VII Workshop de Engenharia de Software Baseada em Busca

CBSOFT.ORG

REALIZAÇÃO:

EXECUÇÃO:

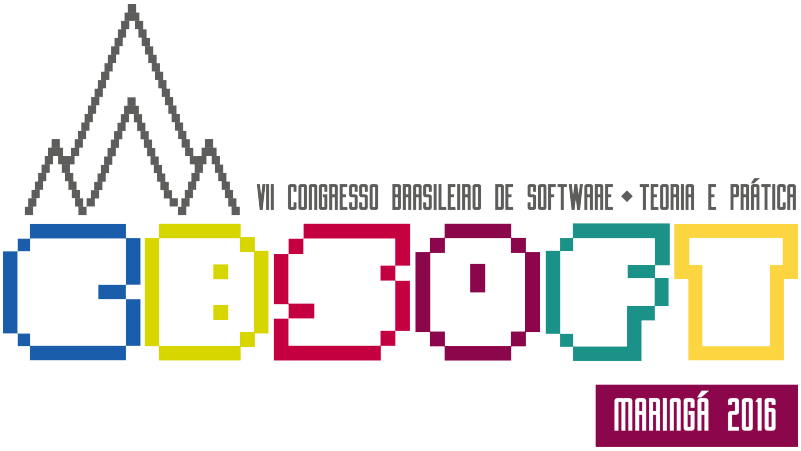
ORGANIZAÇÃO:

APOIO:

FOMENTO:

PATROCINADORES GIGA:

PATRCCINADORES MEGA:



VII WORKSHOP DE ENGENHARIA DE SOFTWARE BASEADA EM BUSCA (WESB 2016)

21 de setembro de 2016 | *September 21, 2016*
Maringá, PR, *Brazil*

ANAIS | *PROCEEDINGS*

Sociedade Brasileira de Computação – SBC

COORDENADORES DO COMITÊ DE PROGRAMA | *PROGRAM COMMITTEE CHAIRS*
Leila Maciel de Almeida e Silva (UFS)
André Britto de Carvalho (UFS)

EDITORES | *PROCEEDINGS CHAIRS*
Marco Aurélio Graciotto Silva (UTFPR)
Willian Nalepa Oizumi (IFPR)

COORDENADORES GERAIS | *GENERAL CHAIRS*
Edson Oliveira Júnior (UEM)
Thelma Elita Colanzi (UEM)
Igor Steinmacher (UTFPR)
Ana Paula Chaves Steinmacher (UTFPR)
Igor Scaliante Wiese (UTFPR)

REALIZAÇÃO | *REALIZATION*
Sociedade Brasileira de Computação (SBC)

EXECUÇÃO | *EXECUTION*
Universidade Estadual de Maringá (UEM) – Departamento de Informática (DIN)
Universidade Tecnológica Federal do Paraná (UTFPR) – Câmpus Campo Mourão (UTFPR-CM)

ISBN: 978-85-7669-338-3

Apresentação

Bem-vindos ao Workshop de Engenharia de Software Baseada em Busca – WESB 2016! O WESB vem se consolidando como o principal fórum nacional para a divulgação e discussão de resultados de pesquisas em Engenharia de Software Baseada em Busca (em inglês, *SBSE – Search Based Software Engineering*), contribuindo assim para o desenvolvimento desta área no país. Através do workshop pretende-se estimular a cooperação dos grupos de SBSE de diversas regiões do país, através da identificação de interesses comuns de pesquisa, que possam resultar em projetos de pesquisa multi-institucionais. Neste sentido, esta edição do workshop inova com a inclusão de uma sessão de pôsteres, cujo objetivo primordial é a discussão dos trabalhos em andamento dos grupos de SBSE que atuam no país.

No contexto do WESB, técnicas de busca englobam tanto técnicas tradicionais, como força bruta ou *branch-and-bound*, quanto meta-heurísticas, como algoritmos genéticos e outros algoritmos bioinspirados. O WESB é um workshop sobre fundamentos teóricos, sobre experiências práticas e de automatização da Engenharia de Software Baseada em Busca em projetos acadêmicos e industriais.

Os nove trabalhos completos submetidos para esta sétima edição do evento foram cuidadosamente revisados por três membros do comitê de programa ou por avaliadores designados pelo comitê, pertencentes a diversas regiões do país. Nestes anais constam os seis trabalhos completos selecionados para a apresentação nas sessões técnicas do evento. Os principais temas abordados foram: testes, requisitos, planejamento e arquitetura de software. Além das sessões técnicas a programação do evento também inclui uma sessão de pôsteres, englobando sete trabalhos de diversos grupos de SBSE do país. Em conjunto com o CBSoft, o evento incorpora como cerne de sua programação a palestra do Prof. Mark Harman, University College London, pesquisador pioneiro na área de SBSE e grande responsável pela disseminação desta área em nível mundial. A palestra versará sobre melhoramento genético no contexto de evolução de software.

O WESB não aconteceria sem a contribuição de inúmeras pessoas. Agradecemos inicialmente aos autores dos artigos e pôsteres submetidos e em particular, parabenizamos aqueles que foram selecionados para apresentação no evento. Agradecemos especialmente aos membros do comitê de programa e aos revisores por eles designados pelo cumprimento dos prazos estabelecidos, facilitando assim a organização do evento. Por fim, agradecemos a todos os que diretamente ou indiretamente nos auxiliaram na organização do WESB 2016, em especial aos organizadores do CBSoft 2016, pelo apoio, infraestrutura disponibilizada e pela oportunidade de disseminar a área de SBSE no Brasil.

Maringá, 21 de setembro de 2016

Leila Maciel de Almeida e Silva (DComp-UFS)
André Britto de Carvalho (DComp-UFS)
Coordenadores do WESB 2016

Foreword

Welcome to the Workshop on Search Based Software Engineering – WESB 2016! The workshop has been considered an important national event for disseminating and discussing research results on Search Based Software Engineering (SBSE). Its main goal is to foster the integration of the Brazilian SBSE groups, providing the opportunity of discussing collaboration on new projects. In this direction, this edition of the workshop includes a poster session, with the aim of discussing ongoing research projects of these groups.

In the context of WESB, search techniques include both traditional techniques, as for example, brute force and branch-and-bound, and meta-heuristics, as for example, genetic and bio-inspired algorithms. The topics of interest comprise theoretical foundation, practical experiments and tools of SBSE, in academic and industrial projects.

The nine full papers submitted for this edition of the event have been carefully revised by three members of the program committee, or by reviewers suggested by them, from several regions of Brazil. These proceedings include six papers selected for presentation in the technical sessions of the event. The main topics addressed are tests, requirements, software planning and software architecture. In addition to the technical sessions, a poster session and an invited talk of Prof. Mark Harman, from University College London, are scheduled. The talk will discuss genetic improvement in the context of software evolution.

The event cannot happen without the collaboration of many people. We would like to express our gratitude to all authors who submitted their full papers and posters to WESB 2016, to the members of program committee and reviewers, for their effort and accomplishment of the deadlines during the paper selection process, and to the CBSoft 2016 organizers, for their support and for the opportunity for widespread the SBSE area in Brazil.

Maringá, 21st September 2016

Leila Maciel de Almeida e Silva (DComp-UFS)
André Britto de Carvalho (DComp-UFS)
Program Committee Chairs

Comitê técnico | *Technical committee*

Coordenadores de comitê de programa | *PC chairs*

Leila Maciel de Almeida e Silva (UFS)
André Britto de Carvalho (UFS)

Comitê de programa | *Program committee*

Adriana C. F. Alvim (UNIRIO)
André Britto de Carvalho (UFS)
Arilo Claudio Dias Neto (UFAM)
Auri Marcelo Rizzo Vincenzi (UFG)
Aurora Pozo (UFPR)
Breno Piva Ribeiro (UFS)
Geraldo Robson Mateus (UFMG)
Gledson Elias (UFPB)
Gustavo Augusto Lima de Campos (UECE)
Jerffeson Teixeira de Souza (UECE)
Leila Silva (UFS)
Márcio de Oliveira Barros (UNIRIO)
Maria Cláudia Figueiredo Pereira Emer (UTFPR)
Silvia Regina Vergilio (UFPR)
Thelma Elita Colanzi (UEM)

Revisores externos | *External reviewers*

Awdren Fontão (UFAM)
Kariny Oliveira (UFAM)
Sílvia Meireles (UFAM)
Wesley Assunção (FASUL)

Artigos técnicos | *Technical papers*

Engenharia de Software Baseada em Busca e em Preferência: Uma Visão Geral

Thiago Nascimento Ferreira[1], **Silvia Regina Vergilio**[1], **Jerffeson Teixeira de Souza**[2]

[1] DInf - Universidade Federal do Paraná,
CP: 19081, CEP: 81.531-980, Curitiba, Brasil

[2]Universidade Estadual do Ceará,
Avenida Dr. Silas Munguba, 1700. Fortaleza, Brasil

{tnferreira, silvia}@inf.ufpr.br and jerffeson.souza@uece.br

Abstract. *In the past years, optimization algorithms have been successfully applied to offer solutions for different problems in the Search-based Software Engineering (SBSE) area. However, in practice, the user can reject and not recognize the obtained solutions, as his/her preferences were not taken into account. Therefore, the use of preference-based algorithms has raised interest in SBSE. In this sense, this paper presents an overview of works in this new field, called here Preference and Search Based Software Engineering, by providing results of a mapping that show the most used algorithms and addressed software engineering areas, and by presenting some research opportunities.*

Resumo. *Nos últimos anos, algoritmos de otimização têm sido aplicados com sucesso para oferecer soluções para diferentes problemas na área de Engenharia de Software Baseada em Busca (Search-based Software Engineering - SBSE). No entanto, na prática, o usuário pode rejeitar e não reconhecer as soluções obtidas, já que suas preferências não foram levadas em consideração. Por isso, o uso de algoritmos baseados em preferências do usuário tem despertado interesse em SBSE. Neste sentido, este artigo apresenta uma visão geral de trabalhos neste novo campo, chamado aqui de Engenharia de Software Baseada em Busca e em Preferência, fornecendo resultados de uma mapeamento que mostra os algoritmos mais usados e áreas de engenharia de software mais investigadas, e também discutindo algumas oportunidades de pesquisa.*

1. Introdução

Algoritmos de busca têm sido utilizados para resolver problemas de otimização na área de Engenharia de Software (ES). Os trabalhos com este objetivo se agrupam na área denominada Engenharia de Software Baseada em Busca (*Search-based Software Engineering* - SBSE) e apresentam resultados relevantes e promissores em várias atividades da ES, tais como Ferramentas e Técnicas de Codificação [Kukunas et al. 2010], Ferramentas e Técnicas de Design. [Simons et al. 2014], e Requisitos/Especificações [de Souza et al. 2011].

Em SBSE, geralmente existe uma solução exata para o problema, mas ela não é conhecida pois o esforço computacional necessário para identifica-la é inviável e, devido a isto, é necessário aplicar alguma técnica de busca para resolvê-lo. Para aplicar

tais técnicas, Harman and Jones [2001] definem que os problemas de ES precisam ser reformulados como um problema de busca e, para isso, é necessário definir: a) uma representação do problema; b) um conjunto de operadores de manipulação da solução; e c) uma função de aptidão.

Entretanto, existem algumas situações nas quais não é possível definir facilmente a função de aptidão (ou função objetivo). Por exemplo, algumas características específicas do problema não podem ser modeladas matematicamente. Assim, o uso de algoritmos baseados em preferências é necessário pois o conhecimento e capacidade de julgamento humano podem auxiliar as técnicas de busca a alcançar as melhores soluções.

Algoritmos baseados em preferências são algoritmos que incorporam as preferências humanas (fornecidas pelo tomador de decisão) dentro do processo de busca [Takagi 2001]. O uso de tais algoritmos tem despertado o interesse dos pesquisadores nos últimos anos pois resolve uma limitação prática de SBSE: o usuário pode rejeitar as soluções geradas pelos algoritmos pois suas preferências não foram levadas em consideração no processo de otimização.

Com o objetivo de contribuir para o crescimento deste novo campo de pesquisa, chamado aqui de Engenharia de Software Baseada em Busca e em Preferência (*Preference and Search Based Software Engineering* - PSBSE) e motivar novos trabalhos que aplicam algoritmos baseados em preferências em SBSE, este artigo apresenta resultados de um mapeamento sistemático, no qual busca-se identificar algumas especificidades de PSBSE, como por exemplo, áreas de ES investigadas, algoritmos utilizados e os momentos nos quais as preferências são incorporadas no processo de otimização.

Muitos *surveys* da área de SBSE apontam o uso de algoritmos baseados em preferência como uma tendência [Harman et al. 2012]. Entretanto não há *surveys* em SBSE abordando especificamente PSBSE. Por outro lado, *surveys* focados em algoritmos baseados em preferências [Bechikh et al. 2015] não mencionam abordagens da área de SBSE.

Este artigo está organizado da seguinte forma: Seção 2 revisa a área de algoritmos baseados em preferências. Seção 3 descreve o método do mapeamento. Seção 4 apresenta os principais resultados e discute tendências e oportunidades de pesquisa identificadas. Seção 6 contém os trabalhos relacionados. Por fim, Seção 7 discute as considerações finais e trabalhos futuros.

2. Algoritmos Baseados em Preferências

Um algoritmo baseado em preferência é um algoritmo de otimização que incorpora preferências humanas, intuições, emoções ou aspectos psicológicos dentro do processo de busca [Takagi 2001]. Neste tipo de algoritmo, o Tomador de Decisão (TD) é a pessoa (ou um grupo) que fornece as preferências.

O TD pode fornecer suas preferências em diferentes momentos dentro do processo de busca, dependendo do problema ou algoritmo utilizado. Tais momentos foram definidos por Hwang and Masud [1979] para classificar algoritmos baseados em preferências nas seguintes categorias:

- A priori – o TD fornece suas preferências antes do processo de busca (*a priori*);
- Interativamente – o TD fornece suas preferências durante o processo de busca (também conhecido como *interactive* ou *human-in-the-loop*);

- A posteriori – o TD fornece suas preferências depois do processo de busca.

Cada categoria tem vários métodos ou, ainda, um método pode pertencer a várias categorias. Entretanto, alguns são básicos ou bem conhecidos como, por exemplo, o *Weighting Method* [Branke et al. 2008] no qual todos os objetivos do problema são convertidos para um simples objetivo, no qual o usuário define o peso w_i de cada um.

3. Processo de Mapeamento

Este trabalho seguiu o processo de mapeamento proposto por Petersen et al. [2008] , o qual inclui o seguintes passos: a) definição das questões de pesquisa; b) condução da busca e triagem dos trabalhos; c) definição do esquema de classificação; e d) extração dos dados e mapeamento. Cada passo é descrito a seguir.

3.1. Questões de Pesquisa

O objetivo deste trabalho é apresentar uma visão geral das pesquisas existentes em PSBSE. Para isso, foram definidas as seguintes questões de pesquisa:

RQ1: *Em que momento as preferências são fornecidas?*

RQ2: *Que áreas da ES são investigadas pelos trabalhos existentes?*

RQ3: *Quais são os algoritmos utilizados?*

3.2. Condução da Busca e Triagem dos Artigos

Um conjunto de palavras-chave foi definido baseado no objetivo do trabalho e nas questões de pesquisa. As palavras-chave extraídas foram categorizadas em três grupos apresentados na Tabela 1. Os termos do primeiro e segundo grupos foram extraídos do trabalho de Harman et al. [2012] e compreendem as áreas da ES e algoritmos de busca respectivamente. Os termos do terceiro grupo foram extraídos do trabalho de Bechikh et al. [2015] e estão relacionados a algoritmos baseados em preferências. Em ambos os grupos, os termos foram atualizados ou novos foram adicionados. As palavras-chave de cada grupo foram agrupadas usando o operador "OR" e cada grupo foi agrupado usando o operador "AND".

A busca e a seleção foram conduzidas em seis passos. No primeiro passo, a string de busca foi executada nas bases de dados eletrônicas mais relevantes e que estão apresentadas na Tabela 2. A busca considerou artigos publicados de 2000 até 14 de Março de 2016 e levou em consideração o título, resumo e palavras-chaves. Algumas modificações foram realizadas afim de adequar a string de busca a base de dados como, por exemplo, dividir a string em pequenas partes. No final desse passo, um conjunto de 3593 foram retornados.

No segundo passo, 768 artigos repetidos foram removidos restando 2825 artigos. No terceiro e quarto passos, os títulos e resumos foram analisados, descartando-se artigos irrelevantes e resultando no final 78 artigos. No quinto passo, os critérios de inclusão/exclusão descritos na Tabela 3 foram aplicados e, ao final, 19 artigos foram selecionados por estarem de fato relacionados com este trabalho. No último passo, as citações e referências listadas em cada artigo foram usadas para identificar novos artigos admissíveis. Neste passo, 15 novos foram adicionados e o conjunto final foi composto por 34 artigos. Assim, os artigos selecionados são apresentados no apêndice. Tabela 7.

Tabela 1. Termos de Pesquisa.

Grupo	Área	Palavras-chave
Engenharia de Software	General	software engineering
	Requirements/Specifications	requirement OR specification OR next release OR release planning OR requirements selection OR requirements analysis OR COTS OR requirements prioritisation OR requirements triage
	Design Tools and Techniques	software design OR design pattern OR software architecture OR QoS OR component OR synthesis OR OO design OR software product line
	Software/Program Verification	model checking OR verification OR testing OR validation OR test OR defect analysis OR debugging
	Distribution, Maintenance and Enhancement	maintenance OR refactoring OR modularization OR evolution OR real time OR quality prediction OR legacy systems OR migration OR software reverse engineering
	Management	project planning OR project management OR cost estimation OR effort estimation OR risk management
	Distributed Artificial Intelligence	agent OR multiagent OR multi-agent
	Software System Structure	embedded software OR model-driven software engineering OR distributed systems
	Software Properties	cohesion OR coupling OR fault tolerance OR software reliability OR software safety OR software usability OR software quality OR software security
Técnicas de Busca	General	search based OR multi-objective optimization OR multi-objective algorithm OR genetic algorithms OR GAs OR genetic programming OR GP OR hill climbing OR simulated annealing OR local search OR integer programming OR ant colony optimization OR ACO OR PSO OR EDA OR metaheuristic OR meta-heuristic OR evolutionary algorithm OR bio-inspired OR moea OR moa OR interactive genetic algorithm
Algoritmos baseados em preferências	General	interactive algorithm OR interactive search OR preference OR decision-maker OR user-provided OR user-specified OR weight-based OR ranking-based

Tabela 2. Número de artigos selecionados em cada base de dados eletrônica.

Base de Dados	Site	#
Scopus	`http://www.scopus.com`	1,389
Science Direct	`http://www.sciencedirect.com`	876
Web of Science	`http://www.isiknowledge.com`	315
IEEE Xplore	`http://ieeexplore.ieee.org`	236
ACM Digital Library	`http://dl.acm.org`	194
Springer	`http://www.springerlink.com`	560
SBSE Repository	`http://crestweb.cs.ucl.ac.uk/resources/sbse_repository`	23
Total		**3593**

Tabela 3. Critérios de Inclusão/Exclusão aplicados nos estudos.

Inclusão	• Artigos em inglês • Publicações em revistas, conferências e workshops; tutoriais, artigos curtos, demonstrações de ferramentas, teses completas, capítulos de livros e relatórios técnicos • Disponíveis em um formato eletrônico: HTML, etc. • Mapeamentos, *surveys*, estado da arte e revisão da literatura • Com foco em PSBSE
Exclusão	• Position papers and doctoral symposium • Resumos • Artigos não disponíveis online • Sem foco em PSBSE

3.3. Esquema de Classificação e Extração dos Dados

Algumas análises foram conduzidas de acordo com as questões de pesquisa e, com isso, foi definido um esquema de classificação com as seguintes dimensões.

- **Momento:** os artigos foram classificados de acordo com as categorias descritas

na Seção 2. Adicionalmente a categoria "A priori Incremental" foi incluída para englobar abordagens que utilizam algoritmos incrementais;
- **Áreas de ES:** foram identificadas as principais áreas da ES usando o sistema de classificação de computação da ACM[1];
- **Algoritmos:** esta dimensão apresenta quais são os algoritmos de busca mais usados. Além disso, procurou-se identificar o tipo de formulação para o problema considerando o tratamento para os objetivos associados.

Por fim, foi utilizado um formulário de extração de dados para extrair todas as informações necessárias para responder às questões de pesquisa. Outras informações como título, ano, local de publicação e autores também foram capturadas.

4. Resultados

Nesta seção são apresentadas as respostas para cada questão de pesquisa.

4.1. RQ1 – Em que momento as preferências são fornecidas?

A Tabela 4 apresenta os momentos utilizados nos artigos selecionados. A maioria dos artigos (21 (61.7%)) pertence a categoria "Interativamente". Quatro artigos (11.7%) pertencem a ambas as categorias "A priori" e "Interativamente". Sete artigos (20.5%) pertencem à categoria "A priori Incremental" e somente dois artigos (5.8%) pertencem à categoria "A priori". Não foi encontrado nenhum artigo que utiliza o momento "A posteriori".

Tabela 4. Momentos utilizados.

Momento	# de Artigos	Referências
Interativamente	21	[S2–S7,S10–S13,S15–S19,S28,S30–S34]
A priori Incremental	7	[S8,S14,S20–S24]
A priori e Interativamente	4	[S25–S27,S29]
A priori	2	[S1,S9]

4.2. RQ2 – Que áreas da ES são investigadas pelos trabalhos existentes?

A Tabela 5 mostra que os trabalhos de PSBSE abordam somente quatro áreas e nove atividades de ES. Dentre as atividades, a mais investigada é Planejamento de *Releases* (8 artigos (23.52%)), seguida por Projeto e Teste de Software (6 artigos (17.6%) cada uma). Este resultado difere do trabalho de Harman et al. [2012] no qual, tradicionalmente, teste de software é a atividade de software mais abordada.

4.3. RQ3 – Quais são os algoritmos utilizados?

A Tabela 6 apresenta os algoritmos mais utilizados na qual, na sua maioria, são evolutivos. O Algoritmo Genético Interativo (IGA) é o mais utilizado (12 artigos ou 35.2%) seguido pelo Algoritmo Genético Incremental (AG Incremental) e NSGA-II com 7 artigos (20.5%) cada um. Com relação a formulação, a maioria das abordagens usa algoritmos mono-objetivos (24 artigos ou 70.5%).

A Figura 1 apresenta o número de trabalhos encontrados considerando os momentos e algoritmos utilizados por atividades de software encontrados nos trabalhos selecionados. A figura mostra que os algoritmos ACO, AG Incremental, IBEA e Evolução

[1] www.computer.org/portal/web/publications/acmsoftware

Tabela 5. Atividades de Engenharia de Software abordadas.

Área (Tabela 1)	Atividades	# de Artigos	Referências
Requirements/Specification	Seleção de Requisitos	1	[S2]
	Planejamento de Release	8	[S4,S8,S14,S20–S24]
	Priorização de Requisitos	2	[S32,S33]
Distribution, Maintenance and Enhacement	Re-modularização de Software	1	[S3]
	Refatoração de Software	3	[S1,S7,S15]
Design Tools and Techniques	Engenharia de LPS	2	[S5,S34]
	Projeto de Interface de Usuário	5	[S16–S19,S31]
	Projeto de Software	6	[S25–S30]
Software/Program Verification	Teste de Software	6	[S6,S9–S13]

Tabela 6. Algoritmos utilizados.

