ANAIS PROCEEDINGS

CBSOFT * 2015

BRAZILIAN CONFERENCE ON SOFTWARE: THEORY AND PRACTICE

BELO HORIZONTE

WTDSoft 2015

5th WORKSHOP ON THESES AND DISSERTATIONS OF CBSOFT

CBSOFT.ORG

Sponsors:

ThoughtWorks · AvenueCode · Google Brasil · take.net · RARO LABS · Avanti · IBM

Promotion:

CNPq Conselho Nacional de Desenvolvimento Científico e Tecnológico · CAPES · FAPEMIG · SBC Sociedade Brasileira de Computação

Organizing Institutions:

UFMG · PUC Minas · CEFET-MG

WTDSoft 2015

5th WORKSHOP ON THESES AND DISSERTATIONS OF CBSOFT
September 21-22, 2015
Belo Horizonte – MG, Brazil

VOLUME 01
ISSN: 2178-6097

ANAIS | *PROCEEDINGS*

COORDENADOR DO COMITÊ DE PROGRAMA DO WTDSoft 2015 | *PROGRAM COMMITTEE CHAIR OF WTDSoft 2015*
Vander Alves (UnB)

COORDENADORES GERAIS DO CBSOFT 2015 | *CBSOFT 2015 GENERAL CHAIRS*
Eduardo Figueiredo (UFMG)
Fernando Quintão (UFMG)
Kecia Ferreira (CEFET-MG)
Maria Augusta Nelson (PUC-MG)

REALIZAÇÃO | *ORGANIZATION*
Universidade Federal de Minas Gerais (UFMG)
Pontíficia Universidade Católica de Minas Gerais (PUC-MG)
Centro Federal de Educação Tecnológica de Minas Gerais (CEFET-MG)

PROMOÇÃO | *PROMOTION*
Sociedade Brasileira de Computação | Brazilian Computing Society

APOIO | *SPONSORS*
CAPES, CNPq, FAPEMIG, Google, RaroLabs, Take.net,
ThoughtWorks, AvenueCode, Avanti Negócios e Tecnologia.

APRESENTAÇÃO

É com grande satisfação que, em nome do Comitê de Programa e da Comissão Organizadora, saudamos os participantes do V Workshop de Teses e Dissertações do CBSoft (WTDSoft 2015).

O WTDSoft é um evento promovido anualmente pelo CBSoft e este ano conta com uma série de artigos referentes a trabalhos de mestrado e doutorado na área. Foram submetidos 17 artigos ao WTDSoft, dos quais foram selecionados 12 para apresentação durante o evento e publicação em seus anais. O processo de avaliação garantiu que cada artigo tivesse três avaliações, sendo também utilizada uma fase de discussão para minimizar discrepâncias nas avaliações.

Gostaríamos de agradecer a todos que contribuíram para a realização deste evento. A qualidade deste programa é resultado da dedicação dos membros do Comitê de Programa. Somos também imensamente gratos aos professores convidados, aos participantes, e todos os autores pelos trabalhos submetidos. Finalmente, desejamos a todos um excelente evento!

Belo Horizonte, Setembro de 2015.

Vander Alves (UnB)

Coordenador do WTDSoft 2015

FOREWORD

It is with great pleasure that, on behalf of the Program and Organizing Committees, we welcome the participants of the V Workshop on Theses and Dissertations of CBSoft (WTDSoft 2015).

WTDSoft is an annually promoted event by CBSoft and this year comprises a series of papers related to masters and doctoral work in the area. 17 papers were submitted to WTDSoft, 12 of which were selected for presentation during the event and published in its proceedings. The review process ensured that each submission had three reviews and also used a discussion phase to minimize discrepancies in the ratings.
We would like to thank everyone who contributed to the realization of this event. The quality of this program is the result of the dedication of the members of the Program Committee. We are also immensely grateful to the invited professors, participants, and all authors of submitted papers. Finally, we wish everyone a great event!

Belo Horizonte, September 2015.

Vander Alves (UnB)
WTDSoft 2015 PC Chair

COMITÊ DE ORGANIZAÇÃO | *ORGANIZING COMMITTEE*

CBSOFT 2015 GENERAL CHAIRS

Eduardo Figueiredo (UFMG)
Fernando Quintão (UFMG)
Kecia Ferreira (CEFET-MG)
Maria Augusta Nelson (PUC-MG)

CBSOFT 2015 LOCAL COMMITTEE

Carlos Alberto Pietrobon (PUC-MG)
Glívia Angélica Rodrigues Barbosa (CEFET-MG)
Marcelo Werneck Barbosa (PUC-MG)
Humberto Torres Marques Neto (PUC-MG)
Juliana Amaral Baroni de Carvalho (PUC-MG)

WEBSITE AND SUPPORT

Diego Lima (RaroLabs)
Paulo Meirelles (FGA-UnB/CCSL-USP)
Gustavo do Vale (UFMG)
Johnatan Oliveira (UFMG)

COMITÊ TÉCNICO | *TECHNICAL COMMITTEE*

COORDENADOR DO COMITÊ DE PROGRAMA | *PC CHAIR*

Vander Alves (UnB)

COMITÊ DE PROGRAMA | *PROGRAM COMMITTEE*

Adenilso Simão (ICMC/USP)
Alexandre Mota (UFPE)
Augusto Sampaio (UFPE)
Auri Marcelo Rizzo Vincenzi (UFSCar)
Cecilia Rubira (UNICAMP)
Claudio Sant'Anna (UFBA)
David Deharbe (UFRN)
Edward Hermann Haeusler (PUC-Rio)
Elisa Yumi Nakagawa (ICMC/USP)
Fernando Pereira (UFMG)
Fernando Castor (UFPE)
Flavia Delicato (IM/UFRJ)
Francisco Carvalho-Junior (UFC)
Franklin Ramalho (UFCG)
Genaina Rodrigues (UnB)
Gledson Elias (UFPB)
Leila Silva (UFS)
Marcelo Fantinato (EACH/USP)
Marcelo d'Amorim (UFPE)
Marcelo de Almeida Maia (UFU)
Marco Túlio Valente (UFMG)
Martin Musicante (UFRN)
Nabor Mendonça (UNIFOR)
Raul Wazlawick (UFSC)
Rodrigo Bonifacio (UnB)
Rohit Gheyi (UFCG)
Rosângela Penteado (UFSCar)
Tiago Massoni (UFCG)
Toacy Oliveira (COPPE/UFRJ)
Uirá Kulesza (UFRN)
Vander Alves (UnB)

ARTIGOS | *PAPERS*

Mestrado | *MSc*

Doutorado | *PhD*

Folha de Rosto

Título: Um Estudo Empírico sobre Máquinas de Tradução em Tempo Real para Equipes Distribuídas de Desenvolvimento de Software

Aluno: João Henrique Stocker Pinto

Orientador: Rafael Prikladnicki

Nível (mestrado ou doutorado): Mestrado

Programa de pós-graduação: PPGCC-PUCRS

E-mail de contato do aluno: joaohenrique.jhsp@gmail.com

E-mail de contato do orientador: rafaelp@pucrs.br

Ano de ingresso no programa: Março/2014

Mês/Ano previsto para conclusão: Março/2016

Data da aprovação da proposta de tese ou dissertação (qualificação):
02/03/2015

Resumo. O desenvolvimento distribuído de software demanda uma grande cooperação entre pessoas que, em muitos casos, não dominam o mesmo idioma. Máquinas de Tradução em Tempo Real são uma alternativa a este cenário, realizando a tradução simultânea de um idioma para outro. Este artigo apresenta o andamento da pesquisa que consiste na análise de diversas ferramentas de reconhecimento, tradução e síntese de voz, na realização de experimentos e no desenvolvimento de um protótipo que faz a integração entre Reconhecedores de Voz, Máquinas de Tradução e Síntese de Voz com o objetivo de facilitar a comunicação entre equipes distribuídas de projetos de software.

Palavras-chave (de acordo com os assuntos acima relacionados): Máquinas de Tradução de Voz, Desenvolvimento Distribuído de Software, Reconhecimento de Voz, Síntese de Voz.

Sigla do Evento: SBES

Um Estudo Empírico sobre Máquinas de Tradução em Tempo Real para Equipes Distribuídas de Desenvolvimento de Software

João Henrique Stocker Pinto[1], Rafael Prikladnicki[1]

Programa de Pós-Graduação em Ciência da Computação
[1]Faculdade de Informática (FACIN)
Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS)
Caixa Postal 1429 – 90.619-900 – Porto Alegre – RS – Brasil

joaohenrique.jhsp@gmail.com, rafaelp@pucrs.br

Abstract. *Distributed Software Development demands a huge cooperation between people that, in many cases, do not master the same language. A Speech Translation System is an alternative to this scenario, simultaneously translating from a language to another. This paper presents the actual state of this research, an analysis over many tools of speech recognition, translation and voice synthesis, the executed experiments and the development of a tool that gathers Automatic Speech Recognition, Machine Translation and Voice Synthesis, having the goal of facilitating the communication between distributed software teams.*

Resumo. *O desenvolvimento distribuído de software demanda uma grande cooperação entre pessoas que, em muitos casos, não dominam o mesmo idioma. Máquinas de Tradução em Tempo Real são uma alternativa a este cenário, realizando a tradução simultânea de um idioma para outro. Este artigo apresenta o andamento da pesquisa que consiste na análise de diversas ferramentas de reconhecimento, tradução e síntese de voz, na realização de experimentos e no desenvolvimento de um protótipo que faz a integração entre Reconhecedores de Voz, Máquinas de Tradução e Síntese de Voz com o objetivo de facilitar a comunicação entre equipes distribuídas de projetos de software.*

1. Caracterização do Problema

O desenvolvimento distribuído de software requer uma grande cooperação entre pessoas com culturas diferentes. Estas culturas originam-se pelas diferenças de localização, religião, modo de governo e idioma [Carmel et al. 2001]. Em razão de fatores históricos da economia e política, o inglês se tornou a língua mais importante em termos de comunicação global. Em projetos distribuídos de software, sejam eles de empresas multinacionais ou de empresas com clientes que não falam a mesma língua, o inglês é normalmente o idioma adotado para a comunicação [Lutz et al. 2009].

Em projetos globais de desenvolvimento de software, é comum que a comunicação ocorra entre pessoas que dominam o inglês e pessoas que não têm o completo entendimento deste idioma. Em muitos destes casos, a proficiência na língua

não é o mais importante, e sim o entendimento das questões relacionadas ao projeto. Sendo assim, a comunicação ocorre entre dois grupos de pessoas: os que falam inglês como idioma nativo e os que não falam inglês como idioma nativo. Há também um terceiro grupo distinto que se caracteriza pela comunicação sem utilizar o inglês como base, por exemplo, um brasileiro que está comunicando-se com um cliente italiano. O idioma para a comunicação entre essas duas pessoas pode ser tanto o português como o italiano e estas pessoas muitas vezes acabam recorrendo a dicionários e tradutores para entender o que é falado ou lido [Calefato et al. 2014].

Considerando este cenário, um estudo sobre máquinas de tradução, reconhecimento de voz e síntese de voz está sendo feito com o objetivo de analisar quais dificuldades existem, que tipos de limitações no reconhecimento de voz podem ocorrer, quais são os idiomas disponibilizados para tradução na forma de texto e em quais idiomas já existem sintetizadores de voz. Além disso, testes com requisitos reais de software são utilizados nas tecnologias alvo e métricas são coletadas com o intuito de analisar o quão desenvolvidas estas ferramentas estão. O estudo está sendo feito com as seguintes tecnologias: *Google Translate*, *Google Web Speech API*, *Chrome.tts API, Bing Translator* e a nova ferramenta do *Skype* para conversas, o *Skype Translator*.

Desta forma, o principal objetivo de pesquisa é verificar como a tradução em tempo real pode ajudar na comunicação e no entendimento entre equipes distribuídas de desenvolvimento global de software, focando principalmente no processo de reconhecimento de voz e nas máquinas de tradução.

2. Fundamentação Teórica

Nesta seção será apresentada uma base teórica, destacando-se os principais aspectos sobre Reconhecimento de Voz, Máquinas de Tradução e Síntese de Voz.

2.1 Reconhecimento de Voz

Reconhecimento de voz ou *Automatic Speech Recognition (ASR)* é uma área que cresce desde 1960, quando este tipo de sistema era usado somente em laboratórios, tinham uma taxa de erros elevada e um custo inaceitável para o uso em aplicações cotidianas [Reddy 1976]. O reconhecimento de voz consiste basicamente da transcrição de palavras faladas para texto escrito [Huang et al. 2014]. A partir da década de 70, grandes investimentos na área foram feitos, o que resultou em diversos projetos e demonstrações da tecnologia, aumento do entendimento de fonética, sintaxe e de contexto. Conforme o entendimento geral aumentava, a complexidade também crescia, e mais desafios eram encontrados.

Um desses desafios se dá pelo fato de que a fala pode acontecer de forma ininterrupta, ou seja, falamos uma frase inteira sem que seja necessária uma pausa, por exemplo. Esta característica faz com que seja difícil identificar quando uma palavra termina e a outra começa. Outro fator que aumenta a complexidade de desenvolver um *ASR* é que a fala pode ter entonações diferentes, dependendo do contexto em que estamos. Além disso, o tamanho do vocabulário, o número de pessoas falando e o barulho ambiente são fatores que podem dificultar um reconhecimento aceitável de voz [Reddy 1976].

2.2 Máquinas de Tradução

A ideia de um processo mecanizado de tradução remonta do século 17, porém somente no século 20, em meados de 1930 os primeiros passos reais foram dados quando Georges Artsrouni e Petr Troyanskii requisitaram uma patente para máquinas de tradução. Troyanskii propôs não somente um método automatizado para tradução, mas também um esquema baseado em códigos gramaticais e saídas de como a análise e a síntese deveriam funcionar [Hutchins et al. 2005].

Existem alguns modelos diferentes para se desenvolver uma máquina de tradução, mas neste estudo serão apresentados 2 destes modelos. O modelo baseado em regras, conhecido como *Rule Based*, foi a primeira abordagem criada e consiste basicamente de duas grandes áreas: as regras, que cuidam da parte sintática da gramática e a parte léxica, que é responsável por manter todo o acervo de palavras de um determinado idioma [Bennett et al. 1985]. Neste sistema, uma pessoa com experiência em tradução usa seu conhecimento para desenvolver as regras que serão usadas na tradução. O processo ocorre quando um determinado texto de entrada passa pelas regras e é traduzido para um idioma destino. A ideia é simples, porém este é um método trabalhoso e de difícil manutenção, pois as regras devem ser constantemente verificadas [Lagarda et al. 2009]. O segundo modelo é conhecido como *Corpus Based.* Sua principal diferença em relação ao *Rule Based* é que este método não utiliza a intervenção humana para traduzir as sentenças e sim cálculos matemáticos baseados em estatística [Calefato et al. 2014]. A essência deste método é organizar frases, grupos de palavras e palavras individuais e então calcular as probabilidades de uma melhor correspondência acontecer, por meio de padrões de tradução já existentes em outros documentos [Hutchins 1995].

2.3 Síntese de Voz

Os primeiros registros de um sistema capaz de sintetizar voz datam do século 18, quando em 1779, na Rússia, Kratzenstein construiu um modelo mecânico capaz de reproduzir algumas vogais. Em 1791, Wolfgang von Kempelen conseguiu pela primeira vez obter sucesso em sintetizar a fala de forma conectada, utilizando um sistema pneumático para expelir o ar de um balão e reproduzir palavras do início ao fim [Mannell 2010]. Ao longo do tempo, o interesse em sintetizadores de voz se manteve. Inicialmente construídos como sistemas mecânicos, foram evoluindo até que pudessem analisar texto, passando a usar sistemas eletrônicos de codificação, como o ASCII, por exemplo. Nas décadas de 1960 e 1970 essa ideia inicial foi desenvolvida: o MITalk podia converter texto em fala [Holmes et al. 2001] e a partir deste momento a ideia de que um sintetizador de voz era simplesmente falar palavra por palavra, convertê-las uma por uma em texto e concatená-las em uma *String* final deu lugar a real complexidade do problema [Sproat et al. 1999]. A pronúncia do leitor, a ênfase em determinadas palavras e a não ênfase em outras, a variação de entonação entre os diferentes leitores criaram possibilidades de pesquisa e desafios.

3. Contribuições

Este trabalho tem como objetivos: (1) verificar e validar o uso de ferramentas de tradução em tempo real para equipes distribuídas de desenvolvimento de software, mas

também (2) propor uma ferramenta capaz de facilitar a comunicação nas etapas do processo de desenvolvimento, construindo um protótipo que faz a integração automatizada entre as tecnologias que o Google disponibiliza, como o *Web Speech API*, *Google Translate* e o *Chrome.tts* API, unindo o reconhecimento de voz de um idioma origem, sua tradução para um idioma destino e sua síntese para este idioma e por fim (3) revisar e apresentar uma base teórica sobre as origens, características e a complexidade que envolve o desenvolvimento de Sistemas Reconhecedores de Voz, das Máquinas de Tradução e da Síntese de Voz.

4. Estado Atual do Trabalho

A seguir descrevemos as tarefas que se encontram em atividade ou que estão planejadas para a continuidade do trabalho:

- Revisão da literatura nas áreas de Reconhecimento de Voz, Máquinas de Tradução e Síntese de Voz, com o objetivo de aprofundar a argumentação teórica da ferramenta proposta.
- Realizar o último experimento previsto, que deve ocorrer entre os meses de agosto e outubro, totalizando 3 experimentos – um *mobile* e dois *web*. Os dois experimentos anteriores foram realizados em parceria com a Universidade de Bari, na Itália e tiveram a participação de, até o momento, 10 voluntários e um universo de 1470 frases analisadas.
- Continuar o desenvolvimento do protótipo cujo principal objetivo é a automatização dos processos mencionados da Seção 2, pois atualmente as entradas e saídas em cada componente devem ser feitas de forma manual. Por fim terminar a escrita da dissertação.

5. Comparação com Trabalhos Relacionados

Em relação à Tradução de Voz e equipes distribuídas de desenvolvimento de software o trabalho de Duarte et al. (2014) apresenta um estudo teórico sobre alternativas para a falta de profissionais fluentes em idiomas como o inglês em países com potencial para atuação no mercado global de T.I. Este trabalho apresenta a ideia de unir os três conceitos apresentados na Seção 2, porém se limita em descrever a base teórica, não propondo nenhum ferramental. Calefato et al. (2014) executou um estudo envolvendo um conjunto de 60 sentenças utilizando as ferramentas *Google Web Speech API* e *Google Translator* com 8 voluntários da área de desenvolvimento de software para avaliar a eficiência destas tecnologias na comunicação entre equipes distribuídas. Os resultados obtidos foram em ambos os casos satisfatórios, chegando a uma média geral de 70% de precisão na tradução correta das palavras e frases [Calefato et al. 2014].

6. Avaliação dos Resultados

A avaliação final dos resultados está prevista para acontecer após o terceiro experimento, porém resultados dos dois experimentos anteriores mostram uma qualidade significativa no reconhecimento de voz, chegando a 75% de precisão. Já na tradução obtivemos um resultado inesperado, com uma precisão de aproximadamente 35%. Os idiomas avaliados foram o inglês, o italiano e o português e as tecnologias usadas foram as citadas anteriormente, fornecidas pelo *Google*. No último experimento

está previsto o uso do *Skype Translator* e os voluntários para os testes serão da Universidade de Bari e da PUCRS. Com os resultados prévios, acreditamos ser possível que equipes distribuídas de desenvolvimento de software possam se comunicar e desenvolver projetos utilizando este tipo de tecnologia de tradução em tempo real. Outros testes ainda podem acontecer ao longo do desenvolvimento do trabalho para maturar mais os conceitos, entretanto a perspectiva obtida até o momento é satisfatória.

Referências

Bennett, W. S. and Slocum, J. (1985) "The LRC Machine Translation System," Computational Linguistics, vol. 11, p. 111-121.

Calefato, F., Lanubile, F., Prikladnicki, R., Pinto, J., (2014) "An Empirical Simulation-based Study of Real-Time Speech Translation for Multilingual Global Project Teams", in ESEM 2014.

Carmel, E. and Agarwal, R. (2001) "Tactical Approaches for Alleviating Distance in Global Software Development", in IEEE Softw., vol. 18, no. 2, p. 22-29.

Duarte, T. e Prikladnicki, R. (2014) "Máquinas de Tradução Aplicadas à Comunicação em Tempo Real para o Desenvolvimento Distribuído de Software", Dissertação de Mestrado, PPGCC-PUCRS, Porto Alegre.

Holmes, J. and Holmes, W. (2001) "Speech Synthesis and Recognition", CRC Press 2001, vol. 2, pp 91-92.

Huang, X., Baker, J., and Reddy, R., (2014) "A historical perspective of speech recognition," Commun. *ACM*, vol. 57, no. 1, January 2014, pp. 94-103.

Hutchins, J. (1995) "Machine Translation: A Brief History", Edited by E.F.K.Koerner and R.E.Asher. Oxford: Pergamon Press, 1995. Pages 431-445.

Hutchins, J. (2005) "The History of Machine Translation in a Nutshell", http://www.hutchinsweb.me.uk/Nutshell-2005.pdf, acesso em 21/05/2014.

Lagarda, A.-L., Alabau, V., Casacuberta, F., Silva, R., and Díaz-de-Liaño, E. (2009) "Statistical Post-Editing of a Rule-Based Machine Translation System" in NAACL HLT, May 31 - June 5, Boulder, Colorado.

Lutz, B. (2009) "Linguistic Challenges in Global Software Development: Lessons Learned in an International SW Development Division", in 4th IEEE ICGSE, July 13-16, Limerick, Ireland.

Mannell, R. (2010) "A Brief Historical Introduction to Speech Synthesis: A Macquarie Perspective", http://clas.mq.edu.au/speech/synthesis/history_synthesis, acesso em 24/05/2014.

Reddy, D.R., (1976) "Speech Recognition by Machine: A Review," Proceedings of the IEEE, vol. 64, no. 4, 1976, pp. 501-531.

Sproat, R. & Olive, J. (1999) "Text-to-Speech Synthesis," Digital Signal Processing Handbook, Ed. Vijay K. Madisetti and Douglas B. Williams, Boca Raton: CRC Press LLC, 1999.

Contratos REST robustos e leves: uma abordagem em Design-by-Contract com NeoIDL

Lucas F. Lima[1]
Orientadores: Rodrigo Bonifácio de Almeida[2]**, Edna Dias Canedo**[1]

[1]Departamento de Engenharia Elétrica – Universidade de Brasília – UnB
CEP 70910-900 – Campus Darcy Ribeiro – Asa Norte – Brasília – DF – Brasil

[2]Departamento de Ciência da Computação – Universidade de Brasília – UnB
CEP 70910-900 – Campus Darcy Ribeiro – Asa Norte – Brasília – DF – Brasil

lucas.lima@aluno.unb.br, rbonifacio@cic.unb.br, ednacanedo@unb.br

Nível: Mestrado
Ano de Ingresso: 2014
Previsão de conclusão: Dezembro/2015
Aprovação da Proposta: Janeiro de 2014
Eventos Relacionados: SBES

Resumo. *O trabalho de pesquisa de mestrado, sumarizado neste artigo, objetiva fortalecer a especificação de contratos para soluções concebidas sob o paradigma de computação orientada a serviços. A robustez é buscada por meio de construções que agregam Design-by-Contract à linguagem NeoIDL, de modo que os serviços especificados assegurem as pós-condições, desde que satisfeitas suas pré-condições. A proposta será submetida a validação empírica, verificando regras de transformações em um novo plugin para a NeoIDL.*

1. Introdução e Caracterização do Problema

A computação orientada a serviços (*Service-oriented computing, SOC)* tem se mostrado uma solução de *design* de *software* que favorece o alinhamento às mudanças constantes e urgentes nas instituições [Chen 2008]. Os benefícios de SOC estão diretamente relacionados ao baixo acoplamento dos serviços que compõem a solução, de forma que as partes (nesse caso serviços) possam ser substituídas e evoluídas facilmente, ou ainda rearranjadas em novas composições. Contudo, para que isso seja possível, é necessário que os serviços possuam contratos bem definidos e independentes da implementação.

Por outro lado, as linguagens de especificação de contratos para SOA apresentam algumas limitações. Por exemplo, a linguagem WSDL (*Web-services description language*) [Zur Muehlen et al. 2005] é considerada uma solução verbosa que desestimula a abordagem *Contract First*. Por essa razão, especificações WSDL são usualmente derivadas a partir de anotações em código fonte. Além disso, os conceitos descritos em contratos na linguagem WSDL não são diretamente mapeados aos elementos que compõem as interfaces do estilo arquitetural REST (*Representational State Transfer*). Outras alternativas para REST, como Swagger e RAML[1], usam linguagens de propósito geral (em particular JSON) adaptadas para especificação de contratos. Ainda que façam uso de contratos mais sucintos que WSDL, essas linguagens não se beneficiam da clareza típica das linguagens específicas para esse fim (como IDLs CORBA) e não oferecem mecanismos semânticos de extensibilidade e modularidade.

Com o objetivo de mitigar esses problemas, a linguagem NeoIDL foi proposta para simplificar a especificação de serviços REST com mecanismos de modularização, suporte a anotações, herança em tipos de dados definidos pelo desenvolvedor, e uma sintaxe simples e concisa semelhante às *Interface Description Languages* – IDLs – presentes em *Apache Thrift*™e CORBA™. Por outro lado, a NeoIDL, da mesma forma que WSDL, Swagger e RAML não oferece construções para especificação de contratos formais de comportamento como os presentes em linguagens que suportam DBC (*Design by Contract*) [Meyer 1992], como JML, Spec# e Eiffel.

Dessa forma, os objetivos dessa pesquisa envolvem inicialmente investigar o uso de construções DBC no contexto da computação orientada a serviços e conduzir uma revisão sistemática da literatura para identificar os principais trabalhos que lidam com a relação entre DBC e SOC. Em seguida especificar e implementar novas construções para a linguagem NeoIDL, de tal forma que seja possível especificar contratos mais precisos; além de definir regras de transformação das novas construções NeoIDL para diferentes tecnologias (como Twisted) que suportam a implementação de serviços em REST. Por fim, conduzir uma validação empírica da proposta usando uma abordagem mais exploratória, possivelmente usando a estratégia *pesquisa-ação*.

2. Fundamentação Teórica

SOC é um estilo arquitetural cujo objetivo é prover baixo acoplamento por meio da interação entre agentes de software, chamados de serviços [He 2003]. A chave para que a solução baseada em SOC tenha custo-benefício favorável é o reuso, o qual somente é possível se os serviços possuírem interfaces ubíquas, com semânticas genéricas

[1] http://raml.org/spec.html

e disponíveis para seus consumidores. Usualmente, a comunicação com os serviços, e entre eles, é feita por meio da troca de mensagens com uso de *serviços web*, que seguem padrões abertos de comunicação e que atuam sobre o protocolo HTTP. SOAP e REST [Fielding 2000] são as tecnologias mais usadas para implementação de serviços web, sendo que REST tem ganhado mais relevância nos últimos anos.

Entre os oito princípios para desenvolvimento SOA descritos em [Erl 2008], existe um especial interesse no *contrato padronizado* (*Standardized Service Contract*), o qual sugere que serviços de um mesmo inventário devem seguir os mesmos padrões de *design*, de modo a favorecer o reuso e a composição. Esse princípio prega a abordagem *Contract First*, em que a concepção do serviço parte da especificação do contrato e não com a geração do contrato a partir código. É importante destacar que não existe um padrão para especificar contratos em REST, o que motivou o desenho da NeoIDL [Bonifácio et al. 2015], uma linguagem específica do domínio para especificação de contratos REST e que, diferente das linguagens existentes, provê recursos de modularidade e herança de tipos de dados customizados. Utilizando uma abordagem transformacional, a NeoIDL oferece suporte ferramental que gera código fonte para diferentes linguagens e plataformas REST, a partir de um conjunto de módulos NeoIDL onde são especificados os tipos de dados e os serviços.

A Figura 1 apresenta um exemplo de módulo NeoIDL com a definição de um tipo de dado `ItemDoCatalogo` e duas operações (chamadas de capacidades na terminologia REST): `atualizarItem` (operação do tipo POST associada ao *endpoint* `catalogo`) e `pesquisarItem` (operação do tipo GET, do mesmo *endpoint*).

```
1  module Catalogo {
2    path = "/catalogo";
3
4    struct ItemDoCatalogo {
5      string id;
6      string descricao;
7      float valor;
8    };
9    service Catalogo {
10     path = "/catalogo";
11     @put Catalogo atualizarItem(string id, ItemCatalogo itemCatalogo);
12     @get Catalogo pesquisarItem(string id);
13   };
14 }
```

Figure 1. Exemplo de um módulo escrito em NeoIDL

Conforme mencionado, o suporte ferramental da NeoIDL processa um conjunto de módulos escritos na linguagem NeoIDL, gerando o código com a estrutura para implementação dos serviços descritos. A versão atual do ambiente NeoIDL suporta geração de códigos em Java e Python para as plataformas *neoCortex*[2] e *Twisted*. Entretanto, esse ambiente é extensível por meio da implementação de novos plugins [Bonifácio et al. 2015].

