ANAIS

I Workshop sobre Aspectos Sociais, Humanos e Econômicos de Software (WASHES 2016)

I WORKSHOP SOBRE ASPECTOS SOCIAIS, HUMANOS E ECONÔMICOS DE SOFTWARE (WASHES 2016)

25 de Outubro de 2016

Maceió, Alagoas – Brasil

ANAIS

Sociedade Brasileira de Computação – SBC

COORDENADORES DO COMITÊ DE PROGRAMA

Davi Viana – Universidade Federal do Maranhão (UFMA)
Rodrigo Santos – Universidade Federal do Estado do Rio de Janeiro (UNIRIO)
Igor Steinmacher – Universidade Tecnológica Federal do Paraná (UTFPR)
Sabrina Marczak – Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS)

REALIZAÇÃO

Sociedade Brasileira de Computação (SBC)

EXECUÇÃO

Universidade Federal de Alagoas (UFAL)
Instituto Federal de Alagoas (IFAL)

ISBN: 978-85-7669-352-9

I WORKSHOP ON SOCIAL, HUMAN, AND ECONOMIC ASPECTS OF SOFTWARE (WASHES 2016)

October 25th, 2016

Maceió, Alagoas, Brazil

PROCEEDINGS

PROGRAM COMMITTEE CHAIRS

Davi Viana – Universidade Federal do Maranhão (UFMA)
Rodrigo Santos – Universidade Federal do Estado do Rio de Janeiro (UNIRIO)
Igor Steinmacher – Universidade Tecnológica Federal do Paraná (UTFPR)
Sabrina Marczak – Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS)

REALIZATION

Brazilian Computer Society (SBC)

EXECUTION

Federal University of Alagoas (UFAL)
Federal Institute of Alagoas (IFAL)

ISBN: 978-85-7669-352-9

Apresentação

O Workshop sobre Aspectos Sociais, Humanos e Econômicos de Software (WASHES 2016), em sua primeira edição, traz como principal tema os "Desafios da Qualidade e os Aspectos Sociais, Humanos e Econômicos de Software". O WASHES foca na interação entre questões críticas que exercem influência sobre a engenharia e qualidade de software. Fatores humanos e aspectos sociais em qualidade de software têm sido discutidos por pesquisadores e profissionais da indústria, dado que métodos, técnicas e ferramentas afetam (e são afetados) pelos *stakeholders* e suas interações, impactando as atividades de engenharia de software. Da mesma forma, o software é uma fonte de valor para os negócios de diversas organizações, sejam elas fornecedoras ou adquirentes, representando um elemento crucial para o seu sucesso econômico.

Decisões tomadas no desenvolvimento de software têm implicações econômicas substanciais, seja na perspectiva de custo e lucro, ou incluindo outras perspectivas de valor, tais como benefícios, riscos e oportunidades. Tais decisões perpassam aquisição, requisitos, arquitetura, verificação e validação, manutenção e reutilização, e demandam uma análise mais detalhada do ponto de vista da qualidade de processo e de produto. Nesse contexto, o WASHES tem por objetivo reunir pesquisadores e profissionais da indústria interessados nos aspectos sociais, humanos e econômicos de software a fim de discutir modelos, métodos, técnicas e ferramentas para abordar esses aspectos visando à qualidade de software, bem como os desafios existentes nesse contexto.

O Comitê Diretivo do WASHES 2016 é constituído por 4 pesquisadores. Por sua vez, o Comitê de Programa do WASHES 2016 é formado por 20 pesquisadores de instituições do Brasil e do exterior, com atuação e produção relevantes nas áreas de pesquisa envolvidas no workshop. Os membros do Comitê de Programa conduziram um processo rigoroso de revisão, sendo que cada artigo foi avaliado e discutido por pelo menos dois membros durante a fase de consenso.

O WASHES 2016 recebeu 30 artigos submetidos, sendo 5 em inglês e 25 em português. Após o processo de revisão do Comitê de Programa, além da análise final do Comitê Diretivo, foram aceitos 6 artigos completos e 6 artigos curtos. Haverá premiação dos melhores artigos e convite para submissão de versão estendida para uma Edição Especial a ser divulgada. Para estimular a discussão de pesquisas com potencial, 3 trabalhos serão apresentados como pôsteres. O WASHES 2016 vai contar ainda com um painel para discutir o tema desta edição e com uma dinâmica para a organização da Agenda de Pesquisa e de Colaboração do WASHES.

É com satisfação que damos as boas-vindas aos autores e apresentadores de artigos, da academia e da indústria. Também recebemos com grande prazer os demais participantes do SBQS 2016, que gostaríamos de convidar a tomar parte ativamente das discussões e momentos de integração proporcionados pelo workshop. Adicionalmente, gostaríamos de agradecer a todos os demais autores que submeteram seus artigos, aos membros do Comitê de Programa e do Comitê Diretivo, e aos organizadores e patrocinadores do SBQS 2016 pelo suporte na realização deste workshop.

Esta edição é organizada conjuntamente pela Universidade Federal do Maranhão (UFMA), Universidade Federal do Estado do Rio de Janeiro (UNIRIO), Universidade Tecnológica Federal do Paraná (UTFPR) e Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS). O evento será realizado no Centro Cultural e de Exposição Ruth Cardoso, em Maceió, Alagoas, no dia 25 de outubro, em conjunto com o XV Simpósio Brasileiro de Qualidade de Software (SBQS 2016).

Esperamos que tenham uma ótima estada em Maceió!

Maceió, 25 de outubro de 2016.

Davi Viana
Rodrigo Santos
Igor Steinmacher
Sabrina Marczak
Coordenadores do WASHES 2016

Foreword

The Workshop on Social, Human, and Economic Aspects of Software (WASHES 2016), in its first edition, aims at putting together competencies and technologies of these three related aspects: social, human and economic. This year's main topic is "Challenges of Quality and the Social, Human, and Economic Aspects of Software". WASHES focuses on the interaction between critical aspects that influence software engineering and software quality. Human and social aspects in software quality have been discussed by researchers and practitioners since methods, techniques, and tools affect (and are affected by) stakeholders and their interactions. Such interactions affect software engineering activities. Similarly, software is a source of value for business in several organizations, either being software suppliers or acquirers, representing the key factor for their economic success.

Decisions made in the software development processes and activities have economic implications on the profit and/or cost perspectives. Such decisions are beyond software acquisition, software requirements, architecture, verification and validation, maintenance, and software reuse. Therefore, to make a decision, one needs a more detailed analysis from the point of view of software product and process quality. In that context, WASHES aims to bring together researchers and practitioners that have interests in social, human, and economic aspects of software in order to discuss models, methods, techniques, and tools to achieve software quality and deal with the existing challenges in this context.

The WASHES 2016 Steering Committee is composed of 4 researchers. The WASHES Program Committee is composed of 20 researchers from national and international institutions, with relevant expertise and production in the related research areas of the workshop. Members of the Program Committee conducted a rigorous review process, in which each paper was assessed and discussed in details by at least two members.

In this WASHES edition, we received 30 papers (5 papers in English and 25 in Portuguese). After the review process from the Program Committee and the final analysis of the Steering Committee, 6 full papers and 6 short papers were accepted. There will be an award for the best papers and invitations to submit extended versions to a Special Issue (to be announced). Additionally, 3 papers will be presented as posters, since they discuss potential research ideas. Finally, we will promote a panel to discuss this edition's main topic, and a special session to define the WASHES Research and Collaboration Roadmap.

We welcome the authors and articles presenters with delight, from both academia and industry. We also welcome with great pleasure the SBQS 2016 participants, which we would like to invite to actively take part in discussions and integration moments provided by the workshop. Additionally, we would like to thank all the other authors who submitted their papers, the Steering and Program Committees' members, and the organizers and sponsors of SBQS 2016, for their support for the accomplishment of this workshop.

This edition is jointly organized by the Federal University of Maranhão (UFMA), Federal University of the State of Rio de Janeiro (UNIRIO), Federal University of Technology – Paraná (UTFPR), and Pontifical Catholic University of Rio Grande do Sul (PUCRS). The event will be held in the Centro Cultural e de Exposições Ruth Cardoso, in Maceió, State of Alagoas, Brazil, in October 25th, co-located with XV Brazilian Symposium on Software Quality (SBQS 2016).

We wish a pleasant stay in Maceió!

Maceió, October 25th 2016.

Davi Viana
Rodrigo Santos
Igor Steinmacher
Sabrina Marczak
WASHES 2016 Chairs

I Workshop sobre Aspectos Sociais, Humanos e Econômicos de Software (WASHES 2016)
I Workshop on Social, Human, and Economic Aspects of Software (WASHES 2016)

Coordenadores Gerais / General Chairs

Davi Viana - UFMA

Rodrigo Santos - UNIRIO

Igor Steinmacher - UTFPR

Sabrina Marczak - PUCRS

Comitê de Programa / Program Committee

Adler Diniz, UNIFEI
Aline Vasconcelos, IFF
Christoph Treude, University of Adelaide
Claudia Cappelli, UNIRIO
Cláudia Werner, COPPE/UFRJ
Cristiano Maciel, UFMT
Emília Mendes, Blekinge Institute of Technology
Fernando Figueira Filho, UFRN
Gleison Santos, UNIRIO
Henrique Cukierman, COPPE/UFRJ
Heitor Costa, UFLA
Igor Wiese, UFTPR-CM
Inaldo Costa, UFMA
João Porto de Albuquerque, University of Warwick
José Maria David, UFJF
Júlio César Sampaio do Prado Leite, PUC-Rio
Marco Gerosa, IME/USP
Simone Vasconcelos, IFF
Tayana Conté, UFAM
Vanessa Nunes, UnB

Revisores Externos / External Reviewers

Awdren Fontão - ICOMP/UFAM
Bernardo Estácio - PUCRS
Cristina Cerdeiral - UNIRIO
Ivaldir de Farias Junior - UFPE
Jose Jorge Dias Júnior - UFPB
Joselaine Valaski - PUCPR
Letícia Machado - PUCRS
Odette Passos - ICET/UFAM
Simone Amorim - IFBA
Taisa Gonçalves - University of Valenciennes
Thaiana Lima - COPPE/UFRJ

Sociedade Brasileira de Computação

Diretoria

Presidente: Lisandro Zambenedetti Granville (UFRGS)
Vice-Presidente: Thais Vasconcelos Batista (UFRN)
Administrativa: Renata de Matos Galante (UFRGS)
Finanças: Carlos André Guimarães Ferraz (UFPE)
Eventos e Comissões Especiais: Antônio Jorge Gomes Abelém (UFPA)
Educação: Avelino Francisco Zorzo (PUCRS)
Publicações: José Viterbo Filho (UFF)
Planejamento e Programas Especiais: Cláudia Lage Rebello da Motta (UFRJ)
Secretarias Regionais: Marcelo Duduchi Feitosa (CEETEPS)
Divulgação e Marketing: Eliana Silva de Almeida (UFAL)

Diretorias Extraordinárias

Relações Profissionais: Roberto da Silva Bigonha (UFMG)
Competições Científicas: Ricardo de Oliveira Anido (UNICAMP)
Cooperação com Sociedades Científicas: Raimundo José de Araújo Macêdo (UFBA)
Articulação de Empresas: Sérgio Castelo Branco Soares (UFPE)

Conselho

Membros Titulares

Mandato 2013-2017

Alfredo Goldman (IME/USP)
Daltro José Nunes (UFRGS)
José Palazzo Moreira de Oliveira (UFRGS)
Maria Cristina Ferreira de Oliveira (ICMC/USP)
Wagner Meira Junior (UFMG)

Mandato 2015-2019

Altigran Soares da Silva (UFAM)
Ana Carolina Salgado (UFPE)
Fabio Kon (USP)
Paulo Roberto Freire Cunha (UFPE)
Rodolfo Azevedo (UNICAMP)

Suplentes - 2015-2017

Cláudia Maria Lima Werner (UFRJ)
Itana Maria de Souza Gimenes (UEM)
Flávio Rech Wagner (UFRGS)
Guilherme Horta Travassos (UFRJ)

WASHES 2016

Sumário / Contents

Artigos Completos / Full Papers

Artigos Curtos / Short Papers

Pôsteres / Posters

Um Modelo para o Gerenciamento do Crowdsourcing em Projetos de Software

William Simão de Deus[1], **Renata Marques Barros**[1], **Alexandre L'Erario**[1]

[1] Programa de Pós Graduação em Informática (PPGI)
Universidade Tecnologica Federal do Parana (UTFPR)
Cornélio Procópio – Paraná – Brasil

{williamsimao,renata_mbarros}@outlook.com, alerario@utfpr.edu.br

Abstract. *Crowdsourcing is an emerging business model applied to software development projects due to the cost reduction and involvement of experts. However, there are challenges on how to manage it in such projects due to the dispersion of participants and the activities of parallelization. In this study, we conducted a literature review and verified practices applied to the management of crowdsourcing in software projects. We have compiled the results of our analysis on a management model that has been validated in a quasi-experiment crowdsourcing. The application of the model demonstrated capacity and efficiency to realize the management of dispersed participants and activities.*

Resumo. *O crowdsourcing é um emergente modelo de negócios aplicado ao desenvolvimento de projetos de software em virtude da redução de custos e envolvimento de especialistas. Entretanto, existem desafios sobre como gerenciá-lo em tais projetos devido à dispersão de participantes e a paralelização de atividades. Neste estudo, nós realizamos uma análise da literatura e verificamos práticas aplicadas para o gerenciamento do crowdsourcing em projetos de software. Nós compilamos o resultado da nossa análise em um modelo de gerenciamento que foi validado em um quase-experimento crowdsourcing. A aplicação do modelo demonstrou capacidade e eficiência para realizar o gerenciamento entre os participantes dispersos e as atividades.*

1. Introdução

O termo *crowdsourcing* (CS) foi originalmente cunhado em 2006 por [Howe 2006] que definiu o conceito no "ato de desenvolver uma atividade por um agente designado ou terceirizar através de uma chamada para um grande grupo de pessoas". O CS tornou-se um modelo de negócios utilizando a inteligência coletiva para solucionar problemas ou completar atividades [Pan and Blevis 2011]. Segundo [Schwartz et al. 2015] a aplicação do CS providenciou de forma simples o acesso, o fornecimento e a geração de conhecimento, oferecendo benefícios para realização de serviços, produção de ideias ou conteúdo a partir da solicitação da contribuição de um grupo [Benedek et al. 2015].

Segundo [LaToza and van der Hoek 2016] fatores como redução de custo, paralelização de atividades e a colaboração de especialistas impulsionaram sua prática na Engenharia de Software, tornando-se um tópico popular e emergente no desenvolvimento de projetos de software [Dwarakanath et al. 2016, Tajedin and Nevo 2013].

Entretanto, os desafios causados pela dispersão geográfica, realização de atividades e descentralização de grupos de trabalho denotam a complexidade de gerenciar o CS [Satzger et al. 2014]. A necessidade de gerenciar o CS é paralelo a realidade das novas plataformas que focam sua gestão em fatores humanos (*feedbacks* e comunicação) e tecnológicos (*microtasks*) [Kucherbaev et al. 2016]. Porém, apesar do crescimento de utilização do CS em projetos de software, existe uma escassez de pesquisas concentradas no desenvolvimento e gerenciamento do CS nestes projetos [LaToza and van der Hoek 2016, Mattauch 2013].

Neste estudo, nós apresentamos um modelo de gerenciamento do CS em projetos de software a partir da coordenação dos participantes e atividades, contribuindo para os aspectos sociais e humanos do desenvolvimento de software em ambientes CS. Nós conduzimos uma análise da literatura e identificamos cinco práticas caracterizadas pela atomização de atividades, documentação do projeto, formação dos grupos, comunicação entre participantes e realização de *feedbacks*. Estas práticas foram compiladas em um modelo de gerenciamento CS validado em um quase-experimento. Os resultados demonstraram que o modelo auxiliou a execução do projeto, facilitando o gerenciamento dos participantes e das atividades desenvolvidas.

Este trabalho foi estruturado em 7 seções. Na primeira seção foi apresentada a contextualização, as lacunas da área e o objetivo deste estudo. Na segunda seção abordamos tópicos referentes à caracterização do CS em projetos de software. Um levantamento de publicações com respostas parciais ao deste estudo é encontrado na terceira seção. Na quarta seção apresentamos a construção do modelo de gerenciamento. Na quinta seção apresentamos a execução do quase-experimento e na sexta seção as análises e discussões dos resultados. Por fim são demonstrados os pontos alcançados com a condução deste estudo, seus fatores de limitação e os trabalhos para pesquisas futuras.

2. Crowdsourcing

O CS é formado pelos participantes do projeto, a organização responsável, a plataforma e as atividades que serão executadas [Hosseini et al. 2014]. Segundo [Zakariah et al. 2015] a utilização do CS como um modelo de negócios oferece benefícios para colaboradores e organizações como a flexibilidade de tempo e localização, aquisição de habilidades, diminuição de gastos, atração de talentos, aquisição de expertise externa, entre outros [Hossain 2012, Pedersen et al. 2013].

Apesar de similaridades com o *open source*, o CS geralmente agrega participantes motivados por fatores financeiros. Outra característica percebida no CS se refere ao desenvolvimento concorrente, diferente do desenvolvimento distribuído. Desta maneira, são amplificados os desafios de gerenciamento do CS devido ao nível de conhecimento entre os participantes, recrutamento de colaboradores, verificação do desenvolvimento de múltiplas atividades, coordenação das contribuições e compartilhamento de informações para membros do projeto [Hosseini et al. 2015, LaToza and van der Hoek 2016].

3. Trabalhos Relacionados

No estudo conduzido pelos autores [Kucherbaev et al. 2016] é apresentada uma abordagem para possibilitar o gerenciamento por meio de um processo CS. Sua análise produziu um conjunto de considerações com objetivos de direcionar novas pesquisas e entusiastas

da área. Entretanto, a pesquisa possui um caráter explanatório e suas conjecturas ainda não foram avaliadas a fim de comprovar, potencializar ou confrontar resultados.

Uma pesquisa conduzida pelos autores [Cullina et al. 2016] direcionou a necessidade de identificar, regular e refinar sub processos CS. Uma contribuição inicial refere-se a definição entre os conceitos chaves identificados na literatura e a geração de um modelo conceitual para facilitar sua aplicação. Porém, os autores declararam que o modelo está em um estágio de maturidade inicial e pode receber modificações para a sua implementação. A Tabela 1 apresenta a comparação entre os estudos similares e o modelo de gerenciamento CS desenvolvido neste artigo:

Tabela 1. Comparação Entre os Estudos

Estudo	Métodos	Validação	Contribuição
[Kucherbaev et al. 2016]	Análise da Literatura	-	Teórica
[Cullina et al. 2016]	Análise da Literatura	-	Teórica
Modelo desenvolvido	Análise da Literatura	Quase-experimento	Prática

Os dois estudos relacionados utilizaram métodos de pesquisa baseado na análise da literatura e não apresentaram validação em ambientes CS, assim a contribuição deu-se apenas de forma teórica. O modelo desenvolvido neste estudo também foi gerado com base na análise da literatura, porém a validação ocorreu através de um quase-experimento simulando a aplicação do CS e demonstrando os resultados de forma prática.

4. Modelo de Gerenciamento Crowdsourcing

Com o objetivo de compreender o estado da arte sobre o gerenciamento do CS nós conduzimos uma análise da literatura, selecionando estudos sob quatro critérios, apresentados a seguir:

- C1: O estudo corresponde ao âmbito do *crowdsourcing*;
- C2: O estudo corresponde ao âmbito de projeto de software;
- C3: O estudo apresenta práticas sobre como realizar o gerenciamento;
- C4: O estudo está completo e foi publicado em uma conferência, revista, jornal ou *workshop*.

Nós realizamos buscas combinando os termos *"management", "crowdsourcing", "crowd sourcing"e "software"* em bibliotecas digitais que possuem amparo de pesquisa com a Universidade Tecnologica Federal do Paraná - Campus Cornélio Procópio. As bibliotecas possuem um acervo de estudos na área computacional, apresentando publicações voltadas ao CS e projetos de software. As bibliotecas utilizadas foram: *IEEE Xplore (http://ieeexplore.ieee.org)*, *ACM Digital Library (http://dl.acm.org)* e *ScienceDirect (http://www.sciencedirect.com/)*.

Após a realização das buscas nas bases especificadas, nós selecionamos publicações que atendiam os quatro critérios (C1, C2, C3 e C4). Os estudos selecionados foram analisados para extração de informações sobre as fases do CS, definições sobre o gerenciamento do CS e se é uma característica referente ao fator humano (H) e/ou tecnológico (T), a relação de estudos é apresentado na Tabela 2.

A partir da análise dos estudos foram identificadas cinco práticas para a criação do modelo de gerenciamento do CS, a atomização das atividades, a documentação do projeto, a formação do grupo, a realização de comunicação e feedbacks.

Tabela 2. Estudos Analisados

Referência	Fases do CS	Definição	Fator
[Dwarakanath et al. 2015]	Design	Elicitação e análise de um projeto	H
	Documentação	Descrever os componentes do projeto	H/T
	Implementação	Criação e postagem das tarefas em uma plataforma	H
	Verificação	Execução de testes unitários	H
[Tsai et al. 2014]	Apenas definições/ atividades	(a) requisitos; (b) design; (c) codificação (d) teste; (e) comunicação e (f) documentação.	H/T
[Saremi and Yang 2015]	Atomização	Distribuição com o máximo possível de tarefas	T
	Plataforma	Enviar para uma plataforma *crowdsourcing* as tarefas	T
	Registro	Mecanismo para o registro de tempo utilizado	T
	Comunicação	Comunicação e feedbacks entre colaboradores e companhia	H
[Murray-Rust et al. 2015]	Tarefas Independentes	Coordenação de colaboradores em ambiente de desenvolvimento usando tarefas atômicas e paralelas, canais de comunicação e feedbacks	H/T
[Stol and Fitzgerald 2014]	Apenas definições/ atividades	Utilizar a atomização de tarefas, coordenação, feedbacks, comunicação, planejamento e conhecimento. Assegurar a qualidade, motivação e remuneração.	H/T

Atomização: o CS é executado a partir do desenvolvimento de atividades por um grupo de participantes. Desta forma, os autores [Saremi and Yang 2015, Murray-Rust et al. 2015, Stol and Fitzgerald 2014] sugerem que para ocorrer o gerenciamento do CS é necessário minimizar ou atomizar as atividades. O processo de atomização cria o conceito de *microtasks* definidas como tarefas que podem ser executadas de forma independente e em curto prazo de tempo. Porém, a atomização de atividades pode gerar um conjunto grande de *microtasks*, dificultando o gerenciamento. Com isto, cada projeto deve ser analisado para estabelecer um critério de atomização. O fator de atomização foi considerado tecnológico, pois impactará no tamanho do projeto.

Documentação: os estudos de [Dwarakanath et al. 2015, Tsai et al. 2014] apresentaram que conceitos de documentação são necessários para gerenciamento das atividades. O projeto deve possuir um conjunto de manuais, diagramas ou materiais de apoio para a realização das *microtasks*. A documentação foi considerada como fator tecnológico, pois deve estar presente nas atividades do projeto.

Formação de grupos: o CS é formado por um grupo de pessoas com diferentes níveis de conhecimentos e habilidades. Projetos com diversidade de participantes podem oferecer uma gama maior de soluções, porém também maximizam os desafios de gerenciamento devido aos fatores culturais, fronteiras geográficas e o tamanho do grupo. Com isto, os estudos de [Murray-Rust et al. 2015, Stol and Fitzgerald 2014] sugerem que a formação do grupo possua uma coordenação para selecionar participantes que possam auxiliar, estejam engajados e familiarizados com o projeto. A formação de grupos refere-se a um fator humano de gerenciamento.

Comunicação: A comunicação é um fator humano de gerenciamento citada nos estudos de [Tsai et al. 2014, Saremi and Yang 2015, Murray-Rust et al. 2015]. Todos os participantes devem possuir acesso a um canal de comunicação para realizar interação entre o grupo. A prática da comunicação durante o CS é um processo para auxiliar o encontro de informações, tomada de decisões e verificar dúvidas ou soluções encontradas.

Feedbacks: O *feedback* é um processo circular de interação entre participantes e o(s) responsável(is) pelo desenvolvimento do projeto e foi citado nos estudos de [Saremi and Yang 2015, Murray-Rust et al. 2015, Stol and Fitzgerald 2014]. O *feedback*

deve ser utilizado como uma prática para verificar o andamento das *microtasks* e para os participantes apresentarem eventuais dúvidas ou sugestões do projeto. Do outro lado, ao(s) responsável(is) do projeto, cabe salientar pontos como correção de atividades e a visão geral do projeto. O *feedback* é um fator de gerenciamento humano.

A partir da compilação deste conjunto de práticas formado por fatores humanos e tecnológicos, foi gerado um modelo de gerenciamento do CS para projetos de software, apresentado na Figura 1.

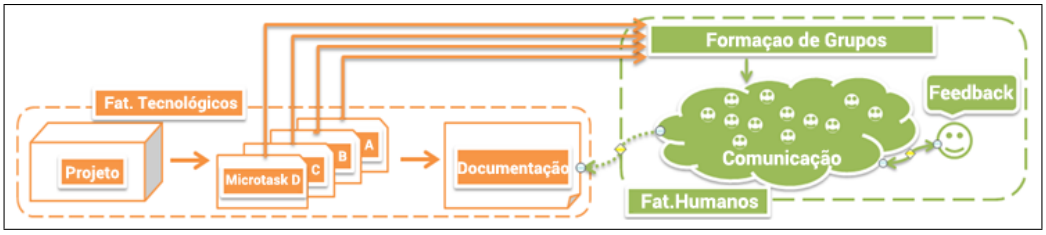

Figura 1. Modelo de Gerenciamento CS

O modelo foi organizado pela execução das cinco práticas anteriormente descritas. Após o projeto ser instanciado todas as atividades devem ser atomizadas. Para auxiliar o gerenciamento, as atividades atomizadas devem possuir documentação (diagramas, planilhas, casos de testes, etc...) que facilitem sua compreensão e execução. Após estas duas etapas é iniciado a formação de grupos, utilizando as *microtasks* e documentação como apoio para seleção de participantes. O projeto deve possuir um mecanismo que providencie a comunicação, como um chat ou wiki. Por fim, são performados *feedbacks* durante a execução das atividades entre o responsável do projeto e os participantes.

5. Validação do Modelo de Gerenciamento

Para efetuar o procedimento de validação e verificar o gerenciamento de CS foi utilizado a execução de um quase-experimento em uma iniciativa CS a fim de verificar o comportamento do modelo de gerenciamento previamente apresentado.

5.1. Iniciativa Crowdsourcing

A iniciativa pertence à utilização de robôs Legos Mindstorms para demonstração dos conceitos de gerenciamento de projetos e programação para os discentes dos cursos de graduação em Análise e Desenvolvimento de Sistemas e Engenharia da Computação da Universidade Tecnológica Federal do Paraná - Campus Cornélio Procópio.

Os robôs Lego Mindstorms possuem um conjunto de peças[1] que podem ser combinadas para a montagem de quase 25 modelos a partir da documentação oficial[2]. O processador permite o recurso de programação utilizando a linguagem Java e C por meio de um ambiente para aprendizado e ensino de programação. Os robôs Mindstorms apresentam possibilidades de apresentar noções de robótica, programação e trabalho em equipe.

Selecionamos uma turma de graduação que possuía alunos de Análise e Desenvolvimento de Sistemas e de Engenharia da Computação para verificar a execução do modelo fornecendo gerenciamento com a perspectiva dos fatores humanos e tecnológicos.

[1]Disponível em: http://arstechnica.com/gadgets/2013/08/review-lego-mindstorms-ev3-means-giant-robots-powerful-computers/

[2]Disponível em: http://www.lego.com/en-us/mindstorms/build-a-robot/

5.2. Execução do Quase-Experimento

A turma recebeu um treinamento de montagem e de programação do kit Lego Mindstorms. No início do semestre letivo foi concordado que uma das notas que integravam a média final dos alunos era composta pela participação em experimentos e quase-experimentos da disciplina, desta forma não foi necessário a assinatura de um termo de consentimento.

O quase-experimento foi executado em uma turma com os 27 discentes organizados em dois grupos. A composição dos grupos misturou os cursos, porém devido ao número ímpar uma equipe reteve mais participantes. Este desbalanceamento não foi percebido como um motivo de falha ou desafio na condução, visto que ambos os grupos concluíram o projeto de forma gerenciada e em tempos similares.

Para a configuração do ambiente CS estabeleceu-se as seguintes características: **Canal de comunicação:** Cada grupo deveria, obrigatoriamente, utilizar o chat eletrônico da plataforma acadêmica moodle[3]. **Dispersão:** Cada grupo era formado por dois papéis, estruturado como *Crowd* (equipe responsável por planejar estratégias) e *Crowdsourcer* (participante responsável por executar a estratégia). **Base de informação:** *Crowd* e *Crowdsourcer* possuíam acesso à documentação, porém apenas o *Crowdsourcer* possuía acesso às peças do robô. Com esta configuração enquanto o *Crowd* deveria adotar estratégias de montagem e enviá-las por chat ao *Crowdsourcer*, este deveria receber, executar e reportar o andamento do projeto para o *Crowd*. Com esta configuração foi verificado se o modelo poderia gerenciar o CS e realizar a construção de um robô, obedecendo as seguintes regras:

- O *Crowdsourcer* ficou afastado de seu respectivo *Crowd*.
- Todas as ordens deveriam acontecer pela troca de mensagens eletrônicas entre *Crowd* e *Crowdsourcer*.
- Todos os discentes tiveram acesso à documentação com as seguintes informações: código, imagem, quantidade, cor e categoria de todos os componentes do kit Lego.
- O robô deveria possuir o processador central e ao menos um motor.
- O *Crowd* deveria definir as estratégias de montagens e enviar os códigos de peças a serem encaixados para o *Crowdsourcer*.
- O *Crowdsourcer* deveria consultar o código na documentação, identificar a peça, realizar a montagem e enviar o *feedback* do encaixe e o andamento do projeto.

Não foi definido um limite mínimo de peças ou tempo para realizar a montagem do robô. A motivação foi trabalhada apenas na competição de qual grupo concluiria primeiro a montagem. O protocolo do quase-experimento é exposto na Tabela 3.

Para otimizar a aderência de *feedbacks*, a cada 30 minutos de duração do quase-experimento um membro do *Crowd* poderia deslocar-se até ao *Crowdsourcer* para verificar o andamento do projeto. Esta regra foi aplicada para evidenciar dois fatores: comunicação e *feedback*. O tempo máximo utilizado para a realização da montagem do robô foi de 44 minutos e 25 segundos. E somente um *feedback* deste estilo foi necessário para a conclusão do projeto, todos os outros *feedbacks* ocorreram via chat eletrônico.

[3]Disponível em: https://moodle.org

Tabela 3. Modelo de gerenciamento aplicado ao quase-experimento

Prática	Aplicação
Atomização	Para garantir a atomicidade, selecionamos o seguinte conjunto de *microtasks*: 1) *Crowd* deve identificar o componente; 2) *Crowd* deve enviar o código para *Crowdsourcer*; 3) *Crowdsourcer* deve identificar o componente; 4) *Crowdsourcer* deve realizar a montagem e 5) *Crowdsourcer* deve enviar feedback.
Documentação	A documentação foi aplicada a partir de uma planilha descritiva dos componentes do kit Lego Mindstorms, descrevendo o código, imagem, quantia, cor, categoria, nome e encaixes possíveis de cada peça
Formação de Grupos	Para garantir que os grupos fossem formados com certo engajamento, foram realizadas apresentações sobre a iniciativa Lego Mindstorms aos participantes, também ocorreu a apresentação da documentação descrita anteriormente. A formação de dois grupos ocorreu aleatoriamente, apenas o processo de seleção dos papéis a serem executados foram discutidos entre cada grupo.
Comunicação	O *Crowd* e seu respectivo *Crowdsourcer* deveriam comunicar-se através do chat eletrônico.
Feedbacks	Os feedbacks foram reportados pelos *Crowdsourcer* a cada execução de uma ordem.

6. Discussão e Análise

Os dois grupos apresentaram êxito para a conclusão do projeto, ambos concluíram o desafio utilizando a capacidade de inteligência coletiva do *Crowd* e do *Crowdsourcer*. Destacando ainda aderência ao conjunto de regras apresentadas.

Para a nomenclatura dos grupos optou-se por identificá-los como Grupo A e Grupo B. A primeira equipe que concluiu a montagem do robô foi o Grupo A utilizando um total de 39m e 24s. O Grupo A utilizou o total de 167 interações de mensagens eletrônicas durante todo o desenvolvimento do projeto. A segunda equipe, Grupo B, empregou 44m e 25s para o desenvolvimento do projeto, e utilizou 103 interações de mensagens eletrônicas. O produto final de ambas as equipes está apresentado na Figura 2.

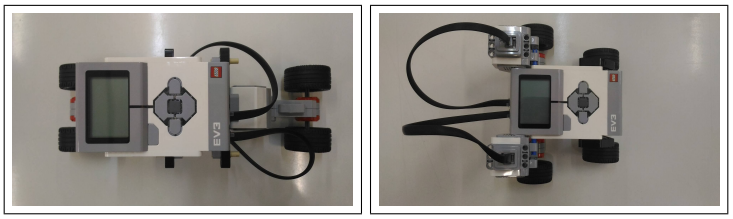

Figura 2. Estrutura final do robô do Grupo A (esquerda) x Grupo B (direita)

A análise individual dos log's do chat de cada grupo identificou que todas as ordens enviadas do *Crowd* para o *Crowdsourcer* foram executadas. Cada *Crowdsourcer* enviou o status do *feedback* sobre a montagem. Uma característica percebida se refere ao fato dos *Crowdsourcers* enviarem para o *Crowd* não somente o *feedback*, mas também sugestões sobre componentes para otimizar a montagem do robô. Fortalecendo e maximizando o ciclo presente nas práticas de Feedbacks e Comunicação apresentadas anteriormente.

Apesar de todas as ordens terem sido executadas, alguns componentes desenvolvidos não foram utilizados na montagem final do robô, devido ao fato da *Crowd* optar por outras táticas durante o desenvolvimento do projeto.

6.1. Atomização

A atomização foi executada a partir da definição de cinco *microtasks* (ver Atomização na Tabela 3) que deveriam ser executadas sequencialmente. Sua correta utilização forneceu mecanismos para o gerenciamento do CS, oferecendo balanceamento de *microtasks* entre o *crowd* e *crowdsourcer*.

6.2. Documentação

Ficou evidenciado que a documentação, neste caso a planilha descritiva dos componentes, foi uma prática importante para o gerenciamento do CS. Oferecendo mecanismos de autogestão para o *crowd* com a descrição de cada componente e facilitando a concepção de estratégias para o procedimento de montagem.

6.3. Formação de Grupos

A formação de grupos (ver Tabela 3) foi realizada para proporcionar aos participantes do quase-experimento um engajamento de conhecimento sobre os robôs Legos Mindstorms. A documentação contribuiu para a formação do grupo, visto que os participantes possuíam a mesma base conceitual para o desenvolvimento facilitando o gerenciamento.

6.4. Comunicação

Para garantir que houvesse constante comunicação durante o quase-experimento, os participantes utilizaram um chat eletrônico para a troca de mensagens no seu grupo. A comunicação foi explorada como forma de dispersar o andamento do projeto para todos os participantes e oferecer autogestão do *Crowd*. Oferecendo a possibilidade de visão geral do projeto entre todo o grupo, facilitando o gerenciamento do CS apesar da dispersão. Nós encontramos exemplos em que o *Crowd* perguntou se o *Crowdsourcer "compreendeu a estratégia de montagem"* e se poderia *"retornar o andamento da montagem"*.

6.5. Feedback

Apesar do quase-experimento possuir um mecanismo formal para a ocorrência de *feedbacks*, com o aumento da maturidade do projeto os *feedbacks* foram mais proveitosos. Cada *Crowdsourcer* percebeu que a sua função poderia auxiliar o desenvolvimento do projeto não somente executando ordens, mas apresentando soluções ou otimizações para as estratégias adotadas pelo *Crowd*. Um exemplo ocorreu quando o *Crowdsourcer* enviou ao *Crowd* se *"apenas faltava a base do robô"* e verificou *"se poderia utilizar uma determinada peça para concluir a montagem"*.

7. Conclusão

O CS é um emergente modelo de negócios para o desenvolvimento de projetos de software. Analisando a literatura sobre o gerenciamento do CS foi identificado um conjunto de práticas que foram compiladas em um modelo de gerenciamento do CS. O modelo pode ser aplicado como um guia prático para maximizar as chances de êxito em projetos de software.

Conduzimos um quase-experimento verificando o gerenciamento a partir dos aspectos humanos e tecnológicos em uma iniciativa CS que evidenciou o modelo do CS de competição e recrutamento. O projeto final apresentou resultados satisfatórios, gerenciando as atividades desenvolvidas, a colaboração dos participantes e a compreensão do projeto no ambiente disperso. Todos os grupos alcançaram o objetivo final dentro do projeto e apresentaram aderência as ordens preestabelecidas.

Nós acreditamos que este estudo contribuiu para a compreensão e tratamento de aspectos sociais e tecnológicos de gerenciamento em projetos de software aplicados em

ao novo modelo de desenvolvimento CS. O objetivo deste estudo em analisar o gerenciamento do CS verificou que a partir da utilização de um conjunto bem especificado de etapas e atividades, os fatores humanos e tecnológicos do CS podem ser executados com a atomização de atividades, documentação, formação de grupos, realização de *feedbacks* e comunicação entre o grupo.

7.1. Limitações e Trabalhos Futuros

Os projetos de software podem oferecer desafios maiores aos simulados no quase-experimento, com isto os fatores de atomização, documentação, formação de grupos, feedbacks e comunicação podem ter sido limitadas. Com isto, o modelo pode oferecer insuficiências quando aplicado em determinados cenários. Um risco percebido refere-se ao *Crowdsourcer* executar ordens sem avisar o *Crowd*, constatado somente com a análise dos log's do chat. Outro adendo é que não foi aplicado conceitos de programação.

Uma contribuição futura refere-se ao amadurecimento do conceito de gerenciamento de CS em projetos de software, através de experimentações com o modelo apresentado neste estudo.

Referências

Benedek, A., Molnar, G., and Szuts, Z. (2015). Practices of crowdsourcing in relation to big data analysis and education methods. In *Intelligent Systems and Informatics (SISY), 2015 IEEE 13th International Symposium on*, pages 167–172.

