



# CBSOFT

## II Congresso Brasileiro de Software: Teoria e Prática

**SBES**

**SBLP**

**SBMF**

**SBCARS**

**XXV Simpósio Brasileiro de Engenharia de Software**

**XV Simpósio Brasileiro de Linguagens de Programação**

**XIV Simpósio Brasileiro de Métodos Formais**

**V Simpósio Brasileiro de Componentes, Arquiteturas e Reutilização de Software**

Miniconferência Latino-Americana de Linguagens de Padrões para Programação

XVIII Sessão de Ferramentas | IV FEES | II Trilha da Indústria | I WTDSoft

V LA-WASP | V WDDS | V SAST | II AutoSoft | II WB-DSDM | II WESB | I WBVS

# Anais

**WESB 2011**

II Workshop de Engenharia de Software Baseada em Buscas



# WESB 2011

## II Workshop de Engenharia de Software Baseada em Buscas

26 de setembro de 2011

São Paulo - SP - Brasil

# ANAIIS

**Volume 12  
ISSN: 2178-6097**

### **Coordenação do WESB 2011**

Gledson Elias

### **Coordenação do CBSoft 2011**

Marcelo Fantinato - Coordenador Geral  
Luciano Silva - Vice Coordenador

### **Coordenação de Workshops 2011**

Marco Túlio Valente

### **Realização**

Universidade de São Paulo (USP)  
Universidade Presbiteriana Mackenzie (Mackenzie)

### **Promoção**

Sociedade Brasileira de Computação (SBC)

### **Patrocínio**

Fapesp, Capes, CNPq, Mack Pesquisa, PRCEU-USP, Google



# WESB 2011

## 2nd Brazilian Workshop on Search Based Software Engineering

September 26, 2011

São Paulo - SP - Brazil

# PROCEEDINGS

**Volume 12**  
**ISSN: 2178-6097**

### **WESB 2011 Chairs**

Gledson Elias

### **CBSOFT 2011 Chairs**

Marcelo Fantinato - General Chair

Luciano Silva - Co-chair

### **Workshops 2011 Chair**

Marco Túlio Valente

### **Organization**

Universidade de São Paulo (USP)

Universidade Presbiteriana Mackenzie (Mackenzie)

### **Promotion**

Brazilian Computing Society (SBC)

### **Sponsorship**

Fapesp, Capes, CNPq, Mack Pesquisa, PRCEU-USP, Google

## DADOS INTERNACIONAIS DE CATALOGAÇÃO-NA-PUBLICAÇÃO

(Universidade de São Paulo. Escola de Artes e Ciências e Humanidades. Biblioteca)

Congresso Brasileiro de Software : Teoria e Prática (2. : 2011 : São Paulo, SP).

WESB 2011 : anais [do] II Workshop de Engenharia de Software Baseada em Buscas / II Congresso Brasileiro de Software : Teoria e Prática ; coordenação [de] Marcelo Fantinato, Luciano Silva ; coordenação do WESB 2011 [de] Gledson Elias. – São Paulo, 2011.

xii, 72 p.

Evento no dia 26 de setembro de 2011.

Realização da Universidade de São Paulo e Universidade Presbiteriana Mackenzie. Promoção da Sociedade Brasileira de Computação.

Publicação composta por 14 volumes, sendo este o volume 12.

ISSN : 2178-6097.

1. Engenharia de software. I. Fantinato, M., coord. II. Silva, L., coord. III. Elias, G., coord. IV. Título.

CDD 22. ed. – 005.1

Autorizo a reprodução parcial ou total desta obra, para fins acadêmicos, desde que citada a fonte.

## Apresentação

---

É com enorme satisfação que aceitei coordenar o II Workshop de Engenharia de Software Baseada em Buscas (WESB 2011), que acontece como evento co-alocado ao II Congresso Brasileiro de Software (CBSOFT 2011).

O WESB reúne pesquisadores, professores, estudantes e profissionais, constituindo assim um importante fórum de discussão sobre a aplicação de técnicas de busca para solucionar problemas da Engenharia de Software. No contexto do workshop, técnicas de busca englobam tanto técnicas tradicionais, como força bruta ou branch-and-bound, quanto técnicas baseadas em meta-heurísticas, como algoritmos genéticos e arrefecimento simulado. O WESB é um workshop sobre fundamentos teóricos e experiências práticas da Engenharia de Software Baseada em Buscas (*Search Based Software Engineering – SBSE*) em projetos acadêmicos e industriais. Relatos de avanços no estado da arte e no estado da prática constituem as principais contribuições dos trabalhos selecionados, promovendo a troca de experiências, opiniões e debates entre os participantes.

As contribuições e experiências da edição anterior foram bastante enriquecedoras, o que motivou a realização do workshop também neste ano de 2011, visando reforçar as contribuições e experiências da edição anterior e trazendo ainda mais conhecimento aos participantes. Além disso, acredito ser esta segunda edição uma excelente oportunidade para consolidar e expandir no Brasil a pesquisa teórica e aplicada na área.

Embora seja uma área nascente em nosso país, diversos grupos de pesquisa têm atuado neste tema, o que pode ser comprovado pelos 14 artigos submetidos, dos quais foram selecionados 9, visando manter a qualidade técnica, e, ao mesmo tempo, proporcionar uma representatividade institucional satisfatória para uma segunda edição de um evento com potencial de crescimento. O processo de seleção garantiu que cada submissão tivesse pelo menos 3 avaliações independentes de especialistas da área.

O programa do evento se destaca pela excelência das sessões técnicas, constituídas pelos trabalhos selecionados de abrangência nacional, identificando-se assim potenciais grupos de pesquisa na área.

Agradeço o apoio recebido do CBSOFT por abrigar, pela segunda vez, este evento. Registro aqui meus sinceros agradecimentos ao Prof. Marcelo Fantinato, responsável pela coordenação geral do CBSOFT 2011. Agradeço aos autores pelo interesse e pela qualidade de suas contribuições, e, também, aos membros do Comitê de Programa pelo excelente trabalho realizado na seleção de artigos. Por fim, agradeço ao Comitê de Organização pelo apoio ao longo de todas as etapas.

Em nome daqueles que direta ou indiretamente contribuíram para a realização do WESB 2011, agradeço a participação e desejo um evento bastante produtivo.

São Paulo, setembro de 2011.

**Gledson Elias**  
Coordenador do WESB 2011  
CBSOFT 2011

## Foreword

---

It is with great pleasure that I accepted to coordinate the 2nd Brazilian Workshop on Search Based Software Engineering (WESB 2011), which is held as a co-located event of the II Brazilian Conference on Software (CBSOFT 2011).

The WESB workshop gathers researchers, professors, students and professionals, constituting a key discussion forum about the adoption of search techniques for solving problems of the Software Engineering field. In the context of the workshop, search techniques include both traditional ones, such as branch-and-bound, and meta-heuristic based ones, such as genetic algorithms and simulated annealing. WESB is a workshop about theoretical fundaments and practical experiences of adopting Search Based Software Engineering (SBSE) in academic and industrial projects. State-of-the-art and state-of-the-practice advances constitute the main contributions of the selected papers, promoting the exchange of experiences, opinions and debates.

The contributions and experiences of the previous edition have been very enriching and have motivated the realization of the second edition in 2011, strengthening the contributions and experiences of the previous edition, and bringing more knowledge to participants. Furthermore, I believe that this second edition is an excellent opportunity to consolidate and expand in Brazil the theoretical and applied research in the area.

Although it is an emergent area in our country, several research groups have worked in this research theme, which can be proven by the 14 submitted papers, from which 9 were selected in order to maintain technical quality and also to provide a satisfactory institutional representation in the second edition of an event with growth potential. The selection process has ensured that each submitted paper had at least 3 independent evaluations from specialists in the field.

The event program is highlighted by the excellence of the technical sessions, consisting of the selected papers of national coverage, thus identifying potential research groups in the area.

I thank the support received from CBSOFT for co-locating the second edition of the event. I record here my sincere thanks to Prof. Marcelo Fantinato, responsible for the overall coordination of CBSOFT 2011. I would like to thank all authors for their interest and the quality of their contributions, and also to members of the Program Committee for their excellent work in selecting papers. Finally, I thank the Organizing Committee for the support throughout all stages.

On behalf of those who directly or indirectly contributed to the success of WESB 2011, I would like to thank the participation of each one and wish a very productive event.

São Paulo, September 2011.

**Gledson Elias**  
WESB 2011 Chair  
CBSOFT 2011

## Breve Biografia do Coordenador do WESB 2011

---

### Gledson Elias, UFPB - Brasil

**Gledson Elias** é Doutor em Ciência da Computação na área de Engenharia de Software pelo CIn/UFPE (2002). Iniciou a carreira acadêmica em 1993, e, atualmente, é Professor do Programa de Pós-Graduação em Informática do DI/UFPB. Suas principais áreas de interesse são: Engenharia de Software Baseada em Buscas, Desenvolvimento Distribuído de Software, Reutilização de Software, Middleware e Computação Móvel. Atualmente coordena e participa de projetos de pesquisa relacionados a sistemas de recomendação baseados em buscas meta-heurísticas para agrupamentos de componentes e alocação de equipes de desenvolvimento distribuídas. Foi coordenador geral do WDBC 2004, SBES 2007 e WDDS 2007; coordenador do comitê de programa do WDBC 2004, WDDS 2008, e Sessão de Ferramentas SBES 2009; e membro do comitê diretivo do SBCARS e WDDS. Tem participado como membro do comitê de programa de diversos eventos nacionais e internacionais.

## **WESB 2011 Chair Short Biography**

---

### **Gledson Elias, UFPB - Brazil**

*Gledson Elias holds a Ph.D. degree in Computer Science from Federal University of Pernambuco (2002). He is affiliated to academic institutions since 1993, and, currently, he is a researcher in the Informatics Post-Graduate Program at Federal University of Paraíba. Research interests include: search based software engineering, global software development, software reuse, middleware and mobile computing. At present, coordinates and contributes in research projects related to recommendation systems based on metaheuristic search techniques for software component clustering and allocation of distributed development teams. He was the general chair of WDBC 2004, SBES 2007 and WDDS 2007; technical program chair of WDBC 2004, WDDS 2008 and SBES 2009 Tools Session; and member of the steering committee of SBCARS and WDDS. He has been participating as member of the technical program committee in several conferences and workshops.*

## Comitês Técnicos / Technical Committees

---

### Comitê Diretivo / Steering Committee

Arilo Claudio Dias Neto, UFAM  
Gledson Elias, UFPB  
Jerffeson Teixeira de Souza, UECE  
Márcio de Oliveira Barros, Unirio  
Silvia Regina Vergilio, UFPR

### Comitê de Programa / Program Committee

Adriana C. F. Alvim, Unirio  
Arilo Claudio Dias Neto, UFAM  
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  
Mariela Inés Cortés, UECE  
Mark Harman, University College London - UK  
Massimiliano Di Penta, University of Sannio - IT  
Mel Ó Cinnéide, University College Dublin - IE  
Pedro de Alcântara dos Santos Neto, UFPI  
Phil McMinn, University of Sheffield - UK  
Rosiane de Freitas Rodrigues, UFAM  
Silvia Regina Vergilio, UFPR

### Revisores Adicionais / Additional Reviewers

Ricardo Britto, UFPI

## **Comitê Organizador / Organizing Committee**

---

### **Coordenador Geral do CBSoft 2011 / CBSoft 2011 General Chair**

Marcelo Fantinato, EACH-USP

### **Vice-coordenador do CBSoft 2011 / CBSoft 2011 Co-chair**

Luciano Silva, FCI-Mackenzie

### **Coordenadores Locais do SBES 2011 / SBES 2011 Local Chairs**

Marcos Lordello Chaim, EACH-USP

Marco Aurelio Gerosa, IME-USP

### **Coordenadores Locais do SBLP 2011 / SBLP 2011 Local Chairs**

Denise Stringhini, FCI-Mackenzie

Alfredo Golmand, IME-USP

### **Coordenadores Locais do SBMF 2011 / SBMF 2011 Local Chairs**

Fátima L. S. Nunes Marques, EACH-USP

Ana Cristina Vieira de Melo, IME-USP

### **Coordenadores Locais do SBCARS 2011 / SBCARS 2011 Local Chairs**

Marcelo Fantinato, EACH-USP

Luciano Antonio Digiampietri, EACH-USP

### **Coordenadores Locais do MiniPLoP Brasil 2011 / MiniPLoP Brasil 2011 Local Chairs**

Eduardo Martins Guerra, ITA

Fabio Kon, IME-USP

## **Comitê de Apoio / Support Committee**

Anarosa Alves Franco Brandão, Poli-USP

Arnaldo R. A. Vallim Filho, FCI-Mackenzie

Ismar Frango Silveira, FCI-Mackenzie

Flávio Soares Correia da Silva, IME-USP

José Carlos Maldonado, ICMC-USP

Marcelo Morandini, EACH-USP

Márcio Delamaro, ICMC-USP

Paulo César Masiero, ICMC-USP

Sarajane Marques Peres, EACH-USP

Selma Melnikoff, Poli-USP

**Apoio na EACH-USP / Support at EACH-USP**

Adriano Vieira Fernandes (Web site - desenvolvimento)

Paulo Rodrigues Felisbino (Web site - desenvolvimento)

Roberto dos Santos Rocha

PET - Sistemas de Informação

DASI - Diretório Acadêmico de Sistemas de Informação

SI Jr - Empresa Junior

## Índice de Artigos / Table of Contents

---

### **Sessão Técnica 1 / Technical Session 1 - Requisitos e Teste de Software ..... 1**

Uma Avaliação do Uso de Diferentes Algoritmos Evolutivos Multiobjetivos para Integração de Classes e Aspectos..... 1

*Wesley Klewerton Guez Assunção, Thelma Elita Colanzi, Aurora Trinidad Ramirez Pozo, Silvia Regina Vergilio (UFPR)*

Priorização de Casos de Teste Baseada em Sistemas de Inferência Fuzzy ..... 9

*Thiago de A. C. Soares, Werney A. L. Lira, Ricardo de S. Britto (UFPI); Ricardo A. L. Rabêlo (USP); Pedro de A. dos Santos Neto (UFPI)*

Abordagem Híbrida para o Problema Multiobjetivo do Próximo Release..... 17

*Thiago Gomes Nepomuceno da Silva, Fabrício Gomes de Freitas, Jerffeson Teixeira de Souza (UECE)*

### **Sessão Técnica 2 / Technical Session 2 - Alocação de Equipes e Projetos ..... 25**

Uma Abordagem Baseada em Algoritmo Genético para Alocação de Equipes Distribuídas.. 25