Limitação da NeoIDL. Atualmente a NeoIDL permite apenas a especificação de *weak contracts*, o que não permite estabelecer obrigações entre fornecedores e clientes de

[2]proprietária do Exército Brasileiro

serviços. Esse tipo de obrigação pode ser estabelecida com alguma técnica de *Design-by-contract* [Meyer 1992] (DBC) – na qual o consumidor e o fornecedor de serviços firmam entre si um conjunto de garantias. Em um contexto mais simples, de um lado o consumidor deve garantir que, antes da chamada a um serviço (ou um método de uma classe), os parâmetros de entrada devem ser respeitados (essas garantias são denominadas pré-condições). Do outro lado, o fornecedor deve garantir que, uma vez respeitadas as pré-condições, as propriedades relacionadas ao sucesso da execução (pós-condições) são preservadas. DBC tem o objetivo de aumentar a robustez do sistema e tem na linguagem Eiffel [Meyer 1991] um de seus precursores. Segundo Eiffel Software[3], o uso de DBC na concepção de sistemas é uma abordagem sistemática que tende a reduzir a quantidade de erros observados nos sistemas. Mais recentemente, algumas técnicas de *design-by-contract* foram especificadas e implementadas para outras linguagens de programação, como JML para a linguagem Java [Leavens et al. 2006] e `Spec#` para a linguagem `C#` (e demais linguagens da plataforma .NET) [Barnett et al. 2005].

3. Estado Atual do Trabalho

As construções para especificar pré-condições e pós-condições[4] na linguagem da NeoIDL estão sendo definidas, de forma a agregar às especificações mecanismos para estabelecer as condições de execução e as garantias dos serviços. Note que, além da validação dos parâmetros de entrada e valores de retorno típicos do DBC, a abordagem proposta nessa dissertação permitirá a inclusão de chamadas a serviços REST como pré e pós condições. Para ilustrar algumas decisões de projeto, a seguir são apresentados dois exemplos do uso da abordagem que está sendo proposta.

O trecho de módulo NeoIDL constante da Figura 2 destaca a capacidade `incluirItem` (l. 7). Para que se tenha o item incluído no catálogo, é necessário fornecer um valor para o atributo descrição (obrigatório). A pré-condição (l. 5) se inicia com a notação `/@require` e possui validação sobre o atributo descrição. Note que o atributo é precedido da palavra `old`, que indica o valor do atributo antes do processamento do serviço. De forma análoga, está previsto o prefixo `new`, que indica o valor do atributo após a execução do serviço. Na Figura 2 ainda é estabelecida uma cláusula `/@otherwise` (l. 6), estabelecendo que a falha no atendimento da pré-condição deve levar a uma resposta com o código HTTP 412 (falha de pré-condição).

```
1 module Catalogo {
2     service Catalogo {
3     path = "/catalogo";
4
5     /@require old.descricao != null
6     /@otherwise HTTP_Precondition_Failed
7     @post Catalogo incluirItem (string id, string descricao, float valor);
8     (...)
9 }
```

Figure 2. Exemplo da notação DBC na linguagem NeoIDL

A Figura 3 apresenta a capacidade `excluirItem` (l. 8). Nesse

[3]Building bug-free O-O software: An introduction to Design by Contract(TM). https://archive.eiffel.com/doc/manuals/technology/contract/

[4]Não se identificou, até o estágio atual da pesquisa, aplicabilidade de invariantes em serviços REST.

caso são estabelecidas pré e pós condições, ambas com chamadas ao serviço `Catalogo.pesquisarItem`. Antes da execução da capacidade `excluirItem`, é verificado se o item exite. Caso exista, a pré-condição é satisfeita e o item é excluído. A pós-condição (l. 6) confirma que o item não consta mais do catálogo. Se a pré-condição não for satisfeita, a cláusula `/@otherwise` (l. 7) informa ao cliente do serviço que o item não foi localizado (código HTTP 404 - Objeto não encontrado).

```
1  module Catalogo {
2      (...)
3      service Catalogo {
4      path = "/catalogo";
5      /@require call Catalogo.pesquisarItem(old.id)==HTTP_OK
6      /@ensure call Catalogo.pesquisarItem(old.id)==HTTP_Not_Found
7      /@otherwise HTTP_Not_Found
8      @delete Atividade excluiItem(string id);
9      (...)
10 }
```

Figure 3. Exemplo da notação DBC na linguagem NeoIDL com chamada a serviço

Ou seja, a proposta envolve elementos concebidos na linguagem JML [Leavens et al. 2006] (como as construções `old` e `new`) com elementos típicos do estilo arquitetural REST (chamadas a serviços sem estado utilizando métodos HTTP, semântica de retorno das operações observando os códigos de retorno HTTP, etc.), os quais habilitam não apenas o estabelecimento de contratos mas também uma forma de guiar a composição de serviços. Novos *plugins* NeoIDL devem ser implementados para permitir chamadas a serviços a partir de pré e pós condições.

Validação. Em paralelo à especificação das extensões de linguagem NeoIDL e implementação de plugins, está sendo feito o planejamento da validação da proposta – que deve seguir uma abordagem de pesquisa-ação considerando cenários reais (possivelmente com exemplos do Exército Brasileiro, parceiro na concepção da primeira versão da NeoIDL). Um cenário candidato é de verificação de controle de acesso aos serviços, a partir de pré-condições com chamada ao serviço de autorização, aplicando-se assim solução a um problema real (pesquisa-ação). Serão empregadas técnicas de avaliação empírica em engenharia de software [Shull et al. 2008] para coleta e análise dos dados.

4. Trabalhos Relacionados

Verifica-se que, embora definido já há alguns anos, DBC continua sendo objeto de pesquisa [Poyias 2014], [Rubio-Medrano et al. 2013], [Belhaouari et al. 2012]. Nesse dois últimos casos, o estudo de caso está associado a controle de acesso, cenários aderentes a que se pretende atingir com esta pesquisa.

Durante a pesquisa bibliográfica, muito embora o primeiro princípio SOA estabelecido por [Erl 2008] recomende *Contract First*, não se identificou publicação que associasse diretamente *Design-by-Contract* à especificação de contratos SOA.

Os trabalhos que mais se aproximam são os de [Ling et al. 2003] e [Heckel and Lohmann 2005]. O primeiro define uma forma de *Design-by-Contract* para arquiteturas de *workflow*. O autor considera, porém, que para grandes arquiteturas, a complexidade aumenta o risco de falha de *design*. Já [Heckel and Lohmann 2005] foca

na especificação de modelos para DbC em Web Services, não tratando a especificação concreta do contrato. Além disso, se restringe a Web Services SOAP.

Referências

Barnett, M., Leino, K. R. M., and Schulte, W. (2005). The `Spec#` programming system: An overview. In *Construction and analysis of safe, secure, and interoperable smart devices*, pages 49–69. Springer.

Belhaouari, H., Konopacki, P., Laleau, R., and Frappier, M. (2012). A design by contract approach to verify access control policies. In *Engineering of Complex Computer Systems (ICECCS), 2012 17th International Conference on*, pages 263–272. IEEE.

Bonifácio, R., de Castro, T. M., Fernandes, R., Palmeira, A., and Kulesza, U. (2015). Neoidl: A domain-specific language for specifying rest services. *SEKE*, page 10.

Chen, H.-M. (2008). Towards service engineering: service orientation and business-it alignment. In *Hawaii International Conference on System Sciences, Proceedings of the 41st Annual*, pages 114–114. IEEE.

Erl, T. (2008). *Soa: principles of service design*, volume 1. Prentice Hall Upper Saddle River.

Fielding, R. (2000). Fielding dissertation: Chapter 5: Representational state transfer (rest).

He, H. (2003). What is service-oriented architecture. *Publicação eletrônica em 30/09/2003*, 30:50.

Heckel, R. and Lohmann, M. (2005). Towards contract-based testing of web services. *Electronic Notes in Theoretical Computer Science*, 116:145–156.

Leavens, G. T., Baker, A. L., and Ruby, C. (2006). Preliminary design of jml: A behavioral interface specification language for java. *ACM SIGSOFT Software Engineering Notes*, 31(3):1–38.

Ling, S., Poernomo, I., and Schmidt, H. (2003). Describing web service architectures through design-by-contract. In *Computer and Information Sciences-ISCIS 2003*, pages 1008–1018. Springer.

Meyer, B. (1991). *Eiffel: The Language*, volume 1. Prentice Hall.

Meyer, B. (1992). Applying'design by contract'. *Computer*, 25(10):40–51.

Poyias, K. (2014). *Design-by-contract for software architectures*. PhD thesis, Department of Computer Science, University of Leicester.

Rubio-Medrano, C. E., Ahn, G.-J., and Sohr, K. (2013). Verifying access control properties with design by contract: Framework and lessons learned. In *Computer Software and Applications Conference (COMPSAC), 2013 IEEE 37th Annual*, pages 21–26. IEEE.

Shull, F., Singer, J., and Sjøberg, D. I. (2008). *Guide to advanced empirical software engineering*, volume 93. Springer.

Zur Muehlen, M., Nickerson, J. V., and Swenson, K. D. (2005). Developing web services choreography standards—the case of rest vs. soap. *Decision Support Systems*, 40:9–29.

Uma Extensão de Elementos BPMN para Modelagem de Características Autonômicas em Processos de Negócio

Aluno
Bruno Nascimento de Figueiredo
Orientador[1]/Co-Orientadora[2]
[1]Jaelson Freire Brelaz de Castro
[2]Karolyne Maria Alves de Olivera

Pós-graduação em Ciência da Computação – Centro de Informática (CIn)
Universidade Federal de Pernambuco (UFPE) – Pernambuco, PE – Brasil

{bnf,jbc,kmao}@cin.ufpe.br

Nível: Mestrado
Ano de Ingresso: Abril de 2014
Previsão de conclusão: Fevereiro de 2016
Avaliação da Proposta: Não qualificado
Evento: SBES

Resumo. *O Business Process Modeling Notation (BPMN) 2.0 é uma notação bastante difundida no mercado, utilizada em modelagem de processos de negócio. No entanto, BPMN 2.0 tem algumas limitações relacionadas à sua modelagem, dificultando a descrição de fenômenos encontrados em situações reais. Entre as limitações do BPMN 2.0 encontra-se a baixa expressividade de recursos de computação autônoma, variabilidade pelo uso de contexto e operacionalização de Requisitos não funcionais (NFR) por exemplo. Neste cenário, a abordagem Multi Level Autonomic Business Process (MABUP) é uma opção para auxiliar no gerenciamento de processos de negócios autonômicos, por considerar aspectos relativos à variabilidade, compreensibilidade, escalabilidade e requisitos não-funcionais. Entretanto, esta abordagem necessita de modelos com maior expressividade, como BPMN 2.0, de modo a facilitar sua aplicação. Este estudo propõe desenvolver uma extensão conservativa, ou seja, sem invalidar as regras presentes no modelo original BPMN 2.0 e incorpore elementos com características autonômicas, presente na abordagem MABUP, aumentando assim tanto a expressividade de características autonômicas do BPMN 2.0, como a usabilidade da abordagem MABUP e ainda auxilie na aplicação da abordagem MABUP.*

Palavras - Chaves: *Modelagem de Processos, BPMN, Processo de Negócio Autonômico, MABUP.*

1. Caracterização do Problema

A evolução dos modelos de gestão, assim como da modelagem de processos de negócio, está ligada diretamente à capacidade que a comunidade acadêmica aprimorou no tocante à simulação do comportamento das empresas, seu mercado, bem como sua estrutura. Dentre as técnicas de modelagem de processos de negócio, uma das mais adotadas é a notação BPMN 2.0 [Model, Business Process 2011]. Acompanhado pela necessidade de automação do fluxo de trabalho, isso inclui o trabalho de modelagem de processos, com objetivo de aumentar eficiência, diminuir de custo e tempo na execução das atividades inerentes ao processo comparado a métodos manuais surgiu a tecnologia Business Process Management System (BPMS) [T. Cruz, 2008]. Nesse contexto de automação de processos, uma abordagem multiníveis em modelagem de processos de negócio autonômicos denominada MABUP, do inglês *Multi Level Autonomic Business Process* [Oliveira, et al. 2012], é uma opção para auxiliar no gerenciamento e automação de processos de negócios considerando aspectos relativos a variabilidade, compreensibilidade, escalabilidade e NFR (requisitos não-funcionais).

A expressividade de recursos de computação autônoma, tais como, variabilidade pelo uso de contextos e operacionalização de NFR em modelagem de processo de negócio ainda é uma questão pouco explorada tanto pela comunidade acadêmica ou pelo mercado [Oliveira, et al. 2012]. O BPMN 2.0 tem algumas limitações relacionadas à construção de modelos, apesar da sua ampla adoção, sugerindo que os usuários finais não estão aptos a descrever fenômenos encontrados em situações reais [Recker, et al. 2006]. Embora os benefícios da abordagem MABUP incluam atingir mais expressividade e facilidade de compreensão em cenários autonômicos [Oliveira, et al. 2013], atualmente a abordagem não possui uma ferramenta para gerar modelos autonômicos, ou seja, modelos que auxiliem na execução das atividades inerentes ao processo, que expressem seus conceitos e auxiliem na aplicação da abordagem autonômica. Identificada as limitações atuais da abordagem MABUP, este estudo se propõe desenvolver uma extensão conservativa, sem comprometer as regras já existentes no BPMN 2.0, incorporando novos elementos com características autonômicas, auxiliando na aplicação na abordagem MABUP, aumentando, assim, tanto a expressividade de características autonômicas do BPMN 2.0, como a usabilidade da abordagem MABUP.

Neste estudo apresenta-se o desenvolvimento da extensão da especificação do BPMN 2.0, que permite a integração dos conceitos de características autonômicas dentro de modelos de processos de negócios autonômicos tais como variabilidade pelo uso de contextos e operacionalização de NFR. Os principais resultados apresentados neste estudo são: i) Um conjunto de conceitos relacionados a sistemas autonômicos que serão considerados na extensão de BPMN 2.0; ii) O metamodelo (sintaxe abstrata) referente às entidades da ferramenta proposta de forma integrada ao metamodelo de BPMN 2.0; iii) A extensão da sintaxe concreta de BPMN 2.0 referente aos elementos do metamodelo; iv) A evolução de uma ferramenta de modelagem de BPMN 2.0 de modo a incorporar a extensão proposta na linguagem; e por fim, v) um experimento inicial relacionado ao uso da ferramenta e da extensão proposta.

2. Fundamentação Teórica

A modelagem visa criar um modelo de processos por meio da construção de diagramas operacionais sobre seu comportamento, testando suas reações sob diversas condições, a fim de certificar que seu funcionamento atenderá aos requisitos estabelecidos. Para otimizar o trabalho de modelagem é preciso escolher uma das técnicas de representação de processos e selecionar uma ferramenta apropriada. Contudo, caso a empresa tenha uma prática avançada em gestão de processos, é possível tirar melhor proveito dos modelos construídos. Uma opção é adquirir uma ferramenta de gestão de processos para avançar com análise e simulação, criando um ambiente para automação de processos [Valle, R., & Oliveira, S. B. 2012].

MABUP é uma abordagem multiníveis para automatizar processos de negócio que através do uso de contexto e atributos de qualidade e para orientar uma adaptação necessária em sistemas autonômicos. Com este fim, esta abordagem utiliza alguns conceitos como modularidade [H. Reijers, 2008], contextualização [J. Luis, D. Vara, et al. 2010], atributos não-funcionais para conduzir configuração [Santos, Emanuel, et al. 2011] e Análise de Comunicação [S. Espana, 2009]. MABUP consiste em quatro níveis distintos, mas complementares, de abstração: Análise de Comunicação (Nível Organizacional), Nível Tecnológico, Nível Operacional e de Nível de Serviço. Estes níveis fornecem aspectos que representam adequadamente características autonômicas tais como: variabilidade pelo uso de contextos e operacionalização de NFR.

Uma das alternativas para implantar a automatização de processos é através do uso dos BPMS que são ferramentas de tecnologia da informação com objetivo de possibilitar a implantação do motor BPM, integrando qualquer elemento que necessite interagir por meio da automatização dos processos de negócio [T. Cruz, 2008]. Dentre as ferramentas BPMS disponíveis no mercado, uma vem se destacando por ser de código aberto, o Activiti, com suporte à BPMN 2.0, que pode ser usado para modelar e estender diagramas BPMN [Activiti, 2015].

3. Comparação com trabalhos relacionados

Com objetivo de tratar indicadores chave de desempenho necessários para garantir a atualidade e a eficácia dos processos operacionais de negócio, o estudo de [Friedenstab, J., et al. 2012] utilizou o conceito de *Business Activity Monitoring* (BAM) para analisar e apresentar informações em tempo real sobre as atividades de negócios. Para modelar os conceitos envolvidos (por exemplo, processos de negócios, logs de auditoria, medidas de desempenho) os autores fizeram extensões de BPMN 2.0, mas não apresenta uma ferramenta de auxílio para modelagem da abordagem proposta. Em Time-BPMN [Gagne, et al. 2009] é apresentada uma extensão do BPMN 2.0 que capta a perspectiva temporal de processos de negócios. Esta extensão aborda várias restrições temporais e dependências que podem ocorrer durante a modelagem de processos de negócios, mas não apresenta uma ferramenta de auxílio para modelagem da abordagem proposta

4. Estendendo o BPMN

Inicialmente foi necessário agrupar os conceitos relacionados a sistemas autonômicos considerados na abordagem MABUP. Assim, foram definidos os elementos por nível: (i) Nível Organizacional - elemento Atividade Crítica, que é um elemento quando

utilizado representa a criticidade para o negócio e deve ser refinado para fornecer informações sobre o seu comportamento para indicar as subatividades que deverão ser monitoradas de acordo com características autonômicas. Foi mapeado como extensão de um sub-processo, por ter semântica similar. (ii) Nível Tecnológico - elemento Tarefa Monitorada, tipo especial de Tarefa que fornece a característica autonômica necessária de garantida e para que atributos de qualidade. (iii) Nível Operacional – elemento Ponto de Variação, tipo especial de gateway de evento com a semântica de indicar a variabilidade de desvio de um atributo de qualidade de acordo com um determinado contexto. Por sua vez o Evento de Contexto, tipo especial de evento onde são inseridas as expressões de contexto que caso sejam atendidas vão ser operacionalizadas. A operacionalização de uma ação autonômica é feita através de uma Tarefa Operacional, tipo especial de tarefa que implementa as adaptações autonômicas no sistema afim de garantir os atributos de qualidades definidos na modelagem.

Neste estudo foi realizada uma extensão conservativa, ou seja, sem invalidar as regras presentes no modelo original BPMN 2.0. Os elementos centrais de uma extensão BPMN 2.0 são: *ExtensionDefinition* e *ExtensionAttributeDefinition*. Este último, define uma lista de atributos que podem ser anexados a qualquer elemento BPMN. A lista de atributos, que define o nome e o tipo, pode ser adicionada a qualquer elemento filho de *BaseElement*. A classe *Extension* é um atributo da classe *Definition* e, através dela, é possível associar um *ExtensionDefinition* a qualquer elemento filho de *BaseElement*. Deste modo, torna-se viável estender qualquer *BaseElement* de quaisquer uma de suas classes filhas [Model, Business Process 2011]. Na Figura 1 (c), pode-se visualizar uma representação da classe *BaseElementInstantiated* na qual apenas são apresentados os atributos adicionados pela extensão proposta, bem como as classes da extensão que originaram cada um deles.

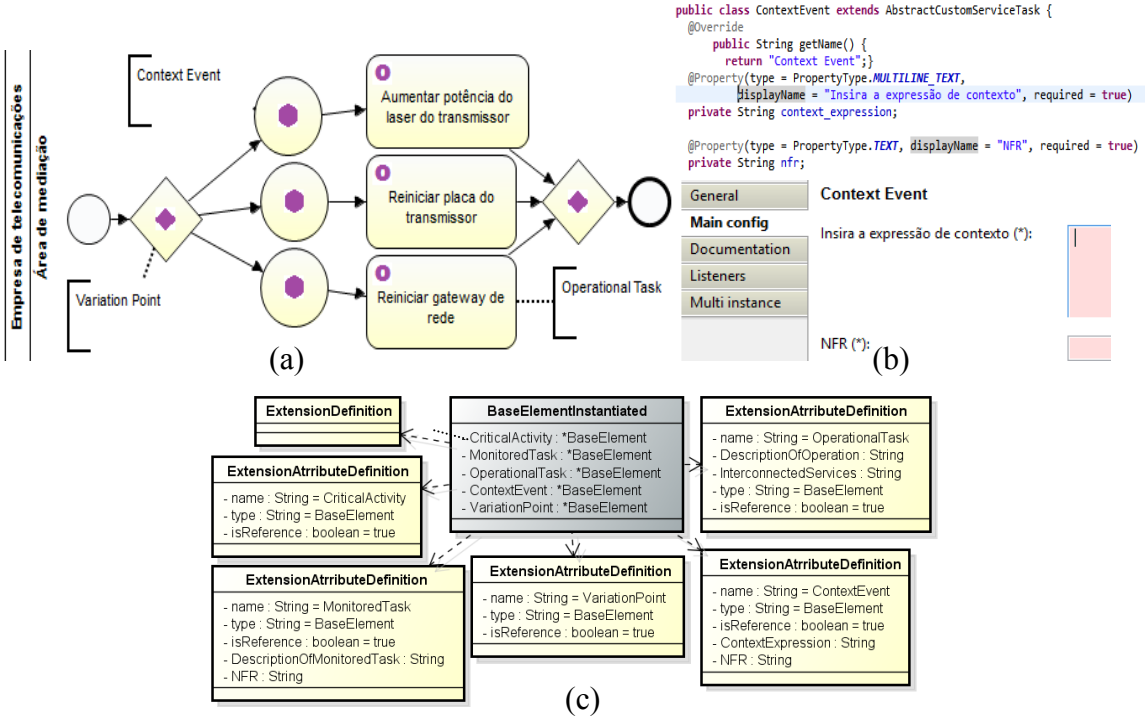

Figura 1. (a) Modelagem utilizando elementos estendidos (b) Implementação do elemento Evento de Contexto (c) Classe *BaseElement* após ser instanciada

Segundo o Activiti para implementar qualquer extensão, é necessário estender a classe *AbstractCustomServiceTask* e configurar os seus parâmetros, que seguem padrões definidos em sua documentação. Todas as propriedades dos elementos estendidos seguiram a representação elaborada na Figura 1 (c), mas por limitações de espaço, apenas um elemento, *Evento de Contexto*, está representado na Figura 1 (b). Na Figura 1 (a) representa uma modelagem do Nível Operacional MABUP elaborada por um dos sujeitos do experimento piloto, que está descrito na seção 6.

5. Contribuições

Dentre as contribuições esperadas com este estudo são: (i) aumentar expressividade de características autonômicas em modelagem de processos utilizando BPMN 2.0; (ii) auxiliar uma melhor compreensão e aplicação da abordagem MABUP, uma vez que os elementos estendidos tratam de forma mais detalhada, as características autonômicas presentes na abordagem. (iii) auxiliar no processo de automação de processos de negócio, de modo que um dos objetivos do estudo final é gerar modelos autonômicos executáveis, com características autonômicas, ou seja, modelos que auxiliem na execução das atividades inerentes ao processo.

6. Estado atual do trabalho

Alguns elementos da notação BPMN 2.0 foram estendidos, englobando três dos quatro níveis da abordagem MABUP, porém, se faz necessário, para contemplar conceitos resentes quarto nível da abordagem MABUP (Nível de Serviço), adaptando o motor de execução do Activiti para suportar os modelos autonômicos e torna-los executáveis. Um experimento piloto foi executado com um grupo de estudantes de graduação e pós-graduação. Este experimento teve como objetivo avaliar o estudo construído até este momento e verificar se os sujeitos eram capazes assimilar, aplicar os elementos estendidos e se o uso deles auxiliaria na compreensão e aplicação da abordagem MABUP. O experimento teve a participação de 15 sujeitos, sendo 9 do grupo Experimental (que recebeu o treinamento da extensão) e 6 do Grupo de Controle (que não recebeu o treinamento). Os alunos do grupo experimental receberam treinamento de aproximadamente 20 minutos dos elementos estendidos para compreender a dinâmica da solução proposta, a fim de ter resultados coerentes para este ensaio. Esta capacitação compreendeu a modelagem de processo de negócio e prática utilizando a técnica BPMN e modelagem de processos de negócio utilizando a abordagem MABUP. Os dois grupos fizeram a modelagem de um problema utilizando a abordagem MABUP, cuja descrição está disponível em https://goo.gl/gz22my, O Grupo de Experimento utilizou os elementos estendidos, já o Grupo de Controle não utilizou os elementos estendidos.

7. Avaliação dos resultados

Os resultados do experimento apontaram que 78% dos participantes concordaram que os elementos estendidos auxiliaram na modelagem de processos autonômicos e na compreensão da abordagem MABUP. Além disso, 67%, concordaram que a utilização de elementos estendidos auxiliou na modelagem de características autonômicas. O tempo médio de cada grupo no experimento foi de: 73 minutos dos participantes que empregaram os elementos estendidos e 77.5 minutos dos participantes que não empregaram elementos estendidos.

Grande parcela dos envolvidos neste experimento inicial concorda que os elementos estendidos auxiliaram na modelagem de características autonômicas de forma significativa. Houve um pequeno ganho no tempo médio de modelagem usando os elementos estendidos, mas não existem subsídios suficientes para afirmar se está relacionado com o auxílio da compreensão da abordagem. Portanto, é possível concluir que os resultados obtidos são promissores uma vez que os dados iniciais indicam que a adoção dos elementos estendidos potencializa o auxílio da compreensão da abordagem MABUP e auxiliam na modelagem de processos autonômicos. Entretanto, deve-se reconhecer que os resultados não são conclusivos, por se tratar de um experimento piloto, conduzido com um pequeno número de participantes, de um estudo em andamento. Portanto, pretende-se realizar outros estudos experimentais com o objetivo de investigar a qualidade dos modelos gerados e apontar precisamente as vantagens e desvantagens de se utilizar os elementos estendidos para representar características autonômicas.

Referencias

Activiti (2015). Activiti BPM Plataform. Disponível em: http://www.activiti.org. Acesso em 05/2015.

Friedenstab, J., et al. (2012). "Extending BPMN for business activity monitoring."System Science (HICSS), 2012 45th Hawaii International Conference on. IEEE, 2012.

Gagne, et al. (2009). "Time-bpmn." Commerce and Enterprise Computing, 2009. CEC'09. IEEE Conference on. IEEE, 2009.

H. Reijers, (2008). “Modularity in process models: Review and effects,” Business Process Management, pp. 20-35.

J. Luis, D. Vara, et al. (2010). “Business Processes Contextualisation via Context Analysis,” Context, pp. 1-6.

Model, Business Process (2011). "Notation (BPMN) Version 2.0." Object Management Group specification.

Oliveira, et al. (2012). "A Multi Level approach to Autonomic Business Process." Software Engineering (SBES), 2012 26th Brazilian Symposium on. IEEE.

Oliveira, et al. (2013). Multi-level Autonomic Business Process Management. In: Business Process Modeling, Development, and Support, 2013, Valencia, Spain. Enterprise, Business-Process and Information Systems Modeling, 2013. v. 147.

Recker, et al. (2006). How good is bpmn really? Insights from theory and practice. In Proceedings 14th European Conference on Information Systems, pages 1582–1593, Goeteborg, Sweden.

S. Espana, (2009). “Communication Analysis: A Requirements Engineering Method for Information Systems,” in Proceedings of the 21st International Conference on Advanced Information Systems Engineering, pp. 530-545.

Santos, Emanuel, et al. (2011). "Using NFR and context to deal with adaptability in business process models." Requirements@ Run. Time (RE@ RunTime), 2011 2nd International Workshop on. IEEE.

T. Cruz, (2008). BPM & BPMS-Business Process Management & Business Process Management Systems. Brasport,.

Valle, R., & Oliveira, S. B. (2012). *Análise e Modelagem de Processos de Negócio.* São Paulo: Atlas.

Detection and Description of Variability Smells

Gustavo Vale, Eduardo Figueiredo (Orientador)

Programa de Pós-Graduação em Ciência da Computação (PPGCC),
Departamento de Ciência da Computação – Universidade Federal de Minas Gerais (UFMG), Belo Horizonte – MG - Brasil

{gustavovale, figueiredo}@dcc.ufmg.br

Nível: Mestrado
Ano de ingresso no programa: Fevereiro de 2014
Época prevista de conclusão: Janeiro de 2016
Data de aprovação da proposta de dissertação: Março de 2015
Eventos relacionados: SBES, SBCARS

Abstract. *Software Product Line (SPL) is a set of software systems that share a common, managed set of features satisfying the specific needs of a particular market segment. The systematic and large reuse adopted in SPLs aim to reduce time-to-market and improve software quality. In spite of the benefits of SPLs and like any software, undesired properties may be present in all related artifacts, such as, source code and feature models. Undesired properties, in a general context, are called bad smells. Bad smells are symptoms that something may be wrong in system design or code. On the context of SPLs, bad smells are called variability smells. Variability smells is a relative new topic of research and need to be further explored. Additionally, variability smells are more useful when it is known to answer the three following questions: What are they? How to detect each of them (detection strategies)? How to solve each of them (refactoring methods)? We performed a Systematic Literature Review (SLR) to answer these three and other questions. As results of the SLR, we found 91 variability smells, detection strategies for 63 variability smells, 106 refactoring methods, and, 23 different SPLs in the context of bad smells and SPLs. The majority of detection strategies are based on logical combination of metrics. For this, it is necessary to know a method to derive software metric thresholds and, what metrics are proposed in the context of SPLs. Therefore, we are performing another SLR related to software metrics in the context of SPLs. We performed a comparison and proposed a method to derive thresholds after this comparison. As future plans, we want to propose variability smell detection strategies more effective than traditional ones and propose some variability smells to fill gaps found in the literature.*

Keywords: *Software Product Lines, Variability Smells, Detection Strategies, Refactoring Methods, Software Metrics, Thresholds*

1. Introduction

Software Product Line (SPL) is a set of software systems that share a common, managed set of features satisfying the specific needs of a particular market segment [Pohl et al., 2005]. The systematic and large-reuse adopted in SPLs aim to reduce time-to-market and improve software quality [Pohl et al., 2005]. The software products derived from an SPL share common features and differ themselves by their specific features [Pohl et al., 2005]. A feature represents an increment in functionality or a system property relevant to some stakeholders [Kastner et al., 2007]. The possible combinations of features to build a product, is called, SPL variability [Weiss and Lai, 1999] and, it can be represented in a feature model [Kang et al., 1990]. A feature model is a formalism to capture and to represent the commonalities and variabilities among the products in an SPL [Asikainen et al., 2006].