Cullina, E., Conboy, K., and Morgan, L. (2016). Choosing the right crowd: An iterative process for crowd specification in crowdsourcing initiatives. In *2016 49th Hawaii International Conference on System Sciences (HICSS)*, pages 4355–4364.

Dwarakanath, A., Chintala, U., Shrikanth, N. C., Virdi, G., Kass, A., Chandran, A., Sengupta, S., and Paul, S. (2015). Crowd build: A methodology for enterprise software development using crowdsourcing. In *CrowdSourcing in Software Engineering (CSI-SE), 2015 IEEE/ACM 2nd International Workshop on*, pages 8–14.

Dwarakanath, A., Shrikanth, N. C., Abhinav, K., and Kass, A. (2016). Trustworthiness in enterprise crowdsourcing: A taxonomy & evidence from data. In *Proceedings of the 38th International Conference on Software Engineering Companion*, ICSE '16, pages 41–50, New York, NY, USA. ACM.

Hossain, M. (2012). Crowdsourcing: Activities, incentives and users' motivations to participate. In *Innovation Management and Technology Research (ICIMTR), 2012 International Conference on*, pages 501–506.

Hosseini, M., Phalp, K., Taylor, J., and Ali, R. (2014). The four pillars of crowdsourcing: A reference model. In *2014 IEEE Eighth International Conference on Research Challenges in Information Science (RCIS)*, pages 1–12.

Hosseini, M., Shahri, A., Phalp, K., and Ali, R. (2015). Recommendations on adapting crowdsourcing to problem types. In *2015 IEEE 9th International Conference on Research Challenges in Information Science (RCIS)*, pages 423–433.

Howe, J. (2006). The rise of crowdsourcing. *Wired magazine*, 14(6):1–4.

Kucherbaev, P., Daniel, F., Tranquillini, S., and Marchese, M. (2016). Crowdsourcing processes: A survey of approaches and opportunities. *IEEE Internet Computing*, 20(2):50–56.

LaToza, T. D. and van der Hoek, A. (2016). Crowdsourcing in software engineering: Models, motivations, and challenges. *IEEE Software*, 33(1):74–80.

Mattauch, T. (2013). Innovate through crowd sourcing. In *Proceedings of the 41st Annual ACM SIGUCCS Conference on User Services*, SIGUCCS '13, pages 39–42, New York, NY, USA. ACM.

Murray-Rust, D., Scekic, O., Papapanagiotou, P., linh Truong, H., Robertson, D., and Dustdar, S. (2015). A collaboration model for community-based software development with social machines. *EAI Endorsed Transactions on Collaborative Computing*, 1(5).

Pan, Y. and Blevis, E. (2011). A survey of crowdsourcing as a means of collaboration and the implications of crowdsourcing for interaction design. In *Collaboration Technologies and Systems (CTS), 2011 International Conference on*, pages 397–403.

Pedersen, J., Kocsis, D., Tripathi, A., Tarrell, A., Weerakoon, A., Tahmasbi, N., Xiong, J., Deng, W., Oh, O., and de Vreede, G. J. (2013). Conceptual foundations of crowdsourcing: A review of is research. In *System Sciences (HICSS), 2013 46th Hawaii International Conference on*, pages 579–588.

Saremi, R. L. and Yang, Y. (2015). Dynamic simulation of software workers and task completion. In *CrowdSourcing in Software Engineering (CSI-SE), 2015 IEEE/ACM 2nd International Workshop on*, pages 17–23.

Satzger, B., Zabolotnyi, R., Dustdar, S., Wild, S., Gaedke, M., Gobel, S., and Nestler, T. (2014). Chapter 8 - toward collaborative software engineering leveraging the crowd. In Mistrik, I., , Bahsoon, R., , Kazman, R., , and Zhang, Y., editors, *Economics-Driven Software Architecture*, pages 159 – 182. Morgan Kaufmann, Boston.

Schwartz, C., Borchert, K., Hirth, M., and Tran-Gia, P. (2015). Modeling crowdsourcing platforms to enable workforce dimensioning. In *Telecommunication Networks and Applications Conference (ITNAC), 2015 International*, pages 30–37.

Stol, K.-J. and Fitzgerald, B. (2014). Researching crowdsourcing software development: Perspectives and concerns. In *Proceedings of the 1st International Workshop on CrowdSourcing in Software Engineering*, CSI-SE 2014, pages 7–10, New York, NY, USA. ACM.

Tajedin, H. and Nevo, D. (2013). Determinants of success in crowdsourcing software development. In *Proceedings of the 2013 Annual Conference on Computers and People Research*, SIGMIS-CPR '13, pages 173–178, New York, NY, USA. ACM.

Tsai, W. T., Wu, W., and Huhns, M. N. (2014). Cloud-based software crowdsourcing. *IEEE Internet Computing*, 18(3):78–83.

Zakariah, Z., Janom, N., and Arshad, N. H. (2015). Business model of crowdsourcing: Review paper. In *2015 IEEE 6th Control and System Graduate Research Colloquium (ICSGRC)*, pages 66–69.

Applying Software Craftsmanship Practices to a Scrum Project: an Experience Report

Percival Lucena, Leonardo P. Tizzei

[1] IBM Research
São Paulo - Brazil

{plucena,ltizzei}br.ibm.com

Abstract. *The Software Craftsmanship manifesto has defined values and principles that software development teams should follow to deliver quality software that fulfills functional and non-functional requirements without dealing with high amounts of technical debt. Software craftsmanship approach to software development prioritizes technical practices in order to provide a clean code base. This work analyzes a set of practices that can be applied to a Scrum project that aims to incorporate Software Craftsmanship values. The process implementation described may be a useful contribution for software development teams who also intend to implement Software Craftsmanship on their projects.*

1. Introduction

The 2015 CHAOS report [Hastie and Wojewoda 2015] published by Standish Group has analyzed more than 50,000 software development projects from different sizes and complexities. Although the projects that have adopted Agile software development methodology had a higher success rate than those that had adopted Waterfall approach, more than 40% of Agile projects had problems related to incomplete scope, low quality, or have exceeded the estimate delivery time. In the last few years, several software development companies have adopted Agile Software development methodologies, most of them adopting the Scrum framework [VersionOne 2016]. According to the study published by [Melo et al. 2013], those companies face similar problems as described in the CHAOS report. The high number of unsuccessful Scrum projects suggests that the successful framework use requires specific conditions to be met.

In order to maximize the number of successful Agile projects, several metrics that measure some quality aspects of the code have been proposed. Technical debt is one of these metrics and it is often used in current Agile projects. It was introduced by [Cunningham 1993] to define incomplete software artifacts that barely satisfy the functional requirements. Under these circumstances, adding features to the software, or fixing defects, requires interest by writing more code to complete features that should already be available in prior released versions. Technical debt is also related to software that does not meet non-functional requirements such as performance, security, and usability.

According to the Scrum Guide [Schwaber and Sutherland 2011], the Product Owner is responsible for maximizing the value of the product and the work of the development team. The Scrum Guide itself does not define how the Product Owner should prioritize the backlog stories. In many tight schedule projects, the Product Owner may decide to

maximize the number of delivered stories by reducing or even eliminating the quality assurance related tasks and other technical activities required to implement non-functional requirements, increasing the overall technical debt. Martin Fowler has identified this problem and described it as Flaccid Scrum [Fowler 2009]. The [VersionOne 2016] report has identified this issue as one of the main causes of failure of Scrum projects .

Software Craftsmanship presents an alternative craft model that places people at the center of the software development process [McBreen 2002]. Although Software Craftsmanship presents useful values to tackle some of the Scrum limitations, these values are abstract, and few practical guidelines are available in the literature. Due to this lack of guidelines to adopt Software Craftsmanship values, developers often adopt *ad-hoc* approaches. This paper presents an experience report that describes a set of guidelines, which include practices and tools, to adopt Software Craftsmanship values on a Scrum project. These guidelines have been applied to a real world project. Besides investigating technical practices that can help reduce technical debt, it also analyzes the impact Software Craftsmanship can bring on the organization including the role and responsibility of the team, the customers, the Product Owner and the ethical questions and trade offs involved in delivering software with quality.

The rest of this paper is structured as follows: Section 2 presents basic concepts of Software Craftsmanship and Section 3 describe practices and tools that were used to adopt Software Craftsmanship values on a Scrum project. Section 4 discusses benefits and limitations of practices and tools described in previous sections and Section 5 concludes and suggests future work.

2. Software Craftsmanship Overview

This section introduces the Software Craftsmanship software development approach followed by a brief discussion of related work.

2.1. Background

[McConnell 1998] has discussed if software development should be considered art, craft, engineering or science. Although the craftsmanship metaphor is disputable, the related movement brought to light important discussions about the importance of adopting good technical practices as a part of the software development methodology [Bria 2008]. The movement has started in December 2008, when the Software Craftsmanship Summit was held in Chicago, Illinois establishing a set of principles for Software Craftsmanship. In March, 2009, after an online group conversation, Doug Bradbury wrote a summary of the general conclusions in the form of a Manifesto for Software Craftsmanship [Pagel 2009].

Table 1. Software Craftsmanship Manifesto compared to Agile Manifesto

Software Craftsmanship	Agile
Well-crafted software	Working software
Steadily adding value	Responding to change
Community of professionals	Individuals and interactions
Productive partnerships	Customer collaboration

Table 1 presents a comparison between Agile Manifesto values the Software Craftsmanship manifesto. The first value described on the Craftsmanship manifesto offered an alternative approach for the lack of quality in Agile projects as described by

[Martin 2008b] on Agile conference keynote. The manifesto also aims at extending the original Agile manifesto values by proposing new ways to deliver software, to organize teams, and to deal with customer demands.

2.2. Related Work

The seminal book about Software Craftsmanship [McBreen 2002] describes concepts that help differentiate Software Craftsmanship from traditional Software Engineering. Although this work defines several quality assurance concepts that could be added to traditional Agile Software development, it does not provide details on how the software development teams could implement such concepts. [Winter 2015] offers a more concrete view on how to implement Software Craftsmanship on a Agile Project based on clean code concepts [Martin 2008a] [Martin 2011] and Extreme Programming techniques [Beck 2004].

[Oliveira et al. 2015] have applied a framework to identify and measure technical debt on Scrum projects, but the authors do not provide guidelines on how to improve the Scrum process to minimize technical debt. [Brown et al. 2010] have analyzed technical debt causes to Scrum projects and suggest using Extreme Programming and Software Craftsmanship technical practices to the project. [Mushtaq and Qureshi 2012] also propose integrating Extreme Programming techniques to the Scrum project in order to reduce technical debt, which is only one of the several goals of the craftsmanship manifesto. This work will discuss a broader set of aspects including software delivery, team organization and customer relationship.

3. Software Craftsmanship Applied to Scrum

This section describes how we applied software craftsmanship to a Scrum project. An overview of this Scrum project is presented in Section 3.1. Sections 3.2-3.5 describe how existing practices and tools have been used to adopt Software Craftsmanship. The goal is to report a set of guidelines for developers willing to implement quality code according to Software Craftsmanship values. These guidelines were developed based on the experience of developers and specificities of the aforementioned Scrum project.

3.1. Target Application

An e-commerce application was developed for a telecommunications company that handles customized mobile devices through its sales channel. The application was designed to be available on web and mobile devices through its own website, but also embedded as a multi-tenant application inside third party e-commerce websites. The developer team was composed of ten developers, an architect, a Scrum master, and a Product Owner. The team members were distributed in United States and in Brazil and they developed the application in 12 2-week sprints. The application back-end was developed in Java and Ruby and the front-end was developed using AngularJS. Software developers had good knowledge of technologies involved and the company has traditionally adopted software engineering best practices.

3.2. Well-crafted software vs Working Software

The Agile Manifesto [Fowler and Highsmith 2001] values working software over comprehensive documentation. Many agile teams believe architecture activities are part of the

system documentation and therefore are not essential to the project. Nevertheless, system architecture is a key activity for creating mechanisms that enable a software to meet all the non-functional requirements. Weak architectures often result in systems with high technical debt.

In order to incorporate good architecture practices to our Scrum project our team has adopted Scott Ambler's Agile Model Driven Development (AMDD) [Ambler 2003] which proposes to include modeling tasks to Scrum stories. According to AMDD, models should be created before coding. An initial domain based model was also created at Sprint 0 together with the definition of the basic architecture mechanisms. We have combined the practice with Model Driven Architecture (MDA) tools capable of generating basic code from models and keep them in sync with the source code base.

Source code quality is also an essential attribute of well-crafted software. Martin's "clean code principles" [Martin 2008a] is a practical guide to implement Software Craftsmanship code quality practices. Clean code principles define coding standards, rules for unit tests creation, and help to define the system architecture.

A small subset of the "clean code principles" was implemented by our team using the SonarQube static analysis tool. This tool is based on the Squale method [Mordal-Manet et al. 2009] which defines practices as an intermediate level between quality metrics and criteria. A Squale practice abstracts the extracted code information combining and weighting different user defined metrics. Squale practices cover documentation, programming conventions, and test coverage. The SonarQube tool were run as part of the code review process. Figure 1 shows standard code analysis reports generated by this tool. The software development team has agreed to follow SonarQube Standards. No source code pull request was accepted in case it presented any blocker severity issue.

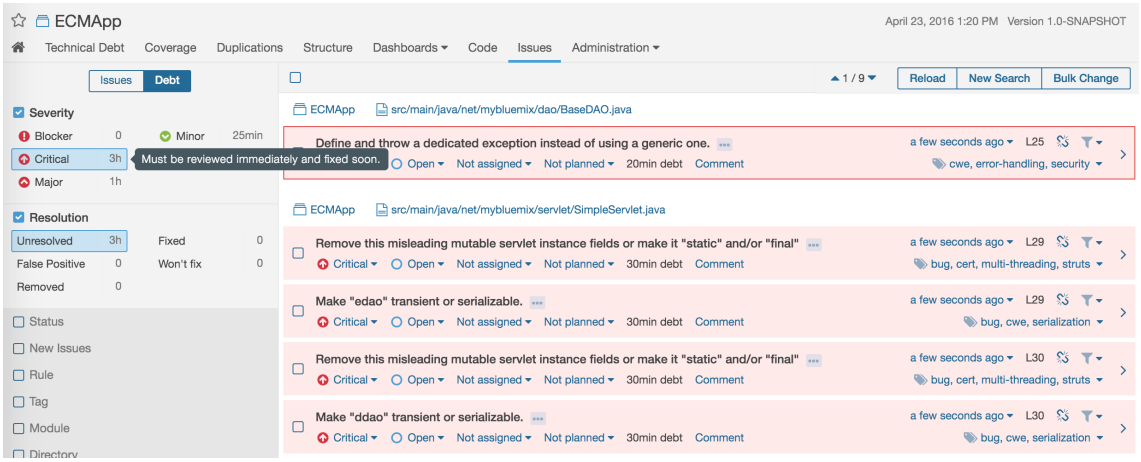

Figure 1. Static Code Analysis using SonarQube

Despite teams effort to eliminate technical debt, according to [Oliveira et al. 2015] a small amount of it is inevitable in project. Since there is no simple way of eliminating technical debt completely, our project team has decided to measure and manage it. During the project Sprints, the gathered technical debt was measured in the project by logging

hours spent on technical debt tasks separated from the hours spent on backlog tasks. Each type of task was kept on a separate backlog to help better manage it and visualize the tasks. Although many times, the technical debt tasks could not be paid immediately, the team made some effort to slow down the technical debt growth rate. Figure 2 shows total technical debt accumulated measured by SonarQube tool and the total numbers of hours worked on backlog stories. The gathered data suggests that team velocity slowed down as the technical debt hours grew because the team spent more hours on technical debt tasks and less time on backlog stories tasks.

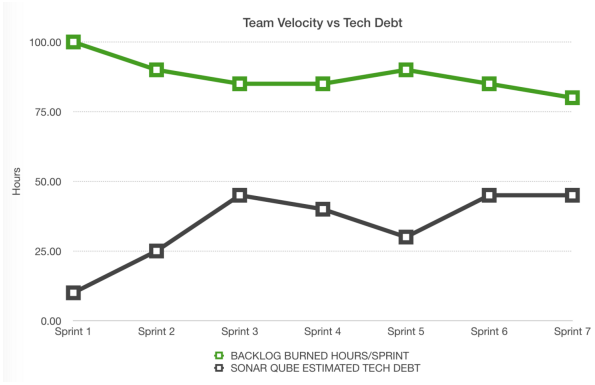

Figure 2. Team Velocity measured in backlog worked hours vs SonarQube Total Technical Debt Hours

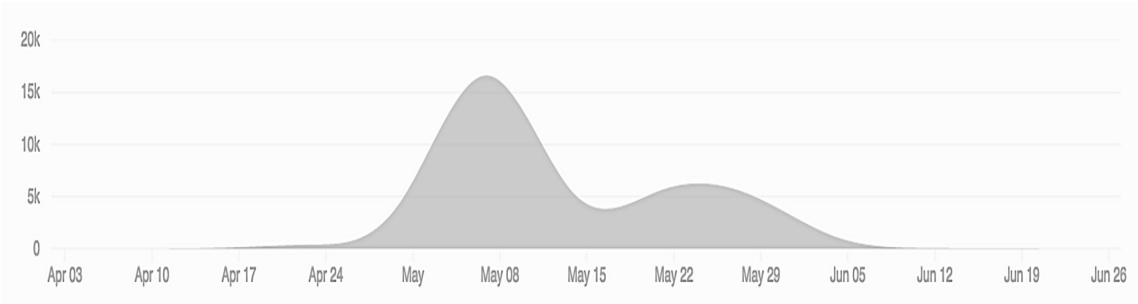

Figure 3. Number of code lines deleted over time

Another metric used to estimate technical debt gathered was the number of code lines deleted from the git source code repository. Although some re-factoring was often needed we have inferred that big changes were usually derived from poor planing or loose understanding of the requirements. Figure 3 shows source code deletions accumulated during a certain period of time. Despite the fact this technical debt metric is less accurate than the one provided by the static analysis source code tool it can be easily extracted from git repositories and can be used as an indicator to track the effort spent paying technical debt.

In order to improve source code quality and thus reduce technical debt our team has adopted a few Extreme Programming technical practices [Beck 2004] to the Scrum based software development process. The co-located team was organized in a way that team members could do Pair Programming by rotating positions in the room. We also

have implemented a mandatory code review process using git pull requests. Once a pull request is sent to the version control system, someone who is available on team reviews the set of changes, discuss the modifications, and even push follow-up commits if necessary. During our Sprint retrospective meetings we have learned that the code review process has helped the team to improve the application architecture and fix potential defects that were not previously covered neither by our unit tests nor by SonarQube tool.

3.3. Steadily adding value vs Responding to change

Scrum's short Sprints iterations allows Product Owners to re-prioritize backlog stories, allowing the team to respond to business changes. Sprint reviews are good opportunities for the team to present the increment developed on the Sprint. Unfortunately, the software presented on Sprint reviews is sometimes executed in hardware environments different from the customer real enterprise environment and it includes mocks to external system interfaces. When the software is finally deployed in the real customer environment one often finds integration and performance issues which could be prevented earlier. Late software deployments also limit useful feedback for the development team and delays customer return of the investment.

Extreme Programming has addressed this problem though the Continuous Integration technical practice. This practice focused on integrating and test source code changes as often as possible [Beck 2004]. The Devops movement [Bass et al. 2015] has evolved this concept into the Continuous Delivery process that allows the software to be delivered automatically to its hosting environments. Devops adoption requires a paradigm change. The Devops movement has worked on a set organizational culture that removed the traditional Operation processes that manually controlled the infrastructure responsible for running IT projects.

Due to organizational or technical constraints a large percentage of Agile teams still do not implement a continuous delivery process [VersionOne 2016]. Continuous deployments offer constant feedback, helping the team to focus on issues that can not be foreseen by the Product Owner and that can only be discovered in real use cases. The Devops practice is usually implemented with pipelines which consist of an automated process that executes tasks such as building the source code, packaging the dependent libraries, run unit, integration and performance tests and deploy the resulting software to a specific environment. A Devops pipeline is usually composed of several deployment environments used for different proposes.

Our project team has created separate environments for development, testing, preproduction, and production. The process was automated using Jenkins Multi-Tenant as Continuous Integration tool. Git pull requests trigger our pipeline which runs a maven script to execute a Junit test-suite. In case the build breaks, the team receives a message so they can fix the broken issues. In case no test fails, a new executable package is created and then deployed to the first development stage of the Devops pipeline. Then, Selenium integration tests and JMeter performance tests are run against the development environment. In case all the tests succeed, the pipeline allows to move the new artifact to the remaining environments.

All the team members receive quick feedback on the results of the changes. The automated pipeline helped the team to improve the confidence level for deployments

which most of times took only a few minutes to complete. The agility provided by the Devops process allowed the Product Owner to verify and use the software latest releases providing the team useful feedback about the ongoing Sprint. Production deployments occurred flawlessly providing value to the end user as soon as possible.

3.4. Community of professionals vs Individuals and interactions

The Agile manifesto [Fowler and Highsmith 2001] values Individuals and Iterations over processes and tools. Based on such concept, the development team could choose what practices are the best to deliver working software without wasting time with processes that do not aggregate value to the customer. Unfortunately, some Agile lightweight frameworks, such as Scrum, do not define what technical practices should be adopted [Fowler and Highsmith 2001]. In some Scrum projects the Scrum Master and the Product Owner may also interfere in the project organization, preventing the software development team to self organize and decide which software development process should be followed. In such case, a professional attitude is required from the software development team who should commit to its code of ethics, being able to deliver quality software, no matter how the enterprise is organized and what internal and external demands are in place.

[Mancuso 2015] suggests that Software Craftsman practitioners should embrace a code of ethics to guarantee professionalism in software development activities. The code of ethics should protect the development team with principles that would be strictly followed under any circumstances. By following a code of ethics, self organizing teams should be able to define their development process with lesser external interference. A cultural change in relationship with customers is also necessary. Software developer teams should not be intimidated to change their set of useful technical practices despite of any external pressures on their work.

Our project team has adopted the ACM Code of Ethics [Association for Computing Machinery 2016] as the base for our nonnegotiable principles and work rules. This code of ethics shares several points in common with Software Craftsmanship professionalism values including the commitment to achieve the highest quality of work, acquire, and maintain professional competence and to be honest and trustworthy.

3.5. Productive partnerships vs Customer collaboration

The Agile Manifesto [Fowler and Highsmith 2001] values customer collaboration over contract negotiation. In traditional software development contracts we have triple constraints: cost, time, and schedule. The Scrum framework provides a prioritized backlog that changes over the Sprints executions. It is not feasible to change the contracts, every time the Product backlog changes. Traditional software development contracts should have a big slack in order to accommodate those changes.

Time and material contracts are arrangements under which the software development contractors are paid on the basis of the worked hours agreed upon fixed add-on to cover the contractor's overheads and profit. Time material contracts are used on other industries who require open scope activities and fit well to backlog changes required on Agile based development. By establishing a flexible work model the software develop-

ment team can work together with customers implementing valuable stories as needed without major concerns about exceeding fixed budgets and schedules.

On top of collaborating with the customer, the software development team should establish a productive partnership with the customer which requires engagements from the mutual parties. For the sake of successfully applying Software Craftsmanship values to a Scrum project, one needs to rethink the relationship among the software development team, the management and the customers. Extreme Programming offers the Planning Game technical practice to help the software development team to prioritize the backlog together with the Product Owner. The software development team estimates candidate stories for the next Sprint so the Product Owner can choose among the most valuable set of stories that can be added to the Sprint. Estimates are created as late as possible so they are based on the best possible information. The planning game has helped our software development team to have a closer partnership with our customer by providing the most valuable stories to the Sprints.

Towards the goal of creating a valuable product for the customer the software development team should also be committed to understand the customer business to provide the best solution for the problem. Backlog stories written by a single Product owner might not provide all the necessary information about the business scenarios. Domain driven design [Evans 2004] proposes that software developers and the customer Domain Experts should collaborate to create an accurate description and model of the domain problems. This cultural change involves a true partnership between the customer organization and the software development team so they can work together and share responsibilities about the project. Domain Driven Design also require the software developers to speak a ubiquitous language so the software model represents the same business concept on their bounded context. Domain Driven Design was implemented by our team as part of the Sprint Planning and also as part of Agile Modeling tasks added to the stories implementation. We found that those extra activities helped to reduce technical debt because stories details would be defined before hand reducing the re-factoring tasks required by incomplete or inaccurate requirements and models.

4. Discussion

[Jacobson and Seidewitz 2014] consider Scrum an incomplete software development methodology as it does not provide technical practices required by the software development team to create a quality product. As a result Scrum teams should define their own set of technical practices based on their needs. As discussed on previous sections, our team has identified a set of practices congruent to Software Craftsmanship values that complement the Scrum framework. Table 2 summarizes the technical practices adopted by our team and the life cycle phase in which they were used. Different development teams may find other technical practices more suitable for their working context.

We have evaluated our practices regarding the technical debt gathered through the Sprint. We have kept technical debt tasks separated from the Sprint backlog stories in order to understand the impact of those tasks in the Sprint execution. Static code analysis and code re-factoring measures provided useful indicators for the accumulated technical debt. This approach can be followed by other project teams who need to track the impact of adopting a specific set of technical practices in their projects.

Table 2. Software Craftsmanship Practices added to the Scrum Process

#	Technical Practice	Life Cycle Phase
1	Code of Ethics	All phases
2	Planning Game	Planning
3	Agile Modeling	Development
4	Domain Driven Design	Development
5	Coding Standards	Development
6	Static Code Analysis	Development
7	Code Review	Development
8	Pair Programming	Development
9	Automated Unit Testing	Testing
10	Automated Integration Testing	Testing
11	Automated Performance Testing	Testing
12	Automated Build and Deploy	Deploy

5. Conclusions

Software craftsmanship goal is to improve existing agile manifesto principles raising the bar of quality level delivered to software development projects. This paper has discussed a set of technical practices that can be added to the Scrum project in order to implement all the software craftsmanship manifesto values. Our approach has extended the work proposed by [Mushtaq and Qureshi 2012] that incorporated Extreme Programming practices to Scrum. Our approach has added more practices from different sources including Agile modeling, Devops, and from Software Craftsmanship. The combination of all those different practices have helped our team to define requirements more accurately through agile modeling, improve the source code quality through code standards, code reviews and static code analysis, and do extensive automated testing and deployment.

Although the team velocity at the first Sprints was smaller than similar projects who had adopted fewer technical practices, we have noticed that our team had a more stable velocity through the project. Overall, the team and was able to deliver a more valuable project to the customer with a higher quality than previous projects.

References

Ambler, S. W. (2003). Agile model driven development is good enough. *Software, IEEE*, 20(5):71–73.

Association for Computing Machinery (2016). Software engineering code of ethics and professional practice. `http://www.acm.org/about/se-code\#full`.

Bass, L., Weber, I., and Zhu, L. (2015). *Devops: A Software Architect's Perspective*, chapter 1, pages 10–15. Pearson Education (US).

Beck, K. (2004). *Extreme Programming Explained: Embrace Change, 2nd Edition*. The XP Series. Addison-Wesley.

Bria, M. (2008). Craftsmanship—the fifth agile manifesto value. *InfoQ, Aug*, 20.

Brown, N., Cai, Y., Guo, Y., Kazman, R., Kim, M., Kruchten, P., Lim, E., MacCormack, A., Nord, R., Ozkaya, I., et al. (2010). Managing technical debt in software-reliant systems. In *FSE/SDP Workshop on Future of software engineering research*, pages 47–52. ACM.

Cunningham, W. (1993). The wycash portfolio management system. *ACM SIGPLAN OOPS Messenger*, 4(2):29–30.

Evans, E. (2004). *Domain-driven design tackling complexity in the heart of software*. Addison W.

Fowler, M. (2009). Flaccid scrum.

Fowler, M. and Highsmith, J. (2001). The agile manifesto. *Software Development*, 9(8):28–35.

Hastie, S. and Wojewoda, S. (2015). Standish group 2015 chaos report. `bit.ly/1JOlpiW`.

Jacobson, I. and Seidewitz, E. (2014). A new software engineering. *Communications of the ACM*, 57(12):49–54.

Mancuso, S. (2015). *The Software Craftsman: Professionalism, Pragmatism, Pride*, chapter 1, pages 24–36. Prentice Hall.

Martin, R. (2008a). *Clean Code: A Handbook of Agile Software Craftsmanship*. Prentice Hall.

Martin, R. (2008b). Quintessence: The fifth element for the agile manifesto. *bit.ly/1WRAGeL*.

Martin, R. (2011). *The Clean Coder: A Code of Conduct for Professional Programmers*. Robert C. Martin Series. Pearson Education.

McBreen, P. (2002). *Software craftsmanship: The new imperative*. Addison-Wesley Professional.

McConnell, S. (1998). The art, science, and engineering of software development. *Software, IEEE*, 15(1):120–118.

Melo, C. d. O., Santos, V., Katayama, E., Corbucci, H., Prikladnicki, R., Goldman, A., and Kon, F. (2013). The evolution of agile software development in Brazil. *Journal of the Brazilian Computer Society*, 19(4):523–552.

Mordal-Manet, K., Balmas, F., Denier, S., Ducasse, S., Wertz, H., Laval, J., Bellingard, F., and Vaillergues, P. (2009). The squale model—a practice-based industrial quality model. In *International Conference on Software Maintenance*, pages 531–534. IEEE.

Mushtaq, Z. and Qureshi, M. R. J. (2012). Novel hybrid model: Integrating Scrum and XP. *International Journal of Information Technology and Computer Science*, 4(6):39.

Oliveira, F., Goldman, A., and Santos, V. (2015). Managing technical debt in software projects using scrum: An action research. In *Agile Conference*, pages 50–59. IEEE.

Pagel, P. (2009). History of the software craftsmanship manifesto. `bit.ly/1TGsLza`.

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

VersionOne (2016). 7th annual state of agile development survey.

Winter, B. (2015). *Agile Performance Improvement: The New Synergy of Agile and Human Performance Technology*, chapter 5: The Agile Software Engineer's Toolkit. Apress.

Uma avaliação de ferramentas de análise de sentimentos aplicadas a comentários da plataforma GitHub

Giuseppe Portolese[1], **Guilherme A. M. da Cruz**[1],
Elisa H. M. Huzita[1], **Valéria D. Feltrim**[1]

[1]Departamento de Informática – Universidade Estadual de Maringá (UEM)
Av. Colombo, 5.790 – 87020-900 – Maringá - PR - Brasil

{giuportolese, guilherme.maldonado.cruz}@gmail.com
{emhuzita, vfeltrim}@din.uem.br

Abstract. *Distributed software development has become frequent and the interaction between those involved, which is often influenced by social and cultural aspects, reflects in the performance of the teams. Sentiment analysis has been used to capture subjective information and get a better understanding of the interactions of these teams. Therefore, it is interesting to evaluate the performance of available tools when applied to that domain. In this work nine sentiment analysis tools were evaluated using GitHub comments manually annotated according to their polarity. Results showed that SentiStrength performed best among the evaluated tools, but with average performance below 50%.*

Resumo. *O desenvolvimento distribuído de software tem se tornado frequente e a interação entre os envolvidos, muitas vezes influenciada por aspectos sociais e culturais, reflete no desempenho das equipes. A análise de sentimentos vem sendo empregada para capturar informações subjetivas e obter um maior entendimento das interações dessas equipes. Portanto, é interessante avaliar o desempenho das ferramentas disponíveis quando aplicadas a esse domínio. Neste trabalho nove ferramentas de análise de sentimentos foram avaliadas usando comentários extraídos da plataforma GitHub e que foram manualmente anotados quanto à polaridade. Os resultados mostraram que a ferramenta SentiStrength se saiu melhor, porém com desempenho médio abaixo de 50%.*

1. Introdução

A distribuição geográfica dos membros caracteriza o desenvolvimento distribuído de software (DDS) e tem como objetivo trazer benefícios, como a facilidade em encontrar mão de obra qualificada, redução dos custos e agilidade na entrega dos produtos por meio da utilização do desenvolvimento *follow-the-sun* [O'Conchuir et al. 2006]. Contudo, a distância acrescenta desafios ao desenvolvimento de software que podem ser divididos, de forma geral, em socioculturais e técnicos. Desafios socioculturais englobam comunicação limitada e diferenças culturais, entre outros. Os desafios técnicos, por sua vez, referem-se a problemas com a rede, segurança das informações, processos/ferramentas de trabalhos diferentes, entre outros [Herbsleb and Moitra 2001, Sengupta et al. 2006].

Devido aos desafios oriundos da distribuição geográfica, diversos estudos e ferramentas visam auxiliar essas equipes e diminuir os efeitos da distância, tais como fóruns,

wikis, ferramentas de mensagens instantâneas, de videoconferência, para acompanhamento de atividades, monitoramento da equipe e de versionamento. Alguns desses estudos têm usado a análise de sentimentos sobre artefatos e comunicações de equipes desenvolvimento como forma de capturar informações subjetivas e, assim, buscar um maior entendimento a respeito das interações e emoções dos membros dessas equipes. Por exemplo, Guzman et al. [2014] empregam análise de sentimentos em artefatos produzidos ao longo de projetos a fim de obter o clima emocional dos membros envolvidos. Sinha et al. [2016] analisaram *commit logs* do GitHub usando análise de sentimentos e buscaram relações com os dias da semana e a quantidade de modificações. Cruz et al. [2016] usaram análise de sentimentos como parte de um *framework* para estimar a confiança entre membros de equipes de DDS.

Embora ferramentas de análise de sentimentos estejam sendo empregadas nesses e em outros estudos no contexto do desenvolvimento distribuído de software, em geral, essas ferramentas não foram criadas visando a aplicação nesse domínio. O mais comum é que as ferramentas tenham foco na análise de textos provenientes de redes sociais e *reviews*, e, por isso, tendem a não ter o mesmo desempenho quando aplicadas ao domínio de DDS [Tourani et al. 2014, Sinha et al. 2016]. Jongeling et al. [2015] argumentam ainda que dependendo da escolha da ferramenta, o estudo pode obter conclusões contraditórias, uma vez que além da baixa precisão, as ferramentas podem discordar entre si.

Tendo em vista esse cenário, este trabalho teve por objetivo avaliar diferentes ferramentas de análise de sentimentos no contexto das comunicações entre membros de equipes distribuídas. Como fonte de dados utilizamos a plataforma GitHub, mais especificamente, comentários feitos em *pull requests*, uma vez que esse tipo de comentário tem sido utilizado em vários trabalhos no contexto de DDS [Sinha et al. 2016, Guzman et al. 2014, Cruz et al. 2016]. O restante deste artigo está organizado em quatro seções. Na Seção 2 são descritos trabalhos que empregaram análise de sentimentos no contexto de desenvovimento de software. Na Seção 3 é descrita a metodologia, incluindo os dados utilizados e as ferramentas avaliadas. Na Seção 4 são apresentados os resultados e, por fim, na Seção 5, são apresentadas as conclusões e direções para possíveis trabalhos futuros.

2. Motivação

A análise de sentimentos (AS) é uma área de pesquisa ampla e interdisciplinar que diz respeito ao estudo de opiniões, sentimentos, atitudes e emoções. Dentre as diferentes tarefas tratadas por ferramentas de AS, a mais comum é a de determinar a polaridade de um texto – se positiva, negativa ou neutra – e, em geral, se estabelece em um de três níveis: (i) nível de documento, (ii) nível da sentença e (iii) nível da entidade ou aspectos. A área tem recebido muita atenção por parte da comunidade científica e tem encontrado aplicações em quase todos os domínios [Liu 2012].

No contexto do desenvolvimento de software, pesquisadores vêm empregando análise de sentimentos com o objetivo de entender e capturar aspectos subjetivos relativos à comunicação e ao relacionamento dos membros dessas equipes. Essa análise é especialmente interessante no caso das equipes de DDS, uma vez que a comunicação mediada por computador, que é característica dessas equipes, é uma fonte abundante de dados para a análise de sentimentos.

Guzman et al. [2014] usaram análise de sentimentos em comentários de *com-*

mits de 90 projetos da plataforma GitHub, desenvolvidos em diferentes linguagens de programação. Os autores utilizaram a ferramenta SentiStrength [Thelwall 2013] para classificar a polaridade dos comentários e analisaram a existência de relação entre as médias de polaridade observadas e a linguagem de programação usada no projeto, o dia da semana em que os *commits* foram criados, a distribuição geográfica, e a aprovação do projeto. Os resultados mostraram que projetos em Java tiveram a polaridade média levemente negativa em comparação a outras linguagens e que *commits* criados nas segundas-feiras tendem a ser mais negativos. Não foi encontrada relação entre a distribuição geográfica e a polaridade dos comentários, mas notou-se que, quanto maior é a distribuição, maior é a força da polaridade nos comentários positivos. Também não foi encontrada relação entre a aprovação do projeto e a polaridade dos comentários, mas foi observada uma correlação positiva fraca entre a média dos comentários positivos e a aprovação do projeto.

Tourani et al. [2014] analisaram a presença e a evolução de sentimentos negativos e positivos nas listas de e-mails de desenvolvedores e usuários dos dois maiores projetos da Apache: Tomcat e Ant. Um subconjunto desses e-mails foi anotado manualmente em relação à polaridade e essa anotação foi então comparada à classificação fornecida pela ferramenta SentiStrength. Os autores relatam que a ferramenta obteve baixa precisão quando comparada à anotação manual e destacam a necessidade de costumização das ferramentas de AS ao domínio de desenvolvimento de software. Os resultados da análise mostraram que a polaridade dos comentários evolui com o tempo, havendo momentos de picos positivos e negativos. Além disso, constatou-se que usuários e desenvolvedores apresentam sentimentos diferentes durante diferentes fases do projeto.

Sinha et al. [2016] analisaram *commit logs* do GitHub que foram disponibilizados como parte do MSR 2016 *challenge*. Assim como nos trabalhos anteriores, a SentiStrength foi usada para determinar os valores de polaridade. Os resultados mostraram 74,74% comentários neutros, 7,19% positivos e 18,05% negativos e que, em projetos maiores, a diferença entre a quantidade de comentários positivos e negativos é maior do que em projetos menores. Em termos das polaridades para os dias da semana, as terças-feiras tiveram comentários com polaridade negativa mais alta. Além disso, foi encontrada forte correlação entre a quantidade de arquivos alterados e o sentimento desses *commits*.

Cruz et al. [2016] utilizaram análise de sentimentos como parte de um *framework* automático para estimar a confiança entre membros de equipes DDS. A estimativa é feita por meio da extração de indícios de confiança que podem ser observados nas interações de um sistema de versionamento. Entre os indícios considerados estão: confiabilidade, tom positivo da comunicação, mímica de vocabulário, aceitação de conhecimento, colaboração e delegação. A extração dos indícios é feita a partir dos valores fornecidos pela ferramenta SentiStrength aplicada em comentários de *pull requests* nas quais os membros interagem, e de outras informações extraídas do GitHub.