Algoritmos	Formulação	# de Artigos	Referências
Evolução Diferencial	Mono-objetivo	4	[S10–S13]
IBEA	Multiobjetivo	2	[S5,S34]
NSGAII	Multiobjetivo	7	[S5,S9,S15,S25–S27,S29]
AG Incremental	Mono-objetivo	7	[S8,S14,S20–S24]
IGA	Mono-objetivo	12	[S1–S4,S7,S16–S19,S31–S33]
VEGA	Multiobjetivo	1	[S28]
ACO	Mono-objetivo	1	[S30]
Algoritmo Biomimético	Multiobjetivo	1	[S6]

Diferencial foram utilizados em somente uma atividade da ES. Com relação aos momentos, a categoria A priori Incremental foi utilizada somente em Requisitos e a categoria A priori e Interativamente foi utilizada somente em Projeto de Software. Estes resultados mostram que existem muitas oportunidades de pesquisas considerando algoritmos, momentos e atividades da ES.

4.4. Discussão

Durante a análise dos artigos selecionados, algumas tendências e oportunidades de pesquisas foram identificadas. Por exemplo, alguns grupos de pesquisas trabalham especificamente com alguma atividade da ES e usam alguns algoritmos específicos para incorporar as preferências. Explorar outros algoritmos nessas áreas, ou ainda deixar o usuário escolher o algoritmo pode ser considerado como uma oportunidade de pesquisa. Ou ainda, outras áreas da ES, ainda não investigadas, também podem se beneficiar com os algoritmos baseados em preferência, como as áreas de Manutenção e Reúso de Software, dentre outras.

Além disso, não foi encontrada nenhuma abordagem que utilizasse o momento "A posteriori". Nesta categoria, um conjunto de soluções é apresentado ao TD e ele escolhe a mais satisfatória. O problema torna-se mais complicado quando vários objetivos são utilizados e o TD tem que escolher a melhor solução dentro da fronteira de Pareto. Dessa forma, desenvolver abordagens "a posteriori" pode ser considerado como oportunidade de pesquisa uma vez que tais métodos podem diminuir o esforço psicológico de selecionar uma solução dentre as várias geradas. Outra oportunidade de pesquisa é o tratamento de

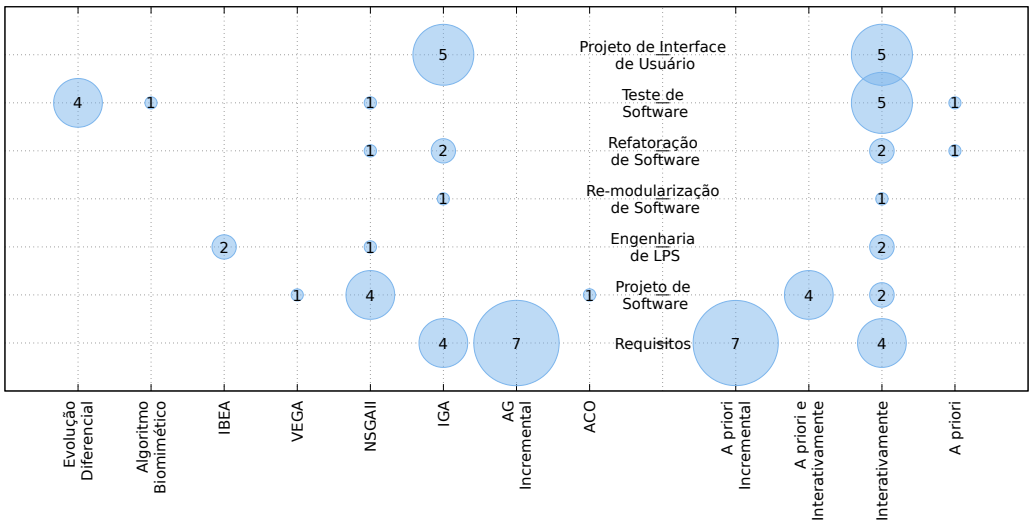


Figura 1. Algoritmos e momentos utilizados por atividade de software.

múltiplas e, possivelmente conflitantes, preferências fornecidas de vários TDs.

Adicionalmente, uma verificação foi realizada para descobrir quem fornece as preferências do usuário (simuladores ou TDs) na fase de avaliação do algoritmo. Foi identificado que, na maioria dos trabalhos, o TD é a pessoa que fornece as preferências para o algoritmo. Este resultado difere do encontrado por Branke et al. [2008] no qual os autores alegam que a maioria dos trabalhos utilizam simuladores para avaliar suas abordagens.

5. Limitações e Ameças à Validade

Algumas ameaças à validade foram encontradas neste trabalho. Alguns trabalhos podem não ter sido incluídos neste estudo devido à falta de uma termilogia clara para a área de PSBSE. Para diminuir esse risco, a *string* de busca foi extraída dos trabalhos relacionados e refinada várias vezes até que os trabalhos mais conhecidos fossem selecionados. Além disso, em alguns estudos o conceito de algoritmos baseados em preferência pode não estar tão claro para o leitor e algumas decisões subjetivas foram tomadas. Assim, para diminuir esta ameaça, os critérios de inclusão e exclusão foram aplicados cuidadosamente analisando cada artigo.

6. Trabalhos Relacionados

Não foi encontrado na literatura *surveys* ou mapeamentos que abordam especificamente a área de PSBSE. Os trabalhos mais relacionados com este são os que abordam algoritmos baseados em preferências e SBSE separadamente. Por exemplo, no contexto de algoritmos baseados em preferencias, Bechikh et al. [2015] descrevem varias questões de pesquisa que ainda estão em aberto dentro da área. Em SBSE, Harman et al. [2012] apresentam uma visão geral de trabalhos que usam algoritmos baseados em busca em várias atividades da ES citando, inclusive, que algoritmos baseados em preferências são a tendência dentro da área. Assim, este trabalho ajuda a identificação de lacunas existentes na área de PSBSE, apontando algumas oportunidades de pesquisa para trabalhos futuros.

7. Considerações Finais

Algoritmos baseados em busca têm sido aplicados com sucesso em várias áreas da ES e o número de artigos continua crescendo. Considerar as preferências do usuário em tais aplicações é uma questão fundamental para o estado da prática em SBSE.

Os resultados do mapeamento descrito neste trabalho contribuem para PSBSE. Em resumo, as áreas mais investigadas são Ferramentas e Técnicas de Projeto, e Requisitos/Especificações. O algoritmo IGA é o mais empregado e, na maioria dos trabalhos, as preferências do usuário são fornecidas interativamente.

Como trabalho futuro, pretende-se explorar especificamente como as preferências do usuário são gerenciadas nos algoritmos e como as abordagens são avaliadas.

Referências

Bechikh, S., Kessentini, M., Said, L. B., and Ghédira, K. (2015). Preference incorporation in evolutionary multiobjective optimization: A survey of the state-of-the-art. In Hurson, A. R., editor, *Advances in Computers*, volume 98, pages 141 – 207. Elsevier.

Branke, J., Deb, K., Miettinen, K., and Słowinski, R. (2008). Multiobjective optimization - interactive and evolutionary approaches. *Lecture Notes in Computer Science, Springer-Verlag, New York*, 5252.

de Souza, J. T., Maia, C. L. B., Ferreira, T. N., Carmo, R. A. F., and Brasil, M. (2011). An ant colony optimization approach to the software release planning with dependent requirements. In *Proceedings of the 3rd SSBSE*, pages 142–157, Szeged. Springer.

Harman, M. and Jones, B. F. (2001). Search-based software engineering. *Information and software Technology*, 43(14):833–839.

Harman, M., Mansouri, S. A., and Zhang, Y. (2012). Search-based software engineering: Trends, techniques and applications. *ACM Computing Surveys*, 45(1):11:1–11:61.

Hwang, C.-L. and Masud, A. S. M. (1979). Multiple objective decision making - methods and applications: A state-of-the-art survey. In *Lecture Notes in Economics and Mathematical Systems*. Springer-Verlag, Berlin, Heidelberg.

Kukunas, J., Cupper, R. D., and Kapfhammer, G. M. (2010). A genetic algorithm to improve linux kernel performance on resource-constrained devices. In *Proceedings of the 12th GECCO*, pages 2095–2096, New York, NY, USA. ACM.

Petersen, K., Feldt, R., Mujtaba, S., and Mattsson, M. (2008). Systematic mapping studies in software engineering. In *Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering*, EASE'08, pages 68–77, Swinton, UK, UK. British Computer Society.

Simons, C. L., Smith, J., and White, P. (2014). Interactive ant colony optimization (iACO) for early lifecycle software design. *Swarm Intelligence*, 8(2):139–157.

Takagi, H. (2001). Interactive evolutionary computation: fusion of the capabilities of EC optimization and human evaluation. *Proceedings of the IEEE*, 89(9):1275–1296.

Apêndice

Tabela 7: Artigos Selecionados.

Id	Ano	Autor	Título	Veículo/Local da Publicação
[S12]	2015	B. Marculescu et al.	An initial industrial evaluation of interactive search-based testing for embedded software	Applied Soft Computing
[S10]	2013	B. Marculescu et al.	Objective Re-weighting to Guide an Interactive Search Based Software Testing System	Proceedings of the 12th ICMLA
[S11]	2013	B. Marculescu et al.	Practitioner-oriented visualization in an interactive search-based software test creation tool	Proceedings of the 20th APSEC
[S26]	2008	C. L. Simons and I. C. Parmee	User-centered, evolutionary search in conceptual software design	Proceedings of the CEC
[S30]	2014	C. L. Simons et al.	Interactive ant colony optimization (iACO) for early lifecycle software design	Swarm Intelligence
[S29]	2010	C. L. Simons et al.	Interactive, evolutionary search in upstream object-oriented class design	IEEE Transactions on Software Engineering
[S32]	2010	P. Tonella et al.	Using interactive GA for requirements prioritization	Proceedings of the 2nd SSBSE
[S8]	2004	D. Greer and G. Ruhe	Software release planning: an evolutionary and iterative approach	Information and Software Technology
[S27]	2009	C. L. Simons and I. C. Parmee	An empirical investigation of search-based computational support for conceptual software engineering design	Proceedings of the SMC
[S21]	2003	G. Ruhe and D. Greer	Quantitative studies in software release planning under risk and resource constraints	Proceedings of the ISESE
[S33]	2013	P. Tonella et al.	Interactive requirements prioritization using a genetic algorithm	Information and software technology
[S9]	2013	S. Kalboussi et al.	Preference-based many-objective evolutionary testing generates harder test cases for autonomous agents	Proceedings of the 5th SSBSE
[S2]	2014	A. A. Araújo and M. Paixão	Machine learning for user modeling in an interactive genetic algorithm for the next release problem	Proceedings of the 6th SSBSE
[S19]	2007	J. C. Quiroz et al.	Interactive genetic algorithms for user interface design	Proceedings of the CEC
[S34]	2014	E. Yamany et al.	OPTI-SELECT: an interactive tool for user-in-the-loop feature selection in software product lines	Proceedings of the 18th SPLC
[S1]	2014	B. Amal et al.	On the use of machine learning and search-based software engineering for Ill-defined fitness function: a case study on software refactoring	Proceedings of the 6th SSBSE

[S15]	2014	M. W. Mkaouer et al.	Recommendation system for software refactoring using innovization and interactive dynamic optimization	Proceedings of the 29th ACM/IEEE international conference on Automated software engineering
[S4]	2015	A. Dantas et al.	Interactive Software Release Planning with Preferences Base	Proceedings of the 7th SSBSE
[S7]	2013	A. Ghannem et al.	Model refactoring using interactive genetic algorithm	Proceedings of the 5th SSBSE
[S6]	2002	R. Feldt	An interactive software development workbench based on biomimetic algorithms	Gothenburg, Sweden, Tech. Rep
[S28]	2012	C. L. Simons and I. C. Parmee	Elegant object-oriented software design via interactive, evolutionary computation	IEEE Transactions on Systems, Man, and Cybernetics, Part C
[S3]	2012	G. Bavota et al.	Putting the developer in-the-loop: an interactive GA for software re-modularization	Proceedings of the 4th SSBSE
[S5]	2015	A. E. El Yamany and M. S. Elgamel	Smart OptiSelect Preference Based Innovative Framework for User-in-the-Loop Feature Selection in Software Product Lines	International Conference on Information Technology (ICIT),
[S20]	2008	G. Ruhe et al.	A systematic approach for solving the wicked problem of software release planning	Soft Computing
[S31]	2013	D. Sorn and S. Rimcharoen	Web page template design using interactive genetic algorithm	Proceedings of the ICSEC
[S16]	2002	A. Oliver et al.	Interactive Design of Web Sites with a Genetic Algorithm	Proceedings of the IADIS International Conference WWW/Internet (ICWI'02)
[S18]	2007	J. C. Quiroz et al.	Interactive evolution of XUL user interfaces	Proceedings of the 9th GECCO
[S17]	2007	J. C. Quiroz et al.	Human guided evolution of xul user interfaces	CHI'07 Extended Abstracts on Human Factors in Computing Systems
[S13]	2015	B. Marculescu et al.	Tester Interactivity makes a Difference in Search-Based Software Testing: A Controlled Experiment	CoRR, abs/1512.04812, 2015
[S23]	2004	G. Ruhe et al.	Intelligent support for software release planning	Product Focused Software Process Improvement
[S24]	2005	O. Saliu and G. Ruhe	Supporting software release planning decisions for evolving systems	Proceedings of the 29th Annual IEEE/NASA Software Engineering Workshop
[S14]	2006	S. Maurice et al.	Decision Support for Value-Based Software Release Planning	Value-Based Software Engineering
[S22]	2005	G. Ruhe and J. Momoh	Strategic release planning and evaluation of operational feasibility	Proceedings of the 38th HICSS
[S25]	2011	C. Simons	Interactive evolutionary computing in early lifecycle software engineering design	PhD thesis, University of the West of England

Reviewing Six Years of Brazilian Workshop on Search-Based Software Engineering

Thiago Nascimento Ferreira[1], **Thainá Mariani**[1], **Silvia Regina Vergilio**[1]

[1]DInf - Universidade Federal do Paraná (UFPR) – Curitiba, PR – Brasil

{tnferreira, tmariani, silvia}@inf.ufpr.br

Abstract. *The Brazilian Workshop on Search-Based Software Engineering (WESB) congregates researchers from the Brazilian Search-Based Software Engineering (SBSE) community. WESB has became the main event and fundamental to consolidate the area in Brazil. However, after six years, it is important to understand the outcomes and impacts of the six WESB editions. To this end, this paper presents a review of all papers published in the event, in order to collect different informations, such as the authors, universities, research groups, software engineering areas, and search-based algorithms. A set of research questions was defined, and thus, the studies were categorized based on a classification schema. We observe some similarities with the international SBSE scenario, such as a preference for software testing and evolutionary algorithms. We can conclude that the event played an important role to increase collaboration and incentive the researchers to create research groups on SBSE in different universities. Besides we identified some perspectives for future work and unexplored areas such as software maintenance.*

Resumo. *O Workshop Brasileiro de Engenharia de Software Baseada em Busca (WESB) congrega pesquisadores da comunidade brasileira de Engenharia de Software Baseada em Busca. WESB tornou-se o principal e fundamental evento para consolidação da área no Brasil. Entretanto, após seis anos, é importante compreender os resultados e impactos das seis edições do WESB. Com este objetivo, este artigo apresenta uma revisão de todos os artigos publicados no evento, a fim de coletar diferentes informações, tais como autores, universidades, grupos de pesquisa, áreas da engenharia de software e algoritmos de busca. Um conjunto de questões de pesquisa foi definido, e então os estudos foram categorizados de acordo com um esquema de classificação. Foram observadas similaridades com o cenário internacional, tais como uma preferência por teste de software e algoritmos evolutivos. Pode-se concluir que o evento foi muito importante para aumentar colaboração e incentivar pesquisadores a criar grupos de pesquisa em Engenharia de Software Baseada em Busca em diferentes universidades. Além disso, foram identificadas algumas perspectivas para trabalho futuro e áreas não exploradas como modelos e métodos para engenharia de software e manutenção.*

1. Introduction

Search-Based Software Engineering (SBSE) [Harman and Jones 2001] is the field devoted to the application of search-based techniques to solve Software Engineering (SE)

problems. It has been successfully applied in many SE areas, such as software testing, design, maintenance, and so on. SBSE can be now considered a consolidated field, and as consequence, raises interest of many researchers around the world [Harman et al. 2012].

In the last decade a Brazilian SBSE community emerged and has provided different contributions for a variety of SE areas [Colanzi et al. 2013]. In 2010, the Brazilian Workshop on Search-Based Software Engineering (WESB) was created to promote and discuss studies on SBSE. The workshop seeks for theoretical fundamentals and practical experiences applied in the academic and industrial contexts. WESB is co-located with the Brazilian Conference on Software: Theory and Practice (CBSoft) promoted by the Brazilian Computer Society (SBC) and attracts Brazilian academic researchers, including professors and students of many universities. Until now, six editions of the workshop have been organized. WESB has became fundamental to congregate the Brazilian community and contribute to its consolidation. However after six years it is important to understand the outcomes and impacts of WESB for the Brazilian SBSE community. In this sense, this paper presents a review of all WESB editions, by identifying, evaluating and interpreting studies published in such a workshop.

Reviews covering the SBSE field in Brazil are found in the literature [Assunção et al. 2013, Colanzi et al. 2013, Vergilio et al. 2011]. Some of them select studies published in the 1st and 2nd editions of WESB. However, they do not specifically focus the workshop, since their goal is to present a review of the field in Brazil. Surveys covering the SBSE field [Harman et al. 2009, Harman et al. 2012] are also found in the literature. They cover the main aspects of the field, such as the involved SE areas and search techniques. Our work presents a different goal: to characterize the research developed and present initial evidence about the impacts and evolution of the WESB editions.

To this end, we followed the method of Petersen et al. (2015) to establish some research questions and a classification schema. The questions were defined in order to identify different informations, such as the main authors, the main universities, collaborations between universities, research groups, the addressed SE areas, the used algorithms, type of studies and evaluation aspects. Thus, the search for papers was conducted in all WESB proceedings published from 2010 to 2015. As a main finding, we observe that the event was fundamental to congregate the Brazilian community, to increase collaboration, and incentive the researchers to create research groups on SBSE in different universities. In addition to this, we observe such unexplored areas and perspectives for the area.

This paper is organized as follows. Section 2 describes the research method used, including the research questions, how the search for papers was conducted, and the classification schema. Section 3 shows the obtained results and answers to the research questions. Section 4 discusses the results. Section 5 presents related work. Finally, Section 6 concludes this paper and shows some future works.

2. Research Method

This review was performed following the guidelines presented by Petersen et al. (2013). The essential steps involve: i) definition of the research questions; ii) search for relevant papers; and iii) data extraction. In this sense, the next paragraphs show details about the conduction of this review based on such steps.

Considering our goal, the research questions were elaborated to characterize the

research developed and evolution of the WESB editions. They are:

[RQ1:] **How are the workshop and the area in Brazil evolving?** Rationale: the idea is to analyze the workshop and the SBSE area in Brazil regarding the number of submissions, number of authors, number of papers per authors, the top universities, the existing research groups, and the level of collaboration among researchers. All these points can be evaluated along the years and WESB editions to identify tendencies.

[RQ2:] **What are the most explored software engineering topics and algorithms?** Rationale: it aims at analyzing the main SE areas and algorithms investigated, knowledge areas specificities of each group, international influences, as well as, similarities and differences.

[RQ3:] **What are the type of studies that have been published?** Rationale: in workshops, it is very common to find papers that only report insightful proposals without application. So this question analyses the type of studies published and can indicate the maturity level of the area and workshop.

[RQ4:] **How have the proposals been evaluated?** Rationale: for the works that report evaluation results, this question aims at investigating some related aspects such as used instances and evaluation methods.

In order to select the primary studies, a manual search was conducted in the proceedings of all WESB editions, covering all papers published from 2010 to 2015, totalizing 51 papers. Table 1 shows the number of papers of each edition.

Table 1. Number of papers selected by edition.

Edition	1st	2nd	3rd	4th	5th	6th
Year	2010	2011	2012	2013	2014	2015
Number of Papers	9	9	6	12	8	7

A classification schema was created including the following dimensions: i) SE area: defined according to the Software Engineering Body of Knowledge (SWEBOK) [Bourque and Fairley 2014]. Moreover, we added the topic *Introductory/Surveys* to classify papers that are not in any other category; ii) type of study: defined according to Montesi and Lago [2008] ; and iii) type of instances used in experimental papers. Table 2 shows such dimensions and corresponding categories.

After the selection of primary studies, the relevant data was extracted to be analyzed in order to answer the research questions: title, authors and affiliations, SE area, used algorithm, type of study, used instances, and evaluation methods. The details about each paper were omitted due to space restrictions, but they can be found at[1].

2.1. Threats to Validity

We identified some threats to validity of the results. The research questions are considered a threat, since they may not address all the aspects related to the workshop. To minimize such a threat, the research questions were elaborated covering the main aspects to review workshops and some details related to the SBSE field.

[1] `http://www.inf.ufpr.br/gres/wesb-stats/`

Table 2. Classification schema.

Dimension	SE Area	Type of Study	Type of Instance
Categories	Software Testing	Extended Versions of Conference Papers	Artificial
	Software Design	Empirical Research Reports	Real
	Software Requirements	Experience Papers	Both
	Software Engineering Management	Theoretical Papers	
	Software Configuration Management	Tutorial Papers	
	Software Construction	Surveys	
	Software Maintenance	Short Papers	
	Software Engineering Process	Papers Oriented Towards Practice	
	Software Engineering Models and Methods		
	Software Quality		
	Introductory/Survey		

The classification schema and the way that the papers were classified are other threat to be considered. Some studies were classified based on subjective aspects. So, a double checking was performed to mitigate this threat. Besides, some papers could be classified in more than one category. In such a case, we consider the most related one.

3. Main Findings

In this section, we answer each research question, based on the extracted information and the classification schema.

3.1. RQ1 – WESB Evolution

By analyzing Table 1, we observe that the number of papers published considering all the editions is 51. Along this period, the number of submissions was 76. However, the acceptance rate has decreased in the last two years, a very common phenomenon in events along the years. Among the papers, only four papers are written in English.

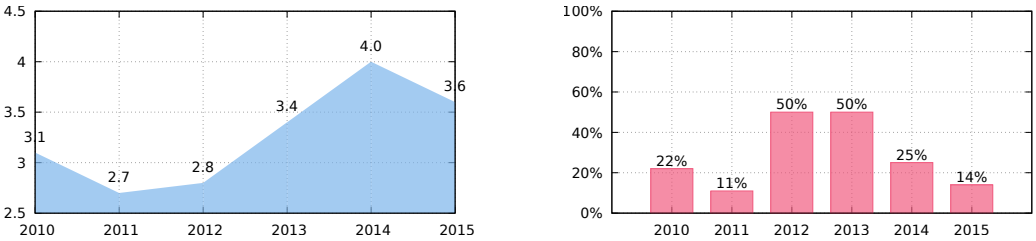
Table 3 shows the list with the top seven WESB authors, corresponding universities and number of publications. We identified 83 different authors and two of them lead the list with 12 published papers each one: Jerffeson Texeira de Souza from UECE and Silvia Regina Vergilio from UFPR. By analyzing the authors and universities, we identified eight main research groups, from the following universities: UECE, UEM, UFAM, UFG, UFPB, UFS, UFPR, and UNIRIO. Most of them have been publishing in WESB since its first edition, they are UECE, UFAM, UFPB, UFPR and UNIRIO. The group from UEM has been publishing since 2012, and the ones from UFS and UFG has appeared recently, publishing since 2013.

