The identification of equivalent is an indecisive problem for which there is no general purpose algorithm that support it. However, ignore them is not a good practice because it may lead to inaccurate mutation scores. The solution would be the generation of a good set of test cases that would kill almost all mutants so that the effort to analyze equivalent mutants would be low.

Deterministic vs. non-deterministic execution. In the context of testing concurrent programs, several approaches were defined to deal with the non-determinism inherent in such programs. However, there is a difference in efficiency and cost among them (i. e. an approach with high efficiency have a high cost and *vice-versa*). To the best of our knowledge, no tool supports the execution of a single synchronization sequence and that

allows deterministic execution of threads in Java programs. The Java PathFinder model checking tool is, so far, the most useful tool that saves the execution trace and allows the deterministic execution. The problem with JPF is that it is not a testing tool, but a model checking tool, therefore, it uses a graph with states and transitions to execute all possible paths. One thing that is important to highlight is that even executing all possible paths, JPF does not execute all possible synchronization sequences due to the race condition algorithms used to identify states with possible thread interleaving.

5. Opportunities

Catalog of programming languages. An important aspect to be considered is the creation of a catalog of programming languages and all the techniques, support tools and scientific papers available in the literature for each concurrent programming language, library, and API. **Benchmark and repository of programs.** Along with the catalog of the programming languages, an opportunity arises from the generation and categorization of concurrent programs in different programming languages. Those programs must be categorized using software metrics as size (or Lines Of Code), complexity, toy or real program, etc. This may help researchers and developers to evaluate supporting tools and be favorable in the conduction of empirical studies the field of mutation testing for concurrent programs. A research opportunity would be to catalog the benchmarks among the results with different test cases used, to reduce execution time and facilitate the comparison of the mutant score. **Support tools.** Supporting tools are required for an efficient application of the software testing activity. However, some tools are made just as a proof of concept of proposed approaches and sometimes are not available for download. Others need maintenance and adaption to work as it should be. An opportunity is to contact authors and ask for tools and available documentation to evolve and maintain those tools. Besides that, new tools can be developed to support the testing activity in environments where no tool is available.

Testing oracles. the results of test executions is a time-consuming and at times error-prone activity. In the context of concurrent programs, the problems with manual verification are exponential. A single test input in a concurrent program can generate different and correct outputs due to the different synchronization sequences. Depending on the testing strategy, a given input must be executed several times in an attempt to exercise different synchronization sequences, therefore generating different outputs. In addition, the inherent complexity of concurrent programs generally means that significantly larger test sets are required to achieve adequate test coverage, thus further increasing the cost of manual verification. An alternative to manual verification is to develop a test oracle that automatically verifies the results of test executions. However, the cost of developing an oracle by hand can approach that of implementing the system itself (Hunter & Strooper, 2001). The opportunity in this context is the creation of test oracles to support the application of mutation testing of concurrent programs.

Equivalent mutants suggestion. The identification of equivalent mutants is, so far, a manual activity. Therefore, an opportunity in this context is the use of heuristics to identify equivalent mutants and present the differences (looking for aspects of reachability, infection, and propagation of the execution) of the mutant in comparison with the original program. **Empirical studies.** Experimental studies in Software Engineering are important for the evaluation and evolution of existing techniques and tools. Through eval-

uation, you can identify gaps in approaches and limitations in tools. One opportunity is in planning and conducting experimental studies to verify the effectiveness in finding interesting synchronization sequences and evaluation of deterministic and non-deterministic execution of mutants.

6. Conclusion

The application of mutation testing for concurrent programs is a field in software engineering that has been received more attention. Therefore, several testing strategies and tools have been developed. The testing of concurrent programs has its challenges and the mutation testing has its own that must be investigated. In this paper, a roadmap in the field of mutation testing for concurrent programs is presented. It is known that covering into one article all ongoing and foreseen research directions is impossible, therefore, the main challenges and opportunities were presented. The contribution of this paper relies on the guide to new researches about trends and problems in the application of mutation testing of concurrent programs.

References

Almasi, G. S., & Gottlieb, A. (1989). *Highly parallel computing*. Benjamin-Cummings Publishing Co., Inc.

Andrews, G. (2001). *Foundations of Multithreaded, Parallel, and Distributed Programming*. Addison-Wesley.

Bradbury, J. S., Cordy, J. R., & Dingel, J. (2006). Mutation operators for concurrent Java (J2SE 5.0). In *Proceedings of the second workshop on mutation analysis* (pp. 11–20). IEEE.

Carver, R. (1993). Mutation-based testing of concurrent programs. In *Proceedings of test conference* (pp. 845–853).

Delamaro, M., Maldonado, J., Pezzè, M., & Vincenzi, A. (2001). Mutant operators for testing concurrent Java programs. In *Xv simpósio brasileiro de es.*

Delamaro, M. E. (2004). Using instrumentation to reproduce the execution of Java concurrent programs. In *Simpósio brasileiro de qualidade de software.*

Farchi, E., Nir, Y., & Ur, S. (2003). Concurrent bug patterns and how to test them. In *Proceedings of the 17th international symposium on parallel and distributed processing.* IEEE.

Ghosh, S. (2002). Towards measurement of testability of concurrent object-oriented programs using fault insertion: a preliminary investigation. In *Second ieee international workshop on source code analysis and manipulation* (pp. 17–25).

Giacometti, C., Souza, S. R. S., & Souza, P. S. L. (2003). Teste de mutação para a validação de aplicações concorrentes usando PVM. In *Revista eletrônica de iniciação científica* (Vol. 2).

Gligoric, M., Jagannath, V., & Marinov, D. (2010). Mutmut: Efficient exploration for mutation testing of multithreaded code. In *Proceedings of the 2010 third international conference on software testing, verification and validation* (pp. 55–64). Washington, DC, USA: IEEE.

Gligoric, M., Zhang, L., Pereira, C., & Pokam, G. (2013). Selective mutation testing for concurrent code. In *Proceedings of the 2013 international symposium on software testing and analysis* (pp. 224–234).

Hausen, A. C., Vergílio, S. R., Souza, S. R. S., Souza, P. S. L., & Simão, A. S. (2007). A tool for structural testing of MPI programs. In *Ieee latin-american testworkshop - latw.* IEEE.

Hunter, C., & Strooper, P. (2001). Systematically deriving partial oracles for testing concurrent programs. In *Proceedings 24th australian computer science conference. acsc 2001* (p. 83-91).

Jagannath, V., Gligoric, M., Lauterburg, S., Marinov, D., & Agha, G. (2010). Mutation operators for actor systems. In *Proceedings of the 2010 third international conference on software testing, verification, and validation workshops* (pp. 157–162).

Kusano, M., & Wang, C. (2013). Ccmutator: A mutation generator for concurrency constructs in multithreaded c/c++ applications. In *Ase* (p. 722-725).

Lei, J., & Carver, R. H. (2010). A stateful approach to testing monitors in multithreaded programs. *9th IEEE International Symposium on High-Assurance Systems Engineering*, *00*, 54-63.

Melo, S. M., Souza, S. R. S., Silva, R. A., & Souza, P. (2015). Concurrent software testing in practice: A catalog of tools. In *6th international workshop on automating test case design, selection and evaluation* (pp. 31–40).

Offutt, A. J., Voas, J., & Payne, J. (1996). *Mutation operators for Ada* (Tech. Rep.). George Mason University.

Schuler, D., & Zeller, A. (2009). Javalanche: Efficient mutation testing for java. In *Esec/fse '09* (pp. 297–298). New York, NY, USA: ACM.

Silva, R. A. (2013). *Mutation testing applied to concurrent programs in MPI* (MSc Thesis). ICMC, Instituto de Computação e Matemática Computacional, USP.

Silva, R. A. (2018). *Search based software testing for the generation of synchronization sequences for mutation testing of concurrent programs* (PhD Thesis). Institute of Mathematics and Computer Sciences University of São Paulo.

Silva, R. A., Souza, S. R. S., & Souza, P. S. L. (2012a). Deterministic execution of concurrent programs during the mutation testing. In *6th brazilian workshop on systematic and automated software testing.*

Silva, R. A., Souza, S. R. S., & Souza, P. S. L. (2012b). Mutation testing for concurrent programs in MPI. In *13th latin american test workshop* (pp. 69–74).

Silva-Barradas, S. (1998). *Mutation analysis of concurrent software* (PhD Dissertation). Dottorato di Ricerca in Ingegneria Informatica e Automatica, Poli. di Milano.

SIR. (2018). *Software-artifact infrastructure repository.* Retrieved 2018-02-20, from `http://sir.unl.edu`

Souza, S. R. S., Brito, M. A. S., Silva, R. A., Souza, P. S. L., & Zaluska, E. (2011). Research in concurrent software testing: A systematic review. In *Proceedings of the workshop on parallel and distributed systems: Testing, analysis, and debugging.*

Souza, S. R. S., Souza, P. S. L., Brito, M. A. S., Simao, A. S., & Zaluska, E. J. (2015). Empirical evaluation of a new composite approach to the coverage criteria and reachability testing of concurrent programs. *Softw. Test. Verif. Reliab.*, *25*(3), 310–332.

Tai, K., Carver, R., & Obaid, E. (1989). Deterministic execution debugging of concurrent Ada programs. In *Proceedings of the computer software and applications conference* (pp. 102–109).

Wu, G., & Kaiser, L. (2011). Constructing subtle concurrency bugs using synchronization-centric second-order mutation operators. In *Seke* (p. 244-249).