Jongeling et al. [2015] avaliaram o desempenho das ferramentas SentiStrength, NLTK, Alchemy e *Stanford NLP sentiment analyser*, quando aplicadas a comentários do repositório da *Apache software foundation*. Os 392 comentários usados haviam sido anotados manualmente como parte do trabalho de Murgia et al. [2014]. As ferramentas NLTK e SentiStrength obtiveram os melhores resultados na avaliação, embora abaixo dos reportados para outros domínios. Assim como Jongeling et al. [2015], este trabalho também se propõe a avaliar ferramentas de análise de sentimentos aplicadas a comentários

no domínio de desenvolvimento de software, porém usando comentários extraídos da plataforma GitHub e ampliando a quantidade de ferramentas avaliadas para nove.

3. Avaliação das Ferramentas de Análise de Sentimentos

3.1. Dados

Uma vez que o interesse deste trabalho está na análise de sentimentos aplicada à comunicação entre membros de equipes distribuídas, optamos por utilizar como fonte de dados comentários feitos em *pull requests* de projetos hospedados na plataforma GitHub. Assim, foram extraídos automaticamente 350 comentários a partir de quatro projetos, escolhidos por apresentarem um número elevado de comentários. Cada comentário foi manualmente pré-processado para separação das sentenças, uma vez que as ferramentas avaliadas fazem classificação sentencial.

Após o pré-processamento, as 2.041 sentenças resultantes foram manualmente classificadas como positivas, negativas ou neutras, de acordo com a polaridade da emoção expressa na sentença avaliada. A classificação manual de todas as sentenças foi feita por um anotador com formação em Ciência da Computação. O número e o percentual de sentenças classificadas para cada valor de polaridade são mostrados na Tabela 3.1.

Uma característica particular de comentários como os que foram extraídos do GitHub é a presença de trechos de código-fonte, bem como observações sobre o seu funcionamento, como parte do texto do comentário. Nesses casos, o anotador procedeu a classificação da seguinte forma: sentenças que se referiam ao correto funcionamento do código foram classificadas como positivas; sentenças em que usuário reportava o mal funcionamento do código foram classificadas como negativas; e sentenças que correspondiam apenas a trechos de código-fonte foram classificadas como neutras.

Classificação	**#Sentenças**	**%Sentenças**
Positiva	260	12,7
Neutra	1657	81,2
Negativa	124	6,1
Total	**2.041**	**100**

Tabela 1. Quantidade de sentenças por polaridade

Conforme mostra a Tabela 3.1, 81% das sentenças analisadas foram classificadas como neutras, enquanto 19% foram consideradas positivas ou negativas, sendo que a proporção de sentenças positivas e negativas é de aproximadamente dois para um. Essa distribuição é similar as observadas por Jongeling et al. [2015] e Murgia et al. [2014] e se deve ao fato de muitos dos comentários descreverem aspectos técnicos do desenvolvimento e não expressarem emoção de forma perceptível ao anotador. Há também comentários em que a polaridade (positiva ou negativa) pode ser identificada com maior facilidade, devido ao uso de expressões características de uma polaridade, emoticons e formas de escrita que procuram imitar a linguagem oral, como reticências, pontos de exclamação e onomatopéias, no entanto, esse não é o caso mais frequente.

Para avaliar a reprodutibilidade da anotação manual, 215 comentários foram anotados por um segundo anotador, também com formação em Ciência da Computação. A concordância entre os dois anotadores, estimada por meio da estatística *Kappa*

[Cohen 1960], foi de $0,46$, evidenciando a subjetividade da classificação de polaridade. Uma maior concordância foi observada para as sentenças positivas ($K = 0,59$) e negativas ($K = 0,45$), o que mostra uma maior dificuldade em distinguir entre sentenças positivas/negativas e neutras do que entre sentenças positivas e negativas.

3.2. Ferramentas

As seguintes ferramentas de análise de sentimentos foram avaliadas neste estudo:

- **Análise por *emoticons*** [Park et al. 2013]: Utiliza uma tabela que relaciona os *emoticons* mais populares com a polaridade a qual são atribuídos com maior frequência. A tabela empregada foi a de Gonçalves et al. [2013] e considerou-se a polaridade da sentença como sendo a mesma do primeiro *emoticon* encontrado.
- **SentiStrength** [Thelwall 2013]: Analisa o texto de entrada e atribui a cada sentença pontuações referentes às polaridades positiva e negativa, usando um modelo baseado em aprendizado de máquina. Embora tenha sido desenvolvida para a análise de textos curtos, como os do *Tweeter*, tem sido empregada em vários trabalhos no contexto do desenvolvimento de software [Cruz et al. 2016, Jongeling et al. 2015, Guzman et al. 2014]. Neste trabalho a ferramenta foi configurada para retornar os valores positivo, negativo ou neutro a cada sentença.
- **SentiWordNet** [Baccianella et al. 2010]: É uma base de dados lexical para a mineração de opinião baseada na WordNet , na qual cada *synset* possui valores referentes à objetividade, positividade e negatividade. A base retorna valores entre 1 e -1 para os termos dependendo dos papéis que exercem na frase. Para atribuir polaridade a uma sentença, foi feita a soma dos valores médios retornados para cada termo. Caso a soma fosse igual a 0, atribuiu-se polaridade neutra à sentença; caso contrário, a polaridade foi atribuída com base no sinal do valor da soma.
- **SenticNet** [Cambria et al. 2010]: é uma base de conhecimento de senso comum que usa diferentes técnicas de inteligência artificial para inferir a polaridade de um texto a partir de conhecimento semântico. Dessa forma, a análise se dá em nível conceitual e não apenas em nível léxico e sintático. A base retorna valores de -1 a 1 referente à polaridade do termo pesquisado. A atribuição de polaridade a uma senteça foi feita por meio da média aritimética simples dos termos da frase. Caso o resultado fosse 0, atribuiu-se polaridade neutra; caso contrário, a polaridade foi atribuida levando em consideração o sinal da média obtida.
- **Happiness Index** [Dodds and Danforth 2010]: Analisa a frequência de termos pré-classificados do dicionário ANEW *(Affective Norms for English Words)*, os quais tiveram valores atribuídos por juízes humanos usando uma escala contínua de ”felicidade”. Como a ferramenta atribui valores de 0 a 9 aos termos analisados, para avaliar a polaridade de uma sentença, foi feita a média aritimética simples dos valores obtidos para cada palavra. Caso o valor resultante fosse maior ou igual a 6, atribuiu-se polaridade positiva; caso fosse menor ou igual a 4, atribuiu-se polaridade negativa; no restante dos casos, a polaridade foi considerada neutra.
- **PANAS-t** [Goncalves et al. 2012]: Escala psicométrica para a detecção de humor na plataforma Twitter que aplica uma versão adaptada do *Positive Affect Negative Affect Scale* (PANAS) [Watson et al. 1988] utilizada na psicologia. Para atribuir polaridade a uma sentença, busca-se identifcar se a sentença fala sobre o estado emocional de seu autor; caso contrário, a sentença é considerada neutra. Nos casos

em que a sentença fala sobre um estado emocional, atribui-se a polaridade positiva ou negativa com base no estado emocional identificado.

- **AFINN** [Nielsen 2011] e **Sentiment140 Lexicon** [Mohammad et al. 2013]: Ambas são bases léxicas que possuem valores de polaridade pré-determinados atribuídos a termos da língua inglesa. Para avaliar a polaridade de uma sentença, neste trabalho, foi feita a soma dos valores de polaridade das palavras da sentença; caso o resultado fosse 0, atribuiu-se polaridade neutra; caso contrário, a polaridade foi atribuida levando em consideração o sinal do valor obtido.

Além das ferramentas citadas, também foi avaliado um método combinado, que estima a polaridade da sentença por meio da combinação linear das saídas das oito ferramentas analisadas. Esse método é similar ao proposto por Araujo et al. [2014] para a classificação de *tweets*, mas se diferencia na forma do sistema de pesos e na busca pelos pesos mais apropriados. No método combinado usado neste trabalho, cada ferramenta recebeu três pesos, um para cada polaridade. Dessa forma, dependendo da polaridade aferida pela ferramenta, o peso correspondente foi utilizado. Esses pesos foram estimados por um algoritmo genético, que buscou maximizar o desempenho da classificação em termos da medida-F obtida para cada polaridade.

3.3. Medidas

As feramentas foram avaliadas usando as sentenças resultantes do pré-processamento, conforme descrito na Subseção 3.1, e tiveram o seu desempenho registrado em termos das seguintes medidas, estimadas para cada valor de polaridade p:

- Precisão: total de setenças corretamente classificadas como p sobre o total de sentenças classificadas como p;
- Cobertura: total de setenças corretamente classificadas como p sobre o total de sentenças com polaridade p no conjunto;
- Medida-F: média harmônica dos valores de precisão e cobertura obtidos para p.

Além das medidas por polaridade, também foi calculada a macro-F para cada ferramenta, que corresponde à média aritmética dos respectivos valores de medida-F.

4. Resultados Obtidos

Os resultados obtidos pelas ferramentas avaliadas em termos da macro-F e da precisão, cobertura e medida-F para os três valores de polaridade – positiva, negativa e neutra –, são apresentados na Tabela 2. Como pode ser observado, a ferramenta que obteve o melhor resultado em termos de macro-F foi a SentiStrength (47,8%), seguida pelo método combinado (47%). A ferramenta SenticNet obteve a menor macro-F registrada nas avaliações (18,5%), ficando abaixo de métodos mais simples, como a análise por *emoticons* e *hapiness index*. A baixa macro-F obtida pela SenticNet pode ser explicada analisando-se as distribuição das classificações nas polaridades positiva, negativa e neutra geradas pela ferramenta, que foi de 62,3%, 21,9% e 15,9%, respectivamente, enquanto a distribuição observada na anotação manual foi de 12,7%, 6,1% e 81,2%. Com exceção da SenticNet, as classificações de todas as ferramentas apresentaram distribuição similar a da anotação manual, sendo as sentenças neutras majoritárias e as sentenças negativas minoritárias.

Conforme mostra a Tabela 2, os resultados obtidos com o método combinado ficaram consistentemente entre os melhores quando considera-se cada ferramenta individualmente, porém nunca estão em primeiro lugar em cada ponto individual. Assim, é

possível inferir que o método combinado incorpora os pontos fortes das ferramentas que o compõe, mas também os erros de classificação que essas ferramentas apresentam.

	Precisão			Cobertura			F1			
	Positivo	**Neutro**	**Negativo**	**Positivo**	**Neutro**	**Negativo**	**Positivo**	**Neutro**	**Negativo**	**Macro-F**
SentiWordNet	13,68%	77,45%	6,86%	35,77%	37,72%	30,65%	19,79%	50,73%	11,21%	27,24%
Análise de Emoticons	94,59%	82,76%	100,00%	13,46%	99,94%	2,42%	23,57%	90,54%	4,72%	39,61%
SentiStrength	36,29%	90,39%	15,12%	67,69%	60,71%	54,03%	47,25%	72,64%	23,63%	**47,84%**
SenticNet	12,04%	67,59%	8,52%	58,85%	13,22%	30,65%	19,99%	22,11%	13,33%	18,48%
Happiness Index	15,08%	81,44%	4,65%	23,08%	76,52%	3,23%	18,24%	78,90%	3,81%	33,65%
PANAS-t	25,00%	81,21%	14,29%	1,15%	99,09%	0,81%	2,21%	89,26%	1,53%	31,00%
AFINN	21,65%	84,64%	16,39%	50,38%	61,19%	31,45%	30,29%	71,03%	21,55%	40,96%
Sentiment140	41,53%	83,15%	15,38%	18,85%	93,24%	8,06%	25,93%	87,91%	10,58%	41,47%
Genético	50,00%	84,06%	24,64%	26,92%	92,94%	13,71%	35,00%	88,28%	17,62%	46,96%

Tabela 2. Resumo dos resultados obtidos pelas ferramentas avaliadas

Os gráficos das figuras 1, 2 e 3 mostram a comparação dos valores de precisão, cobertura e medida-F, respectivamente, obtidos pelas ferramentas avaliadas. No gráfico da Figura 1 percebe-se que a análise por *emoticons* teve precisão superior, especialmente para as polaridades positiva e negativa, para as quais a precisão das outras ferramentas é relativamente baixa. No entanto, a cobertura desse método para essas polaridades está entre as mais baixas observadas, conforme mostra a Figura 2. O método combinado teve uma precisão boa comparada às outras ferramentas, mas manteve a mesma tendência de baixa precisão para as polaridades negativa e positiva.

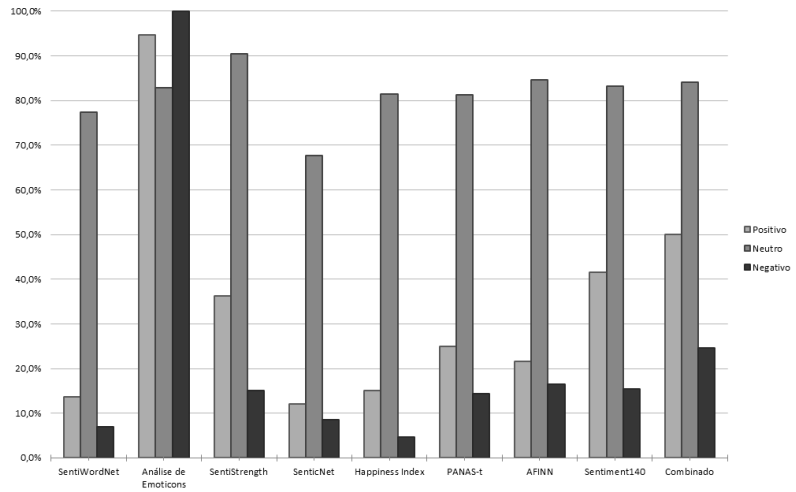

Figura 1. Comparação das ferramentas em termos da precisão

Com relação à cobertura, houve uma discrepância maior no comportamento das ferramentas. Conforme mostra o gráfico da Figura 2, ferramentas com alta cobertura para sentenças neutras tendem a possuir baixa cobertura para as outras duas polaridades (p.e., análise por *emoticons* e PANAS-t), enquanto outras ferramentas apresentaram cobertura mais baixa para sentenças neutras do que para sentenças positivas/negativas (p.e., SentiStrength e SenticNet). Analisando-se a média dos valores de cobertura para as três polaridades, o melhor resultado foi obtido pela SentiStrength ($60,8\% \pm 6,8\%$), seguida pela ferramenta AFINN ($47,7\% \pm 15,1\%$) e pelo método combinado ($44,5\% \pm 42,4\%$). O pior resultado foi obtido pela PANAS-t ($33,7\% \pm 56,6\%$).

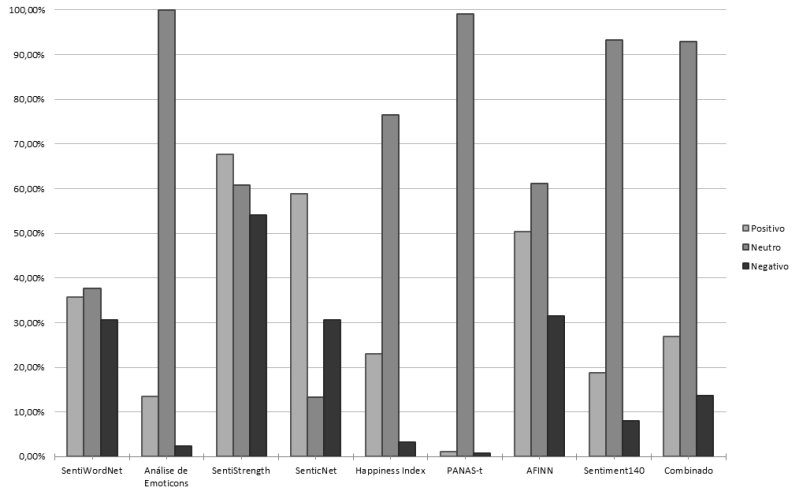

Figura 2. Comparação das ferramentas em termos da cobertura

O gráfico da Figura 3 mostra um comparativo das ferramentas em termos da medida-F. Como pode ser observado, a ferramenta SentiStrength obteve a melhor medida-F para as polaridades positiva e negativa, enquanto a PANAS-t obteve a pior. O método combinado, por sua vez, superou a SentiStrength para sentenças neutras, ficou em segundo lugar para sentenças positivas e em terceiro lugar para sentenças negativas, sendo superado, nesse caso, pela ferramenta AFINN.

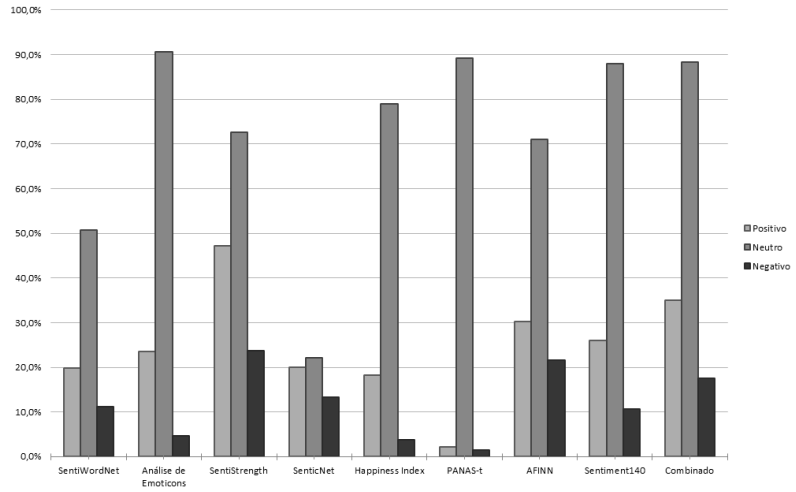

Figura 3. Comparação das ferramentas em termos da medida-F

5. Conclusões e Trabalhos Futuros

O desenvolvimento de software é caracterizado pela forte interação que se estabelece entre as pessoas envolvidas. Essas pessoas podem carregar características oriundas de uma herança socio-cultural, que nem sempre são expressas verbalmente, mas que podem aparecer embutidas na comunicação escrita. Nesse sentido, a análise de sentimentos se mostra como uma ferramenta importante para a captura automática de aspectos subjetivos relacionados à interação e ao relacionamento dos membros das equipes de desenvolvimento. Para isso, é necessário que se conheça o desempenho das ferramentas de análise

de sentimentos disponíveis quando aplicadas a esse domínio, uma vez que a classificação produzida influencia diretamente nas conclusões dos estudos conduzidos com base nessas informações. Cabe ressaltar que a correta identificação e uso de informações acerca da interação e relacionamento entre membros da equipe pode impactar as atividades de um processo de desenvolvimento e, consequentemente, na qualidade do produto final.

Neste trabalho foram avaliadas nove ferramentas de análise de sentimentos aplicadas a comentários da plataforma GitHub, incluíndo ferramentas populares, como a SentiStrength, e um método combinado similar ao proposto por Gonçalves et al. [2014]. A análise dos resultados mostrou que a SentiStrength saiu-se melhor, seguida pelo método combinado, quando se considerou o desempenho médio para os três valores de polaridade considerados. Também foi possível observar que métodos simples como a análise por *emoticons* podem obter valores altos de precisão, embora tendam a ter baixa cobertura. Os resultados desse estudo mostraram uma variação considerável de desempenho entre as ferramentas e que, mesmo para a ferramenta melhor avaliada, o desempenho médio ficou abaixo de 50%. Isso mostra o quanto a escolha da ferramenta de análise de sentimentos pode influenciar os estudos que as utilizam. Além disso, evidencia a necessidade de investigação na área e do desenvolvimento de ferramentas que sejam capazes de capturar as especificidades de textos como os produzidos por equipes de DDS.

Como trabalhos futuros pretende-se melhorar a base de comentários anotados, aumentando o volume de sentenças anotadas, refinando os critérios adotados na anotação das polaridades e realizando experimentos de anotação com mais anotadores. Também pretende-se criar outras versões do método combinado por meio do uso de outros algoritmos para a estimativa de pesos e outras formas de combinar as ferramentas.

Referências

Araújo, M., Gonçalves, P., Cha, M., and Benevenuto, F. (2014). ifeel: A system that compares and combines sentiment analysis methods. In *Proc. of the 23rd Int. Conf. on World wide web Companion*, pages 75–78.

Baccianella, S., Esuli, A., and Sebastiani, F. (2010). Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *Proc. of the 7th Int. Conf. on Language Resources and Evaluation*, pages 2200–2204.

Cambria, E., Speer, R., Havasi, C., and Hussain, A. (2010). Senticnet: A publicly available semantic resource for opinion mining. In *AAAI fall symposium: commonsense knowledge*.

Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, (20):213–220.

Cruz, G., Huzita, E., and Feltrim, V. (2016). Estimating trust in virtual teams - a framework based on sentiment analysis. In *Proc. of the 18th Int. Conf. on Enterprise Information Systems (ICEIS 2016)*, pages 464–471.

Dodds, P. S. and Danforth, C. M. (2010). Measuring the happiness of large-scale written expression: Songs, blogs, and presidents. *Journal of Happiness Studies*, 11(4):441–456.

Goncalves, P., Benevenuto, F., and Almeida, V. (2013). O que tweets contendo emoticons podem revelar sobre sentimentos coletivos. In *Proc. of the II Brazilian Workshop on Social Network Analysis and Mining (BraSNAM)*, pages 1–12.

Goncalves, P., Dores, W., and Benevenuto, F. (2012). Panas-t: Uma escala psicometrica para analise de sentimentos no twitter. In *Proc. of the I Brazilian Workshop on Social Network Analysis and Mining (BraSNAM)*.

Guzman, E., Azócar, D., and Li, Y. (2014). Sentiment analysis of commit comments in github: An empirical study. In *Proc. of the 11th Working Conf. on Mining Software Repositories*, pages 352–355.

Herbsleb, J. D. and Moitra, D. (2001). Global software development. *IEEE Software*, 18(2):16–20.

Jongeling, R., Datta, S., and Serebrenik, A. (2015). Choosing your weapons: On sentiment analysis tools for software engineering research. In *IEEE Int. Conf. on Software Maintenance and Evolution*, pages 531–535. IEEE.

Liu, B. (2012). *Sentiment Analysis and Opinion Mining*. Morgan & Claypool Publishers.

Mohammad, S. M., Kiritchenko, S., and Zhu, X. (2013). Nrc-canada: Building the state-of-the-art in sentiment analysis of tweets. *Computing Research Repository (CoRR)*, abs/1308.6242.

Murgia, A., Tourani, P., Adams, B., and Ortu, M. (2014). Do developers feel emotions? an exploratory analysis of emotions in software artifacts. In *Proc. of the 11th Working Conf. on Mining Software Repositories*, pages 262–271.

Nielsen, F. Å. (2011). A new anew: Evaluation of a word list for sentiment analysis in microblogs. *Computing Research Repository (CoRR)*, abs/1103.2903.

O'Conchuir, E., Holmstrom, H., Agerfalk, P., and Fitzgerald, B. (2006). Exploring the assumed benefits of global software development. In *Int. Conf. on Global Software Engineering*, pages 159–168.

Park, J., Barash, V., Fink, C., and Cha, M. (2013). Emoticon style: Interpreting differences in emoticons across cultures. In *Proc. of the 7th Int. AAAI Conf. on Weblogs and Social Media*.

Sengupta, B., Chandra, S., and Sinha, V. (2006). A research agenda for distributed software development. In *Int. Conf. on Software Engineering*. ACM.

Sinha, V., Lazar, A., and Sharif, B. (2016). Analyzing developer sentiment in commit logs. In *Proc. of the 13th Int. Conf. on Mining Software Repositories*, pages 520–523.

Thelwall, M. (2013). Heart and soul: Sentiment strength detection in the social web with sentistrength. *Proceedings of the CyberEmotions*, pages 1–14.

Tourani, P., Jiang, Y., and Adams, B. (2014). Monitoring sentiment in open source mailing lists-exploratory study on the apache ecosystem. In *Proc. of the 2014 Conf. of the Center for Advanced Studies on Collaborative Research*, pages 74–95.

Watson, D., Clark, L. A., and Tellegen, A. (1988). Development and validation of brief measures of positive and negative affect: the panas scales. *Journal of personality and social psychology*, 54(6):1063.

Padrões de Projeto em Java: Um Estudo Prático sobre a Utilização e Benefícios

Marina Gabriela do Amaral Santos, Maurício R. de A. Souza, Eduardo Figueiredo

Laboratório de Engenharia de Software (LabSoft) – Universidade Federal de Minas Gerais (UFMG) – Belo Horizonte – Brasil

{marina.santos, mrasouza, figueiredo}@dcc.ufmg.br

Abstract. *Design patterns are defined as reusable solutions to recurring problems. These solutions have many expected benefits for the project, such as ease of communication, maintainability, and organization. Design patterns also bring a common vocabulary to the development team. However, it is necessary for the project to be designed for the correct application of design patterns and organizations do not always provide time for such. This paper presents the results of a survey conducted with Java software developers in order to investigate their knowledge, incentives, and difficulties in the adoption of design patterns. As a result, we observed great influence of the company's culture and adopted process practices.*

Resumo. *Os padrões de projeto são definidos como soluções reusáveis para problemas recorrentes. Essas soluções visam diversos benefícios para o projeto, tais como: facilidade de comunicação, uma vez que os padrões de projeto trazem um vocabulário comum; facilidade de manutenção e organização do projeto. No entanto, é necessário que o projeto seja modelado para a aplicação correta dos padrões de projeto e nem sempre as organizações disponibilizam tempo para tal modelagem. Este artigo apresenta os resultados de um questionário aplicado à desenvolvedores de software Java com o objetivo de investigar seus conhecimentos, incentivos e dificuldades no uso de padrões de projeto. Como resultado, observou-se grande influência da cultura da empresa e seus processos adotados.*

1. Introdução

De acordo com Gamma e colegas (1995), os padrões de projeto são descrições de objetos e classes comunicantes que são customizadas para resolver um problema geral de projeto num contexto particular. Um padrão possui quatro elementos essenciais (Gamma et al., 1995): nome do padrão, problema, solução e conseqüências. O nome do padrão é um identificador que podemos usar para descrever o problema do projeto, suas soluções e as consequências em uma ou duas palavras. Isso torna mais fácil para pensar em projetos, comunicá-los e as suas vantagens e desvantagens para outros. O problema, por outro lado, descreve quando aplicar o padrão. Ou seja, ele explica o problema e seu contexto. O problema pode descrever os problemas de design específicos, tais como a forma de representar algoritmos como objetos. Algumas vezes o problema inclui uma lista de condições que precisam ser atendidas antes para que se faça sentido aplicar o padrão.

Além do nome e problema, outros dois elementos essenciais são a solução e as consequências. A solução descreve os elementos que fazem parte do desenho, seus relacionamentos, responsabilidades e colaborações. A solução não descreve um projeto concreto particular ou implementação. Isso porque o padrão é como um esboço que pode ser aplicado em diferentes situações. Finalmente, as consequências são os compromissos e resultados de aplicar o padrão. Embora as consequências sejam muitas vezes mudas quando descrevemos decisões de projeto, elas são críticas para avaliar alternativas de projeto e para a compreensão dos custos e benefícios da aplicação do padrão (Cardoso e Figueiredo, 2015) (Garcia et al., 2005).

Este artigo apresenta os resultados de um questionário aplicado a 34 profissionais de desenvolvimento de software atuantes na indústria, visando investigar o nível de conhecimento destes sobre padrões de projeto e sua utilização no desenvolvimento de sistemas. A ferramenta para coleta de informações utilizada neste estudo foi o questionário [Wohlin et al., 2000]. Os questionários se destacam pela popularidade e abrangência de utilização, sendo eles usados desde votações a definição de requisitos em software. O questionário é um método para coletar informação de pessoas acerca de suas idéias, sentimentos, planos, bem como origem social, educacional e outros fatores humanos. Além disso, uma pesquisa por meio de questionários é utilizada para identificar características de uma ampla população de indivíduos, comumente associada ao uso de questionários para a coleta dos dados (Easterbrook et al., 2008).

A partir da aplicação do questionário, procurou-se entender de desenvolvedores de software atuantes na indústria sobre os principais fatores que incentivam e que dificultam o uso de padrões de projeto, e a influência da colaboração da equipe, da alta gestão e de processos na utilização dos padrões. Espera-se o entendimento de como estes aspectos humanos e organizacionais podem influenciar em práticas e decisões técnicas durante o desenvolvimento de software, enquanto atividade colaborativa. Os resultados deste estudo mostram que aproximadamente 65% dos desenvolvedores concordam que lideranças da empresa, tais como gerentes e coordenadores de equipe, incentivam a utilização de padrões de projeto. Por outro lado, os principais fatores apontados que dificultam a adoção de padrões de projeto são prazos curtos dos projetos e a falta de conhecimento a respeito dos padrões.

Além desta seção introdutória, este artigo está organizado da seguinte forma. Na seção 2 são analisados os benefícios obtidos com o uso de padrões de projeto. A seção 3 descreve a metodologia de execução do questionário. Na seção 4, são apresentados e discutidos os resultados obtidos neste estudo. A seção 5 discute as ameaças à validade do estudo. Por fim, na seção 6 são realizadas as considerações finais e apontadas direções para trabalhos futuros.

2. Padrões de projeto: benefícios esperados

O objetivo principal dos padrões de projeto é propor soluções reusáveis para problemas recorrentes. Assim, a adoção de padrões de projeto pode trazer diversos benefícios para o projeto ou organização (Cacho et al., 2014) (Cardoso e Figueiredo, 2015) (Gamma et al., 1995). Esta seção lista e descreve alguns dos principais benefícios esperados pela adoção de padrões de projeto, tais como aqueles associados à facilidade de manutenção e organização do projeto. Por exemplo, em relação a facilidade de manutenção, os

padrões de projeto prezam pelo baixo acoplamento entre as classes e a alta coesão. Com isso, extensões e alterações possam ser feitas sem grandes traumas. Além deste benefício esperado, padrões de projeto defendem a padronização da estrutura das classes. Desta forma, eles podem contribuir também para a organização da estrutura geral do sistema.

Gamma (1995) enumera alguns pontos sobre o que se esperar de um padrão de projeto: (i) fornecer um vocabulário comum do projeto, (ii) um auxílio para a documentação e o aprendizado, (iii) um acréscimo aos métodos existentes e (iv) um alvo para a refatoração. Em relação ao primeiro ponto, padrões de projeto fornecem um vocabulário comum, permitindo que os designers o usem para se comunicar, documentar e explorar alternativas. Os padrões de projeto fazem um sistema parecer menos complexo, permitindo que você fale sobre ele em um nível mais alto de abstração do que em uma notação de design ou linguagem de programação.

Conhecer os padrões de projeto também torna mais fácil o entendimento de sistemas existentes (Cacho et al., 2006) (Garcia et al., 2005) (Gamma et al., 1995). A maioria dos grandes sistemas orientados a objetos utiliza padrões de projeto. Pessoas aprendendo programação orientada a objetos muitas vezes se queixam de que os sistemas que estão trabalhando usam a herança de forma complicada e que é difícil seguir o fluxo de controle. Em grande parte, isso é porque eles não entendem os padrões de projeto no sistema. Aprender esses padrões de design vai ajudar a entender sistemas orientados a objetos existentes.

Os padrões de projeto mostram como usar técnicas essenciais, tais como objetos, herança e polimorfismo. Padrões de projeto fornecem uma maneira de descrever mais do "porquê" de um projeto e não apenas registrar os resultados de suas decisões. A aplicabilidade, consequências e implementação dos padrões de projeto ajudam a guiar nas decisões do que é necessário fazer. Finalmente, de acordo com Opdyke e Johnson (1990), um dos problemas no desenvolvimento de softwares reutilizáveis é que muitas vezes ele tem de ser reorganizado ou refatorado. Padrões de projeto ajudam a determinar como reorganizar um projeto e eles podem reduzir a quantidade de refatoração que é preciso fazer mais tarde.

Figura 1 - Ciclo de vida de software orientado a objetos

O ciclo de vida do software orientado a objetos tem várias fases. Algumas destas fases são: prototipação, expansão e consolidação; conforme Figura 1 adaptada de Foote (1992). A necessidade contínua de satisfazer mais requisitos, juntamente com a necessidade de maior reutilização, faz o software orientado a objetos passar por repetidas fases de expansão e consolidação – de expansão à medida que novos

requisitos são atendidos, e de consolidação à medida que o software se torna mais genérico. Um ponto importante a se observar no uso de padrões de projeto é que o tempo gasto na identificação e estruturação dos padrões dentro do projeto é inicialmente maior que o desenvolvimento da solução diretamente sem uso destes padrões. No entanto, o ganho com a manutenção do código, um de seus principais benefícios, assim como com a clareza de entendimento para os próximos profissionais a manter ou evoluir o sistema é grande.

3. Metodologia

O método de pesquisa questionário é quantitativo, a pesquisa deve ser planejada pelo pesquisador e a aplicação deve estar ligada aos objetivos da pesquisa (Wohlin et al., 2000). A aplicação é desejada quando o pesquisador pretende investigar o que, porque, como ou quanto se dá determinada situação. Por outro lado, através de questionários não é possível determinar variáveis dependentes e independentes. A pesquisa por meio de questionários dá-se no momento presente ou recente e trata situações reais do ambiente. De maneira complementar, um estudo experimental completo poderia, por exemplo, adquirir a informação inicial utilizando um questionário, elaborar uma teoria através de um experimento controlado e verificar a teoria proposta em condições reais por meio de um estudo de caso (Wohlin et al., 2000).

Para este estudo, o questionário atua como base para levantar as informações necessárias que permitam analisar o conhecimento de desenvolvedores de software Java acerca de padrões de projeto e seu efetivo uso. Este estudo visa identificar quais fatores encorajam ou desencorajam o uso destes padrões no desenvolvimento diário de aplicações em organizações. Sendo assim, as questões de pesquisa que guiam este estudo podem ser enumeradas como a seguir.

- ***QP1:*** *Qual o grau de utilização de padrões de projetos, segundo a visão de profissionais em empresas de software?*

  O objetivo desta questão é identificar a utilização dos padrões de projeto pelo desenvolvedor na organização.

- ***QP2:*** *A adoção de padrões de projeto é influenciada pelo processo de desenvolvimento de software?*

  Partindo do princípio que o processo de desenvolvimento de uma empresa define as etapas que serão seguidas para execução de um projeto de software, visa-se identificar o quanto o processo de desenvolvimento influencia na utilização de padrões de projeto.

- ***QP3:*** *Quais os principais fatores que dificultam a adoção de padrões de projetos em organizações desenvolvedoras de software?*

  Dados os principais fatores que usualmente dificultam o desenvolvimento de software, o objetivo é avaliar as dificuldades encontradas pelos desenvolvedores que os desencoraja a utilizar padrões de projeto.

- ***QP4:*** *Quais os principais benefícios esperados pela adoção de padrões de projeto, na perspectiva de profissionais de organizações desenvolvedoras de software?*

Identificados os fatores que influenciam a utilização ou não de padrões de projeto, visa-se esclarecer o entendimento dos desenvolvedores quanto aos benefícios que seriam alcançados com a utilização.

A partir destas questões de pesquisa, a elaboração do questionário foi realizada em seis etapas, ilustradas Figura 2. Em cada uma delas, o questionário foi alvo de sucessivas revisões e aplicações, garantindo que suas perguntas adequadas o atendimento dos objetivos da pesquisa e para o entendimento da população alvo de participantes. A Figura 2 mostra que, inicialmente, foi elaborado o questionário piloto após revisão com pesquisadores e profissionais. Este questionário foi revisado posteriormente para avaliar a coerência das perguntas com o estudo a ser realizado e uma versão final do questionário foi definida. O questionário final foi confeccionado e disponibilizado na plataforma *Google Form*[1], e dispõe de um total de treze questões, sendo sete questões para de caracterização do *background* dos participantes, e seis questões relacionadas às questões de pesquisa definidas.

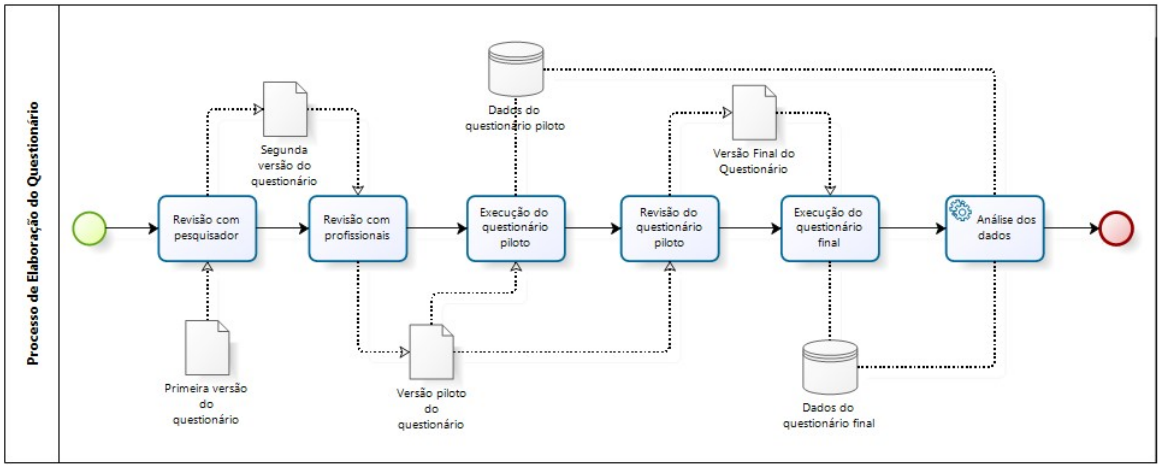

Figura 2 - Processo de Elaboração do Questionário

Uma característica fundamental do questionário é a seleção de uma amostra representativa de uma população bem definida e as técnicas de análise de dados usadas para generalizar a partir dessa amostra para a população (Easterbrook et al, 2008). Para este estudo, foi utilizada uma pequena amostra, visando não à generalização, mas a caracterização de dificuldades encontradas por um grupo de profissionais. No entanto, esta pesquisa pode ser estendida a um grupo maior de profissionais. Assim, poderá ser inferida a generalização dos resultados. Desta forma, o questionário contou com a participação de 34 profissionais de organizações desenvolvedoras de software.

4. Análise dos dados

Os dados obtidos a partir dos itens do questionário são analisados nesta seção. As ferramentas utilizadas para a análise dos dados foram *Google Form*[1], que provê suporte para análise e quantificação dos dados obtidos. Uma planilha eletrônica foi utilizada para sumarização dos dados e criação de gráficos.