*Bruno Ribeiro, Gledson Elias (UFPB)*

Otimização Heurística de uma Técnica para Seleção e Priorização de Portfólios Balanceados de Projetos de Software..... 33

*Fábio V. Figueiredo, Márcio de O. Barros (Unirio)*

Uma Abordagem Otimizada para o Problema de Alocação de Equipes e Escalonamento de Tarefas para a Obtenção de Cronogramas Eficientes ..... 41

*Ítalo Mendonça Rocha, Gerardo Valdisio R. Viana, Jerffeson Teixeira de Souza (UECE)*

### **Sessão Técnica 3 / Technical Session 3 - Modelagem, Estudos e Experimentos ..... 49**

Desenvolvendo uma Abordagem Sistemática para Avaliação dos Estudos Experimentais em Search-Based Software Engineering ..... 49

*Márcio de O. Barros (Unirio); Arilo Claudio Dias Neto (UFAM)*

Uma Proposta de Framework de Apoio à Seleção de Tecnologias de Software aplicando Estratégias de Busca ..... 57

*Arilo Claudio Dias Neto, Rosiane de Freitas Rodrigues (UFAM)*

Utilizando Algoritmo de Otimização para Recomendação de Módulos em Projetos de Linhas de Produto de Software .....	65
--	----

*Thaís Alves Burity Pereira, Gledson Elias (UFPB)*

# Uma Avaliação do Uso de Diferentes Algoritmos Evolutivos Multiobjetivos para Integração de Classes e Aspectos

**Wesley Klewerton Guez Assunção<sup>1</sup>, Thelma Elita Colanzi<sup>1</sup>,  
Aurora Trinidad Ramirez Pozo<sup>1</sup>, Silvia Regina Vergilio<sup>1</sup>**

<sup>1</sup>Departamento de Informática, Programa de Pós-Graduação em Informática,  
Universidade Federal do Paraná (UFPR) – Curitiba, PR – Brasil

{wesleyk, thelmae, aurora, silvia}@inf.ufpr.br

**Resumo.** No teste de integração de sistemas orientados a aspectos é preciso determinar uma ordem de integração e teste de classes e aspectos de forma a reduzir os custos relativos à criação de *stubs*. Em estudos recentes os algoritmos NSGA-II e SPEA2 foram explorados. O presente trabalho investiga o uso do algoritmo PAES neste contexto e realiza uma comparação com os demais algoritmos evolutivos em um experimento com dois sistemas reais. Neste experimento, o PAES apresentou os melhores resultados.

**Abstract.** During the integration testing of aspect-oriented software it is necessary to determine a class and aspect integration and test order, which minimizes the *stubs* creation costs. Recently, the algorithms NSGA-II and SPEA2 have been explored. The present work investigates the use of the algorithm PAES in this context and a comparasion with the other evolutionary algorithms is accomplished by using two real systems. In the experiment, the algorithm PAES presented the best results.

## 1. Introdução

A fase de teste de integração de software orientado a aspectos (OA) envolve o teste das interações entre classes e entre classes e aspectos, e, para fazer isso, diferentes estratégias foram propostas nos últimos anos [Ceccato et al. 2005]. Esse problema, conhecido como CAITO (*Class and Aspect Integration and Test Order*) [Galvan et al. 2010], consiste em determinar uma ordem ótima para integrar e testar classes e aspectos de forma a minimizar os custos de criação de *stubs*. Diversos fatores que influenciam o processo de criação de *stubs* devem ser considerados para determinar tais ordens. Isso não é uma tarefa trivial, especialmente pela existência de ciclos de dependência entre módulos (classes ou aspectos). Um estudo recente mostra que é comum encontrar ciclos de dependência complexos em programas Java [Melton and Tempore 2007]. E, no contexto OA, [Ré and Masiero 2007] encontraram interesses transversais que são interdependentes, o que implica em dependência entre aspectos, e entre classes e aspectos.

No contexto de orientação a objetos (OO), a maioria das estratégias existentes para resolver o problema são baseadas em grafos [Abdurazik and Offutt 2006, Briand and Labiche 2003]. Estas estratégias foram estendidas recentemente para o contexto OA [Ré et al. 2007, Ré and Masiero 2007], entretanto, elas não são satisfatórias porque em muitos casos encontram soluções subótimas. Além disso, elas precisam de adaptação para considerar diferentes fatores que afetam a criação de *stubs*. Outra

limitação é que elas tentam reduzir o número de ciclos quebrados, sem considerar que há casos em que quebrar duas dependências é menos custoso do que quebrar somente uma.

A fim de reduzir tais limitações, no contexto OO, estratégias bio-inspiradas se mostraram mais promissoras. [Briand et al. 2002] exploraram o uso de algoritmos genéticos (AG) e uma função objetivo que agrupa duas medidas de acoplamento: número de métodos e de atributos necessários para criar *stubs*. [Cabral et al. 2010] investigaram uma solução baseada no algoritmo de colônia de formigas usando conceitos de Pareto [Pareto 1927] para tratar o problema como multiobjetivo. A abordagem multiobjetivo apresenta melhores soluções do que a função baseada em agregação de Briand et al., não precisa de ajuste de pesos e gera um conjunto de boas soluções para uso do testador. Em [Pozo et al. 2011] os autores avaliaram a abordagem multiobjetivo com três diferentes algoritmos, e o NSGA-II, que é um MOEA (*Multi-Objective Evolutionary Algorithm*), obteve os melhores resultados. Este algoritmo também foi usado com diferentes medidas de acoplamento [Assunção et al. 2011].

O uso de algoritmos evolutivos no contexto OA é recente. [Galvan et al. 2010] utilizaram um AG que apresentou melhores resultados que estratégias tradicionais baseadas em grafos, usando uma função de *fitness* que agrupa os objetivos a serem minimizados, mas similarmente para programas OO, o AG requer ajuste de pesos. Para superar esta limitação, em um trabalho anterior [Colanzi et al. 2011] dois MOEAs foram usados para resolver o problema CAITO: NSGA-II e SPEA2. Estes algoritmos alcançaram soluções viáveis considerando dois objetivos que impactam diretamente no custo de criação de *stubs*: número de atributos e número de métodos.

Motivados pelos resultados obtidos nos trabalhos anteriores realizados no contexto OO [Assunção et al. 2011, Pozo et al. 2011] e no contexto OA [Colanzi et al. 2011], e buscando melhorar ainda mais os resultados, este trabalho descreve um experimento de aplicação de um outro MOEA ao problema CAITO: o PAES (*Pareto Archived Evolution Strategy*) [Knowles and Corne 2000]. O PAES é um algoritmo multiobjetivo que usa uma estratégia de evolução um tanto diferente da estratégia adotada pelos outros dois MOEAs utilizados anteriormente. Saber qual algoritmo se comporta melhor para o problema é uma questão que só pode ser respondida por meio de experimentos. Por isto, são apresentados os resultados da aplicação dos três MOEAs a dois sistemas OA reais. O custo dos *stubs* é medido em relação a dois objetivos a serem minimizados, avaliados segundo duas medidas de acoplamento comumente usadas: número de métodos e número de atributos.

Este artigo está organizado em seções. Na Seção 2 apresenta-se o contexto de otimização e algoritmos multiobjetivos utilizados. Na Seção 3 descreve-se como foi realizada a aplicação dos MOEA ao problema CAITO. Na Seção 4 descreve-se o estudo experimental realizado, cujos resultados são apresentados e analisados na Seção 5. Finalmente, na Seção 6 são apresentadas as conclusões do trabalho.

## 2. Algoritmos Evolutivos Multiobjetivos

Problemas de otimização com duas ou mais funções objetivo são chamados multiobjetivos. Os objetivos costumam estar em conflito e, por isso, não há uma solução única. Assim, o objetivo é encontrar soluções com o melhor compromisso entre os objetivos. Estas soluções são chamadas de não-dominadas e formam a fronteira de Pareto [Pareto 1927].

Os algoritmos evolutivos multiobjetivos evoluem uma população de potenciais

soluções simultaneamente, obtendo um conjunto de soluções próximas à fronteira de Pareto em uma única execução do algoritmo. Existem alguns variantes dos AGs tradicionais adaptados para problemas multiobjetivo. Cada um adota estratégias distintas para evoluir e diversificar as soluções. Como mencionado, neste trabalho os seguintes algoritmos foram avaliados: NSGA-II, SPEA2 e PAES.

O NSGA-II (*Non-dominated Sorting Genetic Algorithm*) [Deb et al. 2002] ordena a população em várias fronteiras não-dominadas. A cada iteração ele utiliza as soluções não dominadas formando uma nova fronteira. As fronteiras representam a sua estratégia de elitismo. Além disso, este algoritmo usa um operador de diversidade (*crowding distance*) que ordena os indivíduos de acordo com a sua distância em relação aos vizinhos da fronteira para cada objetivo, visando a garantir maior espalhamento das soluções.

O algoritmo SPEA2 (*Strength Pareto Evolutionary Algorithm*) [Zitzler et al. 2001], além de ter sua população regular, mantém um arquivo externo que armazena soluções não-dominadas encontradas a cada geração. Para cada solução que está no arquivo e na população é calculado um valor de *strength*, que é usado no cálculo do *fitness* do indivíduo e durante o processo de seleção. O valor de *strength* de uma solução *i* corresponde a quantidade de indivíduos *j*, pertencentes ao arquivo e a população, dominados por *i*. O tamanho do arquivo é fixo e, caso este tamanho seja excedido, um algoritmo de agrupamento é usado a fim de reduzi-lo.

No processo evolutivo do algoritmo PAES [Knowles and Corne 2000] o conceito de população é diferente das estratégias tradicionais de AGs, pois apenas uma solução é mantida em cada geração, e a criação de novos indivíduos é feita somente através da aplicação do operador de mutação. Uma vez que o algoritmo trabalha com apenas uma solução por geração, não existe motivo para usar o operador de cruzamento. Assim como no SPEA2, existe um arquivo externo de soluções que é populado com as soluções não dominadas encontradas durante o processo. Uma solução filha é comparada com a solução pai e a solução dominante é mantida. E caso nenhuma das soluções domine a outra, a escolha da solução que vai permanecer no processo evolutivo é feita considerando a diversidade entre pai, filha e as soluções do conjunto externo. Caso o tamanho do arquivo externo exceda, é aplicada uma estratégia de diversidade (*crowded region*) para eliminar soluções similares nesse arquivo e manter a exploração de um espaço de busca maior. Portanto, a estratégia de evolução do PAES é bastante diferente das outras duas, o que pode levar a resultados diferentes na resolução do problema CAITO.

### **3. MOEAs aplicados ao problema CAITO**

[Ré et al. 2007] especificam as várias dependências que podem existir entre classes ou entre classes e aspectos. São elas: (I) herança, (U) dependência entre adendos e pontos de corte, (As) associação de entrecorte e (It) declaração intertipo. Além disso, [Ré and Masiero 2007] estudaram quatro estratégias de integração e teste de classes e aspectos: Combinada, Incremental+, Reversa e Aleatória. As duas primeiras provaram ser melhores por produzir um menor número de *stubs*. Entretanto, a estratégia Combinada parece ser mais prática uma vez que as classes e os aspectos são testados juntos, ao contrário da Incremental+ em que primeiro se testam as classes e depois os aspectos.

Para aplicar os MOEAs a fim de resolver o problema CAITO é preciso encontrar uma boa representação para o problema. Como o problema é relacionado à permutação de

módulos (classes e aspectos), os quais formam as ordens de teste, o cromossomo é representado por um vetor cujas posições contêm inteiros que representam os módulos. O tamanho do vetor é igual ao número de módulos de cada sistema e contém a ordem para integrar e testar os módulos. Esta representação é a mesma utilizada em [Colanzi et al. 2011].

Neste trabalho foram usadas duas funções baseadas em diferentes medidas de acoplamento comumente utilizadas na literatura [Briand et al. 2002, Cabral et al. 2010, Ré and Masiero 2007]: acoplamento de atributos e de métodos. Então, considerando-se que: (i)  $m_i$  e  $m_j$  são dois módulos acoplados, (ii) módulo são classes ou aspectos, e (iii) o termo operação representa métodos de classes, métodos e adendos de aspectos, essas medidas de acoplamento são definidas como segue: 1) **Acoplamento de atributos (A)**: Número de atributos localmente declarados em  $m_j$  quando referências ou apontadores para uma instância de  $m_j$  aparecem na lista de argumentos de algumas operações de  $m_i$ , como o tipo de seu valor de retorno, na lista de atributos de  $m_i$ , ou como parâmetros locais de operações de  $m_i$  (adaptado de [Briand et al. 2002]); 2) **Acoplamento de Operações (O)**: Número de operações (inclusive construtores) localmente declaradas em  $m_j$  que são invocadas por operações de  $m_i$  (adaptado de [Briand et al. 2002]).

Os dados de entrada de cada MOEA são formados por três matrizes que são lidas de um arquivo de texto e são: uma (1) matriz de dependência entre os módulos, e outras duas correspondentes a cada uma das métricas descritas acima; (2) matriz de acoplamento de atributos (métrica A) e (3) matriz de acoplamento de operações (métrica O). A matriz de dependência é representada por um vetor de duas dimensões que contém o tipo de dependência entre os módulos. Com base nessa matriz são definidas as restrições para cada um dos algoritmos. A restrição que deve ser considerada durante o processamento do algoritmo é não quebrar as dependências relacionadas a herança e a declaração intertipos. As métricas de acoplamento A e O são usadas para calcular o *fitness* de cada solução, sendo que a soma das dependências entre os módulos de cada métrica corresponde a um objetivo, e para reduzir o custo dos *stubs* deve-se minimizar todos os objetivos (métricas). A estratégia de integração e teste adotada neste trabalho é a Combinada [Ré et al. 2007].

#### 4. Estudo Experimental

O estudo experimental realizado compreende a aplicação dos MOEAs: NSGA-II, SPEA2, e PAES em sistemas OA desenvolvidos em AspectJ. Os MOEAs foram implementados usando o *framework* JMetal [Durillo et al. 2010]. Os dois sistemas OA usados são: AJHotDraw<sup>1</sup> e AJHSQLDB<sup>2</sup>, e a Tabela 1 contém algumas informações sobre eles.