We collected the amount of authors in each paper and calculated the mean of authors over the years. This information is presented in Figure 1(a). It shows an increase in the collaboration between researchers over the years. In addition, we also analyzed the collaboration of researchers from different universities. We consider that there is a collaboration between universities if two or more researchers from different universities are authors of a same paper. In this sense, Figure 1(b) shows the percentage of papers published in collaboration between universities in each year.

In general, the collaboration between universities has increased after the first two editions of WESB (2010 and 2011). Besides, 2012 and 2013 were the years containing

Table 3. Top seven authors.

#	Author	University	# of Papers
1	Jerffeson Teixeira de Souza	State University of Ceará (UECE)	12
	Silvia Regina Vergilio	Federal University of Paraná (UFPR)	12
3	Márcio de Oliveira Barros	Federal University of the State of Rio de Janeiro (UNIRIO)	8
4	Thelma Elita Colanzi	State University of Maringá (UEM) / UFPR	7
5	Arilo Claudio Dias Neto	Federal University of Amazonas (UFAM)	6
6	Celso Gonçalves Camilo Junior	Federal University of Goiás (UFG)	5
	Glêdson Elias	Federal University of Paraíba (UFPB)	5



(a) Mean of authors by paper

(b) % of papers published in collaboration

Figure 1. Collaboration between researchers and universities over the years

the greater number of collaborations, totalizing 50% of the papers. To provide a more deep analysis, Figure 2 shows a network of the collaboration between universities. The stronger the line, the greater the number of collaborations between the universities. The greater the circle, the greater the number of publications.

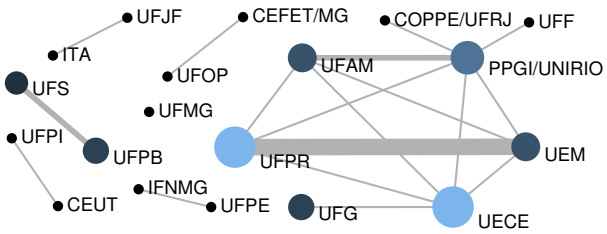


Figure 2. Collaboration Network.

We observe that about 50% of the collaboration studies ensued from the cooperation between advisors and advisees, which is the sort of collaboration that takes place in the Brazilian academic setting. It also turns out that once advisees establish their own research groups, most of them carry on maintaining a long-term cooperation with their former research group. This is the case observed in the cooperation between UFPR and UEM with five common papers. We can observe other strong lines representing cooperation between UNIRIO and UFAM, and UFS and UFPB with two papers. Besides, individually, UNIRIO is the university that has collaborated with a great number of distinct universities (six universities: UFF, COPPE/UFRJ, UFPR, UFAM, UECE and UEM).

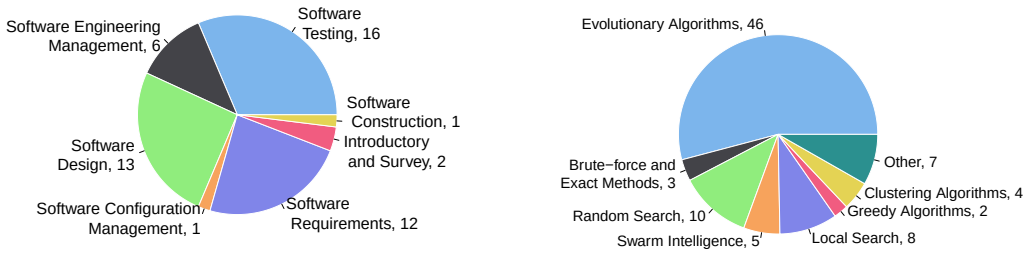
The papers with the greatest number of authors were published in 2013 and 2014. The studies are entitled *Mapeamento da Comunidade Brasileira de SBSE* (in English: "Brazilian SBSE Community Mapping") and *SBSTFrame: uma proposta de framework*

para o teste de software baseado em busca (in English: "SBSTFrame: A framework proposal for the search-based software testing") with seven authors each one. Notice that the first paper involved five universities.

3.2. RQ2 – SE Areas and Algorithms

We can observe by analyzing Figure 3 that six SE areas were addressed in all editions. The most addressed is Software Testing (in 16 studies (31.1%)) followed by Software Design and Software Requirements with 13 (25.5%) and 12 (23.5%) studies respectively. These results are similar from those found in SBSE surveys [Harman et al. 2012]. Traditionally the most addressed area in SBSE is Software Testing.

Regarding the algorithms, the most used are Evolutionary Algorithms (EAs) with 46 studies (54.1%) followed by Random Search and Local Search (as Hill Climbing) with 10 (11.8%) and 8 (9.4%) studies respectively. Among the EAs, the most preferred is NSGA-II, addressed by 17 (37%) of the studies. It is followed by Genetic Algorithms (GA), used in 11 (23.9%) studies. This preference for EAs is also observed in SBSE surveys [Harman et al. 2012]. In our analysis, Clustering Algorithms category includes algorithms such as K-means, K-medoids, and Hierarchical ones, and the Swarm Intelligence category includes the algorithms Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO) and Multi-Objective PSO (MOPSO).



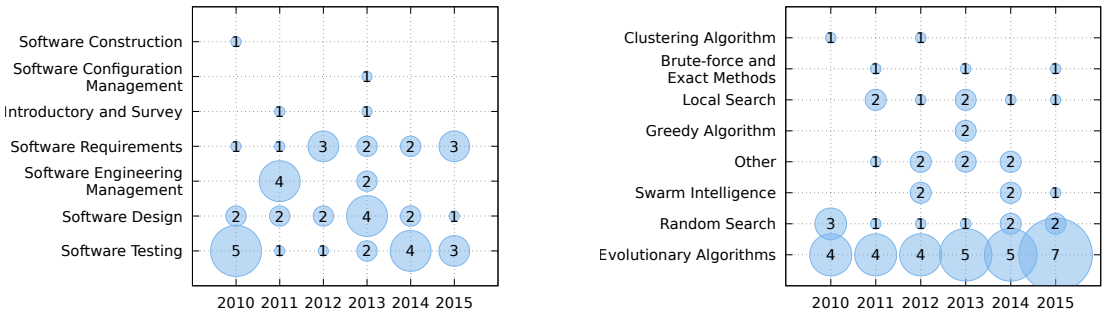
(a) SE areas. (b) Algorithms.

Figure 3. SE areas and algorithms addressed.

We observe similarities between the SBSE Brazilian research and the international one. However, we notice that other areas have raised interest in the international SBSE community and to analyze if this has been happening in Brazil, we use Figure 4 that shows the SE areas and algorithms addressed along the WESB editions. Regarding the SE areas, some of them were addressed in all editions such as Software Testing, Software Design, and Software Requirements. On the another hand, SE areas as Software Construction and Software Configuration Management were addressed only once, and Software Engineering Management area shows a decrease regarding to the number of publications. Concerning the algorithms, the Evolutionary Algorithms (EAs) and Random Search were used in all editions, especially EAs that show a growing interest in the last years. Differently, Clustering Algorithms were used only in the first WESB editions.

3.3. RQ3 – Types of Studies

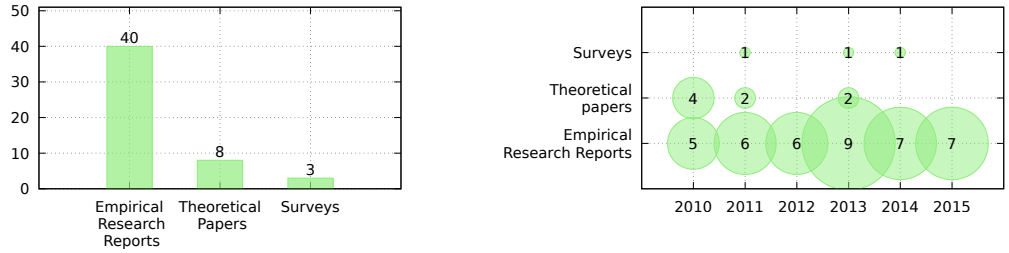
Figure 5 shows that we found studies of only three categories. Most of them belong to the category Empirical Research Report, with 40 (78.4%) studies, followed by Theoretical Papers category, with 8 (15.6%) studies. Only three studies are surveys. The Empirical



(a) SE areas per year. (b) Algorithms per year.

Figure 4. SE areas and algorithms addressed along the years.

Research Report category may be divided into five categories: case studies; experimental papers; field studies; observational studies; and meta-analysis. Most studies in the Empirical Research Report category can be classified as experimental papers.



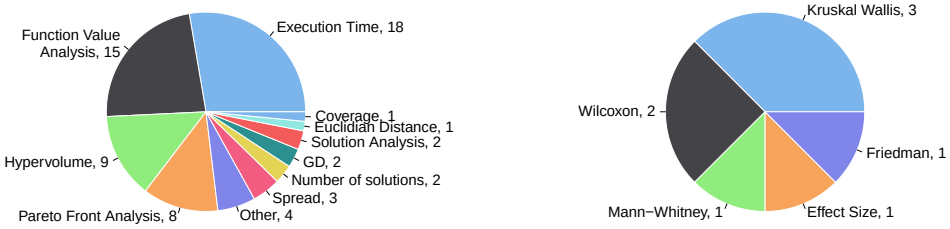
(a) Study types. (b) Study types per year.

Figure 5. Types of studies and the ones addressed along the years.

Besides, we observe that papers in the category Theoretical Papers were published in the first editions of the conference and, currently, most papers belong to the Empirical Research Reports category. This can be observed in Figure 5(b) that shows the types of studies along the years. This maybe indicates the event has reached a maturity.

3.4. RQ4 – Evaluation

To answer this question, only experimental papers were analyzed. Figure 6 shows the evaluation methods and statistical tests addressed by the papers.



(a) Evaluation Methods. (b) Statitical tests.

Figure 6. Evaluation methods used.

Many papers analyze the function value obtained for the solution (*Function Value Analysis*) and the Execution Time (18 and 15 papers respectively). Related to the multi-objective context, five quality indicators are used (*Hypervolume*, *Spread*, *Generational Distance*, *Coverage* and *Euclidean Distance*), being *Hypervolume* the most used one (9 studies). In addition, two papers show the number of solutions obtained by the algorithms, and eight manually analyze the obtained Pareto Front (*Pareto Front Analysis*). Two other papers manually analyze the obtained solution structure. Other four studies are related to specific metrics for evaluating clustering algorithms. Regarding the statistical test, only eight papers address these ones. Hence, the tests Kruskal Wallis and Wilcoxon are used, respectively, by three and two papers. Effect Size, Mann-Whitney and Friedman are used by only one paper.

For the studies that conducted an evaluation, we classified them according to the type of instances. We observe that 20 (50%) studies use artificial instances. However, the difference is not significant since real instances are used in 19 studies (47,5%). One study used both types of instances. This finding is different from that one reported by Barros and Dias-Neto [2011]. The authors claim that few SBSE studies involve real instances.

4. Discussion

Additional results of our review are in Figure 7, which shows the number of publications considering SE areas and algorithms by research groups. Works from UECE most address Software Requirements, with 7 articles. Furthermore, works from UFPR address Software Design and Software Testing areas, with 6 and 5 papers respectively. Regarding the algorithms, we can observe a preference for evolutionary ones (EAs) in UECE and UFPR universities, followed by UFG, UEM, and UNIRIO.

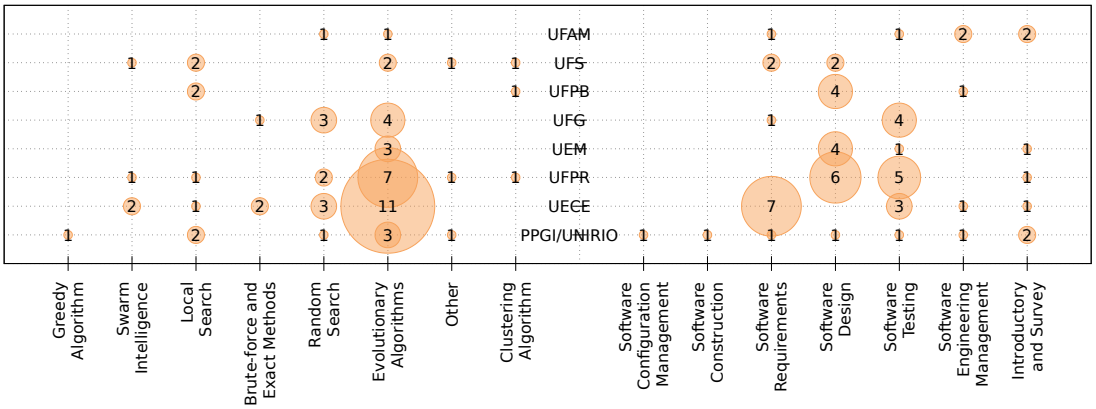


Figure 7. SE areas and algorithms by university.

During the conduction of this review, some trends and research opportunities were identified. For example, only six SE areas were investigated. Comparing the SE areas and universities, the results show that a preference or domain of some SE areas by some of them. Although other SE areas such as Software Engineering Models and Methods, and Software Maintenance may be addressed, since they were already explored by the SBSE community in other events. Thus, it can raise interest of other researchers by WESB, and as a consequence, it can increase the demand and popularity of the workshop. Regarding

the algorithms, the EAs are the preferred ones. However, other algorithms such as ACO may be investigated in other SE areas, for example, Software Design.

Finally, with respect to the study type, we observe that the number of studies that conduct experiments has been increasing over the last years. But we find a lack of studies in the category Experience Papers and Papers Oriented Towards Practice. We think this is a research opportunity. In addition, the studies explore different evaluation methods, but a few employed statistical tests to validate the results. Hence, it may be a direction for future research. To investigate the usefulness of the proposed approaches in practice is an important future research subject.

5. Related Work

There are some surveys [Harman et al. 2009, Harman et al. 2012] reviewing the SBSE field. They describe papers related to different SE areas, such as maintenance, testing, design, and quality. These surveys also present the most used algorithms, trends, and directions for the field. Other paper [Freitas and Souza 2011] present a bibliometric analysis of the field by collecting different data, such as authors and collaborations.

Works most related to ours are regarding the field in Brazil. Assunção et al. [2013] present an overview showing the main researchers, groups and algorithms employed. The authors collected primary studies of the SBSE field containing at least one Brazilian author. Other papers [Colanzi et al. 2013, Vergilio et al. 2011] also present an overview of the field in Brazil. Nevertheless, these papers are based on primary studies published only in Brazil, by analyzing the Brazilian Symposium of Software Engineering (SBES) and the first two editions of WESB.

Although WESB is included in some of the mentioned works, they do not address papers published in more recent editions. Furthermore, there is no paper presenting an overview of the workshop such as types of studies and evaluation conducted. No one of them present results related to the workshop evolution considering a timeline. This is one of the motivations and contributions of our paper.

6. Concluding Remarks

This paper presents results from a review of the papers published in the six editions of WESB. The main goal is to investigate the workshop evolution and impacts, and characterize the SE areas and algorithms addressed, as well as the main research groups.

The review adopted a method to extract information of the papers and to define a classification schema. The results show a predominance of works that address the areas of Software Testing, Software Requirements and Software Design. Besides, EAs are the preferred algorithms. Furthermore, most studies belong to the category of Empirical Research Reports in which experimental papers are the most common. In this context, it is important to notice an increase in the number of papers belonging to the category related to Empirical Research. This improves the reliability of the proposals being published, since this means that the proposed ideas are generally evaluated and points out a maturity in the SBSE works in Brazil. As future work, we intend to analyze more specific characteristics of the published studies.

The results herein presented show WESB plays a fundamental role to the consolidation of the area in Brazil, to increase collaboration, and incentive the researchers to

create research groups on SBSE in different universities. In addition to this, we observe such unexplored areas such as Maintenance, Software Development Tools, and so on. Finally, as future work, we pretend to extend this study by seeking to understand the interest in each SE area by analyzing deeply some particularities, such as the addressed problems of each area and the algorithms used to solve them.

Another result of our review shows similarities with the international community such as SE areas and algorithms addressed. In a future work we intend to explore other aspects, relating WESB with SSBSE, the international event in the area. In this sense, other points that should be investigated in future are regarding the impact of the Brazilian community in the international scenario. We have known that this community has been involved in collaborative research networks, inside and outside Brazil, and has published a number of papers in top conferences and has educated a new generation of researchers.

References

Assunção, W. K. G., Barros, M. d. O., Colanzi, T. E., Dias-Neto, A. C., ao, M. P., de Souza, J. T., and Vergilio, S. R. (2013). Mapeamento da Comunidade Brasileira de SBSE. In *Proceedings of 4th WESB'13*, pages 46–55.

Barros, M. d. O. and Dias-Neto, A. C. (2011). Threats to Validity in Search-based Software Engineering Empirical Studies. Technical report, Postgraduate Information Systems Program - UNIRIO.

Bourque, P. and Fairley, R. E. (2014). *Guide to the Software Engineering Body of Knowledge (SWEBOK (R)): Version 3.0*. IEEE Computer Society Press, 3rd edition.

Colanzi, T. E., Vergilio, S. R., Assunção, W. K. G., and Pozo, A. (2013). Search Based Software Engineering: Review and analysis of the field in Brazil. *Journal of Systems and Software*, 86(4):970–984.

Freitas, F. G. d. and Souza, J. T. d. (2011). Ten years of search based software engineering: A bibliometric analysis. In *Proceedings of the 3rd SSBSE'11*, pages 18–32.

Harman, M. and Jones, B. F. (2001). Search-Based Software Engineering. *Information and Software Technology*, 43:833–839.

Harman, M., Mansouri, S. A., and Zhang, Y. (2009). Search Based Software Engineering: A Comprehensive Analysis and Review of Trends Techniques and Applications. Technical report, Department of Computer Science, King's College London.

Harman, M., Mansouri, S. A., and Zhang, Y. (2012). Search-based Software Engineering: Trends, Techniques and Applications. *ACM Computing Surveys*, 45(1):11:1–11:61.

Montesi, M. and Lago, P. (2008). Software engineering article types: An analysis of the literature. *Journal of Systems and Software*, 81(10):1694–1714.

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

Vergilio, S. R., Colanzi, T. E., Pozo, A. T. R., and Assunção, W. K. G. (2011). Search Based Software Engineering: A Review from the Brazilian Symposium on Software Engineering. In *Brazilian Symposium on Software Engineering*, pages 50–55.

Uma Proposta para Alocação de Requisitos em Times Ágeis Utilizando Programação Inteira Mista

Victor José Aguiar Teixeira de Melo França[1,3], Mariana Alves Moura[2], Silvana Bocanegra[3], Ana Cristina Rouiller[3]

[1]SWQuality Consultoria e Sistemas (SWQ)
Rua do Apolo, 202, Recife Antigo, Recife - PE, Brasil

[2]Centro de Informática - Universidade Federal de Pernambuco (Cin-UFPE)
Av. Prof. Moraes Rego, 1235 - Cidade Universitária, Recife - PE -
CEP: 50670-901

[3]Departamento de Estatística e Informática - Universidade Federal Rural de Pernambuco (DEINFO-UFRPE)
Rua Dom Manuel de Medeiros, s/n, Dois Irmãos, Recife - PE, Brasil

victorjatmf@gmail.com, marianalvesmoura@gmail.com
silvana@deinfo.ufrpe.br, anarouiller@gmail.com

Resumo. *Neste artigo estamos propondo uma ferramenta baseada em programação inteira mista para resolver o Next Release Problem em empresas de manutenção e evolução de software que utilizam Scrum. O objetivo da ferramenta é auxiliar o gerente de projetos na tomada de decisões para alocação de requisitos entre suas equipes de desenvolvimento, com foco em três funções objetivos distintas: a primeira maximiza o percentual de satisfação dos clientes, tendo como parâmetro o Business Value dos requisitos, a segunda minimiza o tempo de desenvolvimento e a terceira minimiza o custo. As restrições envolvem capacidade da equipe, dependências entre os requisitos, entre outras. Como estudo de caso, foram selecionadas três empresas do estado da Bahia. Com os resultados foi possível adequar o NRP à realidade destas organizações, focando no planejamento da próxima iteração de acordo com os objetivos estratégicos da empresa em atender seus clientes da melhor forma.*

Abstract. *In this article we are proposing a tool based on mixed integer programming to solve the "Next Release Problem" in development and maintenance software companies that use Scrum. With it, the project manager will be supported in making decisions based on allocation of requirements among its development teams, focused on three different objective functions: the first in order to maximize customer satisfaction percentage, having as parameter the requirements Business Value, the second minimizes the development time and the third minimizes costs. The restrictions involve staff capacity, dependencies between requirements, among others. As a case study, three companies from Bahia were selected. The results show it was possible to adapt the NRP to reality of these organizations, focusing on the next iteration planning in accordance with the strategic objectives of the company in attend its customers in the best way.*

1. Introdução

O *Next Release Problem* (NRP), ou Problema do Próximo *Release* é originado da dificuldade enfrentada pelas equipes de desenvolvimento para selecionar o conjunto de requisitos que irá compor a *release* seguinte, levando em consideração que estes requisitos são demandados por diferentes clientes, com graus de importância variados para o negócio. Cada requisito possui um custo de desenvolvimento e o conjunto selecionado deve satisfazer uma limitação de custos pré-definida. Assim, o problema é selecionar um conjunto ideal de requisitos que satisfaça os clientes da melhor maneira e respeite a limitação de custos da organização. A formulação original do NRP foi apresentada como um problema de otimização mono-objetivo cuja função maximiza a satisfação dos patrocinadores do projeto de software [Bagnall et al. 2001]. Em 2007 foi proposta uma formulação multiobjetivo que maximiza a satisfação e minimiza o custo de execução [Zhang et al. 2007]. A função de satisfação dos clientes considera não só a importância de cada cliente, mas também o nível de importância que cada requisito tem para cada cliente. Técnicas de otimização exata foram aplicadas e comparadas com metaheurísticas, algoritmos genéticos e experiência humana. As técnicas exatas e as metaheurísticas geraram resultados similares nas instâncias testadas, porém os obtidos com a técnica exata foram superiores. A experiência humana gerou resultados de 18% à 40% piores do que as técnicas de otimização exata [Freitas et al. 2011]. Assim, nota-se a necessidade do uso de técnicas de otimização para auxiliar os gestores na solução NRP. No entanto, o entendimento dos modelos matemáticos e métodos de solução e sua adequação à realidade das empresas são limitações para o uso.

Neste trabalho está sendo proposta uma ferramenta de apoio à decisão para auxiliar na solução do NRP em empresas que utilizam *Scrum* [Schwaber and Sutherland 2011] para o gerenciamento de projetos *Software*. Para isso, foi necessário mapear as especificidades de algumas dessas empresas para adequar a formulação do NRP. Após a coleta de informações foram propostos modelos usando novas funções objetivo e restrições que representam melhor a realidade destas empresas. Os modelos foram implementados e resolvidos utilizando o AIMMS[1]. Foi proposta uma interface gráfica para gerar cenários que podem auxiliar os gestores do projeto na tomada de decisão de priorização de requisito e alocação de equipe. Como estudo de caso particular, a ferramenta foi aplicada a problemas reais, com dados obtidos em três empresas da região nordeste.