[1] http://www.google.com/Forms

4.1. Caracterização dos participantes

Em relação aos participantes deste estudo, foram selecionados apenas desenvolvedores atuantes em organizações. Ou seja, o público alvo deste estudo é desenvolvedor de software em atividade. O tamanho da organização dos participantes variou bastante, como pode ser observado conforme Figura 3. Entretanto, a Figura 3 mostra que quase 60% dos participantes pertencem a organizações de grande porte, com mais de 100 funcionários. Quanto a sua formação acadêmica, 45% dos participantes são graduados e 55% são pós-graduados.

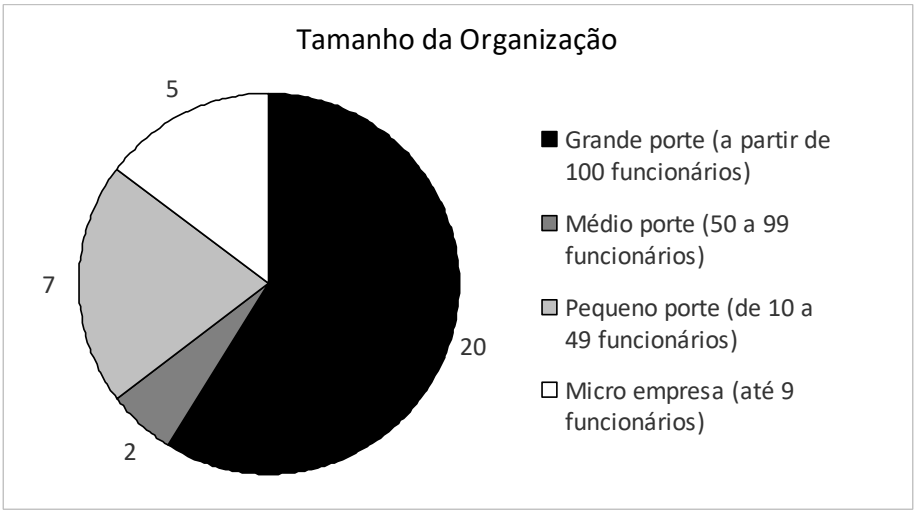

Figura 3 - Tamanho da organização dos participantes

Foi avaliado também o conhecimento dos participantes quanto a quatro aspectos relevantes ao estudo: programação orientada a objetos, modelagem de software (UML), reutilização de software e padrões de projeto. A distribuição dos conhecimentos pode ser observada no gráfico de barras da Figura 4. Este gráfico mostra que, na grande maioria das áreas de conhecimento questionadas, os participantes se classificaram como experientes ou com conhecimento moderado. Nenhum participante afirmou desconhecer qualquer dos quatro aspectos questionados.

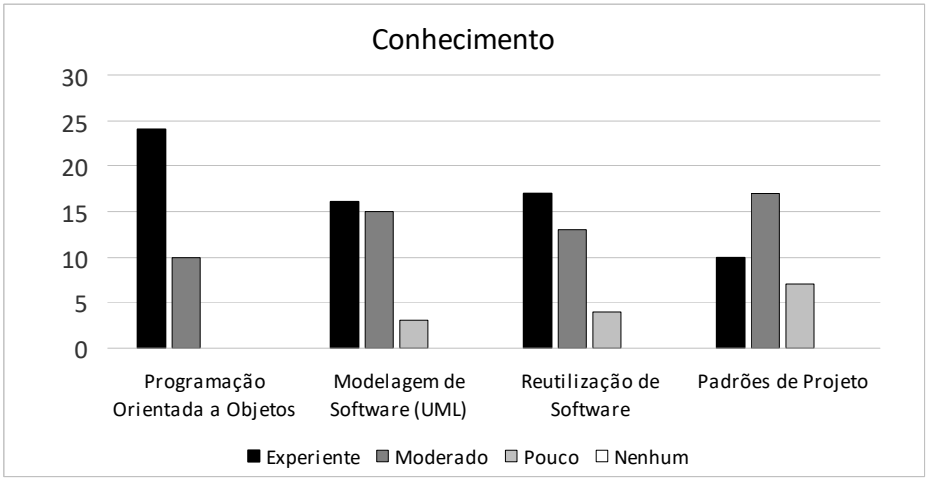

Figura 4 - Nível de conhecimento dos participantes

4.2. Influências na utilização de padrões de projeto

Para que fosse possível avaliar a utilização ou não dos padrões de projeto, foi solicitado aos participantes que avaliassem a seguinte afirmação: "*Com que frequência você utiliza padrões de projeto no desenvolvimento de software na sua empresa?"* Nessa questão, treze participantes afirmaram que utilizam frequentemente, dezesseis participantes afirmaram que utilizam às vezes, três afirmaram que raramente são utilizados e dois participantes afirmaram que nunca utilizam.

Em seguida, os participantes avaliaram a afirmação: "*A liderança (alta gestão, gerente de projetos, líder técnico, etc.) incentiva a utilização de padrões de projeto*." O resultado é apresentado na Figura 5. Como podemos observar por esta figura, vinte e dois participantes concordaram totalmente (7) ou parcialmente (15) com a afirmação. Em contrapartida, doze discordaram totalmente (6) ou parcialmente (6).

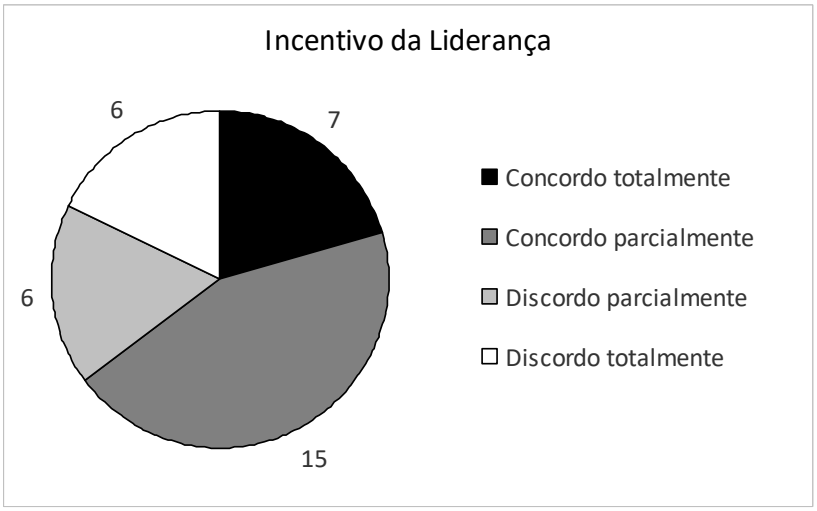

Figura 5 - Opinião dos participantes sobre o incentivo da liderança no uso de padrões de projeto

A partir da resposta da questão anterior, podemos observar a grande influência da liderança na utilização de padrões de projeto. Assim, foi solicitado aos participantes que enumerassem dentre algumas opções pré-definidas, quais os principais fatores que dificultam a utilização de padrões de projeto nos projetos em que eles trabalham. O resultado pode ser observado na Figura 6.

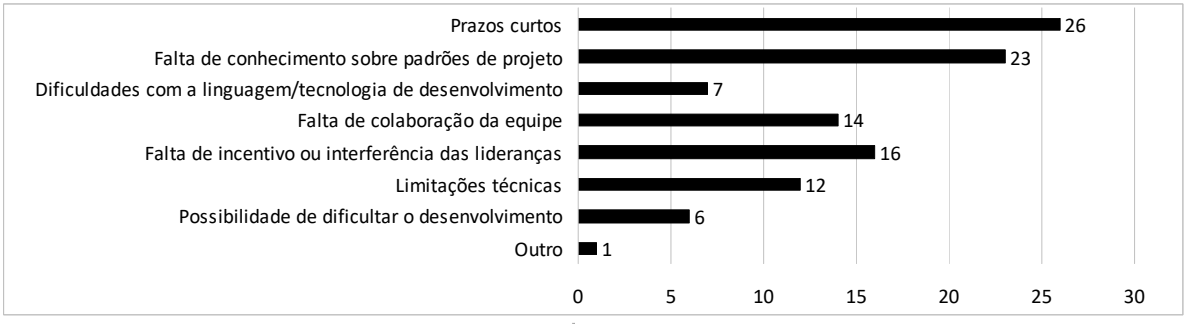

Figura 6 - Principais fatores que dificultam a utilização de padrões de projeto

Como mostra a Figura 6, na opinião dos participantes, os prazos curtos e a falta de conhecimento sobre padrões de projeto são os principais fatores que dificultam a utilização de padrões de projeto. A cultura da empresa conforme discutido anteriormente engloba os processos de software adotados, a dinâmica de desenvolvimento de software e solução de problemas. Se o foco da empresa é entregar um software com o menor tempo de desenvolvimento sem considerar manutenções e evoluções futuras, muito provavelmente os padrões de projeto serão descartados pelo líder técnico do projeto. Como foi mencionado anteriormente, o prazo inicial de desenvolvimento de software utilizando padrões de projeto é maior que o prazo onde a solução é desenvolvida diretamente. Com isso, os desenvolvedores destacaram que o curto prazo para desenvolvimento também é um dos principais fatores que dificultam a utilização dos padrões. Além destes dois fatores principais, foi citada também a falta de incentivo ou interferência das lideranças da empresa. Novamente, este resultado reforça a importância da liderança na adoção de padrões de projeto.

Após a avaliação dos principais fatores que dificultam a utilização, foi questionado aos participantes sobre os benefícios que seriam alcançados com a utilização de padrões de projeto. Os benefícios mencionados pelos participantes são apresentados na Figura 7. De acordo com a maioria dos participantes, os três maiores benefícios obtidos a partir da utilização de padrões de projeto são: (i) facilitar na manutenção e evolução do código, (ii) dar apoio na solução de problemas e (iii) favorecer a reutilização de soluções. Em seguida, é destacada a facilitação no entendimento do código e diminuir o tempo de desenvolvimento.

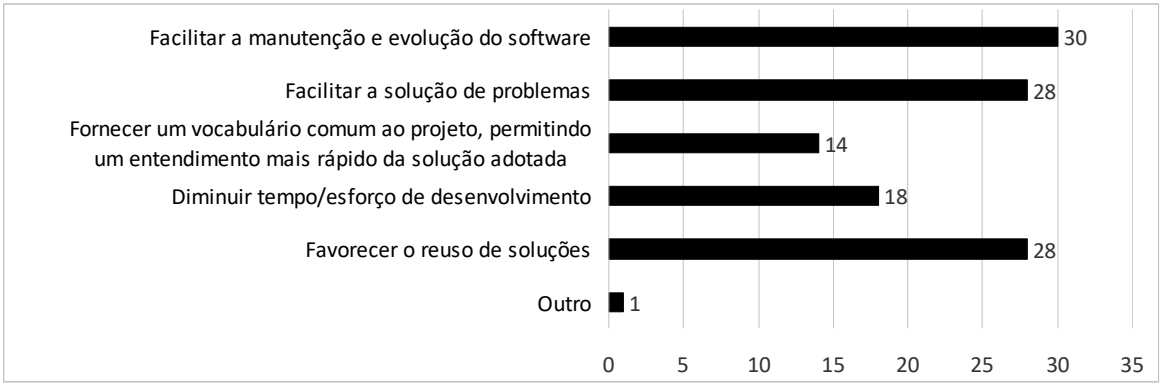

Figura 7 - Percepção dos participantes acerca dos benefícios obtidos com a utilização de padrões de projeto

5. Ameaças à Validade

Conforme apresentado na Seção 3 deste artigo, foi analisada uma pequena amostra de 34 desenvolvedores de software. Esta amostra não pode ser considerada para a generalização da situação em que os padrões de projeto não são aplicados, em geral, devido à cultura e dos processos de desenvolvimento de software das organizações. No entanto, a aplicação deste questionário a um grupo maior de participantes, com organizações diversas, bem como seus processos de desenvolvimento, permitirá com que seja inferida uma generalização acerca do uso de padrões de projeto na prática das organizações.

6. Considerações Finais

Neste artigo foram apresentados os resultados de um questionário realizado com 34 profissionais com experiência em desenvolvimento de software. O questionário teve o propósito de identificar, segundo a percepção dos participantes, quais os principais fatores que influenciam na utilização de padrões de projeto nas organizações e seus benefícios.

Os resultados do questionário fornecem indícios de que a cultura das organizações e seus processos de desenvolvimento são os grandes influenciadores na utilização de padrões de projeto. Os benefícios que seriam obtidos são bem claros para os participantes e os mais destacados são associados à manutenção e evolução do código, solução de problemas e a reutilização de soluções. Apesar de ter sido observado o entendimento dos profissionais sobre os benefícios que seriam obtidos pela adoção de padrões de projeto, os resultado indicam os padrões podem não estar sendo de fato aplicados devido ao curto prazo e a falta de incentivo das organizações.

Em questão a falta de incentivo das organizações, por exemplo, acredita-se que melhores resultados no projeto seriam obtidos se houvesse antes do início do desenvolvimento, uma fase de especificação técnica. Esta fase, que aconteceria logo após a finalização da especificação de casos de uso, avaliaria tecnicamente a estrutura do projeto e quais os padrões mais adequados a serem utilizados. A especificação técnica seria realizada por um arquiteto de software que produziria para os desenvolvedores, a estrutura-modelo a ser seguida no desenvolvimento com o esboço dos padrões a serem seguidos. Com isso, no início do desenvolvimento, o profissional teria de forma clara, a estrutura a ser seguida para a solução do problema. Assim, o desenvolvedor se preocupa com a implementação das regras de negócio do sistema e não em estruturar as classes da melhor forma para resolver o problema.

Como trabalhos futuros, pretende-se investigar mais a fundo como as organizações incentivam a adoção dos padrões de projeto e como os times são afetados por este fator. Assim, planejamos a replicação desta pesquisa em outros contextos e com desenvolvedores de variados perfis.

Agradecimentos

Este trabalho recebeu apoio financeiro das agências de fomento CAPES, CNPq e FAPEMIG.

Referências

Fatimah Mohammed Alghamdi and M. Rizwan Jameel Qureshi (2014). "Impact of Design Patterns on Software Maintainability." *International Journal of Intelligent Systems and Applications,* 6.10: 41.

Brian Foote (1992). "*A Fractal Model of the Lifecycles of Reusable Objects*". *OOPSLA Workshop on Reuse.* Vancouver, Canada.

David Budgen (2013). "Design Patterns: Magic or Myth?" *IEEE Software*, vol.30, no. 2, pp. 87-90.

Nelio Cacho, Claudio Sant'Anna, Eduardo Figueiredo, Francisco Dantas, Alessandro Garcia, and Thais Batista (2014). "Blending design patterns with aspects: A quantitative study". *Journal of Systems and Software (JSS)*, vol. 98, pp. 117-139.

Nelio Cacho, Claudio Sant'Anna, Eduardo Figueiredo, Alessandro Garcia, Thais Batista, and Carlos Lucena (2006). "*Composing Design Patterns: A Scalability Study of Aspect-Oriented Programming*". In proceedings of the 5th International Conference on Aspect Oriented Software Development (AOSD), pp. 109-121. Bonn, Germany.

Bruno Cardoso and Eduardo Figueiredo (2015). "Co-Occurrence of Design Patterns and Bad Smells in Software Systems: An Exploratory Study". In proceedings of the Brazilian Symposium on Information Systems (SBSI). Goiania, GO.

Bruno Cardoso and Eduardo Figueiredo (2014). "Co-Occurrence of Design Patterns and Bad Smells in Software Systems: A Systematic Literature Review". In proceedings of the Workshop on Software Modularity (WMod), co-allocated with CBSoft. Maceio, Brazil.

S. Easterbrook et al. (2008). "Selecting Empirical Methods for Software Engineering Research". Guide to advanced empirical software engineering. Springer-Verlag. p. 285-311, 2008.

Erich Gamma, Ralph Johnson, Richard Helm, John Vlissides (1995). "Design Patterns: Elements of Reusable Object-Oriented Software". Pearson Education India.

Alessandro Garcia, Cláudio Sant'Anna, Eduardo Figueiredo, Uirá Kulesza, Carlos Lucena, and Arndt von Staa (2005). "*Modularizing Design Patterns with Aspects: A Quantitative Study*". In proceedings of the 4th International Conference on Aspect Oriented Software Development (AOSD), pp. 3-14.

William F. Opdyke and Ralph E. Johnson (1990). "*Refactoring: An Aid in Designing Application Frameworks and Evolving Object-Oriented Systems*". In SOOPPA Conference Proceedings, pages 145–161.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., and Wesslén, A. (2000) "*Experimentation in Software Engineering: an Introduction*." Kluwer Academic Publishers.

Meta-modelo para Mudança Organizacional em Melhoria de Processo de Software

Monica Anastassiu[1], Flavia Maria Santoro[2], Gleison Santos[1,2]

Programa de Pós-Graduação em Informática – Universidade Federal do Estado do Rio de Janeiro (UNIRIO)

Departamento de Informática Aplicada, Universidade Federal do Estado do Rio de Janeiro (UNIRIO)

{monica.anastassiu, flavia.santoro, gleison.santos}@uniriotec.br

Abstract. *Change involves process, people, skills, organizational culture and leadership, among other topics that include the critical factors to be observed for a successful change. This paper presents a conceptual meta model for organizational change, which is able to represent the important concepts involved with organizational change in a software process improvement (SPI) initiative such as stakeholders, resistance and competence. A case study was applied in a software process improvement initiative in an organization to assess the meta model applicability.*

Resumo. *Mudança envolve processo, pessoas, competências, cultura organizacional e liderança, dentre outros tópicos que englobam os fatores críticos a serem observados para uma mudança bem-sucedida. Esse artigo apresenta um meta-modelo conceitual de mudança organizacional, que é capaz de representar os conceitos importantes envolvidos com mudança organizacional em uma iniciativa de melhoria de processo de software, tais como stakeholders, resistência e competência. Um estudo de caso foi aplicado em uma iniciativa de melhoria de processo de software de uma empresa para avaliar a aplicabilidade do meta-modelo.*

1. Introdução

Para se manterem competitivas dentro de uma conjuntura volátil, as empresas são impulsionadas à realizarem mudanças no seu ambiente corporativo, como por exemplo introduzindo uma nova tecnologia. QUATTRONE e HOPPER (2001) declaram que a despeito da literatura sobre mudança organizacional difundir o porquê e o como das organizações mudarem, a definição de mudança não é trivial. BARCAUI (2012) declara que a mudança organizacional altera processos de trabalho e reestrutura áreas de negócio buscando ajustar o foco da organização para implantar novas estratégias e manter a condição de vantagem competitiva. No entanto, implantá-la não é uma tarefa simples, com uma receita certa e definitiva.

No domínio do processo de *software*, ALLISON e MERALI (2007) consideram melhoria de processos de *software* (SPI, do inglês *software process improvement*) como sendo a própria mudança, declarando que iniciativas em melhorias de processos de *software* são emergentes e restritas ao contexto no qual estão inseridas. BEECHAM *et al*. (2003) conduziram um estudo sobre os problemas identificados por profissionais envolvidos com implementações do modelo CMM (*Capability Maturity Model*), reportando

que diferentes níveis de maturidade reportam tipos diferentes de problemas.

Observa-se na literatura que é possível sintetizar que mudança organizacional é um tema complexo, envolvendo distintos aspectos e dimensões, dentro e fora de uma organização. Possui suas particularidades, tais como: seu caráter mais emergente do que previsível (ALLISON e MERALLI, 2007) e sua conexão com a transformação individual, já que a questão fundamental não é provocar a mudança, mas remover os bloqueios das pessoas para exercerem mais plenamente a sua espontaneidade (MOTTA, 1998). Demanda planejamento (BORIA *et al*., 2012), implementação, medição dos resultados obtidos ante ao novo *status quo* alcançado, bem como análise dos impactos gerados na organização como um todo (WINCEK *et al*., 2014). No entanto, não há um consenso para a definição de mudança organizacional, o que nos leva ao seguinte desafio: estabelecer uma conexão formal deste conjunto de conceitos e particularidades, de modo a obter um material que possa contribuir no conhecimento sobre o conceito de mudança organizacional. O corrente trabalho tem como objetivo principal definir a mudança organizacional no contexto de melhoria de processo de *software* e propor um meta-modelo conceitual de mudança organizacional que descreva conceitos envolvidos no tema e apresente os relacionamentos existentes entre eles. Dessa forma, pretende-se contribuir para as abordagens que tratam de melhoria em processos de *software*, apoiando o planejamento de estratégias e ações, principalmente àquelas relacionadas aos aspectos humanos de *software* como é o caso da resistência a mudanças. Adicionalmente, o meta-modelo conceitual poderá servir de modelagem inicial para a construção de um sistema de mudança organizacional.

O restante do artigo está estruturado da seguinte forma: a Seção 2 trata da fundamentação teórica, seguida de uma análise e discussão da literatura pesquisada. Na Seção 3 é apresentado o meta-modelo conceitual de mudança organizacional. Por fim, as considerações finais são apresentadas na Seção 4.

2. Fundamentação Teórica e Análise Crítica

Inicialmente foi conduzida uma revisão informal da literatura sobre mudança organizacional. Observou-se que o tema é amplo, possui aspectos distintos e desencadeia diferentes enfoques. Para auxiliar a análise do mapeamento sistemático que relacionou mudança organizacional à melhoria em processos de *software,* foram definidas questões de pesquisa e critérios, de exclusão e de inclusão, para filtrar as publicações inicialmente obtidas. As evidências identificadas sobre conceitos, abordagens, métodos, frameworks e modelos aplicados sobre o tema, em áreas distintas e sob diferentes enfoques, bem como sua análise crítica, estão sintetizadas na Tabela 1.

Tabela 1. Síntese da fundamentação teórica

Critério de análise: Existência de definição de mudança organizacional		
Fonte	**Conteúdo analisado**	**Análise Crítica**
Wincek *et al.* (2014), Cao *et al.* (2000) e Chrusciel e Field (2003)	Wincek *et al.* (2014) e Cao *et al.* (2000) definem mudança organizacional como qualquer mudança na posição ou na responsabilidade em uma organização (mudanças nas funções organizacionais) ou qualquer mudança em uma política organizacional ou procedimento (mudanças nos processos organizacionais), enquanto Chrusciel e Field (2003) fazem distinção entre mudança e mudança significativa ou de natureza estratégica, que é aquela onde existe impacto sobre a empresa devido a alguma adaptação organizacional radical.	Convergência do entendimento de que mudança organizacional, sobretudo, acarreta em mudança nos processos organizacionais e em mudanças nas funções organizacionais.

Critério de análise: Aspectos relacionados a mudanças organizacionais advindos de melhoria de processos, incluindo processo de *software*.		
Fonte	**Conteúdo analisado**	**Análise Crítica**
Quattrone e Hopper (2001)	Fatores contextuais são uma das razões de mudança organizacional.	Fator contextual não conceituado, não havendo uma especificidade quanto a quais são os fatores contextuais causadores de uma mudança organizacional e de que contexto o processo de gestão de mudança é dependente.
Nurcan e Rolland (2003)	Um processo de mudança não é determinístico porque se deve pressupor o contexto relacionado do qual o processo de gestão de mudança é dependente.	
Mathiassen *et al.* (2005)	Uma melhoria de processo de *software* bem-sucedida exige gestão eficaz da mudança levando em conta o contexto, dado que processo, estrutura, pessoas e gestão serão mudados de forma relacionada.	
Allison e Merali (2007)	Iniciativas de melhoria de processos de *software* são emergentes, originadas e restritas ao contexto no qual a melhoria de processos de *software* está inserida e também por ações das pessoas envolvidas nesse processo.	A adoção de modelos de maturidade é um exemplo de contexto no qual a melhoria está inserida.
Albuquerque (2014)	Apoio, comprometimento e envolvimento da alta direção são um dos principais fatores críticos de sucesso na manutenção de modelos de processos de *software* após a sua avaliação.	A compreensão é de que iniciativas e continuidade de programas de melhoria de processos de *software* estão principalmente relacionadas a aspectos humanos.
Beecham *et al.* (2003)	Questões relacionadas às pessoas são problemas relacionados às iniciativas de processos de melhoria de *software*, citados tanto por gerentes seniores quanto por gerentes de projeto e desenvolvedores.	
Heikkilä (2009), Boria *et al.* (2012) e Müller *et al.* (2010)	Heikkilä (2009) e Boria et al. (2012) associam a mudança ao processo de aprendizagem, assim como Müller et al. (2010) concluíram que conhecimento em mudança organizacional é chave para o sucesso de iniciativas de melhoria de processo de software.	Convergência no pensamento de que a aprendizagem é inerente ao processo de mudança e de que as competências relativas à mudança organizacional são necessárias para as iniciativas de SPI.
Beecham *et al.* (2003)	A escassez de competências é um problema relacionado às iniciativas de processos de melhoria de *software*.	
Critério de análise: Impactos causados por mudança organizacional		
Fonte	**Conteúdo analisado**	**Análise Crítica**
Boria *et al.* (2012)	A mudança diminui a produtividade pela necessidade da aprendizagem. Mas, se planejada de acordo com a cultura organizacional, reduz impactos.	Embora identificados alguns efeitos que a mudança possa gerar, não foi observada na literatura menção sobre a medição de tais efeitos sob a perspectiva da relação custo e benefício.
Zhao e Liu (2008)	Mudanças organizacionais aumentam o vigor da organização e dos funcionários, mantendo a vantagem competitiva sustentável.	
Wincek *et al.* (2014)	Necessidade de olhar o processo durante a mudança e verificar os riscos (impactos) após a mudança.	
Critério de análise: Aplicação de práticas de gerenciamento de mudanças aplicadas em processos de *software*		
Fonte	**Conteúdo analisado**	**Análise Crítica**
Boria *et al.* (2012)	A mudança deve ser planejada em etapas, de acordo com a cultura organizacional.	Existem divergências, entre os autores, quanto à adequação ou não de planejar uma mudança organizacional.
Nurcan e Rolland (2003)	O processo de gestão de mudança não pode ser totalmente prescrito por ser muito dependente do contexto no qual está inserido.	
Wincek *et al.* (2014)	Uma gestão de mudança organizacional efetiva deve incluir um sistema para gerenciar outras mudanças potenciais, tais como: modificação nas condições de trabalho, mudança pessoal, mudanças na alocação de tarefas e mudanças na hierarquia organizacional.	Mudanças organizacionais podem gerar outras mudanças organizacionais que precisam ser igualmente tratadas.

3. Meta-modelo Conceitual de Mudança Organizacional

O objetivo principal desta pesquisa é entender a mudança organizacional no contexto de melhoria de processo de *software* e definir uma meta-modelo, capaz de representar os conceitos importantes envolvidos com mudança organizacional, a fim de apoiar o planejamento de estratégias e ações para melhorias em processos de *software*, especialmente àquelas relacionadas aos aspectos humanos. O meta-modelo conceitual para mudança organizacional, mostrado na Figura 1, está representado por uma ontologia que utiliza parte de outras ontologias como a de contexto (MATTOS, 2012), onde é explicitado o conceito de elemento contextual e situação (contexto) e, a ontologia de processos de software (VILLELA, 2004) onde são explicitados os conceitos de artefato, documento, bem, bem de produção, *software* e *hardware,* competência, conhecimento, habilidade e experiência.

Com base na literatura pesquisada, 4 questões de competência (QC) foram formuladas para guiar a construção do meta-modelo conceitual para mudança organizacional. A QC1 refere-se aos aspectos que podem caracterizar o conceito de mudança organizacional, assim formulada: "O que caracteriza a mudança organizacional? "**.** Dentre os tipos de mudança organizacional pesquisados consideramos as classificações: estratégica e não estratégica. A QC2 – "O que origina a necessidade de mudança? ", baseia-se na observação de que a situação é apresentada como uma fonte motivadora para a necessidade de mudança organizacional, bem como um agente de possíveis adaptações nas estratégias e alternativas que são adotadas para as mudanças. O inverso desta questão também pode ser aplicado quando se constata que a mudança organizacional gera efeitos na organização podendo afetar processos, sistemas e estruturas organizacionais, dentre outros elementos, bem como afeta as diversas partes interessadas, internas e externas à organização, que apresentam respostas como forma de reagir às mudanças. Neste sentido, a QC3 elaborada é "Em que a mudança organizacional impacta? ". Finalmente, a ausência de trabalhos de pesquisa relacionados à medição sobre o resultado gerado por uma mudança, do ponto de vista do alcance do objetivo desta mudança, foi observada nos levando a QC4 – "De que maneira pode-se afirmar que o objetivo de uma mudança foi alcançado?".

3.1 Descrição e Relação entre as Classes do Meta-modelo Conceitual

A classe Mudança Organizacional é o foco deste meta-modelo conceitual apresentado na Figura 1. Seu relacionamento com as demais classes representa as informações e ligações chave para este trabalho. As classes e suas relações são descritas a seguir e apresentadas no meta-modelo na Figura 1. Para compreender a relação entre estas classes, descritas ao longo desta subseção, tomemos o exemplo a seguir. As classes estão citadas em caixa alta ao longo do exemplo.

Suponhamos que uma dada organização X, cuja CULTURA ORGANIZACIONAL é do tipo AUTORITÁRIA, tenha constatado que a previsão de suas futuras despesas com servidores e bancos de dados (*HARDWARE*), ultrapassaria o limite (ELEMENTO CONTEXTUAL) estabelecido pela ALTA ADMINISTRAÇÃO da empresa. Neste cenário, a seguinte SITUAÇÃO foi criada: o orçamento relativo aos BENS DE PRODUÇÃO deveria ser reduzido em 20%. Motivada por esta situação (NECESSIDADE DE MUDANÇA ORGANIZACIONAL), a empresa optou por descartar parte de seu *HARDWARE* com a aquisição dos serviços de tecnologia em nuvem para todos os seus sistemas e bases de informação (MUDANÇA ORGANIZACIONAL). Por afetar pelo menos um dos ELEMENTOS ORGANIZACIONAIS, os recursos de *HARDAWARE*, esta mudança organizacional foi considerada ESTRATÉGICA e, portanto, requereu um

PLANO. Do contrário, consequências tais como sistemas indisponíveis e problemas de acesso aos sistemas de informação e de confidencialidade de dados, afetariam a organização levando esta iniciativa ao insucesso. O projeto de mudança de plataforma tecnológica, que teve como OBJETIVO diminuir custos com bens de produção, afetou os EMPREGADOS João e Maria, administradores de bancos de dados (CARGO TÉCNICO), que se sentiram ameaçados em seus empregos respondendo (RESPOSTA) com uma REAÇÃO de RESISTÊNCIA. A ALTA ADMINISTRAÇÃO e os GERENTES, satisfeitos com a possibilidade de economizar um percentual significativo das despesas mensais da empresa X, porém sem o CONHECIMENTO e a HABILIDADE para lidar com uma mudança organizacional, responderam com uma reação de ADERÊNCIA, mas não souberam lidar com a RESISTÊNCIA de seus empregados, que prejudicou a implantação da nova tecnologia atrasando o prazo final em 20% do tempo inicialmente estimado.

A MUDANÇA ORGANIZACIONAL, como qualquer mudança na posição ou na responsabilidade em uma organização ou qualquer mudança em uma política organizacional ou procedimento que possam induzir riscos aos processos organizacionais considerados críticos (WINCEK *et al.*,2014), é tipificada como MUDANÇA ORGANIZACIONAL ESTRATÉGICA ou MUDANÇA ORGANIZACIONAL NÃO ESTRATÉGICA. Quando estratégica, a mudança organizacional impacta nas PARTES INTERESSADAS e/ou em um ou mais ELEMENTOS ORGANIZACIONAIS (RAJAGOPALAN e SPREITZER, 1996).

A PARTE INTERESSADA, conceituada como um indivíduo, grupo ou organização que possa afetar, ser afetado ou sentir-se afetado, direta ou indiretamente, por uma ou mais mudanças organizacionais (Adaptado do PMI, 2013), é tipificada como EMPREGADO, TERCEIRIZADO ou FORNECEDOR. O EMPREGADO, representado por um indivíduo que presta pessoalmente a outrem ou a uma organização, serviços não eventuais, subordinados e assalariados, pode ser um USUÁRIO (indivíduo ou grupo que se serve dos sistemas que apoiam os processos de negócio nos quais está envolvido), e/ou um AGENTE de MUDANÇA, responsável pela condução das mudanças organizacionais dentro da organização. O empregado tem um CARGO, que é o conjunto de atividades a serem executadas pelas pessoas que os ocupam, suas responsabilidades, competências desejadas, além das condições de trabalho oferecidas (VILLELA, 2004) e pode ter ou não uma FUNÇÃO, incumbência recebida independentemente de seu cargo. O cargo é tipificado como CARGO ADMINISTRATIVO (atribuições de nível administrativo), CARGO TÉCNICO (atribuições de nível técnico) ou CARGO EXECUTIVO (atribuições de nível executivo), sendo este último tipificado como GERENTE (função qualificada responsável por dirigir e gerir os assuntos de uma área estabelecida na estrutura organizacional) ou ALTA ADMINISTRAÇÃO (pelo corpo dos dirigentes máximos de uma organização).

O cargo executivo pode ser um PATROCINADOR, que é representado por um indivíduo ou grupo que fornece os recursos e suporte para o projeto de mudança organizacional, bem como toma decisões sobre o rumo do projeto (Adaptado do PMI, 2013). A parte interessada pode ser um CLIENTE (indivíduos ou organizações que recebem o produto, serviço ou resultado de um projeto ou de um processo) tipificada como CLIENTE INTERNO ou CLIENTE EXTERNO. A parte interessada pode ou não ter COMPETÊNCIA (o que torna as pessoas capazes de executar atividades que envolvem algum grau de dificuldade), tipificada como CONHECIMENTO (o que torna as pessoas capazes de executar atividades que envolvem algum grau de dificuldade), HABILIDADE (aptidão nata ou adquirida não associada a uma atividade ou domínio de conhecimento especifico) ou

EXPERIÊNCIA (adquirido através da prática, ou seja, através da execução de atividades), (VILLELA, 2004).

O ELEMENTO ORGANIZACIONAL, característica, domínio ou propriedade de uma organização potencialmente impactado por mudanças organizacionais (NWOKEJI *et al*, 2015), é tipificado como: ESTRUTURA ORGANIZACIONAL, representando uma linha de autoridade que interliga as posições da organização e define quem se subordina a quem (CHIAVENATO, 2004); MODELO DE GERENCIAMENTO, representando as principais determinações, vontades e expectativas do proprietário ou principal gestor, de como as coisas devem acontecer na organização (CROZATTI, 1998); PLANEJAMENTO ESTRATÉGICO, representando objetivos, investimentos e planos a partir de uma análise dos pontos fortes, pontos fracos, oportunidades e ameaças relacionadas à organização (MINTZBERG *et al.* 2000); PROCESSO, representando uma coleção de atividades relacionadas, iniciadas em resposta a um evento, para atingir um resultado específico para o seu cliente (SHARP e McDERMOTT, 2010) ou ARTEFATO.

O ARTEFATO, qualquer elemento produzido pelo homem e não por causas naturais, podendo exercer diferentes papéis em uma organização, tais como o de insumo ou produto de uma atividade, é tipificado como DOCUMENTO (artefato escrito ou impresso, cuja função é fornecer informação, conhecimento ou prova) ou BEM (artefato concluído, no sentido de não participar da composição de outros artefatos, além de não serem documentos). O bem é tipificado como BEM DE PRODUÇÃO, cuja funcionalidade apoia a criação ou a transformação de artefatos, que por sua vez é tipificado como *HARDWARE* (representado por um computador, um de seus periféricos ou uma máquina qualquer operada com o auxílio de um *software*) ou *SOFTWARE* (representado por um conjunto de instruções e dados que, utilizado em conjunto com um hardware, é capaz de executar ou apoiar a execução de atividades) (VILLELA, 2004).

O PROCESSO é tipificado como PROCESSO DE NEGÓCIO, representando um conjunto de atividades parcialmente ordenadas que visam atingir um objetivo (HAMMER e CHAMPY, 1993), ou PROCESSO DE DESENVOLVIMENTO DE *SOFTWARE* como um conjunto de atividades estruturadas e destinadas a resultar em um artefato ou serviço de valor para a organização ou para um determinado cliente ou mercado. Implica em uma ordenação específica das atividades com começo, fim, insumos e produtos claramente identificados (VILLELA, 2004).

Uma mudança organizacional pode gerar outras mudanças organizacionais. Uma mudança estratégica está associada a um ou mais PLANOS (conjunto de ações voltadas para o alcance de um determinado objetivo) que podem conter um ou mais SUBPLANOS a serem usados para alcançar um ou mais OBJETIVOS (o que se espera obter como resultado a partir de uma mudança organizacional). Os objetivos devem ser mensurados por meio de METAS, representando a quantificação de um objetivo possibilitando medir seu alcance. A unidade de medição pode ser tempo, qualidade, quantidade, moeda etc.

MUDANÇA ORGANIZACIONAL ESTRATÉGICA tem uma relação de requisição com o conceito de plano e de e subplano que, por sua vez, tem uma associação com o objetivo da mudança organizacional. O plano ou subplano é composto por ATIVIDADES, representando o conjunto de ações destinadas a alcançar um ou mais objetivos, que consome e produz informações e artefatos e requer os intervenientes para executá-lo, estabelecendo o foco (MATTOS, 2012). O plano é executado em uma das fases do CICLO

DE VIDA DE MUDANÇA que representa a série de fases pelas quais uma mudança organizacional passa do início ao término (adaptado do PMI, 2013).

Mudanças organizacionais são causadas por uma ou mais NECESSIDADES DE MUDANÇA ORGANIZACIONAL que representam a percepção de que pode ser necessário acontecer uma determinada mudança organizacional. Uma ou mais de uma necessidade de mudança organizacional pode ser motivada por uma SITUAÇÃO (conjunto de elementos contextuais instanciados que motiva a necessidade de uma adaptação), caracterizada por um conjunto de ELEMENTOS CONTEXTUAIS (propriedade usada para caracterizar uma entidade contextual, esta última podendo ser, por exemplo, uma pessoa, um lugar, um objeto, um usuário ou uma aplicação. Esta propriedade é identificada por um conjunto de atributos e relacionamentos associados a uma entidade contextual) (MATTOS, 2012).

A CULTURA ORGANIZACIONAL, parte do elemento organizacional que é um sistema de valores compartilhados pelos membros de uma organização, diferindo de uma para outra (ROBBINS, 2005), é tipificada como AUTORITÁRIA ou PARTICIPATIVA.