**Tabela 1. Sistemas Utilizados**

Sistema	Versão	LOC	# Classes	# Aspectos	# Dependências						
					I	U	As	It	Pontos de Corte	Adendos	Total
AJHotDraw	0.4	18586	290	31	234	1177	140	40	0	1	1592
AJHSQLDB	18	68550	276	25	107	960	271	0	0	0	1338

Neste experimento os MOEAs foram configurados empiricamente, seguindo a mesma metodologia de [Colanzi et al. 2011]. Os valores de parâmetros utilizados para

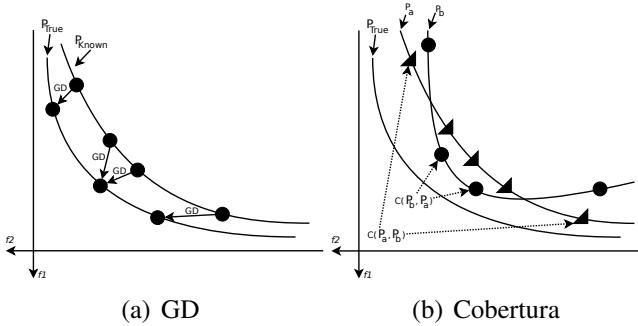
<sup>1</sup>Refatoração do JHotDraw, um *framework* para desenho bidimensional, disponível em: <http://sourceforge.net/projects/ajhotdraw/>

<sup>2</sup>Refatoração do HSQLDB, um gerenciador de banco de dados desenvolvido em Java, disponível em: <http://sourceforge.net/projects/ajhsqldb/files/>

o NSGA-II e o SPEA2 foram: tamanho da população = 300, taxa de mutação = 0,02 e taxa de cruzamento = 0,95. O PAES não recebe parâmetro de tamanho de população e não usa o operador de cruzamento. Para o PAES adotou-se a taxa de mutação = 1,0. No caso da taxa de mutação não era interessante adotar o mesmo valor utilizado para o NSGA-II e o SPEA2, já que estes dois algoritmos utilizam dois operadores evolutivos (cruzamento e mutação) e o PAES utiliza somente o operador de mutação. Tanto o PAES com o SPEA2 usaram um arquivo de tamanho = 250. Para os três MOEAs o número de avaliações de *fitness* foi 20000. Este valor foi utilizado como critério de parada e também como medida do custo computacional para os MOEAs.

A fim de conhecer o comportamento de cada algoritmo para o problema CAITO, foram executadas 30 rodadas independentes de cada MOEA para cada sistema. Em cada rodada, cada algoritmo encontra um conjunto aproximado de soluções ( $PF_{approx}$ ). Além disso, para cada MOEA, obtém-se o conjunto  $PF_{known}$  que é formado pelas soluções encontradas por cada algoritmo nas 30 rodadas, eliminando-se as repetidas e dominadas. Como neste caso, o conjunto Pareto real ( $PF_{true}$ ) não é conhecido, no experimento o  $PF_{true}$  foi obtido através da união de todos os  $PF_{approx}$  encontrados por cada MOEA, eliminando-se as soluções dominadas e repetidas [Zitzler et al. 2003]. Os conjuntos  $PF_{true}$  e  $PF_{known}$  foram utilizados na análise e comparação dos algoritmos, realizadas por meio do indicador de qualidade Distância Geracional(GD).

O indicador GD [Van Veldhuizen and Lamont 1999] é usado para calcular a distância entre um conjunto de Pareto encontrado, o Pareto Conhecido ( $PF_{known}$ ), em relação ao  $PF_{true}$ , ou seja, é uma medida de erro na qual verifica-se o quanto distante está um ponto encontrado do seu correspondente mais próximo no conjunto  $PF_{true}$ . O indicador GD é exemplificado na Figura 1(a). Além do GD foi utilizado o indicador de Cobertura (C) [Knowles and Corne 2000] para medir a dominância entre dois conjuntos. Comparando-se  $C(P_a, P_b)$  obtém-se um valor entre 0 e 1 referente a quanto o conjunto  $P_b$  é dominado pelo conjunto  $P_a$ , por outro lado, analisa-se  $C(P_b, P_a)$  para obter o quanto  $P_a$  é dominado por  $P_b$ . A Figura 1(b) apresenta um exemplo desse indicador, para um problema de minimização com dois objetivos.



**Figura 1. Indicadores de Qualidade**

## 5. Resultados e Análise

Através da análise dos resultados é possível verificar o comportamento do PAES quando comparado com o NSGA-II e SPEA2. A Tabela 2 apresenta na segunda coluna a cardinalidade de  $PF_{true}$  após todas as 30 execuções dos MOEAs, ou seja, após 90 execuções. Na terceira, quarta, e quinta colunas, tem-se a cardinalidade média, o desvio padrão de  $PF_{approx}$  e entre parênteses o total de diferentes soluções de  $PF_{approx}$  de cada algoritmo.

**Tabela 2. Número de soluções não dominadas**

Sistema	# PF <sub>true</sub>	NSGA-II	SPEA2	PAES
AJHotDraw	6	6,733 / 2,518 (9)	5,267 / 1,461 (11)	4,700 / 1,643 (6)
AJSQLDB	41	14,933 / 4,323 (22)	13,433 / 3,839 (21)	22,633 / 6,156 (41)

A análise do comportamento dos MOEAs foi feita utilizando os dois indicadores de qualidade GD e C. Para o indicador GD, os valores de média e desvio padrão calculados para as 30 execuções de cada MOEA são apresentados na Tabela 3. Os melhores valores são destacados em negrito. Para os dois sistemas o algoritmo PAES obteve a menor média de GD, ou seja, em cada execução o PAES encontra soluções mais próximas ao conjunto de melhores soluções. Para verificar a diferença estatística foi utilizado o teste de Friedman [Gárcia et al. 2009], que afirma com 95% de confiança que para o sistema AJHotDraw existe diferença entre o PAES em relação ao NSGA-II e SPEA2, confirmando o PAES como melhor. Já entre o NSGA-II e SPEA2 não existe diferença estatística. Para o sistema AJSQLDB existe diferença entre os três MOEAs. Considerando o AJSQLDB o PAES foi o melhor MOEA, seguido pelo NSGA-II e pelo SPEA2.

**Tabela 3. Média e Desvio Padrão do indicador GD**

Sistema	NSGA-II		SPEA2		PAES	
	Média	Desvio Padrão	Média	Desvio Padrão	Média	Desvio Padrão
AJHotDraw	1,104	1,285	1,416	2,131	<b>0,522</b>	<b>0,335</b>
AJSQLDB	0,395	0,119	0,542	0,136	<b>0,078</b>	<b>0,051</b>

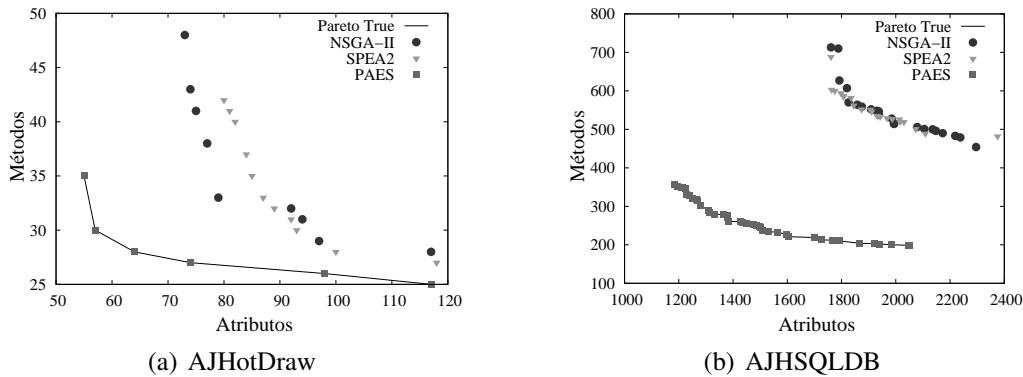
Os valores calculados para o indicador C são apresentados na Tabela 4. Para o cálculo foram utilizados os conjuntos  $PF_{known}$ . A leitura da tabela é feita por linha-coluna, na qual o MOEA que aparece na linha exerce determinada dominância sobre o MOEA que aparece na coluna. Para os dois sistemas as soluções do PAES dominam todas as soluções do NSGA-II e do SPEA2. Para o AJHotDraw as soluções do NSGA-II dominam mais de 50% as soluções do SPEA2. Por outro lado, no sistema AJSQLDB, as soluções do SPEA2 dominam mais de 60% as soluções do NSGA-II.

**Tabela 4. Análise de cobertura dos algoritmos**

Sistema	MOEA	NSGA-II	SPEA2	PAES	Sistema	MOEA	NSGA-II	SPEA2	PAES
AJHotDraw	NSGA-II	-	0,54	0	AJSQLDB	NSGA-II	-	0,23	0
	SPEA2	0,33	-	0		SPEA2	0,68	-	0
	PAES	1	1	-		PAES	1	1	-

Considerando a cobertura em cada uma das execuções, o teste pareado de Wilcoxon [Gárcia et al. 2009] apontou diferença estatística entre PAES e NSGA-II para os dois sistemas, e diferença entre PAES e SPEA2 para o sistema AJSQLDB. Nos outros casos não existe diferença estatística. A Figura 2 apresenta as soluções encontradas por cada MOEA no espaço de busca. É possível verificar que o PAES alcançou as melhores soluções. Suas soluções formaram o conjunto  $PF_{true}$ . Por outro lado as soluções do NSGA-II e SPEA2 estão localizadas nas mesmas áreas dos gráficos.

Em relação ao uso das soluções encontradas pelos MOEAs, considere o AJHotDraw e o custo de dois conjuntos ( $a$  e  $b$ ) de 3 soluções cada, correspondendo respectivamente às soluções do NSGA-II, SPEA2 e PAES:  $a=\{(A=79,O=33); (A=80,O=42); (A=57,O=30)\}$  e  $b=\{(A=77,O=38); (A=81,O=41); (A=64,O=28)\}$ . O testador deve escolher qual ordem utilizar de acordo com suas prioridades. As soluções  $a$  e  $b$  representam

**Figura 2. Espaço de Busca**

as melhores alternativas, pois elas têm o melhor compromisso entre os dois objetivos e estão mais próximas dos valores mínimos possíveis para cada objetivo. Mas, como este resultado pode ser utilizado pelo testador a fim de escolher entre *a* e *b*? Se a solução *a* do PAES fosse escolhida seria preciso criar *stubs* para simular dependências de 57 atributos e 30 operações. Por outro lado, se a solução *b* do mesmo algoritmo fosse escolhida seria preciso criar *stubs* para simular dependências de 64 atributos e 28 operações. Desta forma, no primeiro caso, o objetivo priorizado seria a medida A e, no segundo caso, a medida O. As soluções *a* e *b* do PAES têm o melhor *trade-off* entre os objetivos A e O.

## 6. Conclusão

Neste trabalho o problema CAITO foi resolvido utilizando MOEAs (NSGA-II, SPEA2 e PAES) visando a minimização de dois objetivos que representam fatores que influenciam na criação de *stubs* durante o teste de integração de classes e aspectos. Ao serem executados com dois sistemas OA reais, característicos do problema, os MOEAs se mostraram eficazes para resolver o problema em questão, pois conseguiram encontrar soluções de compromisso entre os objetivos em sistemas que contêm objetivos conflitantes.

O algoritmo PAES apresentou o melhor desempenho para os dois sistemas. Isso fornece evidências de que talvez a sua estratégia de evolução seja mais eficaz do que as estratégias adotadas pelos outros dois MOEAs quando existem objetivos muito conflitantes no sistema a ser testado. Apesar disso, como vários fatores podem influenciar o custo de construção de *stubs* durante a fase de teste de integração, é importante analisar o comportamento deste algoritmo quando se utilizam mais de dois objetivos. Neste contexto, pretende-se realizar novos experimentos utilizando esses três MOEAs com um número maior de objetivos, e, possivelmente utilizando outros sistemas, a fim de confirmar as evidências identificadas neste trabalho. Em nosso estudo foi utilizada a estratégia Combinada para testar classes e aspectos conjuntamente. No entanto, uma outra estratégia poderia ser usada e o testador poderia usar as soluções do MOEAs de outra maneira.

## Agradecimentos

Os autores agradecem ao apoio de Edison K. Fillus e ao CNPq e CAPES.

## Referências

- Abdurazik, A. and Offutt, J. (2006). Coupling-based class integration and test order. In *Proceedings of the International Workshop on Automation of Software Test*. ACM.

- Assunção, W., Colanzi, T., Pozo, A., and Vergilio, S. (2011). Establishing integration test orders of classes with several coupling measures. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*. to appear.
- Briand, L. C., Feng, J., and Labiche, Y. (2002). *Experimenting with Genetic Algorithms and Coupling Measures to Devise Optimal Integration Test Orders*. Carleton University, Technical Report SCE-02-03.
- Briand, L. C. and Labiche, Y. (2003). An investigation of graph-based class integration test order strategies. *IEEE Transactions on Software Engineering*, 29(7):594–607.
- Cabral, R., Pozo, A., and Vergilio, S. (2010). A Pareto Ant Colony Algorithm Applied to the Class Integration and Test Order Problem. In *Proceedings of the International Conference on Testing Software and Systems (ICTSS)*. Springer.
- Ceccato, M., Tonella, P., and Ricca, F. (2005). Is AOP code easier or harder to test than OOP code. In *Workshop on Testing Aspect-Oriented Program (WTAOP)*, Chicago, Illinois.
- Colanzi, T., Assunção, W., Vergilio, S., and Pozo, A. (2011). A multi-objective approach to reduce stubbing costs in the integration test of aspect-oriented software. In *Proceedings of the Latinamerican Workshop on Aspect Oriented Software (LA-WASP)*. to appear.
- Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197.
- Durillo, J., Nebro, A., and Alba, E. (2010). The jMetal framework for multi-objective optimization: Design and architecture. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*, pages 4138–4325.
- Galvan, R., Pozo, A., and Vergilio, S. (2010). Establishing Integration Test Orders for Aspect-Oriented Programs with an Evolutionary Strategy. In *Proceedings of the Latinamerican Workshop on Aspect Oriented Software (LA-WASP)*.
- García, S., Molina, D., Lozano, M., and Herrera, F. (2009). A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 Special Session on Real Parameter Optimization. *Journal of Heuristics*, 15(6):617–644.
- Knowles, J. D. and Corne, D. W. (2000). Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy. *Evolutionary Computation*, 8(2):149–172.
- Melton, H. and Temporo, E. (2007). An empirical study of cycles among classes in Java. *Empirical Software Engineering*, 12:389–415.
- Pareto, V. (1927). *Manuel D'Economie Politique*. Ams Press, Paris.
- Pozo, A., Bertoldi, G., Arias, J., Cabral, R., and Vergilio, S. (2011). Multi-objective optimization algorithms applied to the class integration and test order problem. *Software Tools for Technology Transfer*. submitted.
- Ré, R., Lemos, O. A. L., and Masiero, P. C. (2007). Minimizing stub creation during integration test of aspect-oriented programs. In *Proceedings of the Workshop on Testing Aspect-Oriented Programs (WTAOP)*, pages 1–6.
- Ré, R. and Masiero, P. C. (2007). Integration testing of aspect-oriented programs: a characterization study to evaluate how to minimize the number of stubs. In *Proceedings of the Brazilian Symposium on Software Engineering (SBES)*, pages 411–426.
- Van Veldhuizen, D. A. and Lamont, G. B. (1999). Multiobjective evolutionary algorithm test suites. In *Proceedings of the ACM Symposium on Applied Computing (SAC)*, pages 351–357.
- Zitzler, E., Laumanns, M., and Thiele, L. (2001). SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Technical report, Zurich, Switzerland.
- Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C. M., and da Fonseca, V. G. (2003). Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on Evolutionary Computation*, 7:117–132.