Estudo preliminar sobre os impactos do desenvolvimento orientado a testes no processo de revisão de código

Altieres de Matos[1], Larissa Bonifácio Roder[1], Luciane Baldo Nicolodi[1,2], Reginaldo Ré[3] e Marco Aurélio Graciotto Silva[1,3]

[1]Programa de Pós-Graduação em Informática – PPGI
Universidade Tecnológica Federal do Paraná (UTFPR)
Cornélio Procópio – PR – Brasil
{altitdb,larissaroder,baldoluciane}@gmail.com

[2]Programa de Pós-Graduação em Ciência da Computação – PCC
Universidade Estadual de Maringá (UEM)
Maringá – PR – Brasil

[3]Departamento Acadêmico de Computação
Universidade Tecnológica Federal do Paraná (UTFPR)
Campo Mourão – PR – Brasil
{reginaldo,magsilva}@utfpr.edu.br

Abstract. *Even using agile methods and practices like TDD and code review, many problems are encountered during coding. Code review and TDD focus on the quality of software, but their relationship is not sufficiently understood. The objective of this study was to investigate problems encountered by the code review process in software developed using TDD. A survey was conducted with software developers and architects of a medium-sized Brazilian company, seeking to characterize problems found by code reviews for code developed using TDD. The results characterize the main problems with respect to frequency, severity, time, effort, and techniques that could help to address such problems. With this study, it was possible to list important problems in code developed using TDD, presenting opportunities for TDD improvement to avoid such problems and for code review techniques to address them properly.*

Resumo. *Mesmo utilizando métodos e práticas ágeis como TDD e revisão de código, muitos problemas são encontrados durante a codificação. Revisão de código e TDD focam na qualidade do código fonte do software, mas a relação entre essas técnicas não é suficientemente compreendida. O objetivo desse estudo foi investigar problemas encontrados pelo processo de revisão de códigos em software desenvolvido com TDD. Foi realizado um survey com desenvolvedores e arquitetos de uma indústria de software nacional de médio porte, buscando a caracterização dos problemas encontrados por revisões de códigos desenvolvidos com TDD. Os resultados identificaram e caracterizaram os principais problemas quanto à frequência, criticidade, tempo, esforço e técnicas ou ferramentas que podem auxiliar no tratamento de tais problemas. Com a condução desse estudo foi possível elencar problemas importantes em códigos desenvolvidos com TDD, apresentando oportunidades para aprimoramento do TDD para evitar tais problemas e de técnicas de revisão de código para tratá-los adequadamente.*

1. Introdução

O desenvolvimento orientado a testes (TDD) é uma técnica iterativa de projeto e desenvolvimento de software organizada em três estágios: (i) definição de casos de teste relativos à funcionalidade, antes mesmo da implementação da funcionalidade em si; (ii) implementação do código fonte necessário para que o caso de teste seja executado com sucesso; e (iii) refatoração do código fonte, melhorando o design da solução sem alterar a funcionalidade que foi implementada [Beck 2001]. Quando um defeito de software é encontrado durante a utilização do TDD, casos de teste de unidade são adicionados ao pacote antes da correção feitas [Beck 2001]. Além disso, os casos de testes elaborados durante o TDD são definidos para satisfazer especificações, definindo uma medida de cobertura e forçando a indicar circunstâncias quanto à escrita do código da aplicação e dos casos de teste [Pachulski Camara e Graciotto Silva 2016].

No entanto, mesmo com a utilização de TDD, existem problemas no desenvolvimento de software [Rafique e Misic 2013]. Por exemplo, problemas relacionados ao uso da orientação a objetos, classes estruturadas de forma inadequada e que não possuem cobertura de testes prejudicam a qualidade do código, onerando o tempo necessário para correção de defeitos [George e Williams 2003]. Todos os problemas citados anteriormente são pontuais e comuns no desenvolvimento de software, sendo geralmente encontrados durante a sessões de revisão de código (Code Review). De fato, revisão de código pode ser complementar ao uso de TDD, permitindo a detecção de características indesejáveis, determinando oportunidades para refatoração do código. Desta forma, com a técnica de TDD focada em testes de unidade, testando pequenas partes, e a técnica de revisão de código centrada na análise tipicamente parte por parte do código escrito, em conjunto essas técnicas podem reduzir a ocorrência dos problemas previamente citados [George e Williams 2003]. Considerando um cenário comum de utilização de revisão de código, posterior a iterações de TDD e imediatamente anterior à integração do código resultante no código final do software em desenvolvimento, revisores de código devem considerar cuidadosamente os impactos dos problemas e mudanças solicitadas, e participar de reuniões com os demais desenvolvedores para estabelecer estratégias para adequar o código e prevenir erros similares nas próximas iterações de TDD e sessões de revisão de código [McIntosh et al. 2014].

Em estudos realizados anteriormente, percebeu-se uma forte ligação entre revisão de código e TDD, pois ambos focam na qualidade do desenvolvimento do código fonte do software [George e Williams 2003, McIntosh et al. 2014]. Observa-se também que a revisão de código é mais eficiente evolutivamente do que funcionalmente e que sua importância é significativa para o usuário final [Mäntylä e Lassenius 2009]. Nesse cenário, a adoção de atividades e critérios de qualidade são importantes para melhorar a eficiência e eficácia do uso em conjunto dessas técnicas. Dessa forma, a motivação deste trabalho é gerar insumos para trabalhos futuros quanto a essa integração, possibilitando a melhoria das técnicas e processos de empresas que trabalham com desenvolvimento de software e que utilizam técnicas de TDD e revisão de código. Logo, o objetivo deste estudo é investigar quais são os maiores problemas encontrados pelo processo de revisão de código quanto ao código de software desenvolvido com TDD, agregando assim nos estudos sobre TDD a partir das principais falhas encontradas pelo revisão de código.

O método de pesquisa utilizado foi o levantamento (*survey*). As questões aborda-

ram a identificação e caracterização quanto à criticidade e esforço dispendido em revisão de código, além de atividades utilizadas para abordar os principais problemas encontrados nos códigos revisados. O público alvo foi composto por desenvolvedores e arquitetos de software que realizam revisões de código em empresas que desenvolvem código usando TDD, sendo assim pessoas chaves em relação à qualidade do desenvolvimento do código fonte.

O restante deste artigo está organizado da seguinte forma. Na Seção 2 são apresentados os principais tipos de problemas identificados em revisões de código, conforme relatados na literatura e pela prática na indústria. Na Seção 3 são descritas as características do *survey*. Na Seção 4 são apresentados e discutidos os resultados, relacionando-os com os tipos de problema apresentados na Seção 2. Trabalhos relacionados são tratados na Seção 5. Na Seção 6 elencam-se as principais ameaças de validade deste estudo. Considerações finais e trabalhos futuros são descritos na Seção 7.

2. Revisão de código

A prática de revisão de código consiste no ato de inspecionar o código fonte em busca de problemas relacionados a qualidade interna do software [Mäntylä e Lassenius 2009]. Tipicamente, durante esse ato, um revisor inspeciona o código fonte de outro membro do time de desenvolvimento, linha a linha [Kalyan et al. 2016]. revisão de código tem sido realizada e pesquisada desde 1970 e, desde então, se tornou uma prática popular no desenvolvimento de software. Com o passar do tempo, a revisão de código vem se adaptando a métodos de desenvolvimento moderno e ágil [Kalyan et al. 2016]. Entretanto, atualmente não existe uma lista de problemas encontrados durante a revisão de código especificamente em cenários em que se emprega TDD. Para fins deste estudo, foi compilada a seguinte lista dos principais tipos de problemas encontrados nesse contexto, considerando a literatura científica e a prática da indústria:

- **P01: Código fonte com baixa cobertura de testes automáticos.** A cobertura de código também é usada como uma medida de qualidade do produto, na qual um caso de teste realiza a verificação e validação do código fonte de produção [Hemmati 2015].
- **P02: Código fonte com code smells.** Geralmente não são defeitos. Eles não são tecnicamente incorretos e podem não impedir o funcionamento do software. Em vez disso, eles indicam fraquezas no software que podem estar atrasando o desenvolvimento ou aumentando o risco de erros no futuro [Tufano et al. 2015].
- **P03: Código fonte com complexidade ciclomática alta.** A complexidade ciclomática é uma métrica de software que afere a quantidade de caminhos de execução independentes a partir de um código fonte [McCabe 1976].
- **P04: Código fonte com defeitos.** O termo defeito refere-se geralmente a algum problema com o software, quer com o seu comportamento externo quer com as suas características internas [Card 1990].
- **P05: Código fonte com design ruim.** O design geralmente é ruim quando seus objetos não são bem definidos, seus métodos ou procedimentos são extensos e complexos, e o código fonte não favorece a manutenibilidade [Fowler 1999].
- **P06: Código fonte com duplicação de código.** Significa que dois ou mais fragmentos de código fonte são idênticos ou muito semelhantes, particularmente em sua estrutura [Toomim e Graham 2004].

- **P07: Código fonte com layout errado.** Está relacionado à organização do código fonte, incluindo o uso de espaços em branco, agrupamento de código, linhas em branco, alinhamento, indentação, entre outros [ISO et al. 2010].
- **P08: Código fonte com testes automáticos errados.** Compreende-se como teste automático errado um caso de teste executado em condições especificas e cujo resultado não é igual àquele definido na especificação do software [Ma et al. 2015].
- **P09: Código fonte fora do padrão da empresa.** Conjunto de requisitos que estabelecem uma abordagem disciplinada e uniforme para assegurar a consistência do código fonte do software, ou seja, um padrão ou forma uniforme para a organização do código [ISO et al. 2010].
- **P10: Código fonte não atende o requisito.** Um requisito é uma condição ou capacidade que deve ser cumprida ou possuída por um sistema, componente do sistema, produto ou serviço para satisfazer um acordo, padrão, especificação ou outros documentos formalmente impostos [ISO et al. 2010].
- **P11: Código fonte sem testes automáticos.** Quando um caso de teste exercita um trecho de código, dizemos que esse trecho de código é coberto de testes. No entanto, existem cenários em que nenhum trecho de código é exercitado por casos de teste [Aniche et al. 2015].

3. Método

Neste estudo utilizamos o método *survey*, que compreende em coletar informações de um grupo de pessoas por amostragem de indivíduos a partir de uma grande população. Uma vez que o método baseia-se na amostragem, é tipicamente realizado primeiro o planejamento e depois a realização do estudo de acordo com o plano. Desta maneira, a realização pode ser dividida em um número de etapas sequenciais [Tichy e Padberg 2007, Linaker et al. 2015].

As seguintes características de problemas encontrados em revisões de código foram consideradas neste *survey*: frequência, criticidade, tempo, esforço e característica. A população avaliada consistiu de desenvolvedores e arquitetos de software que trabalham com desenvolvimento de software e com TDD. A amostragem foi não probabilística, classificada como amostra acidental, restringindo-se a apenas uma empresa. Isso deveu-se ao fato de que nela existia acompanhamento durante as iterações de desenvolvimento (*sprints*) para confirmar que o código era produzido utilizando TDD, permitindo a análise da questão tratada neste artigo.

O questionário foi desenvolvido na plataforma *Google Forms* e foi enviado por e-mail. Foi realizado um estudo prévio sobre TDD e revisão de código, a partir dos quais, e considerando as variáveis previamente estabelecidas, foi preparado o questionário a ser utilizado no survey. Para aferir a qualidade do instrumento construído, foram realizados a evolução do questionário por meio de pesquisador mais experiente e *surveys* pilotos. No primeiro estudo piloto, enviamos o questionário para uma amostra reduzida de pessoas, de empresa distinta daquela alvo do questionário final, com intuito de encontrar possíveis falhas nas perguntas elencadas. Após receber as respostas, os dados foram analisados e algumas perguntas foram ajustadas. Após a evolução do questionário, o mesmo foi enviado novamente para uma nova amostra, diferente da anterior e pertencente a um setor específico da empresa alvo (e que não participou como sujeito do questionário final), com propósito de garantir que o questionário atendia os propósitos esperados. Desta vez, após

análise das respostas e avaliação do questionário, pudemos confirmar que ele estava apto para ser enviado para a amostra da empresa alvo.

O questionário final utilizado na coleta dos dados foi dividido em 7 passos e 9 questões, além do Termo de Consentimento Livre e Esclarecido (TCLE) de forma a prover segurança aos respondentes[1].