Uma parte interessada, quando afetada por uma mudança organizacional, retorna uma RESPOSTA revelando a sua REAÇÃO (comportamento de uma parte interessada associado à resposta). A reação é tipificada como ADERÊNCIA (aprovação ou estímulo), NEUTRALIDADE (imparcialidade) ou RESISTÊNCIA (rejeição).

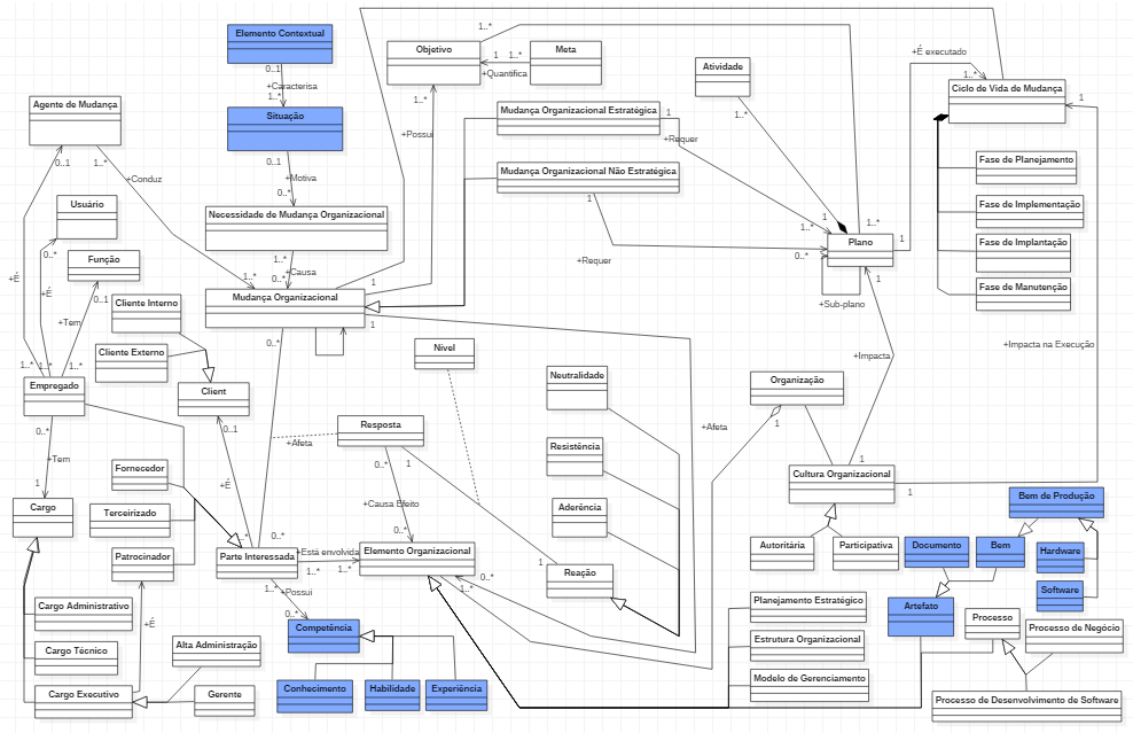

Figura 1. Modelo Conceitual de Mudança Organizacional

3.2 Avaliação do Meta-modelo Conceitual

A aplicabilidade do meta-modelo foi avaliada por meio de um estudo de caso exploratório e retroativo, mediante o histórico de uma iniciativa de melhoria no processo de software de uma empresa atuante na área de previdência complementar. Os dados foram coletados a partir do relato de dois gestores que participaram ativamente da implementação de melhorias no processo de software da empresa e, posteriormente, utilizados para instanciar

o meta-modelo. A análise do instanciamento foi baseada em dois critérios: (i) completude das classes do modelo para verificar se as classes do modelo respondem a todos os fatos relacionados à mudança ocorrida e; pertinência das classes aos fatos para verificar se as classes identificadas na correlação com os fatos estão perfeitamente adequadas a eles e se as correlações que os fatos apresentam estão identificadas nas relações entre as classes a eles atribuídas.

A análise dos dados evidenciou que o meta-modelo é capaz de representar parcialmente os conceitos importantes envolvidos com mudança organizacional no contexto de uma iniciativa de melhoria de *software*, o que gerou a versão revisada do meta-modelo apresentada na Figura 1.

4. Considerações Finais

Para uma mudança organizacional ser bem-sucedida é importante considerar tópicos como processo, pessoas, competências, cultura organizacional e liderança, dentre outros, que incluam fatores críticos de sucesso a serem observados. O objetivo principal deste trabalho foi entender a mudança organizacional no contexto de melhoria de processo de *software* e definir um meta-modelo conceitual de mudança organizacional capaz de representar os conceitos importantes envolvidos com mudança organizacional, a fim de apoiar o planejamento de estratégias e ações para melhorias em processos de *software*, especialmente àquelas relacionadas aos aspectos humanos.

O meta-modelo conceitual de mudança organizacional construído teve sua aplicabilidade avaliada por meio do planejamento e realização de um estudo de caso. A análise dos dados evidenciou que o meta-modelo é capaz de representar parcialmente os conceitos importantes envolvidos com mudança organizacional no contexto de uma iniciativa de melhoria de *software*, o que gerou a versão revisada do meta-modelo apresentada na Figura 1.

Associado aos demais trabalhos pesquisados, o diferencial deste trabalho é a proposta de descrição formal de conceitos relacionados à mudança organizacional, podendo ser usada para apoiar o planejamento, a implantação e a manutenção de iniciativas de melhoria de processo de *software,* bem como servir de modelagem inicial para a construção de um sistema de mudança organizacional.

Observada a relevância dos aspectos humanos perante iniciativas de melhorias de processo de *software*, representados no meta-modelo pelas classes *Stakeholder*, *Response* e *Reaction*, elegemos como foco do trabalho a ser desenvolvido de agora em diante os temas "resistência das pessoas ante a uma iniciativa de melhoria de processo de *software*" e "motivação das pessoas com melhoria de processo de *software*". A resistência, a aceitação a mudanças e a motivação das pessoas, assim como o envolvimento da alta administração, o treinamento e a comunicação, sendo estes últimos fortemente relacionados aos três primeiros, são fatores críticos de sucesso abordados por ALBUQUERQUE (2014), PINO *et al.* (2008), MONTONI (2010), NIAZI *et al*. (2006), KOUZARI *et al*. (2015). Compreender os fatores críticos de sucesso em iniciativas de melhoria de processos de software é fundamental para apoiar a gerência de iniciativas de melhoria e de melhores práticas de implementação MONTONI (2010).

Agradecimentos

Os autores agradecem à CAPES, à FAPERJ (projetos E-26/203.446/2015 – BBP, E-

26/210.643/2016) e à UNIRIO (Edital PQ-UNIRIO no 01/2016) pelo apoio financeiro.

Referências Bibliográficas

Albuquerque, R. (2014), "Estudo sobre Fatores que influenciam a Manutenção de Processos de Software em Empresas avaliadas por Modelos de Referência". Dissertação de Mestrado, Pontifícia Universidade Católica do Paraná, PR, Brasil.

Allison, I., Merali, Y. (2007), "*Software* process improvement as emergent change: A structurational analysis", In: Information and *Software* Technology, Vol. 49, pp. 668–681.

Barcaui, A. (2012), PMO Escritórios de Projetos, Programas e Portfólio na prática, Brasport, 5ª edição.

Beecham, S., Hall, T., Rainer, A. (2003), "Software Process Improvement Problems in Twelve Software Companies: An Empirical Analysis", In: Empirical Software Engineering, Vol. 8, ed. 1, pp. 7-42.

Boria, J., Rubinstein, V., Rubinstein, A. (2012), "Cambio y Cultura". WAMPS 2012.

Cao, G., Clarke, S., Lehaney, B. (2000), "A systemic view of organizational change and TQM", In: The TQM Magazine, Vol. 12, Iss 3, pp. 186 - 193.

Chiavenato, I. (2004), Administração nos Novos Tempos, 2ª edição. Rio de Janeiro, Elsevier.

Chrusciel, D. and Field, D. W. (2003), "From Critical Success Factors into Criteria for Performance Excellence – An Organizational Change Strategy", In: Journal of Industrial Technology, Vol. 19, No. 4. (August 2003 to October 2003).

Crozatti, J. (1998), "Modelo de gestão e cultura organizacional: conceitos e interações ". Caderno de Estudos, No 18, São Paulo.

Hammer, M., Champy, J. (1993), "Reengineering the Corporation: A Manifesto for Business Revolution", Eua: Harper Business Essentials.

Heikkilä, M. (2009), "Learning and Organizational Change in SPI Initiatives", Springer-Verlag Berlin Heidelberg 2009.Holanda, A.B., (1999), Novo Aurélio Século XXI - O Dicionário da Língua Portuguesa, 5a ed., Nova Fronteira.

Kouzari, E., Gerogiannis, V. C., Stamelos, I., Kakarontzas, G. (2015), "Critical Success Factors and Barriers for Lightweight Software Process Improvement in Agile Development A Literature Review", In: 10th International Conference on Software Engineering and Applications (ICSOFT-EA-2015), pages 151-159.

Mathiassen, L., Ngwenyama, K. O., Aaen I. (2005), "Managing Change in Software Process Improvement", IEEE SOFTWARE 2005.

Mattos, T. C. (2012), "Caracterização de situações em processos de negócio sensíveis a contexto". Dissertação de Mestrado, PPGI/UNIRIO, Rio de Janeiro, RJ, Brasil.

Mintzberg, H., Ahlstrand, B., LAMPEL, J. (2000), Safári de Estratégia: um Roteiro pela Selva do Planejamento Estratégico, Porto Alegre, Bookman.

Montoni, M. A. (2010), "Uma Investigação sobre os Fatores Críticos de Sucesso em Iniciativas de Melhoria de Processos de Software". Tese de Doutorado, COPPE – Universidade Federal do Rio de Janeiro, RJ, Brasil.

Motta, P. R. (1998), "Transformação Organizacional - A Teoria e a Prática de Inovar", Editora Qualitymark.

Müller, S. D., Mathiassenb, L., Balshøj, H. H. (2010), "Software Process Improvement as Organizational Change: A metaphorical Analysis of the Literature", In: The Journal of Systems and Software 83 (2010) 2128–2146.

Niazi, M., Wilson, D., Zowghi, D. (2006), "Critical success factors for software process improvement implementation: An empirical study", In: Software Process Improvement and Practice, Volume 11, n. 2, Pages 193-211.

Nurcan, S. and Rolland, C. (2003), "A Multi-Method for Defining the Organizational Change", In: Information and Software Technology, Volume 45, Issue 2, Pages 61–82.

Nwokeji *et al*. (2015), "A Data-Centric Approach to Change Management", In: Enterprise Distributed Object Computing Conference (EDOC), 2015 IEEE 19th International.

Pino, F. J., García, F., & Piattini, M., 2008. Software process improvement in small and medium software enterprises: a systematic review. Software Quality Journal. 16(2), 237-261.

Quattrone, P. and Hopper, T. (2001), "What does Organizational Change Mean? Speculations on a Taken for Granted Category", In: Management Accounting Research, 2001, 12, 403–435.

Rajagopalan, N., Spreitzer, G.M. (1996), "`Towards a theory of strategic change: a multi-lens perspective and integrative framework", In: Academy of Management Review, Vol. 22 No. 1, pp. 48-79.

Robbins, S. Organizational Behavior. 11th ed., New Jersey: Prentice-Hall, Pearson Education, 2005.

Sharp A., McDermott P. (2010), Workflow Modeling: Tools For Process Improvement And Application Development. Norwood, Ma, USA: Artech House.

Villela, K. (2004) "Definição e Construção de Ambientes de Desenvolvimento de Software Orientados à Organização", Tese de D.Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil, maio. Disponível em http://www.cos.ufrj.br/taba.

Zhao, Y., Liu, Y. (2008), "Organizational change: A case study on Anhui Telecom Company". 2008 International Seminar on Business and Information Management.

Wincek, J., Sousa, L. S., Myers, M. R., Ozogc, H. (2014), "Organizational Change Management for Process Safety", In: Wiley Online Library (wileyonlinelibrary.com). DOI 10.1002/prs.11688.

Explorando a Personalidade do Desenvolvedor em Ecossistemas de Software Móvel

Bruno Pedraça de Souza[1], Bruno Araújo Bonifácio[1], Priscila Silva Fernandes[1], Awdren L. Fontão[2], Arilo Claudio Dias-Neto[2]

[1]Instituto de Ciências Exatas e Tecnologia - ICET/ UFAM
Rua Nossa Senhora do Rosário, Itacoatiara, Amazonas - Brasil

[2]Instituto de Computação – Universidade Federal do Amazonas (UFAM)
Caixa Postal 69077-000 – Manaus – AM – Brazil

{brunopedraca.es, brunnoboni, pry.bila}@gmail.com,

{awdren, arilo}@icomp.ufam.edu.br

Abstract. *In a mobile software ecosystem (MSECO), the external developer is essential element and the central organizations (e.g. Google, Apple and Microsoft) have been investing in actions for the engagement of the developer so that MSECO expands in contributions with quality (e.g. download and the users' evaluations). Inside of this scenery, the human factors compose the developer's experience (DX) and, among them, the personality can have impact in the execution of processes for the developer. In that context, the present paper presents a study of exploratory case for analysis of the influence of the personality type in DX in MSECO. The results indicate a correlation between the psychological profile and the developers' productivity in a new platform of mobile software.*

Resumo. *Em um ecossistema de software móvel (MSECO), o desenvolvedor externo é elemento essencial e as organizações centrais (e.g. Google, Apple e Microsoft) tem investido em ações para o engajamento do desenvolvedor para que o MSECO se expanda em contribuições com qualidade (e.g. downloads e avaliações dos usuários). Dentro deste cenário, os fatores humanos compõem a experiência do desenvolvedor (DX) e, dentre eles, a personalidade pode ter impacto na execução de processos pelo desenvolvedor. Nesse contexto, o presente artigo apresenta um estudo de caso exploratório para análise da influência do tipo de personalidade na DX em MSECO. Os resultados indicam indícios de uma correlação entre o perfil psicológico e a produtividade de desenvolvedores em uma nova plataforma de software móvel.*

1. Introdução

As aplicações móveis (apps) compõem o modelo de negócio que envolve os dispositivos móveis, como *smartphones*, *tablets*, *smartwatches*, onde as organizações detentoras das plataformas (e.g. *Apple*, *Google* e *Microsoft*) disponibilizam ferramentas, lojas de *apps*, portais de suporte ao desenvolvedor para ajudar na expansão da plataforma por meio da criação de *apps* e disponibilização para os usuários.

Esse ambiente é chamado na literatura existente de Ecossistema de Software Móvel (MSECO), em uma analogia com Ecologia consiste na participação de elementos como o desenvolvedor, a organização central, a *app*, que cooperam/competem entre si para a expansão do ecossistema (LIN e YE, 2009). Diante disso, os desenvolvedores tornam-se elementos fundamentais para a estrutura organizacional, pois somente com a estrutura interna uma organização não conseguiria atender toda a demanda de *apps* dos usuários de sua plataforma e estimula os desenvolvedores de software a utilizar suas plataformas e seus meios de distribuição para que eles possam oferecer inovações tecnológicas (JAIN, 2011).

Para isso a organização central precisa trabalhar mecanismos para a entrada, engajamento e retenção de desenvolvedores no MSECO, como forma de manter a estrutura e expandir o ecossistema em quantidade e qualidade de desenvolvedores (FONTÃO *et al.*, 2015b). O desenvolvedor possui expectativas ao começar a participação em um ecossistema, tem experiências durante o processo de desenvolvimento e ao publicar uma *app* na loja possui sentimentos e percepções acerca de sua contribuição.

Por isso, percepções e sentimentos externos baseados em aspectos cognitivos podem influenciar na produtividade (criação de contribuições) dentro dos MSECOs. Um desenvolvedor que não tem uma boa experiência durante a utilização de algum artefato pode abandonar ou não se dedicar para a produção de uma *app* de qualidade (que atenda os critérios da plataforma) (FAGERHOLM e MÜNCH, 2012). Sintomas como a desistência de desenvolvedores e queda na quantidade de *apps* e avaliações de usuário, são gerados na saúde do ecossistema que está relacionada ao desempenho de cada elemento e do todo (FONTÃO e DIAS-NETO, 2016).

Nesse contexto, o presente trabalho visa explorar o que se pode aprender do perfil psicológico do desenvolvedor, durante o processo de desenvolvimento em um MSECO. Para isto, foi realizado um estudo de caso exploratório, durante um treinamento oficial de desenvolvedores Windows da Microsoft. Neste estudo com um grupo de 15 desenvolvedores foi utilizado o indicador de perfil psicológico MBTI[1] (MYERS; BRIGGS, 2015) para classificar os perfis. Os desenvolvedores ainda utilizavam uma mesma abordagem baseada em processos para certificação de aplicações móveis.

O restante deste artigo está organizado da seguinte forma: a Seção 2 apresenta os conceitos e trabalhos relacionados. A Seção 3 apresenta os materiais e métodos do estudo experimental. A Seção 4 consiste nos resultados obtidos. A Seção 5 apresenta as ameaças a validade do estudo. E por fim, a Seção 6 consiste na conclusão e perspectivas de trabalhos futuros.

2. Conceitos e Trabalhos Relacionados

2.1. MSECO

De acordo com Bosch (2009), um Ecossistema de Software (ECOS) pode ser relatado como um entrelaçado de elementos interagindo entre si, e soluções de software que

[1] http://inspiira.org/

sustentam essas interações, com o suporte de uma plataforma tecnológica. A evolução da computação móvel tem instigado os usuários a integrar gradualmente tecnologias para a realização de afazeres diários, dado que além de permitir as operações básicas de um aparelho celular, os aparelhos móveis passaram a agregar novos recursos e facilidades, tais como a Internet móvel e a personalização por meio de aplicações (MIAO e YUAN, 2005).

Um MSECO pode ser definido como um grande sistema de cooperação de aplicações móveis, desenvolvedores e usuários que formam complexos relacionamentos e preenchendo nichos, competindo e cooperando, de uma forma similar a ecossistemas biológicos (LIN e YE, 2009). O MSECO está divido em elementos e atividades.

Cada elemento possui responsabilidade dentro de um MSECO, assim como um desempenho esperado pela organização para que o ecossistema se mantenha "saudável". O desempenho destes elementos precisa ser acompanhado com o objetivo de identificar e prever áreas de melhorias, assim como validar mudanças (MANIKAS e HANSEN, 2013).

Os elementos indicados por Fontão *et al.* (2015a) que compõem um MSECO detalhados a seguir: Aplicação Móvel, Desenvolvedores e Usuários podem ser incluídos a Organização Central, que é responsável pelo gerenciamento do MSECO; os Evangelistas, são especialistas internos da organização alocados para suportar os desenvolvedores; e a Loja de Aplicações, que consiste num repositório de aplicações e que possui um papel fundamental para o sucesso do MSECO, definindo políticas de aceitação de aplicações móveis que irão ser disponibilizadas aos usuários.

Os elementos executam um fluxo de atividades, consumindo e produzindo artefatos, no caso, do desenvolvedor há um processo de desenvolvimento para a produção de uma *app* que irá compor a loja do MSECO. Para a organização há um processo de orquestração, que tem como objetivo prover a infraestrutura necessária para a expansão do ecossistema.

2.2. Perfil Psicológico

Pesquisas relacionadas sobre fatores humanos tornaram-se mais vigorosos com o fim da mobilização pela 2ª Guerra Mundial. A computação, por volta dos anos 60, passa a se desenvolver no caminho da utilização de linguagens de programação com maior grau de generalização, possibilitando a ideação de sistemas grandes, mais complexos e de aplicação variada (FAQUIN *et al.*,2016).

Assim, houve pesquisas sobre fatores humanos e suas personalidades, definindo-se personalidade como sendo um agregado de elementos internos específicos do caráter de uma pessoa que influencia o comportamento em circunstâncias diferentes (SCHULTZ, 2002). Segundo Cruz *et al.* (2011), a forma do teste de personalidade mais utilizado em estudos em Computação é o Indicador MBTI.

O MBTI (*Myers-Briggs Type Indicator*) é um tipo de informativo para descobrir peculiaridades importantes de pessoas, pontos fortes e aspectos chaves de desenvolvimento (Myers e Briggs, 2015). O MBTI possui 16 possibilidades de tipos psicológicos. Alicerçado nos estudos de Jung (1991), é estabelecido com base em quatro dimensões bipolares da personalidade: (1) Extrovertidos e Introvertidos (E/I), (2)

Sensoriais e Intuitivos, (3) Racionais e Emocionais; e (4) Julgadores e Perceptivos. A seguir, a Tabela 1 exibe a definição de cada tipo de personalidade.

O MBTI admite que, todos os indivíduos tenham as qualidades de personalidade contidos em cada escala ou parâmetro, cada um escolhe algumas qualidades ou é mais confortável com alguns traços do que outros. Os tipos de personalidades são identificados por meio de um código de quatro letras (PAIXÃO *et al.*, 2013).

Tabela 1. Tipos de Perfil Psicológico

Temperamento	Descrição
SP (Sensorial Perceptivo) - Artesãos	Têm uma grande afinidade com ferramentas e instrumentos. Buscam a sensação e suas ações têm como objetivo fazer com que cheguem o mais rápido possível até onde pretendem ir. São impulsivos, adaptáveis, competitivos e ousados.
SJ (Sensorial Julgador) - Guardiões	Preferem trabalhar consistentemente com o sistema, porque acreditam que, ao longo prazo, lealdade, disciplina e cooperação realizam o trabalho de forma correta. Buscam a segurança e são cuidadosos quanto aos prazos e têm uma visão perspicaz para o excesso ou escassez. São cautelosos quanto às mudanças.
NF (Intuitivo Sentimental) – Idealistas	Têm um talento único para ajudar as pessoas a resolver as suas diferenças e assim trabalharem juntas. Buscam a identidade e tendem a concentrarem-se no que poderia ser, em vez de concentrarem-se no que é. Altamente éticos em suas ações, sempre se comprometem a um estrito padrão de integridade pessoal.
NT (Intuitivo Pensador) – Racionais	São rigorosamente lógicos e independentes em seu pensamento. Têm um intenso desejo de atingir seus objetivos. Esforçam-se por compreender o mundo natural em toda a sua complexidade. São pragmáticos acerca do como ganharão esse conhecimento. Sentem-se à vontade ao encontrar as soluções mais eficientes e elegantes para os problemas.

2.3. Trabalhos Relacionados

Diversos autores propuseram pesquisas para investigar o tipo de personalidade de pessoas da área de tecnologia da informação (Branco *et al.*, 2012; Paixão *et al.*, 2013; Branco *et al.*, 2015). Algumas pesquisas almejam descobrir o tipo de personalidade apropriada para tarefas específicas (WANG e LI, 2009). Para verificar esse tipo de personalidade e qual personalidade a pessoa possui, ela pode responder o questionário MBTI. Após isso, tem-se a resposta mostrando qual o perfil psicológico desse participante.

Paixão *et al.* (2013) realizaram um estudo empírico para observar o perfil psicológico de alunos de computação. Foi observado aspectos individuais que podem afetar no aprendizado e desempenho acadêmico dos alunos. No estudo, foi feita uma comparação entre perfis de alunos que concluíram o curso e com alunos que desistiram, além da análise dos alunos periodizados e não periodizados, pois pode-se averiguar as causas da desistência dos alunos. O artigo apresentou resultados da existência de relação entre o perfil psicológico do aluno e sua motivação no curso de computação.

Branco *et al.* (2015) propuseram um estudo para analisar qual a influência da personalidade que gerente de projeto possui, assim como o seu comportamento no

ambiente de trabalho, pode ajudar ou prejudicar a equipe ao longo do desenvolvimento de um software. Branco *et al.* (2015) levaram em consideração dois teorias psicológicas: MBTI e a teoria das funções da equipe baseada no trabalho de Belbin (2010). Teve como resultado que o perfil do gerente de projeto tem um impacto significante na equipe de desenvolvimento. Todavia, Branco *et al.* (2015) sugerem que algumas características como criatividade e habilidade de ideias em práticas sejam levadas em consideração. Essa pesquisa realizou apenas estudo com apenas uma companhia de desenvolvimento de software e não se pode generalizar os dados. Um fator positivo, foi a aplicação da pesquisa direto na indústria. A relação destas pesquisas são o estudo do perfil psicológico dentro de uma equipe de projeto de software.

Outro fator importante é que os vários trabalhos relacionados propõem pesquisas sobre aprendizagem dentro de um curso de computação. Ou ainda, estudos sobre o perfil psicológico dentro de organizações que desenvolvem software, todavia, não foram realizados trabalhos que investigam tipos de personalidade dentro do MSECO (FONTÃO *et al.*, 2014) (FONTÃO *et al.*,2015a).

3. Estudo de Caso

Um estudo de caso investiga um fenômeno contemporâneo dentro do contexto da vida real (principalmente quando os limites fenômeno-contexto não estão claros). Quando este estudo é do tipo exploratório, ele ajuda na investigação inicial de algum fenômeno para derivar novas hipóteses e construir teorias.

Logo, a questão de pesquisa que norteia este estudo é "*O que é possível investigar a partir da personalidade durante um processo de desenvolvimento em MSECO?*"

3.1. Participantes

Os participantes deste estudo foram provenientes de uma seleção de desenvolvedores dos cursos de graduação (Engenharia de Software e Sistemas de Informação) de uma Universidade Federal. Todavia, para participar do estudo, os desenvolvedores tiveram que possuir dois pré-requisitos:

- Manifestar interesse em participar do estudo, concordando com o Termo de Consentimento Livre e Esclarecido e Formulário de caracterização, para ter-se conhecimento do grau de experiência de cada desenvolvedor.
- Para participar do treinamento, os desenvolvedores tiveram ainda que responder um questionário para a identificação de seu perfil psicológico.

Um total de 15 desenvolvedores foram convidados a participar do estudo. Levando-se em consideração os cursos de graduação e conhecimento prévio em desenvolvimento de aplicações móveis. Quanto ao conhecimento em desenvolvimento de aplicações móveis e o curso de graduação, o formulário de caracterização do participante apoiou no conhecimento sobre os participantes.

Todos os participantes possuíam conhecimento prévio em desenvolvimento de aplicações móveis para o MSECO *Android* como parte de conhecimentos adquiridos em disciplinas do curso de graduação.

3.2. Execução do Estudo

Inicialmente, os participantes responderam ao Termo de Consentimento Livre e Esclarecido (TCLE), ao Formulário de Caracterização visando identificar o perfil e experiência em desenvolvimento de aplicações móveis.

O estudo, por mais que tenha sido realizado num ambiente acadêmico, está dentro do contexto real de uma atividade desenvolvimento de aplicações móveis em um MSECO. É comum as organizações estabelecem parcerias com instituições para o treinamento de desenvolvedores (Fontão *et al.,* 2014). O estudo consistiu na execução do treinamento e acompanhamento dos desenvolvedores.

Para a execução do treinamento no desenvolvimento de aplicações móveis para o MSECO *Windows Phone*, o instrutor foi um evangelista oficial da Microsoft Brasil com experiência na plataforma *Windows Phone*. A plataforma utilizada foi a *Windows Phone 7.5*, com o *Visual Studio 2012*, esta configuração era suportada pela infraestrutura oferecida pela universidade.

O treinamento utilizado para a turma, considerando um curso básico: a) conceitos de desenvolvimento de aplicações móveis; b) ferramentas de desenvolvimento; c) padrões de desenvolvimento; d) interface de usuário; e) classes, métodos e comportamento da aplicação móvel; f) depuração de código; g) empacotamento de aplicação móvel; h) publicação de aplicações móveis. E as duas turma tinham como meta o desenvolvimento e a publicação da aplicação móvel. A carga horária de treinamento para a da turma foi de 18 horas.

Após a execução do treinamento, os desenvolvedores iniciaram a construção das suas aplicações móveis. Após isto, o evangelista iniciou uma fase de acompanhamento dos desenvolvedores, durante duas semanas após o treinamento. Para isto, grupos no *WhatsApp* e *Facebook* foram criados para a Turma. O acompanhamento realizado teve o objetivo de concluir o desenvolvimento de aplicações e submetê-las a loja de aplicações móveis.

4. Análise Qualitativa

Nesta seção, analisamos de forma qualitativa os comentários dos participantes de cada turma. Os comentários dos participantes, relativos ao questionário pós-treinamento, se encontram a seguir:

Os ***participantes com temperamento SP (artesãos)***, que possuem uma afinidade grande com ferramentas e instrumentos, relacionaram seus comentários ao aproveitamento do conhecimento adquirido durante o treinamento e a fase de acompanhamento. Como pode ser observado na Tabela 2.

Tabela 2. Comentários Pós-Treinamento - participantes SP, SJ, NF e NT

Participante	Comentário
SP1	"O treinamento foi proveitoso, apesar de pouco tempo, porém o que dificultou foi em relação a infraestrutura, a conexão de internet."
SP2	"Deveria ter outros treinamentos assim e ganhar conhecimento em outras plataformas é sempre interessante. "

Os ***participantes SJ (guardiões)***, que são cuidadosos quanto a prazos e são cautelosos quanto a mudanças, mencionam em seus comentários a abordagem utilizada

para o treinamento que promoveu o entendimento dos processos que ajudam a produzir uma aplicação móvel com qualidade, entendimento de oportunidades do mercado e que o conhecimento foi bem assimilado, inclusive, ajudando na expansão de ideias.De acordo com a Tabela 3 abaixo.

Tabela 3. Comentários Pós-Treinamento - participantes SP, SJ, NF e NT

Participante	Comentário
SJ1	"Gostei do treinamento, principalmente da abordagem sobre as atividades que devem ser realizadas durante o desenvolvimento, pois, muitas vezes nos preocupamos muito com o desenvolvimento e esquecemos dos processos que devem ser realizados para obter melhor qualidade. "
SJ2	"O treinamento serviu principalmente para abrir uma nova possibilidade de mercado, uma vez que quando se aprende a utilizar algo novo, e uma nova porta que se abre. "
SJ3	"Como nunca havia desenvolvido nada para dispositivos do Windows Phone o treinamento só agregou conhecimentos para mim. Não houve somente conhecimentos passados sobre a programação, mas também sobre o processo no decorrer de todo o desenvolvimento. Além disso é sempre bom ter contato com quem trabalhou/trabalha em empresas, pois eles têm uma visão diferente de quem só ficou no meio acadêmico. O instrutor foi calmo e atencioso com os participantes do treinamento e nos ajudou a expandir as ideias. "
SJ4	"Foi uma experiência muito boa e enriquecedora e apesar do período de tempo ser curto, as aulas foram bem ministradas e consegui assimilar bem o assunto apresentado. As aulas foram bem dinâmicas, houve teoria e prática nos exercícios e o professor sempre tirando dúvidas e acompanhando o desempenho dos alunos. "

Os ***participantes do temperamento NF (idealistas)***, que acreditam que a melhor forma de atingir objetivos é a partir da cooperação, comentaram principalmente sobre o conhecimento adquirido no treinamento e sobre o evangelista destacando a atenção com os desenvolvedores, o esclarecimento de dúvidas, a paciência de ensinar e o uso de uma linguagem compreensível. Conforme a Tabela 4.

Tabela 4. Comentários Pós-Treinamento - participantes SP, SJ, NF e NT

Participante	Comentário
NF1	"O treinamento foi muito proveitoso, no sentindo de podermos aprendermos a utilizar mais uma plataforma de desenvolvimento de aplicativos, onde o instrutor se fez prestativo e atencioso ao nosso aprendizado. "
NF2	"Excelente instrutor, durante o treinamento esclareceu dúvidas, dando atenção para cada participante. Entre tanto em relação a infraestrutura da instituição ficou um pouco a desejar, exemplo disso, foi a internet. "
NF3	"O instrutor/evangelista fez um ótimo trabalho, teve paciência de ensinar e usou uma linguagem compreensível. O treinamento só fez aprimorar mais ainda o conhecimento sobre desenvolvimento e nos mostrou uma nova ferramenta. "

Os ***participantes NT (racionais)***, que são pragmáticos quanto a forma que ganharão conhecimento, avaliaram o conteúdo abordado durante o treinamento como: as dicas de publicação e de desenvolvimento, conforme a Tabela 5.

Tabela 4. Comentários Pós-Treinamento - participantes SP, SJ, NF e NT

Participante	Comentário
NT1	"Será bem-vindo outros treinamentos iguais a esses. "
NT2	"Gostei bastante do treinamento, apesar de ter chegado atrasado no primeiro dia consegui acompanhar as atividades, com ajuda dos alunos e do instrutor."
NT3	"Ótimo treinamento, principalmente na parte de dicas de publicação da aplicação móvel e de dicas na hora do desenvolvimento. "

5. Ameaças à Validade

Nesta Seção são descritas as possíveis ameaças à validade deste estudo. As ameaças são classificadas em quatro categorias descrita a seguir: validade interna, validade externa, validade de conclusão e validade de constructo.

Validade interna: Quanto aos instrumentos apropriados, foram utilizados para a turma, desde os questionários aplicados a um piloto dos materiais, a ferramenta de desenvolvimento (*Visual Studio*) e o acesso comum à Central do Desenvolvedor da Microsoft. Em relação a *seleção* da Turma, os desenvolvedores não foram selecionados de modo aleatória, levou-se em consideração o perfil técnico e o psicológico.

Validade externa: Os cuidados nesta ameaça foram tomados em questões relacionadas aos resultados do estudo que não podem ser generalizáveis a projetos reais na indústria. Foram selecionados participantes que possuem relação ou refletem o comportamento da população de desenvolvedores. O evangelista era funcionário da Microsoft. Por mais que o treinamento tenha sido realizado no ambiente acadêmico, este cenário é o utilizado pelas organizações em MSECO, o que caracteriza um ambiente real, com a mesma infraestrutura, mesmas ferramentas e um evangelista.

Validade de construção: Problemas relacionados à ameaça de generalizar os resultados do estudo à teoria que o sustenta. Quanto aos fatores humanos, os participantes não estavam envolvidos em outros experimentos durante a realização deste estudo. Os participantes ainda participaram de forma espontânea como forma de adquirir conhecimento no desenvolvimento de aplicações móveis e os resultados não seriam utilizados para avaliação durante o curso de graduação.

Validade de conclusão: para reduzir a ameaça de generalização dos dados foram selecionados participantes que representassem a população de desenvolvedores em ecossistemas. É conhecido que a amostra não é ampla mas a quantidade de 15 participantes por treinamento é utilizada em eventos oficiais de organizações que detém MSECOs.

Os resultados mostram um indicio entre aspectos humanos junto com a equipe de desenvolvimento de software dentro de um ecossistema de software. Os resultados mostram indícios que o perfil psicológico dos participantes pode afetar o ecossistema. Por exemplo, determinados participantes conseguiram concluir suas tarefas e outros não.

7. Conclusão e Trabalhos Futuros

Os resultados mostram traços de um indicio entre o perfil psicológico e a produtividade de desenvolvedores numa plataforma de software móvel. Portanto, é interessante que a equipe de desenvolvimento reflita sobre o estudo do tipo de personalidade que sua equipe possui, além de quais ferramentas deverão ser utilizadas.

Como perspectivas de trabalhos futuros, pretende-se aplicar o estudo novamente com outras abordagens, porém, num ambiente verdadeiro de desenvolvimento (na indústria) e depois fazer um estudo detalhado e comparativo das abordagens aplicadas e qual teve uma a maior eficiência. Além de verificar quais os perfis psicológicos têm

mais desempenho dentro de uma equipe de desenvolvimento. Outro estudo está relacionado a qualidade das aplicações móveis desenvolvidas por cada tipo de personalidade.

Referências

Barbosa, J. and Silva, S. (2010). "Interação Humano - Computador". 1ra ed. Elsevier, Rio de Janeiro, p. 385.

Branco, D., Oliveira, E., Galvão, L., Prikladnicki, R. and Conte, T. (2015) "An Empirical Study about the Influence of Project Manager Personality in Software Project Effort". In: 17th International Conference on Enterprise Information Systems (ICEIS), p. 102-113.

Branco, D., Prikladnicki, R. and Conte, T. (2012) "Um estudo preliminar sobre Tipos de Personalidade em Equipes Scrum". In: 15th Ibero-American Conference on Software Engineering (CIbSE), p.304-311.

Belbin, Meredith R., 2010. "Team Roles at Work". Elsevier Butterworth-Heinemann Ltd.

Bentley, P. and Lim, S. (2012) "How to be a Successful App Developer: Lessons from the Simulation of an App Ecosystem". ACM SIGEVOLution, v. 6, n. 1, p. 2-15.

Berger, T., Pfeiffer, R., Tartler, R., Dienst, S., Wasowski, Andrezj. and She, S. (2014) "Variability mechanisms in software ecosystems. Information and Software Technology", v. 56, n. 11, p. 1520–1535.

Bosch, J. (2009) "From Software Product Lines to Software Ecosystems". In: 13th International Software Product Line Conference (SPLC), p. 111-119.

Cruz, S., Da Silva, F., Monteiro, C., Santos, P. and Rossilei, I. (2011) "Personality in Software Engineering: Preliminar Findings From a Systematic Literature Review". In: Proceedings of 15th Annual Conference on Evaluation & Assessment in Software Engineering EASE, p. 1-10.

Fagerholm, F. e Münch, J. (2012) "Developer experience: concept and definition". In: Proceedings of the International Conference on Software and System Process, p. 73-77.

Faquin, G., Falcin, M. e Araujo, M. (2016) "Uma Metodologia de Avaliação da Relação entre Perfis de Personalidade e Desempenho Acadêmico em Alunos de Sistemas de Informação". In: XII Brazilian Symposium on Information Systems, p. 285-292.

Fontão, A., Bonifácio, B., Dias-Neto, A., Bezerra, A., Santos, R. (2014) "MSECO Skill : Construção de Competências de Desenvolvedores em Ecossistemas de Software Móvel". In: 17th Ibero-American Conference on Software Engineering (CIbSE), p. 81–94.

Fontão, A., Santos, R. e Dias-Neto, A. (2015a) "Mobile Software Ecosystem (MSECO): a systematic mapping study". In: Computer Software and Applications Conference (COMPSAC), p. 653-658.

Fontão, A., Santos, R. e Dias-Neto, A. (2015b) "MSECO-SUP: Support Process in Mobile Software Ecosystems". In: 29th Brazilian Symposium on Software Engineering (SBES). p. 31-40.

Fontão, A. e Dias-Neto, A. (2016) "GoDev-DX: Governança de Desenvolvedores em Ecossistemas de Software Móvel a partir da Experiência do Desenvolvedor (DX)". Third Latin-American School on Software Engineering.