To develop an SPL, we can use different approaches, such as annotative and compositional [Apel et al., 2013]. For these approaches, we have several techniques, such as preprocessors, virtual separation of concerns, aspect-oriented programming, and feature-oriented programming. Those approaches and techniques aim to support configuration management at source code level and improve the software quality. In spite of that, undesired properties may be present in all related artifacts, such as source code and feature models [Apel et al., 2013]. Undesired properties, in a general context, are called bad smells. Bad smells are symptoms that something may be wrong in system design or code [Fowler et al., 1999]. On the context of SPLs, bad smells are called variability smells. A variability smell is a perceivable property of a product line that is an indicator of an undesired code property. It may be related to all kinds of artifacts in a product line, including feature models, domain artifacts, feature selections, and derived products [Apel et al., 2013]. Bad smells are extensively studied in single systems context, nevertheless, variability smells is still a young topic [Apel et al., 2013].

Variability smells are more useful when it is known: what they are, how to detect and how to solving them. Metric-based detection strategies and refactoring methods can be used to detect and solving variability smells, respectively [Fowler et al., 1999]. In addition, to use metric-based detection strategies it is necessary to know about metrics and define thresholds for each metric in which composes the detection strategies. This work aims to answer many questions related to these topics. And, for this we defined a strategy based on GQM (goal-question-metric) method [Basili et al., 1994]. In other words, we performed a task-oriented strategy. Figure 1 presents the nine tasks that we done, ongoing, or have to do in the Master degree period.

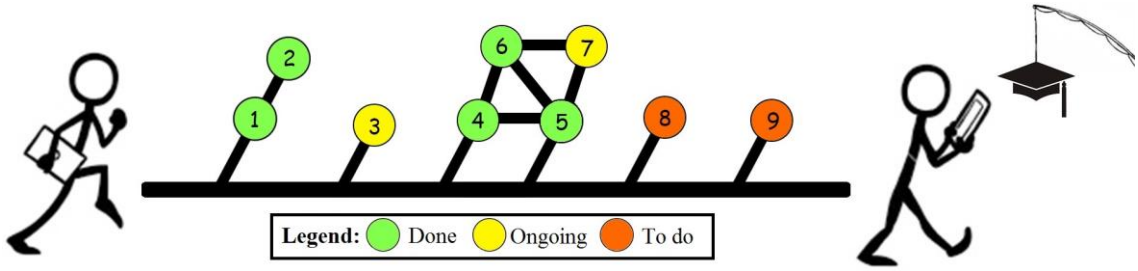

Figure 1. Tasks in Master degree Period

Tasks 1, 2, 3, and 4 are related to literature review and for this reason they appear first. The other tasks are more related to comparisons, evaluations, and proposals of something new. Tasks 1 and 2 provide a Systematic Literature Review (SLR) and an extension to know what are the variability smells, what are the variability smell detection strategies and what are the refactoring methods used to solve or minimize variability smells presented on literature. Based on the literature review, we noted that metric-based detection strategies are more common to detect variability smells. Nevertheless, the effectiveness of metric-based detection strategies is directly dependent on the definition of appropriate thresholds. Hence, we performed a search to find methods to derive thresholds. More than 10 methods were found, but we observed that the methods proposed recently are: (i) systematic, (ii) to consider the *skewed distribution* of the software metric, and, (iii) to receive as input data metrics from a benchmark of software systems. Therefore, we built a benchmark of feature-oriented SPLs (task 4).

Posteriorly, we performed a comparison of three methods to derive thresholds that match the three criteria cited above (task 5). With this comparison, we noted some desirable points in methods to derive thresholds, and as none of the found methods were completely fitted for our purpose, we got the desirable points listed and propose a method to derive thresholds (task 6). As three out of four methods, in spite to be systematic, they do not provide tool support we are developing a tool able to run these four methods to derive thresholds (task 7). At the moment, with the state of art of variability smells, we have plans to explore two points more. These points are related to propose better variability smell detection strategies in terms of effectiveness (task 8) and propose variability smells to fill gaps perceived in the literature (task 9). To achieve the tasks 8 and 9, we believe that it is necessary to know what metrics can be used to express variability mechanisms present in SPLs. For this reason, we are performing a SLR related to metrics proposed in the context of SPLs (task 3).

The rest of this paper is organized as follows. Section 2 presents the methodology and results of tasks that we done or ongoing. Section 3 describes how we pretend to evaluate the two tasks that we have to do. Section 4 presents a brief description of related work. Section 5 highlights the excepted contributions.

2. Literature Review and Thresholds Derivation

We performed a Systematic Literature Review (SLR) – 4 questions – [Vale et al., 2014] and after we extended this SLR – 2 questions more – [Vale et al., 2015a] to find and classify published work about bad smells and refactoring methods in SPLs. Our SLR aimed to answer the following questions: (RQ1) Is the definition of a bad smell the same in the context of SPL and single software systems? (RQ2) What were the SPLs used in studies of bad smells? (RQ3) What are the variability smells already defined and investigated? (RQ4) What are the strategies to identify variability smells? (RQ5) What are the refactoring methods applied to remove variability smells? (RQ6) What refactoring methods were used to minimize or solve an investigated variability smell?

The answers of these questions (task 1 and 2) were based on 22 relevant work found. We could see that bad smells is the general concept. Code and architectural smells are divisions of bad smells (types). Hybrid smells combine architectural and code

smells. Variability smells are bad smells specific to SPLs and can be divided in parts, such as architectural and code smells. Hence, the concept of bad smell is the same for single systems and SPLs (RQ1). We found 23 SPLs (RQ2), we identified 91 variability smells (RQ3), 7 types of variability smell detection strategies for 63 variability smells and 46 of the detection strategies are metric-based detection strategies (RQ4), and 106 refactoring methods (RQ5). Additionally, we matched 31 refactoring methods with 32 variability smells (RQ6).

Related to task 3, we are performing a SLR aiming to provide a catalogue of software metrics proposed in the SPL context. Related to task 4, we searched for feature-oriented SPLs in papers and well-known repositories. In total, we found 64 SPLs to compose our benchmark, in which 38 are different. The step-to-step filtering is further explained on the project website[1]. The benchmarks were built to task 5 in which we performed an evaluation of methods to derive thresholds. In the past few years, thresholds were calculated by software engineers experience and/or using a single system as a reference [Chidamber and Kemerer, 1994; Spinellis et al., 2008]. Recently, this concept has been changing and thresholds have been calculated considering three points [Alves et al., 2010; Ferreira et al., 2012; Oliveira et al., 2014]: (i) well-defined methods (systematics), (ii) methods that consider the skewed distribution of software metrics, and, (iii) derived from benchmarks.

We performed a comparison to find a method to define metric thresholds able to be used in detection strategies (task 5) [Vale et al., 2015b]. For this, we derived the thresholds using three methods [Alves et al., 2010; Ferreira et al., 2012; and Oliveira et al., 2014] found that follows the three points cited above. The thresholds were evaluated individually and by applying the thresholds in a detection strategy proposed to identify God Class in SPLs in terms of recall and precision. God Class is defined as a class that knows or does too much in the software system [Fowler et al. 1999]. As results, none of the methods excelled in the evaluations and we listed eight desirable points in methods to derive thresholds. The desirable points are: (i) to be systematic; (ii) to derive thresholds in a step-wise format; (iii) to be weak dependent with the number of systems; (iv) to be strong dependent with the number of entities; (v) not correlate metrics; (vi) to calculate upper and lower thresholds; (vii) to provide representative thresholds independent of metric distribution and (viii) to provide tool support. These desirable points were used as motivation for propose of a new method (task 6).

The proposed method tries to get the best of each method from the comparison and fits all desirable points, except the tool support which we are developing (task 7). The tool is being designed to run the three methods compared and the proposed method. To evaluate the derived thresholds from the proposed method, we choose a method used (not proposed) to derive thresholds in the context of SPL [Lanza and Marinescu, 2006] (baseline), though it does not consider the skewed distribution of software metrics. For the evaluation we followed similar steps to the previous evaluation, although we added an additional detection strategy to Lazy Class. Lazy Class is defined as a class that knows or does too little in the software system [Fowler et al. 1999]. The results show the proposed method is more effective to detect smells when compared with baseline.

[1] http://labsoft.dcc.ufmg.br/doku.php?id=%20about:spl_list

3. Detecting and Proposing Variability Smells

We aim to propose variability smell detection strategies more efficient than traditional ones (task 8) and new variability smells not yet described in the literature (task 9). For tasks 8 and 9, we think that it is better first to know what are the metrics proposed in the context of SPL and what is the mechanisms those metrics quantify (finish task 3).

Then, for task 8, we want to propose variability smell detection strategies and compare the effectiveness with traditional ones. For the evaluation, we are going to use the built benchmarks of feature-oriented SPLs and the proposed method to derive the thresholds for the metrics that we are going to use. The evaluation is expected to be in terms of effectiveness using recall and precision in target SPLs.

For task 9, we found some gaps in the literature and we use a strategy based on the other authors [Lanza and Marinescu, 2006; Abilio et al., 2015] that proposed bad smells that present at least (i) the description of the proposed variability smells; (ii) which artifact they are applied on (such class and method granularity); (iii) the impact of these variability smells in the SPL; and, (iv) how to detect these smells. Additionally, we want to analyze the source code of some SPLs to see if the undesired properties that we think that are a problem really occur and whether they are problems.

4. Related Work

Several studies can be cited as related with this work. Alves et al. (2010), Ferreira et al. (2012), and, Oliveira et al. (2014), proposed a method to derive thresholds in the context of object-oriented single systems. Differently, our method is designed in the context of SPL. Lanza and Marinescu (2006), Apel et al. (2013), and, Abilio et al. (2015) proposed bad smells. The first one presents 23 bad smells designed to Object-Oriented single system that are reference for the following work. The other studies proposed bad smells for SPLs and are part of our catalogue. Therefore, our work can be considered a continuation of the literature.

5. Intended contributions

As contributions, we expect to have 3 catalogues of bad smells, refactoring methods, and detection strategies for SPLs [Vale et al., 2014; Vale et al., 2015a]. In addition, we have already created a benchmark composed by 64 SPLs, a comparison of methods to derive thresholds [Vale et al., 2015b], and a method proposed for the same purpose [Vale et al., 2015c]. We are currently developing a tool to support four methods to derive threshold, including our own method and a Systematic Mapping related to metrics for SPLs. As future plans up to the end of the dissertation, we want to propose detection strategies more efficient than traditional ones and new variability smells for fill the gap found in feature-oriented software product line literature. Therefore, the dissertation aims to present a technically sound contribution in the three main steps of variability smells solving (description, detection, and refactoring), although the dissertation focusses on the first two steps.

Acknowledgments

This work was partially supported by CAPES, CNPq (grant 485907/2013-5), and FAPEMIG (grant PPM-00382-14).

References

Abilio, R. et al. (2015) “Detecting Code Smells in Software Product Lines - An Exploratory Study”. In: Int'l Conf. on Information Tech.: New Generations (ITNG).

Alves, T.L., Ypma, C., and Visser, J. (2010) “Deriving Metric Thresholds From Benchmark Data”. In: Proc. of 26th Int. Conf. on Soft. Maint. (ICSM), pp. 1–10.

Apel, S., Batory, D., Kastner C., and Saake, G. (2013) “Feature-Oriented Software Product Lines: Concepts and Implementation”. Springer, p.315.

Asikainen, T., Mannisto, T., and Soininen, T. (2006) “A Unified Conceptual Foundation for Feature Modelling”. In: 10th Int'l Soft. Product Line Conf. (SPLC), pp.31-40.

Basili, V., Caldiera, G., Rombach, H. (1994) “Goal Question Metrics Paradigm”. Encyclopedia of Software Engineering, pp. 528-532.

Chidamber, S.R., and Kemerer, C.F. (1994) “A Metrics Suite for Object Oriented Design”. IEEE Transactions on Soft. Eng., vol. 20, Issue 6, pp. 476–493.

Ferreira, K. Bigonha, M., Bigonha, R., Mendes L., and Almeida, H.. (2012) “Identifying Thresholds for Object-Oriented Software Metrics”. J. of Syst. and Soft., pp. 244–257.

Fowler, M., Beck, K., Brant, J., Opdyke, W., and Roberts, D. (1999) “Refactoring: Improving the Design of Existing Code”. Addison-Wesley Professional.

Kang, K. C., Cohen, S. G., Hess, J. A., Novak, W. E., and Peterson, A. S. (1990) “Feature-Oriented Domain Analysis (FODA) - Feasibility Study”. SEE Tech. Report.

Kastner, C., Apel, S. and Batory, D. (2007) “A Case Study Implementing Features Using AspectJ”. In: 11th Int'l Soft. Product Line Conf. (SPLC), pp.223-232, 2007.

Lanza, M., Marinescu, R. (2006) “Object-Oriented Metrics Practice”, Springer, p.205.

Oliveira, P., Valente, M., and Lima, F. (2014) “Extracting Relative Thresholds for Source Code Metrics”. In proc. of CSMR, pp.254-263.

Pohl, K., Bockle, G., and Linden, F. J. V. (2005) “Software Product Line Engineering: Foundations, Principles, and Techniques”. Berlin: Springer, p.490.

Spinellis, D. (2008) “A Tale of Four Kernels”. In Proc. of ICSE, pp. 381–390.

Vale, G. et al. (2014) “Bad Smells in Software Product Lines: A Systematic Review”. In: Brazilian Symp. on Soft. Components, Arch. and Reuse (SBCARS), pp. 84-93.

Vale, G., Abilio, R., Santos, I., Figueiredo, E., Costa, H., and Almeida, E. (2015a) “Bad Smells in Software Product Line: A Systematic Review”. Submitted to: Journal of Systems and Software (JSS).

Vale, G., Albuquerque, D., Figueiredo, E., Garcia, A. (2015b) “Defining Metric Thresholds for Software Product Lines: A Comparative Study”. In: 19th Software Product Line Conference (SPLC).

Vale, G., and Figueiredo, E. (2015c) “A Method to Derive Metric Thresholds for Software Product Lines”. In: 29th Brazilian Symp. on Soft. Eng. (SBES).

Weiss, D. M. and Lai, C. T. R. (1999) “Software Product-Line Engineering: A Family-Based Software Development Process”. Addison-Wesley.

Titulo do trabalho:
Mineração de informações no Ecossistema GitHub para apoiar à Elicitação de Requisitos

Nome:
Roxana Portugal

Nome do(s) orientador(es):
Julio Cesar Sampaio do Prado Leite

Nível (mestrado ou doutorado)
Mestrado

Programa de pós-graduação
Programa de Pós-graduação de Informática da PUC-Rio.

Email de contato do aluno
rportugal@inf.puc-rio.br

Email de contato do(s) orientador(es)
julio@inf.puc-rio.br

Ano de ingresso no programa
2013-2

Época prevista de conclusão
2016-1

Data da aprovação da proposta de tese/dissertação (qualificação)
20-03-2015

Resumo em português ou em inglês
português, inglês

Palavras-chave
engenharia de requisitos, elicitação, fontes de informação, reuso, mineração de textos.

Sigla(s) do(s) evento(s) do CBSoft relacionado(s)(SBES, SBCARS, SBLP, SBMF).
SBES

Mineração de informações no Ecossistema Github para apoiar à Elicitação de Requisitos

Roxana Lisette Quintanilla Portugal, Julio Cesar Sampaio do Prado Leite.

Departamento de Informática – Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio)
R. Marquês de São Vicente, 225 - Gávea, Rio de Janeiro - RJ, 22451-900 – Brasil
rportugal@inf.puc-rio.br, julio@inf.puc-rio.br

Abstract. *This paper describes the work in progress for mining texts in the development ecosystem GitHub. It is explained how the content from similar projects can be useful for reuse and thus assist in the tasks of requirements elicitation. Text-mining techniques and GitHub's metadata are the methods used to select relevant projects and the texts within them. An approach to achieve our proposal is explained and initial results are reported.*

Resumo. *Este artigo descreve o trabalho em andamento para a mineração de textos no ecossistema de desenvolvimento GitHub. É explicado como o conteúdo de projetos semelhantes podem ser úteis para a reutilização e, assim, ajudar nas tarefas de elicitação de requisitos. Técnicas de mineração de textos e metadados de GitHub são os métodos utilizados para selecionar projetos relevantes e os textos dentro deles. Uma abordagem para atingir nossa meta é explicada, bem como descrevemos os resultados alcançados até aqui.*

1. Caracterização do Problema.

Leitura de Documentos é uma técnica de elicitação quando a fonte de informação é um documento (Leite, 2007), assim diferentes técnicas de leitura podem ser aplicadas de acordo com a situação; entre elas a técnica de mineração de textos. No caso desta pesquisa, GitHub, por possuir uma série de documentos (artefatos) produzidos em cada um dos mais de 15 milhões de projetos hospedados nesse ambiente, é uma situação onde aplica-se mineração de texto. Em cada projeto no GitHub existe a possibilidade de encontrar artefatos, tais como: código fonte, casos de teste, protótipos, especificações de requisitos, entre outros. Além disso, existem artefatos particularmente criados a partir da interação no ambiente de rede social do GitHub, tais como: as issues, commits, comments, e readmes entre outros, onde as descrições variam desde notas simples, assuntos ou até mesmo requisitos (Salo, 2014).

Por outro lado, ao utilizar métodos ágeis de desenvolvimento, a procura estratégica de informações (Leite, 2007) e a análise dessas informações de maneira eficaz, é usualmente afetada pelas restrições de tempo dos desenvolvedores. De um modo geral, a leitura de documentos é uma das tarefas que demanda mais tempo do engenheiro de requisitos, e como essa é a maneira de explorar o ambiente GitHub para procurar informação reutilizável sob a ótica de requisitos, a mineração de textos torna-se uma estratégia base para o nosso caso.

Esta pesquisa propõe uma abordagem para minerar informações embutidas em projetos de GitHub. Informações como, por exemplo: características de produtos de

software já desenvolvidos (requisitos funcionais e não funcionais), os propósitos dos produtos (metas), o público objetivo desses produtos (atores). Além disso, uma análise mais sintática como, por exemplo, a identificação de substantivos num texto poderia mostrar informações relevantes como: pessoas, lugares e coisas (entidades). Dessa forma, essas informações coletadas automaticamente poderiam ajudar na eficiência de um processo para elicitar informações de interesse, possibilitando uma visão sobre um determinado domínio existente no GitHub. Assim, espera-se que essa abordagem promova o reuso (Figura 1) de projetos ou partes de um projeto a partir da significância dos trechos minerados para o elicitador.

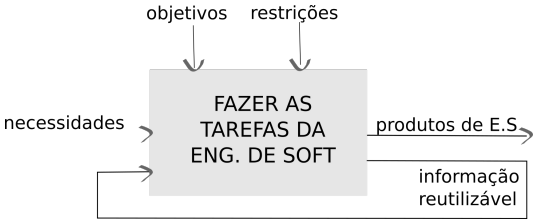

Figura 1. Modelo de Reuso (Leite 2007)

A solução (estratégia) vislumbrada permitirá, através de uma busca por palavras chaves, a seleção de projetos relevantes fazendo uso do API de GitHub. A mineração de textos será apoiada pelas técnicas de mineração de textos na linguagem R (Meyer, 2008). Uma possível organização dos trechos minerados é descrita na Figura 2, abaixo.

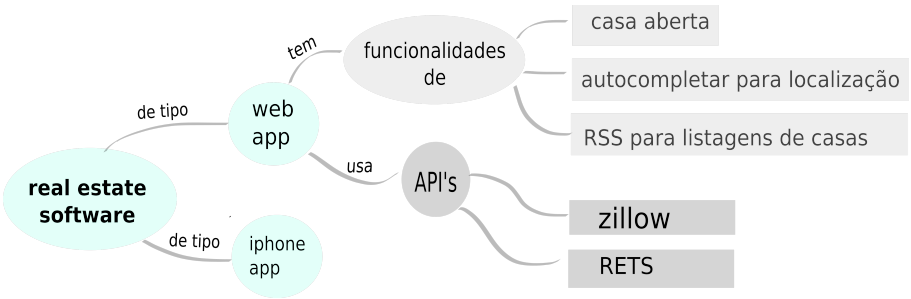

Figura 2. Formatação dos textos minerados.

Tal formatação propõe a utilização de padrões independentes de domínio (Ridao, 2001). Por exemplo, se percebeu que ao descrever cenários, especificamente no caso das histórias de usuário, existe um padrão (Figura 3), tendo como foco os objetivos e a funcionalidade desde o ponto de vista do usuário.

Como um **[ator]** *eu quero/preciso de* **[ação]** *para* **[funcionalidade]**

Figura 3. Padrão independente do domínio.

2. Fundamentação Teórica

O ambiente GitHub vem sendo pesquisado por diversas abordagens, dado que os metadados próprios do GitHub permitem inferir conhecimento de acordo com a intenção do leitor. Assim é o caso do trabalho GiLA (Cánovas, 2015) que permite inferir se um desenvolvedor é perito num tópico com base em sua atividade nas issues de diversos projetos, isto é, analisando tags atribuídas as issues. Outro trabalho, Reviewerrec (Yu, 2014), permite recomendar os usuários que tenham o melhor perfil para avaliar um pull-request, permitindo assim o uso do *crowdsourcing*. Certamente, a

natureza de rede social do GitHub permite também avaliar a reputação de um desenvolvedor dado pela qualidade das suas contribuições (Gousios, 2014).

Com respeito à mineração, fizemos uma revisão de trabalhos que podem nos ajudar na abordagem para minerar o repositório GitHub. Segundo (Meth, 2013) a identificação de requisitos é a atividade de separar os textos que descrevem requisitos de textos que não são relevantes sob a ótica de requisitos. Nesse sentido, vários autores fizeram esforços para descobrir ou minerar dentre a abundância de documentos e para os seguintes fins: desambiguação de requisitos (Osborne, 1996), abstrações de termos e frases para o entendimento do problema (Goldin, 1997), e extração de conhecimento do domínio para ajudar na construção de requisitos (Sawyer, 2005).

3. Contribuições

A principal contribuição deste trabalho é possibilitar o reuso de informações de uma fonte de informação como GitHub com uma enorme quantidade de projetos de software livre e que aumenta constantemente. Essa fonte é comumente utilizada para o reuso de código, seja de caixa branca ou de caixa preta (Ravichandran, 2003). O GitHub poderia, também, nos servir para fazer outro tipo de reuso, como é a utilização de informações de projetos similares para complementar ou criar novos produtos de software. Certamente esse reuso, a nível de requisitos, se tornaria mais relevante quando o tempo para fazer as tarefas de elicitação de requisitos é restrito.

Outra contribuição é promover o reuso de conhecimento do domínio para estimular a criatividade (Fischer, 2005), o que é importante quando o se deseja criar produtos inovadores, como é o caso dos startups de software.

Para alcançar esses resultados estamos usando um estudo de caso como instrumento de investigação exploratória (Ventura, 2007). O caso escolhido, e tratado nesse artigo, é o da Real Estate, que foi selecionado pelo interesse pessoal da pesquisadora em aprender mais sobre este domínio.

4. Estado Atual do Trabalho

Para a pesquisa se definiram três atividades gerais a serem atingidas:

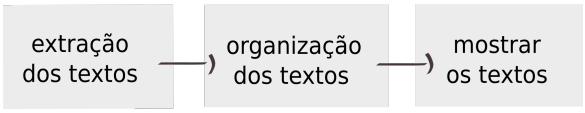

Figura 4. Atividades para a mineração de textos

Existe uma coletânea de estratégias de mineração de texto para a linguagem R que estão sendo adaptadas para o uso específico de nossa pesquisa. Foram identificados 6 passos para o tratamento de esses textos (Meyer, 2008): 1) pré-processar, 2) associar, 3) clusterizar, 4) resumir, 5) categorizar, 6) API para estender o desenvolvido em R.

A atividade de extração de textos se encaixa no passo 1) para o qual teve-se a tarefa inicial de recuperação de textos. É importante aqui mencionar que dentro dos artefatos identificados no GitHub, escolheu-se inicialmente a extração dos readmes por possuir descrições funcionais do projeto. Outro motivo para escolher só um artefato, é que as restrições para requisitar dados utilizando a API do GitHub não permitiriam extração de todos os artefatos de cada projeto.

Para tal recuperação de textos, se desenvolveu um código que seleciona os projetos relevantes de acordo a os termos de busca do usuário, e os parâmetros próprios de GitHub como são, o número de stars e o número de forks atribuído a cada projeto. Cabe ressaltar que o código procura mantendo o *recall* (Berry, 2012) do que seria obtido se fosse uma busca manual. Após a seleção, o código extrai o readme de cada projeto nomeando-o com o nome de usuário e o nome do projeto de origem, de tal forma que se mantém a rastreabilidade para a fonte.

Se fez uma medição para conferir quantos dos readmes coletados efetivamente estavam relacionadas com a noção de Real Estate do caso de estudo. Sem embargo, foram identificados readmes que só continham textos com a palavra "real", outras com a palavra "estate", e outras com a frase "real estate" que é nosso objetivo. A filtragem com técnicas de mineração permitiu melhorar a precisão dos resultados (Figura 5).

Figura 5. Medição da precisão dos readme minerados

5. Passos futuros

Uma vez que a tarefa de recuperação é resolvida, e tendo um conjunto de textos (readmes) a serem processados se prevê continuar com todas as atividades (Figura 4). A exploração dos textos recuperados, usando o caso de investigação exploratória, nos fez perceber o problema da ambiguidade. No caso escolhido, uma busca pelo termo "real estate", resulta tanto em projetos de cunho imobiliário como o de desenho de interfaces. Para atacar o problema de ambiguidade serão utilizadas técnicas de mineração como *k-means* para agrupar readmes segundo o domínio ao qual pertence.

As issues e suas comments serão minerados, pois observou-se que há muitas issues com tarefas a serem desenvolvidas, o que possibilitaria a identificação de funcionalidades.

6. Comparação com Trabalhos Relacionados

As abordagens que vem sendo estudadas nos mostram desafios e também riscos ao manipular documentos em linguagem natural. A ferramenta AbstFinder (Goldin, 1997) difere da nossa pelo fato de eles ter trabalhado num *corpus* de textos bem definidos.

Um segundo trabalho de (Sawyer, 2005) é mais parecido, porem pretende-se usar algumas estratégias apresentadas, mas levando em consideração a questão do ruído presente nos textos que podem entorpecer a mineração. No seu caso, o corpus de textos é definido para o domínio *Air Traffic Control*, o desafio da nossa proposta é encaixar os textos ao domínio que pertencem segundo os termos de busca e usando terminologias e ontologias do domínio para a filtragem de informação.

7. Referencias

Julio Cesar Sampaio do Prado Leite, Livro Vivo: Engenharia de Requisitos, http://livrodeengenhariaderequisitos.blogspot.com/, 2007.

Salo, Risto. "A guideline for requirements management in GitHub with lean approach.". M.Sc.Thesis. University of tampere, School of Information Sciences, 2014.

Julio Cesar Sampaio do Prado Leite, Edson Andrade de Moraes, and Carlos Eduardo Portela Serra de Castro. "A Strategy for Information Source Identification." WER. 2007.

Meyer, David, Kurt Hornik, and Ingo Feinerer. "Text mining infrastructure in R." Journal of Statistical Software 25.5 (2008): 1-54.

Ridao, Marcela, Jorge Doorn, and Julio Cesar Sampaio do Prado Leite. "Domain independent regularities in scenarios." Requirements Engineering, 2001. Proceedings. Fifth IEEE International Symposium on. IEEE, 2001.

Cabot, Jordi, et al. "Exploring the use of labels to categorize issues in Open-Source Software projects." Software Analysis, Evolution and Reengineering (SANER), 2015 IEEE 22nd International Conference on. IEEE, 2015.

Canovas Izquierdo, Javier Luis, et al. "GiLA: GitHub label analyzer." Software Analysis, Evolution and Reengineering (SANER), 2015 IEEE 22nd International Conference on. IEEE, 2015.

Yu, Y., Wang, H., Yin, G., & Ling, C. X. (2014, September). Reviewer Recommender of Pull-Requests in GitHub. In Software Maintenance and Evolution (ICSME), 2014 IEEE International Conference on (pp. 609-612). IEEE.

Gousios, Georgios, Martin Pinzger, and Arie van Deursen. "An exploratory study of the pull-based software development model." Proceedings of the 36th International Conference on Software Engineering. ACM, 2014.

Meth, Hendrik. "A Design Theory for Requirements Mining Systems.". Ph.D. Thesis. Universität Mannheim, 2013.

Osborne, Miles, and C. K. MacNish. "Processing natural language software requirement specifications." Requirements Engineering, 1996., Proceedings of the Second International Conference on. IEEE, 1996.

Goldin, Leah, and Daniel M. Berry. "AbstFinder, a prototype natural language text abstraction finder for use in requirements elicitation." Automated Software Engineering 4.4 (1997): 375-412.

Sawyer, Peter, Paul Rayson, and Ken Cosh. "Shallow knowledge as an aid to deep understanding in early phase requirements engineering." Software Engineering, IEEE Transactions on 31.11 (2005): 969-981.