4. Resultados

O questionário foi divulgado via e-mail para o grupo de desenvolvedores e arquitetos de empresa de software nacional de médio porte, com cerca de 450 profissionais que atuam no desenvolvimento de software para mercado financeiro, abrangendo 101 pessoas. Após disponibilizado por 2 dias, 32 respostas foram obtidas[2]. Para análise, retirados dados que não estavam de acordo com o esperado, como por exemplo pessoas que não são parte da população almejada.

As primeiras duas perguntas estão relacionadas à função desempenhada na empresa e à execução de atividades de revisão de código. Dos 32 respondentes, 25 eram desenvolvedores e 7 são arquitetos, conforme apresentado a seguir:

- **Q1. Qual é sua função?** O questionário contou com 25 respostas de desenvolvedores, correspondendo a 78,1% da amostra, e 7 respostas de arquitetos, correspondendo a 21,9%. Embora existisse a opção de informar outras funções, nenhum dos 32 sujeitos afirmou exercer funções distintas.
- **Q2. Você faz revisão de código?** Dos sujeitos que responderam o questionário, 25 afirmaram realizar revisão de código, sendo 7 arquitetos e 18 desenvolvedores, e correspondem a 78,1% da amostra. Apenas 2 sujeitos não realizam revisão de código, composta apenas por desenvolvedores, e correspondem a 6,3% da população. Por fim, 5 sujeitos não quiseram informar ou não sabem se realizam revisão de código, correspondendo a 15,6% da população e sendo composta apenas por desenvolvedores. Esses 7 sujeitos, por estarem fora do perfil da população investigada, foram excluídos do restante do estudo.

Em relação aos problemas encontrados pelas revisões de código, as questões Q3 e Q4 trataram da frequência com que eles eram encontrados e se existiam problemas não mencionados na lista apresentada na Seção 2.

- **Q3. Quais problemas são frequentemente encontrados durante a revisão de código?** Para cada problema elencado na revisão de código, o sujeito respondente poderia escolher, segundo a escala *Likert*, as opções: nunca (N), raramente (R), algumas vezes (AV), frequentemente (F) e sempre (S). Sumarizamos as respostas na Tabela 1, relacionando a quantidade de sujeitos que escolheram determinada opção com o problema relacionado. Após realizar a sumarização das respostas, realizamos o cálculo da Moda (M) para cada problema elencado. Desta forma, os problemas que são encontrados com mais frequência são código fonte com baixa cobertura de testes automáticos (P01) e sem testes automáticos (P11). De outro

[1]O questionário completo está disponível em `https://github.com/altitdb/utfpr/blob/master/eres2018/questions.pdf`.

[2]Os dados coletados estão disponíveis em `https://github.com/altitdb/utfpr/blob/master/eres2018/answers.xlsx`.

	N	R	AV	F	S	M
P01: Com baixa cobertura de testes automáticos	0	3	10	11	1	4
P02: Com *code smells*	0	1	11	11	2	3
P03: Com complexidade ciclomática alta	0	1	14	10	0	3
P04: Com defeitos	1	8	16	0	0	3
P05: Com *design* ruim	0	2	15	6	2	3
P06: Com duplicação de código	0	8	11	3	3	3
P07: Com *layout* errado	2	13	7	3	0	2
P08: Com testes automáticos errados	2	9	11	3	0	3
P09: Fora do padrão da empresa	0	14	8	3	0	2
P10: Não atende o requisito	4	13	8	0	0	2
P11: Sem testes automáticos	0	4	8	13	0	4

Tabela 1. Relação de problemas encontrados em revisões de código e respectiva frequência. As colunas correspondem à: nunca (N), raramente (R), algumas vezes (AV), frequentemente (F), sempre (S) e cálculo da moda (M).

lado, os problemas que possuem menor frequência são código fonte com *layout* errado (P07), fora do padrão da empresa (P09) e que não atende o requisito (P10).

- **Q4. Existem problemas que são frequentemente encontrados durante a revisão de código e não foram citados? Cite esses problemas, descrevendo-o brevemente e informando quão frequentemente eles são encontrados.** Não obtivemos respostas para essa pergunta. Entendemos que as alternativas supriram as escolhas da amostra.

Identificados os problemas mais frequentes, buscou-se a seguir a caracterização deles, em especial quanto à gravidade, esforço necessário para a identificação e para a correção. Na Tabela 2, são relacionados os problemas mais graves (Q5) e os que demandaram mais tempo para identificação (Q6) e para correção (Q7).

	Q5	Q6	Q7
P01: Com baixa cobertura de testes automáticos	9	5	5
P02: Com *code smells*	14	6	5
P03: Com complexidade ciclomática alta	11	6	15
P04: Com defeitos	14	13	6
P05: Com *design* ruim	10	5	16
P06: Com duplicação de código	8	5	5
P07: Com *layout* errado	3	3	4
P08: Com testes automáticos errados	7	13	9
P09: Fora do padrão da empresa	1	3	1
P10: Não atende o requisito	11	12	12
P11: Sem testes automáticos	13	2	7

Tabela 2. Respostas das questões Q5, Q6 e Q7.

- **Q5. Quais são os problemas mais graves encontrados durante a revisão de código?** Com os resultados obtidos podemos observar que os problemas mais críticos são código fonte com *code smells* (P02) e com defeito (P04). O problema menos crítico observado é o código fonte fora do padrão da empresa (P09).

- **Q6. Quais os problemas levam mais tempo para serem encontrados na revisão de código?** De acordo com os resultados, os problemas que levam mais tempo para serem encontrados são código fonte com defeitos (P04) e com testes automáticos errados (P08). Dentre os problemas, o último a aparecer é o código fonte sem testes automáticos (P11).
- **Q7. Quais itens reportados na revisão de código demandam mais esforço para correção?** Os resultados demonstraram que o problema que demanda mais esforço para correção é o código fonte com *design* ruim (P05). De todos os problemas relacionados, o código fonte fora do padrão da empresa (P09) apareceu em último.

Finalmente, foram investigados fatores relacionados às respostas anteriores que envolvam a aplicação de técnicas e ferramentas de revisão de código, além de outros elementos considerados pertinentes à pesquisa quanto à amostra da população considerada:

- **Q8. Quais técnicas, práticas, *plugins* ou ferramentas você utiliza para reduzir problemas na revisão de código?** As ferramentas que tiveram destaques foram *SonarQube* e *SonarLint*, citadas por 7 sujeitos cada. Além disso, outra ferramenta mencionada foi o Comparador de Arquivos, citado por 3 sujeitos. A experiência e ferramenta Stash foram citadas 2 vezes por cada sujeito. Foram citadas apenas 1 vez as seguintes técnicas, práticas, *plugins* e ferramentas: Ferramentas próprias (não foram detalhadas pelos respondentes), *Test Driven Development*, *TestCoverage*, Ferramenta de Versionamento (*WinMerge*), *Pair Programming*, *Bit-Bucket Code Review*, Bom Senso e *Checklist*.
- **Q9. Caso você tenha alguma sugestão ou queira complementar suas respostas com algo que o questionário não abordou fique a vontade para descrever neste espaço.** Para esta pergunta, os respondentes levantaram alguns pontos que podem ser adicionados a pesquisa. São eles: (i) Nível de conhecimento das técnicas, práticas, *plugins* e/ou ferramentas utilizadas no desenvolvimento de software e (ii) Realizar classificação direcionada a código novo e antigo, com intenção de descobrir os problemas mais próximos para cada realidade.

5. Trabalhos Relacionados

Embora não sejam relatados na literatura estudos que relacionam diretamente atividades de revisão de código desenvolvido com TDD, pode-se identificar trabalhos que tratam desses elementos individualmente e que permitam relação com este trabalho: problemas encontrados em revisões de código no contexto de métodos ágeis e problemas em TDD.

Quanto à revisão de código e como ela pode ser melhorada, Bernhart *et al.* (2010) apresentaram um método baseado na evolução do uso de revisão de código junto com uma ferramenta de revisão de código. Usando uma abordagem contínua para revisão de código, a sobrecarga de revisão pode ser reduzida e a eficácia e a aplicabilidade em ambientes ágeis podem ser melhoradas [Bernhart et al. 2010].

Considerando outra prática típica de métodos ágeis, a programação em pares, Swamidurai *et al*. (2014) realizaram um estudo sobre o uso conjunto dessa com técnicas mais leves de revisão de código, mais especificamente a revisão por pares. Eles demonstraram a maior eficácia da revisão por pares em comparação com a programação

em pares no contexto do TDD. Em suma, eles forneceram evidências de que programas de qualidade igual podem ser produzidos com menor custo usando revisão por pares [Swamidurai et al. 2014].

No âmbito de TDD, Fucci *et al.* (2014) comentam sobre a limitação de estudos empíricos sobre TDD e se desenvolvedores estão seguindo o ciclo *test-code-refactor*. De acordo com eles, nenhuma das pesquisas até aquele momento incluíam a conformidade do processo de TDD como parte fundamental da análise. Desta forma, eless analisaram o impacto da conformidade no processo nos efeitos alegados pelo TDD na qualidade externa, produtividade dos desenvolvedores e qualidade de testes. Com base em uma pequena amostra, os autores demonstram preocupações sobre como TDD é interpretado, além de questionarem a viabilidade econômica da adoção estrita do TDD [Fucci et al. 2014].

Em continuidade ao trabalho anterior, Fucci *et al.* (2015) aprimoraram um modelo para execução do TDD. Esse modelo tinha como objetivo realçar dois fatores adicionais: habilidades de programação e testes de unidade. Desta forma, seria possível incluir tais habilidades em um modelo que representasse a relação que o esforço da atividade teste tem com a produtividade do desenvolvedor e a qualidade externa. Os dados coletados em ambiente acadêmico foram utilizados para avaliar a relação entre esforço de teste, qualidade externa e produtividade [Fucci et al. 2015].

6. Ameaças a validade

Em relação à validade interna, o instrumento de medida foi avaliado por um especialista e por dois estudos pilotos, com sujeitos compatíveis com a população investigada. Também foi possível identificar, pelas questões Q1 e Q2, sujeitos que responderam o questionário, mas que não eram da população de interesse. As respostas relativas a esses sujeitos foram removidas. Embora a participação fosse anônima, foi requerido que cada sujeito respondesse o questionário após autenticação com uma conta no serviço *Gmail*, garantindo assim que cada sujeito respondesse apenas um questionário. Entretanto, embora diversas ameaças quanto ao instrumento tenham sido abordadas, constatou-se uma limitação quanto ao tamanho da amostra e também a a diferença do tamanho das amostras de desenvolvedores e arquitetos, restringindo a comparação entre os tipos de papéis.

Em relação à ameaças de validade externa, observa-se que o estudo engloba apenas uma empresa de engenharia de software de médio porte. Desta forma, não é possível generalizar os resultados obtidos. Não obstante, novos estudos podem ser realizados, contemplando outras empresas, de diferentes tipo, portes e regiões do Brasil e do mundo.