Jain, A. (2011) "Apps marketplaces and the telecom value chain". IEEE Wireless Communications, v. 18, n. 4, p. 4–5.

Lin, F. e Ye, W. (2009) "Operating System Battle in the Ecosystem of Smartphone Industry". International Symposium on Information Engineering and Electronic Commerce (IEEEC), Ternopil, pp. 617-621.

Jung, C. (1991) Tipos Psicológicos. Rio de Janeiro: Vozes.

Keirsey, D. (1998) “Please Undestand Me II”. Prometheus Nemesis Book Company.

Manikas, K., Hansen, M. (2013) Software ecosystems – A systematic literature review, Journal of Systems and Software, vol. 86, no.5, pp.1294-1306.

Miao, Z. Yuan, B. (2005) “Discussion on Pervasive Computing Paradigm”. TENCON IEEE Region 10 Conference, Melbourne, p. 1–6.

Miranda, M., Ferreira, R., Souza, C., Filho, F., Singer, L. (2014) “An exploratory study of the adoption of mobile development platforms by software engineers”. In: 1st International Conference on Mobile Software Engineering and Systems (MOBILESoft), p. 50–53.

Myers, I; Briggs, K. Myers-Briggs Indicator.

Paixão, C., Fortaleza, L. e Conte, T. (2013) “Desafios no Ensino de Computação: um estudo da relação entre perfil psicológico de alunos e evasão”. In: XXI Workshop sobre Educação em Informática, p. 720-729.

Paixão, C., Fortaleza, L. e Conte, T. (2012). “Um estudo preliminar sobre as implicações de tipos de personalidade no ensino de computação”. In: XX Workshop sobre Educação em Informática (WEI).

Schultz, M. (2003) “Metodologias Para Ensino de Lógica de Programação de Computadores”. Monografia de Especialização Ciência da Computação. Universidade Federal de Santa (UFSC), Florianópolis, SC, Brasil. p. 69.

Taylor, R. (2013) “The role of architectural styles in successful software ecosystems”. Institute for Software Research, University of California, Irvine, Irvine, CA 92697-3455, United States. p. 2–4.

Yang, D., Liu, W., Cui, Q., Yang, Y. e Wang, Q. (2011) “Modeling the number of active software users”. In: International Symposium on Empirical Software Engineering and Measurement (ESEM), p. 376–379.

Wang, Y. e Li, F. (2009) “How does project manager’ personality matters? Building the linkage between project managers’ personality and the success of software development projects”. In: 24th ACM SIGPLAN conference companion on Object Oriented Programming Systems Language and Applications. ACM Press.

Uma Reflexão sobre as Dimensões do Ambiente de Aprendizagem em Organizações de Software

José Jorge Lima Dias Júnior

Departamento de Ciências Exatas – Universidade Federal da Paraíba (UFPB)
Rio Tinto – PB – Brasil

jorge@dce.ufpb.br

Abstract. *Continuous learning of software development professionals is a requirement for organizations that want to remain competitive. In this way, we can perceive a software organization as a environment to foster learning processes. Thus, this paper presents a multi-dimensional theoretical model about learning environment in software organizations in order to facilitate the understanding of the subject and generate reflections for future research.*

Resumo. *O aprendizado contínuo dos profissionais de desenvolvimento de software é uma exigência para organizações que pretendem se manter competitivas. Desta forma, podemos perceber uma organização de software como um ambiente que deve fomentar os processos de aprendizagem. Assim, este artigo apresenta um modelo teórico multidimensional sobre o ambiente de aprendizagem em organizações de software a fim de facilitar a compreensão do tema e gerar reflexões para futuras pesquisas.*

1. Introdução

Desenvolvimento de software é um contexto que envolve organizações baseadas em conhecimento onde se destaca a importância da capacidade dos seus membros aprenderem o mais rápido e o mais eficientemente possível, favorecendo a melhoria da qualidade dos processos e dos produtos de software [Menolli et al. 2013]. Nesta perspectiva, reconhece-se a importância da organização promover um ambiente propício para que ocorra a aprendizagem em diferentes níveis (individual, de grupo e organizacional).

Geralmente a organização e a gestão de pessoas consideram os tipos formais de aprendizagem, principalmente através de treinamento e desenvolvimento. Contudo, a aprendizagem pode ser estendida a outros tipos menos estruturados como a aprendizagem informal e incidental que acontece no dia a dia do trabalho [Marsick e Watkins 2001]. A natureza da participação do indivíduo na aprendizagem no trabalho depende tanto do ambiente que oferece oportunidades para essa participação quanto da escolha dos indivíduos em aproveitar essas oportunidades para aprender [Billlett 2004].

Desta forma, Organizações de Software de Aprendizagem (*Learning Software Organization*) [Schneider 2009] precisam criar uma cultura que promova aprendizagem contínua e que adotem a troca de experiência entre os indivíduos e equipes. Isto requer uma abordagem interdisciplinar que integre as ideias da Engenharia de Software, Ciência Organizacional e Ciência Cognitiva [Feldman e Althoff 2001].

Apesar da importância do tema, são poucos os estudos que explicitamente

mencionam teorias da aprendizagem no contexto de Engenharia de Software, principalmente relacionado ao nível individual [Alagarsamy et al. 2006; Menolli et al. 2013] e também relacionado à aprendizagem informal [Reatto e Godoy 2015].

A aprendizagem nas organizações é um fenômeno complexo, afetado por fatores contextuais que impactam a forma como os indivíduos aprendem [Antonacopoulou 2006]. Neste sentido, a partir da perspectiva de que organizações de software são ambientes de aprendizagem, este *position paper* visa promover reflexões sobre a multidimensionalidade deste ambiente para que sirva de ponto de partida para futuras pesquisas. Assim, o restante do artigo está organizado da seguinte forma: a Seção 2 discute a área de Aprendizagem no Trabalho; a Seção 3 apresenta o modelo multidimensional do ambiente de aprendizagem em organizações de software; e a Seção 4 apresenta algumas considerações finais e trabalhos futuros.

2. Aprendizagem no Trabalho

Geralmente quando se pensa em aprendizagem no contexto do trabalho (*Workplace Learning*), lembramos das ações formais de treinamento. No entanto, este artigo está considerando outras situações de aprendizagem que muitas vezes não são percebidas no contexto organizacional, relacionadas à aprendizagem informal, cuja a ocorrência não é determinada ou desenhada pela organização e se dá em função dos interesses dos indivíduos [Abbad e Borges-Andrade 2004].

A literatura internacional sobre aprendizagem informal tem origem na área de educação de adultos. Já a literatura nacional se apoia à área de treinamento e desenvolvimento de recursos humanos [Reatto e Godoy 2015]. Portanto, no Brasil ainda há escassez de pesquisas acerca da aprendizagem informal [Flach e Antonello 2010], particularmente em contextos específicos como o de desenvolvimento de software, por se tratar de um campo multidisciplinar.

Estudos na área de aprendizagem no trabalho enfatizam que é de responsabilidade da organização criar um clima propício para a aprendizagem dos indivíduos e dos grupos [Senge 1990]. Este artigo se apoia nesta visão a partir do momento que percebe a organização de software como um ambiente de aprendizagem composto por diferentes dimensões que podem facilitar ou inibir a participação dos indivíduos nos processos de aprendizagem.

3. Dimensões do Ambiente de Aprendizagem em Organizações de Software

Este artigo apresenta uma visão multidimensional do ambiente de aprendizagem como um esforço para tentar compreender a complexidade envolvida no tema. O ponto de partida para o delineamento deste modelo é a Teoria Social Cognitiva, proposta por Bandura (1986), a qual considera que o comportamento do indivíduo é originado da interação entre fatores pessoais, cognitivos e ambientais.

Desta forma, considerando que a aprendizagem se dá através dessa interação entre o indivíduo e o contexto, poderíamos organizar o ambiente de aprendizagem em duas grandes dimensões: a dimensão pessoal, envolvendo as características idiossincráticas do indivíduo, incluindo fatores cognitivos e comportamentais; e a dimensão contextual que envolve um conjunto de particularidades do contexto que influencia e é influenciado por esse indivíduo. No entanto, a literatura mostra que o contexto envolve ainda diferentes fatores, como por exemplo, as relações sociais, a

liderança, características da organização, práticas organizacionais, tecnologias etc. Com o objetivo de compreender melhor a complexidade diante desta pluralidade de fatores, decidiu-se organizar a dimensão contextual em outras quatro dimensões: física, organizacional, social e tecnológica. Estas dimensões estão ilustradas na Figura 1.

A definição destas dimensões não objetiva chegar a um modelo definitivo ou exaurir as possibilidades de outras taxonomias, ou ainda de restringir o olhar para o fenômeno, mas apenas de promover um ponto de partida para a reflexão sobre o ambiente de aprendizagem em organizações de software. Desta forma, o modelo pode e deve ser evoluído, principalmente através de pesquisas empíricas. É importante destacar também que estas dimensões, apesar de estarem separadas na figura, possuem interrelações e interdependências. Desta forma, cada dimensão pode estabelecer relações unilaterais, bilaterais ou integradas com as outras dimensões.

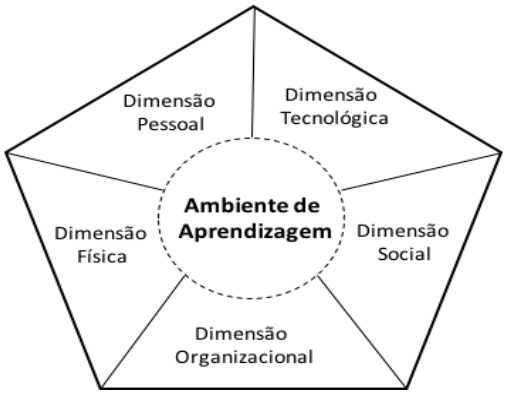

Figura 1. Dimensões do Ambiente de Aprendizagem em Organizações de Software

3.1. Dimensão Organizacional

De acordo com Sambrook (2006), aspectos relacionados às características da organização influenciam na aprendizagem, tais como cultura, tarefas, número de pessoas e estrutura organizacional. Desta forma, no modelo aqui proposto, a dimensão organizacional engloba estes e outros fatores, tais como nível de padronização dos processos e práticas de gestão (de projeto e de equipes). Algumas organizações se diferem no nível de padronização de seus processos, o que pode afetar os eventos que promovem aprendizado dos engenheiros de software. Por exemplo, uma organização pode ter seus processos bem definidos e estruturados; outras podem possuir processos de software que fomentem a troca de experiências e de conhecimento. É o caso de organizações, por exemplo, que adotam as Metodologias Ágeis.

3.2. Dimensão Física

A aprendizagem pode ocorrer em diversos espaços físicos desde a sala de trabalho e salas de reunião, até um almoço ou café durante o intervalo. A dimensão física refere-se ao espaço real por meio do qual a aprendizagem acontece [Merriam e Brockett 2007]. Desta forma, em um ambiente de aprendizagem, a estrutura física inclui todos os espaços, equipamentos e ferramentas dentro e fora da organização que dão suporte a eventos de aprendizagem no trabalho.

3.3. Dimensão Social

A dimensão social engloba as relações sociais dentro da organização. Cunningham e

Hillier (2013) identificaram que a aprendizagem informal acontece quando se desenvolve uma rede de contatos de aprendizagem e se compartilha conhecimentos e descobertas com essa rede através de relacionamentos positivos com colegas no ambiente de trabalho.

A dimensão social também inclui o processo de liderança. O líder deve ser capaz de mobilizar os membros da equipe para colaborarem e se engajarem na resolução de problemas [Faraj e Sambamurthy 2006]. Neste sentido, a liderança tem um papel importante para criar este ambiente facilitador da aprendizagem para que os profissionais se sintam competentes e empoderados para aprender.

3.4. Dimensão Tecnológica

Berg (2008) identificou que o acesso a tecnologia é um dos fatores mais importantes nas atividades de aprendizagem informal em organizações, principalmente porque pode aproximar as pessoas, facilitando as interações e o acesso a informações. Estas tecnologias permitiram a expansão para além do ambiente físico, desde a utilização de softwares que apoiam a gestão do conhecimento e os processos de software até ferramentas de comunicação que permitem a interação social entre as pessoas.

Apesar dos benefícios latentes que são percebidos na adoção de novas tecnologias para agregar valor as organizações, elas também podem trazer consequências negativas. Seppänen et al. (2015), por exemplo, identificaram que as tecnologias podem reduzir o controle percebido pelos indivíduos no trabalho, impactando negativamente em suas percepções de autonomia. Portanto, é preciso considerar a dimensão tecnológica, uma vez que esta tem relações que podem mediar a relação entre o indivíduo e o ambiente.

3.5. Dimensão Pessoal

Mesmo que as organizaçãos ofereçam condições físicas, tecnológicas, sociais e organizacionais, a aprendizagem depende da interação recíproca entre o indivíduo e esse ambiente. De acordo com a Teoria Social Cognitiva [Bandura 1986], os fatores pessoais e cognitivos estão constante interação com o ambiente de aprendizagem, e por isso não pode ser desconsiderado.

Consideramos como fatores pessoais as variáveis associadas às características idiossincráticas do indivíduo, tais como: personalidade, valores, estilos de aprendizagem, estados e estilos psicológicos.

4. Considerações Finais e Trabalhos Futuros

O ambiente de desenvolvimento de software pode ser visto como uma extensão do ambiente formal de educação para os profissionais da área. Para potencializar uma cultura voltada a aprendizagem, os profissionais precisam estar inseridos em um ambiente propício que fomente a aprendizagem em seus diferentes níveis.

Este *position paper* apresentou uma discussão que permite perceber organizações de software como ambientes de aprendizagem de forma multidimensional. Futuros trabalhos poderão se apropriar deste modelo teórico inicial para a realização de pesquisas empíricas, tanto qualitativo quanto quantitativo, para que sejam aprofundadas as relações entre as dimensões propostas.

Referências

Abbad, G., & Borges-Andrade, J. E. (2004). Aprendizagem humana em organizações de trabalho. *Psicologia, organizações e trabalho no Brasil*, 237-275.

Alagarsamy, K., Justus, S., & Iyakutti, K. (2006). A theoretical perspective on knowledge based organizational learning. In *2006 13th Asia Pacific Software Engineering Conference (APSEC'06)* (pp. 393-400). IEEE.

Antonacopoulou, E. P. (2006). The relationship between individual and organizational learning: New evidence from managerial learning practices. *Management learning*, *37*(4), 455-473.

Bandura, A. (1986). Social foundations of thought and action: A social cognitive theory. Prentice-Hall, Inc.

Berg, S. A., & Chyung, S. Y. (2008). Factors that influence informal learning in the workplace. *Journal of workplace learning*, *20*(4), 229-244.

Billett, S. (2004). "Learning through work: Workplace participatory practices". In H. Rainbird, A. Fuller, & A. Munro (Eds.), Workplace learning in context (pp. 109–125). London: Routledge.

Cunningham, J., & Hillier, E. (2013). Informal learning in the workplace: key activities and processes. *Education+ Training*, *55*(1), 37-51.

Faraj, S., & Sambamurthy, V. (2006). Leadership of information systems development projects. *IEEE TRANSACTIONS ON ENGINEERING MANAGEMENT EM*, *53*(2), 238.

Feldmann, R. L. & Althoff, K. D. (2001). On the status of learning software organizations in the year 2001. In *International Workshop on Learning Software Organizations* (pp. 2-6). Springer Berlin Heidelberg.

Flach, L., & Antonello, C. S. (2010). A teoria sobre aprendizagem informal e suas implicações nas organizações. *GESTÃO. Org-Revista Eletrônica de Gestão Organizacional*, *8*(2).

Marsick, V. J., & Watkins, K. E. (2001). Informal and incidental learning. *New directions for adult and continuing education*, *2001*(89), 25-34.

Menolli, A., Reinehr, S., & Malucelli, A. (2013). Organizational learning applied to software engineering: a systematic review. *International Journal of Software Engineering and Knowledge Engineering*, *23*(08), 1153-1175.

Merriam, S. B., & Brockett, R. G. (2007). The profession and practice of adult education: An introduction. John Wiley & Sons.

Reatto, D., & Godoy, A. S. (2015). A Produção sobre Aprendizagem Informal nas Organizações no Brasil: Mapeando o Terreno e Rastreando Possibilidades Futuras. *Revista Eletrônica de Administração*, *21*(1), 57-88.

Sambrook, S. (2006). Developing a model of factors influencing work-related learning: Findings from two research projects. In *Work-related learning* (pp. 95-125). Springer Netherlands.

Schneider, K. (2009). Experience and knowledge management in software engineering. *Springer Science & Business Media*.

Senge, P. (1990). The fifth discipline: The art and science of the learning organization. New York: Currency Doubleday.

Seppänen, M., Pajarre, E., & Kuparinen, P. (2015). The effects of performance-monitoring technology on privacy and job autonomy. *International Journal of Business Information Systems*, 20(2), 139-156.

Análise de Ferramentas para Controle de Versões de Software no Contexto do MPS.BR

Danne da Silva Oliveira[1], Heitor Costa[2], Paulo Afonso Parreira Júnior[2]

[1] Instituto de Ciências Exatas - Universidade Federal de Goiás – Regional Jataí – Caixa Postal 03 – 75801-615 – Jataí-GO – Brasil

[2] Departamento de Ciência da Computação - Universidade Federal de Lavras Caixa Postal 3.037 – 37.200-000 – Lavras – MG – Brasil

{danneoliveirasoft}@gmail.com,{heitor, pauloa.junior}@dcc.ufla.br

Abstract. *Configuration Management (CM) is one of MPS.BR processes, which deals with the management software versions. Software version control along its development cycle is not a simple task and the use of computational tools can impact positively or negatively on this control. Therefore, it is necessary to undertake an analysis of support tools for software version control in the literature to find out whether they are appropriate or not. This work consisted in the analysis of the software version control tools "Subversion", "Git", "Bazaar" and "Mercurial", in order to check if they include the requirements specified by the CM process in MPS.BR model.*

1. Introdução

MPS.BR (Melhoria de Processos do Software Brasileiro) é um modelo de maturidade de processos que tem como objetivo a melhoria de processos de desenvolvimento de software e serviços de organizações desenvolvedoras. O MPS.BR se divide em quatro modelos de referência, sendo que o enfoque deste trabalho está sobre o Modelo de Referência MPS para Software (MR-MPS-SW), que se baseia nas definições dos níveis de maturidade que uma organização pode assumir, bem como dos processos de software que ela deve contemplar. O MR-MPS-SW define sete níveis de maturidade, que estabelecem patamares de evolução de processos de desenvolvimento de software sequenciais e acumulativos. Esses níveis de maturidade vão do G (Parcialmente Gerenciado) ao A (Em Otimização). Para cada um desses sete níveis de maturidade é atribuído um perfil de processos, que apresenta os requisitos sobre os quais a organização deve colocar esforços para melhoria de seu processo de desenvolvimento.

Dentre os processos existentes no MPS.BR, destaca-se o processo de "Gerência de Configuração (GCO)", que pertence ao nível F na evolução dos níveis de maturidade do MPS.BR. Tal processo tem o propósito de estabelecer e manter a integridade de todos os produtos de trabalho de um processo ou projeto e disponibilizá-los a todos os envolvidos. Segundo Sommerville (2011), a GCO trata do gerenciamento das versões de um software. Em outras palavras, a GCO engloba as atividades de identificação e controle das mudanças, garantia de efetividade da implementação das mudanças e divulgação das mudanças aos interessados (Pressman, 2001). Segundo Brito e Yoshidome (2006), o controle das versões de um software ao longo de seu ciclo de desenvolvimento não é uma tarefa simples e o uso de ferramentas computacionais pode impactar positiva ou negativamente neste controle. Sendo assim, faz-se necessário

realizar uma análise das ferramentas de apoio à GCO existentes na literatura, em particular, ao controle de versões do software, para verificar se as mesmas são adequadas ou não. Uma forma de se fazer isso é descobrir se elas contemplam os requisitos exigidos pelos processos dos modelos de maturidade de processo, como por exemplo, o MPS.BR.

O objetivo principal deste trabalho é apresentar uma análise comparativa das principais ferramentas de apoio ao controle de versões existentes na literatura, tais como *SubVersion*, *Git*, *Bazaar* e *Mercurial*, com o intuito de descobrir se elas atendem ou não aos requisitos do processo de Gerência de Configuração do modelo MPS.BR. Este trabalho possui enfoque sobre o modelo de maturidade MPS.BR, pois ele é um modelo nacional voltado às empresas de micro e pequeno porte. Além disso, uma certificação MPS.BR custa em torno de quatro vezes menos do que a certificação para modelos internacionais, como o *Capability Maturity Model Integration* (CMMI).

Na Seção 2 são apresentados os principais trabalhos relacionados. Na Seção 3 estão os critérios utilizados para analisar as ferramentas para controle de versões, assim como os resultados da análise de tais ferramentas. Na Seção 4, estão as considerações finais e as propostas para trabalhos futuros.

2. Trabalhos Relacionados

O trabalho de Furlaneto (2006) apresenta o desenvolvimento de uma ferramenta de Gerência de Configuração, com base no estudo de outras ferramentas similares como *SubVersion* e *Trac* e pesquisas realizadas sobre o MPS.BR. Segundo Furlaneto (2006), a ferramenta desenvolvida se mostrou bem aderente aos resultados previstos no modelo MPS.BR. Para chegar a esta conclusão, ele avaliou as funcionalidades de sua ferramenta, com base nos resultados esperados do processo de GCO do MPS.BR. Entretanto, as principais diferenças entre o trabalho proposto por Furlaneto (2006) e a proposta deste projeto de monografia são: (i) Furlaneto avalia apenas a ferramenta desenvolvida por ele, com base nos resultados esperados do MR-MPS-SW. Este trabalho propõe a avaliação de diversas ferramentas para GCO existentes na literatura; e (ii) além disso, não ficou claro como os resultados esperados foram aplicados para a avaliação da ferramenta proposta por ele.

Já o trabalho desenvolvido por Oliveira (2007) apresentou uma lista de requisitos que podem guiar a escolha de ferramentas para Gerência de Configuração. Para gerar tal lista, Oliveira se utiliza de livros sobre Engenharia de Software (Sommerville, 2011; Pressman, 2001), do Corpo de Conhecimento em Engenharia de Software (SWEBOK) e de algumas normas IEEE. As principais diferenças do trabalho de Oliveira (2007) com relação a esta proposta são: (i) não utiliza o Guia de Referência MPS para Software (MR-MPS-SW), do MPS.BR; e (ii) apesar de Oliveira apresentar um conjunto de critérios para guiar a escolha das boas ferramentas para GCO, ele não avalia nenhuma ferramenta existentes na literatura, ficando assim, a cargo do profissional realizar esta análise para descobrir qual a ferramenta mais adequada. Nesta proposta, além de apresentar os critérios para comparação das ferramentas, será realizada uma análise sobre as principais ferramentas para GCO existentes na literatura.

O presente trabalho tem como intuito a criação de um mapeamento entre os resultados esperados do processo de GCO do MR-MPS-SW e os critérios a serem utilizados para comparação das ferramentas de apoio à GCO.

3. Análise das ferramentas para controle de versões

Por questão de espaço, as Tabelas 1 a 6 apresentam um resumo dos critérios confeccionados a partir dos resultados esperados do processo de GCO do modelo MPS.BR para avaliação de ferramentas de controle de versões de software. Mais detalhes sobre esses critérios podem ser encontradas em Oliveira (2014). A primeira coluna destas tabelas contém o código de cada critério e a segunda coluna apresenta uma descrição de cada critério.

Tabela 1. Critérios Relacionados ao Resultado Esperado GCO1.

GCO 1 – Um sistema de Gerência de Configuração é estabelecido e mantido	
Código	**Breve Descrição**
C.1.1	A ferramenta de Controle de Versão permite armazenamento e recuperação dos itens de configuração em suas diversas versões de forma a preservar e atualizar seu conteúdo.
C.1.2	A ferramenta de Controle de Versão gerencia os ramos (*branches*) das versões de um software.
C.1.3	A ferramenta de Controle de Versão gera relatórios gerenciais que possibilitem fazer um balanço da configuração existente.
C.1.4	A ferramenta de Controle de Versão mantém uma estrutura de pastas com controle de acesso e de manuseio.
C.1.5	A ferramenta de Controle de Versão permite a realização de procedimentos de preservação dos dados (backup).

Tabela 2. Critérios Relacionados ao Resultado Esperado GCO2.

GCO 2 – Os itens de configuração são identificados com base em critérios estabelecidos	
Código	**Breve Descrição**
C.2.1	A ferramenta de Controle de Versão, no caso dos itens de configuração ser muito grande, ela limita as possibilidades de gerência.

Tabela 3. Critérios Relacionados ao Resultado Esperado GCO3.

GCO 3 – Os itens de configuração sujeitos a um controle formal são colocados sob *baseline*	
Código	**Breve Descrição**
C.3.1	A ferramenta de Controle de Versão evita modificações que seja feitas sem a devida aprovação.

Tabela 4. Critérios Relacionados ao Resultado Esperado GCO4.

GCO 4 – A situação dos itens de configuração e das *baselines* é registrada ao longo do tempo e disponibilizada	
Código	**Breve Descrição**
C.4.1	A ferramenta de Controle de Versão aplica registro de inclusão.
C.4.2	A ferramenta de Controle de Versão aplica registro de alteração de itens no repositório.
C.4.3	A ferramenta de Controle de Versão identifica as diferenças entre duas versões de um mesmo item de configuração.
C.4.4	A ferramenta de Controle de Versão utiliza o mecanismo de ramos (*branches*) em seus controles de versões.
C.4.5	A ferramenta de Controle de Versão no nível gerencial é possível visualizar precisamente o andamento das modificações realizadas.

Tabela 5. Critérios Relacionados ao Resultado Esperado GCO5.

GCO 5 – Modificações em itens de configuração são controladas	
Código	**Breve Descrição**
C.5.1	A ferramenta de Controle de Versão analisa o impacto das modificações e notifica e descreve quais itens de configuração serão afetados pela modificação.
C.5.2	A ferramenta de Controle de Versão atribui solicitações aos responsáveis pelas mudanças.
C.5.3	A ferramenta de Controle de Versão aplica retirada (*check-out*) nos itens de configuração.
C.5.4	A ferramenta de Controle de Versão aplica registro (*check-in*) nos itens de configuração.

Tabela 6. Critérios Relacionados ao Resultado Esperado GCO6.

GCO 6 – O armazenamento, o manuseio e a liberação de itens de configuração e *baselines* são controlados	
Código	**Breve Descrição**
C.6.1	A ferramenta de Controle de Versão estabelece conexão à internet para estabelecer canais de segurança.

Para aplicação dos critérios elaborados, as seguintes ferramentas para controle de versões de software foram consideradas: 1. Bazaar Explorer (Versão 1.2.2); 2. Git

(Versão 1.11.1); 3. Mercurial (Versão 1.34.1.42.1); e 4. SubVersion (Versão 1.32.1.42.1). Tais ferramentas foram escolhidas por estarem dentre as mais utilizadas no mercado, segundo recente relatório de tecnologias e ferramentas Java, produzido pelo laboratório *Rebellabs*, da empresa *Zeroturnaround* (2016). A Tabela 7 apresenta os resultados das avaliações realizadas, exibindo a porcentagem de critérios contemplados por cada ferramenta. Os dados desta tabela foram obtidos de acordo com as avaliações feitas por três especialistas em ferramentas para controle de versão de software.

Na primeira e na segunda colunas desta tabela é possível visualizar o código do critério, bem como a qual resultado esperado do processo de GCO ele se refere; das colunas 3 à 6, estão os resultados para cada ferramenta analisada, com base na seguinte legenda: (X) a ferramenta não atende ao critério em análise; (O) a ferramenta atende parcialmente ao critério em análise; e (√) a ferramenta atende totalmente ao critério em análise. Da sétima à nona coluna, apresenta-se a porcentagem de ferramentas analisadas que atendem totalmente, parcialmente e não atendem aos critérios analisados. As últimas três linhas da Tabela 7 apresentam a porcentagem de critérios atendidos totalmente, parcialmente e não atendidos para cada ferramenta analisada.

Tabela 7. Resultado da Avaliação das Ferramentas.

Resultado Esperado	Critérios	Ferramentas				Total (%)		
		Bazaar	Git	Mercurial	SubVersion	X	O	√
GCO 1	C.1.1	√	√	√	√	0,0	0,0	100,0
	C.1.2	√	√	√	√	0,0	0,0	100,0
	C.1.3	X	X	X	X	100,0	0,0	0,0
	C.1.4	X	O	X	X	75,0	25,0	0,0
	C.1.5	X	X	X	X	100,0	0,0	0,0
GCO 2	C.2.1	X	X	X	X	100,0	0,0	0,0
GCO 3	C.3.1	√	√	√	√	0,0	0,0	100,0
GCO 4	C.4.1	√	√	√	√	0,0	0,0	100,0
	C.4.2	√	√	√	√	0,0	0,0	100,0
	C.4.3	√	√	√	√	0,0	0,0	100,0
	C.4.4	√	√	√	√	0,0	0,0	100,0
	C.4.5	O	√	O	√	0,0	50,0	50,0
GCO 5	C.5.1	X	X	X	X	100,0	0,0	0,0
	C.5.2	√	√	√	√	0,0	0,0	100,0
	C.5.3	√	√	√	√	0,0	0,0	100,0
	C.5.4	√	√	√	√	0,0	0,0	100,0
GCO 6	C.6.1	X	√	X	O	50,0	25,0	25,0
Subtotal X (%)		35,29	23,52	35,29	29,41			
Subtotal O (%)		5,88	5,88	5,88	5,88			
Subtotal √ (%)		58,83	70,60	58,83	64,71			

Legenda: X = Não atende; O = Atende Parcialmente; √ = Atende Totalmente.

Para gerar os resultados da Tabela 7, levou-se em consideração o seguinte procedimento: caso dois avaliadores escolhessem uma das opções X, O ou √, esta opção seria transcrita na Tabela 7; no caso de empate, a opção O foi transcrita para a tabela. Como pode ser visto nesta tabela, nenhuma das ferramentas atende aos critérios C.1.3, C.1.5, C.2.1 e C.5.1. Esse é um ponto crítico a ser considerado, uma vez que o não atendimento deste critério pelas ferramentas de controle de versões pode levar as empresas que utilizam estas ferramentas a terem resultado indesejados, quanto à avaliação de seu processo de GCO.

As ferramentas de controle de versões analisadas não atendem ao critério C.1.3, uma vez que elas não possuem modo de apresentação de relatórios específicos sobre as versões do software. As ferramentas apenas abordam *logs* e histórico das versões; com isto, há certa dificuldade de se fazer um balanço da configuração existente do software. Quanto ao critério C.1.5, não foi possível observar nas ferramentas analisadas, mecanismos de *backup* dos arquivos do software. Foi notado ainda uma limitação das

ferramentas ao lidarem com o versionamento de itens de configuração grandes (critério C.2.1). Por fim, quanto ao critério C.5.1, notou-se que as ferramentas analisadas não permitem analisar o impacto das modificações a serem realizadas no software e não há notificações para o usuário quanto aos itens de configuração que serão afetados pela modificação. Por fim, como pode ser observado na Tabela 7, a ferramenta que atende a maior quantidade de critérios (70,6%) é a ferramenta *Git*; já a ferramenta *SubVersion* é a segunda melhor ferramenta, atendendo à 64,71% dos critérios. Já as ferramentas *Bazaar* e *Mercurial* atendem a 58,83% dos critérios. Sendo assim, a escolha entre *Git* e *SubVersion* por parte dos usuários, ficará restrita às características específicas de cada uma, que cada usuário julga atender melhor às suas necessidades. Em outras palavras, considerando o processo de GCO definido pelo MPS.BR, bem como seus resultados esperados, o usuário encontrará um nível de adequação próximo à 100% em qualquer uma destas ferramentas.

4. Considerações Finais

Neste trabalho, realizou-se a análise de quatro ferramentas para controle de versões de software, com base em alguns resultados esperados do processo de GCO, proposto no modelo MPS.BR. Essa análise foi realizada com o auxílio de três especialistas em ferramentas de controle de versões. Espera-se que, com essa análise, pesquisadores e profissionais possam escolher de forma mais adequada a ferramenta que mais atenda à sua necessidade. Como trabalhos futuros pretende-se: (i) verificar a possibilidade de avaliar outras ferramentas para controle de versão para determinar se os resultados esperados pelo MPS.BR são atendidos; (ii) criar novos critérios para a possibilidade de abranger os processos de Manutenção, Implantação, Auditoria entre outros processos do MPS.BR; e (iii) avaliar também a usabilidade das ferramentas para controle de versão, a fim de saber se elas facilitam o trabalho de seus usuários.

Referências

BRITO, C. J. A.; YOSHIDOME E. Uma Análise Avaliativa de Ferramentas de Software Livre no Contexto da Implementação do Processo de Gerência de Requisitos do MPS.BR. WER. 2006. Cuenca, Equador.

FURLANETO, R. Ferramenta de Apoio a Gerencia de Configuração de Software. Monografia de Especialização. Universidade Regional de Blumenau. Blumenau, 2006.

OLIVEIRA, N. P. V. Requisitos de Ferramentas de Gerenciamento de Configuração. UFMG. Belo Horizonte, 2007. Disponível em: http://homepages.dcc.ufmg.br/~rodolfo/dcc823-2-07/Entrega4/Viviane4.pdf. Acesso: Mar/2015.

OLIVEIRA, D. S. Análise de Ferramentas para Controle de Versões de Software no Contexto do Processo de Gerência da Configuração do MPS.BR. Monografia de graduação. UFG/Regional Jataí. 2014.

PRESSMAN, Roger S. Software Engineering - A practitioner's Approach. 5ª Ed., McGraw-Hill, 2001, p. 253.

SOFTEX - Sociedade para Promoção da Excelência do Software Brasileiro. Guia Geral MPS de Software, dezembro 2012. Disponível em: http://www.softex.br/wp-content/uploads/2013/07/MPS.BR_Guia_Geral_Software_2012.pdf. Acesso: Mar/2015.

SOMMERVILLE, I. Engenharia de Software. 9. Ed. São Paulo: Pearson Prentice Hall, 2011.

ZEROTURNAROUND. Java Tools and Technology Landscape Report. Disponível em: http:// zeroturnaround.com/rebellabs/java-tools-and-technologies-landscape-2016. Acesso: Ago/2016.

Uma Abordagem para a Implementação da Gerência do Fornecedor usando Metodologias Ágeis

Elisiane Monteiro Soares[1], Sandro Ronaldo Bezerra Oliveira[1], Melquizedeque Cabral dos Santos[2], Alexandre Marcos Lins de Vasconcelos[2]

[1]Programa de Pós-Graduação em Ciência da Computação (PPGCC) – Instituto de Ciências Exatas e Naturais (ICEN) – Universidade Federal do Pará (UFPA), Rua Augusto Corrêa, 1 – Guamá – 66075-110 – Belém – PA – Brasil

[2]Centro de Informática – Universidade Federal de Pernambuco (UFPE), Caixa Postal 7851 – 50740-560 – Recife – PE – Brasil

elismclean@gmail.com, srbo@ufpa.br, {mcs6, amlv}@cin.ufpe.br

Abstract. *This paper proposes an approach about best practices using agile methods to manage the software acquisition process in the context of software development companies, based on the Supplier Agreeement Management process area in CMMI-DEV and the Acquisition process in MR-MPS-SW.*

Resumo. *Esse artigo propõe uma abordagem de boas práticas utilizando métodos ágeis para gerenciar o procedimento de aquisição no contexto de empresas de desenvolvimento de software, baseado na área de processo Supplier Agreeement Management do CMMI-DEV e no processo de Aquisição do MR-MPS-SW.*

1. Introdução

Desenvolver software de qualidade, dentro do escopo, tempo, custo e com os riscos controlados é um dos grandes desafios das organizações. No intuito de mitigar esses desafios, as empresas de desenvolvimento de software estão buscando novas estratégias como a subcontratação, aquisição de partes dos produtos e/ou serviços, aos clientes na tentativa de atender o mercado dinâmico e melhorar a qualidade do processo e do produto final [SEI, 2010; Sökmen, 2009]. Várias experiências de sucesso no uso da estratégia de subcontratação têm sido publicadas [Calvo-Manzano *et al*., 2007; Weber *et al*., 2007]. No entanto, as organizações continuam com problemas na gestão do processo de aquisição de produtos ou serviços junto aos fornecedores, pois, existe a dificuldade em manter o controle sobre o processo de desenvolvimento desses fornecedores, problemas no gerenciamento da qualidade dos produtos e serviços adquiridos, entre outros.

Visando resolver os problemas mencionados, as empresas estão adotando cada vez mais modelos de qualidade e metodologias de desenvolvimento de software com foco na melhoria do processo de software e na qualidade do produto. Por exemplo, o CMMI-DEV e o MR-MPS-SW que possuem, respectivamente, entre outras, a área de processo SAM – *Supplier Agreement Management* e o processo de AQU - Aquisição, cujo objetivo é gerir a aquisição de produtos de fornecedores através de um contrato formal [SEI, 2010; SOFTEX, 2016]. Metodologias ágeis também têm sido utilizadas nesse sentido, devido os benefícios de seu uso no desenvolvimento de software [Petersen e Wohlin, 2009].

Assim, este estudo tem como objetivo investigar o atual estado sobre as abordagens ágeis que apoiam o gerenciamento de acordos com fornecedores na aquisição de partes do produto e/ou serviços em projetos de desenvolvimento de software no contexto do CMMI-DEV e MR-MPS-SW. Assim, pretende-se fornecer uma abordagem que contenha as metodologias ágeis e suas práticas. Inicialmente, analisou-se as atividades propostas no *framework* de Furtado (2011), baseado em modelos e normas de melhoria de processo, mas

agora observando os princípios e as práticas ágeis de metodologias como Scrum e XP. Este resultado foi avaliado por especialista da área de Qualidade de Software, para verificar a adequação das práticas ágeis com relação ao procedimento de aquisição e utilizado durante a implementação de um programa de melhoria do processo usando o CMMI-DEV. O artigo está organizado da seguinte maneira: a Seção 2 apresenta a abordagem ágil para a gestão do fornecedor; na Seção 3 são apresentados, de forma sucinta, os resultados obtidos com a avaliação da abordagem; e a Seção 4 apresenta as considerações finais do trabalho.