Este trabalho está organizado da seguinte forma. Na Seção 2 são apresentados alguns trabalhos relacionados e listadas as principais contribuições dessa pesquisa. A Seção 3 apresenta uma proposta de modelagem para o NRP em empresas que utilizam *Scrum*. A solução para os modelos desenvolvidos e uma interface gráfica inicial estão apresentadas na Seção 4. A Seção 5 traz resultados obtidos com a ferramenta em um estudo de caso com problema real. Finalizando têm-se as Considerações Finais e Referências Bibliográficas.

2. Trabalhos Relacionados e Contribuições da Pesquisa

Em 2001, [Bagnall et al. 2001] apresentaram a formulação do NRP usando Programação Linear Inteira e o resolveram com métodos exatos e metaheurísticas. Desde então, uma variedade de formulações mono-objetivo e multiobjetivo foram propostas.

[1] http://aimms.com/

[Feather and Menzies 2002] propuseram uma abordagem iterativa para otimização de requisitos envolvendo a tomada de decisão com a experiência humana. O grande conjunto de dados produzido é em seguida condensado em uma pequena lista de decisões críticas, que são apresentadas para que a expertise humana selecione. A cada iteração, os especialistas selecionam algumas opções para diminuir as alternativas e produzir um conjunto quase ótimo de requisitos. A maior parte dos trabalhos encontrados na literatura utiliza metaheurísticas para solucionar o problema. Apesar desses algoritmos serem mais flexíveis e populares, são inexatos e portanto não podem ser usados com inteira garantia em determinadas aplicações reais. Trabalhos mais recentes mostram que técnicas exatas podem ser usadas com sucesso na solução do NRP. [Harman et al. 2014] utilizam um algoritmo de programação dinâmica para análise de sensibilidade em instâncias reais do NRP e disponibilizam uma ferramenta para essa aplicação. [Veerapen et al. 2015] mostram que grandes instâncias mono-objetivas e pequenas instâncias bi-objetivas podem ser solucionadas rapidamente com métodos exatos. No caso de instâncias bi-objetivas de maior dimensão, um algoritmo de aproximação baseado em Programação Linear Inteira supera a abordagem genética NSGA-II e os tempos de operação para ambos os métodos são baixos o suficiente para serem usados em situações reais. Em nosso trabalho, estamos usando um método exato para resolver uma variante do NRP aplicada a empresas reais. As principais contribuições da pesquisa são:

1. Modelagem e solução do NRP para empresas que usam *Scrum* - Neste modelo, são consideradas restrições de dependência entre requisitos, custos e tempo de desenvolvimento. No entanto, a satisfação dos clientes é calculada de forma qualitativa, e não quantitativa, como abordada nos trabalhos anteriores.
2. Desenvolvimento de uma ferramenta para auxiliar gestores de empresas de softwares na tomada de decisões. A ferramenta proposta possibilita a geração de cenários que auxiliem a escolha do conjunto de requistos e a composição das equipes de desenvolvimento considerando diferentes prazos e orçamentos.

3. Proposta para modelagem do NRP em empresas que usam *Scrum*

Uma das metodologias ágeis para desenvolvimento de software mais difundias é o *Scrum* [Schwaber and Sutherland 2011]. Tal metodologia divide a gestão do desenvolvimento em partes menores e de mesma duração, conhecidas como *sprints*, que representam um ciclo de trabalho. A cada *sprint* um conjunto de requisitos é implementado, tendo como resultado um incremento do produto que está sendo desenvolvido. Os requisitos devem ser selecionados da melhor maneira para compor um destes ciclos de trabalho, de tal forma que o valor da satisfação dos clientes atendidos seja maximizado e as dependências entre requsitos sejam atendidas. Com a utilização deste processo, surge a necessidade de distribuir entre as *sprints* os requisitos para cada iteração, contudo, esta distribuição não é trivial. O custo de desenvolvimento por *sprint* é limitado pelo tempo de duração do ciclo e as empresas possuem vários clientes demandando ao mesmo tempo. Assim, faz-se necessário priorizar e escolher da melhor forma conjuntos de requisitos, que atendam essas limitações, desenvolvendo o necessário para manter os clientes satisfeitos, considerando a importância de cada um deles e as decisões estratégicas das organizações.

Para a adaptação do Problema do Próximo *Release* a empresas que usam *Scrum*, foram observados os processos de desenvolvimento e manutenção de três organizações de

software do estado da Bahia. Na Tabela 1 estão relacionados os termos usados na metodologia com os apresentados na formulação original do NRP. Os conjuntos, parâmetros e variáveis utilizados na modelagem estão descritos na Tabela 2. No caso destas empresas

Tabela 1. Associação dos elementos do NRP original ao do Estudo de Caso.

Problema original	Estudo de caso
Equipe de desenvolvimento.	Time.
Release.	*Sprint*.
Conjunto de requisitos.	*Backlog*.
Requisito.	História.
Custo de implementação do requisito.	Tamanho do requisito.
Custo máximo disponível por time.	Capacidade do Time.
Limitação de Custo para a Empresa .	Soma das Capacidades dos Times de desenvolvimento.

o foco não é em função dos clientes, e sim dos requisitos, assim, não precisa haver uma garantia de que todas as histórias de um cliente j sejam implementadas para que ele esteja satisfeito. O objetivo é calcular a satisfação do cliente de forma qualitativa e não quantitativa, ou seja, deve ser implementado o máximo possível de requisitos levando em conta o valor do requisito para o cliente (vcr_{ij}). No modelo tradicional, a satisfação global do cliente para a empresa é dada por uma varíavel y_j que assume 1 quando todos os requisitos do cliente foram selecionados para o *release* ou 0 caso contrário. Na adequação realizada, esta variável poderá assumir valores reais entre 0 e 1, que indicam a porcentagem de satisfação, e pode ser formalizada como:

$$y_j = \sum_{i \in Requisitos}^{n} pvrc_{ij} * r_i.$$

Estamos propondo o uso de três funções objetivo para auxiliar o gerente de projetos na alocação de requisitos em seus times para a *sprint*, podendo:

- Maximizar a satisfação do cliente

$$max \sum_{j \in Clientes} vc_j * y_j.$$

- Minimizar o tempo de desenvolvimento

$$min \sum_{i \in Requisitos} \sum_{t \in Times} tempoDev_{it} * x_{it}.$$

- Minimizar o custo de desenvolvimento

$$min \sum_{i \in Requisitos} \sum_{t \in Times} tempoDev_{it} * valorHora_t * x_{it}.$$

Sujeitas às seguintes restrições:

1. Disponibilidade do Time - A disponibilidade dos times de desenvolvimento não pode ser extrapolada:

$$\sum_{i \in Requisitos} tempoDev_{it} * x_{it} \leq disp_t, \quad \forall t \in Times.$$

Tabela 2. Notação da modelagem do problema.

Conjuntos	**Descrição**
$Requisitos$	Conjunto de requisitos a serem desenvolvidos, i = {1,2,...,n}.
$Clientes$	Conjunto de clientes com valor para a empresa, j = {1,2,...,m}.
$Times$	Conjunto de times de desenvolvimento disponíveis na empresa, t = {1,2,...,s}.
Parâmetros e Símbolos	**Descrição**
vrc_{ij}	É valor do requisito i para o cliente j, podendo variar de 0 a 10 de acordo com a sua importância.
vc_j	É o valor do cliente j para a empresa.
$pvrc_{ij}$	É o percentual da importância do requisito i para o cliente j. ($\frac{vrc_{ij}}{vtr_j}$).
vtr_j	É a soma dos valores atribuídos a cada requisito i do cliente j. Estes valores são determinados pelo grau de importância que cada requisito i tem para o cliente j ($\sum_{i=1}^{n} vrc_{ij}$).
$tamTime_t$	Número de integrantes time t.
$capPPP_t$	Representa a capacidade máxima de desenvolvimento do time t em pontos do *Planning Poker*.
$dursprint$	O tempo em horas disponíveis no *time-box* da *sprint*.
$disp_t$	Representa a disponibilidade em horas produtivas levando em consideração. os integrantes do time t. ($tamTime_t * dursprin.$).
tam_i	Representa o tamanho em pontos do *Planning Poker* que cada requisito i possui.
$horaPonto_t$	Representa a estimativa em horas da duração de um ponto do *Planning Poker* no time t, ou seja, quantas horas são necessárias para a implementação de um ponto. ($\frac{disp_t}{capPPP_t}$).
$tempoDev_{it}$	É o tempo que o requisito i leva para ser desenvolvido pelo time t. $tam_i * horaponto_t$.
$valorHora_t$	É o valor pago pela empresa por hora de desenvolvimento do time t.
$G(R, E)$	Representa a dependência entre os requisitos, como no modelo tradicional. O grafo é direcionado e acíclico. $(r, r') \in (G)$ se e somente se r é um pré-requisito de r'.
$G1(R, E)$	Representa os requisitos que devem ser implementados juntos (And). O grafo não é direcionado e pode haver ciclos. $(r, r') \in E(G)$ se e somente se r e r' assumem o mesmo valor. Ambos são selecionados ou não.
$G2(R, E)$	Representa os requisitos conflitantes (XOr). O grafo não é direcionado e pode haver ciclos. $(r, r') \in E(G)$ se e somente se r e r' assumem valores distintos. Se um deles é selecionado, o outro não é.
Variáveis	**Descrição**
r_i	Variável binária que assume o valor 1 se o requisito i for selecionado para a *sprint*.
x_{it}	Variável binária que assume o valor 1 quando o requisito i é implementado pelo time t.
y_j	Variável que pode assumir valores reais no intervalo $[0, 1]$ e que indica o percentual de satisfação do cliente j. Por exemplo, se y_j=1, todos os requisitos do cliente j são selecionados.

2. Requisito Implementado por Apenas um Time - Cada requisito i só pode ser alocado em apenas um time t:

$$\sum_{t \in Times} x_{it} = r_i, \quad \forall i \in Requisitos.$$

3. Precedência entre Requisitos - O requisito r é pré-requisito do requisito r':

$$x_r \geq x_{r'}, \quad \forall (r, r') \in E(G).$$

4. Requisitos que devem ser Implementados Juntos - Quando um requisito r é selecionado, o requisito r' também tem que ser escolhido:

$$x_r - x_{r'} = 0, \quad \forall (r, r') \in G1(R, E).$$

5. Requisitos Conflitantes - Os requisitos r e r' são conflitantes entre si. Somente um entre r e r' pode ser selecionado:

$$x_r + x_{r'} = 1, \quad \forall (r, r') \in G2(R, E).$$

4. Ferramenta de apoio à decisão para o NRP - uma proposta inicial

O modelo matemático foi implementado na linguagem de modelagem matemática do software de otimização AIMMS. Este software permite o desenvolvimento avançado de aplicações baseadas em pesquisa operacional e pode ser obtido gratuitamente para propósitos acadêmicos. Ele inclui diversos pacotes de otimização (CPLEX, MINOS, XPRESS, entre outros) para solucionar os principais tipos de programação matemática tais como programação linear, programação inteira mista e programação não-linear. Além disso, está integrado com uma interface gráfica completa, tanto para os desenvolvedores como para usuários finais.

Os modelos propostos se enquadram na classe de problemas de programação inteira mista e foram solucionados no AIMMS com o *solver* CPLEX versão 12.6.2. O CPLEX[2] é um *solver* que hoje pertence à IBM e resolve problemas de programação inteira, problemas grandes de programação linear, problemas de programação quadrática convexos e não-convexos e problemas de programação inteira mista. Para solução de problemas de programação inteira mista, o CPLEX disponibiliza um algoritmo do tipo *Branch-and-Cut*. Como padrão, o CPLEX escolhe entre um algoritmo de busca dinâmica para escolha dos nós ou o algoritmo *Branch-and-Cut* convencional. A escolha automática é feita com base nas características do modelo.

A Figura 1 apresenta a tela do AIMMS com o modelo matemático. Foram inseridos os modelos com as três funções objetivos, as restrições, variáveis e parâmetros apresentados.

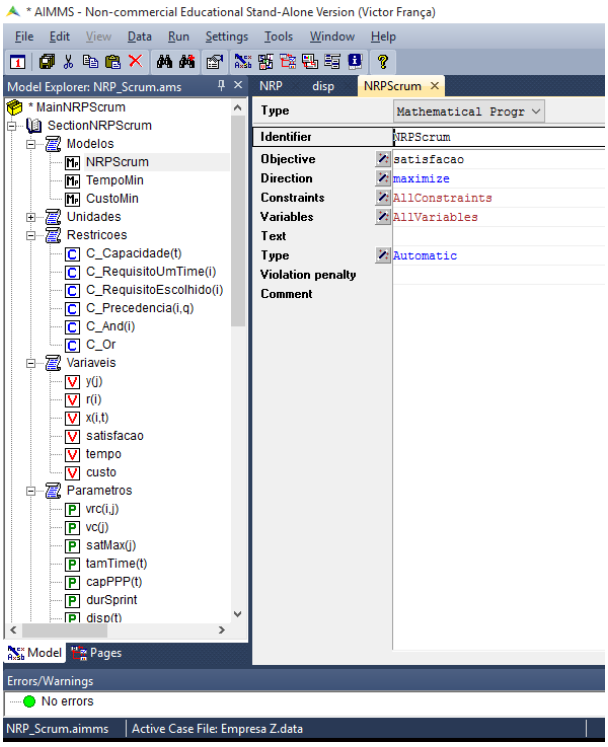


Figura 1. Modelagem do Sistema no AIMMS.

[2]http://main.aimms.com/aimms/solvers/cplex/

A interface inicial é apresenta na Figura 2. A esquerda da tela, encontra-se a alocação dos requisitos selecionados para a próxima *sprint* em cada time, assim como seu tempo de desenvolvimento. Também é possível escolher qual das três funções objetivos se deseja executar, utilizando os botões, nrp, tempo e custo. É possível observar o percentual de satisfação dos clientes para a empresa, o custo do desenvolvimento da sprint e o tempo em horas da iteração.

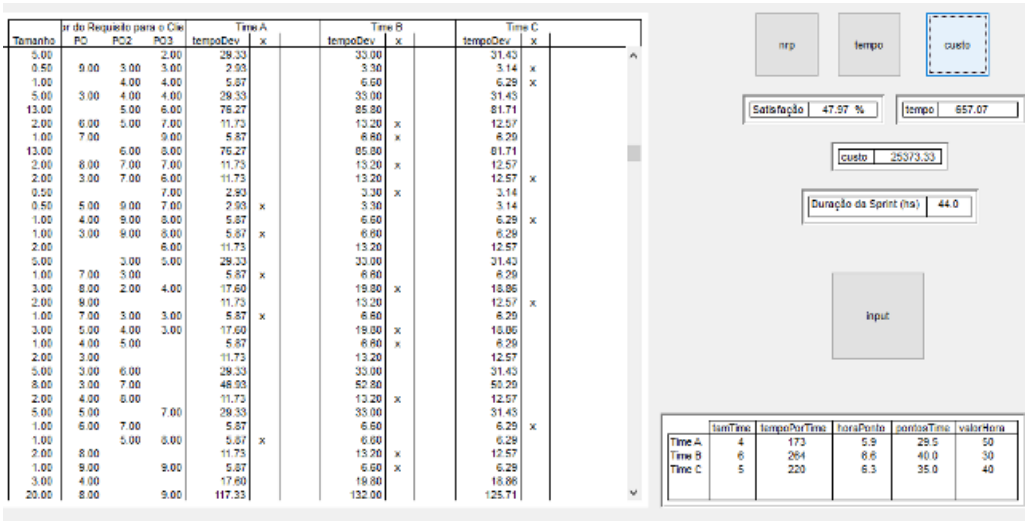


Figura 2. Interface Inicial.

Por fim, na Figura 3 são apresentadas as informações técnicas referentes à execução do modelo matemático no AIMMS, como por exemplo, o *Solver* utilizado (CPLEX 12.6.2), o número de iterações (816), o tempo total da execução (0,28s).

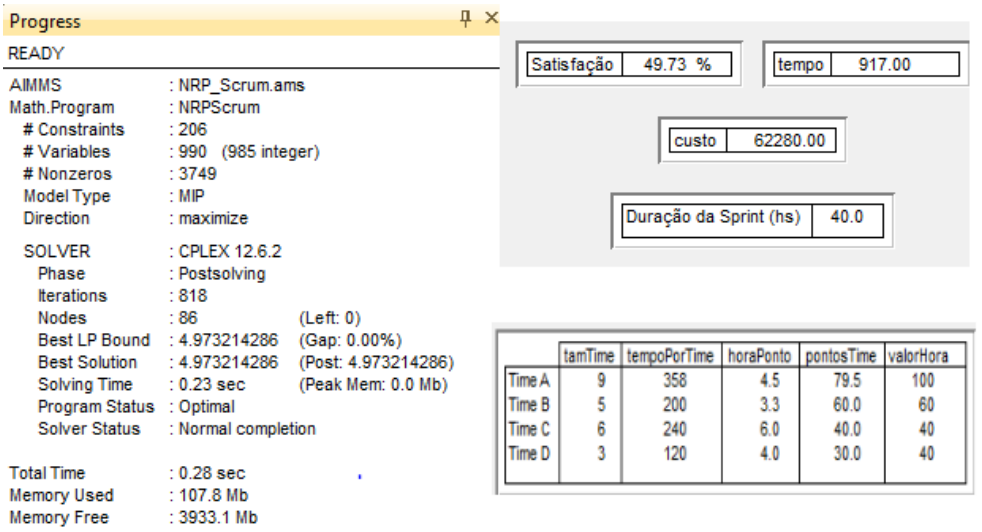


Figura 3. Execução do Modelo Minimizando o custo.

5. Estudo de Caso

As empresas selecionadas estavam passando por consultoria para melhoria de processos de desenvolvimento de software alinhados ao nv.2 de maturidade no *Capability Maturity Model Integration for development* - CMMI-DEV, [Institute 2010] com a implantação e maturação da metodologia ágil *Scrum* iniciada em novembro de 2014. Apesar dos testes terem sido realizados com todas as empresas, selecionamos apenas um delas para reportar os resultados, já que as análises realizadas foram similares e nosso objetivo é apenas ilustrar o uso da ferramenta proposta. A empresa selecionada possui duração de sprint semanal, ou seja 40 horas para entrega dos requisitos alocados para a sprint. Os custos de desenvolvimento por time são definidos por produtividade, sendo o time mais caro, aquele com melhor desempenho baseado no histórico obtido com o gerente de projetos. Para gerar cenários de desenvolvimento foi realizada uma variação da duração da sprint semanal para duas e três semanas com o objetivo de verificar a distribuição dos pontos por time nas diferentes funções objetivo, minimizando o tempo e o custo. No momento da coleta das informações, a empresa estava com 198 itens no seu *backlog* para serem desenvolvidos, estimados e priorizados. Na Tabela 3 são apresentados os times da empresa, quantos membros cada time possui, qual a relação hora por ponto e o valor pago pela hora em cada time:

Tabela 3. Informações dos times da Empresa X.

Time	Tamanho do Time	Hora por Ponto	Valor da Hora
Time A	9	4,5	R$100,00
Time B	5	3,5	R$60,00
Time C	6	6,0	R$40,00
TIme D	3	6,0	R$40,00

Foi realizada a variação da duração da *sprint* entre uma, duas e três semanas com as funções objetivo de minimizar o tempo e em seguida minimizar o custo. Os resultados estão apresentados na Figura 4. Como há mais requisitos no *backlog* do que a capacidade

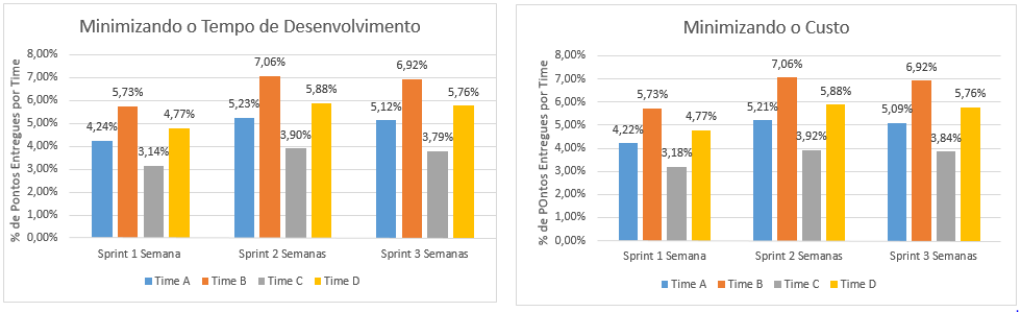


Figura 4. Minimizando o Tempo e o Custo para a Empresa X.

disponível de desenvolvimento das equipes, o modelo aloca a maior quantidade possível de requisitos em cada time, ou seja, a capacidade alocada tende a ser a mais próxima possível da capacidade disponível. Quando o objetivo é minimizar o custo percebe-se uma diminuição de pontos alocados por membro da equipe no Time A, que possui um valor de hora mais elevado e um aumento de pontos alocados ao time C, que possui

menor valor de hora. Essa diminuição poderia ser observada tanto no Time C quanto no D, que possuem a mesma produtividade e mesmo custo para a empresa.

Para uma segunda análise, foi realizada uma distribuição temporal com todos os itens do backlog em sprints semanais com a satisfacão dos clientes para a empresa, obtida em cada iteração, como mostrado na Figura 5. Podemos observar que seria necessário sete semanas e um custo de R$ 361.425,00 para desenvolver todo o *backlog* com os times existentes na organização. Não estão sendo consideradas novas entradas no *backlog*. Simulações de contratação de novos integrantes para os times foram realizadas. Primeiro

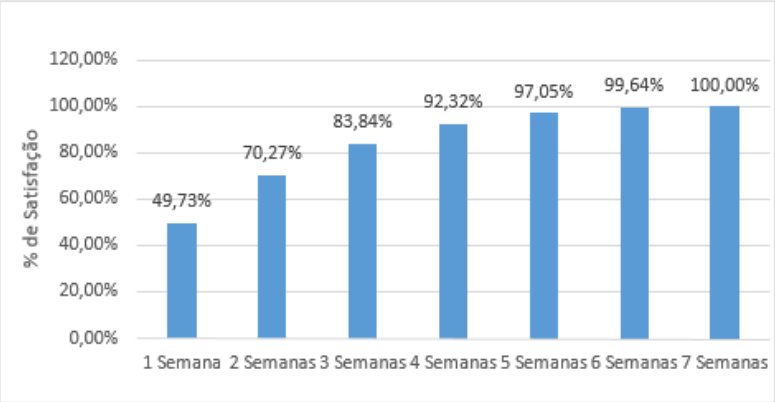


Figura 5. Distribuição do backlog.

contratando-se três membros para o time menos produtivo, e em seguida para o mais produtivo, como mostrado nas Tabela 4. Com isso é possível perceber que as empresas podem diminuir o tempo de desenvolvimento de todo o *backlog* sem necessariamente aumentar o seu custo, contratando membros para os times certos. E o maior percentual de satisfação é obtido sempre nas primeiras *sprints*, devido a priorização correta dos requisitos e sua estimativa realizada previamente. Esta análise é apenas um tipo de estudo que

Tabela 4. Simulação de Contratação na Empresa X.

Contratação	Duração	Custo
Formação Original	7 *sprints*	R$361.425,00
3 membros para o Time C	5 *sprints*	R$355.875,00
3 membros para o Time B	4 *sprints*	R$385.725,00

pode ser realizado com esta proposta de ferramenta. Além de alocar os requisitos dentre as equipes de desenvolvimento utilizando qualquer uma das três funções objetivos, pode-se realizar análises como por exemplo: dado um prazo mais curto pelo cliente, qual seria a hora extra necessária para atender o novo prazo de entrega da *sprint*; analisar individualmente a satisfação de cada cliente para decisões estratégicas de fidelização deles, entre outras análises. E também atende ao principal objetivo, que é planejar a próxima *Sprint*.