Em relação à validade da conclusão, avaliou-se a confiabilidade das respostas. Em relação à questão Q3, calculou-se o *Cronbach's Alpha*, resultando no valor de $\alpha = 0{,}7606$, considerando toda a amostra respondente. Avaliando-se somente os sujeitos desenvolvedores, obteve-se o valor $\alpha = 0{,}7637$. Considerando-se os arquitetos de software, obteve-se o valor $\alpha = 0{,}7346$. Com base nesses resultados obteve-se consistência interna aceitável.

7. Conclusão

O objetivo desse estudo foi investigar problemas que são gerados durante o desenvolvimento de software utilizando TDD e que são encontrados durante o processo de revisão de código. Primeiramente, foram levantados os problemas considerados na revisão de

código por desenvolvedores e arquitetos. Um *survey* foi realizado para uma amostra de 101 pessoas que participam do processo de desenvolvimento de software de uma empresa nacional de médio porte, com foco no mercado financeiro, com 32 respostas.

A partir da análise dos resultados obtidos, foram identificados que os problemas mais frequentes de revisão de código são código fonte com baixa cobertura de testes automáticos e código fonte sem testes automáticos. Os problemas mais críticos são *code smells* e código fonte com defeito. Em relação aos problemas que levam mais tempo para serem encontrados na revisão de código, destacam-se o código fonte com defeitos e o código fonte com testes automáticos errados. Para problemas que afetam o esforço para correção, identificou-se o código fonte com *design* ruim.

Por fim, foi possível analisar as variações de percepção de problemas relacionados a revisão de código de acordo com a característica do sujeito, que nesse estudo contemplou apenas dois grupos, sendo eles desenvolvedores e arquitetos. Com o resultado desse trabalho, foi possível elencar problemas importantes em códigos desenvolvidos com TDD, apresentando oportunidades para aprimoramento do TDD para evitar tais problemas e de técnicas de revisão de código para tratá-los adequadamente. Encorajamos também outros pesquisadores a replicarem este estudo, provendo maior confiabilidade e gerando mais conhecimento para a comunidade.

Referências

Aniche, M. F., Oliva, G. A., e Gerosa, M. A. (2015). Why statically estimate code coverage is so hard? a report of lessons learned. In: *29th Brazilian Symposium on Software Engineering*, p. 185–190. IEEE.

Beck, K. (2001). Aim, fire. *IEEE Software*, 18(5):87–89.

Bernhart, M., Mauczka, A., e Grechenig, T. (2010). Adopting code reviews for agile software development. In: *2010 Agile Conference*, p. 44–47. IEEE.

Card, D. N. (1990). Software quality engineering. *Information and Software Technology*, 32(1):3–10.

Fowler, M. (1999). *Refactoring: Improving the Design of Existing Code*. Addison-Wesley, Boston, MA, USA.

Fucci, D., Turhan, B., e Oivo, M. (2014). Impact of process conformance on the effects of test-driven development. In: *8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, p. 10:1–10:10, New York, NY, USA. ACM.

Fucci, D., Turhan, B., e Oivo, M. (2015). On the effects of programming and testing skills on external quality and productivity in a test-driven development context. In: *19th International Conference on Evaluation and Assessment in Software Engineering*, p. 25:1–25:6, New York, NY, USA. ACM.

George, B. e Williams, L. (2003). An initial investigation of test driven development in industry. In: *18th ACM Symposium on Applied Computing*, p. 1135–1139, New York, NY, USA. ACM.

Hemmati, H. (2015). How effective are code coverage criteria? In: *2015 IEEE International Conference on Software Quality, Reliability and Security (QRS 2015)*, p. 151–156. IEEE.

ISO, IEC, e IEEE (2010). Iso/iec/ieee 24765:2010 – systems and software engineering – vocabulary.

Kalyan, A., Chiam, M., Sun, J., e Manoharan, S. (2016). A collaborative code review platform for GitHub. In: *2016 21st International Conference on Engineering of Complex Computer Systems (ICECCS)*, p. 191–196. IEEE.

Kollanus, S. (2011). Critical issues on test-driven development. In: *12th International Conference on Product-Focused Software Process Improvement*, volume 6759 of *Lecture Notes in Computer Science*, p. 322–336.

Linaker, J., Sulaman, S. M., Maiani de Mello, R., e Martin, H. (2015). Guidelines for conducting surveys in software engineering. techreport, Lund University, Sweden.

Ma, L., Zhang, C., Yu, B., e Sato, H. (2015). An empirical study on effects of code visibility on code coverage of software testing. In: *10th International Workshop on Automation of Software Test (AST 2015)*, p. 80–84. IEEE.

McCabe, T. J. (1976). A complexity measure. *Transactions on Software Engineering*, 2(4):308–320.

McIntosh, S., Kamei, Y., Adams, B., e Hassan, A. E. (2014). The impact of code review coverage and code review participation on software quality: A case study of the Qt, VTK, and ITK projects. In: *11th Working Conference on Mining Software Repositories*, p. 192–201, New York, NY, USA. ACM.

Mäntylä, M. V. e Lassenius, C. (2009). What types of defects are really discovered in code reviews? *Transactions on Software Engineering*, 35(3):430–448.

Pachulski Camara, B. H. e Graciotto Silva, M. A. (2016). A strategy to combine test-driven development and test criteria to improve learning of programming skills. In: *47th ACM Technical Symposium on Computing Science Education*, p. 443–448, New York, NY, USA. ACM.

Rafique, Y. e Misic, V. B. (2013). The effects of test-driven development on external quality and productivity: A meta-analysis. *Transactions on Software Engineering*, 39(6):835–856.

Swamidurai, R., Dennis, B., e Kannan, U. (2014). Investigating the impact of peer code review and pair programming on test-driven development. In: *IEEE SOUTHEASTCON 2014*, p. 1–5. IEEE.

Tichy, W. F. e Padberg, F. (2007). Empirical methods in software engineering research. In: *29th International Conference on Software Engineering (ICSE'07 Companion)*, p. 163–164. IEEE.

Toomim, M., B. A. e Graham, S. L. (2004). Managing duplicated code with linked editing. In: *20th IEEE Symposium on Visual Languages and Human Centric Computing*, p. 173–180. IEEE.

Tufano, M., Palomba, F., Bavota, G., Oliveto, R., Penta, M. D., Lucia, A. D., e Poshyvanyk, D. (2015). When and why your code starts to smell bad. In: *37th International Conference on Software Engineering*, volume 1, p. 403–414, Piscataway, NJ, USA. IEEE.

Anais da 2ª Escola Regional de Engenharia de Software (ERES 2018)
em Dois Vizinhos, PR, Brasil, 22 a 24 de outubro de 2018

Reúso de Software: Do Oportunista ao Sistemático

Wesley K. G. Assunção[1], Willian D. F. Mendonça[1], Silvia R. Vergilio[2]

[1]COTSI – Universidade Tecnológica Federal do Paraná (UTFPR)
CEP 85902-490 – Toledo – PR – Brasil

[2]Dinf – Universidade Federal do Paraná (UFPR)
CP 19097 – Curitiba – PR – Brasil

wesleyk@utfpr.edu.br, williandouglasferrari@gmail.com, silvia@inf.ufpr.br

https://www.overleaf.com/6422976981wfqrbvtkvwzd

Abstract. *Opportunistic reuse, also known as copy-and-paste, is a common practice in industry. This strategy presents short term benefits, however, when the number of clone products grows, technical problems of management, maintenance, and evolution arise. Software Product Line (SPL) is a systematic reuse approach presented as an alternative to deal with such problems. In an SPL, a core of similar artifacts are shared among products, avoiding duplicity and independent development of common parts. In industry, we can see that products variants developed opportunistically are the base for the SPL development, by using a re-engineering process. This paper presents a generic re-engineering process and discusses benefits and difficulties to adopt systematic reuse. We aim at motivating industries to consider adoption of systematic reuse and develop products which are easily maintained, evolved and with better quality.*

Resumo. *O reúso oportunista, também chamado de copia-e-cola, é uma prática comum na indústria. Apesar de esta estratégia apresentar benefícios a curto prazo, quando a quantidade de produtos clonados aumenta, problemas técnicos de gerenciamento, manutenção e evolução surgem. Linha de Produtos de Software (LPS) é uma abordagem de reúso sistemático que se apresenta como alternativa promissora para estes problemas. Em uma LPS, um núcleo de artefatos similares é compartilhado entre os produtos, evitando duplicidades e desenvolvimento independente de partes comuns. Na indústria, observa-se a origem de sistemas desenvolvidos de forma oportunista, que são utilizados como base para o desenvolvimento da LPS, utilizando um processo de reengenharia. Este trabalho apresenta um processo genérico de reengenharia, além disso, discute benefícios e dificuldades para aplicar o reúso sistemático. Deseja-se motivar que indústrias considerem a adoção do reúso sistemático e desenvolvam produtos mais fáceis de manter, evoluir, e com melhor qualidade.*

1. Introdução

Reúso de software é o processo de construir novos produtos de software com base na reutilização de artefatos já existentes, ao invés de construí-los do princípio [Krueger 1992]. O reúso aumenta a qualidade e produtividade de novo software e reduz o custo de desenvolvimento.

Uma prática comum em empresas de desenvolvimento é o *reúso oportunista*, também conhecido como reúso *ad hoc* ou copia-e-cola [Holmes and Walker 2013]. Nesta prática, quando existe a demanda por um novo produto ou nova funcionalidade, artefatos existentes são copiados/clonados e modificados/adaptados para satisfazer os requisitos. O reúso oportunista é uma maneira simples e intuitiva de construir novos produtos, pois não exige muito investimento inicial, e obtém bons resultados em curto período de tempo.

Contudo, a adoção intensiva e não planejada de reúso oportunista pode gerar a necessidade de refatorações constantes e acarretar débitos técnicos, tais como: comportamentos inesperados, violação de restrições, não atendimento de requisitos não-funcionais, estrutura frágil, e software inchado [Kulkarni and Varma 2017]. Neste cenário, a manutenção e evolução de produtos desenvolvidos individualmente com reúso oportunista são tarefas complexas e suscetíveis a erros [Faust and Verhoef 2003].

Para contornar os débitos técnicos relacionados ao reúso oportunista, a abordagem de Linha de Produto de Software (LPS) é uma alternativa já estabelecida e consolidada [Pohl et al. 2005]. A LPS é uma abordagem de reúso sistemático em que uma família de produtos relacionados são desenvolvidos compartilhando partes comuns para atender um segmento de mercado ou domínio específico [Clements and Northrop 2001, Linden et al. 2007]. LPSs incorporam o reúso sistemático no processo de desenvolvimento de software [Galimberti and Wazlawick 2015].