Ravichandran, Thiagarajan, and Marcus A. Rothenberger. "Software reuse strategies and component markets." Communications of the ACM 46.8 (2003): 109-114.

Fischer, Gerhard, et al. "Beyond binary choices: Integrating individual and social creativity." International Journal of Human-Computer Studies 63.4 (2005): 482-512.

Ventura, Magda Maria. "O estudo de caso como modalidade de pesquisa." Rev Socerj 20.5 (2007): 383-386.

Berry, Daniel, et al. "The case for dumb requirements engineering tools." Requirements Engineering: Foundation for Software Quality. Springer Berlin Heidelberg, 2012. 211-217.

Avaliação Experimental de Abordagens para a Geração Automática de Dados de Teste para Programas Concorrentes

Ricardo Ferreira Vilela
ricardovilela@usp.br

Orientadora: Profa. Dra. Simone do Rócio Senger de Souza
srocio@icmc.usp.br

Mestrado

Instituto de Ciências Matemáticas e de Computação – ICMC/USP
(Programa de Pós-graduação em Ciências de Computação e Matemática Computacional)

Ano de ingresso: 2014

Conclusão: Março/2016

Data de aprovação da qualificação: 27/03/2015

Resumo: Este projeto de mestrado tem o objetivo definir e aplicar um estudo experimental visando avaliar diferentes técnicas de geração de dados de teste analisando sua aplicabilidade para o contexto de programas concorrentes. O objetivo deste estudo é encontrar evidências sobre o custo das técnicas para geração de dados de teste, a eficácia em revelar defeitos dos conjuntos de teste gerados e a qualidade desses conjuntos de teste, avaliando esses aspectos para programas concorrentes, visto que não existe ainda estudo realizado nesta direção.

Palavras-chave: teste de software; geração automática de dados de teste; teste de software baseado em busca; teste de programas concorrentes.

SBES

USP - São Carlos
Agosto/2015

Avaliação Experimental de Abordagens para a Geração Automática de Dados de Teste para Programas Concorrentes

Ricardo F. Vilela[1], **Simone R. S. de Souza**[1], **Paulo S. L. Souza**[1]

Instituto de Ciências Matemáticas e de Computação (ICMC/USP)
São Carlos/SP, Brazil, 13560-970

{ricardovilela@usp.br, srocio.icmc.usp.br, pssouza@icmc.usp.br}

Abstract. *The objective of this master's project is the definition and execution of an experimental study to evaluate different techniques of test data generation, analyzing their applicability to the context of concurrent programs . The aim of this study is to find evidence about the cost of these techniques the quality and effectiveness in revealing defects of the generated test sets, assessing these aspects for concurrent programs, since there is still no study done in this direction.*

Resumo. *Este projeto de mestrado tem o objetivo definir e aplicar um estudo experimental visando avaliar diferentes técnicas de geração de dados de teste analisando sua aplicabilidade para o contexto de programas concorrentes. O objetivo deste estudo é encontrar evidências sobre o custo das técnicas para geração de dados de teste, a eficácia em revelar defeitos dos conjuntos de teste gerados e a qualidade desses conjuntos de teste, avaliando esses aspectos para programas concorrentes, visto que não existe ainda estudo realizado nesta direção.*

1. Introdução

O teste de software é uma atividade dinâmica de verificação, ou seja, para testar uma aplicação é necessária a execução do programa ou modelo. A principal atribuição do teste de software é fornecer entradas (dados de teste) e analisar se o comportamento do programa está de acordo com sua especificação. No entanto, para que um programa seja devidamente testado é necessário que o testador ou ferramenta de teste forneça entradas específicas que possuam grandes chances de revelar defeitos. O ideal para o teste de um programa seria executar todas as suas possíveis entradas, entretanto, devido a cardinalidade do domínio de entrada das aplicações essa prática é inviável na maioria dos casos.

Para auxiliar a seleção de entradas do programa foram estabelecidos os critérios de teste. Os critérios podem ser considerados como regras que definem elementos que devem ser exercitados (elementos requeridos) durante o teste de software. Na literatura existem diferentes critérios de teste, os quais são classificados em diferentes técnicas, sendo as mais conhecidas: técnica funcional, estrutural e baseada em defeitos. A diferença entre as técnicas está na informação de origem usada para gerar os elementos requeridos. Neste trabalho são considerados apenas os critérios estruturais de teste, mais detalhes sobre os critérios funcionais e baseados em defeitos podem ser encontrados em [Myers et al. 2011, Delamaro et al. 2007].

Os critérios estruturais utilizam aspectos estruturais do código-fonte para orientar a seleção de casos de teste exigindo a execução de partes ou de componentes elementares do programa [Myers et al. 2011]. A ideia central no teste estrutural é selecionar entradas capazes de exercitar caminhos, comandos e condições no programa.

Na maioria dos casos a determinação de quais entradas selecionar para o teste de aplicações é realizada manualmente, com base no conhecimento do software e experiência do testador. Embora a automatização dessa atividade (seleção de dados de teste) seja muito desejada, não existe um algoritmo de propósito geral e eficiente que seja capaz de gerar automaticamente e a baixo custo um conjunto de teste efetivo. Nem mesmo é possível determinar automaticamente se esse conjunto existe [Delamaro et al. 2007].

Apesar de existirem limitações, algumas técnicas têm sido propostas para geração automática de dados de teste, as quais, em geral, são guiadas por critérios de teste [Delamaro et al. 2007, McMinn 2004, Koirala and Sheikh 2009]. Umas das técnicas mais simples e baratas é a geração aleatória, a qual tem como objetivo selecionar pontos específicos do domínio de entrada de forma aleatória até que o critério de teste seja satisfeito (ou coberto) [Chen et al. 2005]. O problema principal dessa técnica é que ela não garante que trechos importantes do código serão executados.

Uma técnica que tem sido extensamente explorada para a geração de dados de teste é meta-heurística. Essa técnica busca resolver de forma genérica problemas que não possuem um algoritmo eficiente para a sua solução e possuem um espaço de soluções extenso. Algumas técnicas de meta-heurística conhecidas são: algoritmos genéticos, *simulated annealing*, busca tabu, colônia de formigas e colônia de abelhas [McMinn 2004]. Na literatura existem diversos trabalhos que utilizam técnicas meta-heurística para geração de dados de teste, os quais são propostos essencialmente para programas com características sequenciais.

Diferentemente dos programas sequenciais, a computação distribuída envolve processos concorrentes que interagem para realizar as tarefas. Essa interação pode ocorrer de forma sincronizada ou não, sendo que esses processos podem ou não concorrer pelos mesmos recursos computacionais. Esse tipo de computação vem sendo cada vez mais empregada e necessária, haja visto as tecnologias atuais, com processadores com múltiplos núcleos e com o uso crescente de *clusters* de computadores. É comum o uso de programação concorrente para reduzir o tempo computacional em vários domínios, tais como: previsão de tempo, dinâmica de fluídos, processamento de imagens, química quântica, dentre outros.

2. Caracterização do Problema

A atividade de teste no contexto de programas concorrentes é considerada mais complexa. Além das dificuldades inerentes à atividade de teste, outras ocorrem devido, principalmente, ao comportamento não determinístico dessas aplicações. Múltiplas execuções de um programa concorrente com a mesma entrada podem executar diferentes sequências de sincronização e podem produzir diferentes resultados. Cabe a atividade de teste, nesse cenário, identificar se todas as sequências de sincronização possíveis foram executadas e se as saídas obtidas estão corretas [Souza et al. 2014].

Desse modo, a geração automática de dados de teste pode contribuir efetivamente para o teste de aplicações concorrentes, diminuindo, significativamente o esforço ne-

cessário para o teste e melhorando sua qualidade. No entanto, os desafios descritos anteriormente provenientes desse tipo de aplicação podem dificultar a realização dessa tarefa. Neste caso, um importante tópico a ser investigado são os benefícios que as técnicas existentes de geração automática de dados de teste podem fornecer ao teste de aplicações concorrentes.

Neste cenário, a experimentação pode contribuir para a construção de uma base de conhecimento confiável e reduzir assim as incertezas sobre quais teorias, ferramentas e metodologias são adequadas. Por meio de um método experimental é possível obter modelos sugeridos que auxiliam o processo de experimentação, desenvolver um método qualitativo e/ou quantitativo e conduzir um experimento medindo, analisando e avaliando os seus resultados [Shull et al. 2004].

Considerando o contexto acima, o objetivo deste projeto de mestrado é a definição e a execução de um estudo experimental visando avaliar diferentes técnicas de geração de dados de teste analisando sua aplicabilidade para o contexto de programas concorrentes. O objetivo deste estudo é encontrar evidências sobre o custo das técnicas para geração de dados de teste, a eficácia em revelar defeitos dos conjuntos de teste gerados e a qualidade desses conjuntos de teste, avaliando esses aspectos para programas concorrentes, visto que não existe ainda estudo realizado nesta direção.

3. Trabalhos Relacionados

Nenhuma técnica de geração de dados de teste pode ser considerada superior as demais técnicas, pois cada técnica apresenta características diferentes que se enquadram melhor em problemas específicos. Desta forma, diversas técnicas devem ser comparadas quando surgem novos problemas [Mohi-aldeen et al. 2014]. Nesse sentido, Ahmed e Zamli (2011) apresentam uma comparação de meta-heurísticas para geração de dados de teste baseado no critério de cobertura para interação de elementos. O objetivo do trabalho foi descobrir o quão bem os elementos de interação são cobertos pelos algoritmos meta-heurísticas existentes. Para isso, a medida de cobertura dos elementos de interação foi baseada nos dados de teste gerados pelas meta-heurísticas [Ahmed and Zamli 2011]. As técnicas analisadas foram: *Hill Climbing*, *Simulated Annealing*, Busca Tabu e *Particle Swarm Optimization* - PSO. O resultado obtido aponta que para 25 ou menos dados de teste o algoritmo *Simulated Annealing* apresenta melhor cobertura dos elementos de interação. No entanto, o PSO produz melhor cobertura para as demais quantidades de dados de teste com um conjunto menor de teste.

No contexto de programas concorrentes, poucos trabalhos exploram a proposição de técnicas para geração automática de dados de teste. Os trabalhos [Tian and Gong 2014a, Silva et al. 2014, Tian and Gong 2014b] apresentam soluções que exploram algoritmos genéticos (AG) para geração de dados com esse domínio de aplicação, porém o alto custo e a falta de uma avaliação experimental efetiva não indica que AG seja a melhor opção.

4. Caracterização da Contribuição

A questão de pesquisa que este projeto de mestrado pretende responder é: "*Quais os benefícios e limitações das técnicas existentes para geração automática de dados de teste aplicadas para programas concorrentes?*" Esta questão de pesquisa será explorada por

meio da realização de estudos experimentais os quais irão avaliar diferentes técnicas de geração automática de dados de teste para o contexto de programas concorrentes.

O primeiro passo a ser realizado para condução deste estudo é um levantamento sobre as técnicas existentes para geração de dados de teste. O levantamento dessas técnicas será realizado recorrendo a buscas na literatura por trabalhos que apresentem diferentes técnicas de geração. Outro levantamento que será realizado é o de ferramentas que implementem as técnicas catalogadas para a geração de dados de teste. O intuito nesta etapa é realizar um levantamento das ferramentas existentes de geração, as quais serão avaliadas durante o estudo. Dessa forma, os resultados obtidos são importantes para definir qual será o conjunto de técnicas de geração de dados de teste selecionado para este estudo.

Na terceira etapa do estudo será desenvolvido um protocolo experimental que deve guiar a condução do experimento. Na concepção do protocolo experimental todas as fases do experimento serão definidas, como exemplo: os critérios de teste, ferramentas de apoio e *benchmarks* para o teste.

Com a definição do protocolo experimental serão realizadas as etapas nele definidas. Nesse momento serão reunidas todas as ferramentas de geração selecionadas, as quais serão executadas em conjunto com uma ferramenta de apoio ao teste. A ferramenta de apoio deve ser capaz de: implementar os critérios definidos no experimento, gerar elementos requeridos com base nos critérios selecionados, executar os dados de teste fornecidos pelas ferramentas de geração e fornecer os resultados obtidos com base nos conjuntos de teste utilizados.

Como objetos de teste serão consideradas aplicações concorrentes que utilizam os modelos de comunicação de passagem de mensagem e memória compartilhada desenvolvidas utilizando a linguagem Java. A definição desses modelos é fundamentada na diversidade de aplicações que utilizam esses modelos para comunicação entre processos ou *threads*.

Após a condução do experimento será realizada uma avaliação dos dados obtidos por meio de análises estatísticas. Dessa forma, o pesquisador deste trabalho investigará os resultados obtidos no intuito de responder as questões de pesquisa investigadas.

5. Estado Atual do Trabalho

A etapa atual deste projeto, consiste no levantamento de ferramentas de geração de dados de teste, o qual é guiado por buscas na literatura. Além disso, também está sendo realizado um experimento piloto do projeto, o qual tem como objetivo auxiliar a definição do protocolo experimental. Para este piloto considera-se uma pequena quantidade de ferramentas (já identificadas), critérios concorrentes e sequenciais, *benchmarks* de teste (já selecionados) e a ferramenta de apoio ao teste de programas concorrente ValiPar.

6. Avaliação dos Resultados

Para avaliação dos resultados obtidos pelas técnicas de geração foram estabelecidas as seguintes métricas: **M1**) avaliar com base em mais de um conjunto de teste gerado, o custo em relação à quantidade de dados de teste necessários para satisfazer os requisitos dos critérios de teste considerados; **M2**) avaliar o custo em relação ao tempo necessário

para geração dos dados de teste; **M3**) avaliar a qualidade dos conjuntos de teste em relação a diferentes critérios de teste, tanto para aspecto sequencial, como para concorrente; e **M4**) avaliar a habilidade dos conjuntos gerados em revelar defeitos injetados propositalmente nos programas concorrentes.

É importante ressaltar que, para cada técnica serão consideradas diferentes configurações, por exemplo, quantidade de repetições, complexidade dos algoritmos sob teste e critérios selecionados.

Referências

Ahmed, B. and Zamli, K. (2011). Comparison of metahuristic test generation strategies based on interaction elements coverage criterion. In *IEEE Symposium on Industrial Electronics and Applications*, pages 550–554.

Chen, T., Leung, H., and Mak, I. (2005). Adaptive random testing. In Maher, M., editor, *Advances in Computer Science - ASIAN 2004. Higher-Level Decision Making*, Lecture Notes in Computer Science. Springer Berlin Heidelberg.

Delamaro, M., Maldonado, J., and Jino, M. (2007). *Introdução ao teste de software*. Elsevier.

Koirala, S. and Sheikh, S. (2009). *Software Testing Interview Questions*. Computer science series. Ebsco Publishing.

McMinn, P. (2004). Search-based software test data generation: A survey: Research articles. *Softw. Test. Verif. Reliab.*, 14(2):105–156.

Mohi-aldeen, S. M., Deris, S., and Mohamad, R. (2014). Systematic mapping study in automatic test case generation. In *New Trends in Software Methodologies, Tools and Techniques*, Frontiers in Artificial Intelligence and Applications. IOS Press Ebooks.

Myers, G. J., Sandler, C., and Badgett, T. (2011). *The Art of Software Testing*. Wiley Publishing, 3rd edition.

Shull, F., Mendonça, M. G., Basili, V., Carver, J., Maldonado, J. C., Fabbri, S., Travassos, G. H., and Ferreira, M. C. (2004). Knowledge-sharing issues in experimental software engineering. *Empirical Software Engineering*, 9(1-2):111–137.

Silva, J. D. P., Souza, S. R., and Souza, P. S. L. (2014). Geração automática de dados de teste para programas concorrentes com uso de meta-heurísticas. In *8th Brazilian Workshop on Systematic and Automated Software Testing (SAST)*, pages 71–80.

Souza, P. S., Souza, S. R., and Zaluska, E. (2014). Structural testing for message-passing concurrent programs: an extended test model. *Concurrency and Computation: Practice and Experience*, 26(1):21–50.

Tian, T. and Gong, D. (2014a). Evolutionary generation approach of test data for multiple paths coverage of message-passing parallel programs. *Chinese Journal of Electronics*.

Tian, T. and Gong, D. (2014b). Test data generation for path coverage of message-passing parallel programs based on co-evolutionary genetic algorithms. *Automated Software Engineering*.

Avaliação da qualidade de oráculos de teste utilizando mutação

Ana Claudia Maciel - ***anamaciel@usp.br***

Orientador: Prof. Dr. Márcio Delamaro - ***delamaro@icmc.usp.br***

Mestrado

Instituto de Ciências Matemáticas e de Computação — ICMC/USP (Programa de Ciências de Computação e Matemática Computacional)

Ano de ingresso: 2014

Conclusão: Fevereiro/2016

Data aprovação da qualificação: 27/03/2015

Resumo: No desenvolvimento de software, a qualidade do produto está diretamente relacionada à qualidade do processo de desenvolvimento. Diante disso, atividades de Verificação, Validação & Teste (VV&T) realizadas por meio de métodos, técnicas e ferramentas são necessárias para o aumento da produtividade, qualidade e diminuição de custos no desenvolvimento de software. Do mesmo modo, técnicas e critérios contribuem para a produtividade das atividades de teste. Um ponto crucial para o teste de software é sua automatização, tornando as atividades mais confiáveis e diminuindo significativamente os custos de desenvolvimento. Para a automatização de atividades de testes, os oráculos são essenciais, representando um mecanismo (programa, processo ou dados) que indica se a saída obtida para um caso de teste está correta. Este projeto de mestrado utiliza a ideia de mutação para criar implementações alternativas de oráculos e assim avaliar a sua qualidade. O teste de mutação se refere à criação de versões do sistema em desenvolvimento com pequenas alterações sintáticas de código. A mutação possui alta eficácia na detecção de defeitos e é bastante flexível na sua aplicação, podendo ser utilizado em diversos tipos de artefatos. O projeto visa também à proposição de operadores de mutação específicos para oráculos, desenvolvimento de uma ferramenta de suporte à utilização dos operadores para oráculos, e também avaliação empírica dos operadores, destacando benefícios e desafios associados ao seu uso.

Palavras-chave: oráculos de teste; teste de mutação; operadores de mutação; teste de software.

SBES

USP - São Carlos
Julho/2015

Avaliação da qualidade de oráculos de teste utilizando mutação

Ana Claudia Maciel[1], **Márcio Delamaro**[1]

[1]Instituto de Ciências Matemáticas e de Computação – Universidade de São Paulo (USP)
Avenida Trabalhador São-Carlense, 400 - Centro CEP: 13566-590 - São Carlos - SP

anamaciel@usp.br, delamaro@icmc.usp.br

1. Caracterização do Problema

Teste de software é o processo de execução de um produto de software para determinar se ele atinge suas especificações e funciona corretamente no ambiente para o qual foi projetado [Myers et al. 2004]. Dentro do contexto de teste, um oráculo automatizado é a fonte mais precisa sobre os resultados esperados de uma determinada execução do sistema em teste [Oliveira et al. 2009].

Apesar da importância dos mecanismos dos oráculos, não existe uma forma sistematizada de se avaliar sua qualidade. Algo análogo aos critérios de teste para oráculos, que indique se o oráculo se comporta como o testador espera. As avaliações de oráculos são geralmente feitas em domínios particulares e não podem ser generalizadas para diferentes tipos de oráculos. Ainda que a implementação e avaliação de oráculos ainda esteja longe de ser sistemática [Shrestha and Rutherford 2011], o número de publicações sobre oráculos tem aumentado e estratégias de oráculos vêm sendo altamente discutidas em congressos de teste [Harman et al. 2013].

Nesse contexto, o objetivo deste trabalho é a definição de critérios de avaliação da qualidade de oráculos especificamente no contexto de testes de unidade, descritos no formato do *framework* JUnit. Pretende-se utilizar para tanto uma abordagem baseada em conceitos de mutação, semelhante à que se utiliza para programas convencionais. Desse modo, serão criados operadores de mutação específicos para oráculos de teste.

2. Fundamentação Teórica

Nesta seção são apresentados os conceitos relevantes para o entendimento do projeto de mestrado em andamento. São abordados conceitos relacionados a oráculos de teste, teste de mutação e ao *framework* JUnit.

2.1. Oráculos de Teste

Um oráculo de teste é um mecanismo capaz de decidir se uma execução está correta ou não, ou seja, se apresenta ou não os resultados esperados [Weyuker 1982]. Entretanto, não é uma tarefa trivial determinar para um programa, um conjunto de saídas esperadas [Nardi 2013]. Muitas vezes o papel de oráculo é desempenhado pelo próprio testador que deve consultar a especificação e dar o veredito sobre a correção da execução. Para diminuir esforços, os testadores devem automatizar os oráculos [Nunes et al. 2009].

O problema de oráculo está definido nos casos em que, utilizando meios práticos, é impossível ou muito difícil decidir sobre a correção de casos de teste e saídas de teste [Weyuker 1982]. A decisão sobre a correção de uma execução e, consequentemente,

revelações de falhas é o aspecto mais essencial relacionado a qualquer atividade de teste, mesmo quando realizada de forma manual [Staats et al. 2011]. Dependendo do tipo de oráculo, podem ocorrer os seguintes problemas:

- Falsos positivos: casos em que o resultado do oráculo apresenta falha enquanto o SUT está funcionando adequadamente; e
- Falsos negativos: o resultado do teste não apresenta inconsistência. No entanto, alguma falha pode não ter sido verificada no processo de teste.

A qualidade dos oráculos pode ser medida levando em conta diversas características, de acordo com a precisão dos resultados, ou seja, qual a porcentagem de resultados esperados foram gerados pelo oráculo. Outro fator é qual o montante de falsos positivos o oráculo produz, ou quantas falhas, são identificadas. Os critérios de avaliação para oráculos são particulares dentro do contexto de cada trabalho.

2.2. Teste de Mutação

O Teste de Mutação [DeMillo et al. 1978] ou Análise de Mutantes é um critério de teste da técnica baseada em erros. O programa que está sendo testado é alterado diversas vezes, criando-se um conjunto de versões alternativas com alterações sintáticas, chamadas mutantes. O trabalho do testador é selecionar casos de teste que identifiquem a diferença de comportamento entre o programa original e os programas mutantes [Delamaro et al. 2007], "matando" os mutantes. Neste projeto de mestrado pretende-se estender a ideia do teste de mutação, aplicando-a em um novo contexto, ou seja, na avaliação da qualidade de oráculos de teste escritos na forma de classes JUnit.

2.3. *Framework* JUnit

A utilização do JUnit facilita a criação de casos de teste, pelo fato de automatizar os testes de software, evitando escrever testes duplicados e permite escrever testes que retêm seu valor ao longo do tempo. Tecnicamente, o JUnit implementa a base para a automatização dos oráculos por meio das suas classes de teste e os comandos `Assert`. Um conjunto de teste no JUnit consiste de programas escritos em Java, com funcionalidades específicas para definir as saídas esperadas. Assim, é possível utilizar técnicas de teste empregadas para programas convencionais, nesses oráculos.

3. Contribuições

A proposta de pesquisa do mestrado é utilizar conceitos de mutação para criar implementações alternativas do procedimento de oráculo e verificar se elas apresentam os mesmos resultados do oráculo original. Caso isso aconteça, o testador deve analisar a situação pois dispõe de dois oráculos distintos para o mesmo caso de teste e deve decidir qual deles é o mais adequado.

Para a execução desse projeto é necessário que sejam desenvolvidos operadores de mutação coerentes. Por exemplo, em um conjunto de teste em que todos os casos de teste são executados com sucesso, se forem trocadas as implementações de oráculo por outras que sempre aceitam o resultado produzido, serão obtidos diversos oráculos mutantes vivos, mas que em nada contribuem para a avaliação do oráculo original.

No exemplo apresentado na Figura 1 uma classe `MyMath` está sendo testada. O caso de teste apresentado verifica se o resultado de $\sqrt{9} = 3$, com um possível erro de 10^{-8},

chamando o método `MyMath.raizQuadrada`. O método `assertEquals` faz o papel do procedimento de oráculo, especificando que o resultado obtido deve ser igual a 3.0. As duas implementações apresentadas na Figura 1 podem apresentar o mesmo resultado do caso de teste original. Cabe ao testador analisar as duas implementações e decidir qual seria a mais adequada e ainda, responder a questão: em qual situação as implementações se comportariam de forma distinta e se o oráculo de teste estaria sendo exercitado em tal situação. Em outras palavras, a mutação do oráculo poderia ainda sugerir novos casos de teste para o programa, contribuindo para a qualidade do produto final.

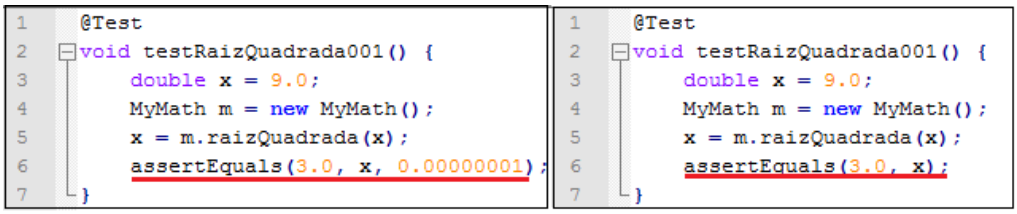

Figura 1. Exemplo de mutação na classe JUnit.

Uma vez definidos operadores de mutação para oráculos, eles serão implementados e adicionados à ferramenta MuJava[1]. Os mutantes equivalentes devem ser identificados manualmente pelo testador na ferramenta.

4. Estado Atual do Trabalho

O cronograma de trabalho foi dividido em 5 etapas:

1. **Definição e desenvolvimento da técnica:** operadores de mutação específicos para oráculos baseados em assertivas foram definidos para posteriormente utilizá-los na avaliação da qualidade dos oráculos de teste;
2. **Implementação dos operadores:** os operadores de mutação já definidos serão incluídos na ferramenta de mutação já existente, MuJava. O trabalho, atualmente, está nesta fase de execução;
3. **Validação:** pretende-se utilizar programas e casos de testes pilotos que utilizem a abordagem desenvolvida, para que haja a avaliação da qualidade dos oráculos de teste, objetivo principal do trabalho;
4. **Disseminação:** os resultados do trabalho desenvolvido durante o mestrado devem ser publicados em forma de artigos científicos em conferências, *workshops* e eventos da área. Esta etapa do trabalho está sendo realizada em paralelo às demais etapas;
5. **Escrita da dissertação e defesa do mestrado:** a dissertação terá como base o texto já escrito para a qualificação e artigos que serão desenvolvidos durante a realização do mestrado.

A etapa de definição e desenvolvimento da técnica já foi realizada. O trabalho está na fase de implementação dos operadores e disseminação dos resultados. Logo, os trabalhos posteriores serão a validação da técnica desenvolvida no projeto e a escrita da dissertação e defesa do mestrado.

[1]https://cs.gmu.edu/ offutt/mujava/

5. Descrição e Avaliação dos Resultados

Foram propostos operadores de mutação genéricos para introduzir pequenas modificações nos tipos mais comuns das assertivas da ferramenta de teste unitário, JUnit, onde as variações e assinaturas das assertivas foram derivadas, adicionando, removendo, alterando ou substituindo alguns valores de definição. Para automatizar e sistematizar a avaliação dos oráculos de teste, foram definidas quatro classes de operadores: (i) Adição; (ii) Remoção; (iii) Alteração; e (iv) Substituição. E são classificados em dois níveis: (i) Nível de assinatura em que as alterações são feitas no tipo do método `assert` em execução, ou nos parâmetros recebidos pelo método `assert`; e (ii) Nível de anotação em que as alterações são aplicadas substituindo as anotações, removendo, ou alterando seus parâmetros. Na Tabela 1 são apresentados os operadores com suas respectivas siglas. Esses operadores estão em fase de implementação.

Nível de Assinatura			
#	Classe	Descrição	Sigla
1	Adição	Adicionar mensagem (String)	ASM
2	Adição	Adicionar valor constante	ATV
3	Remoção	Remover mensagem (String)	RSM
4	Remoção	Remover o método fail().	RFM
5	Remoção	Remover valor constante	RTV
6	Alteração	Modificar parâmetro de tipo primitivo	MPPT
7	Alteração	Modificar parâmetro de tipo primitivo para objeto	MPPTO
8	Alteração	Incrementar valor constante	ICtTV
9	Alteração	Decrementar valor constante	DCfTV
10	Alteração	Modificar mensagem (String)	MSM
11	Substituição	Substituir assertiva booleana	RBA
12	Substituição	Substituir assertiva nula	RNA
13	Substituição	Substituir assertiva de igualdade	RSA

Nível de Anotação			
#	Classe	Descrição	Sigla
1	Adição	Adicionar classe esperada	AEC
2	Remoção	Remover classe esperada	REC
3	Remoção	Remover timeout	RT
4	Remoção	Remover anotação Ignore	RIA
5	Alteração	Modificar classe esperada	MEC
6	Alteração	Incrementar constante no *timeout*	ACtT
7	Alteração	Decrementar constante no *timeout*	DCfT

Tabela 1. Tabela de operadores (Nível de assinatura e anotação).

Os operadores serão incluídos na MuJava, portanto, seguirão a mesma estrutura de código já existente na ferramenta. Para isso, o pacote `Util` será utilizado, pelo fato de ele implementar classes básicas para execução dos operadores implementados na ferramenta. Cada operador possui uma classe para gerar as alterações sintáticas nos códigos originais e uma classe para a geração dos arquivos dos programas mutantes.