6. Considerações Finais e Trabalhos Futuros

O novo modelo traz como vantangens alguns conceitos ágeis utilizados nas organizações analisadas, como por exemplo o *Business Value* dos requisitos e seu tamanho dado pela

técnica *Planning Poker*[4]. A escolha dos requisitos é dada de forma qualitativa e não quantativa como no modelo tradicional e a satisfação dos clientes varia de acordo com o valor de negócio de cada requisito solicitado que é selecionado para a próxima *sprint*, e não pela quantidade. A partir das três funções objetivo do novo modelo (maximizar a satisfação, minimizar o tempo e minimizar o custo), é possível tomar decisões gerenciais estratégicas para um melhor planejamento da próxima iteração. Esta ferramenta está em fase inicial de desenvolvimento e por isso ainda não está disponível. São necessários testes adicionais para validação e escalabilidade do modelo e também aprimoramento da interface gráfica. A seguir são listadas algumas limitações da ferramenta que sevirão como propostas de trabalhos futuros:

- Realizar o planejamento de mais *sprints* com o backlog existente, não apenas da próxima;
- Ajustar reserva de capacidade para escopo não planejado;
- Montar a projeção temporal do planejamento das sprints;
- Mapear habilidades dos times para usar como parâmetro na alocação dos requisitos selecionados;
- Permitir a utilização do modelo com times que possuam diferentes durações de *sprints*;
- Melhoria na interface, proporcionando uma melhor experiência para o usuário;
- Utilizar a ferramenta em uma empresa de software para o planejamento das *sprints*.

Agradecimentos: Este trabalho foi parcialmente financiado pelo Instituto Nacional de Ciência e Tecnologia para Engenharia de Software (INES), fundado pelo CNPq.

Referências

Bagnall, A., V.J., R.-S., and Whittley, I. (2001). The next release problem. *Information and software technology*, 43(14):883–890.

Feather, M. and Menzies, T. (2002). Converging on the optimal attainment of requirements. *International Conference on Requirements Engineering*, pages 263–279.

Freitas, F., Coutinho, D., and Souza, J. (2011). Software next release planning approach through exact optimization. *International Journal of Computer Application*, 22(08).

Harman, M., Krinke, J., Medina-Bulo, I., Palomo-Lozano, F., Ren, J., and Yoo, S. (2014). Exact scalable sensitivity analysis for the Next Release Problem. *ACM Transactions on Software Engineering and Methodology*, 23(2):19:1–31.

Institute, C. (2010). *CMMI for Development, Version 1.3.* CARNEGIE MELLON, Software Engineering Institute.

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

Veerapen, N., Ochoa, G., Harman, M., and Burk, E. (2015). An linear programming approach to the single and bi-objective next release problem. *Information and Software Technology*.

Zhang, Y., Harman, M., and Mansouri, S. (2007). The multi-objective next release problem. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 1129–1137. ACM.

[4]https://www.planningpoker.com/

Uma Abordagem Multiobjetivo baseada em Otimização Interativa para o Planejamento de Releases

Raphael Saraiva, Allysson Allex Araújo, Altino Dantas, Jerffeson Souza

[1]Universidade Estadual do Ceará (UECE)
Avenida Dr. Silas Munguba, 1700. Fortaleza, Ceará. Brasil
Grupo de Otimização em Engenharia de Software da UECE
http://goes.uece.br
{raphael.saraiva}@aluno.uece.br
{allysson.araujo,altino.dantas,jerffeson.souza}@uece.br

Abstract. *The Release Planning (RP) is an important step in the iterative and incremental software development, because it involves deciding which requirements will be allocated in each release of the system. Recently, proposals based on the fusion of the concepts of Search Based Software Engineering and Interactive Optimization have been proposed to tackle such problem. However, there is a gap in this scenario regading the usage of multiobjective techniques. Therefore, this paper proposes a multiobjective approach in which the preferences of the Decision Maker are treated as one of the objectives to be optimized, thus providing a trade-off with other evaluated metrics. The approach also includes the usage of the Reference Point Method to decide which solution from the Pareto Front will be chosen. Regarding the performed experiments, it was evaluated the performance of two multiobjective evolutionary algorithms (NSGA-II and MOCell) and the random search. At the end, it was concluded that NSGA-II was superior and the method to choose the solution shown to be useful in an interactive scenario.*

Resumo. *O Planejamento de Releases (PR) é uma importante etapa no desenvolvimento de software iterativo e incremental, pois envolve a decisão de quais requisitos serão alocados em cada release do sistema. Recentemente, abordagens baseadas na fusão dos conceitos de Engenharia de Software Baseada em Busca e Otimização Interativa têm sido propostas para lidar com tal problema. Todavia, verifica-se neste cenário uma lacuna no que se refere ao uso de técnicas multiobjetivos. Portanto, o presente trabalho propõe uma abordagem multiobjetivo na qual as preferências do tomador de decisão são tratadas como um dos objetivos a serem otimizados, provendo assim um trade-off com as demais métricas avaliadas. A abordagem também inclui a utilização do Método do Ponto de Referência para decidir qual solução da Frente de Pareto será escolhida. Em relação aos experimentos realizados, avaliou-se a performance de dois algoritmos evolucionários multiobjetivos (NSGA-II e MOCell) e um algoritmo aleatório. Ao fim, constatou-se que o NSGA-II foi superior e que o referido método para escolha de solução mostra-se útil em um cenário interativo.*

1. Introdução

O Planejamento de *Releases* (PR) abrange todas as decisões relacionadas à seleção e atribuição de requisitos em uma sequência consecutiva de *releases* [Ngo-The and Ruhe 2008]. Definir um conjunto adequado de *releases* é inerentemente desafiador devido ao alto esforço computacional e cognitivo envolvido [Ruhe and Saliu 2005]. Nesta etapa, considera-se relevante conhecer os *stakeholders* envolvidos, as restrições técnicas, o orçamento disponível, além dos aspectos subjetivos intrínsecos ao negócio.

A Engenharia de Software Baseada em Busca (do inglês, *Search Based Software Engineering* - SBSE) aplica algoritmos de otimização para resolver problemas complexos da Engenharia de Software [Harman et al. 2012]. Porém, conforme destacado em [Ngo-The and Ruhe 2008], PR é caracterizado por envolver fatores subjetivos e dinâmicos, ou seja, difíceis ou impossíveis de serem modelados matematicamente. Além disso, quando o *Decision Maker*[1] (DM) não sente-se parte do processo de resolução, ele tende a se sentir excluído da análise e apresentar certa resistência ou pouca confiança no resultado final [Miettinen 2012]. Diante de tais particularidades, pode-se destacar a oportunidade em se agregar os conceitos oriundos da Otimização Interativa à SBSE. Através dessa perspectiva, torna-se factível incorporar as preferências do DM durante o processo de otimização e, consequentemente, refleti-las na solução final [Takagi 2001].

Recentemente, trabalhos baseados na fusão entre os conceitos de Otimização Interativa e SBSE têm sido propostos sob o contexto da Engenharia de Requisitos. Em [Araújo et al. 2016] propõe-se uma arquitetura para o Problema do Próximo *Release* (*Next Release Problem* - NRP), ou seja, focando apenas na seleção de requisitos para o próximo *release*. Tal proposta consiste em requisitar ao DM, durante um determinado período, notas subjetivas em relação às soluções, enquanto um Modelo de Aprendizado aprende seu perfil de avaliação. Posteriormente, este Modelo de Aprendizado substitui o DM e passa a fornecer as notas para as demais soluções. Ainda em relação ao NRP, em [Ferreira et al. 2016] é proposto um *Interactive Ant Colony Optimization*, onde solicita-se ao DM especificar quais requisitos ele espera que estejam presentes ou não no próximo *release*. Enquanto os trabalhos anteriores focam na seleção de requisitos, em [Tonella et al. 2013] apresenta-se uma abordagem interativa para Priorização de Requisitos, onde a interação com o usuário ocorre quando o algoritmo se depara com soluções igualmente boas, cuja qualidade não pode ser computada precisamente com as informações disponíveis.

Em relação ao planejamento envolvendo mais de uma *release*, em [Dantas et al. 2015] formaliza-se um modelo mono-objetivo que explora diferentes tipos de preferências as quais o DM pode expressar em relação à alocação de requisitos. Tais preferências são armazenadas em uma Base de Preferências cuja finalidade é influenciar o processo de busca. No experimento realizado verificou-se que as soluções geradas são aptas a satisfazer quase todas preferências, inclusive priorizando as mais importantes. Entretanto, percebeu-se também um certo conflito entre a otimização das métricas próprias do problema (importância e risco) e o atendimento das preferências

[1]Este trabalho considera o *Decision Maker* como o profissional imerso nas particularidades do projeto, capaz de gerenciar os diferentes conflitos de interesse entre os *stakeholders* e responsável pela decisões inerentes a etapa de planejamento.

humanas, sugerindo assim, um inevitável *trade-off* entre as mesmas.

Traçado este breve panorama, constata-se uma lacuna no que se refere ao uso dos conceitos de Otimização Interativa e algoritmos multiobjetivos para o PR. Portanto, o presente trabalho objetiva estender o trabalho de [Dantas et al. 2015] e propor uma modelagem multiobjetivo cujas preferências do DM são tratadas como um objetivo a ser maximizado. Inspirado no Método do Ponto de Referência [Wierzbicki 1980], a abordagem inclui uma estratégia para mitigar o esforço em selecionar uma solução da Frente de Pareto através da atribuição de "níveis de aspiração" desejados para cada objetivo.

Por fim, em relação aos trabalhos que avaliam a utilização de preferências humanas em algoritmos multiobjetivos, pode-se destacar a proposta definida em [Li and Deb 2016] o qual propõe uma métrica denominada *R-metric* para averiguar a performance dos algoritmos. A ideia consiste na utilização de um ou mais pontos de referência para realizar um pré-processamento das soluções antes da avaliação de desempenho realizada por outras métricas (Hypervolume, IGD, Spread). São explorados diversos problemas de *benchmark* e cenários artificiais para demonstrar a eficiência da técnica em avaliar a qualidade dos subconjuntos de soluções de acordo com as preferências do DM.

O restante do trabalho é organizado da seguinte forma: na Seção 2 apresenta-se sintetizadamente a proposta que serviu de base para este trabalho; na Seção 3 define-se a modelagem multiobjetivo proposta; na Seção 4 discute-se os resultados atingidos a partir dos experimentos realizados. Finalmente, na Seção 5 destacam-se as conclusões e trabalhos futuros.

2. Planejamento de Releases baseado em Otimização Interativa

Conforme mencionado anteriormente, este trabalho segue fundamentalmente a abordagem definida em [Dantas et al. 2015], excetuando-se o processo de otimização. Visualizando a Figura 1, pode-se constatar que a proposta é composta por três componentes: Gerente de Interações, Base de Preferências e Processo de Otimização.

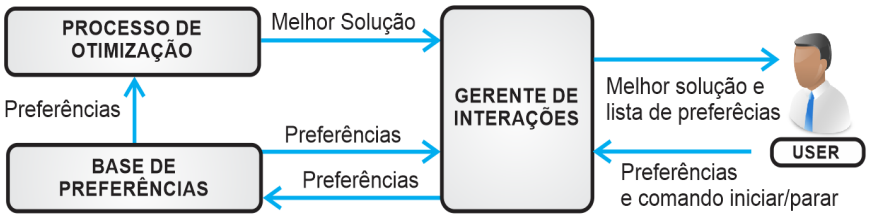


Figura 1. Abordagem proposta em [Dantas et al. 2015]

O Gerente de Interações possui a função de viabilizar as interações do usuário, através dele é possível interagir com a Base de Preferências, adicionando, alterando e/ou removendo preferências, visualizar as melhores soluções e iniciar ou finalizar o processo de busca. A Base de Preferências armazena todas as preferências de forma que se facilite a obtenção de dados relevantes como, por exemplo, a quantidade de preferências presentes na base e que são satisfeitas por uma determinada solução. Finalmente, o Processo de Otimização tem a responsabilidade de prover soluções através de um algoritmo de busca guiado pelas informações contidas na Base de Preferências.

Outro aspecto refere-se aos tipos de preferências que o DM pode expressar em relação à alocação dos requisitos em *releases*. A princípio, no referido trabalho, foram modelados 8 tipos distintos de preferências que também serão utilizados na presente pesquisa: Acoplamento Junto, Acoplamento Separado, Posicionamento Precedente, Posicionamento Sucedente, Posicionamento Antes, Posicionamento Depois, Posicionamento Absoluto e Posicionamento Excludente. Cada preferência é modelada seguindo um modelo genérico de formalização o qual define, por exemplo, os parâmetros necessários.

Experimentos realizados demonstraram que foi possível satisfazer mais de 80% das preferências, porém perdendo 11,7% de $score$ (métrica que pondera a importância e o risco da solução). Este *trade-off* é natural, visto que a avaliação subjetiva do DM não precisa estar atrelada ao $score$ e, inclusive, pode haver conflitos entre os mesmos. Portanto, justifica-se investigar uma formulação multiobjetivo para este cenário em busca de uma perspectiva mais próxima da realidade, visto que a Engenharia de Requisitos é caracterizada pela presença de diversos aspectos conflitantes e complexos, para os quais necessita-se encontrar um balanceamento adequado [Zhang et al. 2007].

3. Modelagem Proposta

Considere o conjunto de requisitos $R = \{r_1, r_2, ..., r_N\}$ disponíveis para serem selecionados para o conjunto de *releases* $K = \{k_1, k_2, ..., k_P\}$, onde N e P são o número total de requisitos e *releases*, respectivamente. Como representação da solução utiliza-se o vetor $S = \{x_1, x_2, ..., x_N\}$ onde $x_i \in \{0, 1, ..., P\}$, de modo que se $x_i = 0$ o requisito r_i não está alocado para alguma *release*, caso contrário, o requisito está alocado para uma *release* k_p, sendo $p = x_i$. Considere um conjunto de clientes $C = \{c_1, c_2, ..., c_M\}$ onde M é o número de clientes e cada cliente c_j possui um respectivo peso para a empresa dado pelo valor w_j. A Função $Value(i)$ retorna a soma ponderada das importâncias que cada cliente c_j especificou para cada requisito r_i, calculada da seguinte forma:

$$Value(i) = \sum_{j=1}^{M} w_j \times \textit{Importance}(c_j, r_i), \tag{1}$$

onde *Importance*(c_j, r_i) retorna o valor de importância atribuído ao requisito r_i pelo cliente c_j. Logo, o valor do objetivo referente a satisfação global dos clientes é dado por:

$$\textit{Satisfaction}(S) = \sum_{i=1}^{N} (P - x_i + 1) \times Value(i) \times y_i, \tag{2}$$

onde $y_i \in \{0, 1\}$ é uma variável de decisão que contém 1 se o requisito r_i está alocado para alguma *release* e 0 caso contrário. Esta estratégia é necessária para evitar que um requisito x_i seja computado quando não está alocado para alguma *release* (x_i = 0). Em razão de $(P - x_i + 1)$, o valor de *Satisfaction(S)* é maior à medida que requisitos com maior $Value(i)$ são alocados para as primeiras *releases*, maximizando assim, a satisfação global dos clientes envolvidos no projeto.

Além de maximizar a satisfação dos clientes, outro aspecto relevante a ser considerado no PR é a minimização do risco envolvido na implementação de cada requisito. Considere o conjunto $D = \{d_1, d_2, ..., d_N\}$ onde cada d_i é o valor de risco associado ao requisito r_i. O valor do objetivo referente ao risco de uma solução é definido por:

$$Risk(S) = \sum_{i=1}^{N} x_i \times d_i, \tag{3}$$

onde o valor de $Risk(S)$ é menor a medida que requisitos com maior risco são alocados nas primeiras *releases*, ou seja, minimizando o risco global do projeto.

Entretanto, o principal diferencial desta modelagem reside na inclusão das preferências do DM como um dos objetivos a serem otimizados. Assim, considere $T = \{t_1, t_2, ..., t_Z\}$ como o conjunto que simboliza a Base de Preferências expressa pelo DM em relação à alocação dos requisitos, onde Z é o total de preferências. Cada preferência t_i é uma tupla constituída por duas definições: a preferência baseada em um dos 8 tipos apresentados anteriormente e um nível de importância que varia de 1 a 10 para a distinguir em termos de relevância. Por exemplo, $t_1 =< posicionamento\_antes(1, 2), 8 >$ denota que o DM deseja que o requisito 1 seja posicionado antes da *release* 2 e que tal preferência apresenta um nível de importância estimado como 8. Para maiores detalhes sobre a formalização das preferências, verificar descrições em [Dantas et al. 2015]. Logo, o valor do objetivo referente às preferências do DM é mensurado por:

$$\textit{Preferences(S,T)} = \begin{cases} \left(\frac{\sum_{i=1}^{Z} L_i \times satisfy(S,t_i)}{\sum_{i=1}^{Z} L_i} \right) & se\ T \neq \emptyset \\ 0, & \text{caso contrário,} \end{cases} \tag{4}$$

onde L_i representa o nível de importância da preferência t_i e $satisfy(S, t_i)$ retorna 1 se a solução S satisfaz a preferência t_i e 0 caso contrário.

Em relação às restrições do problema, primordialmente cada requisito r_i possui um custo $cost_i$ e cada *release* k_p tem um valor de orçamento s_p a ser obedecido. Portanto, é necessário que a soma dos custos de todos os requisitos alocados para cada *release* planejada não exceda o limite orçamentário previamente definido, conforme especificado abaixo pela função $Budget(S)$:

$$Budget(S) = \sum_{i=1}^{N} inRelease(x_i, j) \times cost_i < s_j \ \forall\ j \in \{1, 2, 3, ..., P\}, \tag{5}$$

$$inRelease(x_i, j) = \begin{cases} 1, & se\ x_i = j \\ 0, & \text{caso contrário.} \end{cases} \tag{6}$$

Além disso, outra restrição refere-se a necessidade de que cada *release* tenha uma quantidade de requisitos alocados superior a 0. Essa restrição é tratada pela função $ReqForRelease(S)$:

$$ReqForRelease(S) = \sum_{i=1}^{N} inRelease(x_i, j) > 0\ \forall j \in \{1, 2, 3, ..., P\}. \tag{7}$$

Por fim, a função $Precedence(S)$ utiliza uma matriz binária m_{nxn} para garantir as restrições de precedência e acoplamento:

$$Precedence(S) = x_i \geq x_j, \forall i, j \mid m_{ij} = 1, \tag{8}$$

onde, se $m_{ij} = 1$, o requisito r_i depende do requisito r_j e quando $m_{ij} = m_{ji} = 1$, os requisitos r_i e r_j devem obrigatoriamente ser implementados em uma mesma *release*.

Portanto, a formulação multiobjetivo e interativa proposta para o Planejamento de *Release* consiste em:

$$\begin{array}{ll} \text{maximizar} & \textit{Satisfaction}(S), \\ \text{maximizar} & \textit{Preferences}(S,T), \\ \text{minimizar} & Risk(S), \\ \text{sujeito a:} & 1)\ Budget(S), \\ & 2)\ ReqForRelease(S), \\ & 3)\ Precedence(S). \end{array} \tag{9}$$

3.1. Método do Ponto de Referência

Para um problema modelado com muitos objetivos é possível aplicar diversos algoritmos de busca multiobjetivo, todavia há um desafio em decidir qual solução da Frente de Pareto será selecionada. Solicitar ao DM que escolha uma solução pode ocasionar em um demasiado esforço cognitivo. Visando lidar com este problema, a presente abordagem optou por empregar o Método do Ponto de Referência [Wierzbicki 1980]. Tal método busca, através da especificação dos "níveis de aspiração" desejados para cada objetivo, encontrar a solução que melhor se adéqua aos valores informados pelo DM.

Assim, dado um conjunto de G objetivos, o DM interage na escolha do nível de aspiração a_i para cada objetivo i. Supondo que o DM possua 100 pontos, o mesmo deve distribuir tal quantidade de pontos para cada a_i de acordo com sua importância. Esses valores são utilizados para geração dos pesos μ_i os quais são atribuídos a cada objetivo i, conforme a Equação 10:

$$\mu_i = \frac{1}{\Delta q_i (o_i^{nad} - o_i^*)}, \tag{10}$$

onde $\Delta q_i = a_i/100$. Os vetores $O^{nad} = \{o_1^{nad}, ..., o_G^{nad}\}$ e $O^* = \{o_1^*, ..., o_G^*\}$ são formados respectivamente, pelos maiores e menores valores alcançados pela Frente de Pareto avaliada para cada objetivo. Os pesos μ_i são usados para normalizar os valores utilizados na função $MaxValue(S)$ de acordo com os diferentes intervalos dos objetivos além de pondera-los em relação aos níveis de aspiração inseridos. A função $MaxValue(S)$ é definida como:

$$MaxValue(S) = \max_{i=1,...,G} \{\mu_i (f_i(S) - o_i^*)\}, \tag{11}$$

onde $f_i(S)$ representa a função objetivo i do problema. Portanto, verifica-se que $MaxValue(S)$ busca encontra dentre os objetivos, a maior diferença entre o valor da solução para o objetivo i e o respectivo melhor valor alcançado pela Frente de Pareto naquele objetivo. Logo, o valor de $MaxValue(S)$ representa para a solução a distância do objetivo menos atendido pela solução S para o melhor valor alcançado.

Exemplificando o método apresentado acima: considere um cenário onde o DM possui 100 pontos a serem distribuídos para diferenciar seus níveis de aspiração entre os três objetivos. O mesmo optou por uma solução que atenda aproximadamente os três objetivos ($Satisfaction$, $Risk$ e $Preferences$), ou seja, atribuiu 34 pontos ao nível de aspiração p_1 e 33 pontos aos demais (p_2 e p_3). Com base nestes dados, a Equação 10 gera os pesos μ_i que serão atribuídos a cada objetivo i. Após o algoritmo multiobjetivo gerar

uma Frente de Pareto com E soluções, o método cria um vetor de E posições, onde cada posição representa uma solução da Frente E associado com um $MaxValue$ determinado pela Equação 11. Consequentemente, a solução que apresentar o menor $MaxValue$, será considerada a que melhor atende os níveis de aspiração (a_i) expressos inicialmente pelo DM. Finalmente, a Equação 12, também reconhecida como *scalarizing function*, representa o processo de busca da solução na Frente de Pareto E, que atenda aos níveis de aspiração do DM:

$$\begin{aligned} &\text{minimize} \quad MaxValue(S), \\ &\text{sujeito a:} \quad S \in E. \end{aligned} \tag{12}$$

4. Estudo Empírico

Visando avaliar a modelagem proposta, conduziu-se um estudo empírico com diferentes algoritmos de busca. Para tanto, foram utilizadas duas instâncias baseadas em dados de projetos reais obtidos em [Karim and Ruhe 2014]. O dataset-1, que corresponde a um processador de texto, possui 50 requisitos, 4 clientes e 5 *releases*. O dataset-2, uma ferramenta de suporte à decisão para o planejamento de *releases*, possui 25 requisitos, 9 clientes e 8 *releases*. Para ambas as instâncias foi considerado um conjunto de preferências geradas aleatoriamente.

Buscando variação no objetivo que visa maximizar a quantidade de preferências, foram avaliados dois cenários para cada instância. No Cenário 1, 10 e 5 preferências aleatórias foram adicionadas à Base de Preferências para o dataset-1 e dataset-2, respectivamente. No Cenário 2, as quantidades de preferências foram 50 e 25, ou seja, a mesma quantidade de requisitos presente das instâncias correspondentes.