A utilização de produtos existentes para o desenvolvimento de uma LPS é conhecida como uma abordagem extrativa [Krueger 1992]. Nesta abordagem os artefatos dos produtos desenvolvidos independentemente são analisados, organizados, e transformados, por meio de um processo de reengenharia, para alcançar o reúso sistemático. Na literatura é possível observar que a aplicação de uma abordagem extrativa é o caminho mais comum para a adoção de LPSs [Laguna and Crespo 2013, Assunção et al. 2017]. Uma vez que os produtos são migrados para uma LPS, eles não são mais mantidos e evoluídos individualmente, mas como um grupo que compartilha similaridades e implementa variabilidades.

Neste trabalho o objetivo é apresentar um processo genérico de reengenharia e discutir suas características, bem como benefícios e dificuldades que são encontrados durante a condução deste processo para a construção de LPSs.

As contribuições do trabalho são: (i) fornecer para empresas de software uma visão geral sobre uma abordagem extrativa para construção de LPSs; (ii) provocar a discussão e troca de experiências entre academia e empresas em relação à utilização de LPSs; e (iii) motivar e popularizar a adoção na prática de reúso sistemático.

2. Linhas de Produtos de Software

Uma *Linha de Produtos de Software* é um conjunto de sistemas que compartilham um conjunto comum e gerenciável de características (do inglês, *features*) para satisfazer as necessidades de um segmento de mercado ou domínio específicas [Clements and Northrop 2001]. Uma *característica* é um aspecto, qualidade, ou funcionalidade de um software visível ou percebido pelo usuário do sistema [Kang et al. 1990]. Características são os blocos de construção de uma LPS.

Uma LPS é composta por duas partes principais: (i) as *partes comuns* (também conhecidas como similaridades) que são reusadas em todos os produtos, e (ii) as *partes*

variantes (também conhecidas como variabilidades) relacionadas com as características que estão presentes em apenas alguns produtos.

Engenharia de LPS (ELPS) é a atividade responsável pelo desenvolvimento de LPSs. A ELPS explora o desenvolvimento e gerenciamento das similaridades e variabilidades para a instanciação dos produtos [Clements and Northrop 2001]. A ELPS baseada em características é a abordagem de desenvolvimento mais utilizada atualmente [Apel et al. 2013]. A Figura 1 apresenta a estrutura desta abordagem. Existem basicamente duas atividades principais:

- A *engenharia de domínio* (no topo da figura), que tem como objetivo principal a organização e representação dos produtos que compõem o segmento ou domínio em que a LPS está inserida;
- A *engenharia de aplicação* (na base da figura), responsável pelos aspectos de implementação das similaridades e variabilidades, e pela derivação dos produtos reutilizando-se os artefatos desenvolvidos e as características selecionadas.

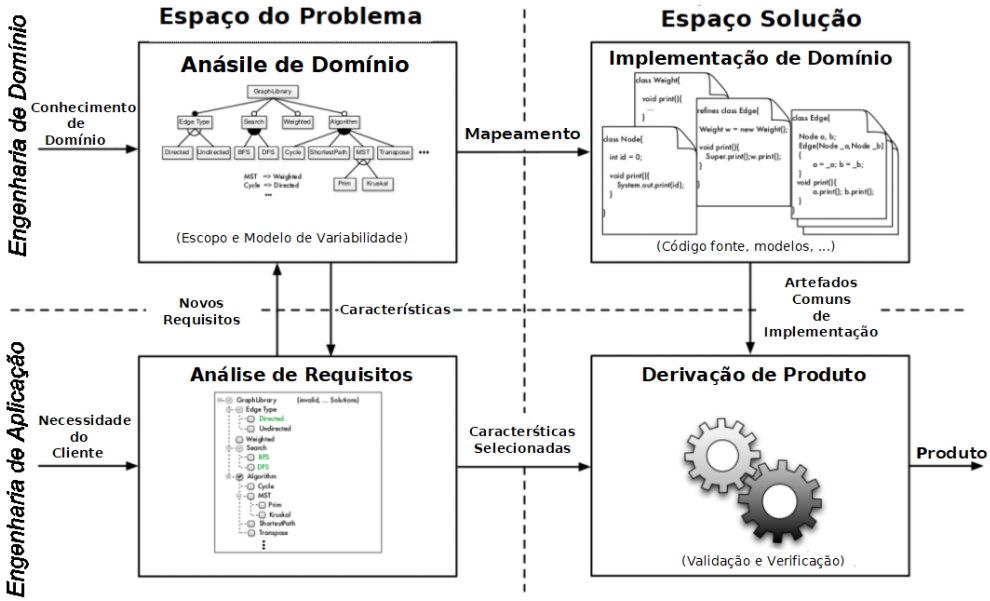


Figura 1. Framework de ELPS, adaptado de [Apel et al. 2013]

Durante as atividades de ELPS, dois artefatos principais são construídos: (i) o *modelo de características*, que representa a organização das similaridades/variabilidades e dá suporte à comunicação dos envolvidos com a LPS; e (ii) a *arquitetura da linha de produtos*, responsável por descrever como as características/funcionalidades são implementadas e quais artefatos são utilizados na instanciação de cada produto.

3. O Processo de Reengenharia

Muitos trabalhos são encontrados sobre a extração de LPSs a partir de produtos variantes. Em um estudo recente foi definido um processo de reengenharia com base nas atividades comuns na extração de LPSs [Assunção et al. 2017]. A Figura 2 apresenta uma visão geral das fases do processo de reengenharia. Do lado esquerdo da figura apresentam-se

as variantes desenvolvidas com reúso oportunista. A linha sólida apresenta o processo completo de reengenharia, produzindo uma LPS, apresentada do lado direito da figura. As fases do processo, apresentadas por linhas tracejadas, são descritas a seguir.

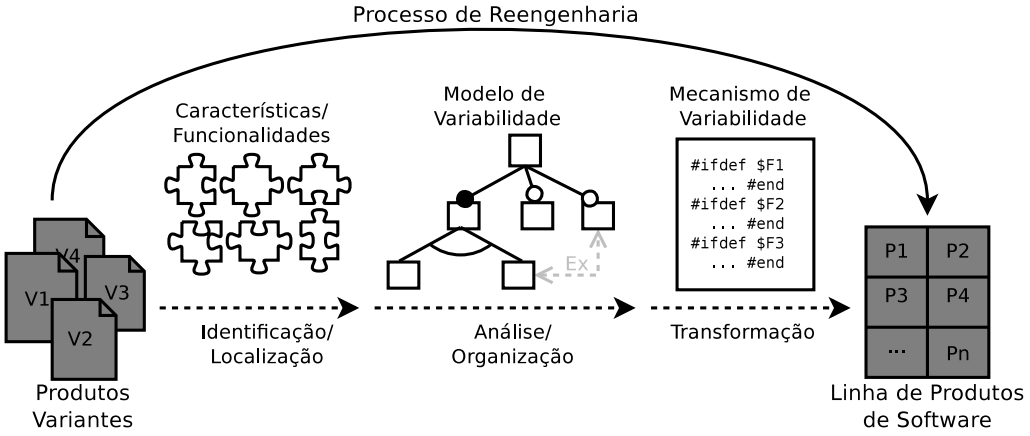


Figura 2. Processo de Reengenharia, adaptado de [Assunção et al. 2017]

O processo de reengenharia é formado por três fases genéricas: (i) *Identificação/Localização* é responsável por identificar as características/funcionalidades similares e variantes espalhadas pelos produtos, e ainda, fazer a rastreabilidade entre características e os artefatos de implementação. (2) *Análise/Organização* é responsável por definir como as características são organizadas, para a criação de um modelo de características. (3) *Transformação* tem como foco a construção de artefatos de LPS, como por exemplo a arquitetura da linha de produtos, e a alteração dos artefatos de implementação para incluir mecanismo de variabilidades, tal como diretivas `#ifdef`, possibilitando a posterior instanciação de produtos.

4. Discussões

Exitem relatos da adoção de LPS por um processo de reengenharia na indústria. Ommering apresentou a aplicação na Philips [van Ommering 2005], já Jansen et al. demonstram a aplicação da reengenharia em organizações de pequeno e médio porte [Jansen et al. 2008]. Galimberti e Wazlawick reportam a utilização de LPS com foco em internacionalização de produtos [Galimberti and Wazlawick 2015]. Contudo, estes trabalhos não apresentam explicitamente vantagens e desvantagens observadas durante a condução dos estudos. A partir de um estudo prévio [Assunção et al. 2017], e com base na literatura recente, foram identificados benefícios da adoção de LPS e dificuldades na condução do processo de reengenharia.

Em geral, os benefícios da adoção de reúso sistemático são:

- *Facilidade para desenvolver novos produtos*: uma vez que as características dos sistemas estão organizadas em artefatos reutilizáveis, a criação de novos produtos, aumentando o portfólio das empresas, é uma atividade simples, pois compreende somente a seleção de uma nova configuração ou o desenvolvimento de apenas uma nova funcionalidade, não requerendo adaptações *ad hoc*;
- *Facilidade para manter os produtos*: para os produtos que reutilizam uma base comum de artefatos, a manutenção das características similares são executadas

somente uma vez, e automaticamente serão propagadas para todos os produtos variantes. Já no reúso oportunista, todos os produtos deveriam ser mantidos individualmente;
- *Facilidade para evoluir o sistema*: atividades de evolução, como por exemplo a troca de plataforma *desktop* para *web*, pode ser conduzida mais facilmente, já que uma vez migradas as similaridades, todos os produtos já terão o novo comportamento, restando apenas a evolução das variabilidades.
- *Melhor comunicação entre os envolvidos*: artefatos gerados durante o processo de reengenharia, tal como o diagrama de características e a arquitetura da linha de produtos, oferecem aos envolvidos com o sistema uma visão ampla e geral de como os produtos são organizados, o que implementam, quais as funcionalidades mais reutilizadas, etc. Isso auxilia a comunicação e tomada de decisões tanto técnicas quanto gerenciais;
- *Geração de casos de teste melhores*: através da visão geral do sistema, casos de teste que exercitam interações específicas entre características, ou que preveem possíveis "gargalos" de funcionamento, podem ser definidos, aumentando a qualidade do produto.

Por outro lado, através da observação da literatura e relatos de experiências, pode-se verificar que existem dificuldades em conduzir o processo de reengenharia:

- *Falta de ferramentas*: muitos trabalhos apresentam ferramentas com propósito muito específico para a reengenharia, o que impossibilita a sua aplicação em outros cenários. Sem uma ferramenta para automatizar e dar suporte ao processo, empresas necessitam muito esforço humano, algumas vezes inviabilizando a adoção de LPS;
- *Mudança de paradigma nas empresas*: empresas que construíram produtos sem considerar reúso sistemático durante o ciclo de desenvolvimento podem necessitar de uma maior conscientização para que essa nova perspectiva seja bem-sucedida. Em situações em que a reengenharia é um processo complexo devido ao domínio do sistema, a equipe deve estar coesa para alcançar o objetivo;
- *Poucos casos de sucesso*: apesar da existência de relatos na literatura, empresas ainda necessitam de exemplos mais concretos para decidir sobre os benefícios da adoção de LPS. A maior parte da literatura refere-se à empresas de grande porte, mas em geral empresas menores, que são mais numerosas, não são consideradas.