# Priorização de Casos de Teste Baseada em Sistemas de Inferência Fuzzy

**Thiago de A. C. Soares<sup>1</sup>, Werney A. L. Lira<sup>1</sup>, Ricardo de S. Britto<sup>1</sup>,  
Ricardo A. L. Rabêlo<sup>2</sup>, Pedro de A. dos Santos Neto<sup>1</sup>**

<sup>1</sup>EASII - DIE - UFPI - Teresina - PI - Brasil

<sup>2</sup>CEUT - Teresina - PI - Brasil

thiagoacs2@gmail.com, werney.zero@gmail.com, rbritto@ufpi.edu.br

ricardor.usp@yahoo.com.br, pasn@ufpi.edu.br

**Resumo.** *O Teste de software é uma atividade fundamental para obtenção de qualidade de um produto. No entanto, ele não é executado de forma apropriada por muitas organizações. É necessário que sua realização seja feita de forma sistemática e planejada. Este trabalho apresenta um sistema de inferência fuzzy que possibilita a priorização de casos de teste, de forma sistemática, baseada no uso de entradas relacionadas à volatilidade, complexidade e importância dos requisitos associados. O sistema de inferência desenvolvido permite que o testador especifique a estratégia de priorização por meio de regras linguísticas simples de se entender e projetar.*

**Abstract.** *The Software Testing is a fundamental activity related to product quality. However, it is not performed in suitable way for many organizations. It is necessary to execute testing in a systematic and planned way. This work presents a fuzzy inference system for test case prioritization, based on the use of inputs related to volatility, complexity and relevance of the requirements. The developed inference system allows the specification of a prioritization strategy, by the tester, based on linguistic rules in a simple and easy to understand way.*

## 1. Introdução

O Teste corresponde à verificação final do projeto e implementação de um software [Pressman 2006]. Mesmo tendo tal importância, ele não é executado de forma sistemática por muitas empresas, que normalmente deixam essa atividade apenas para o final do desenvolvimento. Devido a problemas no custo e no prazo estimados para o desenvolvimento, o teste é muitas vezes sacrificado ou executado de forma não planejada, sem o acompanhamento devido. Isto se reflete na qualidade dos produtos desenvolvidos.

Por conta dos fatores expostos, é importante que a comunidade ligada ao desenvolvimento de software proponha métodos, técnicas e ferramentas que auxiliem essa atividade, tornando-a menos onerosa, mais efetiva e mais fácil de ser utilizada pela indústria.

Um dos trabalhos diretamente ligados ao objetivo exposto é a priorização de casos de teste. Em alguns ambientes industriais, que utilizam testes como mecanismo de garantia da qualidade do produto, pode ser necessário priorizar os casos de teste existentes. Isto permite uma execução seletiva, a qual usa como base os casos de teste de maior

importância. Para isto, é necessário criar uma ordem de execução que otimize a característica mais importante para o teste em um determinado momento, seja esta a cobertura de código, o ataque aos casos de uso críticos ou até mesmo a execução em menor tempo e no menor custo. Criar essa ordem de execução dos casos de teste é um problema NP-Difícil [Cormen et al. 2002], não solucionável de forma eficiente com técnicas tradicionais [Harman 2001]. Desta forma, muitos pesquisadores direcionam seus esforços para aplicar técnicas de Inteligência Computacional para resolver o problema da priorização dos casos de teste ([Yoo and Harman 2007], [Maia et al. 2010b], [Maia et al. 2010a]).

Este trabalho apresenta uma abordagem baseada em sistemas de inferência *fuzzy* de Mamdani [Mamdani 1977] para resolver o problema da priorização de casos de teste baseada em requisitos ([Kavitha et al. 2010], [Maia et al. 2010b], [Srikanth et al. 2005]). Esta abordagem permite que um especialista na área de teste de software possa facilmente inserir seu conhecimento sobre o processo de priorização. Isso também permite ao sistema de inferência *fuzzy* ter comportamento similar ao que o especialista teria para priorizar os casos de teste.

O sistema de inferência desenvolvido neste trabalho estima o **Grau de Importância do Caso de Teste (GIT)**. O GIT é inferido a partir de entradas como os níveis de volatilidade e complexidade de cada requisito ligado ao caso de testes, além do nível de importância dos requisitos para o cliente. Uma vez gerado o GIT de cada caso de teste, basta ordenar os casos de teste com base nesse valor.

Este trabalho está organizado como se segue: na Seção 2 são apresentados alguns trabalhos relacionados; na Seção 3 é apresentada uma breve descrição sobre sistemas de inferência *fuzzy*; a Seção 4 contém o sistema de inferência *fuzzy* proposto; a Seção 5 discute o experimento realizado para validar a abordagem proposta; a Seção 6 apresenta a conclusão e os trabalhos futuros.

## 2. Trabalhos Relacionados

O trabalho desenvolvido por [Kavitha et al. 2010] apresenta uma abordagem para priorização de casos de teste baseada em requisitos. A abordagem utiliza três fatores para priorizar os casos de teste: o nível de prioridade dos requisitos para o cliente, complexidade de implementação dos requisitos e a volatilidade dos requisitos. Para cada requisito é calculado um fator chamado de RFV (*Requirement Factor Value*), que é a média dos valores dos três fatores para cada um dos requisitos do software. O RFV é utilizado para calcular o peso de cada caso de teste. O peso de um caso de teste é calculado por meio de uma fórmula que leva em consideração os RFV de todos os requisitos que ele cobre. Quanto maior for o peso de um caso de teste, maior deverá ser a prioridade dele.

No trabalho de [Srikanth et al. 2005] é proposta uma técnica para a priorização de casos de teste chamada de PORT (*Priorization of Requirements for Test*). Essa abordagem, baseada em requisitos de software, utiliza quatro fatores para efetuar a priorização de casos de teste: volatilidade dos requisitos, prioridade de uma requisito para o cliente, complexidade de implementação do requisito e a propensão a falha de um requisito.

Os autores do trabalho [Yoo and Harman 2007] apresentam uma abordagem para priorização de casos de teste baseada em cobertura de código, custo de execução e propensão a falha de um requisito. Para priorizar os casos de teste foram utilizadas as metaheurísticas NSGA-II e vNSGA-II.

Em [Maia et al. 2010b], os autores apresenta uma abordagem baseada em requisitos para priorizar casos de teste utilizando três variáveis: volatilidade, complexidade e importância de um requisito. Para priorizar os casos de teste com base nessas variáveis foram utilizadas as metaheurísticas NSGA-II, MoCell e SPEA2.

Os autores de [Maia et al. 2010a] propõem o uso da metaheurística GRASP para a resolução do problema da priorização de casos de teste baseada em cobertura de código.

A abordagem apresentada neste trabalho utiliza como base os trabalhos [Kavitha et al. 2010] e [Maia et al. 2010b]. Porém, ao invés de utilizar metaheurísticas para priorizar os casos de teste, a abordagem proposta utiliza um sistema de inferência *fuzzy*. Isso facilita a modelagem do conhecimento do especialista no sistema que irá efetuar a priorização dos casos de teste. Além disso, os resultados apresentados pelo sistema podem ser facilmente justificados, bastando-se para isso avaliar a base de regras do sistema *fuzzy*. Em abordagens que utilizam metaheurísticas, principalmente metaheurísticas estocásticas, é difícil explicar a origem de um resultado.

### **3. Sistemas de Inferência Fuzzy**

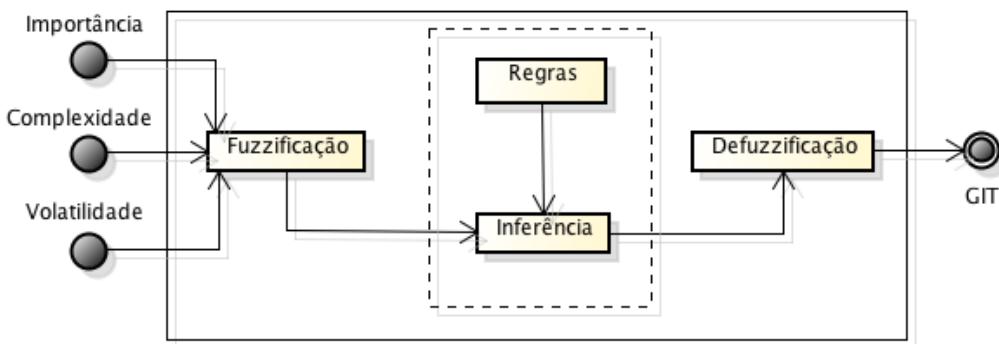
Os sistemas de inferência *fuzzy* se baseiam na teoria de conjuntos *fuzzy* [Zadeh 1965] e na lógica *fuzzy* [Zadeh 1996] para resolver problemas complexos, que normalmente possuem informações imprecisas, incertas e qualitativas. O funcionamento desse tipo de sistema se baseia em três etapas: fuzzificação, inferência e defuzzificação. A fuzzificação consiste no mapeamento de valores numéricos a conjuntos *fuzzy*. Usualmente, esses conjuntos são representados por termos linguísticos, como, por exemplo, "baixo", "alto" e "pouco". O procedimento de inferência *fuzzy*, a partir de uma base de regras de produção e dos valores de entrada fuzzificados, infere um valor de saída fuzzificado. A defuzzificação consiste no processo inverso à fuzzificação, no qual se associa um valor numérico ao valor *fuzzy* inferido a partir da base de regras e dos valores de entrada.

Este trabalho implementou um sistema de inferência *fuzzy* de Mamdani [Mamdani 1977]. As regras de um sistema de inferência de Mamdani possuem conjuntos *fuzzy* nos seus antecedentes e consequentes. Isto permite construir a base de regras do sistema de forma exclusivamente linguística, forma mais próxima da habilidade humana de tomar decisões racionais em um ambiente com imprecisões, incertezas e ruídos.

Uma vez que sistemas de inferência *fuzzy* permitem a manipulação do conhecimento intrínseco de forma qualitativa, os especialistas em teste de software podem mapar a sua experiência e o seu processo de tomada de decisão de forma linguística. Isso permite que a estratégia de ação/controle do sistema de inferência *fuzzy* se comporte de forma similar ao especialista em teste de software que forneceu seu conhecimento para construir esse sistema.

### **4. Sistema de Inferência Fuzzy Proposto**

Esta seção apresenta a abordagem proposta neste trabalho. Nas seções a seguir são apresentadas as entradas e saída do sistema *fuzzy* implementado, além da base de regras específica e do método de defuzzificação utilizado.



**Figura 1. Sistema de inferência Fuzzy proposto.**

#### 4.1. Entradas e Saída do Sistema de Inferência Fuzzy

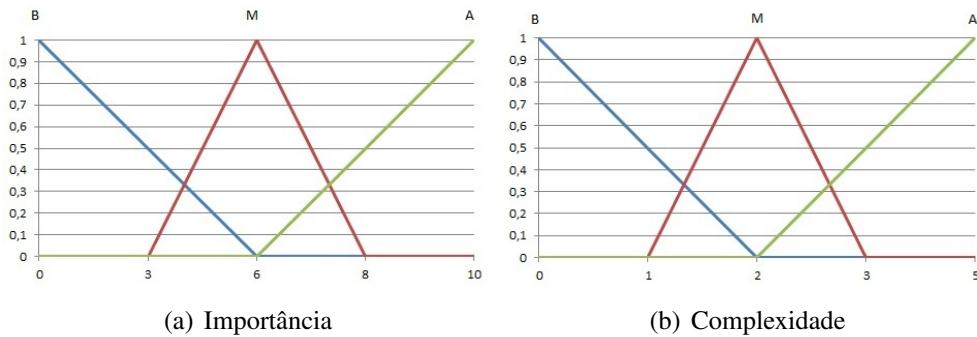
O sistema de inferência *fuzzy* de Mamdani proposto trabalha com três variáveis de entrada e uma de saída (Figura 1). As variáveis de entrada definidas foram: Importância, Complexidade e Volatilidade. A variável de saída é chamada de GIT (Grau de Importância do Caso de Teste). As variáveis de entrada foram definidas tomando como base o trabalho de [Maia et al. 2010b].

Uma vez que o modelo de Mamdani exige que os valores de entrada e saída sejam *fuzzy* e os respectivos valores a serem fornecidos e extraídos do sistema são numéricos, deve-se fuzzificar tais valores de entrada e posteriormente defuzzificar o valor de saída inferido. Cada variável de entrada e a variável de saída do sistema proposto possui conjuntos *fuzzy* mapeados. O mapeamento foi feito para conjuntos relevantes ao problema, conjuntos esses definidos de forma empírica.

O nível de importância reflete o quanto importante para o cliente é um requisito de software. O desejo do cliente deve ser levado em consideração quando se determina a ordem de execução dos requisitos do projeto [Heldman 2009], o que influencia diretamente a ordem de execução dos casos de teste. Os valores de entrada para a variável Importância estão dentro de uma escala inteira, que vai de 0 a 10. A fuzzificação dessa variável é feita por meio de três conjuntos *fuzzy*: Baixa (B), Média (M) e Alta (A). O mapeamento desses conjuntos *fuzzy* em relação às entradas numéricas pode ser visto na Figura 2.

O nível de complexidade de um requisito está relacionado ao quanto complexo será o desenvolvimento desse requisito, que normalmente implica em maior probabilidade de falhas no seu desenvolvimento. Os valores de entrada para a variável Complexidade estão dentro de uma escala inteira, que vai de 0 a 5. A fuzzificação dessa variável é feita por meio de três conjuntos *fuzzy*: Baixa (B); Média (M) e Alta (A). O mapeamento desses conjuntos *fuzzy* em relação às entradas numéricas pode ser visto na Figura 2.