Em relação às técnicas de otimização, foram comparados dois algoritmos evolucionários (NSGA-II e MOCell), e um algoritmo aleatório para teste de sanidade conforme sugerido em [Harman et al. 2012]. Para cada instância e considerando os dois cenários previamente apresentados, as técnicas evolucionárias foram executadas 30 vezes com configuração de 256 indivíduos, 400 gerações, taxa de cruzamento de 90% e mutação de 1%. No caso do algoritmo aleatório foram geradas, aleatoriamente, 102.400 soluções a partir das quais foi gerada uma Frente de Pareto. Todos os parâmetros foram obtidos empiricamente através de testes preliminares.

4.1. Performance dos Algoritmos

A Tabela 1 apresenta o teste estatístico realizado para as métricas Hypervolume (HV), Spread (SP) e Generational Distance (GD) que foram coletadas durante cada execução dos algoritmos. O teste de Wilcoxon (WC) foi aplicado, utilizando a correção de Bonferroni, visando identificar a ocorrência de diferença estatística entre as amostras considerando um nível de confiança de 95%. O teste de Vargha-Delaney $\hat{A}_{12}$ foi empregado para verificar o tamanho de efeito, ou seja, o número de vezes que um algoritmo produziu resultados superiores em relação ao outro.

Observando os dados da Tabela 1, percebe-se que houve diferença estatística na maioria das comparações. Havendo apenas um caso, no Cenário 1, onde o valor *p-value* foi acima de 0,01 na métrica SP entre o algoritmo aleatório e o MOCell para o dataset-1. Tal constatação sugere segurança para se conduzir observações comparativas entre os resultados obtidos.

Tabela 1. Resultado para os testes estatísticos de Wilcoxon e Vargha-Delaney $\hat{A}_{12}$. A comparação ocorre entre o algoritmo da linha pelo da coluna

	Algoritmos	Métricas	dataset-1				dataset-2			
			MOCell		NSGA-II		MOCell		NSGA-II	
			WC	$\hat{A}_{12}$	WC	$\hat{A}_{12}$	WC	$\hat{A}_{12}$	WC	$\hat{A}_{12}$
Cenário 1	NSGA-II	HV	8.00E-09	0.95	-	-	9.10E-11	1.00	-	-
		SP	1.40E-06	0.88	-	-	2.20E-10	0.99	-	-
		GD	5.20E-07	0.11	-	-	3.60E-10	0.02	-	-
	Aleatório	HV	9.10E-11	0.00	9.10E-11	0.00	9.10E-11	0.00	9.10E-11	0.00
		SP	**3.50E-01**	0.38	4.30E-08	0.07	4.70E-08	0.07	9.10E-11	0.00
		GD	9.10E-11	1.00	9.10E-11	1.00	9.10E-11	1.00	9.10E-11	1.00
Cenário 2	NSGA-II	HV	1.30E-09	0.97	-	-	2.20E-10	0.99	-	-
		SP	2.90E-03	0.75	-	-	9.10E-11	1.00	-	-
		GD	5.30E-10	0.02	-	-	1.50E-10	0.01	-	-
	Aleatório	HV	9.10E-11	0.00	9.10E-11	0.00	9.10E-11	0.00	9.10E-11	0.00
		SP	6.85E-01	0.41	9.10E-03	0.28	1.40E-02	0.29	2.20E-10	0.01
		GD	9.10E-11	1.00	9.10E-11	1.00	9.10E-11	1.00	9.10E-11	1.00

O algoritmo aleatório obteve os piores resultados em HV e GD em todas as comparações, tendo resultados um pouco melhores apenas em SP mas, ainda assim, sendo o pior quando comparado aos demais algoritmos. O NSGA-II obteve melhores resultados que o MOCell na métrica GD, garantindo, no Cenário 1 por exemplo, que em apenas 11% das vezes para o dataset-1 e 2% para o dataset-2, uma solução gerada pelo NSGA-II teve GD maior que uma gerada pelo MOCell. Por outro lado, na métrica SP o resultado foi inverso, o NGSA-II consegue em 88% das vezes para o dataset-1 e 99% para o dataset-2, soluções com SP maiores que o MOCell. No Cenário 2 houve um comportamento semelhante, tendo o NSGA-II atingido GD maior em apenas 2% das vezes para o dataset-1 e 1% para o dataset-2, porém com SP maior em 75% das vezes para o dataset-1 e 100% para o dataset-2.

Os testes estatísticos mostram que o NSGA-II possui melhor convergência visto que alcançou menores resultados em GD, o qual representa a distância entre a Frente de Pareto conhecida e a Frente de referência, enquanto o MOCell garante uma melhor distribuição das soluções pois apresenta menor SP, o qual representa a diversidade de soluções na Frente de Pareto.

Como métrica bastante difundida na literatura, o HV mede tanto a convergência como a distribuição das soluções dentro da Frente de Pareto. Por esse motivo, tal métrica pode ser considerada essencial na avaliação dos algoritmos. No Cenário 1, NSGA-II obteve resultados melhores do que o MOCell em 95% das vezes para o dataset-1 e 100% para o dataset-2. No Cenário 2, o NSGA-II foi superior em 97% das vezes para o dataset-1 e 99% para o dataset-2. Logo, em termos de HV, o NSGA-II apresentou o melhor *trade-off* entre distribuição e convergência.

4.2. Análise do Método do Ponto de Referência

A partir da constatação proveniente das análises da seção anterior, o NSGA-II foi utilizado para ilustrar o comportamento do método de escolha de uma solução da Frente de Pareto empregado nesta abordagem. Para isso, foram simuladas 4 diferentes níveis de aspiração que poderiam ser informadas pelo DM durante o processo de resolução. Em todos os casos, utilizou-se o dataset-1 e as 50 preferências com relação a alocação dos requisitos foram inseridas *a priori* na Base de Preferências.

A Figura 2 (A) apresenta uma Frente de Pareto expressa em um espaço com três dimensões, que correspondem a cada um dos objetivos do problema. Além disso, des-

taca qual solução seria apresentada para o usuário caso ele tivesse definido os níveis de aspiração 1, 98 e 1 para cada objetivo respectivamente. Conforme pode-se verificar, a solução escolhida (indicada pela seta vermelha), encontra-se em uma região com baixos valores de $Risk$, o que é preferível, pois pretende-se minimizá-lo. Por outro lado, a mesma solução não apresenta os melhores valores em $Satisfaction$ e $Preferences$.

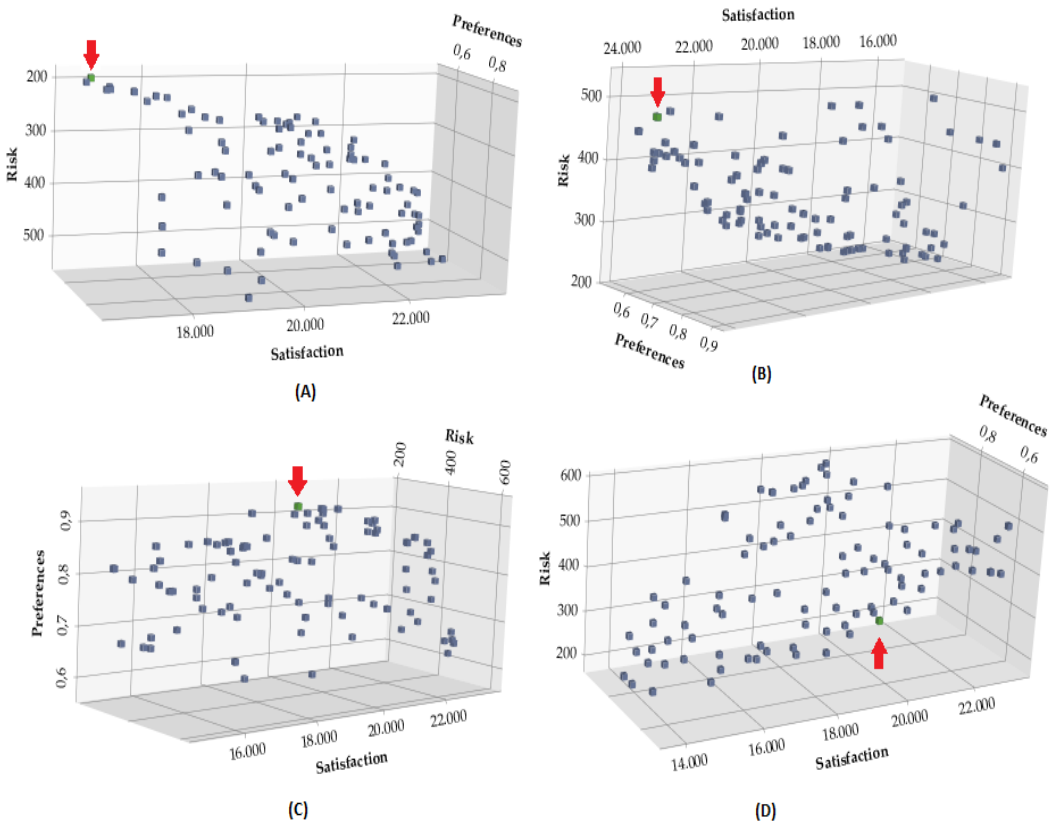


Figura 2. Frentes de Pareto obtidos pelo NSGA-II para o Dataset-1 com 50 preferências

Na Figura 2 (B) o objetivo priorizado foi $Satisfaction$. Assim, a solução escolhida posiciona-se em uma região extrema dos maiores valores dessa dimensão. Contudo, o valor de $Risk$ aumentou enquanto o de $Preferences$ apresentou pouca variação. Notadamente, uma configuração de pesos que privilegie o objetivo $Preferences$ em detrimento aos outros objetivos tenderá a escolher uma solução posicionada de formar similar ao que se vê na Figura 2 (C). Assim como uma configuração equivalente de pesos apresentará uma solução que equilibre o *trade-off* entre os valores das três dimensões.

Por fim, considera-se válido salientar que, em um contexto de utilização prática da proposta, o DM pode interagir quantas vezes desejar tanto para alterar os pesos dos objetivos e, consequentemente influenciar na escolha da solução da frente de Pareto, quanto modificar suas preferências presentes na Base de Preferências. Logo, é plausível considerar que tal proposta de visualização de soluções pode colaborar diretamente numa captura mais eficiente dos aspectos subjetivos humanos.

5. Conclusões

A utilização de SBSE demonstra-se bastante proeminente na resolução do Planejamento de *Releases*. Objetivando agregar os benefícios da Otimização Interativa à SBSE e mo-

tivado pelo *trade-off* entre as preferências humanas e as métricas próprias do problema, o presente trabalho apresenta uma abordagem multiobjetivo onde a expertise do tomador de decisão é modelada como um dos objetivos a ser maximizado. Através dos experimentos realizados, verificou-se que, de maneira geral, o NSGA-II superou o MOCell e o algoritmo aleatório. Além disso, constatou-se que o Método do Ponto de Referência mostra-se factível como auxilio na escolha de uma solução da Frente de Pareto.

Em termos de trabalhos futuros, pretende-se realizar um estudo empírico envolvendo a presença de usuários com experiência em Engenharia de Requisitos e investigar formas mais intuitivas de visualização e interação com as soluções no espaço de busca.

Referências

Araújo, A. A., Paixao, M., Yeltsin, I., Dantas, A., and Souza, J. (2016). An architecture based on interactive optimization and machine learning applied to the next release problem. *Automated Software Engineering*, pages 1–49.

Dantas, A., Yeltsin, I., Araújo, A. A., and Souza, J. (2015). Interactive software release planning with preferences base. In *SSBSE'15*, pages 341–346. Springer.

Ferreira, T. d. N., Araújo, A. A., Neto Basílio, A. D., and Souza, J. T. d. (2016). Incorporating user preferences in ant colony optimization for the next release problem. *Applied Soft Computing*.

Harman, M., McMinn, P., de Souza, J. T., and Yoo, S. (2012). Search based software engineering: Techniques, taxonomy, tutorial. *Empirical Software Engineering and Verification*, 7007:1–59.

Karim, M. R. and Ruhe, G. (2014). Bi-objective genetic search for release planning in support of themes. In *SSBSE'14*, pages 123–137. Springer.

Li, K. and Deb, K. (2016). Performance assessment for preference-based evolutionary multi-objective optimization using reference points.

Miettinen, K. (2012). *Nonlinear multiobjective optimization*, volume 12. Springer Science & Business Media.

Ngo-The, A. and Ruhe, G. (2008). A systematic approach for solving the wicked problem of software release planning. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, 12(1):95–108.

Ruhe, G. and Saliu, M. O. (2005). The art and science of software release planning. *Software, IEEE*, 22(6):47–53.

Takagi, H. (2001). Interactive evolutionary computation: Fusion of the capabilities of ec optimization and human evaluation. *Proceedings of the IEEE*, 89(9):1275–1296.

Tonella, P., Susi, A., and Palma, F. (2013). Interactive requirements prioritization using a genetic algorithm. *Information and Software Technology*, 55(1):173–187.

Wierzbicki, A. P. (1980). The use of reference objectives in multiobjective optimization. In *Multiple criteria decision making theory and application*, pages 468–486. Springer.

Zhang, Y., Harman, M., and Mansouri, S. A. (2007). The multi-objective next release problem. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 1129–1137. ACM.

Towards the Extension of an Evaluation Model for Product Line Architecture Design

Yenisei D. Verdecia, Thelma E. Colanzi

[1]Department of Informatics – State University of Maringa (UEM)
Caixa Postal 15.064 – 91.501-970 – Maringa – PR – Brazil

yni6.slim@gmail.com, thelma@din.uem.br

Abstract. *Product Line Architecture (PLA) is the most important artifact for Software Product Lines (SPL) Engineering. Studies about PLA design evaluation often consider a set of metrics that provide indicators about different factors such as design stability, feature modularization and PLA extensibility. A systematic and automated approach that uses search-based algorithms to optimize a PLA design, named MOA4PLA (Multi-Objective Approach for Product -Line Architecture Design), was recently proposed. MOA4PLA aims at optimizing basic design principles, feature modularization, design elegance and SPL extensibility. Currently, the evaluation model of MOA4PLA has one objective function for each one of those optimization goals. However, there are other design properties important to PLA design, what motivates a study for extending the MOA4PLA evaluation model. In this sense, the objective of this paper is to enrich the evaluation model with metrics related to other properties. Our study was guided by the Goal-Question-Metrics approach followed by a Proof of Concept performed with four PLA designs. Preliminary results provide evidences that the identified metrics can contribute to the evaluation model extension.*

1. Introduction

Software Product Line (SPL) approach is based on the systematic reuse of software artifacts, through the holding of common features and management of variability between products, which are established under a same architecture [Pohl et al. 2005]. A Product Line Architecture (PLA) is the main artifact of an SPL and provides the design that is common to all the products of the SPL. Hence it describes all the mandatory and varying features of its SPL domain [Contieri et al. 2011].

PLA design is an important activity throughout the life cycle of the SPL [Oizumi et al. 2012]. It is even more complex than traditional software architecting because, in order to maximize the reuse, some quality requirements, such as modularity, stability and reusability, represent an important role [Colanzi and Vergilio 2016]. The more extensible are the architecture components, the greater is their reusability rate [OliveiraJr et al. 2013]. Therefore, it is important to analyze the architecture to predict their quality before the products generation, in order to identify potential risks and verify that the quality requirements are being treated in the design [Pohl et al. 2005].

Studies on structural evaluation of PLA often consider a set of metrics to provide indicators on different properties [Oizumi et al. 2012]. Find the best balance of priorities (trade-off) between the metrics as well as choose which should be prioritized is a difficult task that needs great effort and time. In this point of view, the Search Based

Software Engineering (SBSE) [Harman et al. 2012], provides techniques to obtain and evaluate possible alternatives for a PLA design. As the problems of Software Engineering can be formulated as optimization problems to be solved through search-based techniques [Colanzi et al. 2014]. Approaches based on Multi-Objective Evolutionary Algorithms (MOEAs) have been applied to optimize the software design or optimize the order of refactoring to be applied to the design [Räihä 2010], whereas they allow finding a better solution in a set of possible solutions. In this view, the multi-objective optimization approach called Multi-Objective Approach for Product-Line Architecture Design (MOA4PLA) [Colanzi et al. 2014] was recently proposed.

The goal of MOA4PLA is to identify automatically the best alternatives for a PLA design, optimizing basic design principles, feature modularization, design elegance and SPL extensibility. Currently, the maximum number of goals of MOA4PLA is limited to four objective functions available, one for each MOA4PLA goal. However, there are other design properties relevant to PLA design, which were not included in the current evaluation model, such as PLA complexity, adaptability and similarity levels of a PLA design. This fact motivates a study for extending the MOA4PLA evaluation model.

In this sense, the objective of this paper is to enrich the evaluation model with metrics related to other properties. To this end, we proceeded the following methodology: 1) the Goal-Question-Metrics (GQM) approach was applied in the definition phase of the study to identify whether the complexity of developed components as well as the level of coupling, the level of similarity and adaptability of a PLA, are goals that can be measured in the evaluation model of MOA4PLA, and 2) a Proof of Concept (PoC) was carried out to explore the behavior of the metrics identified in the GQM. A set of preliminary results provide evidences that the identified metrics can contribute to the evaluation model extension and provide some perspectives to be considered. So, the main contribution of the work is the identification and selection of metric to enrich the MOA4PLA evaluation model.

This paper is organized as follows. Section 2 briefly describes MOA4PLA and its evaluation model. Section 3 presents the GQM defined in order to find new goals to be evaluated in the approach. Section 4 explores the behavior of identifying metrics, presenting the results of measurement as well as the perspectives identified. Finally, Section 5 concludes the paper and directs further work.

2. Characterizing MOA4PLA

MOA4PLA is a multi-objective optimization approach proposed to identify automatically the best alternatives of PLA design, regardless of the search algorithm adopted. It contributes to the decrease the effort of architects during the design and evaluation of a PLA [Colanzi et al. 2014]. MOA4PLA encompasses four main activities: (1) Construction of the PLA Representation, (2) Definition of the Evaluation Model, (3) Multi-objective Optimization and (4) Transformation and Selection. The approach receives as input the PLA design, modeled in a UML class diagram, where each architectural element is associated with the feature(s) that it realizes. The output is the set of PLA alternative designs that have the best trade-off among the defined objectives to be optimized.

The Multi-objective Optimization activity of MOA4PLA includes conventional mutation operators: MoveMethod, MoveAttribute, AddClass, MoveOperation and Ad-

dComponent [Colanzi et al. 2014]. AddClass moves a method or an attribute to a new class. MoveOperation moves operations between interfaces. AddComponent creates a new interface to a new component, and then moves an operation to this interface. That activity also encompasses mutation and crossover operators to improve feature modularization: Feature-Driven Mutation [Colanzi et al. 2014] and Feature-Driven Crossover [Colanzi and Vergilio 2016]. The first one aims at modularizing a feature tangled with others in a component, considering that a feature usually is solved by a group of architectural elements [Colanzi et al. 2014]. The second one selects a feature at random and then creates children by swapping the architectural elements (classes, interfaces, operations, etc.) that realize the selected feature. In this way, children might combine groups of architectural elements that better modularize some feature(s) inherited from their parents [Colanzi and Vergilio 2016].

The evaluation model of MOA4PLA is composed of four objective functions, related to the main goals of the approach, to evaluate every solution generated in the optimization process. The objective functions include metrics to measure basic design principles, features modularization, SPL extensibility and design elegance [Colanzi et al. 2014]. The objective functions are briefly described below:

CM(pla)[1]: Aims at providing indicators on basic design principles including cohesion, coupling and size. This objective function is composed of an aggregation of various metrics extracted from [Wüst 2013].

FM(pla)[1]: Aims at measuring the features modularization. It aggregates several feature-driven metrics proposed in [Sant'Anna 2008] to measure feature-based cohesion, feature diffusion and feature interaction over architectural elements.

Ext(pla): Aims at indicating the degree of extensibility of SPL, where the extensibility is measured by means of PLA abstraction [OliveiraJr et al. 2013].

Eleg(pla): Aims at providing indicators about the elegance of a object-oriented software design. It is measured by the sum of the metrics defined in [Simons and Parmee 2012].

3. Identifying metrics to expand the evaluation model

Given the present work motivation, we define our evaluation goal using the Goal-Question-Metric (GQM) approach [Basili and Rombach 1988]. Based on the GQM template [Basili and Rombach 1988], the **goal** is presented as follows:

Analyze *PLAs*
For the purpose to *expand the evaluation model of MOA4PLA*
With respect to *architectural design properties*
From the point of view of *software architects*
In the context of *SPL.*

The *questions* and *metrics* that must be answered and interpreted are:

Q1: *How is the complexity of components designed in a PLA?* The architecture has a direct impact on the software complexity [Pohl et al. 2005]. The more dependencies

[1]Currently on review because added metrics with different grandeurs [Santos et al. 2015].

between the components that make up the PLA, the greater their complexity, because a high number of dependencies makes any changes cause major impacts on the related components. In this regard the following metric helps answer the question, in spite of it was originally applied in the context of Service-Oriented SPL:

M1: Weighted Operations per Component or Service (WOCS) [Ribeiro et al. 2010]. WOCS will measure the complexity of the service or component, in terms of its operations (methods) that will be required by other services or components. Hence, consider a Component or Service C with operations O1, ... , On. Let c1, ... , cn be the complexity of the operations. Then: WOCS = c1 +...+ cn. The metric indicates that operations with many formal parameters are more likely to be complex than operations in which require less parameter. In this sense, the complexity of an operation ck can be defined as follows: $ck = \alpha + 1$ where αk denotes the number of formal parameters of operations (Ok) [Ribeiro et al. 2010].

Q2: *The components contained in the PLA has low coupling?* One of the expectations of software architects is to know the level of coupling between the components inside of a PLA, because if that component have least possible dependencies, it will be easier to develop, test, and maintain. The following metric was also applied in the context of Service-Oriented SPL, but it is interesting for SPL in general and satisfies the question:

M2: Coupling Between Components or Services (CBCS) [Ribeiro et al. 2010]. This metric is calculated based on the number of relationships between one service A, for example, and other services in an application. It can be mathematically described in Equation 1 as follows:

$$CBCS = \sum_{i \neq j = 1 K_n}^{n} AiBj \tag{1}$$

In which: n is the number of services in application; $AiBj$ = 0 if Ai does not connect to Bj and $AiBj$ = 1 if Ai connects to Bj [Ribeiro et al. 2010].

For a service A, the larger the value of CBCS metric, the tighter the relationship with other services is. In other words, service A depends much more on others [Quynh 2009]. CBCS goal can be focused in the context of Product Line, based on the number of relationships between interfaces.

Q3: *What is the level of similarity existing in the PLAs?* The identification of common components and variables of a PLA, gives insight into the level of reuse that it can exist in a PLA. To know the level of similarity existing in the PLAs the following metric answers that question:

M3: Structure Similarity Coefficient (SSC) method [Zhang et al. 2008], is proposed to measure similarity of PLA as Equation 2 following:

$$SSC = \frac{|Cc|}{|Cc| + |Cv|} \tag{2}$$

In Equation 2, Cc is number of common components in PLA and Cv is number of variable components in PLA. According to Equation 2, PLA has more common components and less variable components, leads to architectures of products that are more similar. Then the SPL will gain more benefits of reuse [Zhang et al. 2008].