5. Considerações Finais

Este trabalho apresenta um processo genérico de reengenharia de produtos existentes para uma LPS, permitindo que empresas resolvam problemas técnicos associados ao reúso oportunista. A abordagem extrativa é uma alternativa promissora para sistematizar o reúso.

Os benefícios da adoção do reúso sistemático são a facilidade para desenvolver novos produtos por meio do reúso, executar atividades de manutenção e evolução, possibilitar melhor comunicação e tomada de decisões, e apoiar a derivação de casos de teste melhores. Entretanto, existe a carência de ferramentas mais genéricas para apoiar o processo de reengenharia e, além disso, é necessário considerar o comprometimento dos envolvidos no processo de adoção do reúso sistemático. Outro fator é a existência

de poucos casos de sucesso apresentados com dados concretos, para motivar e servir de exemplos para empresas que visam à adoção de LPSs.

Como trabalho futuro deseja-se conduzir o processo de reengenharia em um estudo de caso. Todas as etapas serão documentadas para criar um conjunto de informações, facilitando-se o entendimento por parte dos interessados em adotar o reúso sistemático.

Referências

Apel, S., Batory, D., Kästner, C., and Saake, G. (2013). *Feature-Oriented Software Product Lines*. Springer.

Assunção, W. K. G., Lopez-Herrejon, R. E., Linsbauer, L., Vergilio, S. R., and Egyed, A. (2017). Reengineering legacy applications into software product lines: A systematic mapping. *Empirical Software Engineering*, pages 1–45.

Clements, P. and Northrop, L. (2001). *Software Product Lines: Practices and Patterns*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Faust, D. and Verhoef, C. (2003). Software product line migration and deployment. *Software: Practice and Experience*, 33(10):933–955.

Galimberti, M. and Wazlawick, R. (2015). Active internationalization of small and medium-sized software enterprises - cases of french software companies. *Journal of Technology Management & Innovation*, 10(4):99–108.

Holmes, R. and Walker, R. J. (2013). Systematizing pragmatic software reuse. *ACM Transactions on Software Engineering and Methodology*, 21(4):20:1–20:44.

Jansen, S., Brinkkemper, S., Hunink, I., and Demir, C. (2008). Pragmatic and opportunistic reuse in innovative start-up companies. *IEEE Software*, 25(6):42–49.

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 University Software Engineering Institute.

Krueger, C. W. (1992). Software reuse. *ACM Computing Surveys*, 24(2):131–183.

Kulkarni, N. and Varma, V. (2017). Perils of opportunistically reusing software module. *Software: Practice and Experience*, 47(7):971–984.

Laguna, M. A. and Crespo, Y. (2013). A systematic mapping study on software product line evolution: From legacy system reengineering to product line refactoring. *Science of Computer Programming*, 78(8):1010–1034. Special section on software evolution, adaptability, and maintenance & Special section on the Brazilian Symposium on Programming Languages.

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-Verlag New York, Inc., Secaucus, NJ, USA.

Pohl, K., Böckle, G., and Linden, F. J. v. d. (2005). *Software Product Line Engineering: Foundations, Principles and Techniques*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

van Ommering, R. (2005). Software reuse in product populations. *IEEE Transactions on Software Engineering*, 31(7):537–550.

Estudo e aplicação da ferramenta sikuli para auxílio em testes funcionais e de usabilidade por meio da automação por imagem

Greici Savoldi, Rafael A. P. Oliveira

Coordenadoria do curso de Engenharia de Software

Universidade Tecnológica Federal do Paraná (UTFPR) - Dois Vizinhos - PR - Brasil

greici.sav@hotmail.com, raoliveira@utfpr.edu.br

Abstract: *This article presents a study of the application of the sikuli tool, of automation by image compared with manual tests of functionality and usability with the objective of reducing test effort in these areas.*

Resumo: *Este artigo apresenta um estudo de aplicação da ferramenta sikuli, de automação por imagem comparado com testes manuais de funcionalidade e usabilidade com objetivo de reduzir esforço de testes nessas áreas.*

1. Introdução

O processo de teste engloba uma série de atividades que devem ser validadas sistematicamente com objetivo de encontrar falhas, revelar defeitos e garantir a qualidade do software. Como essa atividade consome um esforço significativo dentro dos projetos de desenvolvimento de software, muitas empresas optam pela automação do teste reduzindo o tempo, custo e esforço da equipe (FARIAS, 2003).

Automação de teste tem se tornado o foco principal de empresas que buscam inovação para garantir que os testes de regressão e funcionais sejam executados com menor esforço e mais agilidade. Esse processo pode ser realizado por meio de uma ferramenta específica de automação de teste e pode agregar muitos benefícios para a empresa como, por exemplo, a diminuição do tempo, custo e aumento da produtividade (FANTINATO *et al*, 2005). Assim o teste é uma forma de validação que mais vem sendo utilizada na prática, principalmente os testes funcionais e de usabilidade por se basearem em especificações do software e possuírem grande importância (FIDELIS & MARTINS, 2004).

Este estudo apresenta um estudo de caso realizado em ambiente real de indústria com o intuito de verificar a utilização da ferramenta sikuli para auxiliar os teste de usabilidade e funcionalidade no intuito de diminuir o esforço de teste nessas áreas. As contribuições associadas ao presente estudo são: (i) uma avaliação empírica executada dentro de um ambiente real de desenvolvimento e teste que pode ser utilizada como referência para decisões de projeto; (ii) uma discussão acerca do uso de ferramentas de teste visual para os testes funcionais e de usabilidade.

2. Referencial Teórico

A presente seção apresenta uma série de informações técnicas associadas ao entendimento completo do presente estudo.

2.1 Teste automatizado

De acordo com Bernardo e Kon (2008), testes automatizados são programas ou scripts simples que exercitam funcionalidades do sistema sendo testado e fazem verificações automáticas nos efeitos colaterais obtidos. A grande vantagem desta abordagem, é que todos os casos de teste podem ser facilmente e rapidamente repetidos a qualquer momento e com pouco esforço.

Duas das modalidades de teste que mais requerem automatização são os funcionais e testes de aceitação. Teste funcionais são aqueles baseados em especificação, nos quais os casos de teste são modelados pelo testador sem nenhum acesso a estruturas de código. Intimamente ligados aos testes funcionais, estão os testes de aceitação, que são aqueles realizados em versões finais do sistema, por meio de sua interface gráfica e avaliando a aceitação do cliente.

2.2Sikuli

A Sikuli[1] é uma ferramenta que permite a automação de testes de interfaces gráficas por imagem com grande facilidade. É uma ferramenta gratuita e funciona de modo independe de qualquer API (Application Program Interface) e possui uma interface que facilita a utilização

[1] acesse: http://www.sikuli.org/
[2] acesse: https://www.scrum.org/

permitindo que pessoas sem experiência possam criar scripts em questão de minutos (Costa, 2015).

De acordo com Martins (2015), o Sikuli é uma ferramenta de automação de teste híbrida, na qual um testador pode trabalhar tanto com GUI, como com scripts de usuário. A ferramenta interage com a tela a partir da captura de imagens ou regiões, definidas no script.

2.3 Perfil dos Colaboradores

Foram selecionados 6 colaboradores entre testadores, automatizadores e desenvolvedores para realizar o desenvolvimento da pesquisa. Todos possuem graduação/especialização em engenharia de software e estão a um tempo considerável na empresa, entre 4 e 7 anos.

3. Metodologia

O presente trabalho visa avaliar a aplicação da ferramenta sikuli como auxílio para diminuir o esforço de teste nos testes repetitivos de usabilidade/funcionalidade, criando testes que utilizem o mínimo de código possível, permitindo trabalhar o máximo possível com reconhecimento de imagens do sistema. A escolha desta ferramenta para o estudo se deu por ser uma ferramenta gratuita e de fácil utilização diferente das demais ferramentas pesquisadas. Esta seção apresenta os principais aspectos necessários para o entendimento do processo conduzido.

3.1 Empresa na qual o estudo foi conduzido

A empresa onde o estudo foi realizado atua no setor de software para gestão de varejo situada no sudoeste do paraná, na cidade de Dois Vizinhos. A empresa conta com aproximadamente 300 colaboradores atualmente e trabalha com a metodologia ágil SCRUM[2], possuindo uma distribuição de colaboradores por equipes de no máximo 10 pessoas. O processo de desenvolvimento é realizado por sprints de 15 dias cada. A equipe na qual a pesquisa foi aplicada trabalha apenas com melhorias, possuindo um grande fluxo de chamados para criação de novos campos e telas. Novos objetos normalmente possuem sempre a mesma validação quanto à funcionalidade e usabilidade. Disponibilizando mais tempo do testador com testes de regra de negócio e testes exploratórios. O objetivo é criar testes automatizados apenas utilizando imagens, com o mínimo de codificação possível.

3.1 Descrição do processo

A aplicação da ferramenta dentro da equipe foi feita no período de 2 sprints (1 mês), em que cada participante recebeu um documento apresentando a tela do sistema a ser testada e a descrição dos testes automatizados a serem realizados.

Ação	Resultado Eperado
Informar no campo Tipo Pagamento a opção PRODUTO	Deve habilitar o flag "Gerar pedido automaticamente"
Informar o Tipo Bonificação como PAGAMENTO DE VERBA	Deve permitir informar
Informar data retroativa no campo "Vencimento"	O sistema nao deve permitir e deve apresentar mensagem de data invalida
Informar data atual no campo "Vencimento"	Deve permitir informar
No campo "Observação" inserir descrição "teste" e caracteres especiais "@@##$%¨&*"	O sistema deve permitir.
Marcar o falg "Gerar pedido automaticamente"	Devem ser apresentados os produtos inseridos no pedido de compra anteriormente
Clicar no botão "Confirmar"	Deve fechar a tela e voltar para a tela anterior

Figura 1. Exemplo de casos de teste enviados ao participantes para criação do teste automatizado.