2. A Abordagem para a Implementação da Gestão do Fornecedor usando Princípios e Práticas das Metodologias Ágeis

A abordagem proposta neste trabalho foi guiada pelas atividades constantes em um fluxo com foco na implementação do processo de gestão do fornecedor. As atividades foram identificadas a partir do alinhamento das práticas do *framework* de processos de aquisição de software e serviços correlatos [Furtado, 2011] e os processos de aquisição do CMMI-DEV e MR-MPS-SW. A pesquisa identificou as melhores práticas para o processo de aquisição de software, a fim de apoiar as empresas que desenvolvem contratos de aquisição com base em práticas ágeis. O fluxo é composto por (4) quatro atividades: (1) Preparação da Aquisição; (2) Seleção do Fornecedor; (3) Monitoramento da Aquisição; e (4) Aceitação do Produto Adquirido. E estas são subdivididas em tarefas.

2.1. A Abordagem de Gestão Ágil do Fornecedor

A abordagem tem como base os resultados obtidos através do mapeamento das 4 atividades com práticas ágeis. Foi constatada a compatibilidade de uso para um total de 25 tarefas com concepção de procedimentos para aquisição de forma ágil, mostradas a seguir.

2.1.1. Atividade Preparação da Aquisição

Nesta atividade têm-se as tarefas de planejamento da aquisição, descritas a seguir, com as orientações de implementação usando princípios e práticas ágeis. Para **Estabelecer as Necessidades e Resultados Pretendidos** deve ser realizada uma reunião para discussão das necessidades e dos resultados pretendidos com a aquisição, e será elaborado um *Backlog* do Produto, ficará exposto em quadro, com a lista de funcionalidades requeridas. Em **Definir os Interessados no Projeto**, será realizada uma reunião para definição dos *stakeholders*. Durante a reunião serão verificados os pontos: Quem pode afetar o projeto; Responsabilidade no projeto; Como cada interessado pode ajudar (Fator positivo) e atrapalhar (Fator Negativo) na aquisição; e os graus de influência de cada interessado. Tais informações farão parte do Quadro de Definição dos *Stakeholders*. Na tarefa **Definir e Priorizar os Requisitos dos Interessados** será definida a ordem de prioridade dos itens do *Backlog* do Produto, elaborado na tarefa "Estabelecer as Necessidades e Resultados Pretendidos". Serão adicionadas ao *Backlog* as colunas "Valor da Prioridade" e "Grau de Prioridade", respectivamente, para indentificar a prioridade de acordo com seu grau de importância, e para definir o grau de maior representatividade dentro de um projeto.

Em **Revisar os Requisitos** os itens do *Backlog* são revisados para verificar a conformidade com as necessidades dos interessados. O detalhamento deve ser feito em um cartão, composto pela identificação única (Id), nome e descrição do item. A identificação será usada no cartão "inconsistência" representando o item revisado, com a descrição e a alteração requerida para corrigir o problema. Em **Realizar Análise da Viabilidade Técnica**, deve ser realizada uma reunião para verificação de viabilidade técnica e elaboração do Quadro de Viabilidade. Para tal, serão verificadas questões que considerem os recursos disponíveis, restrições existentes, a qualidade e o prazo necessários para o projeto, e um Parecer deve ser emitido para cada questão. Para **Desenvolver uma**

Estratégia de Aquisição, será realizada reunião para elaboração da estratégia de aquisição com base nos resultados obtidos na tarefa anterior, pois fornecem os subsídios para o planejamento de uma estratégia compatível com as necessidades da aquisição, considerando as opções viáveis, riscos e custos de cada opção. Em **Definir e Acordar o Cronograma da Aquisição** ocorrerá reunião elaboração do Cronograma de atividades da Aquisição, exposto em Quadro de Trabalho, com a especificação das atividades, responsáveis pela execução e datas para sua realização. Cada atividade concluída receberá um *Checked* (√) para demonstrar sua execução.

Em **Definir os Critérios para Aceitação do Produto Adquirido**, uma reunião será realizada para tratar sobre os critérios de aceitação do produto. Assim, deverá ser confeccionada uma lista com as tarefas e os produtos de trabalho do projeto que devem ser entregues ao final deste projeto. Em **Definir os Critérios de Seleção do Fornecedor**, será realizada uma reunião de planejamento para a escolha dos critérios para seleção do fornecedor. Nesta reunião serão discutidos os critérios relevantes, tais como: orçamento, cronograma para elaboração do produto, experiência em projetos anteriores, entre outros. Um *Checklist* de Seleção do Fornecedor deve ser gerado. A tarefa **Elaborar e Aprovar o Plano da Aquisição** deve usar alguns artefatos gerados em outras atividades: Cronograma, *Checklist* de Seleção do Fornecedor e *Checklist* de Aceitação do Produto. Será elaborada ainda uma lista de produtos a serem fornecidos, e definidas as responsabilidades dos envolvidos no procedimento de aquisição. A partir dessas informações, o Plano de Aquisição é gerado. Na tarefa **Identificar Potenciais Fornecedores**, será feito um levantamento das informações dos fornecedores, e serão listadas em um Quadro. Cada característica terá uma identificação, e será usada como referência para mostrar quais os fornecedores que atendem as características requeridas.

2.1.2. Seleção do Fornecedor

Nesta atividade têm-se as tarefas de identificação e seleção do fornecedor que melhor atende as expectativas da aquisição, descritas a seguir usando princípios e práticas ágeis. Em **Receber Propostas**, as propostas dos fornecedores serão encaminhadas para o responsável pelo projeto de aquisição, o qual avaliará seus conteúdos, com a especificação dos critérios atendidos e não atendidos, já identificando os fornecedores com melhores propostas e expondo em um Quadro de Trabalho da Lista de Propostas dos Fornecedores. Na tarefa **Emitir Parecer Operacional**, será realizada uma reunião para análise da proposta confrontando as características do fornecedor e se sua solução proposta de fato funcionará. A emissão de parecer tomará por base a Estrutura de Avaliação Operacional PIECES, e considera Performance, Informação, Economia, Controle, Eficiência e Serviços. A tarefa **Emitir Parecer Técnico** ocorrerá junto à tarefa anterior e consiste na análise de questões relacionadas às características do fornecedor de disponibilidade dos recursos técnicos e dos profissionais necessários para a implementação da proposta.

A tarefa **Selecionar o Fornecedor** terá reunião para discutir as características dos fornecedores, já identificadas e expostas nos relatórios operacionais e técnicos. Com isso, será elaborado Quadro de Análise e Seleção do Fornecedor, com a especificação do fornecedor selecionado e a justificativa pela seleção e não seleção de cada fornecedor. Em **Preparar e Negociar um Contrato** haverá uma reunião para discussão sobre os termos do contrato a ser gerado, expectativas e obrigações dos envolvidos (adquirente e fornecedor), abordando questões tais como: custo, cronograma, itens passíveis de revisão e monitoramento a serem realizadas, etc. Na tarefa **Garantir o Bom Entendimento dos Termos Contratuais** serão verificados todos os termos contratuais, para que estes sejam compreendidos por todos os envolvidos na aquisição. E ao final deve ser gerado um

Checklist dos itens que irão compor o contrato e a justificativa para a inserção ou não dos itens do contrato. Na tarefa **Emitir o Contrato**, a partir da realização da tarefa anterior, as cláusulas que irão compor o contrato já foram definidas e acordadas, desta forma prepara-se o Contrato. Este será emitido em reunião que contará com a presença do adquirente e do fornecedor selecionado para a emissão e a assinatura do contrato a ser firmado.

2.1.3. Atividade Monitoramento da Aquisição

Nesta atividade têm-se as tarefas de controle e monitoramento do procedimento de aquisição, descritas a seguir, cuja implementação utiliza os princípios e as práticas ágeis. Em **Revisar os Termos Contratuais** terá reunião para verificação de todos os itens constantes do Contrato, onde serão analisadas inconsistências e relevâncias destes, e tomada de decisões sobre as alterações necessárias. Deve ser gerado Quadro com a Lista dos termos contratuais com *checklist* das possíveis alterações devidamente justificadas. Para **Acordar Alterações** ocorrerá a partir de uma reunião, para análise sobre as alterações identificadas na tarefa anterior. Assim, deve ser gerado um cartão de alterações requeridas e inserções no contrato, devidamente acordado entre as partes envolvidas.

Em **Acompanhar Problemas**, será feita reunião para discussão dos problemas encontrados e as soluções propostas para cada situação. Deve ser gerado Quadro de acompanhamento de problemas, com as fases *TO DO*, *DOING* e *DONE,* constituído pelas atividades do projeto e identificação de problemas, solução proposta, responsável por cada solução, data de início e data de conclusão das correções realizadas. Para **Monitorar os Processos do Fornecedor** será realizada uma reunião para verificação dos processos utilizados pelo fornecedor, para checar o atendimento das expectativas do cliente, e execução das atividades em conformidade com o contrato. Será gerado um *Checklist* de análise dos processos desempenhados pelo fornecedor, a partir de critérios para monitorar sua execução.

2.1.4. Atividade Aceitação do Produto Adquirido

Nesta atividade têm-se as tarefas de verificação da conformidade do produto para sua aceitação por parte do adquirente, e de forma ágil são detalhadas a seguir. A tarefa **Avaliar o Produto Entregue** consiste na avaliação do produto considerando o atendimento dos critérios de aceitação do produto. Terá a presença da equipe de aquisição e do fornecedor, a fim de que este último tenha o conhecimento das inconsistências e dos problemas encontrados no produto. A tarefa **Manter Conformidade com o Contrato** trata de uma reunião para análise detalhada do atendimento de cada item constante do contrato. Desta forma, deve ser gerado um Quadro com *checklist* de atendimento dos itens do contrato. Em **Aceitar Produto Entregue**, após a avaliação do produto e a verificação de conformidade com o contrato firmado entre as parte envolvidas, ocorrerá uma reunião para formalizar a aceitação do produto e a assinatura do Termo de Aceitação pelas partes envolvidas.

3. Avaliação da Abordagem de Gestão Ágil do Fornecedor

A avaliação foi feita inicialmente em relação à compatibilidade das atividades que compõem o *framework* de Processo de Aquisição de Software & Serviços Correlatos [Furtado, 2011] em relação às práticas e resultados esperados dos processos de aquisição do CMMI-DEV e MR-MPS-SW. Em seguida foi realizada a avaliação do mapeamento das práticas ágeis com relação às atividades oriundas do *framework* compatíveis com as práticas específicas e resultados esperados do processo aquisição. A Abordagem de Boas Práticas Ágeis para a Gestão do Fornecedor foi avaliada através de revisão por pares. Estas avaliações, foram enviadas a um especialista da área (Qualidade de Software e Melhorias

de Processos de Software) para obtenção de *feedback*, opiniões e sugestões de melhorias, que foram contempladas na versão final da abordagem.

Após isso, a abordagem de boas práticas foi usada para a implementação de um programa de melhoria do processo de software na Empresa Municipal de Informática da Prefeitura do Recife, a EMPREL, que obteve em Junho de 2016 a avaliação do Nível 3 de Maturidade do CMMI-DEV, encontrando-se em fase de Avaliação Complementar para o Nível C do MR-MPS-SW. Como *feedbacks* obtidos pela equipe usuária da abordagem pode-se obter: (i) Melhoria do Compromisso das partes interessadas no processo de aquisição; (ii) Gestão de versão e funcionalidade; (iii) Lista de Recomendações/Guias Ágeis; (iv) Maior Visibilidade da Gestão Ágil da Informação; (v) Planejamento e Controle Ágil; (vi) Avaliação dos Indicadores Ágeis; (vii) Escopo Ágil da aquisição, com o atendimento dos requisitos priorizado e definido pelo cliente na presença do fornecedor.

4. Conclusões

Este trabalho mostrou a importância do uso de melhoria de processos em um ambiente organizacional, principalmente do processo de Aquisição de Software, buscando conhecer e aplicar o que a literatura dispõe sobre essa temática no contexto de desenvolvimento de software. Para alcançar tal fim, foi realizado um mapeamento sistemático a fim de verificar em estudos primários, o que se tem sobre a área de pesquisa do artigo, para identificar as lacunas existentes e mostrar a relevância do estudo. Após essa fase foi realizada uma revisão da literatura e um mapeamento de um *framework* de processo de aquisição com práticas ágeis, para fornecer o embasamento necessário para a elaboração da proposta de uma Abordagem de Boas Práticas para a Gestão Ágil do Fornecedor.

Entre as limitações deste trabalho incluem: (i) um conjunto grande de práticas e princípios ágeis disponíveis na literatura e capazes de atender a implementação das tarefas propostas no fluxo discutido neste trabalho, favorecendo a atualização da abordagem apresentada neste trabalho; e (ii) a dificuldade em encontrar empresas com foco na aplicação da gerência do fornecedor a partir da implementação de programas de melhoria do processo de software.

Referências Bibliográficas

Calvo-Manzano, J., Cuevas, G., Garcia, I., San Feliu, T., Serrano, A., Arboledas, F., Ruiz de, F. (2007) “Requirements Management and Acquisition Management Experiences in Spanish Public Administrations”. IJ ITK, 1(2): 116- 121, Bulgaria.

Furtado, J. C. C. (2011) “Spider-ACQ: Uma Abordagem para a Sistematização do Processo de Aquisição de Produtos e Serviços com Base em Multi-modelos de Qualidade”. Dissertação de Mestrado, PPGCC/UFPA, Brasil.

Petersen, K., Wohlin, C. (2009) “A Comparison of Issues and Advantages in Agile and Incremental Development Between State of the Art and an Industrial Case”. Journal of Systems and Software, Volume 82, Issue 9, 1479-1490.

SEI (2010) “CMMI for Development (CMMI-DEV)”. Versão 1.3., Carnegie Mellon University, Pittsburgh, PA.

SOFTEX (2016) “Melhoria do Processo de Software Brasileiro (MPS.BR) - Guia Geral para Software:2016”. Brasil.

Sökmen, N. (2009) “Turkish Software Producing ICT Companies Approaches in Establishment of Their Subcontractor Selection and Management Processes”. PICMET 2009 Proceedings, Portland, USA.

Weber, K., Araújo, E., Scalet, D., Andrade, E., Rocha, A., Montoni, M. (2007) “MPS Model-Based Software Acquisition Process Improvement in Brazil”. Proc. of the Sixth QUATIC, IEEE Computer Society, pp. 110-119.

Levantamento de Papéis e Atores em um Ecossistema de Software no Domínio Público

Rebeca Teodoro da Silva[1,2], Luiz Gustavo Ferreira Aguiar[1,2], Rodrigo Pereira dos Santos[3], Elias Canhadas Genvigir[1]

[1]Universidade Tecnológica Federal do Paraná (UTFPR) – Cornélio Procópio – PR

[2]Tribunal de Justiça do Paraná (TJ PR)

[3]Universidade Federal do Estado do Rio de Janeiro (UNIRIO)

rebeca.teodoro@gmail.com, lgfaguiar@hotmail.com, rps@uniriotec.br, elias@utfpr.edu.br

Abstract. *Software Ecosystem (SECO) is an interaction of a group of players on a common technology platform, resulting in a number of software solutions or services. The analysis of the roles of the actors involved in a SECO is an approach that can be used for understanding their relationships. In this context, this paper presents a preliminary process that aids the analysis of the roles of the actors involved in the SECO based on their relationships and interactions with the central organization (keystone), more specifically in the public domain. We investigate a real SECO, Projudi System of the Court of Justice, and conducted some interviews as our first steps.*

Resumo. *Ecossistema de Software (ECOS) é uma interação de um conjunto de atores sobre uma plataforma tecnológica comum, que resulta em um número de soluções ou serviços de software. A análise de papéis dos atores envolvidos em um ECOS é uma abordagem que pode ser utilizada para a compreensão das relações envolvidas. Neste contexto, este artigo apresenta um processo preliminar para análise de papéis dos atores envolvidos em ECOSs baseado em seus relacionamentos e interações com a organização central, no domínio público. Uma investigação foi feita em um ECOS real, o Sistema Projudi do Tribunal de Justiça, e entrevistas foram conduzidas como um passo inicial.*

1. Introdução

A estratégia tradicional de desenvolver um produto de software único tem sido substituída pelo desenvolvimento de múltiplos produtos e funcionalidades, criados a partir de uma plataforma tecnológica comum [Santos et al. 2013]. Tal estratégia tem sido analisada por meio de ecossistemas de software (ECOS), que compreendem alguns elementos como um agente centralizador (*keystone*), uma plataforma (que pode ser uma tecnologia ou mercado) e os agentes do nicho relacionado [Manikas e Hansen 2013]. Nesse contexto, a análise de papéis dos atores envolvidos em um ECOS surge como uma abordagem para a compreensão das relações envolvidas [Lima et al. 2014].

O objetivo deste estudo é apresentar um processo preliminar para análise de papéis de atores envolvidos em ECOSs no domínio público. A elaboração do processo é baseada nos relacionamentos e interações com a organização central (*keystone*), para que possibilite uma visão geral dos papéis dos atores que diretamente afetam a saúde da

plataforma e permita uma melhor compreensão das relações envolvidas. Foi utilizado, como objeto deste estudo, o sistema de apoio às atividades do Poder Judiciário, denominado Sistema Projudi. No ECOS Projudi, existem diversos sistemas e atores que estão interligados e sofrem a ação de atores externos e elementos técnicos, transacionais e sociais que formam uma complexa rede de interações [Silva et al. 2015].

Este artigo está organizado da seguinte forma: a Seção 2 apresenta uma sumarização de papéis dos atores em ECOS; a Seção 3 apresenta o processo para análise de papeis em ECOS do domínio público; a Seção 4 descreve o caso do ECOS Projudi; e a Seção 5 conclui o artigo com algumas considerações finais.

2. Papéis dos Atores em ECOS

Um ECOS é uma interação de um conjunto de atores sobre uma plataforma tecnológica comum, que resulta em um número de soluções ou serviços de software que afetam os aspectos sociais de software. Cada ator é motivado por um conjunto de interesses ou modelos de negócio e está conectado aos demais e ao ECOS por relacionamentos [Manikas e Hansen 2013]. Um ator pode ser uma empresa ou outro tipo de organização, um setor de uma empresa, um usuário final do produto de software, um fornecedor ou um cliente e, de maneira geral, pode abranger quaisquer outros envolvidos ou interessados [Lima 2015]. Além disso, atores podem ter papéis específicos em um ECOS, como sumarizado na Tabela 1 [Lima 2015]. Observa-se também na Tabela 1, assim como em [Lima et al. 2014], que alguns termos foram mantidos no original em inglês a fim da manutenção do vocabulário comum utilizado na área.

Tabela 1. Descrição dos papéis dos atores em ECOS.
Adaptado de [Lima, 2015]

Papéis de atores gerais	*Papéis de atores específicos*	Descrição
Hub	***Keystone***	Representa a entidade de influência dominante.
	Dominator	Extrai valor do ECOS, colocando em risco a sua saúde e sustentabilidade.
Niche Player	***Customer***	Representa o cliente que gerou a necessidade dos produtos de software do ECOS.
	Competitor	Tenta extrair valor do ecossistema, porém não ameaça a saúde do ECOS.
	Supplier	Ator que fornece um ou mais produtos ou serviços necessários ao ECOS.
	Reseller	Revende um produto desenvolvido por outro ator sem alterá-lo.
	Independent Software Vendor (ISV)	Produz e vende seu próprio produto.
	Value-added Reseller (VAR)	Revende um produto desenvolvido por outro ator, mas agrega valor ao mesmo.
	Influencer	Desenvolve para o ECOS e contribui para sua saúde ao se comprometer com uma estratégia, complementando o *keystone*.
	Hedger	Desenvolve seus produtos ou serviços para apoiar múltiplas plataformas.
	Disciple	Compromete-se exclusivamente com a plataforma de um ECOS.
External Actor	***3rd-party developers***	Promove o ECOS e seus produtos, pode propor melhorias. Análogo ao *influencer*, porém externo ao ECOS, não tendo vínculo formal com o *keystone*.
	End-user	Usuário final do produto; difere do *customer* por não contratar serviço do *keystone*.
	External Partner	Contribui para o bem-estar do ECOS por meio de atitudes, tais como a promoção do ECOS e de seus produtos, propondo ainda melhorias.

3. Processo para Análise dos Papéis dos Atores

O processo proposto neste artigo pode ser utilizado para auxiliar a classificação dos papéis dos atores envolvidos em ECOS no domínio público. Assim, a questão de pesquisa é: “como identificar os papéis dos atores envolvidos em um ECOS no domínio público?”. Para responder a esta questão, é apresentado um processo para auxiliar na análise dos papéis dos atores envolvidos no ECOS.

A critério de organização, o processo de levantamento de papéis e atores em um ECOS no domínio público foi dividido em três etapas. A primeira é relacionada à coleta de dados, a segunda contempla a classificação dos relacionamentos e interações, e a terceira, e última, se refere à análise dos papéis dos atores.

A primeira etapa, a de coleta, foca na definição do escopo do ECOS que será alvo da análise. Existem ecossistemas que envolvem muitos sistemas e subsistemas e, no domínio público, pode haver ainda mais de um *keystone* presente em um mesmo ECOS. Por exemplo, no Judiciário, o ECOS Projudi pode ser analisado em um contexto delimitado fisicamente, um "estado da federação", como também "nacionalmente", onde pode haver outras instâncias do mesmo sistema com outros relacionamentos e interações em outros estados. Dessa forma, torna-se necessário realizar inicialmente a atividade de escolha do escopo para delimitação do ECOS.

Em seguida, são realizadas as atividades de entrevistas. Para isso, faz-se necessária a escolha dos entrevistados. Os entrevistados devem ser pessoas que estão em contato com a plataforma do ECOS e que tenham uma visão geral dos relacionamentos e interações envolvidas. As entrevistas podem ser conduzidas de forma sistemática ou informal, com o objetivo de elencar os relacionamentos e interações do sistema. Nesta etapa, as relações entre os produtos e serviços podem ser percebidas na forma de um *web service*, como também uma integração complexa com outro sistema. É importante que haja uma descrição e respostas para perguntas: "Por que existe este dependência?" e "Esta relação atende a quem?". Assim, o pesquisador lista as relações existentes entre os sistemas sobre a plataforma e mantém informações sobre cada uma delas.

Na segunda etapa, são ainda realizadas as atividades relacionadas à classificação do relacionamento baseado na abordagem SocialSECO [Lima et al. 2014] e a classificação dos relacionamentos/interações em "Ator->Ator" ou, considerando um artefato como ator do ECOS, até mesmo, "Artefato->Artefato", conforme [Seichter et al. 2010]. O pesquisador selecionará então os relacionamentos/interações que ocorrem com atores diversos no ECOS para classifica-los com o tipo "Ator->Ator" para análise.

Na terceira etapa, os relacionamentos/interações são analisados com os papéis dos atores apresentados na literatura [Lima 2015], tal como apresentado na Tabela 1. Dessa forma, cada relacionamento/interação analisado será comparado com cada uma das opções de papéis dos atores, para verificar se o tipo do relacionamento corresponde ao papel do ator envolvido. Após a análise das relações, poderá ser realizada uma nova entrevista para que sejam suscitadas dúvidas com relação às novas informações. Ao final do processo, tem-se a classificação dos papéis dos atores envolvidos no ECOS.

4. Exemplo de um Caso Real de ECOS: Projudi

O ECOS Projudi possui características peculiares com relação às dimensões sociais e técnicas de um ecossistema, que ainda não estão amplamente presentes na literatura. Observa-se que tais características se devem ao fato deste ECOS estar inserido no domínio público [Silva et al. 2016]. Por exemplo, a atuação de um ator dentro do ECOS de um domínio público pode ser determinado por uma lei. Neste caso, a dimensão social não é apoiada em comunidades, redes sociais ou *sites* e a dimensão técnica não é de livre escolha, mas baseada em acordos, sessões de software e políticas públicas.

Para a coleta dos dados, foram realizadas entrevistas em 2016 com funcionários específicos da área de TI, envolvendo a diretoria da área, que lideram cerca de 57 pessoas, além de técnicos judiciários. Os entrevistados têm contato com a plataforma do presente estudo, o Projudi. Além disso, também foram utilizados como apoio documentos internos para levantamento de quais eram os relacionamentos e interações existentes no sistema.

As entrevistas foram apresentadas em formato livre a fim de obter informações sobre os relacionamentos e as interações do sistema. Adicionalmente, uma descrição, em linguagem natural do relacionamento/interação foi realizada, além de verificar qual o ator (ou atores) estavam envolvidos. Os relacionamentos e interações listadas foram analisadas da seguinte forma: **primeiramente**, foi observado o tipo de interação, selecionando-se aquelas do tipo Ator->Ator. Isso se deve ao fato de que o tipo de interação "Ator->Ator" pressupõe interações entre atores [Seichter et al. 2010]; **em seguida**, foram observados os tipos dos atores de cada interação, ou seja, se o ator está ou não envolvido com o nicho do negócio do ECOS Projudi. Observa-se que uma interação "Ator->Ator" do mesmo nicho de negócio é mais relevante que a de nichos diferentes. Por exemplo, no ECOS Projudi existe uma interação específica entre o sistema Projudi e um sistema da Caixa Econômica Federal, de modo que o nicho de negócio é a área jurídica e a Caixa Econômica Federal não pertence a este nicho, mas sim ao nicho financeiro/bancário. No entanto, na falta desta interação, o ECOS continuaria em operação; **por último**, foi realizada uma análise das classificações dos papeis dos atores (Tabela 1) frente os relacionamentos e integrações coletados.

Após a análise do ECOS Projudi, foram obtidos 17 relacionamentos e interações. Observa-se a atuação de dois principais papéis de atores: *Niche Player-Customer* e *Niche Player-Influencer.* O papel do ator "*Niche Player-Customer*", nesse cenário, é apresentado na forma de uma interação com outro sistema que pode até mesmo participar do processo de desenvolvimento informando os requisitos. Por exemplo, uma integração com o sistema Oráculo para obter informações de antecedentes criminais. Embora o sistema Oráculo faça parte do ECOS Projudi, ele não está comprometido com a estratégia do ECOS. Em contrapartida, o papel "*Niche Player-Influencer*" está intrinsecamente envolvido com a saúde do ECOS, como, por exemplo, a Secretaria da Justiça e o Ministério Público.

Nota-se que há vários membros do ECOS que estão comprometidos com uma estratégia e que complementam a plataforma do ECOS. Assim, aponta-se que, pela quantidade de relacionamentos/interações de "*Niche Player-Influencer*" presentes no ECOS Projudi, eles também são responsáveis pela saúde no ECOS.

5. Considerações Finais

Devido à dificuldade de mapear os papéis dos atores que estão envolvidos em um ECOS no domínio público, este artigo apresentou um processo preliminar para análise de papéis dos atores envolvidos em ECOSs baseado em seus relacionamentos e interações com a organização central, no domínio público. Por meio de entrevistas conduzidas em um caso real de ECOS, o Projudi, foram elencados os relacionamentos e interações existentes no ecossistema, observadas as características dos atores envolvidos neste cenário e, posteriormente, definidos os seus papéis. Os resultados iniciais demonstram que, embora o ECOS Projudi possua várias interações/relacionamentos, os papéis dos

atores envolvidos concentram-se no "*Niche Player-Customer*" e "*Niche Player-Influencer*". Faz-se importante ressaltar que tal característica é oriunda do domínio público, tendo em vista a presença de vários atores fortemente relacionados com o nicho de negócio. Além disso, observa-se uma quantidade expressiva do papel "*Niche Player-Influencer*" no ECOS que, além de complementar a plataforma, também são responsáveis pela saúde do ECOS.

Durante a etapa de coleta, bem como nas entrevistas, uma dificuldade foi encontrar especialistas disponíveis para a atividade. Outra dificuldade foi a de elencar as características dos relacionamentos existentes com a plataforma central; por ser um ECOS que envolve vários atores, foram considerados apenas relacionamentos baseados em alguma interação técnica. Além disso, os resultados obtidos neste estudo não visam a generalização para ECOSs com outras características – as considerações foram realizadas para ECOS no domínio público. Como trabalho futuro, pretende-se evoluir este processo para análise de atores que não possuem interações técnicas com a plataforma do ECOS e ainda compreender o impacto que os diversos papéis dos atores trazem para o ECOS.

Agradecimento

O terceiro autor agradece ao CNPq (Proc. No. PDJ 150539/20016-9) pelo apoio financeiro.

Referências

Lima, T. M. P. (2015). Uma Abordagem Socio-técnica para Apoiar Modelagem e Análise de Ecossistemas de Software. *Projeto Final. Curso de Engenharia de Computação e Informação. UFRJ*, 83p.

Lima, T., Santos, R. P. Dos e Werner, C. (2014). Uma Abordagem Socio-técnica para Apoiar Ecossistemas de Software. *Revista Brasileira de Sistemas de Informação*, v. 7, n. 3, p. 19–37.

Manikas, K. e Hansen, K. M. (2013). Software Ecosystems - A Systematic Literature Review. *The Journal of Systems and Software*, v. 86, n. 5, p. 1294–1306.

Santos, R. P. Dos, Werner, C. M. L., Alves, C. F., et al. (2013). Ecossistemas de Software : Um Novo Espaço para a Construção de Redes e Territórios envolvendo Governo, Sociedade e a Web. *Políticas Públicas: Interações e Urbanidades*, v. 1ed, p. 337–366.

Seichter, D., Dhungana, D., Pleuss, A. e Hauptmann, B. (2010). Knowledge Management in Software Ecosystems: Software Artefacts as First-class Citizens. *Proc. of the 4th European Conference on Software Architecture Companion Volume (ECSA '10)*, Copenhagen, Denmark, p. 119–126.

Silva, R. T., Ferreira, L. G. e Genvigir, E. C. (2015). Ecossistema de Software no Contexto do Poder Judiciário - Apontamentos Sobre o Estado do Paraná. *Anais do VI CBSOFT, IX WDES,* Belo Horizonte, Brasil, v. 01, p. 49–56.

Silva, R. T., Ferreira, L. G. e Genvigir, E. C. (2016). Análise dos Relacionamentos em um Ecossistema de Software no Contexto Público. *11ª Conferencia Ibérica de sistemas y Tecnologías de Información CISTI,* Gran Canaria, España, p. 290–295.

Preliminary Findings of Expert and Systematic Reviews on the Software Ecosystems Research

Olavo Barbosa[1], Rodrigo Santos[2], Davi Viana[3]

[1]State Agency for Information Technology of Pernambuco
Av. Rio Capibaribe, 147, São José – CEP 50020-080 – Recife, Brazil

[2]DIA/CCET – Federal University of the State of Rio de Janeiro (UNIRIO)

[3]Computer Engineering/CCET– Federal University of Maranhão (UFMA)

olavo.barbosa@ati.pe.gov.br, rps@uniriotec.br, davi.viana@ufma.br

Abstract. *Software Ecosystems (SECO) are a set of organizations and actors, as well as their relations that cover technical, social and business aspects of software development. As a research field, several studies and reviews were conducted towards a body of knowledge for the SECO field. In this paper, we preliminarily analyze these studies in order to provide an initial overview of the SECO literature. Our intention is to aid researchers to know some relevant opportunities to foster the field's evolution, such as collaborative governance.*

1. Introduction

The growing importance and special characteristics of software systems increasingly make it an interesting topic of study for social sciences, law, management, government, business, and economy (Santos & Werner, 2011). One of the key characteristics of the software industry like others is that most organizations do not have all the resources to satisfy the needs of customers. In addition, software organizations face some challenges in coping with market and environmental factors that are usually out of control (i.e., economic constraints). As such, infrastructure acquired from many software vendors must work mutually. Frequently, there is a need to integrate the applications in complex ways across organization boundaries in order to create and support new products and satisfy customer needs. This set of organizations, resources, customers, and products have been known as Software Ecosystems (SECO). SECO represents an approach to analyze relationships among players of software industry in which organizations must engage in a perspective that considers both their own business and third parties.

As a research field, SECO analyzes the software industry as a networked industry. Concepts from biological (Moore, 1993) and business (Iansiti & Levien, 2004) ecosystems inspired the SECO field. However, Messerschmitt & Szyperski (2003) have started the scientific research on SECO with their book. It is the oldest reference in the Software Engineering area and has been supported by several citations found in primary and secondary studies already published in the literature. Serebrenik & Mens (2016) point out Microsoft as an example of a SECO in their meta-analysis for the SECO research. In turn, we have identified that many studies and reviews were conducted towards a body of knowledge for the SECO field. In this paper, we preliminarily analyze some of them in order to provide an initial overview of the SECO literature. As such, we included Systematic Mapping Studies (SMSs), Systematic Literature Reviews (SLR) and Personal Opinion Surveys (POS) on the SECO research in our analysis. Our

intention is to aid researchers to know relevant opportunities to foster the field's evolution, such as collaborative governance. This paper is organized as follows: in Section 2, we present related work; in Section 3, we explain the research approach; in Section 4, we discuss preliminary outcomes; and in Section 5, we conclude the paper.

2. Related Work

In the SECO research field, natural ecosystems' concepts are generally accepted. The business ecosystems' perspective is considered as well, since the SECO field inherits properties from both natural and business ecosystems. Santos & Werner (2011) worked on such concepts from 2009 to 2010 and organized them in three perspectives known as architecture, strategies/tactics, and social networks. In turn, Hanssen & Dybå (2012) provided theoretical foundations for the SECO field. They proposed a framework to guide and support future directions. Based on a SMS, Barbosa et al. (2013) concluded that SECO research is concentrated in eight main areas, and the most relevant are open source software, ecosystem modeling, and business issues. They analyze SECOs in three dimensions: technical, business, and social. Manikas & Hansen (2013) also addressed those dimensions in a SLR on SECO research and provide an overview of the field.

Jansen et al. (2013) summarize a collection of studies that cover several aspects, such as SECO definitions, business network management, and SECO visualization and analysis. Fotrousi et al. (2014) provide an overview of existing research on the SECO performance with an SMS. In a SLR study, Franco-Bedoya et al. (2014) explore part of the literature of open source software for identifying quality measures and provide a quality model for the quality assessment in SECO. Axelsson & Skoglund (2015) focus on quality assurance challenges that exist in traditional development practices and also carry over to ecosystems. To complement the collection of reviews, Manikas (2016) provides an updated study on the field in order to document its evolution over the past years. According to the study, literature in SECO has evidenced signs of maturity.

3. Research Approach

Inspired by the work of Wohlin et al. (2013) and by the procedures of Petersen et al. (2015), our research follows four steps (we are focused on the step 2 at this moment). **Step 1** aims to select secondary studies on the SECO research according to the following criteria: (1) SMS/SLR studies in SECO; (2) expert reviews in SECO using *ad hoc* literature selection that adopted some practices of SLR/SMS; and (3) POS studies in SECO. **Step 2** consists of applying the method proposed by Cruzes & Dybå (2011) regarding the levels of interpretation in thematic synthesis. The process of highlighting segments of the text of each study was started with the use of spreadsheets. **Step 3** examines the results in order to reduce the overlapping, and translates codes into themes. Furthermore, this step involves the process of grouping the initial codes into a smaller number of sets and interpretations to create a model with higher-order themes. **Step 4** aims to ensure the reliability of research outcomes by recognizing the limitations of study and how they affect the results. The main threat is the selection of studies and individual bias in the assessment of those studies. Although we have identified few secondary studies well known in the community based on related conferences, workshops and special issues, some works may have been left out of our study.

4. Initial Results and Discussion

For step 1, we selected nine studies that are SLR/SMS, expert reviews or POS in the SECO research field. After performing an initial analysis of this set of studies, we found some preliminary results that we describe bellow. The selected studies are in Table 1.

Table 1. Set of secondary studies on SECO field and its classifications.

Study	Authors	Research methods	Publication venues
S1	Hanssen & Dybå (2012)	Some practices of SMS/SLR	Conferences/workshops
S2	Barbosa et al. (2013)	SMS	Conferences/workshops
S3	Jansen et al. (2013)	Set of scientific studies	Scientific book
S4	Manikas & Hansen (2013)	SLR	Journal
S5	Franco-Bedoya et al. (2014)	SLR	Conferences/workshops
S6	Fotrousi et al (2014)	SMS	Conferences/workshops
S7	Axelsson & Skoglund (2015)	SMS	Journal
S8	Manikas (2016)	SMS	Journal
S9	Serebrenik & Mens (2016)	POS	Conferences/workshops

Outcomes in S3, S6, and S8 identify that the SECO field was inspired by several other fields. Some of them bring concepts of natural, business and digital ecosystems, and proposes new definitions for SECO. In spite of proposing a suitable overlapping and interrelation of the concepts found in the set of studies, we suggest that the social, technical and business perspectives discussed as the key dimensions for SECO (S1, S2, S3, and S6) can be seen as the three levels or themes of governance. As concluded in S8, there is an intersection between Information Technology governance and SECO. One relevant aspect is the decision-making regarding who makes decisions in a SECO (and how to do it) (S2, S3, S4, and S7). Regarding the support for the platform development, a challenge is to provide an actor of a SECO with information to make SECO management activities feasible in order to assist platform updates (S7). An approach to represent such concepts is to use definitions such as codes, themes, and high-order themes, as proposed by Cruzes & Dybå (2011). Codes can be defined as interesting concepts, categories, findings, and results of studies. A theme describes and organizes possible observations and/or interprets aspects of a phenomenon. Once themes are identified, they can be explored and interpreted to create a model consisting of higher-order themes and relationships among them. Preliminary themes presented in this study are represented in a mind-map, i.e., a tool used to do it, as recommended by Cruzes & Dybå (2011). As such, Figure 1 shows a visual illustration to help sorting the different codes into themes and finally in higher-order themes.

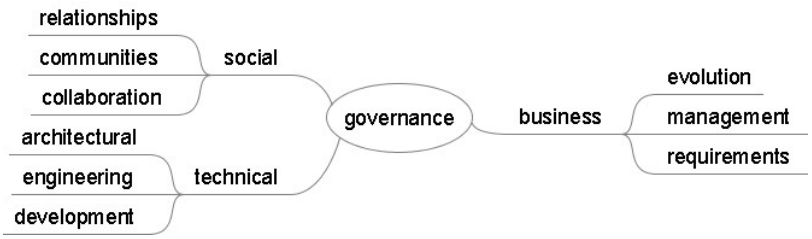

Figure 1. SECO concepts in the context of governance

In Figure 1, governance is seen as a higher-order theme that includes decision-making regarding what best strategies are needed for the actors' survival in any SECO, in any role, taking into account social, technical, and business perspectives as themes. Moreover, concepts of business such as evolution, management and requirements are

defined as codes in the context of SECO, as identify in S9, S2, and S8, respectively. These three themes are linked to SECO governance, as presented in Figure 1. There is an interrelation between each theme, even though they are represented in isolation. Once SECO comprises software development activities in which people take an important role, governance also covers social aspects (S1, S2, S7, and S9), relationships inside the communities, and levels of collaboration (S1, S2, S3, S6, S7, and S9). For each ecosystem participant, for example, there are many demands on how to be (and keep as) a leader, as well as relations with other perspectives, such as technical and business (and vice-versa). As identified in S2, S3, S6, S7, and S8, strategic decision-making in the SECO context can be reflected in the software (and organizational) structure by applying specific rules in the common technological infrastructure.