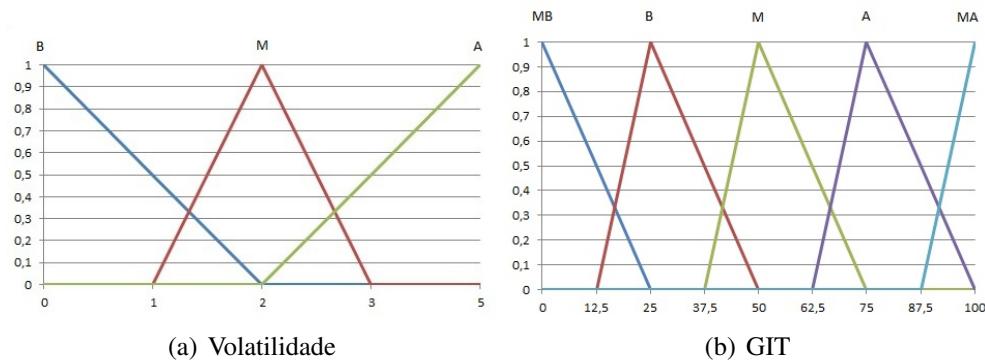
Na fase de análise de requisitos de um projeto, determinados requisitos podem ser difíceis de serem identificados. Uma análise de requisito deficiente pode acarretar diversas mudanças nos requisitos durante o projeto. O nível de volatilidade descreve a frequência com que determinado requisito é alterado. Quanto mais alterações são efetuadas em um requisito, maior é a probabilidade de serem inseridos defeitos nesse requisito. Desta forma, o sistema de inferência *fuzzy* proposto tem também como entrada a variável Volatilidade, que possui valores de entrada que podem variar em uma escala inteira que



**Figura 2. Conjuntos Fuzzy mapeados às variáveis de entrada Importância e Complexidade.**

vai de 0 a 5. A fuzzificação dessa variável é feita por meio de três conjuntos *fuzzy*: Baixo (B), Médio (M) e Alto (A). O mapeamento destes conjuntos *fuzzy* em relação às entradas numéricas pode ser visto na Figura 3.

A variável de saída GIT informa o quanto relevante para o projeto é caso de teste. O engenheiro de teste usará o GIT inferido pelo sistema proposto para cada caso de teste para determinar a ordem de execução dos testes. Quanto maior o GIT de um caso de teste, mais relevante para o projeto ele será. O GIT pode variar dentro de uma escala real que varia de 0,0 a 100,0. Essa variável é representada por cinco conjuntos *fuzzy*: Muito Baixo (MB), Baixo (B), Médio (M), Alto (A) e Muito Alto (MA). O mapeamento destes conjuntos *fuzzy* em relação às saídas numéricas pode ser visto na Figura 3.



**Figura 3. Conjuntos Fuzzy mapeados à variável de entrada volatilidade e à variável de saída GIT.**

#### 4.2. Base de regras Implementada

O GIT é inferido a partir dos valores de entrada fornecidos e da base de regras de produção do sistema de inferência *fuzzy*. A base de regras relaciona as entradas e saídas do sistema por meio de regras do tipo "se ... então". A base de regras implementada neste trabalho, na qual reside a estratégia para priorização dos casos de teste, possui 27 regras. A base de regras é apresentada de forma matricial, como pode ser visto na Tabela 1. Essa tabela relaciona as entradas do sistema e as saídas inferidas por ele. A leitura das regras a partir da matriz é exemplificada para a célula destacada em azul: se (Importância é A) e (Complexidade é A) então (GIT é MA).

**Tabela 1. Base de regras do sistema de inferência Fuzzy proposto.**

Volatilidade	Importância / Complexidade								
	A/A	A/M	A/B	M/A	M/M	M/B	B/A	B/M	B/B
A	MA	MA	A	MA	A	M	A	M	MB
M	MA	M	M	A	B	B	B	B	MB
B	A	A	M	B	B	MB	MB	MB	MB

#### 4.3. Método de Defuzzificação Implementado

O GIT inferido a partir das entradas deve ser defuzzificado para que tenhamos o valor quantitativo dentro da escala de 0,0 a 100,0, apresentada na Subseção 4.1. O método de defuzzificação implementado no sistema de inferência *fuzzy* proposto foi o método do centro de área [Oliveira Jr 2007]. É importante salientar que o sistema de inferência implementado permite a alteração do método de defuzzificação de forma simplificada.

#### 4.4. Discussão sobre o Sistema de Inferência Fuzzy Proposto

É comum em projetos de software o cliente modificar os requisitos ou até mesmo excluir alguns deles. Caso isso aconteça, o engenheiro de teste pode facilmente reordenar os casos de teste a serem executados tomando como base os GIT previamente calculados. Essa facilidade advém do fato da abordagem proposta avaliar os casos de teste individualmente. Os trabalhos relacionados discutidos anteriormente possuem como saída os casos de teste já completamente ordenados. Caso seja necessário excluir algum caso de teste, seria exigido um recálculo da ordenação.

A base de regras e os conjuntos *fuzzy* mapeados às variáveis de entrada e a variável de saída foram definidos de forma empírica, a partir do conhecimento dos autores relacionado a teste de software. A abordagem proposta permite que um especialista em teste altere facilmente a estratégia de priorização de casos de teste, bastando para isso alterar a base de regras de produção.

### 5. Resultados

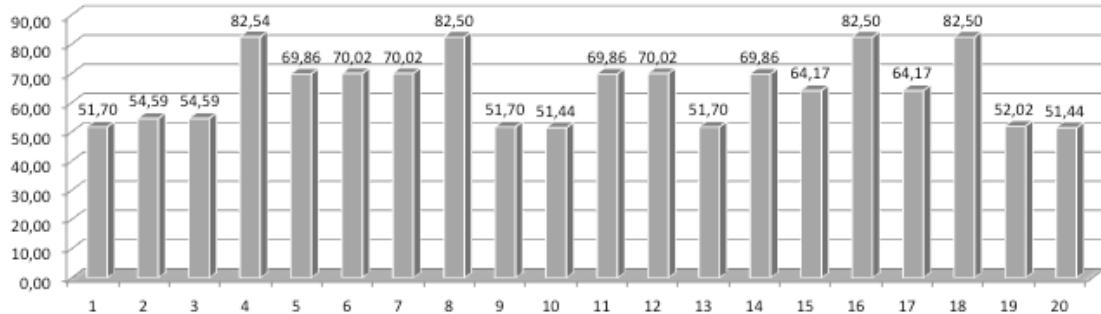
Para demonstrar a viabilidade da abordagem proposta neste trabalho, foi realizado um experimento no qual foram avaliados 20 casos de teste fictícios. Os valores de Importância, Complexidade e Volatilidade foram gerados de forma aleatória, respeitando as escalas numéricas especificadas para cada uma dessas variáveis de entrada. Esses valores podem ser vistos na Tabela 2.

**Tabela 2. Casos de teste avaliados.**

C. de Teste	Importância	Complexidade	Volatilidade	GIT	C. de Teste	Importância	Complexidade	Volatilidade	GIT
1	2	3	4	51,70	11	6	2	5	69,86
2	4	1	5	54,59	12	7	4	5	70,02
3	4	3	5	54,59	13	2	4	5	51,70
4	9	1	5	82,54	14	6	4	5	69,86
5	6	1	5	69,86	15	5	3	5	64,17
6	7	3	5	70,02	16	10	1	5	82,50
7	7	4	5	70,02	17	5	2	5	64,17
8	10	1	5	82,50	18	10	3	5	82,50
9	2	1	5	51,70	19	3	4	5	52,02
10	1	1	5	51,44	20	1	3	5	51,44

A partir dos valores fornecidos como entrada, o sistema de inferência *fuzzy* implementado inferiu os valores de GIT que podem ser vistos na Figura 4. O eixo X da

Figura 4 representa cada um dos 20 casos de teste avaliados. Já o eixo Y da mesma figura representa o GIT inferido pelo sistema *fuzzy* para cada um dos casos de teste.



**Figura 4. Valores de GIT inferidos pelo sistema para os 20 casos de teste do estudo.**

## 6. Conclusão e Trabalhos Futuros

Este trabalho apresenta uma abordagem de sistema de inferência *fuzzy* de Mamdani para resolução do problema de priorização de casos de teste em projetos de software. A abordagem apresentada se diferencia dos trabalhos relacionados citados na Seção 2 pelo fato de que o sistema de inferência *fuzzy* proposto permite que a estratégia de priorização seja facilmente especificada por meio de regras linguísticas. Além disso, o sistema avalia cada caso de teste de forma individual. Isso permite que caso haja necessidade de excluir algum requisito em função da vontade do cliente do projeto, o engenheiro de teste pode reordenar facilmente os casos de teste com base nos GIT pré-calculados. Na maioria dos trabalhos relacionados avaliados, as abordagens propostas entregam para o engenheiro de teste o conjunto de casos de teste completamente alocados. Qualquer mudança necessária exige um recálculo.

Uma vez que tal trabalho ainda está em andamento, alguns elementos no atual estágio do trabalho não garantem a otimalidade dos resultados inferidos. Podemos destacar que o experimento realizado não utilizou dados reais, apenas dados gerados aleatoriamente. Outro elemento que não permite garantir a otimalidade dos resultados é o fato de que as funções de pertinência dos conjuntos *fuzzy* e a base de regras terem sido especificadas de forma puramente empírica. Não menos importante é o fato de não terem sido utilizadas métricas estabelecidas na comunidade de Engenharia de Software para especificar as variáveis de entrada Importância, Complexidade e Volatilidade.

Apesar dos elementos apresentados no parágrafo anterior, pode-se constatar, de forma preliminar, que a abordagem apresentada é mais intuitiva e adequada para o uso por engenheiros de teste do que outras abordagens existentes também baseadas em Inteligência Computacional. A facilidade no uso da abordagem proposta advém do fato que o processo de construção da base de regras do sistema de inferência *fuzzy*, etapa na qual se determina a estratégia desejada para priorizar os casos de teste, está mais próxima da forma de pensar de um engenheiro de teste do que as outras abordagens baseadas fortemente em fórmulas matemáticas.

Alguns trabalhos que visam a evolução da abordagem apresentada estão em andamento, dos quais podemos citar:

- A abordagem proposta está sendo utilizada em um projeto real de uma fábrica de software.
- Estão sendo realizados estudos que visam a aplicação de técnicas de Inteligência Computacional para otimizar a base de regras e as funções de pertinência dos conjuntos fuzzy do sistema de inferência especificados.
- Estudos estão sendo realizados visando a formalização de uma métrica para a aferição dos níveis de importância dos requisitos para os clientes. Essa métrica a ser desenvolvida irá se apoiar no uso de SQFD (*Software Quality Function Deployment*) [Haag et al. 1996], uma vez que essa técnica é bastante adequada para o problema em questão.

## Referências

- Cormen, T. H., Leiserson, C. E., Stein, C., and Rivest, R. L. (2002). *Algoritmos: Teoria e Prática*. Campus.
- Haag, S., Raja, M. K., and Schkade, L. L. (1996). Quality function deployment usage in software development. *Commun. ACM*, 39:41–49.
- Harman, M. (2001). Search-based software engineering. *Information and Software Technology*, pages 833–839.
- Heldman, K. (2009). *Gerência de Projetos*. Elsevier.
- Kavitha, R., Kavitha, V., and Kumar, N. (2010). Requirement based test case prioritization. In *Communication Control and Computing Technologies (ICCCCT), 2010 IEEE International Conference on*.
- Maia, C. L. B., Carmo, R. A. F., Freitas, F. G., Campos, G. A. L., and Souza, J. T. (2010a). Automated test case prioritization with reactive grasp. *Advances in Software Engineering - Special Issue on Software Test Automation*.
- Maia, C. L. B., Freitas, F. G., and Souza, J. T. (2010b). Applying search-based techniques for requirements-based test case prioritization. In *Anais do I Workshop Brasileiro de Otimização em Engenharia de Software*.
- Mamdani, E. H. (1977). Application of Fuzzy Logic to Approximate Reasoning Using Linguistic Synthesis. *IEEE Transactions on Computers*, 26(12):1182–1191.
- Oliveira Jr, H. A. (2007). *Inteligência Computacional Aplicada à Administração, Economia e Engenharia em Matlab*. Thomson.
- Pressman, K. (2006). *Engenharia de Software*. McGraw-Hill.
- Srikanth, H., Williams, L., and Osbome, J. (2005). System test case prioritization of new and regression test cases. In *International Symposium on Empirical Software Engineering*.
- Yoo, S. and Harman, M. (2007). Pareto efficient multi-objective test case selection. In *Proceedings of the International Symposium on Software Testing and Analysis*.
- Zadeh, L. (1965). Fuzzy Sets\*. *Information and control*, 8(3):338–353.
- Zadeh, L. A. (1996). Fuzzy logic = computing with words. *IEEE Transactions on Fuzzy Systems*, 4(2).

# Abordagem Híbrida para o Problema Multiobjetivo do Próximo Release

Thiago Gomes Nepomuceno da Silva<sup>1</sup>, Fabrício Gomes de Freitas<sup>1</sup>,  
Jerffeson Teixeira de Souza<sup>1</sup>

<sup>1</sup>Grupo de Otimização em Engenharia de Software da UECE –  
Universidade Estadual do Ceará (UECE)  
Avenida Paranjana, 1700 – Fortaleza – CE - Brasil

{thi.nepo,fabriogf.uece}@gmail.com, jeff@larces.uece.br

**Abstract.** *The use of metaheuristics techniques has been very efficient in many areas. Such methods do not guarantee the optimality of their solutions. In this article we discuss a hybrid technique that uses solutions generated from an exact method to improve the search metaheuristic. The technique is tested on the multiobjective requirements selection problem.*

**Resumo.** *O uso de técnicas metaheurísticas vem se mostrando muito eficiente em várias áreas. Tais métodos não garantem a otimalidade de suas soluções. Nesse artigo abordamos uma técnica híbrida que usa soluções geradas a partir de uma técnica exata para melhorar a busca da metaheurística. A técnica é testada no problema multiobjetivo da seleção de requisitos.*

## 1. Introdução

No contexto de SBSE, o uso de metaheurísticas se destaca como técnica principal para resolução de problemas (Harman, 2007), incluindo teste de software (Freitas et al., 2010), planejamento de releases (Brasil et al., 2010), entre outros. Outras formas de resolução, como otimização exata, apesar da vantagem da melhor solução, ainda não são amplamente usadas.

Nesse contexto, esse artigo propõe uma nova abordagem, com a união das duas técnicas, isto é, uma abordagem híbrida. A técnica híbrida foi desenvolvida utilizando o algoritmo exato Branch-and-Bound com a metaheurística NSGA-II, mas que pode ser estendida às demais técnicas exatas ou metaheurísticas. O algoritmo Branch-and-Bound poderia ser usado para encontrar a frente real do problema abordado, mas tal operação depende de um alto número de execuções, o que seria custoso. A ideia principal da abordagem híbrida é encontrar algumas soluções exatas da frente real utilizando o Branch-and-Bound e adicionar essas soluções na população inicial da metaheurística, com o intuito de auxiliar sua busca.