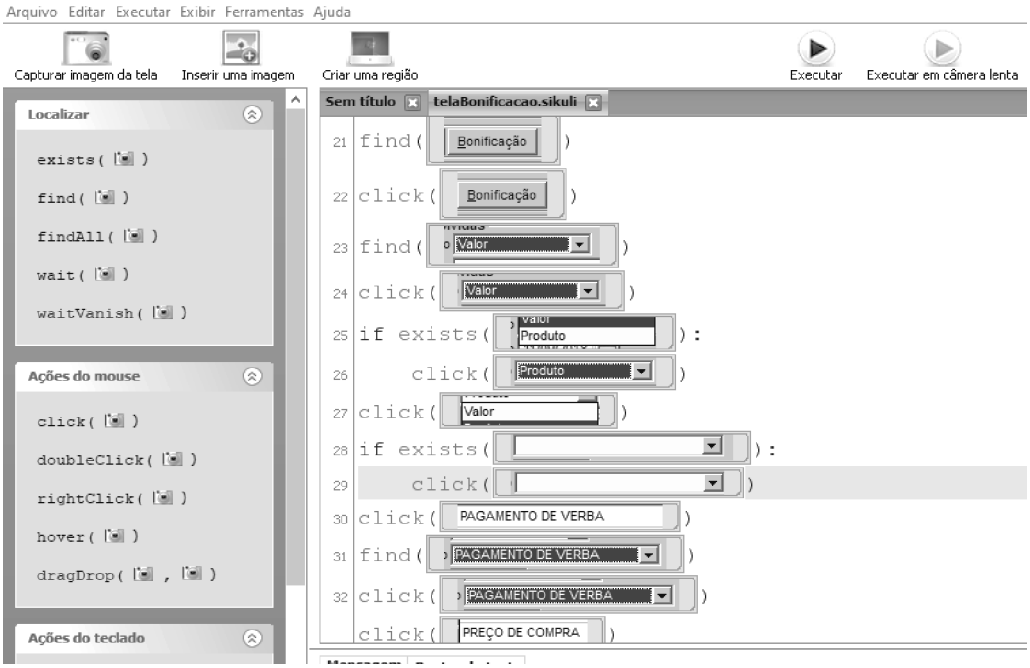


Figura 2. Exemplo de automação utilizando imagens.

O processo de automação foi realizado com o máximo de imagens possível, o mínimo de codificação e sem integração do sikuli com outra ferramenta. A criação dos testes automatizados foi feita após o teste manual ser finalizado e a avaliação da eficácia da ferramenta foi realizada através da utilização desses testes para retestar as telas após a

correção dos defeitos encontrados durante o teste manual. Neste contexto foi possível verificar que a ferramenta permitiu diminuir o esforço de teste sendo que o tempo do teste realizado pelo sikuli foi muito menor comparado ao teste manual.

4. Resultados

A ferramenta foi avaliada conforme os requisitos especificados no tópico anterior e por meio da aplicação de um questionário de satisfação entre os participantes. Nessa análise de resultados foi possível constatar que 90% dos participantes acordaram que a ferramenta se mostrou efetiva para diminuição do esforço de teste repetitivo. Com relação ao tempo e esforço para execução dos testes funcionais e de usabilidade automatizados comparados aos testes manuais e posterior utilização dos mesmos, no início o esforço de automação pode ser grande, porém ele diminui conforme os testes forem sendo executados ao invés de necessitar de uma pessoa para realizar esta função.

De uma forma geral a ferramenta atingiu o resultado esperado apesar de possuir algumas limitações quanto sua utilização em sistemas que não são desenvolvidos para web. Após o estudo foi possível perceber que é necessário um esforço da equipe de teste para ajustar os scripts de modo efetivo, entretanto, tais scripts têm baixo custo de manutenção. Muitos roteiros de teste já são implementados com outras ferramentas, portanto seria interessante ter alguma ferramenta intermediária que fizesse a migração dos scripts de teste antigos para Sikuli e também que a ferramenta, por usar algoritmos de processamento de imagem, pode ser mais efetiva para testadores que tenham conhecimento de processamento de imagem e consigam automatizar scripts mais efetivos.

5. Considerações finais

Desenvolvimento de software é uma tarefa que exige conhecimento técnico e muita atenção para evitar possíveis falhas.Este estudo apresentou os resultados de uma experiência do uso da ferramenta Sikuli para automatizar atividades de teste manuais com objetivo de reduzir o esforço do teste em atividades referentes a testes funcionais e testes de aceitação que são tipicamente manuais dentro do processo da empresa. A partir do estudo, percebeu-se que mesmo que a automação de testes por imagem (tarefa realizada pela Sikuli) seja uma área pouco explorada, com esta pesquisa foi possível comprovar alguns de seus benefícios. O

estudo conduzido oferece base para aprofundar sua aplicação em diferentes contextos futuramente, possibilitando a abertura de possibilidades de transferência tecnológica.

6. Referências

Bernardo, P. C. e Kon, F. A Importância dos Testes Automatizados. Engenharia de Software Magazine, v. 1, n 3, p. 54-57 (2008).

Costa, E. S. Extensão da Ferramenta Sikuli para Apoiar Testes Automatizados para Aplicações Windows Phone Sensíveis ao Contexto (2015). Dissertação (Bacharel em ciência da computação). UFAM, Amazonas.

Fantinato, M. *el al*. AutoTest – Um framework reutilizável para a automação de teste funcional de software. Campinas, SP: FUNTTE (2005).

Farias, C. M. Um Método de Teste Funcional para Verificação de Componentes (2003). Dissertação (Pós-Graduação em Informática). UFCG, Paraíba.

Martins, A. B.T. Automação de testes para plataforma Flex. 2015. Dissertação (Bacharel em ciência da computação). UFSC, Santa Catarina.

Fidelis, W. I. O e Martins, E. Estudo sobre a Utilização de Testes Automatizados em Projeto de Software de Grande Porte. Universidade estadual de Campinas, São Paulo, 2003.

Princípios ágeis na indústria farmacêutica: um estudo de caso

Kainã Chananeco de Souza, Mayra Ubatuba da Silveira, Carlos Francisco Soares de Souza

Campus Charqueadas – Instituto Federal Sul-rio-grandense (IFSul)
96.745-000 – Charqueadas – RS – Brasil

k9chananeco@gmail.com, may.ubatuba@gmail.com,
carlossouza@charqueadas.ifsul.edu.br

Abstract. *This paper presents use aspects of agile methodologies in the projects management of a pharmaceutical industry. The principles of agility were studied through the Scrum methodology, and applied in the project management sector of the company. As a case study, different deployment scenarios were reported and analyzed, where it was observed that the adaptation of the Scrum method to the organizational context of the company had a good adherence. The result was classified as positive, as it brought improvements to the process, in the communication between sectors, in the visibility of projects progress, as well as a better forecast of deliveries.*

Resumo. *Este artigo apresenta aspectos da implantação de metodologias ágeis no gerenciamento de projetos de uma indústria farmacêutica. Os princípios de agilidade foram estudados através da metodologia Scrum, e aplicados no setor de gerenciamento de projetos da empresa. Como estudo de caso, foram reportados e analisados diferentes cenários de implantação, onde se observou que a adaptação do método Scrum ao contexto organizacional da empresa teve uma boa aderência. O resultado foi classificado como positivo, pois trouxe melhorias para o processo, na comunicação entre setores, na visibilidade dos projetos em andamento, assim como uma melhor previsão das entregas.*

1. Introdução

O uso de projetos para alcançar os objetivos estratégicos de uma empresa é uma das formas a qual as organizações têm adotado nos últimos anos (SILVA et al. 2017). Para Aubry et al. (2008) a organização orientada a projeto (*project-oriented organization*) deve atender às complexidades relacionadas a este novo cenário, superando a perspectiva tradicional de eficiência. Uma das técnicas para superar esse desafio é a de Gerenciamento Ágil de Projetos (GAP) que vem se tornando cada vez mais popular e essencial nas empresas que buscam ampliar e melhorar o seu negócio. Apesar de inicialmente os métodos ágeis terem surgido para uso no desenvolvimento de *softwares*, esses métodos foram se tornando interessantes também para empresas que tenham a intenção de gerar produtos inovadores e competitivos em um prazo curto de tempo, visto que os métodos tradicionais de gerenciamento de projetos dificultam sua implantação por exigirem esforço de planejamento prévio à execução. A agilidade

permite que em seus projetos uma organização receba alterações em seu ambiente e reconfigure recursos conforme necessário.

Neste contexto, a grande quantidade de projetos que as empresas demandam às suas áreas de projetos acaba por dificultar a organização dessas equipes quanto à prazos de entrega e acompanhamento contínuo, especialmente quando utilizados métodos de gerenciamento tradicionais que costumam ser inflexíveis, como o modelo Cascata.

De acordo com esta perspectiva, o presente trabalho apresenta a utilização de princípios ágeis no gerenciamento de projetos de uma empresa privada de fabricação de medicamentos.

2. Metodologias ágeis

A concretização dos métodos ágeis, segundo Beck et al. (2001), foi marcada pela publicação do Manifesto Ágil, um documento elaborado por um grupo de desenvolvedores de software. Este documento descreve valores e princípios que servem como guia para as atuais práticas ágeis de gerenciamento de projetos. Atualmente existem diversas técnicas e metodologias que são reconhecidas como ágeis e apesar de cada uma conter ciclos de vida e regras diferentes, muitas empresas optam por utilizar essas técnicas de forma adaptada a seu contexto e suas necessidades, deixando de fazer uso de algumas premissas, inserindo passos novos ou até mesmo juntando conceitos de mais de uma técnica.

O Manifesto Ágil se posiciona priorizando indivíduos e interações mais que processos e ferramentas, software em funcionamento mais que documentação abrangente, colaboração com o cliente mais que negociação de contratos e responder a mudanças mais que seguir um plano.

Para Date et al. (2016), um dos fatores que têm feito os métodos ágeis de gestão de projetos se popularizarem nas empresas, é o de que sua estrutura leve e dinâmica, permite adaptações nos projetos mesmo durante o andamento. O autor analisou o uso de uma técnica ágil no desenvolvimento de um projeto em uma instituição de educação pública. Tal projeto iniciou a sua execução utilizando métodos tradicionais, e por estarem enfrentando dificuldades no tocante a atrasos nas entregas decorrentes a despadronização da base de dados do produto do projeto**,** a equipe responsável optou por dar continuidade no projeto empregando técnicas estabelecidas pelo método ágil *Scrum*.

Ainda assim, no decorrer da continuidade de execução do projeto alguns empecilhos fizeram com que o método não fosse implantado em sua totalidade, tendo que ser adaptado às circunstâncias da instituição. Neste sentido, Date et al. (2016) cita alguns autores, como mais recentemente Simoyama, Bueno e Battisti (2016), que são favoráveis a adaptações nos métodos, personalizando-os de forma a se moldarem às necessidades de diferentes projetos e ambientes organizacionais em distintas naturezas, públicas e privadas. Neste contexto, os autores classificam o resultado da implantação do método ágil como uma experiência positiva, tanto para equipe que desenvolveu o projeto quanto para os clientes do projeto, apesar das dificuldades enfrentadas comumente pelo serviço público, como a limitação de recursos, sejam humanos ou de infraestrutura.

4. Cenários

A partir de alguns treinamentos recebidos sobre metodologias ágeis, viu-se uma oportunidade de melhoria no setor de engenharia de projetos da empresa, uma vez que fora diagnosticado, em momentos anteriores, uma falta de comunicação entre a área de projetos e as outras áreas da empresa.