Q4: *What is the level of adaptability of a PLA?* The variability is constantly evolving, with the extension of scope and currency exchange in applications [Peng et al. 2011]. The greatest variability, more possibilities to adapt the product to customer's tastes. In this regard the following metrics respond to the question:

M4: Strong Coupling Coefficient (SCC) method [Zhang et al. 2008], is proposed to measure strong coupling of variability as follows in Equation 3:

$$SCC = 1 - \frac{|IVP|}{|VP|} \tag{3}$$

In Equation 3 , VP is the variability point number and IVP is the number of independent variability points of PLA, which have no dependence relations with others. PLA has more variability points, more number of product architectures can be derived by instantiating the PLA. However, PLA is more complex and difficult to be designed [Zhang et al. 2008].

M5: Structure Variability Coefficient (SVC) method [Zhang et al. 2008] is defined to measure structure variability of PLA in Equation 4. Cc is number of common components in PLA and Cv is number of variable components in PLA . PLA has more variable components, and there is more structure variability. SVC also can be used to measure structure variability of compound components [Zhang et al. 2008].

$$SVC = \frac{|Cv|}{|Cc| + |Cv|} \tag{4}$$

M6: Architecture variability (AV) [Zhang et al. 2008], is total variability of PLA. AV is defined as following Equation 5:

$$AV = |Cv| + \sum_{i} AV(Ci) \tag{5}$$

Where: $|Cv|$: The number of variable components in PLA, $AV(Ci)$: Variability of interior component Ci . If Ci is compound component, then $AV(Ci)$ can be calculated with Equation 5. If C is basic component and has interior variability, $AV(Ci)$=1, else $AV(Ci)$=0 [Zhang et al. 2008].

To find the metrics, that respond the questions defined in GQM, we used the following search engines for our reviews: ACM Digital Library, IEEE Xplore, Elsevier Scopus, and SpringerLink. Our specific research question was: *What is the state of art the metrics to Software Product Line?*. The selection of metrics is based on the following criteria: (1) it is possible to obtain the information needed to apply the metrics of a UML class diagram, because the evaluation model refers to a representation of the PLA which is based on a metamodel that instantiates a UML class diagram, and (2) the metrics were not implemented in the current evaluation model. In general, the identified metrics in GQM, except CBCS, measure architecture design properties that not evaluated in evaluation model. Even so, CBCS measures different coupling type that the coupling measures currently applied in MOA4PLA.

4. Validating the identified metrics for the context of MOA4PLA

A Proof of Concept (PoC) is a demonstration aimed at verifying that certain concepts or theories has potential of being used. A prototype is designed to determine feasibility, but does not represent the final deliverable, because PoC studies seek to determine whether

or not a product idea will be viable [Gallant 2006]. For the present work, the objective of the PoC is twofold. Firstly, validate the application of metrics in the context of PLA design, aiming at its use in the evaluation model of MOA4PLA. Secondly, identify new perspectives to be analyzed in approach. For this end, four PLA designs were used in the PoC, whose main information are presented in Table 1. The metrics identified in the GQM were calculated for each PLA design[2].

PLAs selected are based on components and follow the layered style. They are briefly described below. Arcade Game Maker (AGM) is a SPL created by the Software Engineering Institute (SEI) that encompasses three arcade games: Brickles, Bowling, and Pong [SEI 2016]. System Banking (Bank) supports the managing of banking systems [Gomaa 2011]. BET supports the bus city transport management. It offers features such as the use of an electronic card for transport payment; automatic toll gate opening; and unified traveling payment. It was conceived based on three existing transportation products of Brazilian cities [Donegan and Masiero 2007]. Mobile Media (MM) is a SPL composed of features that handle music, videos, and photo for portable devices. It provides support for managing different types of media [Contieri et al. 2011].

Table 1. PLA Information

ALP	# Components	# Interfaces	# Classes	# Features	#Variabilities
AGM	9	14	30	11	5
Bank	4	5	25	16	3
BET	56	30	115	18	8
MM	8	15	14	14	7

Following section address the behavior analysis of the obtained results from the metrics application and the perspectives identified during the analysis.

4.1. Behavior Analysis of the Results

This section presents the behavior analysis of each metric centering on the objectives of the metrics identified in GQM (Section 3).

M1: The value of WOCS [Ribeiro et al. 2010] was calculated for each package of the PLAs selected. Due to the number of interfaces in each package of PLAs, the results obtained from WOCS were extensive. Even so, was analyzed each results of WOCS. To illustrate the behavior of WOCS, one package of the PLAs AGM and BET with the same amount of interfaces was selected, as can be observed in Table 2.

Table 2. Results of WOCS in GameBoardCtrl and LinhaMgr packages

PLAs					
AGM			**BET**		
Package	Interface	WOCS	Package	Interface	WOCS
GameBoardCtrl	ICheckScore	9	LinhaMgr	IAtualizarCorrida	1
	IGameBoardData	2		IRegistrarArrecadacao	2
	ISaveScore	19		ILinhaMgt	23

As can be seen in the Table 2, the value of WOCS for each interface of GameBoardCtrl package of AGM, the values of complexity of interfaces are not high. The

[2]The entire results of the measurement as well as the PLA designs can be obtained in https://github.com/yni6delgado/Test

same behavior occurred for the LinhaMgr package of BET. But through this analysis, we observe that had classes with a high number of operations in the PLAs. In this sense, it is interesting to analyze the number of operations by classes, with the objective of reducing them, therefore emerged two new views defined in Section 4.2.

M2: CBCS metric [Ribeiro et al. 2010] can be focused to the context of Product Line in general by analyzing the number of relationships between an interface A and other interfaces of the PLA. Taking into account this focus, CBCS was calculated for each PLA. According to [Ribeiro et al. 2010], a CBCS value greater than 9 is considered a too high value, in our context we follow the same criteria. Overall the results show a view of the behavior of metric, because they allow to see the level of coupling among the interfaces of the PLAs. Values greater than 9 where achieved only for BET in the following interfaces: IProcessarViagem (10), IRegistrarArrecadacao (10), ILinhaMgt (12), IPassageiroMgt (12), ICartaoMgt (27), IViacaoMgt (11) and ITempoMgt (10).

M3: SSC [Zhang et al. 2008] was calculated for all PLAs. The values obtained of SSC range from 0 to 1, are shown in Table 3. It is possible to notice that the number of common components in PLAs is greater than the number of variable components, what is good for maximizing the PLA reusability.

M4, M5, M6: For the variability analysis, were selected the following metrics: SCC, SVC and AV [Zhang et al. 2008]. Each of the metrics allows analyzing different views: 1) coupling between points of variability (SCC), 2) structure variability (SVC), and 3) architecture variability (AV).

SCC metric allows knowing if the points of variability in PLAs have dependency relationships with other points of variability, and in this way can be analyzed the level of coupling between them. However, the value of SCC for all PLAs is 1 because there is no dependency relationship between points of variability in those designs (see Table 3). Furthermore, this metric cannot be calculated actually in MOA4PLA because the metamodel defined for the approach does not provide this level of relationship. In this sense, to apply such metric in the evaluation model, it is necessary to adapt the metamodel used in MOA4PLA.

The values of SVC range from 0 to 1, the results of SVC shown in Table 3, indicate that the number of variable components inside of PLAs is down.

AV metric is set to understand the architecture variability, through the identification of variability within compound and basic components. Analyzing the behavior of AV in the PLAs, taking into account Equation 5, it is noticed that, there are not compound components inside the selected PLAs. Table 3 shows the results of AV.

In general, we conclude that the values of metrics identified in GQM are consistent with the PLA design properties which they are supposed to measure. With regards to the feasibility of the metrics application in the evaluation model of MOA4PLA, we also analyzed if the metrics values could be influenced by the search operators of MOA4PLA during the optimization process. Table 4 shows what mutation operators can influence on the metrics values during the optimization of PLA design solutions. Thus, that is other evidence of the feasibility application of the identified metric on the evaluation model, as every metric value can be changed during the optimization process.

Table 3. Results of metrics

PLAs	#Cc	#Cv	Metrics			
			SSC	SCC	SVC	AV
AGM	7	2	0.77	1	0.22	4
Bank	3	1	0.75	1	0.25	2
BET	53	3	0.95	1	0.54	6
MobileMedia	7	1	0.88	1	0.13	2

Table 4. Relationship metric X mutation operator

Metrics	Search Operator
WOCS	Move Operation Mutation
CBCS	All operators
SSC	Feature-driven Mutation and Add Package
SCC	Feature-driven Mutation
SVC	Feature-driven Mutation and Add Package
AV	Feature-driven Mutation

4.2. Perspectives

Looking at the preliminary results and considering the PLA design optimization, we can highlight the following perspectives when using the identified metrics to extend the MOA4PLA evaluation model:

P_1: The intention will be to decrease the values of CBCS in search to soften the level of coupling between interfaces. In this sense the results of CBCS to BET will be a point of observation to be taken into account when the metric is implemented in MOA4PLA.

P_2: Analyzing the behavior of WOCS, could be seen that had classes with a high number of operations in the PLAs. In this sense classes with a high number of operations are to be analyzed from the following viewpoints [Chidamber and Kemerer 1994]: 1) the larger numbers of methods in a class, the greater the potential impact on children, since children will inherit all the methods defined in the class and 2) classes with large numbers of methods limit the possibility of reuse. Therefore the intention will be to decrease the values WOCS from the point of view of interfaces and class.

P_3: A SSC value close to 1 represents a greater number of common components within the PLA. In this sense the intention will be to maximize the SSC value taking into account the existing variability in PLA.

P_4: In the context of SCC, decreasing the level of coupling between the existing variability points a PLA, provides greater flexibility in time for the instantiating PLA. Accordingly, increasing the number of IVP from a PLA promotes the level of reusability of the PLA.

P_5: A SVC value close to 1, demonstrates the presence of a greater number of variable components in the PLAs. This metric can be applied to composite components of the PLA, with the intention of increasing the number of variable components within the PLA.

P_6: The greater variability within the compound component, the PLA is more complex and difficult to be designed. In this sense to decrease the number of composite components increases the level of reusability of the PLA.

P_7: Analysis results of SVC and SSC point out that they are conflicting metrics.

5. Conclusions and future works

In this paper we presented a study of metrics guided by GQM and based on PoC to identify and validate metrics to extend the evaluation model of MOA4PLA. Through the GQM were identified metrics to provide indicators about the complexity and coupling among components of the PLA design as well as the level of similarity and adaptability of the PLA to be included as new goals to be optimized by the approach.

The PoC points out the metrics values are consistent with the PLA design properties under measurement and that the values can be suffer the action of the search operators during the optimization process, leading to an optimization of the informed PLA design.

As future works we intend: 1) to propose new objective functions for the evaluation model, 2) to perform experiments to investigate the possible correlation between the proposed objective functions and between them and the objective functions existing in the current evaluation model and 3) perform empirical studies to optimize PLA designs according to the new goals of MOA4PLA.

6. Acknowledgment

We would like to thank CNPq for financial support.

References

Basili, V. R. and Rombach, H. (1988). The tame project: Towards improvement-oriented software environments. *IEEE Transactions on Software Engineering*, 14(6):758–773.

Chidamber, S. R. and Kemerer, C. F. (1994). A metrics suite for object oriented design. *IEEE Transactions on Software Engineering*, 20(6):476–493.

Colanzi, T. E. and Vergilio, S. R. (2016). A feature-driven crossover operator for multi-objective and evolutionary optimization of product line architectures. *Journal of Systems and Software*. In press.

Colanzi, T. E., Vergilio, S. R., Gimenes, I. M. S., and Oizumi, W. N. (2014). A search-based approach for software product line design. In *Proceedings of the 18th International Software Product Line Conference (SPLC 2014)*, volume 1, pages 237–241.

Contieri, Jr., A. C., Correia, G. G., Colanzi, T. E., Gimenes, I. M. S., OliveiraJr., E. A., Ferrari, S., Masiero, P. C., and Garcia, A. F. (2011). Extending uml components to develop software product-line architectures: Lessons learned. In *Proceedings of the 5th European Conference on Software Architecture*, ECSA'11, pages 130–138, Berlin, Heidelberg. Springer-Verlag.

Donegan, P. M. and Masiero, P. C. (2007). Design issues in a component-based software product line. In *Proc. of Brazilian Symposium on Software Components, Architectures and Reuse (SBCARS)*, pages 3–16.

Gallant, M. (2006). An ethnography of communication approach to mobile product testing. *Personal Ubiquitous Comput.*, 10(5):325–332.

Gomaa, H. (2011). *Software modeling and design: UML, use cases, patterns, and software architectures*. Cambridge University Press.

Harman, M., Mansouri, S. A., and Zhang, Y. (2012). Search-based software engineering: Trends, techniques and applications. *ACM Computing Surveys*, 45(1):11:1–11:61.

Oizumi, W., Contieri, A., Correia, G., Colanzi, T., Ferrari, S., Gimenes, I., OliveiraJr, E., Garcia, A., and Masiero, P. (2012). On the proactive design of product-line architectures with aspects: An exploratory study. In *2012 IEEE 36th Annual Computer Software and Applications Conference (COMPSAC)*, pages 273–278.

OliveiraJr, E. A., Gimenes, I. M. S., Maldonado, J. C., Masiero, P. C., and Barroca, L. (2013). Systematic evaluation of software product line architectures. *Journal of Universal Computer Science*, 19(1):25–52.

Peng, X., Yu, Y., and Zhao, W. (2011). Analyzing evolution of variability in a software product line: From contexts and requirements to features. *Information and Software Technology*, 53(7):707 – 721.

Pohl, K., Bockle, G., and Linden, F. J. v. d. (2005). *Software Product Line Engineering: Foundations, Principles and Techniques*. Springer-Verlag New York, Inc.

Quynh, P. T.; Thang, H. Q. (2009). Dynamic coupling metrics for service – oriented software. *International Journal of Computer Science and Engineering*, 1.

Ribeiro, H. B. G., d. L. Meira, S. R., d. Almeida, E. S., Lucredio, D., Alvaro, A., Alves, V., and Garcia, V. C. (2010). An assessment on technologies for implementing core assets in service-oriented product lines. In *2010 Fourth Brazilian Symposium on Software Components, Architectures and Reuse (SBCARS)*, pages 90–99.

Räihä, O. (2010). Survey: A survey on search-based software design. *Computer Science Review*, 4(4):203–249.

Sant'Anna, C. N. (2008). *On the Modularity of Aspect-Oriented Design: A Concern-Driven Measurement Approach*. PhD thesis, PUC-Rio, Rio de Janeiro/RJ, Brasil.

Santos, M. C. B., Moschetta, W., Colanzi, T. E., and OliveiraJr, E. A. (2015). Otimizando o projeto de arquitetura de linha de produto de software com muitos objetivos: um estudo exploratório. *VI Workshop de Engenharia de Software Baseada em Busca (WESB)*.

SEI (2016). Arcade Game Maker pedagogical product line. http://www.sei.cmu.edu/productlines/ppl/. Accessed on 10/05/2016.

Simons, C. and Parmee, I. (2012). Elegant object-oriented software design via interactive, evolutionary computation. *IEEE Transactions on Systems, Man, and Cybernetics*, 42(6):1797 –1805.

Wüst, J. (2013). SDMetrics. http://www.sdmetrics.com/. Accessed on 05/05/2016.

Zhang, T., Deng, L., Wu, J., Zhou, Q., and Ma, C. (2008). Some metrics for accessing quality of product line architecture. In *2008 International Conference on Computer Science and Software Engineering*, volume 2, pages 500–503.

Um estudo sobre o uso do algoritmo de Colônia de Formigas para otimização de arquiteturas baseadas em componentes

Mariane Affonso Medeiros[1], **Filipe Roseiro Côgo**[1], **Marco Aurélio Graciotto Silva**[1]

[1] Departamento Acadêmico de Computação
Universidade Tecnológica Federal do Paraná
Campo Mourão, PR

mmedeiros@alunos.utfpr.edu.br

{filiper,magsilva}@utfpr.edu.br

Resumo. *O design arquitetural é uma fase crítica no desenvolvimento de software, pois decisões tomadas nesta fase têm um impacto significativo no custo e qualidade do sistema final. Para auxiliar engenheiros de software a definir arquiteturas adequadas, métodos de otimização arquitetural vêm sendo utilizados para propor diretrizes e recomendações a fim de identificar elementos arquiteturais, recuperar e otimizar arquiteturas. Este trabalho investiga a utilização e o comportamento da metaheurística Otimização por Colônia de Formigas (ACO) para otimização de arquiteturas de software baseado em componentes considerando métricas de coesão, acoplamento e o estilo arquitetural. A abordagem proposta foi avaliada a partir de experimentos, aplicando o algoritmo em um sistema real. Considerando as métricas adotadas, o ACO obteve resultados cerca de 11% melhor que o valor obtido pela arquitetura inicial.*

1. Introdução

A arquitetura de software é uma descrição de sistemas que auxilia na compreensão do comportamento de software [Garlan 2014]. Ela representa diversos atributos de qualidade do sistema, tais como desempenho, modificabilidade e segurança, além disso, a construção de uma arquitetura efetiva permite identificar riscos de design e mitigá-los nos primeiros estágios de desenvolvimento [SEI 2015].

Com a crescente demanda por sistemas de software complexos, o design arquitetural se tornou uma importante atividade, demandando o desenvolvimento de técnicas que auxiliem na construção de arquiteturas de software com qualidade [Garlan 2000]. Embora os arquitetos de software aprendam o ofício de construção arquitetural através de experiências vividas por diversos projetos passados [Garlan 2000] é interessante que sejam auxiliados por métodos que automatizam a procura por uma arquitetura adequada levando em consideração métricas de qualidade e características do sistema. Estes métodos são chamados de métodos de otimização arquitetural [Aleti et al. 2013].

Otimização arquitetural pertence a uma área de pesquisa chamada Engenharia de Software Baseada em Busca [Harman e Jones 2001]. A ideia central desta área é a conversão de problemas da Engenharia de Software em problemas de busca. A otimização arquitetural pode ser modelada como um problema de busca, considerando que utiliza critérios pré-estabelecidos e características do sistema para alterar a arquitetura até que

esta esteja na condição mais aceitável possível. Esta condição é medida por métricas de qualidade que avaliam a qualidade arquitetural. Este trabalho utiliza a metaheurística Otimização por Colônia de Formigas (ACO) para otimizar arquiteturas de software.

Algoritmos ACO utilizam uma colônia de formigas artificiais para solucionar problemas. Estas formigas se movimentam em um espaço de busca em várias direções a fim de construir uma solução para o problema em questão. As formigas artificiais depositam feromônio pelo caminho que passam, com o objetivo de informar a outros membros da colônia que aquele caminho em questão já foi explorado. O feromônio depositado nos caminhos é utilizado pelas formigas durante o processo de construção da solução. A quantidade de feromônio depositado em um caminho é proporcional a qualidade da solução construída. Dessa forma os melhores caminhos (avaliados pela função objetivo) têm mais feromônio, aumentando a probabilidade deste caminho ser escolhido pelas próximas formigas. O ACO pode ainda utilizar uma informação heuristica para auxiliar na construção da solução.

O objetivo deste trabalho é observar o comportamento do algoritmo Otimização por Colônia de Formigas para otimização de arquiteturas de software baseada em componentes quanto a coesão, acoplamento e preservação do estilo arquitetural. Dessa forma, espera-se a diminuição do esforço para reparação da arquitetura pelo engenheiro de software durante o desenvolvimento do sistema.

O restante deste trabalho esta organizado da seguinte forma. A Seção 2 apresenta a abordagem de otimização baseada em ACO. Os resultados quanto à aplicação da abordagem são discutidos na Seção 3. Trabalhos relacionados são tratados na Seção 4. Por fim, a Seção 5 discorre sobre as conclusões e trabalhos futuros.

2. Abordagem Proposta

Esta seção descreve as etapas para o desenvolvimento do algoritmo de otimização baseado em colônia de formigas feito neste trabalho, denominado *ACO2SoftwareArchitecture*.

2.1. Representação da arquitetura

O algoritmo utiliza como entrada uma arquitetura baseada em componentes. Esta arquitetura deve ser representada por um modelo UML, escrito em arquivo XMI, no qual devem estar presentes elementos da arquitetura como: relacionamentos, classes e componentes. Caso não exista um arquivo XMI, o modelo UML pode ser extraído a partir do código fonte com o auxílio da ferramenta Modisco[1]. Para as arquiteturas recuperadas a partir de código fonte e que não possuem componentes definidos explicitamente, os pacotes do projeto foram considerados como componentes.

O algoritmo parte do modelo UML para extrair relacionamentos, classes e componentes da arquitetura. Para fazer esta extração, utilizamos a API UML2[2], para identificar os elementos do modelo UML e extrair a quantidade de classes, componentes, relacionamentos entre classes e componentes e os relacionamentos entre as classes da arquitetura.

Como uma das propostas deste trabalho é, além da otimização da arquitetura, a preservação do estilo arquitetural, é possível informar ao algoritmo projetos com o estilo

[1] `https://eclipse.org/MoDisco/`.

[2] `http://www.eclipse.org/modeling/mdt/?project=uml2`.

arquitetural definido no modelo UML. Para isso, utilizamos Perfis (*Profiles*) e Esteriótipos (*Stereotypes*) para anotar o estilo no modelo UML. O perfil UML provê uma forma genérica para customizar ou personalizar modelos UML para domínios e plataformas específicas [Rumbaugh et al. 2004], definindo estereótipos, *tagged values* ou restrições que podem ser aplicados em determinados elementos do modelo. Sendo assim, criamos um Perfil chamado *ArchitectureStyle* que representa o estilo arquitetural, restringindo-nos, neste trabalho, a representar o estilo arquitetural em camadas. Ele contém o estereótipo chamado *Layered* que representa o tipo de estilo arquitetural considerado. Este estereótipo possui uma variável *id*, que contém o nome da camada a que o componente pertence.

O algoritmo ACO proposto neste trabalho utiliza a informação do estilo arquitetural para verificar se as soluções geradas quebram o estilo imposto no modelo arquitetural. Portanto, os elementos denotados no modelo serão armazenados em estruturas de dados para que o algoritmo possa fazer esta averiguação e levar em consideração a informação do estilo arquitetural para construir uma solução.

O ACO desenvolvido neste trabalho possui duas matrizes de feromônio: uma representando os componentes e outra representando os relacionamentos entre as classes. Essas matrizes informam a quantidade de feromônio depositada em determinada posição, de forma que durante o processo de construção da solução, a formiga seja influenciada pelas combinações mais atrativas (que possuem maior valor de feromônio). Sendo assim, os valores armazenados nestas matrizes estão diretamente relacionados com os valores das métricas de qualidade. Conforme as formigas constroem suas soluções e as melhores são selecionadas, esta matriz é atualizada. A matriz que representa os relacionamentos entre classes possui uma coluna N, a qual representa a possibilidade de não combinar a classe C_x com outras, ou seja, C_x não ter relacionamentos.

2.2. Métricas e função objetivo

Uma solução proposta pela metaheurística precisa ser avaliada através de uma métrica de qualidade. As métricas podem ser representadas em termos de equações para que se possa utilizá-las como função objetivo da metaheurística. Esta seção apresenta a métrica utilizada para medir a qualidade de uma arquitetura e a define em termos da função objetivo utilizada pelo ACO.