## 2. Trabalhos Relacionados

Entre as abordagens híbridas para a resolução de problemas de otimização já propostas na literatura, podem ser citadas aquelas onde as técnicas executam de forma colaborativa (Gallardo et al., 2008). Isso acontece ao usar resultados parciais de um método como entrada para a execução do outro. Após algum critério, o segundo método é parado, continuando a busca com o primeiro. Tal processo foi usado em (Gallardo et

al., 2008), onde a busca é alternada entre Algoritmos Genéticos (GA) e Branch-and-Bound. No caso, a ligação entre os métodos é que a representação do cromossomo no GA é feita de acordo com o estado de cada variável binária no Branch-and-Bound. A variável tem valor 0 ou 1 dependendo se já sofreu branching ou não.

No campo específico de Engenharia de Software baseada em Busca, o trabalho (Kapur et al., 2008) apresenta um framework para a resolução do problema de alocação de funcionários no contexto do planejamento de releases. A técnica consiste de duas fases. Na primeira fase uma técnica de otimização exata é utilizada para encontrar uma solução relaxada do problema, ou seja, uma solução sem a restrição de integralidade. Tal solução é usada como entrada para a busca por um GA, que usando a solução relaxada fornecida na fase anterior deve fazer a busca em um espaço de soluções reduzido. Os autores realizam 200 experimentos para mostrar a eficácia do framework. No caso, resultados da abordagem híbrida são comparados com resultados onde apenas o GA foi usado. É reportado que, em média, os resultados da busca híbrida foram 15% melhores que o da abordagem apenas com GA.

### **3. Abordagem Proposta**

O método híbrido proposto neste trabalho faz uso de uma metaheurística multiobjetiva e de um método de otimização exata. A seguir são apresentados os algoritmos escolhidos, a saber: NSGA-II e Branch-and-Bound, respectivamente. Em seguida, a abordagem propriamente dita é descrita.

#### **3.1. Metaheurística Multiobjetiva - NSGA-II**

A metaheurística NSGA-II (*Non-dominated Sorting Genetic Algorithms II*) (DEB et al., 2002) é uma técnica de otimização multiobjetiva que utiliza os conceitos de permutação e mutação. O NSGA-II inicia com a geração de forma aleatória de um conjunto  $P_0$  de N soluções. Em seguida, um segundo conjunto  $Q_0$ , também de tamanho N, é gerado com o uso dos operadores de crossover e mutação. As soluções dos dois conjuntos formam a população  $R_0$  de tamanho  $2N$ . Então,  $R_0$  é ordenado usando o conceito de dominância da seguinte maneira: as soluções que não são dominadas por nenhuma outra são colocadas na primeira Frente de Pareto, F1; as soluções que são dominadas por uma solução são colocadas na segunda Frente de Pareto, F2; e assim sucessivamente até que todas as soluções de  $R_0$  estejam em alguma Frente. Então, uma população  $P_1$  é formada considerando as Frentes de Pareto iniciais até o limite de N soluções. Quando este limite impede que todas as soluções de uma Frente sejam tomadas, um operador de distância de soluções (*crowding distance sorting*) é utilizado na seleção de quais serão tomadas para  $P_1$ , para selecionar soluções mais distantes entre si e permitir maior nível de diversificação na população. A população  $P_1$  é utilizada como ponto inicial na iteração seguinte, ou seja, uma população será formada, e as soluções das duas populações serão ordenadas de acordo com a dominância para a seleção de quais participarão da próxima geração.

#### **3.2. Otimização Exata - Branch-and-Bound**

As técnicas exatas são métodos que utilizam operações matemáticas sobre os dados do problema. O método mais conhecido no caso de problemas com funções e restrições

lineares é o método Simplex. Contudo, o problema atacado nesse artigo possui a restrição de variáveis binárias, pois a solução é a seleção de requisitos. Assim, no presente trabalho, utilizamos o método *Branch-and-Bound*. Tal abordagem inicialmente resolve o problema utilizando métodos lineares ou não-lineares (dependendo do problema tratado) sem a restrição de solução inteira com o intuito de encontrar os limites (*bounds*) da solução. Além disso, o problema é dividido em subproblemas em estrutura de árvore (*branch*) de acordo com divisões dos domínios das variáveis, e as divisões que se apresentam piores que o limite são descartadas. O processo é realizado até que os subproblemas tenham sido considerados.

### 3.3. Abordagem Híbrida

O termo híbrido em otimização é encontrado na literatura também em relação a métodos que usam duas ou mais metaheurísticas. Contudo, como citado, nesse trabalho propomos uma abordagem com uma metaheurística e uma técnica de otimização exata. Tal tipo de abordagem híbrida pode ser classificada em acoplamento forte ou fraco. No caso, nosso método utiliza um acoplamento fraco. Além disso, nossa abordagem é classificada como uma estratégia sequencial, pois um algoritmo só é executado após o término do outro, e não simultaneamente.

O desempenho de metaheurísticas é fortemente influenciado pela população inicial, pois dependendo do quão relevante é a população inicial gerada para o problema abordado, o algoritmo terá um comportamento bom ou ruim, convergindo rapidamente ou não. Por isso resolvemos atacar essa fraqueza, melhorando a população inicial. A estratégia então consiste em incluir elementos da frente exata como elementos da população inicial da metaheurística. Assim, tais elementos serviriam para que a metaheurística realizasse o processo de busca se beneficiando de soluções ótimas pertencentes à frente de Pareto real.

O processo é realizado de forma que a metaheurística gere as soluções “intermediárias” entre tais soluções ótimas. Por exemplo, um crossover entre dois indivíduos, sendo um deles uma das soluções exatas, gerará uma solução de qualidade, já que tal operação gera soluções similares às soluções parentais. De fato, como teremos indivíduos da frente exata na população inicial da metaheurística a expectativa é que nas primeiras gerações já sejam encontradas boas soluções, de forma que o algoritmo consiga convergir mais rapidamente.

Dessa forma, a abordagem considera vantagens das duas técnicas, ao permitir rápida solução dos problemas pela metaheurística, e usar soluções ótimas como parte do processo, a partir da técnica exata. A abordagem híbrida apresentada é especialmente útil em contextos em que seja difícil gerar uma “boa” população inicial, isto é com soluções com bons valores de função-objetivo e que sejam válidas, ou seja, respeitem as restrições do problema. Tal fato pode ocorrer por diversos motivos, como: grande espaço de possíveis soluções, restrições fortes, ou mesmo quando a quantidade de soluções viáveis é pequena, o que dificulta a geração das mesmas da forma original pela metaheurística (de forma aleatória).

É valido destacar que os algoritmos utilizados (NSGA-II e Branch-and-Bound) foram escolhidos tendo em vista o amplo uso dos mesmos na literatura de otimização. Contudo, a abordagem em si pode fazer uso de outras técnicas de cada tipo.

## 4. Projeto de Avaliação

Nessa seção apresentamos a definição formal do problema tratado, onde a formulação matemática do mesmo é explicada. Em seguida, os parâmetros técnicos dos métodos e dos experimentos são descritos.

### 4.1. Multiobjective Next Release Problem

A formulação do problema é multiobjetiva, dado que o problema possui mais de um aspecto a ser otimizado simultaneamente. A multiobjetividade acontece com a otimização tanto da importância dos requisitos quanto do tempo de implementação (Zhang et al., 2007). Na formulação multiobjetiva, a importância considerada no processo tem tanto o fator do cliente como também a importância que cada cliente associa a cada requisito.

Os aspectos que definem o problema são:

- **Clientes:** Seja o conjunto  $C = \{c_1, c_2, \dots, c_m\}$  formado por  $m$  clientes. Cada cliente  $c_j$  possui um valor de importância  $importance_j$  para a empresa. Tal valor de importância representa o nível de estima que aquele cliente representa para a empresa, sendo que valores maiores significam maior importância.
- **Requisitos:** O conjunto  $R = \{r_1, r_2, \dots, r_n\}$  é formado por  $n$  requisitos. Cada requisito  $r_i$  tem um custo de implementação  $custo_i$ , que pode ser estimado em homens-hora.
- **Clientes e Requisitos:** Esse aspecto trata da relação que contém a visão de cada cliente para cada requisito desejado. Nesse sentido, o cliente não apenas indica quais funcionalidades são desejadas, mas também um nível de importância para cada uma. O valor é definido como  $value_{i,j}$ .

A implementação de um requisito representa um valor geral de valor agregado à empresa. Tal valor é formado pela consideração da importância de cada cliente, e do peso dado para cada requisito. Tal valor final de importância do requisito é denominado *score* na formulação, e é definido como:

$$score_i = \sum_{j=1}^m importance_j * value_{j,i}$$

A formulação matemática do problema é apresentada abaixo. A variável  $X_i$  é um valor booleano que indica a seleção ou não do requisito  $i$ :

$$\max \sum_{i=1}^n score_i * X_i$$

$$\min \sum_{i=1}^n custo_i * X_i$$

## 4.2. Metodologia

Como citado anteriormente, a abordagem híbrida funciona com a inclusão de soluções pertencentes à frente de Pareto ótima na população inicial da metaheurística. Após testes preliminares, verificamos que considerar os pontos que dividem a frente em 10 regiões equidistantes entre si se mostrou aplicável. Isso significa que a população inicial do NSGA-II possui 11 indivíduos da frente real. Tais 10 indivíduos são aqueles que delimitam 10 regiões da frente (para a divisão em 10 partes são necessários 11 pontos). Por definição, os extremos da frente de Pareto estão incluídos entre tais soluções.

Para gerar os pontos da frente exata usando o Branch-and-Bound, a função de importância foi mantida como função-objetivo, enquanto a função de custo foi considerada como uma restrição. No caso, foram realizadas 11 execuções da técnica exata, uma para cada ponto desejado da frente de Pareto.

Como parâmetros técnicos, as seguintes configurações foram utilizadas: tamanho da população de 250, taxa de crossover 0.9, tipo de crossover SinglePointCrossover, taxa de mutação 1/(número de requisitos), e quantidade de avaliações de 250000.

## 5. Resultados

Para realizar avaliação preliminar da abordagem, foram realizados testes com 2 instâncias do problema Multiobjective Next Release Problem (MONRP). A primeira instância (A) possui 100 clientes e 100 requisitos, enquanto a segunda (B) possui 200 clientes e 400 requisitos, além da modificação para incorporar dependência entre requisitos. Assim, os experimentos cobriram dois cenários com tamanhos diferentes, além do contexto da restrição de dependência, onde um requisito só pode ser implementado se seu precedente tiver sido implementado.

As instâncias foram geradas de forma aleatória. O ideal seria usar dados reais, mas o uso de valores aleatórios serve nesse estudo inicial como forma de validar a aplicabilidade técnica da abordagem. A frente real foi gerada pelo Branch-and-Bound.

Inicialmente apresentamos resultados de métricas de desempenho das metaheurísticas multiobjetivas nas Tabelas 1 e 2 a seguir.

Os resultados apresentados são a média de 10 execuções e o tempo está em milissegundo.

**Tabela 1. Métricas do NSGA-II para diferentes gerações.**

Métricas \ Avaliações	500	2500	25000	250000
<b>HV</b>	0.27719780	0.33056193	0.45752635	0.51139839
<b>Spread</b>	0.99295697	0.93634371	0.91069729	0.66454421
<b>Tempo</b>	315	1710	17595	190925

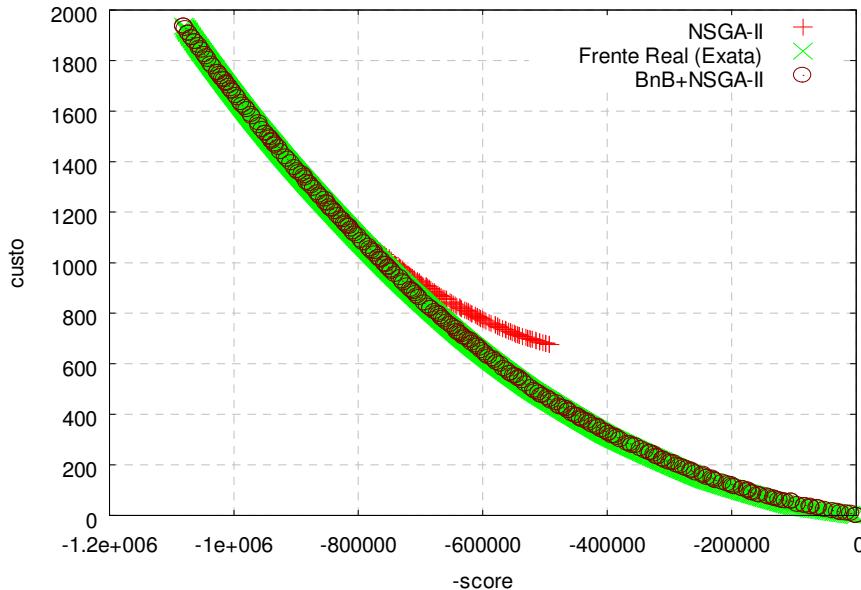
**Tabela 2. Métricas do Branch-and-Bound+NSGA-II para diferentes gerações.**

Métricas \ Avaliações	500	2500	25000	250000
<b>HV</b>	0.56505097	0.62593065	0.63898702	0.64272736
<b>Spread</b>	0.83747254	0.84831489	0.36736531	0.38680125
<b>Tempo</b>	407	2102	17753	170909

A partir das Tabelas acima, nota-se que os valores das métricas são melhores na abordagem híbrida. No caso, os valores de HV (“hypervolume”) são maiores na técnica híbrida, o que indica maior cobertura da frente de Pareto real. Os valores da métrica Spread da abordagem híbrida são menores que os apresentados pelo NSGA-II puro, indicando que as soluções encontradas são mais eqüidistantes, o que é o desejado.