On the technical governance theme, there are codes such as architectural issues, as identified in S1, S2, S3, S4, and S8. They are subsets of the development process concerns in the context of SECO. In turn, S8 argues that a software architecture in SECO supports the ecosystem nature (i.e., be flexible to meet the needs of a specific SECO), management, and business model. Furthermore, patterns and architectural styles need to address other governance themes in order to create better and more reusable platforms. In turn, on the business theme, S1, S2, S8 and S9 identify that the ecosystem management can define, for example, how simple it is for new members to be engaged in a SECO. Moreover, the business model can attract new actors, while the way the software is produced and the common infrastructure's and products' architectures can influence actors' relationships and should reflect SECO management (S1, S2, and S3).

On the context of governance, for example, S9 identifies evolution as a key aspect for the success of a SECO. Once the evolution is related to the business model of a SECO regarding how close/open a SECO is in terms of its platform, procedures and processes are expected to emerge to determine the future evolution of the SECO. S5 discusses evolution as a sub-characteristic of ecosystem quality. Study S5 also defines quality as one dimension the software platform in which the ecosystem's projects are built upon; for example, the Android SECO provides the Android platform used by all the Android mobile apps. In spite of changes or evolution of the platform, how can external developers contribute to partially-closed and -controlled SECO's platforms, such as Apple/iPhone? In this regards, the business model can influence a community of organizations or developers that base their relations to each other on a common interest.

Regarding to the relations of actors on a common interest, a software developer assumes ownership or responsibility of part of the software (S2). Either an open or a closed platform that is defined by the business model may influence important social factors that affect an ecosystem, such as recognition from peers, sense of community, and sense of code ownership, as identified in S2. In tum, the business theme defines the SECO architecture that influences the social perspective, i.e., cooperation/knowledge sharing with multiple and independent entities (S2). On the technical level of governance, there is a need to link the process requirements (S7) with a proper management as part of the software development in the context of business governance. In S5, we identify this relation from business to evolution and requirements in open source SECOs. Such link supports the quality assurance as a way to prevent bad decisions and avoid problems, as well as allow verifying the compliance with the requirements and the business goals.

5. Conclusions and Future Work

In this paper, we preliminarily analyzed studies and reviews in the SECO field in order to provide an initial overview of related literature. We selected a set of nine studies and extracted codes into themes for providing some relevant opportunities to foster the field's evolution, such as collaborative governance. Once this study aimed to provide an initial synthesis by using high-order themes previously presented, we intend to use other tools such as thematic networks, tables, and tree-maps for a depth understanding of the relationships of governance as the central-topic of these themes in a future work. We hope that our preliminary outcomes aid researchers to know future directions according to what is already addressed in the studies in SECO, since SECO field has demonstrated signs of maturity and affect the treatment of economic and social aspects of software.

Acknowledgements

The second author thanks CNPq (Proc. No. PDJ 150539/20016-9) for financial support.

References

Axelsson, J., Skoglund, M. (2015) "Quality Assurance in Software Ecosystems: A Systematic Literature Mapping and Research Agenda". The Journal of Systems and Software 114(2016):69-81.

Barbosa, O. et al. (2013) "A Systematic Mapping Study on Software Ecosystems through a Three-dimensional Perspective". In: Jansen, S. et al. (eds.) Software Ecosystems: Analyzing and Managing Business Networks in the Software Industry, Edward Elgar Publishing, 59-81.

Cruzes, D.S., Dybå, T. (2011) "Recommended Steps for Thematic Synthesis in Software Engineering". 5th Intl. Symposium on Empirical Software Engineering and Measurement, Banff, Canada, 275-284.

Franco-Bedoya, O. et al. (2014) "QuESo: A Quality Model for Open Source Software Ecosystems". In: 9th International Conference on Software Engineering and Applications, Vienna, Austria, 209–221.

Fotrousi, F. et al. (2014) "KPIs for Software Ecosystems: A Systematic Mapping Study". In: 5th Intl. Conference on Software Business, Paphos, Cyprus, 194-211.

Hanssen, G.K., Dybå, T. (2012) "Theoretical Foundations of Software Ecosystems". In: 4th International Workshop on Software Ecosystems, Cambridge, USA, 6-17.

Jansen, S., Brinkkemper, S., Cusumano, M.A. (2013) "Software Ecosystems: Analyzing and Managing Business Networks in the Software Industry". Edward Elgar Publishing.

Iansiti, M., Levien, R. (2004) "Strategy as Ecology". Harvard Business Review 82(3):68-78

Manikas, K. (2016) "Revisiting Software Ecosystems Research: A Longitudinal Literature Study". The Journal of Systems and Software 117(2016):84-103.

Manikas, K. Hansen, K.M. (2013) "Software Ecosystems – A Systematic Literature Review". The Journal of Systems and Software 86(5):1294-1306.

Messerschmitt, D.G., Szyperski, C. (2003) "Software Ecosystems, Understanding an Indispensable Technology and Industry". Cambridge: The MIT Press.

Moore, J.F. (1993) "Predators and Prey: A New Ecology of Competition". Harvard Business Review 71(3):75-86.

Petersen, K., Vakkalanka, S., Kuzniarz, L. (2015) "Guidelines for Conducting Systematic Mapping Studies in Software Engineering: An Update". Information and Software Technology 64(2015):1-18.

Santos, R., Werner, C. (2011) "Treating Business Dimension in Software Ecosystems". In: 3rd ACM/IFIP Intl. Conference on Management of Emergent Digital EcoSystems, San Francisco, USA, 197-201.

Serebrenik, A., Mens, T. (2016) "Challenges in Software Ecosystems Research". In: 9th European Conference on Software Architecture Workshops, Dubrovnik/Cavtat, Croatia, Article No. 40.

Wohlin, C. et al. (2013) "On the Reliability of Mapping Studies in Software Engineering". The Journal of Systems and Software 86(10):2594-2610.

Softwares e Produção Cultural no Brasil

Jerônimo C. Pellegrini, Sérgio Amadeu da Silveira, Claudio Penteado, Daniel P. Astone, Murilo B. Machado, Paulo R. de Sousa, Rodolfo S. Avelino

[1]LabLivre – Universidade Federal do ABC (UFABC)
Alameda da Universidade, s/nº – São Bernardo do Campo – SP – Brazil

{jeronimo.pellegrini,sergio.amadeu,claudio.penteado}@ufabc.edu.br

Abstract. *We present a project developed by a university in partnership with Brazil's Ministery of Culture. The goal is to find innovative solutions to the development of free technologies by public administarion in order to contribute to policy manegement and cultural democratization, encouraging the development and improvement of free cultural software. The project has 4 axes of action: (1) uses and software appropriation by cultural operators; (2) studies of functioning of free software developers communities; (3) testing of relevant cultural software; and (4) proposal of a new management model for the encouragement of integrated development of free cultural software.*

Resumo. *Apresentamos resumidamente o projeto desenvolvido por uma Universidade, em parceria com o Ministério da Cultura. O objetivo é encontrar soluções inovadoras para o desenvolvimento de tecnologias livres pelo poder público que contribuam para a gestão de políticas públicas e para a democratização cultural, incentivando o desenvolvimento e a melhoria dos softwares culturais livres. O projeto articula-se em 4 eixos de atuação: (1) usos e apropriação de software pelos agentes culturais; (2) dinâmicas de funcionamento das comunidades de desenvolvedores de softwares livres; (3) testes dos principais softwares culturais; e (4) apresentação de um novo modelo de gestão para o incentivo de desenvolvimento integrado de softwares culturais livres.*

1. Apresentação

O projeto "Do Mapeamento do Uso de Softwares Culturais a uma Proposta de Novos Modos de Articulação entre Estado, Universidade e Sociedade: análise do desenvolvimento colaborativo de softwares culturais livres" é uma parceria de pesquisa entre a Universidade Federal do ABC (LabLivre: Laboratório de Desenvolvimento de Softwares Livres) e o Ministério da Cultura com o objetivo de realizar um estudo aprofundado sobre o uso de softwares culturais no Brasil e propor um novo modelo para o desenvolvimento de softwares culturais baseado no uso de softwares livres. Trata-se de um arranjo inovador que visa criar um canal de colaboração e cooperação para o desenvolvimento de novos modelos de gestão pública que contribuam para a democratização da produção artística e cultural, por meio do uso de softwares. O projeto direciona seus esforços para uma melhor compreensão dos aspectos humanos, sociais e econômicos do software através do estudo da interação entre agentes culturais e software, bem como das dinâmicas internas de diferentes comunidades de software livre, e ao propor soluções inovadoras em gestão pública do desenvolvimento de software (tanto governamental como direcionado aos agentes culturais).

2. Justificativa e Problematização

O rápido desenvolvimento das tecnologias de informação e comunicação (TIC) tem como resultado profundas transformações nas diferentes atividades humanas. No campo cultural, o uso de recursos tecnológicos, principalmente os softwares, se faz presente não somente no processo de criação e suporte, mas também nas diversas ações que envolvem o universo de produção e gestão de Arte e Cultura na sociedade contemporânea. Nesse contexto, Lev Manovich [Manovich 2013] denomina softwares culturais os programas que são indispensáveis para o nosso cotidiano cultural. Esta definição abrange não apenas o software usado diretamente na concepção e produção da arte, mas também todo software usado na cadeia de produção da Cultura e da Arte, incluindos os softwares usados na gestão de políticas públicas.

A consolidação de uma Sociedade em Rede [Castells 1999], criou um contexto histórico específico no qual as TICs, conectadas por redes de informação e comunicação, desempenham um papel central na reconfiguração da organização da produção, poder e a capacidade de produção de significados.

A dinâmica das redes de comunicação e informação depende do uso eficiente e criativo dos sofwares. No campo artístico e cultural, o uso das tecnologias (softwares) criam novas possibilidades de criação, ao mesmo tempo desenvolvem mencanismos de controle e manipulação, tornando o processo de produção cultural mais complexo [Benkler 2011]. O poder criativo do software oferece para os atores do campo artísitco e cultural infinitas possibilidades de inspiração e inventividade que ampliam a capacidade de produção cultural, oferecendo novos suportes, linguagens e possibilidades de criação artísitca. A difusão dos softwares, principalmente na Sociedade em Rede, proporciona que novos atores possam atuar dentro desse universo, tornando-o mais aberto e plural.

Por outro lado, os softwares também podem representar uma forma de controle e manipulação. Softwares proprietários, controlados por corporações financeiras, podem produzir situação de dependência, modulação e doutrinação. A criação de formatos e especificações limitam a capacidade de apropriação das tecnologias, que induzem os usuários a adotarem modelos e formatos pré estabelecidos, o que reduz e direciona a capacidade de criação e inovação, importantes fatores dentro do campo das artes e cultura.

Uma alternativa a "colonização dos softwares proprietários" é o software livre, que é aquele que garante ao usuário as quatro liberdades fundamentais definidas por Richard Stallman [Stallman 1995, Free Software Foundation 2016]: liberdade para executar o software; para estudar como o software funciona; para redistribuí-lo; e para modificá-lo e publicar as modificações. Estas liberdades permitem ao agente cultural o acesso ao software independente de condição socio-econômica; trazem a possibilidade de adaptação por parte do agente cultural, ou por outros; abrem a possibilidade de estudo do software, reconhecendo-o como possível objeto de trabalho cultural e artístico (e não apenas meio). Neste sentido, os softwares culturais livres estão avançando e contribuindo para a ampliação da nossa diversidade cultural e de nossa riqueza criativa. No contexto da gestão pública da cultura, é relevante também por outros motivos: o software livre é auditável; elimina a possibilidade de aprisionamento a formatos e padrões fechados; representa grande economia aos cofres públicos, e ajuda a levar para a administração pública a prática de compartilhamento. De acordo com Stallman [Stallman 2016], "Órgãos públicos existem para o povo, não para si mesmos. Quando fazem informática, eles a

fazem para as pessoas. Eles têm o dever de manter o controle total sobre essa informática para que possam garantir que essa seja feita corretamente para o povo. (Esta constitui a soberania de informática do Estado). Eles nunca devem permitir que o controle sobre a informática do Estado caia em mãos privadas. Para manter o controle da informática das pessoas, órgãos públicos não devem fazê-la com software privativo (software sob o controle de uma entidade que não o Estado)."

Grande parte destes softwares culturais livres são desenvolvidos por comunidades de desenvolvedores transnacionais com dinâmicas específicas e sua maioria possui colaboradores importantes residentes no país. Com qualidades bastantes distintas, os softwares culturais livres podem ser melhorados e adequados às necessidades dos criadores e produtores de cultura. Apesar de trabalharem de forma colaborativa, as comunidades de desenvolvedores de software livre muitas vezes ficam afastadas do universo de produção cultural. As necessidades dos artistas e agentes culturais são desconhecidas. Existe a necessidade de construir canais de comunicação entre esse público usuário e os desenvolvedores, de forma que seja possível o desenvolvimento e aprimoramento de softwares culturais livres que contribuam para a democratização da cultura e a liberdade no processo criativo. O fomento à cultura e a democratização da produção e acesso à Cultura são atribuições do Ministério da Cultura e vão de encontro as diretrizes da UNESCO. Nesse sentido, o MinC busca desenvolver políticas públicas voltadas a criar condições para que as barreiras socioeconômicas, legais e tecnológicas não limitem as potencialidades artísticas e expressões culturais (digitais, tradicionais, regionais e orais). O projeto se insere como um mecanismo para criação de um canal de comunicação entre os desenvolvedores de softwares culturais livres, a comunidade artística usuária de softwares e o Ministério da Cultura, por meio da realização de uma série de estudos com os agentes culturais e as comunidades de desenvolvedores, assim como da regulamentação legal que envolve as licenças dos softwares.

3. Objetivos

O projeto tem como objetivo encontrar soluções inovadoras para o desenvolvimento de tecnologias livres pelo poder público para o entretenimento, mobilidade, internet das coisas e redes distribuídas que contribuam para a gestão das políticas públicas e para a expansão da diversidade cultural e da criatividade, incentivando o desenvolvimento e a melhoria dos softwares culturais livres. São objetivos específicos do projeto:
a) Mapear os usos dos softwares culturais, proprietários e livres, utilizados pelos artistas, produtores, instituições e movimentos culturais em suas práticas cotidianas, bem como as implicações que os formatos proprietários trazem para a comunicabilidade e interoperabilidade dos acervos e bens artístico culturais;
b) Pesquisar demandas e propostas de apoio ao desenvolvimento colaborativo de software livre empregados nas atividades de criação, produção, distribuição e compartilhamento de bens culturais no Brasil;
c) Identificar modelos produtivos e metodologias de desenvolvimento de software capazes de garantir qualidade de software, usabilidade de interfaces, celeridade nas entregas de softwares funcionais e participação social no processo de produção de soluções livres para o poder público;
d) Promover editais de fomento ao aprimoramento de softwares culturais, conforme necessidades definidas por pesquisa exploratória, e em conjunto com o MinC;

e) Realizar seminários sobre os resultados, benefícios e desafios encontrados no desenvolvimento compartilhado de soluções livres;
f) Avaliar o impacto e resultados tecnológicos, sociais, econômicos e criativos dos editais na cadeia produtiva da cultura e do software, nas comunidades de software livre assim como os benefícios encontrados para o poder público nesse modelo produtivo.

4. Metodologia

A pesquisa articula-se em quatro eixos: usos e apropriação de software por agentes culturais; dinâmicas de funcionamento das comunidades de desenvolvedores de softwares livres; avaliação dos principais softwares culturais livres; e apresentação de um novo modelo de gestão para o incentivo de desenvolvimento de softwares culturais livres.

O primeiro eixo estuda os usos e formas de apropriação dos softwares culturais pelos diferentes atores que compõem o universo artísitco e cultural. Para a realização desse estudo está sendo realizado um levantamento colaborativo dos principais softwares utilizados por esses usuários e uma série de entrevistas com artistas, produtores culturais, professores e gestores sobre a importância dos software em suas práticas e os usos e limitações dos softwares utilizados, tendo como base a abordagem da Teoria Ator-Rede de Latour [Latour 1994, Latour 2001], conforme detalhado a seguir: a) Questionários online: por meio de um formulário online1 está sendo realizado um levantamento dos principais softwares utilizados pelos agentes culturais, identificando: área de atuação, forma de utilização de software, uso de software disponível na internet, aplicativos de smartfones e disponibilidade para entrevista. b) Entrevistas: as entrevistas estão estruturadas em dois blocos; inicialmente busca-se entender como o sofware está (ou não) presente no cotidiano do entrevistado e qual o significado que ele atribui a este; a partir dessas informações, inesperado na Teoria Ator-Rede de Latour, busca-se construir o Mapa de Actantes que envolve o agente cultural; na sequência são feitas perguntas direcionadas buscando identificar as funcionalidades procuradas no softwares, formas de apropriação e sua visão sobre os softwares livres.

O eixo de estudo das dinâmicas das comunidades de desenvolvedores de softwares livre tem por finalidade identificar os procedimentos de trabalho dessas diferentes comunidades de forma a identificar os rotinas de produção e assegurar a qualidade dos softwares desenvolvidos. Esse eixo está realizando um levantamento das comunidades existentes por meio de técnicas de rastreamento e identificação de redes, assim como a realização de entrevistas com os desenvolvedores de cada comunidade.

O terceiro eixo de estudo opera na realização de testes de funcionais dos principais softwares culturais livres. Por meio da comparação entre os softwares proprietários identificados no levantamento, a equipe técnica do projeto realiza testes de usabilidade e funcionalidade com os similares livres. Essa ação é importante para a identificação das limitações técnicas e para possibilitar propor medidas (demandas) para o desenvolvimento dos softwares culturais livres, tornado-os atrativos para o usuário.

Por fim, o quarto eixo de estudo trata do o acarbouço legal que envolve a administração pública e o licenciamento de softwares no Brasil. A ideia desse eixo é buscar alternativas para a produção de editais para o desenvolvimento de software que atendam as especificações dos agentes culturais e estejam de acordo com a dinâmica de trabalho das comunidades. Nesse sentido, além do estudo da legislação, as informações

dos outros eixos são essenciais para a realização dessa proposta.

5. Resultados Parciais

O projeto está em andamento e enfrenta dificuldades relacionadas a instabilidade política do país. Inicialmente a crise econômica e o cenário de incerteza política criaram dificuldades para a implantação da parceria e início das atividades. Posteriromente, o projeto também enfrentou duas situações críticas: a primeira foi a extinção do Ministério da Cultura pelo governo interino e depois, com o retorno do MinC, a exoneração do gestor responsável pela articulação entre o Ministério e a Universidade. Mesmo com esses problemas, há previsão de término para o ano de 2017, e já foi possível atingir as seguintes metas:
1. Criação e estruturação do laboratório: formação e treinamento da equipe de pesquisadores; definição dos procedimentos de pesquisa;
2. Levantamento dos softwares culturais: de questionários online, foi construído um banco de dados com mais de 350 entradas; além disso, foram realizadas 11 entrevistas (são previstas 50 entrevistas no total);
3. Realização do I Seminário de Software e Cultura: em agosto de 2016 foi realizado o evento que reuniu gestores da área de cultura, pesquisadores, desenvolvedores, artistas, ativistas e outros interessados. Foram discutidos temas como a importância do software na cultura, dificuldades do gestor de TI para o uso de software livre, funcionamento das comunidades de desenvolvedores e experiências com software livre;
4. Realização de contatos e entrevistas com as comunidades de desenvolvedores durante o Fórum Internacional de Software Livre em Porto Alegre;
5. Início dos testes de funcionalidade e usabilidade dos principais softwares culturais;
6. Composição de grupo de desenvolvedores para trabalhar na melhoria de softwares de gestão usados pelo MinC, e preparação para estudo dessas melhorias em fase posterior;
7. Levantamento do arcabouço jurídico do sistema de licenciamento de software no país.

Referências

Benkler, Y. (2011). Network theory— networks of power. *Journal of Communication*, 5(39).

Castells, M. (1999). *A era da informação: economia, sociedade e cultura*. Paz e Terra.

Free Software Foundation (2016). O que é o software livre. `https://www.gnu.org/philosophy/free-sw.pt-br.html`, acessado em Agosto de 2016.

Latour, B. (1994). *Jamais Fomos Modernos*. Editora 34.

Latour, B. (2001). *A esperança de Pandora*. Edusc.

Manovich, L. (2013). *Software takes command*. A&C Black.

Stallman, R. (1995). Why software should be free. In *D. G. Johnson & H. Nissenbaum (Eds.), Computers, ethics & social values*, pages 190–200. Prentice Hall.

Stallman, R. (2016). Software livre é ainda mais importante agora. `https://www.gnu.org/philosophy/free-software-even-more-important.html` acessado em Agosto de 2016.

Oportunidades de Pesquisa em um Ecossistema de Software de *E-learning*: ECOS SOLAR

Emanuel F. Coutinho[1], **Ítalo de Oliveira**[1], **Carla I. M. Bezerra**[2]

[1]Instituto Universidade Virtual (IUVI) – Fortaleza – CE
[2]Campus Quixadá – Quixadá – CE
Universidade Federal do Ceará (UFC) – CE – Brasil

{emanuel,italo}@virtual.ufc.br,carlailane@ufc.br

Abstract. *A software ecosystem (ECOS) refers to a set of software with a certain degree of symbiotic relationship, and may consist of actors interacting with a market supported by a technological platform or common market. Virtual Learning Environments (VLE) aim to create environments based on the Internet to enable the process of building knowledge and autonomy from their interactors. The SOLAR AVA is a virtual space for classroom courses and semipresential courses. The aim of this paper is to present some research opportunities in the e-learning SOLAR ECOS development and in software quality.*

Resumo. *Um Ecossistema de Software (ECOS) refere-se a um conjunto de produtos de software com determinado grau de relacionamento simbiótico, podendo consistir de atores interagindo com um mercado, apoiados por uma plataforma tecnológica. Ambientes Virtuais de Aprendizagem (AVA) visam a criação de ambientes na Internet que possibilitem a construção de conhecimento e autonomia de seus interagentes. O AVA SOLAR é um espaço virtual para cursos presenciais ou semi-presenciais. O objetivo deste trabalho é apresentar algumas oportunidades de pesquisa no desenvolvimento e qualidade de software do ECOS de e-learning SOLAR.*

1. Introdução

Um Ecossistema de Software (ECOS) refere-se a uma coleção de produtos de software com algum determinado grau de relacionamento simbiótico [Messerschmitt e Szyperski 2003]. O SOLAR (Sistema *Online* de Aprendizagem) é uma aplicação *web* cujo modelo de participação é orientado ao professor e ao aluno, possibilitando a publicação de cursos e interação com os mesmos. Nesse contexto, diversas complexidades são encaradas por educadores atualmente, e projetar um sistema de *e-learning* deve começar com a adoção de um modelo sustentável [Gütl e Chang 2008].

Para este fim, e para investigar aspectos sociais, humanos e econômicos do software, um modelo de ecossistema de *e-learning* foi proposto. O objetivo principal deste trabalho é apresentar alguns aspectos de desenvolvimento e qualidade de software do ECOS de *e-learning* SOLAR.

2. Oportunidades de Pesquisa e Desafios

O ECOS SOLAR é composto por um conjunto de elementos que se comunicam em diferentes níveis. O AVA SOLAR é a base do ecossitema em questão, sendo a plataforma

tecnológica que suporta o ECOS. Tais elementos envolvem diferentes instituições e diferentes perfis de usuários, produzindo ou recebendo informações, suportados por diferentes tecnologias, indicando a necessidade de gerenciar a qualidade entre todos os componentes envolvidos no ecossistema. Processos podem ser desenvolvidos independentemente, sendo necessário a garantia da qualidade nos produtos de software que estão sendo desenvolvidos, e aplicação de boas práticas no desenvolvimento.

A filosofia de desenvolvimento do SOLAR envolve muitos elementos de metodologias ágeis. Com a integração de diversas tecnologias e plataformas, a complexidade dos vários níveis de testes aumenta consideravelmente. Atualmente o ECOS SOLAR usa diferentes versões do AVA *web* e uma aplicação *mobile*. Sua manutenção implica diretamente na necessidade de testes bem estruturados, devido à adição de funcionalidades com fortes características de integração. A pesquisa no ECOS SOLAR como adição de funcionalidades no produto atualmente foca em acessibilidade. Um desafio é incorporar tais elementos (voz, transcrição de texto, usabilidade e *design*) na aplicação e com qualidade.

Outro ponto a ser abordado é como o ECOS SOLAR fornece suporte às ferramentas de software para os diversos perfis de usuário. Considerando como o ECOS faz essa mediação atualmente, é necessário um outro nível de gestão, engenharia e automação, tornando possível o desenvolvimento de novos produtos que são derivados de uma tecnologia já existente dentro do ECOS. Para isso, requisitos de software devem ser bem especificados para suportar as necessidades dos usuários/clientes, apoiar decisões arquiteturais e buscar utilizar FOSS (*Free and Open Source Software*).

Alguns desafios do ECOS SOLAR que vão além do desenvolvimento do software são: como disponibilizar dados do AVA sem ferir aspectos éticos e como garantir a segurança da infomação (escrita e leitura) no ambiente. Esses aspectos muitas vezes envolvem relações entre as instituições envolvidas. Apesar de ser um desafio relacionado à arquitetura e desenvolvimento de software, a integração do SOLAR com o sistema oficial de controle acadêmico da universidade é uma difícil tarefa pois envolve todas as restrições e complexidades de diferentes equipes de desenvolvimento, processos e tecnologias.

3. Conclusão

O SOLAR é um AVA que atinge uma grande variedade de perfis estando disponível atualmente em versões *web* e *mobile*. Diferentes requisitos surgem conforme interações ocorrem, requerendo um cuidado com processos de software e processos humanos. Além disso, como disponibilizar esses requisitos para os usuários de versões diferentes, alinhados a boas práticas de Engenharia de Software, e visando sempre a melhor qualidade possível, complementa esse desafio. Assim, espera-se que esta pesquisa possa contribuir com a literatura de ECOS, colaborando com a divulgação de ECOS brasileiros, e que possa fomentar a pesquisa no desenvolvimento de sistemas e na qualidade de software.

Referências

Gütl, C. e Chang, V. (2008). The use of web 2.0 technologies and services to support e-learning ecosystem to develop more effective learning environments. In *In proceedings of ICDEM 2008*, pages 145–148.

Messerschmitt, D. e Szyperski, C. (2003). *Software Ecosystem: Understanding an Indispensable Technology and Industry*. The MIT Press, 1 edition.

Uma Análise de Abordagens que Fornecem Suporte à Priorização de requisitos: reqT/CSP e G-4REPrioritization

Cinthya Cavalcanti[1], Maria Lencastre[1], José Júnior[1], João Pimentel[2], Tainã Santos[1], Timóteo Silva[1]

[1] Programa de Engenharia da Computação Universidade de Pernambuco, Pernambuco, Brasil
[2] Universidade Federal Rural de Pernambuco, Pernambuco, Brasil
{ccf, jmsj2, tms, mlpm, tgs}@ecomp.poli.br, joao.hcpimentel@ufrpe.br

Resumo. *Este artigo analisa duas abordagens de apoio ao processo de priorização de requisitos: a Linguagem de Domínio Específico reqT/CSP e o Guia G-4Reprioritization, com o objetivo de identificar se as duas abordagens se sobrepõem ou se são complementares com relação ao suporte à priorização de requisitos.*

1. Introdução

Não é raro projetos de software tecnicamente robustos serem malsucedidos devido à ausência de um tratamento dos fatores sociais, humanos e econômicos. Em geral, softwares que não atendem às necessidades dos seus usuários ou que ultrapassam o tempo e o custo estimados, tendem a enfrentar dificuldades no mercado. A priorização de requisitos se mostra como uma tarefa fundamental para mitigar e amenizar essa situação, uma vez que ela visa auxiliar na seleção dos requisitos mais importantes através da consideração de questões como: a importância dos requisitos, benefícios ao negócio, riscos envolvidos, custos, tempo de desenvolvimento [K. Pohl e C. Rupp, 2011]. Diversas técnicas para priorização de requisitos são amplamente aceitas e utilizadas; contudo, existe uma carência de mecanismos que auxiliem na tarefa de priorização e na definição de estratégias para o seu processo [B. Regnell e K. Kuchcinski, 2013] [J. Junior et al 2015].

Este artigo investiga dois trabalhos voltados para esse escopo, o G-4REPrioritization [J. Junior et al 2015] e o reqT/CSP [B. Regnell e K. Kuchcinski, 2013], como o objetivo de analisar se eles se sobrepõem ou se são complementares. O Guia de Priorização de Requisitos (G-4REPrioritization) é um roteiro que visa orientar engenheiros de requisitos com relação à escolha de uma técnica de priorização, de acordo com o perfil do projeto e o seu escopo. Por outro lado, o reqT/CSP (uma linguagem de domínio específico para modelagem de requisitos) inclui, além da especificação de requisitos, elementos e mecanismos que podem ser utilizados no contexto da priorização, como por exemplo: *stakeholders*, critérios, restrições de projeto. Apesar de terem propósitos distintos, ambas as abordagens trabalham sobre o mesmo domínio e têm potencial para minimizar a lacuna relacionada ao suporte à priorização de requisitos.

2.Análise das Abordagens e Considerações Finais

A análise das abordagens enfatizou 4 aspectos: tipo da abordagem; critérios disponíveis para priorização; classificação dos requisitos; e existência de ferramenta de suporte (Tabela 1). Para facilitar a análise, ambas às abordagens foram aplicadas à priorização de requisitos de um mesmo projeto de software acadêmico.

Tabela 1 – Características das Abordagens

	reqT/CSP	G-4REPrioritization
Tipo de Abordagem	Linguagem de Domínio Específico	Guia de Priorização
Critérios de Priorização	Permite expressar critérios através de regras gerais, com o uso de restrições, além dos critérios associados às 2 técnicas disponíveis na linguagem	Apresenta critérios que estão relacionados às 13 técnicas de priorização suportadas pelo guia
Classificação dos Requisitos	Permite expressar nível de abstração (*goal*, *feature*, cenário, etc.) e o *status* do requisito (sua etapa do ciclo de vida: elicitado, especificado, validado, implementado, etc.)	Possui etapa de identificação e classificação dos requisitos para o pré-tratamento dos requisitos. Essas etapas contemplam: definição da importância e dependência dos requisitos.
Ferramenta	Suporta ferramenta automatizada	Não suporta

Como resultado da análise das abordagens, observou-se que apesar delas terem focos diferentes, seria viável a utilização das duas em conjunto, de maneira complementar em relação:

Ao Tipo da abordagem: O reqT/CSP, por ser uma linguagem de requisitos, suporta a aplicação das técnicas de priorização através de elementos, atributos e relacionamentos específicos do domínio de requisitos, disponibilizados no metamodelo da linguagem. Já o Guia não é uma modelagem de especificação de requisitos. Assim, apesar de ambos permitem definir aspectos como dependência entre os requisitos e importância dos mesmos, apenas o reqT/CSP disponibiliza forma de especificar isso de forma semi-formal. Já o Guia suporta etapas complementares, para o processo de priorização, através da: seleção dos *stakeholders*, seleção dos requisitos, definição do critério; auxílio à seleção de técnicas de priorização de acordo como o perfil e escopo do projeto.

Aos Critérios de Priorização: o reqT/CSP proporciona uma composição de critérios abrangente através de regras, no contexto da priorização; esta forma complementa o poder de expressão com relação à definição de critérios individuais, presentes em ambas as abordagens.

À Classificação dos Requisitos: o guia possui uma classificação detalhada dos requisitos, permitindo uma grande variedade de agrupamento dos mesmos com foco na priorização, atuando também como forma de desempate. Além disso, no guia diferentes cenários podem ser avaliados na escolha das técnicas de priorização.

Ao suporte de Ferramenta: o reqT/CSP possui uma ferramenta gráfica que auxilia a realizar a priorização de requisitos. Já o guia não disponibiliza nenhuma ferramenta. Contudo, o reqT/CSP poderia ser adaptado para incorporar as etapas essenciais do Guia.

Referências

Regnell, K. Kuchcinski.:A Scala Embedded DSL for Combinatorial Optimization in Software Requirements Engineering. In: First Workshop on Do-main Specific Languages in Combinatorial Optimization, Uppsala, Sweden, Sep. 16, pp.19-34, 2013.

J. Junior, M. Lencastre, S. Galdino. G-4REPrioritization: A Guide to Help in the Prioritization of Requirements. In: The International Conference on Software Engineering, Mobile Computing and Media Informatics (SEMCMI), August 2015.

K. Pohl, C. Rupp. Requirements Engineering Fundamentals, Principles, and Techniques, 1 ed., Santa Barbara, CA, 2011.

Avaliação da Aplicação do Modelo Iterativo Incremental na Segmentação de Solicitações de Desenvolvimento de Software

Edneuci D. Audacio[2], Anna Paula de A. Lande[1], Rebeca Teodoro da Silva[2], Elias Canhadas Genvigir[2]

[1]Engenharia da Computação - Universidade Norte do Paraná - Centro de Ciências Empresariais e Sociais Aplicadas - Londrina - PR - Brasil

[2]Programa de Pós-Graduação em Informática (PPGI) - Universidade Tecnológica Federal do Paraná (UTFPR) - Campus Cornélio Procópio - PR – Brasil

deniseaudacio@gmail.com, araujolande.anna@gmail.com, rebeca.teodoro@gmail.com, elias@utfpr.edu.br

Abstract. *This article aims to present the use of Iterative Incremental model as a support tool in the software development process in order to collaborate with the error reduction. For this study, samples of records of a Study Company were taken and analyzed in two phases: the first based on the traditional model, previously used, and the second after adoption of Iterative incremental model.*

Resumo. *Este artigo tem como propósito apresentar o uso do modelo Iterativo Incremental como ferramenta de apoio no processo do desenvolvimento de software de forma a colaborar com a redução de erros. Para realização deste estudo, foram realizadas coletas e análise dos registros de uma Empresa Estudo em duas fases, a primeira baseada no modelo tradicional, previamente utilizado, e a segunda após adoção do modelo Iterativo Incremental.*

1. Introdução

Alguns dos fatores que contribuem de forma significativa para a insatisfação do cliente são o aumento no cronograma, o atraso na entrega dos produtos e a identificação de erros pelo usuário final. Contudo, mesmo com a adoção de técnicas, métodos e ferramentas de otimização, defeitos ainda podem existir [Harold 2000]. Este modelo tem como característica principal a divisão do projeto de software em miniprojetos, os quais são iterações que resultam em incrementos [Thakurta e Frederik 2010].

O modelo Incremental permite que atividades sejam executadas a cada incremento [Atkinson e Hummel 2012] e concluídas antes de dar prosseguimento ao processo, o que pode proporcionar melhorias nas especificações entre desenvolvimento e o teste e, por consequência, a redução no volume de retrabalho [Bolchini et al. 2013].

Este texto aborda os resultados obtidos em um caso real no qual a aplicação do modelo Iterativo Incremental contribuiu na redução de erros, falhas e defeitos encontrados pelo usuário final após entrega das solicitações. O uso do modelo proposto propiciou um melhor detalhamento em um processo lógico e sequencial, que beneficiou a programadores e analistas de testes.

2. Metodologia

A pesquisa ocorreu em dois momentos distintos: uma fase inicial de coleta dos registros e análise das solicitações de clientes utilizando-se o modelo tradicional da empresa e uma

segunda fase, composta por uma nova coleta de dados com base nos resultados obtidos após utilizar o modelo Incremental. Observa-se, que antes da aplicação da prática de fragmentação uma solicitação do cliente era atendida com uma única ordem de serviço (O.S), fato que dificultava o entendimento para os desenvolvedores, analistas, e equipe de teste.

3. Resultados

Os resultados apresentados neste trabalho foram obtidos em um caso real de uma empresa estudo especializada no desenvolvimento de software para o segmento de transporte localizada na cidade de Londrina/PR. Para a pesquisa, foram realizadas coletas e análises dos registros das ordens de serviços enviadas ao departamento de desenvolvimento.

A primeira análise ocorreu nos projetos denominados 129 e 130, no qual contemplou a confecção de 164 O.S, porém após o desenvolvimento houve a necessidade da geração de 83 novas O.S de correção. Portando, nota-se que sem o uso da segmentação foi totalizado 50,6 % dos esforços totais em trabalho de correções, ou seja, retrabalho.

O projeto 132, o qual foi aplicado a segmentação das ordens, foram geradas 224 O.S. e 85 novas O.S de correções, correspondente a 38% dos esforços totais. Observa-se que houve um aumento 37% no número de O.S, decorrente da segmentação realizada. Tal segmentação permitiu que as O.S tivessem uma redução em atividades, mas uma melhor descrição destes itens. Observou-se também que a organização do trabalho em incrementos e a segmentação das O.S permitiu as equipes de desenvolvimento um melhor entendimento das atividades a serem realizadas.

Vale ressaltar que não houve alterações de tecnologias nem das equipes de trabalho entre os projetos avaliados, sendo apenas introduzida as atividades de organização do ciclo de vida.

Notou-se, portanto, que a mudança do ciclo de vida pode trazer benefícios de produtividade. Desta forma, conclui-se que, por meio da adoção do modelo Iterativo Incremental e da fragmentação das solicitações do cliente em múltiplas O.S, que permitiram incluir um maior nível de detalhamento, criou-se condições para a melhoria da compreensão por parte do desenvolvedor e também mais eficácia na realização dos testes, que se tornaram mais pontuais e específicos, reduzindo a quantidade de retrabalho.

Referências

Atkinson, C.,Hummel, O. (2012) “Iterative and Incremental Development of Component-Based Software Architectures”, CBSE '12 Proceedings of the 15th ACM SIGSOFT Symposium on Component Based Software Engineering, p.77-82.

Bolchini, C. Quintarelli, E. Salice, F. Garza, P. (2013) “A Data Mining Approach to Incremental Adaptive Functional Diagnosis”, International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFTS), p.13-18.

Harrold, M. J. (2000) “Testing a roadmap”, Proceedings of the Conference on The Future of Software Engineering (ICSE), p. 61-72.

Thakurta, R., Frederik, A. (2010) "Understanding Requirements Volatility in Software Projects-An Empirical Investigation of Volatility Awareness, Management Approaches and their Applicability." System Sciences (HICSS), 43rd Hawaii International Conference on, p.1-10.