6. Comparação com Trabalhos Relacionados

Poucos trabalhos trazem avaliação de oráculos de teste. Na literatura, encontram-se trabalhos, em sua maioria, relacionados à automatização de oráculos de teste. Os critérios de avaliação para oráculos são particulares ao contexto de cada trabalho, prejudicando o compartilhamento do conhecimento.

Trabalhos como [Shrestha and Rutherford 2011], [Tan and Edwards 2004] e [Shahamiri et al. 2011] avaliam a qualidade de oráculos de teste por meio da aplicação da mutação nos programas em teste. O diferencial deste projeto de mestrado é que os conceitos de mutação serão aplicados diretamente nos oráculos, gerando uma forma sistematizada de avaliar os oráculos baseados em assertivas escritos no formato dos frameworks JUnit.

Os oráculos são desenvolvidos de forma *ad hoc*, ainda não há forma sistemática de implementá-los. De modo consequente, estratégias de avaliação de oráculos de teste

para domínios específicos são pouco exploradas, levando à ocorrência um negligenciamento dos autores ao desenvolver tais estratégias. É importante ressaltar que a proposta deste trabalho é inédita e não foram encontrados trabalhos similares, em que a mutação é aplicada diretamente no oráculo de teste, ou seja, na avaliação da qualidade de oráculos de teste escritos na forma de classes JUnit.

Referências

Delamaro, M., Maldonado, J., ; Jino, M. (2007). *Introdução ao teste de software*. Elsevier, São Paulo, SP, 1ª edição.

DeMillo, R. A., Lipton, R. J., ; Sayward, F. G. (1978). Hints on test data selection: Help for the practicing programmer. *IEEE Computer Society Press*, 11(4):34–41.

Harman, M., McMinn, P., Shahbaz, M., ; Yoo, S. (2013). A comprehensive survey of trends in oracles for software testing. Technical Report CS-13-01, University of Sheffield, Department of Computer Science.

Myers, G., Sandler, C., Badgett, T., ; Thomas, T. (2004). *The Art of Software Testing*. Business Data Processing: A Wiley Series. Wiley.

Nardi, P. A. (2013). *On test oracles for Simulink-like models*. PhD thesis, Universidade de São Paulo, São Carlos, SP.

Nunes, F. L., Delamaro, M. E., ; Oliveira, R. A. (2009). Oráculo gráfico como apoio na avaliação de sistemas de auxílio ao diagnóstico. In *IX Workshop de Informática Médica*, Bento Gonçalves, RS, Brasil.

Oliveira, R. A., Delamaro, M. E., ; Nunes, F. L. (2009). Oráculos de Teste para Domínios GUI: Uma Revisão Sistemática. *III Brazilian Workshop on Systematic and Automated Software Testing*.

Shahamiri, S. R., Kadir, W. M. N. W., Ibrahim, S., ; Hashim, S. Z. M. (2011). An automated framework for software test oracle. *Information and Software Technology*, 53(7):774–788.

Shrestha, K. ; Rutherford, M. (2011). An Empirical Evaluation of Assertions as Oracles. In *Proceedings of the 4th International Conference on Software Testing, Verification and Validation*, p. 110–119.

Staats, M., Whalen, M., ; Heimdahl, M. (2011). Programs, tests, and oracles: the foundations of testing revisited. In *Proceedings of the 33rd International Conference on Software Engineering*, p. 391–400.

Tan, R. P. ; Edwards, S. H. (2004). Experiences evaluating the effectiveness of JML-JUnit testing. *ACM SIGSOFT Software Engineering Notes*, 29(5):1–4.

Weyuker, E. J. (1982). On testing non-testable programs. *The Computer Journal*, 25(4):465–470.

Descoberta de Conhecimento durante a Execução de Projetos de Desenvolvimento de Software para Apoiar o Alinhamento de Processos

Renata Mesquita da Silva Santos[1 2]

Orientador: Toacy Cavalcante de Oliveira[1]

[1]Programa de Pós-Graduação em Engenharia de Sistemas de Computação – COPPE – UFRJ
Caixa Postal 68511 – CEP 21945-970 – Rio de Janeiro, RJ

[2]Instituto Federal de Educação, Ciência e Tecnologia Fluminense
Rua Dr Siqueira, 273 - Parque Dom Bosco - Campos dos Goytacazes, RJ - CEP 28030-130

{renatames, toacy}@cos.ufrj.br

Nível: Doutorado
Ano de ingresso no programa: 2013
Época prevista de conclusão: Março de 2017
Data da aprovação da proposta de tese (qualificação): Outubro de 2015
Eventos Relacionados: SBES

Resumo: Organizações de desenvolvimento de software buscam continuamente a melhoria dos seus processos de desenvolvimento, uma vez que estes processos estão diretamente relacionados com a qualidade do produto de software a ser construído. Em geral uma organização de desenvolvimento de software defini um processo padrão de desenvolvimento de software. Porém durante a execução do processo podem ocorrer desvios, o que em última análise, pode descaracterizar a proposta do processo inicial. Neste sentido, o objetivo geral desta pesquisa de tese é verificar o alinhamento entre o processo de software do projeto e o processo executado, ao longo da condução do projeto de desenvolvimento de software. Os objetivos específicos são: definir uma estruturação dos dados gerados ao longo da execução do processo de desenvolvimento de software; e tornar explícitas as características de processo definidas no Processo de Software de Projeto, durante a execução do processo de desenvolvimento de software. Acreditamos que a estruturação dos dados, gerados ao longo da execução do processo de desenvolvimento de software, e a aplicação de técnicas de Descoberta de Conhecimento em Base de Dados (DCBD), para explicitar as características de processo de desenvolvimento de software, fornecerá uma infraestrutura de apoio à tomada decisão aos profissionais, ao longo da execução do pocesso de desenvolvimento de software.

Palavras-chave: Processo de Desenvolvimento de Software; Característica de Processo; Descoberta de Conhecimento em Base de Dados

1. Caracterização do Problema

Organizações de desenvolvimento de software buscam constantemente o desenvolvimento de produtos de software de qualidade. Uma vez que a qualidade do produto de software está diretamente relacionada a qualidade do processo de software [Fuggetta, 2000]. Muitos esforços têm sido dedicados para o aumento da produtividade, eficiência e efetividade do processo de desenvolvimento e manutenção dos produtos de software, com destaque para a criação de padrões de qualidade de processo, tais como a norma ISO/IEC 12207 [ISO, 2008] e os modelos de maturidade CMMI [Chrissis et al., 2006] e MPS.Br [SOFTE, 2012]. Processos são definidos caso a caso, levando em consideração características específicas do projeto em questão, incluindo as tecnologias envolvidas, o tipo de software a ser desenvolvido, o domínio da aplicação e a equipe de desenvolvimento [Rocha et al., 2001].

Figura 1. Níveis de definição de Processo de Software e Processo de Software Executado

Em geral, uma organização de desenvolvimento de software defini um Processo Padrão de desenvolvimento de software. A Figura 1 apresenta a abordagem de níveis de definição de processo, proposta por Rocha et al. (2011), na qual são considerados os níveis: Processo de Software Padrão da Organização, Processos de Software Padrão Especializado e Processo de Software de Projeto. O Processo Padrão é um processo que serve como referência para guiar a execução de todos os projetos de software dentro de uma organização. Mas em uma organização pode ocorrer de muitos projetos serem realizados para um mesmo tipo de software (sistemas de informação, aplicações Web etc) ou seguindo um mesmo paradigma (por exemplo, orientado a objetos – OO), desta forma, no nível intermediário, o processo padrão da organização pode ser especializado para considerar algumas classes de tipos de software, paradigmas ou domínios de aplicação [Bertollo, 2006]. Porém cada projeto da organização possui características específicas (modelos de ciclo de vida, características do projeto, características da equipe, disponibilidade de recursos, requisitos de qualidade do produto), sendo necessário adaptar o Processo de Software Padrão Especializado, modelando o Processo de Software de Projeto. A partir do Processo de Software de Projeto, pode-se dar início à execução do processo de desenvolvimento de software, no qual as tarefas modeladas são realizadas tanto pelos desenvolvedores quanto automaticamente.

Uma característica importante presente na execução de processos de software é a dependência do ser humano para sua condução. Processos de Software são compostos por atividades criativas (modelagem, verificação, comunicação, tomadas de decisões, entre outras), as quais se faz necessária a interação humana para sua realização

[Bendraou, 2007 e Laurent et al., 2014]. Sendo assim, durante a execução, os elementos de processo definidos podem ser alterados para, por exemplo, atender à situações não previstas, relacionadas a muitos fatores, tais como prazo e gerenciamento de recursos [Laurent et al., 2014]. É comum ver gerentes de projeto realizarem alterações devido a preferências pessoais, experiências anteriores, pressões de custo e prazo, e também por treinamento inadequado, falta de comprometimento e falta de comunicação entre os membro da equipe. Neste sentido, a execução do processo pode se desviar do modelo de Processo de Software de Projeto, o que em última análise, pode descaracterizar a proposta do processo inicial, sendo possível obter um novo processo, o Processo de Software Executado, como mostrado na Figura 1.

Um modelo de processo é caracterizado por conjunto de elementos de processo (atividades, artefatos, recursos e procedimentos), como visto em Dumas et al. (2013), Van Der Aalst (2011) e Weske (2007), e também por um conjunto de restrições ao comportamento dos elementos e do modelo de forma geral. Neste estudo os elementos de processo, e as restrições relacionadas a eles e ao modelo de processo, serão chamados de Características de Processo de Desenvolvimento de Software.

Desvios que ocorrem durante a execução do processo de desenvolvimento de software, em relação ao Processo de Software de Projeto, foi observado em Santos, Oliveira e Brito e Abreu (2015), quando pela aplicação da técnica *VarIdentify*, as variabilidades entre o processo executado e planejado foram destacadas. A técnica proposta utiliza Mineração de Processo, para descobrir o Processo de Software executado. Naquele trabalho foi possível observar os desvios que ocorrem ao longo da condução dos projetos de desenvolvimento, quando comparado ao Processo de Projeto. Outra questão observada foi que, embora sejam gerados uma grande quantidade de dados ao longo da execução do projeto de desenvolvimento de software, é difícil extrair informações úteis. Esta dificuldade encontrada se dá pela falta de estruturação e tratamento dos dados gerados.

O objetivo geral desta pesquisa de tese é verificar o alinhamento entre o processo de software do projeto e o processo executado, ao longo da condução do projeto de desenvolvimento de software. Os objetivos específicos são: definir uma estruturação dos dados gerados ao longo da execução do processo de desenvolvimento de software; e tornar explícitas as características de processo definidas no Processo de Software de Projeto, durante a execução do projeto de desenvolvimento de software. O fato destas características estarem implícitos, ou seja, não serem observadas explicitamente nas infraestruturas que capturam as informações de processo, torna difícil o alinhamento. A questão de pesquisa que orienta o desenvolvimento desta tese consiste em: *Como apoiar a verificação do alinhamento entre o processo de software de projeto e o processo executado executado, utilizando técnicas de descoberta de conhecimento em base de dados gerados ao longo da condução dos projetos de desenvolvimento de software?*

O restante do artigo está organizado da seguinte forma. A Seção 2 introduz os conceitos fundamentais de Processo de Software e Descoberta de Conhecimento em Base de Dados. Na Seção 3 são detalhadas as contribuições deste trabalho. A Seção 4 apresenta a metodologia e estado atual deste trabalho. A Seção 5 descreve os trabalhos relacionados. Por fim, a avaliação dos resultados são apresentados na Seção 6.

2. Fundamentação Teórica

Nesta seção serão apresentados conceitos relacionados a Processo de Software (Seção 2.1), Descoberta de Conhecimento em Base de Dados (Seção 2.2).

2.1. Processo de Software

Processos de software descrevem as diversas fases necessárias para produzir e manter um produto de software. Processos são importantes porque eles impõem consistência e estrutura a um conjunto de atividades [Pfleeger e Atlee, 2010]. Estas características são úteis quando se deseja saber como fazer alguma coisa bem e orientar que outras pessoas façam da mesma forma. A estrutura do processo guia ações que permitem examinar, entender, controlar e melhorar as atividades que compõem o processo.

Para se disseminar a utilização de um processo em uma organização, é primordial que este processo esteja bem documentado e de acordo com Campos e Oliveira (2011) a modelagem de processos, na última década, tornou-se um importante mecanismo para a compreensão do comportamento dinâmico das organizações. Devido a importância da descrição de Processos de Desenvolvimento de Software, é requerido que estes tenham uma organização lógica das diversas atividades técnicas e gerenciais que envolvem agentes, métodos, ferramentas e artefatos, e restrições que possibilitem disciplinar, sistematizar e organizar o desenvolvimento e manutenção de produtos de software [Pressman, 2010].

2.2. Descoberta de Conhecimento em Base de Dados

De acordo com Cardoso (2008), a quantidade de dados disponíveis vem crescendo assustadoramente nos últimos anos e vários fatores contribuíram para esse incrível aumento. O baixo custo de armazenagem pode ser visto como uma das principais causas do surgimento dessas enormes bases de dados. Para que o conhecimento seja descoberto de forma eficiente, é realizado um processo chamado descoberta de conhecimento em banco de dados (DCBD ou KDD do inglês *Knowledge Discovery in Databases*) [Goldschmidt, 2005].

As etapas para Descoberta de Conhecimento consistem em: selecionar os dados, preprocessá-los para retirar dados duplicados ou inconsistentes, transformá-los nos formatos adequados conforme algoritmos e ferramentas utilizadas, submeter os dados transformados à mineração propriamente dita, e, após, fazer a interpretação e validação dos resultados [Fayyad et al, 1996]. O ponto central do processo de descoberta é a etapa de Mineração [Goldschmidt, 2005]. A Mineração de Dados é amplamente difundida, porém quando resultado final desta etapa é a geração de Processo, esta etapa é chamada de Mineração de Processos.

Mineração de Processos é uma área de investigação, relativamente recente, que se posiciona não só entre Inteligência Computacional e Mineração de Dados, mas também entre a modelagem e análise de processos [Van Der Aalst, 2011]. O objetivo da Mineração de Processos é a utilização dos dados dos registros de eventos para extrair processos relacionados às informações, por exemplo, descobrir automaticamente um modelo de processo por observação dos eventos registrados por alguns sistemas de empresas [Van Der Aalst, 2011]. Na área de processos, abordagens de mineração de processos são importantes no sentido de permitir a mineração e descoberta de

conhecimento a partir de repositórios de ambientes que armazenem informações sobre processos [Rubin et al. 2007].

3. Contribuições

Esta pesquisa tem o objetivo de trazer contribuições para a área de Processo de Software, especificamente para o Gerenciamento de Processo de Desenvolvimento de Software. A seguir destacamos as principais contribuições da pesquisa:

Estruturação dos dados gerados ao longo da execução do processo de desenvolvimento de software: Dados de processo são gerados quando os Engenheiros de Software criam ou mantêm seus projetos, e estão contidos nos artefatos de desenvolvimento gerados e produzidos durante a execução do processo de desenvolvimento de software. Porém extrair informações úteis dos dados gerados não é uma atividade trivial, principalmente pela falta de estruturação dos dados. A estruturação dos dados, ou seja, a definição de uma estrutura de armazenamentos dos dados, permitirá que a extração de informações úteis seja mais rápida e otimizada.

Tornar explícitas as características de processo de desenvolvimento de software, definidas no Processo de Software de Projeto, durante a execução do projeto de desenvolvimento de software: A aplicação de técnicas de DCBD, tais como algoritmos de Classificação, Clusterização, Associação, entre outros; podem tornar explícitas as características de processo de desenvolvimento de software, permitindo aos profissionais o acesso ao conhecimento tácito de processo de forma automatizada. Explicitar as características de processo, ou seja, tornar explícito o conhecimento tácito de processo, fortalecerá o processo de tomada de decisão dos profissionais (Gerentes de Projeto, Analistas e Desenvolvedores), durante a execução do projeto de desenvolvimento de software.

Alinhamento entre o planejamento e execução de processo: O acesso às características de processo, durante a execução do projeto de desenvolvimento de software, pode permitir que os profissionais envolvidos com a implementação e manutenção de software, possam tomar decisões mais embasadas e alinhadas ao planejamento, ao longo da condução dos projetos de desenvolvimento de software.

4. Metodologia e estado atual do trabalho

Esta seção apresenta as atividades planejadas para alcançar o objetivo desta pesquisa; o estado atual também será apresentado. A Figura 2 ilustra as atividades dessa pesquisa, que foram delineadas com base no caminho para um trabalho de pesquisa, definido em [Wazlawick, 2009], as quais serão descritas a seguir.

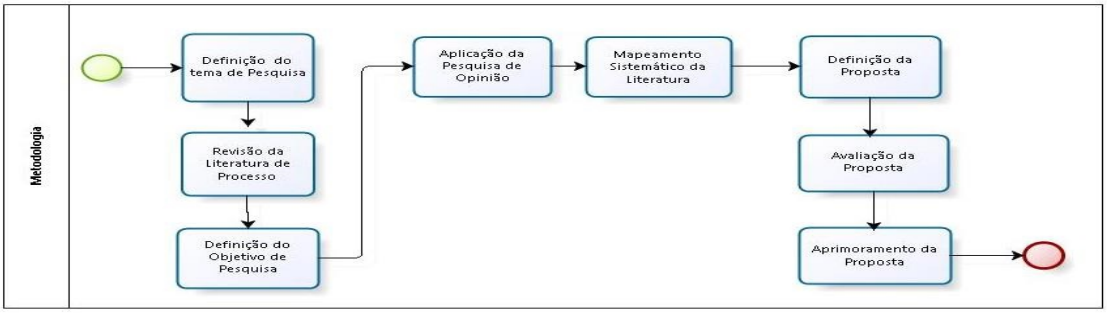

Figura 2. Metodologia da Pesquisa

1. **Definição do tema de Pesquisa:** A motivação para definição do tema, surgiu a partir da proposta da técnica *VarIdentify* [Santos, Oliveira & Brito e Abreu, 2015], que propõe identificar as variações que ocorrem entre o processo planejado e executado.

2. **Revisão da Literatura de Processo:** A partir da motivação apresentada, foi identificada a necessidade de investigar quais são as características de software relevantes para a existência do processo, de forma a permitir que os conceitos do processo estejam explícitas. Neste sentido foi realizada a seleção de autores, por amostra de conveniência, e seus respectivos trabalhos que abordassem a definição de processo.

3. **Definição do Objetivo da Pesquisa:** A partir da definição do tema e da revisão da literatura de processo, definimos que o objetivo geral desta pesquisa de tese é verificar o alinhamento entre o processo de software do projeto e o processo executado, ao longo da condução do projeto de desenvolvimento de software.

4. **Aplicação da Pesquisa de Opinião:** Uma pesquisa de opinião está sendo realizada com o objetivo de investigar se as características de processo, identificados na literatura, estão presentes ao longo da condução dos projetos de desenvolvimento de software. Também objetivamos investigar quais ferramentas são utilizadas, ao longo da condução dos projetos de desenvolvimento de software, para identificar as atuais estruturas que armazenam os dados de processo.

5. **Mapeamento Sistemático da Literatura:** O mapeamento sistemático será realizado para identificar o que tem sido feito para resolver o problema desta pesquisa.

6. **Definição da Proposta:** Para alcançar os objetivos, serão implementadas técnicas de DCBD, para tornar explícito o conhecimento tácito de processo de desenvolvimento de software, ao longo da condução do projeto de desenvolvimento de software. Alinhada à aplicação de técnicas de descoberta de conhecimento, será proposta a definição de uma estrutura de armazenamentos dos dados, objetivando que a extração de informações de processo seja rápida e otimizada.

7. **Avaliação da proposta:** A avaliação da proposta será por meio da aplicação das técnicas propostas. O detalhamento relacionado à avaliação dos resultados serão descritos na Seção 6.

8. **Aprimoramento da Proposta:** A partir dos resultados das avaliações da proposta, que serão realizadas ao longo do trabalho, a proposta poderá evoluir constantemente, para atender os objetivos definidos.

Neste momento estamos na etapa 4, analisando os resultados da primeira rodada do Pesquisa de Opinião, que foi aplicada de forma supervisionada. A população neste rodada foi por conveniência, sendo composta por respondentes da nossa rede de contato. A forma de aplicação foi por meio de entrevistas, que foram realizadas no Brasil e em Portugal. A aplicação supervisionada da pesquisa de opinião foi utilizada para calibrar o questionário para a rodada não-supervisionada, na qual utilizaremos o *Linkedin* como fonte de população. Pretendemos a partir dos resultados encontrados, identificar quais características de processo podem ser observadas pelas equipes de desenvolvimento. Pretendemos também identificar a forma de observação das características, ou seja, quais ferramentas são utilizadas. O resultado da pesquisa de opinião será usado para orientar a implementação de técnicas de descoberta de conhecimento em base de dados

e de tratamento de informações para tornar explícito o conhecimento tácito relacionado ao processo de desenvolvimento de software.

5. Trabalhos Relacionados

Os Projetos de Desenvolvimento de Software geram uma quantidade impressionante de dados [Baysal, 2014]. Diante deste cenário, é possível identificar trabalhos que abordam a utilização dos dados gerados pelos projetos de desenvolvimento de software, para fornecer informações úteis aos profissionais envolvidos no desenvolvimento e manutenção de software.

Mahdiraji et al. (2012) realizaram em seu trabalho uma análise de abordagens e técnicas disponíveis, para automatizar a extração de conhecimento a partir de registros de eventos. É destacado neste trabalho que embora as informações de processo estejam disponíveis, frequentemente pode ser difícil reconstruir o conhecimento sobre os processos ou os processos podem desviar do comportamento planejado. Dohrmann (2014) destaca que os processos ativos desviam de suas definições de processo relacionada, devidos à evolução, durante a execução do processo. Rubin et al. (2007) apresentam em seu trabalho um Framework para Mineração de Processo. Este Framework pode ser usado para obter os modelos de processo de software, bem como analisá-los e otimizá-los. Baysal (2014) propôe em seu trabalho o uso de análises quantitativa e qualitativa em projetos de desenvolvimento de software, com o objetivo de apoiar os profissionais nas tomadas de decisão, oferecendo informações que possam fornecer um melhor entendimento sobre processos, sistemas, produtos e usuários final.

É importante ressaltar que nenhum desses estudos tem como objetivo explicitar as características, definidas no Processo de Software de Projeto, para alinhar planejamento e execução de processo. Outro diferencial deste trabalho é a proposta de uma estrutura de armazenamento dos dados, para permitir que a extração de informações de processo, seja rápida e otimizada.

6. Avaliação dos Resultados

Para a avaliação do trabalho proposto, serão realizados estudos experimentais que tem como objetivo observar o impacto, no gerenciamento dos projetos de desenvolvimento de software, da aplicação das técnicas propostas. Mais precisamente pretende-se avaliar se com a aplicação das técnicas propostas, será possível um alinhamento, efetivo e otimizado, entre o processo de desenvolvimento de software de projeto e o processo executado executado, durante a execução do projeto de desenvolvimento de software.

Referências

Baysal, O. (2014). Supporting Development Decisions with Software Analytics (Doctoral dissertation, University of Waterloo).

Bendraou, R., Sadovykh, A., Gervais, M. P., & Blanc, X. (2007, August). Software Process Modeling and Execution: The UML4SPM to WS-BPEL Approach. In Software Engineering and Advanced Applications, 2007. 33rd EUROMICRO Conference on (pp. 314-321). IEEE.

Bertollo, G., Segrini, B., & Falbo, R. D. A. (2006). Definição de Processos de Software em um Ambiente de Desenvolvimento de Software Baseado em Ontologias. V Simpósio Brasileiro de Qualidade de Software, SBQS, 6, 72-86.

Campos, A. L., & de Oliveira, T. C. (2011). Modeling Work Processes and Software Development-Notation and Tool. In ICEIS (3) (pp. 337-343).

Cardoso, o. N., & Machado, r. T. (2008). Gestão do conhecimento usando data mining: estudo de caso na Universidade Federal de Lavras. Revista de administração pública, 42(3), 495-528.

Crissis, M. B., Konrad, M., & Shrum, S. (2006). Guidelines for Process Integration and Product Improvement.Addison-Wesley Longman Publishing Co., Inc.

Dohrmann, P. (2014, August). Language-Based Process Model Discovery and Enhancement. In Software Engineering and Advanced Applications (SEAA), 2014 40th EUROMICRO Conference on (pp. 131-134). IEEE.

Dumas, M., La Rosa, M., Mendling, J., & Reijers, H. A. (2013). Fundamentals of business process management (pp. I-XXVII). Berlin: Springer.

Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P., & Uthurusamy, R. (1996). Advances in knowledge discovery and data mining. Menlo Park, EUA: AAAI Press, 611 p.

Fuggetta, A. (2000, May). Software process: a roadmap. In Proceedings of the Conference on the Future of Software Engineering (pp. 25-34). ACM.

Goldschmidt, R., & Passos, E. (2005). Data Mining: um guia prático. Campus.

ISO/IEC-12207, 2008. “Systems and Software Engineering - Software Life Cycle Process", The International Organization for Standardization and the International Electrotechnical Commission, v. ISO/IEC 12207, Genebra, Suíça.

Laurent, Y., Bendraou, R., Baarir, S., & Gervais, M. P. (2014). Alloy4spv: A formal framework for software process verification. In Modelling Foundations and Applications (pp. 83-100). Springer International Publishing.

Mahdiraji, A. R., Rossi, B., Sillitti, A., & Succi, G. (2012). Knowledge Extraction from Events Flows. In Methodologies and Technologies for Networked Enterprises (pp. 221-236). Springer Berlin Heidelberg.

Pressman, R. S. (2011). Engenharia de software. McGraw Hill Brasil.

Rocha, A. R. C. D., Maldonado, J. C., & Weber, K. C. (2001). Qualidade de software. São Paulo: Prentice Hall.

Rubin, V., Günther, C. W., Van Der Aalst, W. M., Kindler, E., Van Dongen, B. F., & Schäfer, W. (2007). Process mining framework for software processes. In Software Process Dynamics and Agility (pp. 169-181). Springer Berlin Heidelberg.

Santos, R.M.S.; Oliveira, T; Brito E Abreu, F. (2015) Mining Software Process Development Variations. (Poster Session) In: Proceedings of the 30th ACM/SIGAPP Symposium On Applied Computing (pp 1657-1660).

SOFTEX. (2012) MPS.BR:Melhoria de Processo do Software Brasileiro, Guia Geral de Software, disponível em: http://www.softex.br/mpsbr.

Van Der Aalst, W. (2011). Process mining: discovery, conformance and enhancement of business processes. Springer Science & Business Media.

Wazlawick, R. (2009). Metodologia de pesquisa para ciência da computação. Elsevier Brasil.

Weske, M. (2012). Business process management: concepts, languages, architectures. Springer Science & Business Media.

ARES: Uma Modelagem de Ameaças Baseada em Papéis Específica para o Ecossistema Web

Aluno: Carlo Marcelo Revoredo da Silva[1],
Orientador: Vinicius Cardoso Garcia[1]

[1]CIn – Universidade Federal de Pernambuco (UFPE)
Av. Jorn. Aníbal Fernandes – CDU – 50740-560 – Recife – PE – Brazil

{cmrs,vcg}@cin.ufpe.br

Nível: Doutorado
Programa: Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco (CIn – UFPE)
Ano de Ingresso no programa: 2014
Época Previsão de Conclusão: 2018
Sigla do evento do CBSoft relacionado: SBES

Resumo. *As Modelagens de Ameaças, do inglês Threat Modeling (TM), são artefatos focados em identificar e corrigir ameaças de segurança através de uma classificação pragmática, que oferece estratégias de prevenção e riscos aos ativos durante o desenvolvimento do software. Diante das vulnerabilidades emergentes na Web, é válido o uso de TMs para aplicações ou serviços desse ambiente. Contudo, as TMs disponíveis, em sua maioria, são projetadas para o propósito geral e estruturadas com base no desenvolvimento tradicional. Isso traz como consequência uma modelagem que envolve poucas atividades e indivíduos nas contramedidas às ameaças, além de dificultar sua adesão em escopo incremental, a exemplo do desenvolvimento ágil. Este artigo apresenta a ARES, uma TM específica para o ecossistema Web. Ela traz uma abordagem que delega contramedidas, seja no desenvolvimento tradicional ou ágil, e baseada em quatro papéis, a saber: (i) Papel de Conscientizar* ***(Aware)****, em que todo e qualquer participante precisa saber definir, identificar e classificar ameaças existentes, considerando comportamentos e similaridades; (ii) Papel de Avaliar* ***(Rate)****, que propõe analisar fatores de riscos e impactos aos ativos ainda na atividade de Análise e Design do software, possibilitando mensurar possíveis danos e propagação; (iii) Papel de Garantir* ***(Ensure)****, que propõe executar e conduzir as melhores práticas durante as atividades de Design, Codificação e Teste; e por fim, (iv) Papel de Supervisionar* ***(Supervisor)****, que foca na identificação e tratamento de vulnerabilidades existentes durante as atividades de Teste, Implantação e Manutenção. Com a evolução deste estudo, pretendemos desenvolver uma avaliação sobre vantagens e desvantagens da ARES em comparação com TMs na literatura, através de um estudo de caso com a metodologia CLASP em desenvolvimento ágil de uma aplicação Web.*

Palavras-chave. *Modelagem de Ameaças, Avaliação de Riscos, Desenvolvimento Ágil, Ameaças de Segurança, Segurança em Aplicações ou Serviços Web.*

1. Caracterização do Problema

Atualmente a *Web* vem se consolidando como um ambiente eficiente para o desenvolvimento de aplicações. Com seu advento, diversas organizações motivaram-se a migrar suas aplicações para sua plataforma, e com a Computação em Nuvem, mais precisamente o modelo *Software* como Serviço, SaaS, a tendência ganhou força.