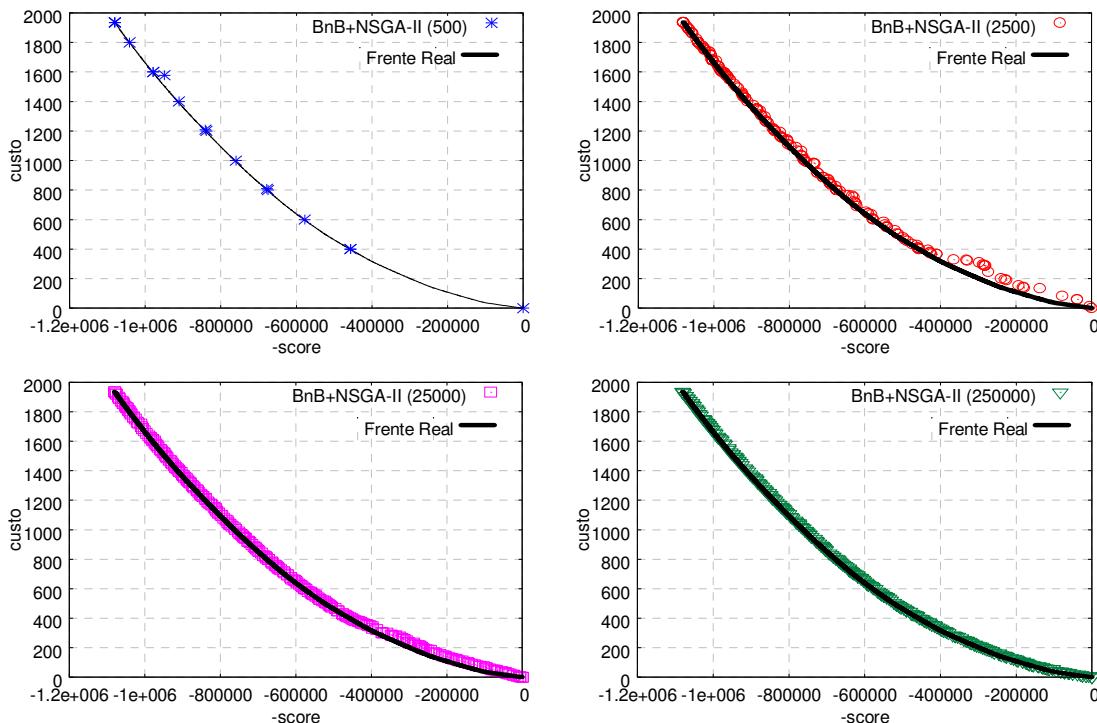
Percebe-se que a diferença de tempo de execução é cada vez menor com o aumento da quantidade de execuções no algoritmo. Nossa hipótese para esse fato é que o NSGA-II deve convergir mais rápido quando possui soluções ótimas na sua população inicial, chegando, inclusive, há ter um tempo menor em relação a sua forma pura quando analisado em 250.000 avaliações.

Apresentamos na Figura 1 a seguir os resultados gráficos para a instância B, com o intuito de ilustrar o desempenho da abordagem. Os resultados para a instância A foram graficamente similares.

**Figura 1. Comparação entre frentes geradas pelas técnicas.**

A Figura 1 ilustra a melhoria da abordagem híbrida em relação ao NSGA-II puro. Percebe-se que a frente da metaheurística pura não foi capaz de encontrar soluções em todo o domínio do eixo dos custos. O uso das soluções exatas (da frente real) na abordagem híbrida, como mostrado na figura, permitiram que a abordagem encontrasse soluções em todo o domínio.

No gráfico abaixo podemos ver a mesma evolução para o algoritmo BnB+NSGAII. Vemos que a frente vai evoluindo em cima de frente real, mostrada como a linha presente nos gráficos, incialmente pouco povoada e com a execução do algoritmo vai sendo preenchida. No final das 250000 avaliações a frente de Pareto resultante do BnB+NSGAII se confunde com a frente real para essa instância. Para outras instâncias testadas o mesmo acontece.



**Figura 2. Frente da abordagem híbrida em diferentes pontos da busca (500, 2500, 25000, 250000).**

## 6. Conclusões

Pode-se verificar que o algoritmo proposto é válido e possui um ótimo desempenho. Mas é necessário testes com problemas mais complexos, que possuam restrições mais pesadas. Onde a geração de uma boa população inicial de forma aleatória seja difícil.

No quesito tempo, vemos um algoritmo com tempo intermediário entre exata e metaheurística, mas quando analisada a boa qualidade das soluções, vemos um algoritmo tão bom quanto o método exato. No problema abordado nesse trabalho vemos uma solução muito semelhante à exata com a metade do tempo da mesma.

Como trabalho futuro pode ser destacado o aprofundamento do estudo dessa abordagem, principalmente em problemas mais complexos, onde a geração de uma população inicial válida seja mais difícil. Em problemas desse tipo, espera-se que o desempenho da abordagem híbrida seja ainda melhor em relação ao das metaheurísticas.

## Referências

- Brasil, M. M. A., Freitas, F. G., Silva, T. G. N., Souza, J. T., Cortés, M. I., (2010) Uma Nova Abordagem de Otimização Multiobjetiva para o Planejamento de Releases em Desenvolvimento Iterativo e Incremental de Software, In: I Workshop Brasileiro de Otimização em Engenharia de Software (WOES 2010), 2010.
- Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2), 182-197. IEEE.
- Gallardo, J. E., Cotta, C. and Fernández, A. J. (2008) Exact, Metaheuristic, and Hybrid Approaches to Multidimensional Knapsack Problems, in Optimization Techniques for Solving Complex Problems (eds E. Alba, C. Blum, P. Isasi, C. León and J. A. Gómez), John Wiley & Sons, Inc., Hoboken, NJ, USA.
- Freitas, F. G., Maia, C. L. B., Campos, G. A. L., Souza, J. T., (2010) Otimização em Teste de Software com Aplicação de Metaheurísticas, Revista Sistemas de Informação, Edição 5, 2010.
- Harman, M (2007) The current state and future of search based software engineering. In: Briand, LC and Wolf, AL, (eds.) FOSE 2007: FUTURE OF SOFTWARE ENGINEERING. (pp. 342 - 357). IEEE.
- Kapur, P., Ngo-The, A., Ruhe, G. and Smith, A. (2008), Optimized staffing for product releases and its application at Chartwell Technology. *Journal of Software Maintenance and Evolution: Research and Practice*, 20: 365–386.
- Zhang, YY and Harman, M and Mansouri, SA (2007) The Multi-Objective Next Release Problem. In: GECCO 2007: Genetic and Evolutionary Computation Conference, VOL 1 AND 2. (pp. 1129 - 1136). ACM.

# Uma Abordagem Baseada em Algoritmo Genético para Alocação de Equipes Distribuídas

Bruno Ribeiro, Gledson Elias

COMPOSE – Component Oriented Software Engineering Group  
Departamento de Informática – Universidade Federal da Paraíba

bruno@compose.ufpb.br, gledson@di.ufpb.br

**Abstract.** *In order to achieve the proclaimed benefits of Global Software Development (GSD), many organizations adopt GSD processes, allocating implementation tasks among globally distributed teams. However, cultural and geographical distances among teams have direct impact on communication effectiveness, which in turn affects projects' progress and success. In order to mitigate such communication issues, this paper proposes a genetic algorithm based approach for allocating distributed development teams to implementation tasks of software modules, taking into account both non-technical features of teams and dependences among software modules.*

**Resumo.** *Para alcançar os benefícios proclamados do Desenvolvimento Distribuído de Software, muitas organizações adotam processos de DDS, alocando atividades de implementação a equipes globalmente distribuídas. Entretanto, as distâncias culturais e geográficas das equipes geram uma série de problemas que impactam diretamente na efetividade da comunicação, que, por sua vez, afeta o progresso e o sucesso dos projetos. Com o objetivo de mitigar tais problemas de comunicação, este artigo apresenta uma abordagem baseada em algoritmo genético para alocação de equipes de desenvolvimento distribuídas a tarefas de implementação de módulos de software, considerando tanto aspectos não-técnicos das equipes quanto informações de dependência entre módulos de software.*

## 1. Introdução

Abordagens de Desenvolvimento Distribuído de Software (DDS) proclamam como principais benefícios a redução do tempo de lançamento de produtos (*time-to-market*), o aumento da qualidade do software, a diminuição dos custos, e a oferta de mão de obra qualificada (Herbsleb e Grinter, 1999) (Audy e Prikladnicki, 2007). Como consequência, projetos e processos de DDS têm sido amplamente adotados pela indústria de software, e, deste modo, muitas organizações que antes praticavam o desenvolvimento tradicional, com equipes co-localizadas, passaram a explorar o potencial mercado global, disseminando ou estabelecendo unidades colaborativas em diversas regiões do globo.

No entanto, a adoção de DDS requer cuidados adicionais nos aspectos relacionados ao gerenciamento de pessoas, pois, de forma análoga aos projetos co-localizados, tais aspectos são determinantes da qualidade e do sucesso de um projeto (ABNT, 2000). É importante destacar que, em abordagens de DDS, o gerenciamento de pessoas torna-se ainda mais crítico devido às diferenças geográficas, temporais e culturais entre os membros das equipes (Audy e Prikladnicki, 2007) (Herbsleb e Mockus, 2003) (Gumm, 2006) (Huzita, Silva, et al., 2008).

Como parte do gerenciamento de pessoas, o processo de alocação da equipe deve selecionar e nomear profissionais com competências apropriadas às necessidades do projeto (ABNT, 2000). Logo, processos de alocação tradicionais levam em consideração essencialmente os aspectos técnicos, como conhecimento, habilidade e experiência (Duggan, Byrne e Lyons, 2004) (Silva, 2007) (Callegari, Foliatti e Bastos, 2009). No entanto, os aspectos não-técnicos são reduzidos a uma análise histórica ou de afinidade entre os profissionais, que não cobre a maior parte dos problemas de comunicação de DDS decorrentes das diferenças culturais, temporais e geográficas.

Logo, por um lado, os aspectos não-técnicos influenciam na comunicação. Por outro lado, de acordo com (Ghezzi, Jazayeri e Mandrioli, 2002), as dependências entre os componentes de software também influenciam na comunicação requerida entre suas respectivas equipes de desenvolvimento. Neste contexto, a fim de mitigar os problemas de comunicação de DDS, considerando aspectos não técnicos das equipes e informações de dependências entre módulos de software, este artigo apresenta uma abordagem para alocação de equipes de desenvolvimento distribuídas a tarefas de implementação de módulos de software. Dada as inúmeras possibilidades de alocação, a engenharia de software baseada em buscas é utilizada com o objetivo de encontrar soluções que favoreçam a redução dos problemas de comunicação enfrentados com o DDS. Nesta direção, a abordagem proposta é baseada na metaheurística algoritmo genético.

O restante do artigo está estruturado da seguinte forma. A Seção 2 apresenta uma visão geral da abordagem proposta, identificando os artefatos manipulados em seus diversos passos, e, em seguida, detalha o algoritmo genético adotado. A Seção 3 discute trabalhos relacionados, evidenciando as contribuições da abordagem proposta. Por fim, a Seção 4 apresenta algumas considerações finais e menciona trabalhos futuros.

## **2. Uma Abordagem de Alocação de Equipes Distribuídas**

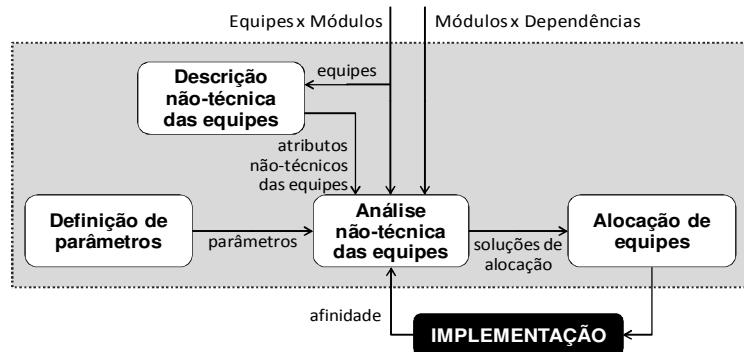
A abordagem aqui proposta representa uma das fases do framework de recomendação para alocação de equipes de desenvolvimento em projetos distribuídos de linhas de produtos de software, cujo conjunto de fases foi brevemente introduzido em (Santos, Pereira, *et al.*, 2010). Neste framework, considerando a arquitetura de uma linha de produtos, a primeira fase agrupa os componentes mais dependentes, gerando módulos de software com alta dependência interna e baixa dependência externa. Em seguida, considerando requisitos técnicos dos módulos e competências técnicas das equipes, a segunda fase seleciona as equipes de desenvolvimento tecnicamente qualificadas para implementá-los, ranqueando-as pela adequabilidade técnica. Por fim, a terceira fase, detalhada neste artigo, considera aspectos não-técnicos das equipes qualificadas e informações de dependência entre módulos para recomendar um conjunto de alocações de equipes para implementação dos módulos.

Portanto, a principal contribuição deste artigo é o detalhamento da fase de alocação de equipes do framework de recomendação, cujo objetivo é auxiliar o gerente de projeto na alocação de equipes distribuídas a módulos de software. A abordagem proposta é dividida em quatro passos (Figura 1), descritos nas sessões que se seguem.

### **2.1. Descrição Não-Técnica das Equipes**

Primeiramente, no passo denominado *descrição não-técnica das equipes*, os atributos não-técnicos das equipes são identificados e coletados. Estes atributos são importantes uma vez que impactam diretamente na qualidade da comunicação entre pares de equipes. Fundamentado nas discussões e sugestões encontradas em (Warren McFarlan, 1996), (Gumm, 2006), (Audy e Prikladnicki, 2007) e (Lamersdorf e Münch, 2010), os atributos não técnicos são divididos em três tipos: *temporal* (*fuso-horário*,

*horário de trabalho, dias semanais de trabalho), cultural (idiomas conhecidos e perfil organizacional e de infra-estrutura) e de afinidade (qualidade da comunicação).*



**Figura 1. Vista geral da abordagem de alocação de equipes**

Os atributos não-técnicos são obtidos através de diferentes questionários que são aplicados aos gerentes das equipes e também aos membros das equipes. É importante destacar que os questionários e as equações adotadas para calcular os valores finais dos atributos foram propositadamente não apresentados em função da limitação de espaço, e, principalmente, porque o foco principal é o detalhamento da metaheurística adotada.

Os atributos temporais são fornecidos por cada gerente para toda a equipe, e seus valores já são usados diretamente. Já os atributos culturais são denominados atributos indiretos, pois são fornecidos por cada membro de cada equipe, e, em seguida, os valores obtidos são processados para derivar os valores para as respectivas equipes.

Para cada idioma, o nível de domínio de cada membro da equipe é calculado como sendo a média do domínio na *escrita, leitura, fala e entendimento*, medidos em uma escala de quatro níveis (*1-baixo, 2-mediano, 3-excelente e 4-nativo*). Para a equipe, o nível de domínio é calculado como sendo a média do nível de domínio de seus membros, e, em seguida, normalizado-a para o intervalo [0, 1].

O perfil organizacional e de infra-estrutura é derivado das respostas do gerente da equipe a um questionário específico. Esse questionário identifica informações sobre os processos de software adotados pela equipe, bem como as ferramentas colaborativas usadas para viabilizar a comunicação. Este atributo também possui um valor no intervalo [0, 1] e representa a média dos valores para cada tipo de informação coletada.