A qualidade da arquitetura pode ser avaliada em termos de coesão e acoplamento. Normalmente é desejável ter uma alta coesão e um baixo acoplamento para uma arquitetura. Coesão mede quanto as classes de um componente estão intra-relacionadas (intra-conectividade) e acoplamento mede quão inter-dependentes (inter-conectividade) são dois componentes [Saed e Kadir 2011]. A intra-conectividade representa relacionamentos entre classes que estão em um mesmo componente. Inter-conectividade representa dependência entre componentes, ou seja, relacionamentos entre classes que pertencem a componentes diferentes.

As medidas de intra e inter-conectividade podem ser encapsuladas na métrica *Modularization Quality* (MQ), que determina a qualidade de um grafo de dependência de módulos. A métrica MQ será utilizada como função objetivo do algoritmo ACO. Os valores desta métrica variam entre -1 e 1, sendo que -1 significa um componente sem coesão interna e 1 significa um componente sem acoplamento externo [Mancoridis et al. 1999].

Portanto a metaheurística irá buscar pelo maior valor de MQ possível.

2.3. ACO para o problema de otimização de arquitetura de software

Estabelecida a representação inicial da arquitetura e a função objetivo, procedemos para a execução das iterações de otimização, ao final das quais o algoritmo retornará um arquivo texto que representa a solução com o melhor valor da função objetivo obtida. O procedimento de construção da solução dá-se em duas partes. Na primeira etapa a formiga k percorre cada linha da matriz de feromônio de classe $\times$ componente e de classe $\times$ classe, fazendo as seguintes combinações: da classe i dentro do componente j, da classe i com a classe j e calcula a probabilidade destas combinações. Esta probabilidade é dada pela Equação 1, cujo conjunto $Element$ pode ser tanto os componentes quanto as classes da arquitetura, dessa forma a variável m assume valor das classes e componentes dependendo do conjunto em questão. Caso seja componente, a equação representa a probabilidade da formiga k inserir a classe i no componente j na iteração t. Caso seja classe, ela estabelece a probabilidade de relacionamento entre duas classes.

$$P_{i,j}^{k}(t) = \frac{\tau_{i,j}^{\alpha}(t) \cdot \eta_{i,j}^{\beta}(t)}{\sum_{m=0}^{|Element|} \tau_{i,m}^{\alpha}(t) \cdot \eta_{i,m}^{\beta}(t)} \tag{1}$$

- $\tau_{i,j}$ é o feromônio depositado na posição i, j da matriz de feromônio;
- $\eta_{i,j}$ informação heurística. Usada neste trabalho como penalizações, as arquiteturas que infringem seu estilo arquitetural serão desvalorizadas e tendem a não ser escolhidas, o cálculo desta penalidade é definida na Seção 2.4;
- α e β são parâmetros que salientam a importância da informação definida pelo feromônio (τ) e da informação heurística definida (η).
- $Element$ é o conjunto de componentes e de classes da arquitetura, representado no algoritmo por uma matriz de componentes e uma matriz de classes.

Cada probabilidade calculada é inserida em um vetor de probabilidades. Ao final da linha é utilizado o método da roleta a fim de escolher uma das probabilidades. Esta solução parcial é armazenada e a formiga continua construindo sua solução, até o fim da matriz. Ao final destes procedimentos a formiga k terá sua solução construída.

No processo de penalizações as arquiteturas que infringem o estilo arquitetural têm por objetivo analisar quais combinações de classes quebram a regra do estilo em camada. Essa informação servirá para a informação heurística no momento do cálculo da probabilidade de combinação entre as classes.

Após todas as formigas construírem suas soluções, é verificado qual possui o melhor valor de função objetivo. Selecionada a melhor solução, os valores de feromônio da matriz de classe por componente e de classes por classe são atualizados. A atualização do feromônio é feita baseada na Equação 2.

$$\tau_{i,j}(t+1) = (1-\rho) \cdot \tau_{i,j}(t) + \Delta\tau_{i,j}(t), \tag{2}$$

Em que $\Delta\tau_{i,j}(t)$ representa o incremento de feromônio no tempo $t+1$ na combinação (i, j). $\Delta\tau_{i,j}(t)$ é diretamente proporcional a função objetivo, pois a quantidade de depósito de feromônio está atrelada ao valor da função objetivo. A constante ρ representa

a taxa de evaporação de ferômonio. Este valor tem como objetivo tirar uma quantia de feromônio das matrizes para que, soluções com valores ruins de função objetivo acabem sendo desvalorizadas e não sejam mais selecionadas.

2.4. Informação Heurística

A informação heurística do algoritmo é abordada neste trabalho como penalizações. As penalizações são dadas verificando o estilo arquitetural da arquitetura de entrada. O estilo arquitetural considerado para este trabalho foi: estilo arquitetural em camada. Segundo [Garlan e Shaw 1993], o estilo arquitetural em camadas é organizado hierarquicamente. Cada elemento de uma camada provê serviços para a camada acima e serve como cliente para a camada abaixo. As camadas são limitadas por regras que delimitam a comunicação entre camadas: elementos em cada camada podem acessar somente elementos em sua própria camada, ou elementos na camada diretamente abaixo. O estilo arquitetural em camadas possui algumas regras, que são: um elemento de uma camada x só pode se relacionar com um elemento de uma camada $x + 1$ ou um elemento da camada x [Mariani et al. 2016].

Para verificar se a arquitetura infringe o estilo arquitetural, percorre-se a matriz de relacionamentos classe$\times$classe e verifica se o relacionamento entre a classe i e j pertencentes à camadas $layer_i$ e $layer_j$, quebra as regras do estilo em questão. Cada relacionamento que quebra a regra é adicionado a uma lista de vetores, que armazena os relacionamentos que quebraram a regra e em um mapa que armazena a classe e a quantidade de vezes que esta quebrou a regra. Dessa forma, teremos todos os relacionamentos que quebraram a regra e quantas vezes cada classe quebrou a regra.

Este processo é feito para que, no momento da combinação das classes para gerar os relacionamentos, seja possível informar para a função de cálculo da probabilidade o total de vezes que cada classe envolvida no relacionamento quebra a regra. Este total é dividido pelo total de quebras de regra que a arquitetura possui. O valor passado para a função de probabilidade é o seguinte:

$$\eta_{i,j} = 1 - \frac{total_i + total_j}{totalGeralDeQuebrasDaRegra} \tag{3}$$

A Equação 3, que caracteriza quebras de regras do estilo arquitetural para um relacionamento entre as classes i, j, é a informação heurística ($\eta_{i,j}$) utilizada para auxiliar na busca pela melhor solução.

3. Avaliação Experimental

Os experimentos foram executados sobre a versão 1.1.0 do software *Apache Ant* [Apache Software Foundation 2000]. Para que fosse possível avaliar a abordagem quanto ao estilo arquitetural, criamos um modelo com o estilo arquitetural em camadas, adotando a seguinte abordagem: cada pacote da raiz principal do projeto é uma camada; subpacotes pertencem à mesma camada que seu pacote pai, desde que dentro deste subpacote tenha apenas classes; caso um subpacote possua mais subpacotes dentro, então ele será uma nova camada. A quantidade de componentes, classes, camadas e o valor de MQ da versão utilizada estão apresentados na Tabela 1.

Tabela 1. Medidas do Apache-Ant considerando estilo arquitetural em camadas.

#Classes	#Componentes	#Camadas	MQ
97	7	6	0.55534

Para cada teste feito, foi aplicado várias combinações dos parâmetros de configuração para verificar qual obtinha melhor resultado do valor MQ. Além disso, cada configuração foi executada 10 vezes devido à característica probabilística do algoritmo. Os resultados obtidos foram comparados com o modelo original da arquitetura, comparando-se resultados de MQ da solução indicada pelo ACO com o valor MQ da solução original.

A partir da análise dos resultados obtidos pelo ACO, é possível averiguar que, partindo da métrica MQ, o algoritmo consegue encontrar soluções melhores do que as iniciais, conforme apresentado na coluna MQ da Tabela 2, sem considerar a avaliação de estilos arquiteturais. A coluna *Id* representa um identificador para a configuração, para que possa ser referenciada mais adiante no texto. É possível verificar que mesmo a pior solução encontrada pelo algoritmo (Id 7), com MQ de 0,5629, é melhor que a solução de entrada, que possui valor MQ de 0,5553, ou seja a melhor solução obteve um valor 11% melhor. Podemos observar que considerando o tamanho da arquitetura e a quantidade de iterações 100, valores de ρ baixo conseguem obter bons resultados. Isso acontece pois, como a arquitetura é relativamente pequena, 100 iterações é um valor alto. Sendo assim, mesmo com uma baixa taxa de evaporação, por ter muitas iterações ainda assim a configuração alcança bons resultados.

Tabela 2. Resultados para Apache Ant 1.1.0 sem avaliar o estilo arquitetural.

Id	Iterações	Formigas	ρ	α	β	MQ
1	100	15	0.2	0.4	0.2	0.61802
2	100	5	0.4	0.2	0.2	0.61765
3	100	15	0.4	0.2	0.4	0.61033
4	100	5	0.1	0.9	0.8	0.59660
5	50	20	0.7	0.4	0.9	0.59499
6	30	20	0.4	0.6	0.9	0.59274
7	20	20	0.6	0.4	0.1	0.59023

A Tabela 4 reporta os resultados obtidos a partir de uma arquitetura de entrada com estilo arquitetural em camadas da versão 1.1.0 do software Apache-Ant. A ordenação dos dados da Tabela se dão de forma decrescente considerando a coluna MQ. Podemos analisar pela coluna Penalizações que as soluções com maiores valores de MQ e iterações, infringiram menos as regras do estilo arquitetural do que as soluções com valores baixos de MQ e iterações. É possível notar que, com a taxa de evaporação (ρ) baixa, os resultados obtidos de MQ são piores e o número de relacionamentos que violam a regra do estilo é mais alto em relação às melhores soluções. Como a taxa de evaporação entre uma iteração e outra é baixa, há uma chance maior das formigas ficarem muitas iterações percorrendo caminhos que não produzem bons resultados, resultando assim em valores MQ mais baixos.

A Tabela 3 mostra o tempo de execução do ACO em uma máquina com sistema operacional Fedora, processador Intel Core i3 2.20GHz e memória de 4 Gb. As config-

Tabela 3. Tempo de execução dos experimento em máquina com processador i3

Id	Versão	Estilo	Iterações	Formigas	ρ	α	β	Tempo De Execução
1	1.1.0	Não	100	15	0.2	0.4	0.2	00:04:32
2	1.1.0	Não	20	20	0.6	0.4	0.1	00:01:21
3	1.1.0	Sim	50	20	0.7	0.4	0.0	00:03:30
4	1.1.0	Sim	20	20	0.2	0.4	0.1	00:01:31

Tabela 4. Resultados para Apache Ant 1.1.0 considerando o estilo arquitetural.

Id	Iterações	Formigas	ρ	α	β	MQ	Penalizações
1	50	20	0.7	0.4	0.0	0.60244	0
2	100	15	0.8	0.2	0.4	0.60019	0
3	30	20	0.7	0.4	0.3	0.59660	3
4	30	20	0.7	0.4	0.7	0.59563	0
5	20	20	0.5	0.4	0.1	0.58182	8
6	20	20	0.2	0.4	0.1	0.56292	16

urações mostradas são referentes as que obtiveram melhor e pior resultados de MQ. A coluna *Estilo* significa se há estilo arquitetural. As colunas *Iterações*, *Formigas*, ρ, α e β mostram os valores dos parâmetros e a coluna *Tempo de Execução* marca o tempo que o ACO demorou para executar a configuração.

A Figura 1 apresenta a taxa de convergência dos valores MQ obtidos pelo algoritmo considerando estilo arquitetural em camadas. Observa-se a evolução do MQ para três configurações definidas na Tabela 4 e distintas com exceção da quantidade de formigas (sempre 20): a linha azul representa a configuração Id 1 ($\rho = 0,7$, $\alpha = 0,4$ e $\beta = 0,0$) e que obteve $MQ = 0,60244$; a linha vermelha representa a configuração Id 3 ($\rho = 0,7$, $\alpha = 0,4$, $\beta = 0,7$) e com $MQ = 0,59648$; e a linha alaranjada representa a configuração Id 4 ($\rho = 0,7$, $\alpha = 0,4$, $\beta = 0,3$) com $MQ = 0,59298$. Observa-se que as configurações, após convergirem para a melhor solução, ficaram algumas iterações produzindo os mesmos valores MQs pois já haviam encontrado seu valor máximo. Esta observação nos leva a concluir que considerar como condição de parada do algoritmo um número máximo de iterações não é uma boa alternativa, por causa do tempo de execução desperdiçado produzindo soluções iguais. O ideal seria adotar como critério de parada a diferença entre os valores da função objetivo entre uma iteração e outra. Podemos observar ainda que a variação do parâmetro β causou impacto nas soluções, fazendo com que a configuração com β maior convergisse mais rápido para um bom resultado.

A Figura 2, referente as mesmas configurações listadas acima, mostra a evolução das penalidades. Podemos observar o valor utilizado de β nas configurações e o que isto causou na evolução das penalidades ao longo das iterações. A linha azul, que representa a configuração de Id 1 e que possui $\beta = 0,0$, iniciou com soluções que infringiam muito as regras do estilo, enquanto as configurações com $\beta = 0,7$ (linha vermelha) e $0,3$ (linha laranjada) produziram soluções que infringiam menos as regras do estilo ao longo das iterações. No entanto, apesar deste comportamento, ambas as configurações acabaram chegando a penalidade igual a zero, independente dos valores de β. Constatamos ao longo dos testes que os valores de β, acabam sendo invalidados pela função MQ, isso pelo fato de que a regra do estilo arquitetural em camadas penaliza relacionamentos entre

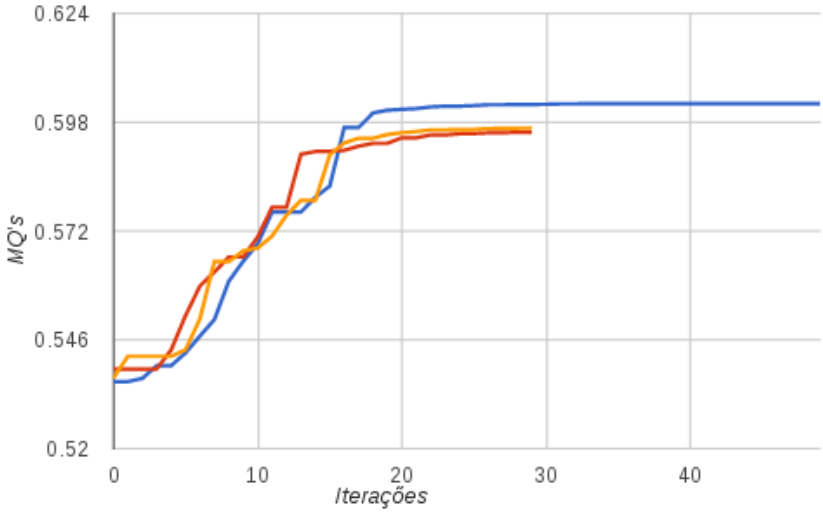


Figura 1. Evolução do valor de MQ para Apache Ant 1.1.0 com estilo em camadas.

classes em que: a classe a do relacionamento está em um componente na camada x e a classe b está em um componente na camada $x + 2$. Dessa maneira, essa penalidade está condicionada à existência de um relacionamento externo entre as classes a e b. Como dito anteriormente, a função MQ busca reduzir a zero o valor de relacionamentos externos. Dessa maneira, o valor de β não influencia no resultado final das penalizações para o estilo arquitetural tratado neste artigo, pois a função objetivo já reduz os relacionamentos externos o que automaticamente fará com que não haja penalidades entre os relacionamentos.

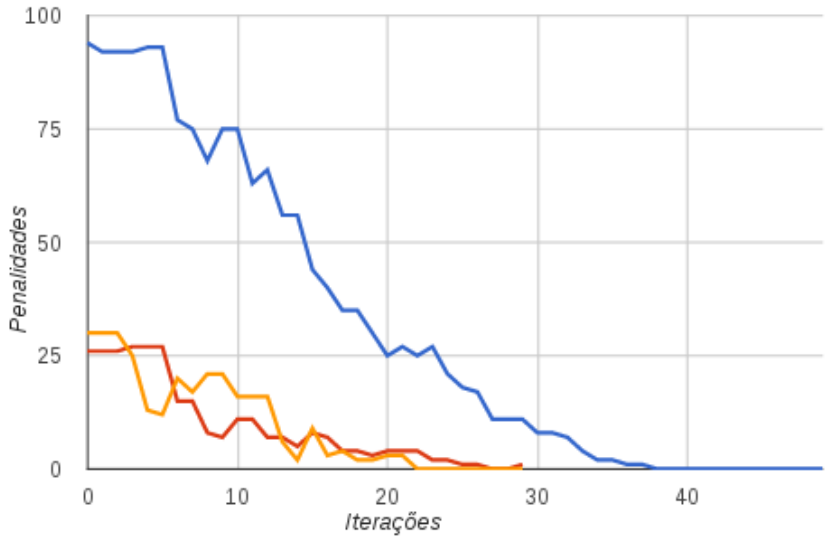


Figura 2. Evolução das penalidades das soluções geradas ao longo das iterações.

4. Trabalhos Relacionados

Dois trabalhos tratam da otimização do design de arquiteturas considerando ACO e da avaliação da abordagem proposta. O primeiro utiliza ACO para propor um design

otimizado de diagrama de classes de um software de acordo com métricas de coesão e acoplamento (CK) [Tawosi et al. 2015]. A representação da arquitetura é um grafo direcionado acíclico, cujos nós são organizados em duas dimensões: responsabilidades e classes. Dessa maneira, a seleção de um determinado nó representa a possível atribuição de uma responsabilidade a uma classe. O algoritmo inicia com o preprocessamento da arquitetura de entrada e, em seguida inicia, a busca, gerando designs candidatos para encontrar a solução mais próxima da ótima possível. As formigas procuram caminhos no grafo acíclico para gerar suas soluções e depositam feromônio nestes caminhos percorridos de acordo com o valor de uma função multi-objetivos [Tawosi et al. 2015]. Os autores propõem ainda uma métrica chamada *Meaningfulness Metric* (MM), utilizada para medir quanto o diagrama de classes é compreensível para um humano. Em comparação ao nosso trabalho, a escolha de métricas possui objetivos similares. No entanto, ao invés de utilizar diversas métricas e uma função multiobjetivo, optamos por uma medida que trata coesão e acoplamento (MQ). Embora não tratemos na compreensibilidade do modelo gerado diretamente, a utilização de penalidades em nossa abordagem busca preservar importantes decisões sobre a arquitetura relacionadas à facilidade de compreensão lógica.

O segundo trabalho compara três métodos para otimização de design de software: otimização por colônia de formigas, algoritmo genético e busca gulosa [Simons e Smith 2013]. O modelo de representação de um design arquitetural (considerando classes, métodos e atributos) é um conjunto com uma sequência de inteiros, em que cada valor do conjunto representa um elemento do design do software. Na implementação apresentada, cada formiga percorre o espaço de busca (combinando os elementos atributos, métodos e classes) para construir sua solução, escolhendo cada combinação através do feromônio depositado na posição a matriz que indica aquela combinação. Após uma formiga finalizar a construção de sua solução, esta é avaliada pela função objetivo e a matriz de feromônio é atualizada. As métricas consideradas para verificar a qualidade de uma solução são: acoplamento do objeto e elegância do modelo arquitetural, esta sendo medida através do número de atributos e métodos de cada classe e da quantidade de atributos em cada método de uma classe do modelo. Em nosso trabalho, não consideramos os atributos de uma classe, o que não permite comparar sob a perspectiva de elegância, mas aspectos de acoplamento são considerados.

5. Conclusões

Neste trabalho foi explorado a utilização da metaheurística de otimização por colônia de formigas para resolver o problema de otimização de arquitetura de software. A implementação desta abordagem proporcionou um melhor conhecimento de como o ACO se comporta para este tipo de problema. A métrica utilizada foi *Modularization Quality* (MQ), que busca reduzir acoplamento e aumentar a coesão dos componentes da arquitetura, melhorando assim a manutenibilidade do software. A métrica proposta neste trabalho busca verificar o estilo arquitetural do sistema, aplicando penalidades às soluções que quebram regras do estilo arquitetural imposto.

Os experimentos foram conduzidos para que fosse possível averiguar se o ACO era apto a obter resultados satisfatórios em relação a arquitetura inicial do sistema. Para a execução dos experimentos foi aplicado a abordagem proposta sobre uma versão do software Apache Ant. Os resultados obtidos mostraram que o ACO tem capacidade para produzir resultados satisfatórios considerando as métricas usadas, obtendo para o primeiro

experimento resultados cerca de 11% maior que da arquitetura inicial e para o segundo experimento obtendo MQ 7% maior que a arquitetura inicial. Dessa maneira, os resultados obtidos pelo algoritmo mostraram melhores valores MQ que a arquitetura inicial. Portanto o ACO se mostra uma opção interessante para ser utilizada e explorada para problemas de otimização arquitetural.

Dado o desenvolvimento deste trabalho podemos apontar alguns trabalhos futuros necessários para melhorar a utilização do ACO para otimização arquitetural. Experimentos com outras metaheurísticas, a fim de comparar os resultados de diferentes abordagens. Experimentos considerando outros estilos arquiteturais, para analisar como se comporta a métrica de penalidades proposta. Necessidade do uso de técnicas para redução do tempo de execução do algoritmo. Utilização de outras métricas de qualidade para verificar a capacidade do ACO de lidar com funções multiobjetivas.

Referências

Aleti, A., Buhnova, B., Grunske, L., Koziolek, A., e Meedeniya, I. (2013). Software architecture optimization methods: A systematic literature review. *IEEE Trans. Softw. Eng.*, 39(5):658–683.

Apache Software Foundation (2000). Ant. Programa de Computador.

Garlan, D. (2000). Software architecture: A roadmap. In *Proceedings of the Conference on The Future of Software Engineering*, ICSE '00, p. 91–101, New York, NY, USA. ACM.

Garlan, D. (2014). Software architecture: A travelogue. In *Proceedings of the on Future of Software Engineering*, FOSE 2014, p. 29–39, New York, NY, USA. ACM.

Garlan, D. e Shaw, M. (1993). An introduction to software architecture. *Advances in software engineering and knowledge engineering*, 1(3.4).

Harman, M. e Jones, B. F. (2001). Search-based software engineering. *Information and Software Technology*, 43(14):833 – 839.

Mancoridis, S., Mitchell, B. S., Chen, Y., e Gansner, E. R. (1999). Bunch: a clustering tool for the recovery and maintenance of software system structures. In *Software Maintenance, 1999. (ICSM '99) Proceedings. IEEE International Conference on*, p. 50–59.

Mariani, T., Colanzi, T. E., e Vergilio, S. R. (2016). Preserving architectural styles in the search based design of software product line architectures. *Journal of Systems and Software*, 115:157 – 173.

Rumbaugh, J., Jacobson, I., e Booch, G. (2004). *Unified Modeling Language Reference Manual, The (2Nd Edition)*. Pearson Higher Education, Boston, MA, USA.

Saed, A. e Kadir, W. (2011). Applying particle swarm optimization to software performance prediction an introduction to the approach. In *Software Engineering (MySEC), 2011 5th Malaysian Conference in*, p. 207–212.

SEI (2015). Defining software architecture.

Simons, C. e Smith, J. (2013). A comparison of meta-heuristic search for interactive software design. *Soft Computing*, 17(11):2147–2162.

Tawosi, V., Jalili, S., e Hasheminejad, S. M. H. (2015). Automated software design using ant colony optimization with semantic network support. *Journal of Systems and Software*, 109:1 – 17.