Contudo, está cada vez mais crescente o número de Ameaças à Segurança neste ambiente. Segundo a Symantec, o número de ataques em aplicações *Web* cresceu mais de 30% entre 2012 e 2014 [Symantec 2015]. Os ataques demonstram motivações variadas, desde ideologias a ganhos financeiros, ou interesses políticos através da espionagem. Não obstante, o projeto OWASP mantem uma lista com mais de 160 vulnerabilidades diferentes [OWASP 2014]. Com seu advento, a *Web* evoluiu e criou um ecossistema entre pessoas, dispositivos e serviços que trafegam dados sensíveis, estes são chamados de ativos por terem valor agregado ao usuário final.

Como alternativa ao combate de ameaças, faz-se o uso de Modelagem de Ameaças, do inglês, *Threat Modeling* (TM). Ela é focada em identificar ameaças e avaliar riscos através de uma classificação pragmática. Contudo, é preciso considerar três problemáticas quanto ao seu uso: (i) a maioria das TMs na literatura tem sua abordagem para o propósito geral, e, uma vez que exerça seus processos de forma análoga em qualquer tipo de *software*, a TM demonstra baixa profundidade ao domínio, pondo em cheque sua eficiência em certas ameaças intrínsecas ao ambiente *Web*.

Outra problemática (ii) diz respeito à relação da TM com as metodologias de desenvolvimento. As TMs, em sua maioria, são direcionadas ao processo convencional, como o modelo em cascata. Esse modelo determina que as atividades sejam desenvolvidas sequencialmente [Sommerville 2010]. Ou seja, para avançar a uma nova, a atividade antecessora deve ser totalmente desenvolvida e quando concluída, todos os esforços serão voltados para a próxima. Diante disso, a utilização da TM fica restrita à atividade de *Design*. A justificativa é que o desenvolvedor terá em mãos requisitos com especificações imutáveis, sugerindo um processo para identificações de ameaças de mão única. Consequentemente, a TM não se envolve em demais atividades e responsáveis, apesar de suscetíveis a ameaças, tornando a eficiência da TM questionável.

E por fim, a terceira problemática (iii) remete ao fato de aplicar TM em desenvolvimento ágil. Por ter seu cerne originado no desenvolvimento convencional, aplicar TM ao processo ágil não é uma tarefa simples. Neste tipo de metodologia as mudanças são esperadas a todo o momento, o que implica dizer que as fases antecessoras e sucessoras trabalham continuamente de forma iterativa e incremental. Isso gera resistência ao uso das TMs disponíveis, provendo desafios para sua adesão.

O objetivo deste trabalho é propor uma TM, intitulada ARES, específica para a *Web*, aplicável ao desenvolvimento ágil, que faz uso de uma taxonomia de vetores de ataque para identificar ameaças, atenuar riscos e auxiliar no tratamento das vulnerabilidades. Ela delega ações baseadas em papéis durante as fases de desenvolvimento. Nossa taxonomia nada mais é que uma teoria, onde levantamos hipóteses, comprovamos fenômenos, e por fim construímos uma teoria fundamentada em dados [Eisenhardt 1989], reforçada por teorias extraídas de revisões sistemáticas na literatura (SLR) [Kitchenham 2004] e estudos empíricos por medições [Basili et al. 1994].

2. Fundamentação Teórica

O principal da TM é garantir boas práticas ao processo de desenvolvimento, orientando a equipe em prevenção e contramedidas à possíveis ameaças de segurança ao *software*. A cada dia surgem novos desafios para o controle das ameaças, seja pela grande diversidade de novas ou pela evolução do comportamento daquelas já conhecidas. Nesse cenário, as TMs são responsáveis por especificar boas práticas de cunho preventivo, capacitando os envolvidos com desenvolvimento guiado em determinadas situações previamente esperadas, fazendo uso de uma modelagem centrada em um determinado contexto da ameaça [Shostack 2014]. Tais contextos podem seguir uma das três abordagens distintas, a saber:

- Centrado no Atacante: nela, a modelagem especifica o comportamento de um atacante, orientando sobre técnicas e propagação de suas possíveis ações. Nessa fase, faz-se interessante uma análise dos aspectos de interface humana da aplicação e vetores relacionados à Engenharia Social.
- Centrado nos Ativos: nela, a modelagem prover uma abordagem com base em ativos do *software*. Ela conduz os envolvidos a elaborar uma listagem de ativos, além de aplicar algum tipo de metodologia para descrever a relação com os ataques correlatos e consequentemente priorizar as prevenções com base no impacto e risco da respectiva ameaça.
- Centrado no *Software*: é a abordagem mais utilizada entre as TMs disponíveis. Ela conduz os envolvidos com boas práticas durante o processo de concepção do *software*, relacionando os tipos de ataques com as especificações predefinidas.

3. Contribuições

Para tratar das problemáticas (i), (ii) e (ii) apresentadas na Seção 1, este trabalho de doutorado pretende defender a elaboração de uma TM, intitulada ARES, que oferece uma nova estrutura de classificação taxonômica que visa oferecer as contribuições descritas na Tabela 1 para o desenvolvimento de recursos do ecossistema *Web*. Tal ambiente, por ser tão heterogêneo e emergente, faz jus à necessidade de uma TM direcionada. O modelo propõe o desenvolvimento de uma taxonomia que inicialmente foi construída em programa de mestrado [da Silva and Garcia 2014] e evoluída no decorrer deste doutorado.

Tabela 1. Relação de Contribuições e Artefatos

Contribuições	Artefatos
Classifica ameaças e riscos estruturados por semântica.	Taxonomia
Relaciona riscos, ameaças e ativos, considerando seus impactos.	Taxonomia
Apresenta uma nova abordagem centrada em vetores de ataques.	Árvore de Ataques
Apresenta estratégias para identificação e prevenção de ameaças, atenuar riscos e auxiliar na correção de vulnerabilidades.	Taxonomia / Abordagem Baseada em Papéis
Segmenta responsabilidade de contramedidas no desenvolvimento convencional ou ágil.	Abordagem Baseada em Papéis
Oferece suporte empírico em contramedidas de ameaças na fase de entrega do *software*.	Abordagem Baseada em Papéis
Avalia de forma empírica a utilização de TM na metodologia CLASP em desenvolvimento ágil para *Web*. Compara a ARES com TMs identificadas na literatura.	Estudo de Caso

A proposta considera o uso de uma metodologia em que todos os participantes do desenvolvimento precisam ter certo nível de consciência sobre as ameaças, em que seu nível de conhecimento é segmentado por papéis. Conforme descrito na Tabela 1, a ARES traz sete contribuições resultantes de quatro artefatos a serem elaborados por este estudo.

3.1. Taxonomia e Árvore de Ataques

A modelagem ARES faz uso da fundamentação de dados [Eisenhardt 1989] dos ataques mais emergentes apresentada na literatura. Com base nesses dados, temos o intuito de construir modelos de ameaças predeterminados, relacionando as similaridades do método de exploração da vulnerabilidade, resultando em uma taxonomia. Com base nessa teoria, acreditamos que nossa abordagem possibilita a construção de uma modelagem capaz de estimar, identificar e prevenir um considerável número de ameaças potencialmente existentes em aplicações do ecossistema *Web*.

A modelagem ARES traz uma nova abordagem para modelagem de ameaças. Diferente das abordagens fundamentadas na Seção 2, nossa proposta é centrada em vetores de ataques, pois é focada no método de exploração e propagação das vulnerabilidades. Para que a abordagem identifique e classifique de forma eficaz, é importante que sua estrutura tenha precisão sobre o comportamento de cada ameaça. A precisão reflete nas relações em comum entre vetores com similaridades em sua semântica, resultando em uma categorização das falhas por um determinado comportamento. Podemos citar as similaridades nos métodos de ataques do tipo *Buffer Overflow* [OWASP 2014], que, apesar de atuarem em diferentes estruturas de dados, estes compartilham o mesmo vetor de ataque, ou seja, são exploradas de forma similar.

Para tanto, a abordagem segue uma estrutura de Árvores de Ameaças e Riscos aos ativos, conforme a Figura 1, como maneira formal e metódica de descrever o comportamento de ataques [Schneier 1999]. O benefício é representar uma determinada ação ilustrada no nó-raiz, observar diferenças e similaridades através dos nós vizinhos e mensurar comportamento de diferentes maneiras até seus objetivos, representado pelos nós abaixo, ao mesmo tempo estimando riscos e impactos aos ativos.

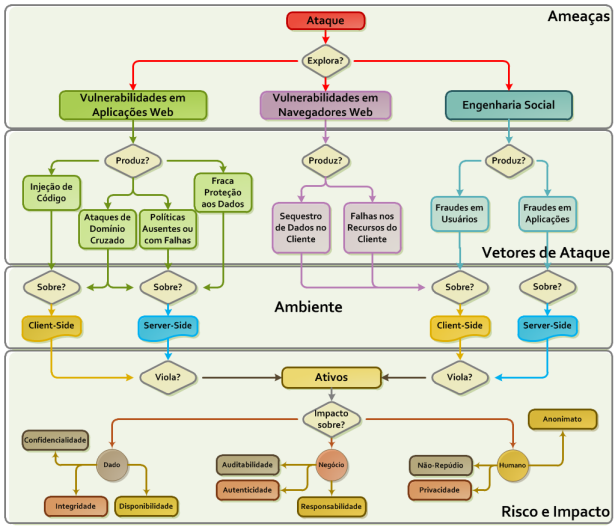

Figura 1. Taxonomia de ameaças do ecossistema *Web*.

A modelagem propõe prevenções e contramedidas através de especificações formais que posteriormente possam produzir diagramas com menor nível de abstração, possibilitando direcioná-los ao processo de *Design*, Codificação e Teste, a exemplo de *Model-Driven Architecture* [OMG 2012], *Feature-Driven Development* [Nebulon 2012] e *Test-Driven Development* [Beck 2002], que resultam com eficiência em modelos de domínio.

A proposta cobre infraestrutura, fases da liberação do *software* como Implantação e Manutenção, componentes de terceiros, serviços remotos, decisões arquiteturais no contexto *Server-side* e *Client-side*. E, como meio de medir riscos e impactos, considera a violação dos atributos de segurança em três fatores distintos: dados, como exemplo da integridade; negócios, como a responsabilidade; e humanos, como a privacidade.

3.2. Abordagem Baseada em Papéis

A modelagem sugere que todos os envolvidos nas fases do *software* tenham um ou mais responsabilidades no combate de ameaças durante todo o ciclo de vida do processo. São quatro papéis distintos que se relacionam nas atividades conforme ilustrado na Figura 2. A justificativa é que nas modelagens convencionais as responsabilidades são direcionadas apenas ao *Design* arquitetural do *software*. Como nosso intuito é envolver todo o ciclo de vida do processo, se faz necessário especificar as responsabilidades de forma pragmática. A abordagem por papéis oferece eficiência na distribuição das responsabilidades e maior gestão na investigação de prevenções e contramedidas.

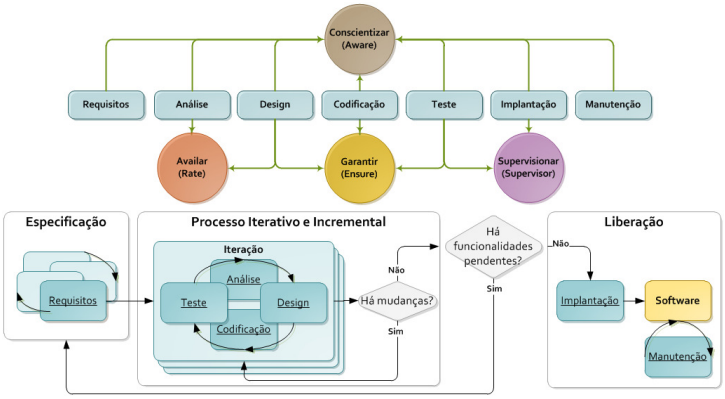

Figura 2. Distribuição dos papéis entre as atividades do desenvolvimento.

Papel de Conscientizar (*Aware*): a modelagem define que todos os participantes do desenvolvimento pertençam a esse papel. O papel declara que todos devem ser capazes de conhecer e reconhecer, em nível de concepção, as potenciais ameaças ao *software*. A análise conceitual é baseada na taxonomia proposta pela modelagem. A importância de considerar TM em todas as fases remete aos vetores de ameaças consequentes a diversos contextos do *software*. Tendo a atividade de Manutenção como exemplo, é de suma importância considerar métodos sistemáticos quanto ao treinamento dos usuários finais, conscientizando-os sobre comportamentos da Engenharia Social.

Papel de Avaliar (*Rate*): esse papel engloba fatores de avaliação de riscos envolvidos no *software* que são resultantes de ameaças. É importante saber analisar a complexidade de solucionar possíveis ameaças durante o desenvolvimento dos requisitos. As equipes de Análise e *Design* do *software* devem discutir as melhores decisões, as práticas e os possíveis *trade-offs*. Para tanto, é necessário avaliar a frequência de ameaças existentes em *softwares* com o mesmo propósito de domínio. Além disso, esse fator também é responsável por avaliar e medir riscos e impactos, dando maior precisão à equipe nas decisões de prioridade quanto às prevenções das ameaças.

Tendo a atividade de Análise como exemplo, o analista tem percepção sobre riscos e impactos aos ativos proporcionados por interoperabilidades ou colaborações externas da

aplicação. Apesar de estarem fora dos domínios da aplicação, estes podem comprometer os ativos da aplicação, pois atuam como intermediários com altos privilégios de acesso aos dados, ou figuram como indispensáveis na experiência com o usuário, a exemplo dos provedores de conteúdo e navegadores *Web*, respectivamente.

Papel de Garantir (*Ensure*): este papel é de suma importância para as atividades de *Design*, Codificação e Teste. O projetista precisa assegurar que os requisitos de segurança serão contemplados na arquitetura. Já o desenvolvedor deve estar atento ao fazer uso de bibliotecas com vulnerabilidades conhecidas ou que proporcionem falhas nas codificações nativas do *software*. Uma vez que ele entenda a semântica das ameaças, terá maior acuidade sobre possíveis vulnerabilidades em sua codificação. E durante os casos de teste, o engenheiro de teste deve ter a perspicácia de certos comportamentos que possam caracterizar ameaças não observadas pela equipe em outras fases do *software*. Isso auxilia na redução de tempo e custo quanto ao processo iterativo do *software* e, consequentemente, um produto mais seguro ao usuário final.

Esse papel reúne um aglomerado de participantes que podem ir mais além quanto aos processos de modelagem de ameaças, uma vez que recebem o *software* em uma versão conceitual ou mesmo experimental. Com ela, os engenheiros de teste podem analisar com maior rigor a experiência de uso do *software* aos seus usuários finais. A verificação guiada proporciona maior precisão e colaboração entre os envolvidos.

Papel de Supervisionar (*Supervisor*): Por fim, o quarto papel declara que todos os participantes das atividades de Teste, Implantação e Manutenção devem supervisionar o que foi desenvolvido pelas atividades anteriores. Esse papel tem o objetivo de prevenir que certas vulnerabilidades sejam exploradas no ambiente de produção, exercendo contramedidas de forma imediata durante a liberação do *software*.

3.3. Estudo de Caso

Também é proposta como contribuição a elaboração de um estudo de caso da modelagem ARES com a metodologia *"Comprehensive, Lightweight Application Security Process"* (CLASP) da OWASP através de um desenvolvimento Ágil de uma aplicação *Web*. Adicionalmente, é apresentada uma avaliação empírica entre a ARES e demais TMs existentes na literatura, identificando vantagens e desvantagens. A identificação das TMs é resultante de uma SLR sobre TMs aplicadas para a *Web*.

4. Estado Atual do Trabalho

Nesta seção é apresentado o estado das atividades no decorrer deste trabalho, bem como seus resultados. As etapas estão segmentadas de acordo com os artefatos da Tabela 1.

4.1. Construção da Teoria

Como medida de obter poder explanatório, a modelagem está sendo construída com base em teoria, que resulta em uma taxonomia em que inicialmente levantamos uma hipótese, posteriormente comprovamos certos fenômenos e por fim obtemos uma teoria fundamentada. A fundamentação da teoria é baseada em dados extraídos de um método misto entre uma SLR e uma avaliação em ambientes controlados, explanando assim o fenômeno do comportamento de vulnerabilidades de modo teórico e prático. Para tanto, foi desenvolvida uma metodologia em quatro etapas, conforme ilustração na Figura 3. Na etapa 1, foi

realizado um levantamento sobre terminologias, riscos e conceitos gerais sobre o tema de ameaças no ecossistema *Web*, obtendo-se uma teoria. Na etapa 2, foi desenvolvida uma SLR como processo de busca pelos estudos primários.

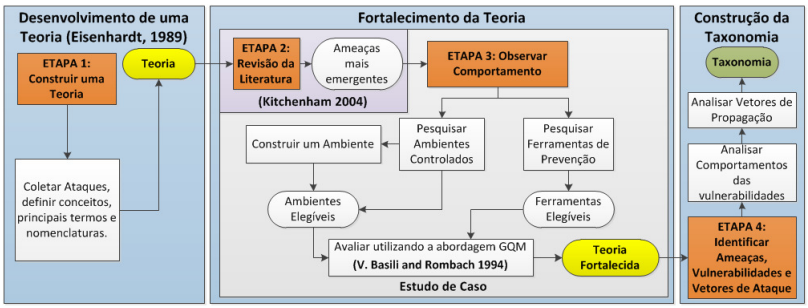

Figura 3. Fluxograma do Desenvolvimento da Taxonomia.

A busca foi realizada de acordo com o plano de pesquisa definido por um protocolo [Kitchenham 2004], a partir do qual pretende-se identificar trabalhos relacionados a ataques e vulnerabilidades no ambiente *Web*. O resultado dessa pesquisa retornou 1.005 artigos a partir de bibliotecas acadêmicas. Posteriormente foi realizada uma triagem em busca dos trabalhos mais relevantes, resultando em 289 artigos[1] selecionados para uma análise mais minuciosa sobre as soluções propostas. Na etapa 3, foi realizada uma avaliação de comportamento entre as ferramentas de prevenção e os ambientes que simulam exploração de vulnerabilidades. O resultado evidenciou certas lacunas, além de explanar comportamentos para o desenvolvimento da etapa 4, possibilitando a construção da Árvore de Ataques ilustrado na Figura 1.

4.2. Construção da Modelagem Baseada em Papéis

O próximo passo foi desenvolver outra revisão na literatura, no intuito de obter embasamento para o desenvolvimento da modelagem de acordo com as fases do *software*. A revisão bibliográfica visa responder questões sobre o uso de TM no ecossistema da *Web*. Tais questionamentos se enviesam em três vertentes: **Adoção**, que tem o intuito de investigar as TMs mais populares através de uma análise quantitativa; **Concepção**, que realiza uma análise taxonômica em cada TM, identificando diferenças e similaridades; e, por fim, **Eficácia**, que propõe submeter cada TM a um processo de avaliação considerando as vulnerabilidades mais emergentes da *Web*.

5. Comparação com Trabalhos Relacionados

TM é um tópico citado na literatura, a exemplo da STRIDE [Shostack 2014]. É uma modelagem centrada em *software*. O grande entrave é que modelagens desta natureza são fundamentadas pelo modelo cascata, o que dificulta a adoção de um modelo incremental que consequentemente favoreça a relação entre papéis distintos diante das fases do desenvolvimento. As modelagens PASTA [UcedaVelez and Morana 2015] e DREAD [Shostack 2014] são centradas em ativos, ou também conhecidas como centradas em risco. Nossa modelagem propõe uma abordagem diferenciada, centrada em vetores de ataques. Com isso, visa atingir um maior número de atividades e delegar responsabilidades mútuas e distintas de acordo com sua atuação no desenvolvimento.

[1]Resultados para a construção da taxonomia: http://bit.ly/1FhlsTo

6. Avaliação dos Resultados

A construção da taxonomia tem como base a Árvore da Figura 1 e a metodologia proposta da Figura 3. Também foram realizados alguns levantamentos bibliográficos sobre a utilização de ontologias formais como forma de obter respaldo nas decisões do modelo taxonômico. Para a Metodologia Baseada em Papéis, foi desenvolvida uma SLR para um levantamento preliminar sobre o tema relacionado à proposta. A pesquisa resultou em 653 artigos extraídos de sete fontes literárias. Até a escrita deste artigo, os resultados preliminares[2] encontravam-se na atividade Classificação.

7. Considerações Finais

Nosso principal objetivo é apresentar uma modelagem de ameaças que possa trazer reais benefícios para o desenvolvimento de aplicações projetadas para a *Web*, e ao mesmo tempo possa ser conduzida com eficiência em processos de metodologias ágeis. Contudo, é importante reconhecer que uma TM que promova ações em diversas atividades do *software* pode proporcionar um considerável esforço. Neste contexto, é importante saber medir a relação do esforço ante aos riscos e consequências. Como trabalhos futuros, concluída as SLR, o próximo passo será extrair seus resultados e dar sequencia ao processo da teoria fundamenta em dados. Em paralelo é proposto realizar uma pesquisa em empresas de desenvolvimento de *software* na área da indústria, comércio e tecnologia, no intuito de analisar o grau da adesão de TMs nesses ambientes.

Referências

Basili, V., Caldeira, C., and Rombach, H. D. (1994). Goal question metric paradigm. *Encyclopedia of Software Engineering, John Wiley and Sons, pp. 528-532.*

Beck, K. (2002). *Test-Driven Development by Example*. Addison-Wesley, 1 edition.

da Silva, C. M. R. and Garcia, V. C. (2014). Aegis: Um modelo de proteção à dados sensíveis em ambientes client-side. *Dissertação de Mestrado, CIn UFPE*.

Eisenhardt, K. M. (1989). Building theories from case study research. *Academy of Management Review*, 14(4):532–550.

Kitchenham, B. (2004). Procedures for performing systematic reviews. Technical report, Departament of Computer Science, Keele University.

Nebulon (2012). Fdd. *Disponível em: http://goo.gl/wPZd5R*.

OMG (2012). Mda. *Disponível em: http://www.omg.org/mda/*.

OWASP (2014). Asdr toc vulnerabilities. *Disponível em: http://goo.gl/15NGCz*.

Schneier (1999). Attack trees. *Dr Dobb's Journal, v.24, n.12. Retrieved 2007-08-16*.

Shostack, A. (2014). *Software Engineering*. Wiley, 1 edition.

Sommerville, I. (2010). *Software Engineering*. Pearson, 9 edition.

Symantec (2015). Internet security threat report. *Disponível em: http://goo.gl/vU4Rjp*.

UcedaVelez, T. and Morana, M. (2015). *Risk Centric Threat Modeling: Process for Attack Simulation and Threat Analysis*. Wiley, 1 edition.

[2]Resultado preliminar da SLR sobre adesão de TMs no ecossistema *Web*: http://bit.ly/1RoPr3b

Estudo e Definição de uma Estratégia Sistemática de Teste para Aplicações na Nuvem

Aluno: Victor Hugo Santiago Costa Pinto, email: victor.santiago@usp.br

Orientadora: Profa. Dra. Simone do Rocio Senger de Souza, email: srocio@icmc.usp.br

Colaborador: Prof. Dr. Paulo Sérgio Lopes de Souza, email: pssouza@icmc.usp.br

Nível: Doutorado.

Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional (ICMC-USP).

Ano de ingresso no programa: 2014/2.

Época prevista de conclusão: 2019/1.

Data da aprovação da proposta de tese/dissertação (qualificação): proposta ainda não aprovada. Previsão para janeiro de 2016.

Resumo: SaaS (*Software as a Service*) é um dos modelos de entrega de serviços na Computação em Nuvem e requer uma nova forma de desenvolvimento de aplicações. A Arquitetura *Multi-tenancy* é um padrão organizacional para SaaS que permite uma instância do software ser compartilhada entre vários *tenants*. Tenants são aplicações altamente customizáveis e suas execuções permitem aos usuários uma experiência isolada, de modo que se houver falha em um *tenant*, os demais não devem ser afetados. Apesar dos esforços de pesquisa nesse contexto, poucos trabalhos exploram o teste aplicado a esse domínio de problema. Neste sentido, este projeto de doutorado visa investigar as implicações desse novo modelo de computação para a atividade de teste, buscando propor uma estratégia sistemática de teste para SaaS que utilizam a arquitetura *Multi-tenancy*. O objetivo é definir ou empregar diferentes técnicas e critérios de teste visando aumentar a qualidade dessas aplicações.

Palavras-chave: Computação em Nuvem, SaaS, *Multi-tenancy*, teste de software.

Sigla(s) do(s) evento(s) do CBSoft relacionado(s): SBES.

Estudo e Definição de uma Estratégia Sistemática de Teste para Aplicações na Nuvem

Victor Hugo Santiago C. Pinto, Paulo S. L. Souza, Simone R. S. Souza

Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação - ICMC
Caixa Postal 668 – 13.560-970 – São Carlos – SP – Brasil

victor.santiago@usp.br, pssouza, srocio{@icmc.usp.br}

Abstract. *SaaS (Software as a Service) is one of the services delivery models in the Cloud Computing and it requires a new way of applications development. Multi-Tenancy Architecture is an organizational pattern for SaaS that allows a single software instance to be shared among numerous tenants. Tenants are applications highly customizable and their execution allow users to have an isolated experience, so that if there are faults in a tenant, the other tenants should not be impacted. Despite the research efforts in this context, few studies explore testing applied to this problem domain. Therefore, this project aims to investigate the implications of this new computing model for the testing activity, in order to propose a systematic test strategy to SaaS that use the Multi-tenancy architecture. The goal is to define or employ different techniques and test criteria aiming to increase the quality of these applications.*

Resumo. *SaaS (Software as a Service) é um dos modelos de entrega de serviços na Computação em Nuvem e requer uma nova forma de desenvolvimento de aplicações. A Arquitetura Multi-tenancy é um padrão organizacional para SaaS que permite uma instância do software ser compartilhada entre vários tenants. Tenants são aplicações altamente customizáveis e suas execuções permitem aos usuários uma experiência isolada, de modo que se houver falha em um tenant, os demais não devem ser afetados. Apesar dos esforços de pesquisa nesse contexto, poucos trabalhos exploram o teste aplicado a esse domínio de problema. Neste sentido, este projeto de doutorado visa investigar as implicações desse novo modelo de computação para a atividade de teste, buscando propor uma estratégia sistemática de teste para SaaS que utilizam a arquitetura Multi-tenancy. O objetivo é definir ou empregar diferentes técnicas e critérios de teste visando aumentar a qualidade dessas aplicações.*

1. Introdução

A computação em nuvem surgiu da contribuição de várias tecnologias e tem se tornado um dos principais objetos de pesquisa na Engenharia de Software por demonstrar um futuro promissor quanto à redução de custo, otimização e a oportunidade de criar novos modelos de negócios [Tai et al. 2010]. Nesse modelo, um conjunto de recursos pode ser acessado via Internet, de forma eficiente e sob demanda, no qual os provedores de serviço devem usar o mínimo de interação possível para gerenciar tal serviço [Mell e Grance 2010].

Segundo Vaquero et al. (2008), uma nuvem pode ser compreendida como um repositório de recursos virtualizados (*hardware*, plataformas de desenvolvimento/ou

serviços), facilmente acessíveis. Esses recursos podem ser reconfigurados dinamicamente para se ajustar a diferentes cargas de trabalho, otimizando sua utilização. Esse repositório de recursos é tipicamente explorado utilizando um modelo do tipo "pagamento-por-uso", em que os fornecedores de infraestrutura oferecem garantias customizadas no formato de SLAs[1] (*Service Level Agreements*).

Os modelos mais comuns de oferta de serviços em computação em nuvem são: SaaS (*Software as a Service*) - software como um serviço; o PaaS (*Platform as a Service*) - plataforma como um serviço; e o IaaS (*Infrastructure as a Service*) - infraestrutura como um serviço [Mell e Grance 2010]. SaaS refere-se às aplicações residentes na infraestrutura da nuvem e que são providas como um serviço aos consumidores por meio da Internet [Armbrust et al. 2010]. *Netflix*, *Google Apps* e *Facebook* são exemplos de SaaS.

Os ambientes de computação em nuvem são diferentes de um ambiente tradicional em termos de implantação, configuração, execução e gerenciamento das aplicações. A principal diferença está na comparação entre os tipos de usuários, arquitetura *Multi-tenancy* (AMT), segurança, etc [Chana e Chawla 2013]. *Multi-tenancy* é uma abordagem organizacional para aplicações SaaS, as quais são aplicações que permitem o compartilhamento dos mesmos recursos de *hardware*, por meio do compartilhamento da aplicação e da instância do banco de dados, enquanto permite configurar a aplicação para atender às necessidades do cliente como se estivesse executando em um ambiente dedicado [Bezemer e Zaidman 2010]. *Tenant* é uma entidade organizacional que aluga uma aplicação *multi-tenancy* e, normalmente, agrupa um número de usuários que são os *stakeholders* da organização.

As aplicações *tenants* são altamente customizáveis, por conta da definição de componentes *multi-tenancy* que permitem combinações para atender os requisitos desses grupos de usuários. Em termos técnicos, essa arquitetura exerce impacto sobre o código, uma vez que a implementação desses componentes deve ser separada da lógica dos *tenants* para evitar que a manutenção torne-se complexa. Ao utilizar essa arquitetura, uma série de benefícios podem ser destacados, como: otimização da utilização de recursos de *hardware*, facilidade e redução de custos com manutenção de aplicações e possibilidade de oferecer um serviço com preço mais baixo do que os concorrentes. Entretanto, existem desafios ligados ao tratamento das variações das aplicações *tenants* e a condução de testes, tais como: (i) realização de teste sob demanda [Tsai et al. 2013, Proko 2012], (ii) teste de regressão [Proko 2012], (iii) verificação das interações entre componentes desenvolvidos para os *tenants* [Sengupta e Roychoudhury 2011], (iv) verificação do impacto da inserção de novos *tenants* [Sengupta e Roychoudhury 2011] e (v) teste do sistema SaaS cobrindo diferentes *tenants* em execução [Sengupta e Roychoudhury 2011].