Inicialmente os setores da empresa eram gerenciados por um modelo tradicional, semelhante ao modelo Cascata, onde um projeto era feito por etapas, utilizando o gráfico de Gantt para analisar o andamento de cada projeto. Como existiam muitos projetos para serem gerenciados e as prioridades de entrega costumavam mudar com o tempo, notou-se que o modelo utilizado era falho e acabava gerando atrasos e desorganização, principalmente por não possibilitar a realização de pequenas entregas durante o desenvolvimento e por não prover uma visão geral do andamento dos projetos que estavam sendo executados.

O modelo tradicional funcionava bem para alguns projetos cujos requisitos eram bem conhecidos e já se tinha conhecimento de como atuar, como, por exemplo, em projetos simples de engenharia, como mudanças estruturais em algum ambiente, que envolviam requisição de compras, requisição de serviço, dentre outros.

Entretanto, em um ambiente farmacêutico/industrial a área de projetos também presta serviços para áreas como Qualificação, Serviços Técnicos e Controle de Qualidade, que possuem uma demanda de projetos bem mais complexa, pois os requisitos nunca são totalmente conhecidos, nem mesmo pelo próprio setor solicitante, fazendo com que estes projetos sofram mudanças de prioridades das atividades quando em execução, sejam por fatores internos ou até mesmo externos, como as auditorias da ANVISA[1], o que por sua vez impactam não apenas no projeto em questão, mas também nos demais, contribuindo para a desorganização do setor.

Desta forma, após estudos, constatou-se a necessidade da implantação de alguma metodologia ágil, como o *Scrum*, em busca de agilizar as entregas, melhorar o acompanhamento dos projetos e conseguir lidar melhor com as mudanças de prioridade que aconteciam durante a execução dos projetos. Essa implantação foi dividida em dois cenários diferentes: no primeiro, tentou-se utilizar os conceitos do *Scrum*, dentro do possível no contexto do setor, de forma a tentar que a utilização fosse mais próxima possível ao modelo original, com o objetivo de avaliar a eficiência do modelo no contexto da empresa; no segundo cenário, algumas adaptações foram feitas ao primeiro, deixando-o, conceitualmente, um pouco mais distante do *Scrum,* porém mantendo algumas características importantes, de forma a melhor atender as necessidades da empresa.

4.1. Equipe

A área de projetos da empresa, atualmente, é constituída por uma equipe de três pessoas, sendo um coordenador, um técnico de projetos e um estagiário. Além de suas atividades rotineiras, executa papel similar a de um *Scrum Master* dos projetos em execução, pois dá condições para que os princípios ágeis sejam seguidos, além de

[1] ANVISA: Agência Nacional de Vigilância Sanitária.

resolver eventuais impedimentos que ocorram durante o processo. O estagiário acumula o papel de *Product Owner* e membro da equipe executora, pois é o responsável pela interação com os *Stakeholders* de cada setor e manter as atividades do *Product Backlog* sempre atualizadas, com suas devidas prioridades, além de auxiliar o técnico de projetos na execução dos mesmos.

4.2. Cenário 1

Na primeira experiência do setor com metodologias ágeis, optou-se pela utilização em apenas um projeto piloto, e se procurou seguir um modelo semelhante ao ciclo de vida da metodologia S*crum*, adaptado ao contexto atual da empresa.

O projeto selecionado foi da área de Serviços Técnicos, onde se realizou um estudo para totalização da área de cada material para realização de limpeza. Material, neste caso são máquinas e ferramentas utilizadas na fabricação dos medicamentos, uma vez que durante este processo, parte dos materiais entra em contato com o que está sendo fabricado, e, portanto, precisa ser limpa.

O setor de projetos era acionado para realizar o levantamento destas áreas, e, rotineiramente, informações como esta eram solicitadas em grandes demandas, porém antes da entrega planejada a prioridade dos materiais a serem limpos acabava mudando, devido a alguma auditoria, seja ela externa ou interna.

Todo material solicitado passou a ser considerado uma atividade do *Product Backlog*, e que depois eram negociadas para compor o *Backlog* de cada *Sprint*, conforme a sua prioridade. As *Sprints* tinham duração de uma semana e ao final desta eram feitas as entregas.

Antes de iniciar o desenvolvimento das atividades, era realizado o Planejamento da *Sprint*, onde a partir da estorias do projeto eram definidas as prioridades para serem entregues em cada *Sprint*. Durante o processo, as Reuniões Diárias aconteciam, e nela a equipe tomava conhecimento de qual material estava sendo medido e por quem, da mesma forma, o que já havia sido e o que seria medido no dia seguinte. Da mesma forma, ao final da *Sprint*, durante a Revisão sempre era feito o contraste entre o que fora prometido e o que estava sendo entregue, e na Retrospectiva as melhorias no processo eram reportadas, como por exemplo a redução na quantidade de máquinas da selecionadas para a *Sprint*.

Este projeto piloto durou três meses e fora dividido em fases, onde cada uma tinha duração de um mês, e possuía quatro *Sprints*.

Depois de três fases, a aplicação se provou um sucesso. Em apenas uma das doze *Sprints* apresentou atraso mas em momento algum isso interfere diretamente no trabalho nosso cliente naquele momento (a área de serviços técnicos). Devido ao framework, foi visto que a visibilidade sobre o que acontecia naquele projeto melhorou, fazendo com que a organização da equipe também melhorasse.

Visto a grande melhora no desempenho da equipe graças ao *Scrum*, começou-se a pensar na próxima etapa, a aplicação em todos os projetos da área.

4.3. Cenário 2

Neste cenário, os princípios ágeis foram utilizados não apenas para o gerenciamento de um único projeto, mas para todos os projetos, como um portfólio. Para tal, foram considerados apenas os projetos cuja execução eram de responsabilidade da própria equipe, uma vez que alguns projetos que o setor gerencia são executados por terceiros, ficando a equipe limitada a apenas realizar a contratação do serviço terceirizado responsável por desenvolver o trabalho.

Nestas circunstâncias, é criado um *Product Backlog* composto pelas atividades de todos os projetos. A negociação da priorização dessas atividades é feita em reuniões presenciais entre a equipe e, se necessário, *Stakeholders* sendo essa reunião intituladas de *Sprint Planning*. Durante o próprio *Sprint Planning* é definido o que será executado durante as próximas *Sprints,* geralmente de uma semana.

A área de projetos da empresa atende a quase todas as outras áreas, executando projetos para diversos setores, inclusive, simultaneamente, devido a natureza de alguns destes. Considerando este fato e de que, conforme visto no capítulo 4.1, a equipe é composta por apenas três funcionários, as entregas eram realizadas após a *Sprint Review* e a *Sprint Retrospective* através de e-mail e não presencialmente, da mesma forma como o feedback recebido do setor.

5. Considerações finais

É de amplo conhecimento que as metodologias ágeis têm sido bastante utilizadas, principalmente no contexto de desenvolvimento de *software*. Todavia, o *Scrum* é uma metodologia ágil para gerenciamento de projetos de diversas naturezas. Existem outras metodologias voltadas especificamente para a área de *software*, como a Programação Extrema, o que apesar de não ser foco deste artigo, segue os mesmos princípios ágeis.

Neste sentido, apesar de ter sido buscada a utilização da metodologia *Scrum*, não existia nesta unidade da empresa, até o momento de fechamento deste artigo, um especialista na área, o que tornou o processo mais desafiador. Todavia, alguns membros da empresa foram treinados para auxiliar no processo.

É notável que apesar dos objetivos, em nenhum dos cenários de implantação foi possível a utilização bruta da metodologia, uma vez que o contexto da empresa não era favorável. Entretanto, a simples mudança no *modus operandi* da execução das atividades gerenciadas pelo setor de Gestão de Projetos, onde foram utilizados alguns dos princípios ágeis, deu um retorno significativo para o setor, que hoje consegue ter melhor visibilidade do andamento projetos, inclusive do *status* de execução dos projetos que são executados por funcionários do setor, o que impactou positivamente inclusive no *follow-up* diário com os fornecedores.

Outro benefício percebido foi uma melhora na previsão das entregas, assim como também se conseguiu aumentar a quantidade de entregas, uma vez que projetos não ficaram mais bloqueados por outros, como acontecia quando se utilizava o modelo tradicional.

Uma das maiores dificuldade enfrentadas no gerenciamento dos projetos do setor foi necessidade de se trabalhar com empresas terceirizadas e o baixo controle diário que o time tem sobre elas, porém muitas das cerimônias do *Scrum* se mostraram muito importantes e eficientes na gestão, como as Reuniões Diárias, que aumentam o contato e a comunicação do time.

Percebe-se que mesmo adaptado, a utilização de metodologias ágeis melhora a visibilidade dos projetos que estão sendo produzidos naquele momento fazendo com que seja possível ter a noção da capacidade de trabalho do time, e, finalmente, as reuniões com os *Product Owners*, representantes de cada setor, se mostraram fundamentais para aproximar as áreas e tornar as relações e os entendimentos do projeto mais legíveis.

Analisando os cenários, pode-se observar que a adaptação do *Scrum* ao modelo tradicional utilizado anteriormente pela empresa é considerada satisfatória e a que mais se encaixa à forma de trabalho da equipe de gerenciamento de projetos, pois a organização do time aumentou e os pacotes de documentos diminuíram drasticamente, o que contribuiu para a agilidade da entrega dos projetos.

Referências

Silva, F. S. M.; Rego, M. L.; Borges, H. G. and Irigaray, H. A. R. (2017) "Uma análise de habilitadores de métodos ágeis em projetos". In: VI Simpósio Internacional de Gestão de Projetos, Inovação e Sustentabilidade (SINGEP). 11., 2017, São Paulo.

Aubry, M.; Hobbs, B.; Thuillier, D. (2008) "A new framework for understanding organizational project management through the PMO". International Journal of Project Management, v. 25, n. 1, p. 38-4

Date, R. N., Pinhochet, L. H. C., Bueno, R. L. P. and Nemoto, M. C. M. O (2016) "Aplicação do Método Ágil Scrum em uma Fundação Educacional do Setor Público". Revista de Gestão e Projetos - GeP, [S.l.], v. 7, n. 2, p. 75-94, aug. 2016. ISSN 2236-0972

Beck, K.; Beedle, M.; Bennekum, A.; Cockburn, A.; Cunningham, W.; Fowler, M.; Grenning, J.; Highsmith, J.; Hunt, A.; Jeffries, R.; Kern, J.; Marick, B.; Martin, R. C.; Mellor, S.; Schwaber, K.; Sutherland, J. and Thomas D. (2001) "Manifesto for Agile Software Development". Disponível em: http://agilemanifesto.org/. Acesso em 27 de setembro de 2018.

Simoyama, F. O.; Bueno, R. L. P.; Battisti, M. C. G. (2016). Adaptation and implementation of Scrum methodology for agile projects in a government agency. Revista Gestão e Tecnologia, Pedro Leopoldo, v. 16, n. 2, p. 260-276, Maio/Agosto

Lista de Autores