Segundo o relatório publicado pela IEEE [Alkhatib et al. 2014], a Computação em Nuvem é uma das 22 tecnologias inovadoras que podem mudar o percurso da indústria até o ano de 2022. No entanto, as normas para o desenvolvimento de aplicações ainda são dispersas, o desenvolvimento não segue uma abordagem sistemática e a comunidade ainda não possui padrões de portabilidade e interoperabilidade nesse paradigma. Além disso, a atividade de teste requer esforços substanciais, tanto da indústria quanto da academia para contornar as dificuldades até então, instransponíveis.

[1] http://searchitchannel.techtarget.com/definition/service-level-agreement

Nesse contexto, propõem-se a elaboração de uma estratégia de teste para atender os desafios anteriormente mencionados de forma eficiente, contribuindo significamente com a área de testes para SaaS considerando a AMT. Este artigo está organizado da seguinte forma: na Seção 2 aponta-se a fundamentação teórica acerca da computação em nuvem e teste de software. Na Seção 3 descreve-se a estratégia de teste proposta e as contribuições esperadas neste projeto. Na Seção 4 aponta-se o estado atual da pesquisa. Na Seção 5 discute-se os principais trabalhos relacionados com a proposta. Por fim, na Seção 6 descreve-se as conclusões preliminares e possíveis trabalhos futuros.

2. Fundamentação Teórica

2.1. Computação em nuvem

A computação em nuvem vem recebendo atenção significativa nos últimos anos, uma vez que transforma a maneira de prover serviços aos clientes. Essa tecnologia pode ser definida como um conjunto de *hardware*, redes, armazenamento, serviços e interfaces que são combinados e disponibilizados como serviço [Priyadharshini 2014]. Esse modelo permite alocação dinâmica e elástica de recursos usando uma combinação de técnicas de computação paralela, distribuída e tecnologias de virtualização de plataformas [Ru e Keung 2013].

O fornecimento de diferentes tipos de recursos reflete diretamente nos níveis de comportamento dinâmico e a diversidade das demandas dos usuários, tornando o gerenciamento de recursos complexo. Para que os provedores possam utilizar o potencial da nuvem em termos de escalabilidade e elasticidade, admite-se o modelo *Multi-tenancy* como um dos principais mecanismos. *Multi-tenancy* permite que múltiplas instâcias de uma aplicação ocupem e compartilhem recursos computacionais, armazenamento, etc, de modo que diferentes usuários possam ter sua própria versão da mesma aplicação executanto e coexistindo no mesmo *hardware*, mas isolada em espaços virtuais. No modelo *Multi-tenancy*, dados e recursos podem ser desenvolvidos na mesma nuvem. No entanto, os mesmos são controlados e diferenciados por meio de uma identificação única que estabelece a individualidade das propriedades dos usuários [Guo et al. 2007].

Dentre os modelos de entrega de serviço, tem-se o SaaS em que as aplicações residentes na nuvem são fornecidas como um serviço via Internet. Assim como as arquiteturas SOA e Cliente-Servidor, a AMT ou *multi-tenant*, multi-inquilino ou ainda multilocação está diretamente relacionada com o conceito de virtualização, que suporta o compartilhamento de computação, armazenamento e recursos de rede entre vários clientes. Em uma nuvem, um cliente (inquilino) poderia ser um usuário, um grupo de usuários, ou até mesmo uma organização/empresa [Chana e Chawla 2013]. Portanto, os termos cliente, inquilino ou *tenant* são sinônimos e podem ser definidos também como a entidade organizacional que aluga uma solução SaaS *multi-tenancy*. As aplicações *tenants* são altamente customizáveis com o apoio de componentes que são disponibilizados no SaaS. Por exemplo, a Salesforce.com[2] oferece um SaaS de gestão empresarial customizável que é utilizado para automação de vendas e conta com milhões de usuários.

[2]http://www.salesforce.com/sales-cloud/overview/

2.2. Teste de Software

O desenvolvimento de software inclui uma série de atividades e mesmo utilizando técnicas, métodos e ferramentas adequadas, defeitos no produto ainda podem persistir no *release* final. Para contornar essa dificuldade, a atividade de teste é uma das atividades mais empregadas para evidenciar a confiabilidade do software em complemento a outras atividades, como o uso de revisões e técnicas formais de especificação e de verificação. A atividade de teste consiste em um processo de execução de um programa com o objetivo de encontrar erros [Myers e Sandler 2004, Jovanovic 2009]. De modo geral, essa atividade apesar de ser altamente tendenciosa a erros e muitas vezes inviável se realizada manualmente, é uma atividade essencial no ciclo de vida de qualquer projeto de software [Delamaro et al. 2007].

No entanto essa atividade possui limitações, como o custo e a impossibilidade de considerar todos os dados de entradas possíveis. Desse modo, os desenvolvedores precisam estabelecer um critério para decidir quais os melhores casos de teste, levando em conta a probabilidade de revelar defeitos e, com isso, definir alguma condição de parada. Dentre os tipos de teste de software convencionais, pode-se destacar: (i) o teste funcional, (ii) o teste estrutural e (iii) o teste baseado em erros.

Na técnica funcional, a especificação do software é utilizada para derivar os requisitos de teste e envolve dois passos principais: identificar as funções que o software deve realizar e criar casos de teste capazes de verificar se essas funções estão sendo executadas corretamente. Nessa técnica, tem-se como critérios mais conhecidos: particionamento em classes de equivalência, análise do valor limite e grafo de causa-efeito [Delamaro et al. 2007]. A técnica estrutural baseia-se no conhecimento da implementação do programa. A maioria dos critérios dessa técnica utiliza uma representação de programa conhecida como grafo de fluxo de controle [Delamaro et al. 2007]. No teste baseado em erros, como é o caso do teste de mutação [Delamaro et al. 2007], a ideia consiste em gerar a partir de uma aplicação e por meio de operadores de mutação, um conjunto de programas ligeiramente modificados. Isso serve para averiguar o quanto um conjunto de casos de teste é adequado e suficiente. A partir disso, procura-se determinar um conjunto de casos de teste que consiga revelar, por meio da execução da aplicação, as diferenças de comportamento entre a aplicação original e seus mutantes.

3. Contribuição

Apesar das vantagens da AMT, existem vários desafios relacionados com o teste de sistemas SaaS. As técnicas tradicionais de teste de software não podem ser aplicadas em aplicações desenvolvidas para nuvem da mesma forma que em aplicações convencionais, visto que tais técnicas são projetadas sob a premissa que o software é um único *tenant* [Mahmood e Saeed 2013]. Vale ressaltar que também há desafios relacionados com desempenho, balanceamento de carga, entre outros testes não-funcionais que não são tratados neste projeto.

Existem vários fatores que impactam a atividade de teste para SaaS, por exemplo: a infraestrutura de aplicações tradicionais pode ser controlada, enquanto que para SaaS, ela é desestruturada e gerenciada pelo provedor de recursos da nuvem. Os componentes em aplicações tradicionais podem estar disponíveis no mesmo ambiente, mas para

um SaaS é comum que eles estejam dispersos em várias nuvens. O desenvolvimento de um SaaS requer o conhecimento e utilização de ferramentas e APIs específicas do provedor da plataforma [Mahmood e Saeed 2013]. Considerando as aplicações *tenants* que compõem o SaaS, os desafios envolvem a elaboração de métricas de qualidade, monitoramento, teste sob demanda e verificação do impacto da implantação de novos *tenants* [Alkhatib et al. 2014]. Para os componentes que são desenvolvidos para customizar as aplicações, abordagens para testar suas interfaces de composição, interações entre componentes e comportamento interno, requerem mais esforços de pesquisa [Zhang et al. 2012].

Assim, a principal contribuição deste projeto de doutorado é uma estratégia sistemática de teste de aplicações SaaS sob a perspectiva da AMT. O termo "sistemática" refere-se a sistematização da elaboração e seleção de casos de testes adotando vários critérios de teste instanciados para esse novo domínio de aplicações. Na Figura 1 ilustra-se uma visão geral da AMT e as atividades de teste pertencentes à estratégia. Na parte (A) tem-se uma representação do acesso de vários grupos de usuários aos *tenants* fornecidos pela aplicação SaaS, executada como uma única instância cujo balanceamento de carga é tratado pelo provedor dos recursos da nuvem. Na parte (B) representa-se a execução isolada dos *tenants* em que os recursos requeridos são compartilhados. Na parte (C), tem-se um repositório de componentes para customização dos *tenants*. As atividades de teste incluem: (i) teste de sistema, (ii) teste funcional, (iii) teste de integração, (iv) teste de regressão e (v) monitoramento. Vale ressaltar que o teste unitário está implícito na estratégia.

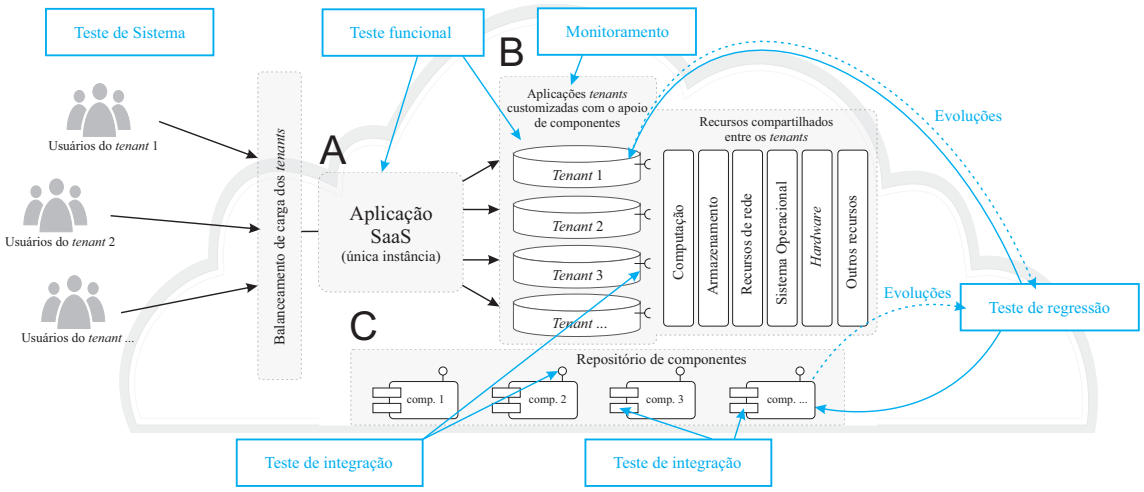

Figura 1. Estratégia sistemática de teste para SaaS

A ideia principal dessa visão geral é apontar quais tipos convencionais de teste podem ser instanciados formando a estratégia proposta. O teste de integração pode ser empregado para verificar a interação entre componentes e a composição deles com um determinado *tenant*. O teste de regressão pode ser utilizado para verificar o impacto da implantação de um novo componente e evoluções aplicadas em *tenants* e em componentes previamente disponibilizados. O teste de sistema pode ser utilizado para verificar a solução SaaS em ambiente de produção. O monitoramento da execução dos *tenants* pode contribuir com a identificação de falhas que são difíceis de serem reproduzidas em ambiente de desenvolvimento.

Como contribuições gerais, espera-se colaborar com a área de teste de software, estendendo os conceitos para um novo domínio de aplicações e possibilitar a transferência tecnológica, visto o interesse da indústria por técnicas e ferramentas de apoio ao teste no contexto de computação em nuvem tem crescido nos últimos anos.

4. Estado Atual do Trabalho

Uma Revisão Sistemática da Literatura está sendo concluída e com a mesma será possível alcançar uma visão completa e atual da AMT. Posteriormente pretende-se: (i) investigar a viabilidade da extensão de critérios de teste consolidados para aplicações na nuvem levando em conta essa arquitetura, (ii) definir um *benchmark* com um conjunto satisfatório de SaaS, (iii) refinar a estratégia de teste proposta e (iv) realizar avaliações dessa estratégia com o apoio do *benchmark* comparando com outros estudos.

5. Trabalhos Relacionados

Tsai et al. (2013) desenvolveram um algoritmo para geração de configurações em teste adaptativo com a finalidade de verificar a existência de falhas nos *tenants* com base nas interações de seus componentes. O algoritmo foi avaliado por meio de simulações. Como resultado, tornou-se possível encontrar falhas nas interações entre novos componentes e os previamente disponibilizados. No entanto, o estudo não inclui técnicas de teste para revelar defeitos nos *tenants* assumindo os desafios da AMT.

Zhang et al. (2012) utilizam a execução simbólica dinâmica (ESD) para gerar dados de entrada e estados de execução para o teste de aplicações na nuvem. Para isso, um modelo parametrizado é instanciado com os dados do estado requerido para testar um determinado conjunto de blocos de execução. Com a utilização da ESD, geram-se os dados correspondentes de forma sequencial com o intuito de exercitar determinados caminhos. A ESD pode contribuir com a estratégia proposta neste projeto, no sentido de atender eficazmente a atividade de teste funcional de sistemas SaaS.

6. Conclusão

A estratégia sistemática de teste proposta para sistemas SaaS deve fornecer diretrizes para que engenheiros de software possam aplicar técnicas convencionais de teste adaptadas para esse novo modelo de aplicações, considerando de maneira efetiva as implicações da arquitetura *multi-tenancy*. Após a validação dessa estratégia, pretende-se investigar como as técnicas de tolerância e predição de falhas poderiam também ser utilizadas no contexto dessas aplicações.

Referências

Alkhatib, H., Faraboschi, P., Frachtenberg, E., Kasahara, H., Lange, D., Laplante, P., Merchant, A., Milojicic, D., e Schwan, K. (2014). "IEEE CS 2022 report (draft)". Technical report, IEEE Computer Society.

Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., e Zaharia, M. (2010). A view of cloud computing. *Commun. ACM*, 53(4):50–58.

Bezemer, C.-P. e Zaidman, A. (2010). Multi-tenant saas applications: Maintenance dream or nightmare? In *Proceedings of the Joint ERCIM Workshop on Software Evolution (EVOL) and International Workshop on Principles of Software Evolution (IWPSE)*, IWPSE-EVOL '10, pages 88–92, New York, NY, USA. ACM.

Chana, I. e Chawla, P. (2013). Testing perspectives for cloud-based applications. In *Software Engineering Frameworks for the Cloud Computing Paradigm*, pages 145–164. Springer.

Delamaro, M. E., Maldonado, J. C., e Jino, M. (2007). *Introdução ao Teste de Software*, volume 394 of *Campus*. Elsevier.

Guo, C. J., Sun, W., Huang, Y., Wang, Z. H., e Gao, B. (2007). A framework for native multi-tenancy application development and management. In *E-Commerce Technology and the 4th IEEE International Conference on Enterprise Computing, E-Commerce, and E-Services, 2007. CEC/EEE 2007. The 9th IEEE International Conference on*, pages 551–558. IEEE.

Jovanovic, I. (2009). Software testing methods and techniques. *Transactions on Internet Research*, 26.

Mahmood, Z. e Saeed, S. (2013). *Software engineering frameworks for the cloud computing paradigm*. Springer.

Mell, P. e Grance, T. (2010). The nist definition of cloud computing. Technical report, National Institute of Standartization.

Myers, G. J. e Sandler, C. (2004). *The Art of Software Testing*. John Wiley & Sons.

Priyadharshini, V. e Malathi, A. (2014). Survey on software testing techniques in cloud computing. Technical Report abs/1402.1925., Cornell University Library (CoRR).

Proko, E. e Ninka, I. (2012). Analysis and strategy for the performance testing in cloud computing. *Global Journal of Computer Science and Technology Cloud & Distributed*.

Ru, J. e Keung, J. (2013). An empirical investigation on the simulation of priority and shortest-job-first scheduling for cloud-based software systems. In *Software Engineering Conference (ASWEC), 2013 22nd Australian*, pages 78–87. IEEE.

Sengupta, B. e Roychoudhury, A. (2011). Engineering multi-tenant software-as-a-service systems. In *Proceedings of the 3rd International Workshop on Principles of Engineering Service-Oriented Systems*, PESOS '11, pages 15–21, New York, NY, USA. ACM.

Tai, S., Nimis, J., Lenk, A., e Klems, M. (2010). Cloud service engineering. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 2*, pages 475–476. ACM.

Tsai, W.-T., Li, Q., Colbourn, C. J., e Bai, X. (2013). Adaptive fault detection for testing tenant applications in multi-tenancy saas systems. In *Cloud Engineering (IC2E), 2013 IEEE International Conference on*, pages 183–192. IEEE.

Vaquero, L. M., Rodero-Merino, L., Caceres, J., e Lindner, M. (2008). A break in the clouds. *ACM SIGCOMM Computer Communication Review*, 39(1):50.

Zhang, L., Ma, X., Lu, J., Xie, T., Tillmann, N., e De Halleux, P. (2012). Environmental modeling for automated cloud application testing. *Software, IEEE*, 29(2):30–35.

Um Método para Geração Otimizada de Testes a partir de Requisitos para Linhas de Produto de Software Dinâmicas

Aluno: Ismayle de Sousa Santos

Orientadores: Rossana Maria de Castro Andrade, Pedro de Alcântara dos Santos Neto

Nível: Doutorado

Programa: Mestrado e Doutorado em Ciência da Computação (MDCC) – Universidade Federal do Ceará (UFC) – Fortaleza, Ceará, Brasil

E-mail aluno: ismaylesantos@great.ufc.br

E-mail orientadores: rossana@great.ufc.br, pasn@ufpi.edu.br

Ano de Ingresso: 2013

Previsão de Conclusão: fevereiro de 2017

Data da aprovação da proposta de tese: não defendida ainda

Resumo*:*

Uma Linha de Produto de Software Dinâmica (LPSD) permite a geração de variantes de software em tempo de execução, lidando assim com variações dinâmicas nos requisitos ou no ambiente. Neste cenário, o principal foco desta pesquisa de doutorado é possibilitar a geração de testes de uma LPSD em dois momentos: nos estágios iniciais de desenvolvimento para favorecer a identificação precoce dos defeitos; e durante a execução dos produtos finais para testar a inclusão de novas features. Para isso, o objetivo desta tese de doutorado é propor um método para geração e priorização de testes, bem como uma ferramenta de apoio, a partir de requisitos para uma LPSD. A avaliação do método será feita por meio do teste de diferentes LPSDs, dentro de ambientes controlados.

Palavras-chave*: Linha de Produto de Software Dinâmica, Teste de Software, Caso de Uso, Otimização, Inteligência Computacional*

Eventos Relacionados: SBES, SBCARS

1. Introdução

De acordo com Northrop e Clements (2007), uma Linha de Produto de Software (LPS) pode ser definida como "um conjunto de sistemas de *software* que compartilham características comuns e que satisfazem as necessidades específicas de um segmento de mercado". Dessa forma, uma LPS maximiza o reúso e traz benefícios como melhor customização dos produtos e melhor *time-to-market* [Lopez-Herrejon *et al.* 2015].

Uma limitação da LPS é que os produtos gerados não podem lidar com variações dinâmicas nos requisitos e no ambiente do produto. Isso porque uma vez o produto configurado, o mesmo não pode ser reconfigurado em tempo de execução. Para tratar dessa limitação emergiram as Linhas de Produto de Software Dinâmicas (LPSD), que produzem sistemas capazes de ser adaptados em tempo de execução para atender dinamicamente novos requisitos ou mudanças de recursos [Parra *et al.* 2009]. Isso é possível porque uma LPSD pode suportar, dentre outras coisas, mudanças na variabilidade estrutural em tempo de execução [Capilla *et al.* 2014], como a inclusão, remoção ou alteração de uma funcionalidade específica.

Com respeito ao teste de uma LPS, este é mais complexo do que o teste de uma aplicação *standalone* porque deve levar em conta os pontos de variação da linha, bem como os dois processos de desenvolvimento (Engenharia de Domínio e Engenharia de Aplicação) [Mohamed e Moawad 2010]. Em uma LPSD, o desafio adicional à atividade de testes vem da dinamicidade. Além disso, uma das propriedades de uma LPSD é a sensibilidade ao contexto [Capilla *et al.* 2014, Hallsteinsen et al. 2008] que é utilizada para guiar a reconfiguração do produto. Dessa forma, os testes de uma LPSD além de verificarem se as *features* estão implementadas corretamente, precisam levar em conta as mudanças de contexto e testar se o produto está se reconfigurando como esperado.

Ressalta-se que como no paradigma de LPS várias aplicações são desenvolvidas com características comuns, a baixa qualidade dos artefatos do núcleo pode ser propagada para todos os produtos da linha [Mohamed e Moawad 2010]. Sendo assim, os métodos para suportar a geração de teste a partir de requisitos são importantes para manter a qualidade dos produtos de uma LPS.

Na literatura, podemos encontrar trabalhos que suportam a geração de teste para uma LPS a partir dos requisitos (e.g. os trabalhos de Neto *et al.* (2013) e Olimpiew e Gomaa (2009)). No entanto, no que diz respeito a uma LPSD, os autores não encontraram trabalhos sobre a geração de testes a partir de requisitos.

O objetivo desta pesquisa de doutorado é então propor um método de geração de testes a partir de requisitos, utilizando técnicas de otimização e de Inteligência Computacional (IC), para Linhas de Produto de Software Dinâmicas. Casos de uso especificando os requisitos da linha serão usados como artefato de entrada para gerar os testes. Escolheu-se tal artefato porque eles são amplamente utilizados e tem informações úteis para o propósito de testes. Além disso, muitas LPS usam casos de uso para documentar os requisitos [Alves *et al.* 2010]. Por fim, com o intuito de dar um melhor suporte a atividade de testes de uma LPSD serão propostos: a) novos critérios de cobertura de testes; e b) uma técnica para priorizar os testes de uma LPSD com o objetivo de otimizar a execução dos testes e a descoberta de defeitos.

2. Fundamentação Teórica

2.1 Linha de Produto de Software Dinâmica

Em uma LPS, a principal característica dos produtos é que eles são desenvolvidos a partir de um conjunto comum de artefatos, maximizando o reuso [Lopez-Herrejon *et al.* 2015]. No caso de uma LPSD, a diferença é que os produtos podem ser reconfigurados em tempo de execução [Parra *et al.* 2009]. Para tanto, uma LPSD apresenta as seguintes propriedades [Capilla *et al.* 2014, Hallsteinsen et al. 2008]:

- Suporte a variabilidade em tempo de execução. Uma LPSD deve suportar a ativação e desativação de *features* e mudanças na variabilidade estrutural que podem ser gerenciadas em tempo de execução;
- Múltiplos e dinâmicos *binding times* dos produtos. Em uma LPSD, *features* podem ser incluídas ou excluídas do produto várias vezes e em diferentes momentos (desde o tempo de desenvolvimento até o tempo de execução);
- Sensibilidade ao contexto: LPS dinâmicas devem usar informações de contexto como dados de entrada para mudar os valores das variantes do sistema dinamicamente e/ou selecionar novas opções do sistema dependendo das condições do ambiente;
- Lida com mudanças inesperadas, bem como mudanças pelos usuários, tais como requisites funcionais ou qualidade; e
- Foca em uma situação do ambiente (contexto) ao invés de um mercado.

2.2 Teste Baseado em Busca

O teste de *software* pode ser definido como a verificação dinâmica do funcionamento de um programa comparando seu comportamento real com o esperado [IEEE 2004]. Segundo McMinn (2011), o Teste de Software Baseado em Busca é o uso de técnicas baseadas em meta-heurísticas de otimização de busca para automatizar ou parcialmente automatizar uma tarefa de teste (e.g. geração de dados de testes). Dessa forma, problemas complexos da atividade de teste (e.g. priorização de testes procurando maximizar cobertura e minimizar o custo) podem ser modelados matematicamente e soluções ótimas ou próximo do ótimo podem ser encontradas em um tempo aceitável.

Neste cenário, a atividade de otimização representa maximizar ou minimizar uma função matemática definida a partir de coeficientes e variáveis, que podem estar sujeitas a restrições [Freitas et al 2010]. Uma evolução nesta área foi a incorporação não apenas de técnicas de busca baseadas em otimização, mas também a introdução de diversas técnicas de inteligência computacional (e.g. Lógica *Fuzzy*), tendo assim ampliado o número de alternativas para a solução dos problemas existentes.

3. Resultados esperados e contribuições

Conforme descrito na Seção 1, esta pesquisa de doutorado tem como objetivo propor um método para geração de testes a partir de requisitos de uma LPSD. Neste cenário, este trabalho tem as seguintes contribuições esperadas:

a) Um método para geração de testes para uma LPSD, de forma que os testes gerados validem tanto as *features* como as reconfigurações dos produtos. Os testes serão gerados a partir de um modelo de *features* enriquecido com informações de contexto e de casos de uso, permitindo assim a geração de testes a partir de requisitos. Além disso, o método deverá utilizar técnicas de otimização e de IC para a seleção e priorização dos testes levando em conta diferentes atributos da LPSD;

b) Critérios de cobertura de testes específicos para uma LPSD. Tais critérios levarão em conta o uso de informações de contexto e o reúso dos testes;

c) Uma ferramenta de apoio à especificação de casos de uso de uma LPSD, geração de testes, e rastreabilidade entre *feature*, contexto, caso de uso e testes; e

d) Um *template* de caso de uso para LPSD. Espera-se que com este template, seja possível fornecer a todos os envolvidos com uma LPSD um entendimento maior sobre a influência do contexto na reconfiguração dos produtos da linha.

4. Trabalhos Relacionados

Com o objetivo de auxiliar a descoberta de defeitos nas fases iniciais do desenvolvimento de uma LPS, algumas abordagens são encontradas na literatura para suportar a geração de teste para uma LPS a partir de casos de uso [Neto *et al.* 2013, Olimpiew e Gomaa 2009, Santos 2013, Bertolino e Gnesi 2003].

O método PLUTO [Bertolino e Gnesi 2003] é uma extensão da partição Método de Partição de Categorias para derivar testes a partir de requisitos da LPS descritos como PLUCs (Product Line Use Case). O CADeT [Olimpiew e Gomaa 2009] é um método de testes baseado em modelo e orientado a *feature* para criar especificações de testes a partir de casos de uso e modelos de *features* baseados em diagramas de atividade e tabelas de decisão. O SPLMT-TE [Neto *et al.* 2013] é uma ferramenta que gera cenários de teste de sistema cenários de casos de uso de uma LPS. Finalmente, o método proposto por Santos (2013) estende o método PLUTO para gerar testes a partir de casos de uso para LPSs no domínio de aplicações sensíveis ao contexto.

Em relação a LPSD, Cafeo *et al.* (2011) apresentam uma abordagem para inferir o grau de confiança de uma configuração não testada com base na similaridade desta configuração com outras configurações previamente testadas. A similaridade é estabelecida com base em um grafo criado a partir do código fonte da LPSD.

Por fim, existem propostas para modelar a variabilidade em casos de uso [Santos et a. 2013, Choi *et al.* 2008], que podem ser utilizados para fins de teste. No entanto, apenas o *template* de Santos (2013) permite especificar as variabilidades da LPS e informação de contexto junto aos cenários de caso de uso.

O diferencial deste trabalho está em tratar do teste de uma LPSD a partir do modelo de *features* e dos casos de uso, não necessitando do código-fonte. Além disso, embora o *template* de caso de uso proposto Santos (2013) seja aplicável para uma LPSD, ele precisa ser modificado para tratar das suas fraquezas, como a ausência de suporte à especificação de variabilidade nos atores, e para especificar as *features* da LPSD junto com os cenários do caso de uso. Com essa última, mudanças na variabilidade estrutural da LPSD podem ser refletidas nos casos de uso do produto.

5. Estado Atual do Trabalho

A execução deste trabalho está dividida em oito atividades: (I) Estudo sobre *templates* de casos de uso para LPS; (II) Proposta de um *template* de caso de uso para LPSD; (III) Estudo sobre o teste de LPS e LPSD; (IV) Estudo sobre a aplicação de técnicas de otimização e IC na atividade de teste; (V) Definição de um método de geração de teste para LPSD; (VI) Definição de uma técnica de priorização dos testes; (VII) Implementação da ferramenta de apoio; e (VIII) Validação dos resultados.

Dado a ausência de trabalhos sobre casos de uso próprios para LPSD, foi feita uma investigação da especificação de requisitos de uma LPS com casos de uso (atividade I) por meio de um mapeamento sistemático. Com este estudo [Santos *et al.* 2014] foram identificados nove *templates* de Caso de Uso para LPS. Com base nos resultados deste estudo foi definido uma adaptação do *template* de caso de uso proposto por Santos (2013) para lidar com LPSD (atividade II).

Durante a atividade III, o autor investigou as pesquisas relacionadas ao teste de LPS e LPSD. Uma vez que a LPSD lida com informações de contexto, também está sendo investigado as técnicas de *design* de teste aplicadas nos testes de aplicações sensíveis ao contexto por meio de uma Revisão Sistemática conduzida no contexto do projeto CAcTUS [1]. Isso porque acredita-se que tais técnicas podem ser adaptadas para o teste de uma LPSD. Além disso, o autor tem estudado a aplicação de diversas técnicas de otimização e de IC na geração e priorização de testes (atividade IV).

As próximas atividades dizem respeito a definição do método de geração de testes (atividade V) e de uma técnica de suporte à priorização dos testes (atividade VI).

Como o foco desta pesquisa é o teste de uma LPSD, os seguintes desafios devem ser tratados: a) o que vai ser testado na Engenharia de Domínio (ED) e na Engenharia da Aplicação (EA); b) a quantidade de testes possíveis é muito grande visto que em uma LPS diferentes combinações de *features* podem ser feitas para gerar centenas ou até milhares de produtos diferentes; e c) como testar a reconfiguração do produto, pois na LPSD as *features* podem ser ativadas ou desativadas em tempo de execução.

Para lidar com o primeiro desafio, o método proposto irá envolver as duas fases de desenvolvimento da LPS, guiando o teste de cada fase. A ideia é que na ED o foco seja o teste da corretude e interação das *features* da linha, enquanto que o foco dos testes da EA seja verificar se está ocorrendo a reconfiguração correta do produto e se novas *features* inseridas em tempo de execução interagem de forma correta com o produto.

Com relação a explosão do número de testes, pretende-se usar técnicas de otimização para sugerir um subconjunto representativo de produtos da LPSD capaz de relevar falhas nas interações entre as *features,* considerando as mudanças de contexto. Com isso, espera-se minimizar os custos dos testes e maximizar a detecção de defeitos. A geração de um subconjunto representativo de produtos reduz o tamanho da suíte de testes e tem sido bastante utilizada para o teste de uma LPS, conforme o Mapeamento Sistemático de Lopez-Herrejon *et al.* (2015). Também almeja-se propor novos critérios

[1] Context Aware Testing for Ubiquitous Systems (CAcTUS) é um projeto de pesquisa envolvendo pesquisadores de três universidades (Universidade Federal do Rio do Janeiro, Universidade Federal do Ceará, e University of Valenciennes and Hainaut-Cambresis).

de cobertura, específicos para o teste de um LPSD, para guiar a geração dos testes. Além disso, por meio de informações da linha e dos testes gerados pretende-se priorizar os testes utilizando técnicas de otimização e de IC para maximizar a descoberta de defeitos amenizando os custos.

Por fim, para tratar do teste da reconfiguração do produto, a ideia é usar um *Label Transaction System* (LTS) na EA de forma que ele represente as possíveis reconfigurações do produto, para a partir dele os testes da reconfiguração do produto serem gerados. Neste ponto, o desafio é a modelagem das reconfigurações do produto dado a presença de informações de contexto e a explosão de estados.

A Figura 1 apresenta uma visão geral da proposta integrando o resultado das atividades II, V e VI. Na ED, o método inicia com casos de uso da LPSD, que relacionam *features*, funcionalidades e informações de contexto. A partir daí, será possível aplicar critérios de cobertura para selecionar os casos de uso que serão usados para gerar o conjunto de cenários de testes das *features* da linha.

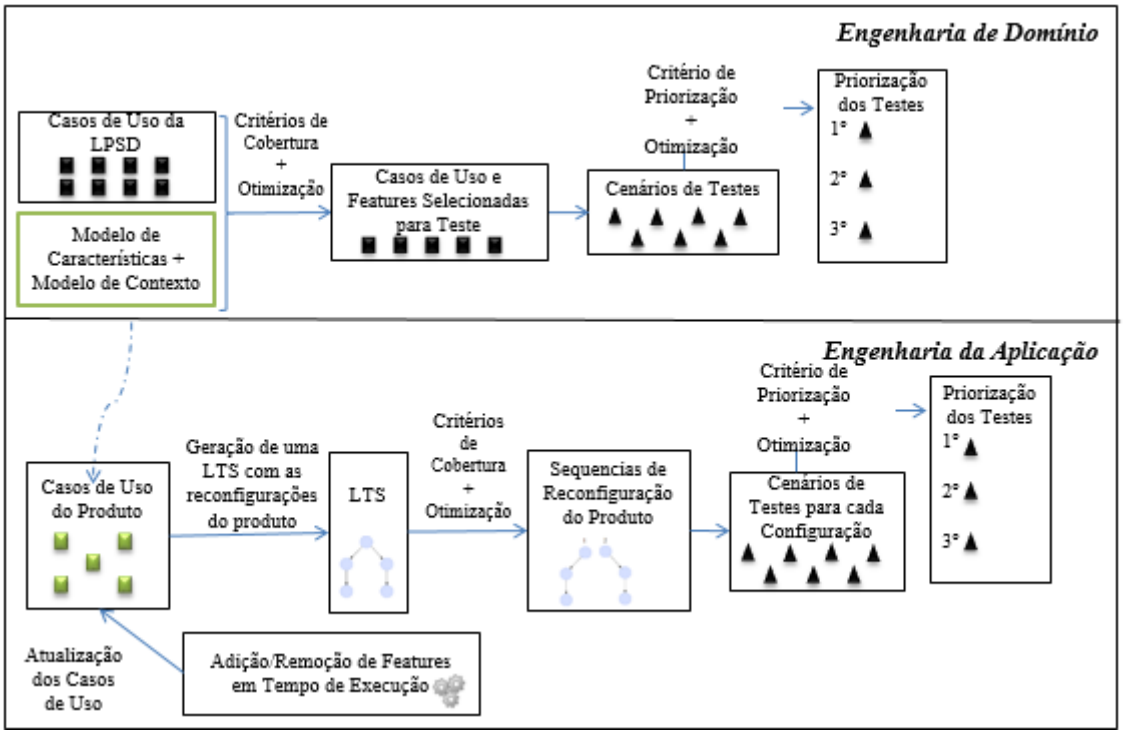

Figura 1. Visão geral do método de geração de testes para uma LPSD

O último passo do método na ED é a priorização dos testes. Essa priorização levará em conta diferentes critérios. Dentre eles, pode-se citar a complexidade das *features* cobertas, o histórico de falhas e a complexidade dos testes. Outros critérios devem ser incluídos após o fim das atividades III e IV.

Na EA o método inicia com os casos de uso da aplicação (ver Figura 1) que são criados a partir dos casos de uso da linha. Na ferramenta que será desenvolvida é esperado o suporte para a instanciação dos casos de uso da aplicação, bem como a exportação destes casos de uso seguindo o *template* definido na atividade II. A partir dos casos de uso da aplicação será gerado uma LTS do produto, onde cada estado corresponde a uma configuração do produto com um conjunto de *feature* ativas e as

transições entre os estados correspondem às mudanças de contexto ou mudanças de requisitos (e.g. reconfigurações disparadas pelo usuário).

Uma vez que muitos estados podem ser automaticamente gerados, pode-se aplicar critérios de cobertura e selecionar um subconjunto de reconfigurações, na forma de caminhos do LTS, para testar. Aqui será também possível especificar probabilidades para as mudanças de contexto, de forma a possibilitar a geração de testes com base na probabilidade de mudança do contexto. Para cada estado um conjunto de cenários de testes será gerado permitindo-se verificar se a reconfiguração ocorreu conforme esperado. Por fim, pretende-se investigar o uso de técnicas de IC (e.g. Lógica Fuzzy) para auxiliar a priorização dos testes.

Ressalta-se que uma LPSD possibilita que *features* sejam adicionadas ou removidas do produto em tempo de execução [Capilla *et al.* 2014]. Neste caso, com base nos casos de uso atualizados (ver Figura 1), o método a ser proposto deverá gerar novos cenários de teste para avaliar o impacto da mudança no produto.

A ferramenta será desenvolvida para integrar toda a abordagem (Atividade VII). Por fim, a validação (Atividade VIII) será conduzida após o término do desenvolvimento da ferramenta conforme descrito na Seção 6.

6. Metodologia de Avaliação dos Resultados

A viabilidade do método será demonstrada por uma prova de conceito usando a linha Mobiline [Marinho *et al.* 2012]. Além disso, utilizando a linha Mobiline e a técnica de Análise de Mutantes [IEEE 2004], experimentos controlados serão conduzidos para comparar o método proposto, com respeito a quantidade de defeitos encontrados e número de testes gerado, com outros métodos propostos para LPS (ver Seção 4). Experimentos também devem ser conduzidos para averiguar o quão ótimo são as soluções encontradas e o tempo de execução do algoritmo para sugestão do subconjunto da LPSD a ser testado. Tais experimentos serão conduzidos com a linha Mobiline, com outras linhas publicadas em artigos científicos e linhas criadas artificialmente a partir destas. Ressalta-se que na ausência dos casos de uso, estes serão criados artificialmente pelo autor para a avaliação do método a ser proposto. Por fim, será conduzida uma avaliação da facilidade de uso e escalabilidade da ferramenta a ser desenvolvida.

Referências

Alves, V., Niu, N., Alves, C., e Valença, G. (2010). Requirements engineering for software product lines: A systematic literature review. Inf. Softw. Technol., 52(8):806–820.

Bertolino, A. e Gnesi, S. (2003). Use case-based testing of product lines. In Proceedings of the 9th European software engineering conference, pg 355–358, NY, USA. ACM.

Cafeo, Bruno B. P.; Noppen, Joost; Ferrari, Fabiano C.; Chitchyan, Ruzanna and Rashid, Awais (2011) Inferring test results for dynamic software product lines. In *European conference on Foundations of software engineering*. NY, USA, 500-503.

Capilla, R.; Bosch, J.; Trinidad, P.; Cortes, A. R.; Hinchey, M.(2014). An overview of dynamic software product line architectures and techniques: Observations from research and industry. Journal of Systems and Software, 91:3-23.

Choi, W., Kang, S., Choi, H., e Baik, J. (2008). Automated generation of product use case scenarios in product line development. In Proceedings of the 8th IEEE International Conference on Computer and Information Technology.

Hallsteinsen, S.; Hinchey, M.; Sooyong Park; Schmid, K.. (2008) Dynamic Software Product Lines, *Computer* , vol.41, no.4, pp.93-95.

IEEE (2004) Guide to the Software Engineering Body of Knowledge, IEEE Computer Society.

Freitas, F. G.; Maia, C. L. B.; Campos, G. A. L.; Souza, J. T. (2010). Otimização em Teste de Software com Aplicação de Metaheurísticas. Revista de Sistemas de Informação da FSMA, n. 5, pp. 3-13.

Lopez-Herrejon, Roberto E.; Fischer, Stefan; Ramler, Rudolf; Egyed, Alexander, (2015) "A first systematic mapping study on combinatorial interaction testing for software product lines," *2015 IEEE Eighth International Conference on Software Testing, Verification and Validation Workshops (ICSTW),* vol. 1, no. 10, pp. 13-17.

Marinho, F. G. ; Andrade, R. M. C. ; Werner, c. ; Carvalho, W. V.; Maia, M.; Rocha, . L. S; Dantas, V. L. L. ; Lima, F. F. P. ; Aguiar, s. B. (2012) MobiLine: A Nested Software Product Line for the domain of mobile and context-aware applications. Science of Computer Programming (Print), v. 77, p. 1-18.

McMinn, Phil (2011) Search-Based Software Testing: Past, Present and Future. In *Proceedings of the 2011 IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops* (ICSTW '11). IEEE, DC, USA, 153-163.

Mohamed Ali, M. e Moawad, R. (2010). An approach for requirements based software product line testing. In Proceedings of the 7th International Conference on Informatics and Systems, INFOS ’10, pages 1–10.

Neto, C. R. L.; Machado, I. C.; Almeida, E. S.; Garcia, V. C. (2013) Analyzing the Effectiveness of a System Testing Tool for Software Product Line Engineering. International Conference on Software Engineering and Knowledge Engineering, 584-588.

Northrop, L. M. e Clements, P. C. (2007). A framework for software product line practice, version 5.0. Technical report, CMU/SEI - Software Engineering Institute.

Olimpiew, E. and Gomaa, H. (2009) Reusable Model-Based Testing, Proc. 11th Internation-al Conference on Software Reuse, Falls Church, VA, 76-85.

Parra, Carlos; Blanc, Xavier; Duchien, Laurence (2009) Context awareness for dynamic service-oriented product lines. In *Proceedings of the 13th International Software Product Line Conference* (SPLC '09). Pittsburgh, PA, USA, 131-140.

Santos, I. S. (2013) Um Ambiente para Geração de Cenários de Testes para Linhas de Produto de Software Sensíveis ao Contexto. Dissertação de Mestrado. Universidade Federal do Ceará.

Santos, I. S.; Andrade, R. M. C.; Santos Neto, P. (2014) How to Describe SPL Variabilities in Textual Use Cases: A Systematic Mapping Study. In: 8th Brazilian Symposium on Software Components, Architectures, and Reuse (SBCARS).

An Approach to the Design of Adaptive and Normative Software Agents

Nome do Aluno: Marx Leles Viana

Nome do Orientador: Carlos José Pereira de Lucena

Nível: Doutorado

Programa de Pós-Graduação: Pontifícia universidade Católica – PUC-Rio – Rio de Janeiro, RJ - Brasil

Email dos Contatos:

Aluno: mleles@inf.puc-rio.br

Orientador: lucena@inf.puc-rio.br

Ano de Ingresso: 2012/2

Ano Previsto de Conclusão 2016/2

Data da Qualificação: 12/12/2015

Eventos Relacionados: SBES

Abstract. *As a new paradigm of building and software modeling, multi-agent systems require modeling languages exploiting its benefits and characteristics. However, software engineering based agents have not been widely adopted, mainly due to lack of modeling languages that explore the use of agent-related abstractions and are responsible for refining the design models to code. In addition, most modeling languages do not define how these abstractions interact at runtime, but many software applications need to change their behavior and react to changes in their environments dynamically. We observed that most languages do not represent some important concepts such as adaptation and normative. Nevertheless, many types of applications currently need to support adaptive and normative mechanisms, including, emergency services and natural disasters (eg, floods, fires, landslides, earthquakes) homes and smart cities, construction of virtual heritage and health (eg, diseases or disorders involving the immune system such as HIV).This doctoral research proposes a novel approach to modeling and implementing adaptive normative software agents.*

Keywords: *— Modeling language, adaptation, norms, multi-agent systems.*

1. Introduction

Agent-Oriented Software Engineering (AOSE) emerged as a new technology for building complex systems. These systems are characterized by being distributed and composed of autonomous entities that interact with each other [Wooldridge 2011]. Multi-Agent Systems (MAS) are societies in which these heterogeneous and individually designed entities (agents) work to accomplish common or independent goals [López 2003]. Thus, the use of agents for construction of such complex systems is considered a promising approach [Zambonelli 2003]. However, MAS has not been widely adopted, mainly due to lack of modeling languages that support the use of agent-related abstractions and promote the refinement of design models to code [Silva 2004]. In addition, most modeling languages do not define how these abstractions interact at runtime [Beydoun 2009], but many software applications need to change their behavior and react to changes in their environments dynamically.

In general, modeling languages should represent the structural and dynamic aspects of software agents, expressing the essential characteristics of the agent-related entities. The structural aspects incorporate the definition of the entities, their properties and their relationships. Several authors recognized the importance of modeling agents in their environment both in design time and in runtime [Silva 2004], [Bresciani 2004], [Beydoun 2009]. These authors also observed that most modeling languages do not represent some important concepts present in SMAs, for example, adaptation and norms [Beydoun 2009], [Hollander 2011]. Nevertheless, many types of applications currently need to support adaptive and normative mechanisms, including, emergency services and natural disasters (eg, floods, fires, landslides, earthquakes) [Beamon 2006] [Cerqueira 2009], [Janssen 2010], homes and smart cities [Cook 2012], [Atzori 2011], [Gubbi 2013], construction of virtual assets (eg, Second Life [Bogdanovych 2011]) and health (eg, diseases or conditions involving the immune system such as HIV [Zang 2005], [Laroum 2011]).

2. Background

This section describes the main characteristics of the concepts related to adaptation and norms. First, we will discuss what is adaptation, how it occurs and how software agents can adapt. We will also discuss what norms are and how they are understood by the agents. Then the ability of norms to adapt in a multi-agent system and the impact of norms on adaptation will be discussed.

The word adaptation means "any change in the structure or functioning of an organism that makes it better suited to its environment" [Oxford Dictionary of Science 2015]. Therefore, according to this definition, the adaptation of agents has two levels of action: structural change and behavioral change. These changes aim to make applications more adapted to their evolutionary context [Da 2011].Generally, there are three types of adaptation [Chefrour 2005]: (i) reactive adaptation, in which the agents will change their behavior due to the execution of changes in their environment; (ii) evaluative adaptation, which aims to extend the functionality of software systems, fix bugs or to enhance their performance; and (iii) adaptation of integration, which occurs when the integration of services or components become incompatible due to other adjustments made to the system.

The general adaptation cycle of a software agent includes environmental monitoring, adaptation selection, and adaptation implementation. This cycle is explained in four different phases: Collection, Analysis, Decision and Execution. In this cycle, in the case of MAS, the system is fully autonomous. First, the agent data Collection activity gathers the information resulting from environmental monitoring. Next, the Analysis activity evaluates the context data and produces an adaptation plan for adaptation, which consists of a single plan or a list of plans. In the Decision activity the risk of each plan will be assessed and the adaptation will occur. Finally, in the Execution activity the adaptation will be performed in accordance with what was described in the chosen plan. The core of the adaptation involves the Analysis and Decision activities.

Regarding the concept of norms, Webster online dictionary [Merriam-Webster 2015] offers three definitions: (i) an authoritarian pattern; (ii) a principle of right actions associated with the members of a group that is used to guide, control or regulate proper and acceptable behavior; and (iii) a weighting. A norm is treated as: (a) a standard set of development or achievement, usually derived from the average or median performance of a large group; (b) a standard understood as the typical behavior of a social group; and (c) a general or usual practice. These definitions are meanings of the term norm that are used in various areas in normative research, including deontic logic, law theory, sociology, social psychology and social philosophy, decision theory and game theory [Verhagen 2000].

When applied to multi-agent systems, norms are understood as mechanisms to regulate the behavior of agents, representing the way in which agents understand the responsibilities of other agents [López 2003]. Thus, the agents work under the belief that other agents will behave according to the prescribed norms, but above all, norms are mechanisms that enable agents to require the other agents to behave in a certain way [Santos Neto 2010]. Thus, the use of norms is a necessity in multi-agent systems in which the members are autonomous, but not self-sufficient, and, therefore, cooperation is required but cannot be assured without the specific introduction of specific mechanisms that support it.

Knowing that norms are mechanisms that society has to influence the behavior of agents [Bogdanovych 2011], norms can be used to achieve different purposes, ranging from the construction of a simple agreement between agents to more complex cases involving the legal system [López 2002]. The norms may persist for different periods of time, for example, while the agent remains in the society, or just for a short period of time until the social objective has been achieved [Lopez 2002]. Therefore, different aspects can be used to characterize norms. First, norms are always created to be met by a particular set of agents in order to achieve specific social goals. Second, norms are not always applied, and their activation conditions depend on the context of the agents involved in a specific interaction. In addition, the type of element that the norm regulates must be known; if it is an action of the agent, or a state of the environment. Finally, in some cases, norms may provide a set of sanctions to be imposed when agents fulfill or violate certain norms [López 2002], [López 2003], [Santos Neto 2010].

An aspect of a norm that needs to be emphasized in the MAS area is its dynamic nature and its tendency to undergo changes over time [Neumann 2008]. An examination

of human society produces clear examples of this phenomenon: the norms of a generation are rarely identical to the norms of the next. This leads us to the concept of adaptive norms. Adaptive norms refer to the process by which the norms of a system change over time [Hollander 2010]. In theory, this is done through the use of social learning processes, such as copying (imitation) and socializing. Once a new norm is created, it has the potential to emerge in certain circumstances and, if it happens, the norm will compete or replace other existing norms, which are internalized in the same context. This type of adaptation occurs at the macro level. Alternatively, a norm can be modified when it is acquired by an agent [Hollander 2010]. During the transmission phase, the polarization of the receiving agent may result in subtle changes. This is the main concern in systems where the norm may not be copied directly. The changes that occur during transmission and internalization of norms are adaptations at the micro level.

3. Modeling Languages

Although some modeling languages involve adaptation abstractions and norms implicitly in the agent design time [Bresciani 2004], [Zambonelli 2003], these approaches do not explicitly show how these abstractions influence both the design time and runtime behavior of the agents and their reactions to environmental changes. Due to this limitation, these languages cannot be used to model many essential aspects in the representation of agents such as those aspects involving adaptation, norms and the interactions between them. For example, certain adaptations may require that the norms are adaptive, that is, that they change over time the applicability, or even their existence. On the other hand, norms can also have an impact on adaptation, since some adaptations must meet some normative criteria (e.g., obligations, permissions, prohibitions).

In recent years, various modeling languages have been proposed for MASs [Silva 2008], [Zambonelli 2003] [Beydoun 2009] [Cervenka 2007] [Bernon 2002]. However, there is still the need for a modeling language that describes concepts related to adaptation and norms and: (i) supports these concepts as first-class citizen abstractions; (ii) supports these concepts through an explicit description of a MAS metamodel; (iii) can be used to model the structural and dynamic aspects often described in MAS for these concepts; and (iv) promote the refinement of these models from design to code.

4. Proposed Approach

Based on these aspects, the objective of this doctoral research is to create an approach that supports the abstractions of adaptive and normative agents (i.e., agents capable of reasoning about a norm addressed to them and adapting). This approach essentially involves: (i) a metamodel for representing MASs; (ii) a generative process based on models for the implementation of the metamodel; and (iii) the introduction of techniques for assessing the metamodel and the generative process.

A metamodel defines a language for specifying models describing the semantics of a set of abstractions and defining how these abstractions are instantiated [UML 2015]. For each abstraction, the metamodel should describe its semantics, the meta-relationship with other abstractions and the graphical representation of the abstraction.

The modeling languages should describe the graphical representation of the new abstractions, their semantics and the relationships between them [Silva, 2004].

A modeling language for MASs should include structural diagrams to model the structural aspects of a system. The set of structural diagrams need to be able to model: (i) the entities usually defined in a MASs, (ii) the properties of these entities, associating properties with specific entities, and (iii) the relationship between the entities. The modeling languages proposed in the literature do not model explicitly adaptations and norms and therefore do not define the relationships between agents and those entities.

The development of appropriate approaches to implement agent-based systems is a key issue that needs to be addressed when agent technology is used in the development of a software system. In order to implement the multi-agent systems developed using a specific modeling language, it is necessary to transform the MAS design models into code. These models are models composed of high-level agent-related abstractions. To turn these models into code, the agent-related abstractions must be mapped into abstractions defined in a certain programming language.

To demonstrate the need for representing adaptation and norms, two application scenarios will be described. The first scenario relates to emergency services and natural disasters. The purpose in this case is to evacuate people from hazardous areas where civilians may be in danger and firemen aim at rescuing these civilians [Cerqueira 2009]. A second application scenario involves the immune system of the human body and its interaction organisms that cause diseases [Zhang 2005].

In the context of evacuation of people in hazardous areas, the ideal outcome would be for firemen to rescue and provide care to the injured. For this purpose, the fire department has a limited number of resources that are regulated by the firemen chief. This regulation is made by means of norms that restrict the behavior of the firemen so that the rescue is undertaken in a coordinated manner and with the best use of resources. For example, these resources can be aircraft, ground vehicles and excavation equipment. Furthermore, in terms of adaptation, the environmental conditions may change during a particular rescue due to weather instability and landslides, among other factors. Thus, there is need for firemen to be able to adapt to different conditions to carry out the rescue.

In relation to the second scenario, it is necessary to represent the interaction dynamics between the immune system and the organisms that cause the diseases. The body reacts to daily attacks of bacteria, viruses and other microbes, through the immune system. Any of these invading organisms can rapidly evolve and adapt, trying to avoid detection and neutralization by the immune system. As a result, various immune system defense mechanisms have also evolved to recognize and neutralize these organisms. By acquiring immunity, the body creates an immune memory following an initial response to a specific agent, which allows it to respond more effectively to new attacks by the same agent. In addition, the elements of the immune system must follow certain norms and strategies to combat the invaders.

To facilitate the use of agents capable of representing adaptation and norms, we will be developed a tool to help build these models, enabling the refinement of design models into code.

5. Expected Contribution

This doctoral research assumes that introducing abstractions related to adaptation and norms in MAS representation languages results in a new metamodel that supports these abstractions, a generative process for implementing these metamodels, and the use of the metamodel and the generative process in different application domains. The main contributions of this research include:

- A metamodel to define the interactions that can occur between the entities present in MASs with adaptation and norm-related abstractions;
- A modeling language that incorporates adaptation and norms, which are particular characteristics of MASs that are not addressed in UML and are not satisfactorily addressed in the proposals available in the literature;
- Design guidelines to help designers to use these new concepts in the proposed language;
- A grammar of the language, formalizing the syntax of the structural and dynamic aspects, and visual diagrams;
- A generative process as well as a tool to refine the structural and dynamic diagrams of the modeling language and generate code in a specific programming language.

6. Evaluation

The approach proposed in this study will be evaluated in various ways and may include:

- Case studies (e.g., emergency services due to natural disasters and diseases of the immune system).
- Experiments involving factors such as the ease of use of the proposed representation and tool.
- A study of the expressiveness of the proposed modeling language in comparison to other MAS representations proposed in the literature.
- A formalization of the models to support the verification of relevant properties related to adaptation and norms.

References

Atzori, Luigi, Antonio Iera, and Giacomo Morabito. "The internet of things: A survey." Computer networks 54.15, 2010.

Beamon, Benita M., and Stephen A. Kotleba. "Inventory management support systems for emergency humanitarian relief operations in South Sudan." The International Journal of Logistics Management 17.2, 2006.

Bernon, C., Gleizes, M. P., Peyruqueou, S., & Picard, G. ADELFE: a methodology for adaptive multi-agent systems engineering. In Engineering Societies in the Agents World III (pp. 156-169). Springer Berlin Heidelberg, 2003.

Beydoun, G., Low, G., Henderson-Sellers, B., Mouratidis, H., Gomez-Sanz, J. J., Pavon, J., & Gonzalez-Perez, C. FAML: a generic metamodel for MAS development. Software Engineering, IEEE Transactions on, 35(6), 841-863, 2009.

Bogdanovych, Anton, et al. Developing virtual heritage applications as normative multiagent systems. Agent-Oriented Software Engineering X. Springer Berlin Heidelberg, p.140-154, 2011.

Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., & Mylopoulos, J. Tropos: An agent-oriented software development methodology. Autonomous Agents and Multi-Agent Systems, 8(3), 203-236, 2004.

Cerqueira, S. L. R. et al. Plataforma GeoRisc Engenharia da Computação Aplicada à Análise de Riscos Geo-ambientais. PUC-RIO. Rio de Janeiro, 2009.

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

Chefrour, D.: Developing component based adaptive applications in mobile environments. In: Proceedings of the 2005 ACM symposium on Applied computing. p. 1146–1150. SAC '05, ACM, New York, NY, USA, 2005.

Cook, D., J. How Samart Is Your Home?, Science Volume 35, p. 1579-1581, March, 2012.

Da, Keling, Marc Dalmau, and Philippe Roose. "A Survey of adaptation systems." International Journal on Internet and Distributed Computing Systems 2.1, (2011)

Gubbi, Jayavardhana, et al. "Internet of Things (IoT): A vision, architectural elements, and future directions." Future Generation Computer Systems 29.7, p. 1645-1660, 2013.

Hollander, Christopher D.; WU, Annie S. The current state of normative agent-based systems. Journal of Artificial Societies and Social Simulation, v. 14, n. 2, p. 6, 2011.

Janssen, Marijn, et al. "Advances in multi-agency disaster management: Key elements in disaster research." Information Systems Frontiers 12.1, p. 1-7, 2010.

Laroum, Toufik, and Bornia Tighiouart. "A multi-agent system for the modelling of the HIV infection." Agent and Multi-Agent Systems: Technologies and Applications. Springer Berlin Heidelberg, p.94-102, 2011.

López, Fabiola López; Luck, Michael; D'Inverno, Mark. Constraining Autonomy through Norms, 2002.

López, Fabiola López. Social Power and Norms. Diss. University of Southampton, 2003.

Merriam-Webster. Merriam-Webster Online Dictionary. http://www.merriam-webster.com/dictionary/norm, 2015.

Neumann, Martin. Homo socionicus: a case study of simulation models of norms. Journal of Artificial Societies and Social Simulation, v. 11, n. 4, p. 6, 2008.

Oxford Dictionary of Science . Oxford Dictionaries Language Matters. http://www.oxforddictionaries.com/definition/english/adaptation, 20015.

Santos Neto, Baldoino Fonseca, Viviane Torres Da Silva, and Carlos Jose Pereira de Lucena. "Using jason to develop normative agents." Advances in Artificial Intelligence–SBIA 2010. Springer Berlin Heidelberg,p. 143-152, 2010.

Silva, V. T., R. Choren, and C.J.P. De Lucena, "MAS-ML: A Multi-Agent System Modeling Language," Int'l J. Agent-Oriented Software Eng., vol. 2, no. 4, pp. 381-421, 2008.

Silva, V. T. Uma Linguagem de Modelagem para Sistemas Multi-agentes Baseada em um Framework Conceitual para Agentes e Objetos. PUC-Rio, Rio de Janeiro –RJ. Brazil, 2004.

UML: Unified Modeling Language Specification, Version 2.0, OMG, Available in:: <http://www.omg.org/uml/>. Accessed: February 21 2015.

Verhagen, Henricus JE. Norm autonomous agents. Diss. Stockholm Universitet, 2000.

Wooldridge, Michael. An introduction to multiagent systems. John Wiley & Sons, 2011.

Zambonelli, F., Jennings, N. R., & Wooldridge, M. Developing multiagent systems: The Gaia methodology. ACM Transactions on Software Engineering and Methodology (TOSEM), 12(3), 317-370, 2003.

Zhang, Shiwu, and Jiming Liu. "A massively multi-agent system for discovering HIV-immune interaction dynamics." Massively Multi-Agent Systems I. Springer Berlin Heidelberg, 2005.