A afinidade é um parâmetro de *feedback* dado pelo gerente da equipe após a implementação dos módulos e representa um indicador histórico da qualidade da comunicação entre as equipes na prática, refletindo quão bem a equipe remota interagiu com a equipe local. Também possui um valor no intervalo [0, 1] e é atualizado a cada iteração de desenvolvimento de um mesmo projeto ou até mesmo diferentes projetos. Cada par de equipes possui dois valores para o atributo de afinidade, uma vez que é um atributo bidirecional. Além disso, cada equipe possui um valor interno de afinidade, modificado com base em todas as avaliações realizadas para essa equipe.

## 2.2. Análise Não-Técnica das Equipes

Uma vez concluída a descrição não-técnica das equipes, o passo **definição de parâmetros** é iniciado. Os parâmetros são valores que podem ser modificados pelo gerente do projeto e identificam a importância de cada um dos aspectos não-técnicos (temporal, cultural e afinidade). Após esse passo, a **análise não-técnica das equipes** (Figura 1) se inicia com o objetivo de encontrar soluções de alocação de equipes a módulos de software, considerando os seguintes dados de entrada: os *atributos*

*não-técnicos das equipes*; o mapeamento *equipes x módulos*, que representa o conjunto de equipes tecnicamente qualificadas a implementar cada módulo; e o mapeamento *módulos x dependências*, que representa as dependências entre os módulos. Neste cenário, o universo de possíveis soluções é muito grande. Por exemplo, no caso de 10 equipes de desenvolvimento e 15 módulos de software, o espaço de busca é de  $10^{15}$ , que representa um número enorme de possíveis soluções de alocação para serem avaliadas integralmente em um tempo aceitável. Para evitar a busca exponencial por todo o espaço de soluções, este passo adota a metaheurística de algoritmo genético, considerada uma técnica de busca probabilística, não determinística e evolucionária, baseada na metáfora do processo biológico de evolução natural (Linden, 2008). Desta forma, a partir de uma mesma população inicial, diferentes soluções de alocação podem ser encontradas a cada execução do algoritmo.

Primeiramente, o algoritmo genético proposto cria a população inicial de soluções de alocação, randomicamente alocando equipes a módulos. Baseado na avaliação da qualidade das soluções, as melhores são selecionadas para conceber a próxima geração da população. Em geral, a seleção tem a tendência de escolher as melhores soluções. No entanto, mecanismos devem ser incorporados para permitir selecionar soluções fracas de modo a diversificar a busca e evitar a rápida convergência em ótimos locais. A avaliação da qualidade das soluções é realizada com base na *função objetivo*, que é definida em termos dos atributos temporais, culturais e de afinidade. Neste ponto, operadores de mutação, recombinação e reinserção são aplicados para conceber a nova geração da população. Este processo então se repete, voltando a seleção de soluções, até que seja alcançada uma condição de parada, que pode ser o número de iterações ou a estabilização da qualidade das soluções.

Na literatura, é possível identificar variantes para estes operadores, cada uma delas com impacto relevante na qualidade das soluções encontradas e na velocidade de convergência do algoritmo. Conseqüentemente, considerando que a abordagem aqui proposta é um trabalho ainda em andamento, os operadores a serem adotados somente serão definidos durante a realização de estudos de casos experimentais, onde será possível avaliar quais operadores são mais efetivos e eficientes. Assim, a seguir, apenas a representação das soluções e a função objetivo são introduzidas.

### 2.2.1. Representação das Soluções

As soluções de alocação são representadas por cromossomos, cujos genes representam o conjunto de módulos de software e os alelos representam as equipes qualificadas a implementá-los. A Figura 2 apresenta um exemplo de cromossomo, cujos genes representam 5 módulos ( $m_1, m_2, \dots, m_5$ ) e os alelos podem indicar 8 equipes ( $e_1, e_2, \dots, e_8$ ). Neste exemplo, todas as equipes são consideradas qualificadas para implementar todos os módulos. Observe que a Figura 2 também mostra as dependências entre os módulos. Por exemplo, considerando os módulos  $m_1$  e  $m_3$ , respectivamente, as equipes  $e_2$  e  $e_1$  são alocadas aos mesmos. Na prática, isso significa que as equipes  $e_2$  e  $e_1$  deverão se comunicar para implementar os módulos, pois os mesmos são inter-dependentes.

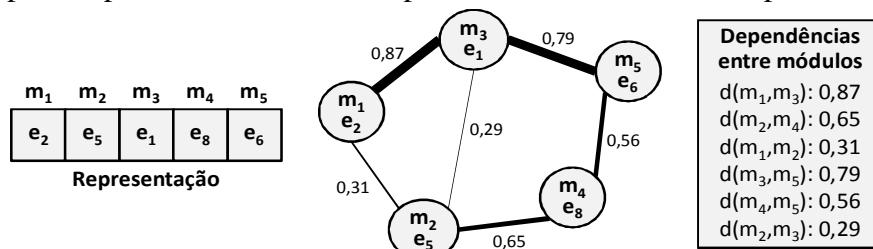


Figura 2. Representação de solução x Dependências entre módulos

### 2.2.2. Função Objetivo

Considerando os atributos temporais, culturais e de afinidade, a avaliação das equipes também é realizada em três diferentes dimensões: *avaliação temporal*, *avaliação cultural* e *avaliação da afinidade*. Dada uma solução de alocação  $s$ , as avaliações temporal, cultural e de afinidade resultam em valores no intervalo  $[0, 1]$ , representadas pelos termos  $A_t(s)$ ,  $A_c(s)$  e  $A_a(s)$ , respectivamente. Com o propósito de comparar a qualidade das soluções de alocação candidatas, estes termos são então aplicados a função-objetivo  $f(s)$ , definida na Equação (1), onde  $w_t$ ,  $w_c$  e  $w_a$  representam os pesos definidos pelo gerente de projeto para as dimensões temporal, cultural e de afinidade, respectivamente. Logo, antes de iniciar a análise propriamente dita, é necessário que o gerente de projeto defina os pesos das diferentes dimensões de modo a refletir o seu interesse na alocação. Conforme indicado na Figura 1, estes pesos são definidos no passo denominado **definição de parâmetros**.

$$f(s) = \left( \frac{w_t \times A_t(s) + w_c \times A_c(s) + w_a \times A_a(s)}{w_t + w_c + w_a} \right) \quad (1)$$

Para uma dada solução de alocação  $s$ , a avaliação em uma dada dimensão  $x$  é calculada como uma média ponderada do nível de dependência entre cada par de módulos ( $m_r, m_s$ ) e o valor da avaliação naquela dimensão entre as equipes ( $e_i, e_j$ ) alocadas ao par de módulos ( $m_r, m_s$ ). Formalmente, as avaliações temporal, cultural e de afinidade são calculadas usando uma expressão similar à indicada na Equação (2), porém substituindo os termos genéricos  $A_x(s)$  e  $a_x(e_i, e_j)$  por  $A_t(s)$  e  $a_t(e_i, e_j)$ ,  $A_c(s)$  e  $a_c(e_i, e_j)$ , e  $A_a(s)$  e  $a_a(e_i, e_j)$ , respectivamente. Note que, nesta equação, o termo  $d(m_r, m_s)$  representa o nível de dependência entre os módulos  $m_r$  e  $m_s$ . Já o termo  $a_x(e_i, e_j)$  representa o valor da avaliação da respectiva dimensão  $x$  entre o par de equipes ( $e_i, e_j$ ) alocadas aos módulos  $m_r$  e  $m_s$ .

$$A_x(s) = \frac{\sum_{r \neq s} d(m_r, m_s) \times a_x(e_i, e_j)}{\sum_{r \neq s} d(m_r, m_s)} \quad (2)$$

Os valores das avaliações temporal  $a_t(e_i, e_j)$ , cultural  $a_c(e_i, e_j)$  e de afinidade  $a_a(e_i, e_j)$  entre o par de equipes alocado aos módulos  $i$  e  $j$  são mensuradas levando-se em consideração os atributos temporais, culturais e de afinidade destas equipes.

Considerando um par de equipes  $e_i$  e  $e_j$ , a avaliação temporal é calculada, conforme indicado na Equação (3). Observe que, o lado esquerdo da equação representa o percentual de horas de trabalho semanal que as equipes  $e_i$  e  $e_j$  compartilham simultaneamente em relação ao conjunto total de horas de trabalho semanal de todas as equipes candidatas. Nesta porção da equação, o termo  $wt(e_i)$  representa o conjunto de turnos e horas de trabalho semanal da equipe  $e_i$ , baseado no tempo médio de Greenwich (GMT) para ajustar os fusos-horários. Como premissa, assumimos que quanto mais tempo simultâneo as equipes dispõem para se comunicar, maior a chance da comunicação ser efetiva. Já o lado direito da equação representa um fator de penalidade proporcional ao esforço de comunicação requerido das equipes  $e_i$  e  $e_j$  para implementar todos os módulos aos quais estão alocados. Nesta porção da equação, o termo  $d(m, m)$  representa a dependência interna do módulo  $m$ , e  $M_{e_i}$  representa o conjunto de módulos alocado à equipe  $e_i$ . Note que, quanto maior o número de módulos alocados as equipes, maior o esforço requerido, e, assim, menor a disponibilidade das equipes para comunicação com outras equipes.

$$a_t(e_i, e_j) = \frac{|wt(e_i) \cap wt(e_j)|}{|\cup_k wt(e_k)|} \times \left( 1 - \frac{\sum_{m \in M_{e_i}} d(m, m) + \sum_{m \in M_{e_j}} d(m, m)}{2 \times \sum_m d(m, m)} \right) \quad (3)$$

Por sua vez, como indicado na Equação (4), a avaliação cultural para um par de equipes é calculada como a média entre a avaliação do melhor idioma comum às equipes  $a_{id}(e_i, e_j)$  e da avaliação do perfil organizacional  $a_{org}(e_i, e_j)$  das equipes. Neste caso, a premissa é que equipes com nível de idioma e perfis organizacionais semelhantes possuem uma boa chance de se relacionarem satisfatoriamente, incrementando a chance da comunicação ser efetiva.

$$a_c(e_i, e_j) = \frac{a_{id}(e_i, e_j) + a_{org}(e_i, e_j)}{2} \quad (4)$$

A princípio, o cálculo do valor da avaliação do melhor idioma  $a_{id}(e_i, e_j)$  e do perfil organizacional  $a_{org}(e_i, e_j)$  pode ser realizado como a média dos níveis de idioma e dos níveis de perfil organizacional de cada equipe, respectivamente. No entanto, a distância entre os níveis de idioma e de perfil organizacional também é importante (Lamersdorf e Münch, 2010). Desta forma,  $a_{id}(e_i, e_j)$  e  $a_{org}(e_i, e_j)$  são calculados conforme indicado na Equação (5), onde os termos  $\mu_x(e_i, e_j)$  e  $\sigma_x(e_i, e_j)$  representam, respectivamente, a média e o desvio padrão do atributo  $x$  (nível de idioma ou perfil organizacional) das equipes. Logo, quanto mais distantes os níveis de idioma ou perfil organizacional, maior a influência negativa na avaliação do respectivo atributo.

$$a_x(e_i, e_j) = \mu_x(e_i, e_j) \times (1 - \sigma_x(e_i, e_j)) \quad (5)$$

No caso da avaliação da afinidade entre um par de equipes  $a_a(e_i, e_j)$ , o valor também é calculado considerando a média e o desvio padrão entre os níveis de afinidade indicados por seus respectivos gerentes de equipe ao final de cada interação de projeto, conforme indicado na Equação (6), onde o termo  $a_{i \rightarrow j}$  representa o nível de afinidade atribuído pelo gerente da equipe  $i$  para a equipe  $j$ . Logo, quanto mais distantes as afinidades, maior a influência negativa na avaliação do respectivo atributo. Caso as equipes estejam colaborando pela primeira vez, o valor neutro 0,5 é adotado para  $a_{i \rightarrow j}$  e  $a_{j \rightarrow i}$ . No caso de uma avaliação  $a_a(e_i, e_i)$ , a afinidade utilizada é a afinidade interna gerada pelas avaliações de todos as equipes realizadas sobre equipe  $e_i$ .

$$a_a(e_i, e_j) = \mu_a(a_{i \rightarrow j}, a_{j \rightarrow i}) \times (1 - \sigma_a(a_{i \rightarrow j}, a_{j \rightarrow i})) \quad (6)$$

### 2.2.3. Alocação de Equipes

Uma vez que as soluções de alocação são identificadas e recomendadas pelo algoritmo genético, o passo denominado **alocação das equipes** é realizado (Figura 1) pelo gerente de projetos com o objetivo de selecionar a solução de alocação que melhor se adéqua aos interesses do projeto.

Um ponto importante a ser destacado é que, no algoritmo proposto, a função objetivo  $f(s)$  avalia a qualidade das soluções de forma integrada em termos dos atributos temporais, culturais e de afinidade das equipes. No entanto, o objetivo desse trabalho é apresentar ao gerente do projeto um conjunto de opções, já avaliadas, para que ele possua subsídios para selecionar, diante de um conjunto menor de possibilidades, a que sua experiência mostra ser a melhor. Dessa forma, o conjunto de soluções deve ser apresentado ao gerente de forma a identificar, para cada solução, os valores das avaliações temporal, cultural e de afinidade, fazendo com que sua experiência também seja utilizada no processo de alocação.

### **3. Trabalhos Relacionados**

Nesta seção alguns trabalhos relacionados são apresentados com o objetivo de evidenciar o diferencial da abordagem proposta. Paulish (Paulish, 2003) apresenta um processo de desenvolvimento de linha de produto de software que considera a atividade de construção do software como sendo realizada por equipes distribuídas. Neste processo, os componentes de software representam a unidade de alocação para implementação, e, cada componente possui restrições de tempo e esforço de desenvolvimento definidos. No entanto, diferentemente da abordagem aqui proposta, não considera as dependências entre os componentes e os aspectos não-técnicos introduzidos pela dispersão geográfica. Além disso, os componentes possuem granularidade muito fina para ser a unidade de alocação, e, como tal, acabam gerando uma alta necessidade de comunicação entre as equipes. A proposta aqui apresentada, de forma distinta, considera os módulos de software, com intra-dependência alta e inter-dependência baixa, favorecendo a redução dos requisitos de comunicação entre equipes.

Duggan, Byrne e Lyons (Duggan, Byrne e Lyons, 2004) propõe um mecanismo de otimização baseado em algoritmo genético para alocação de tarefas na fase de construção do software. No entanto, este mecanismo realiza apenas a análise de aspectos técnicos das equipes, levando em conta o nível de conhecimento nos seguintes assuntos: segurança, rede, base de dados, controle e interface de usuário. Em contraste, a abordagem aqui proposta, juntamente com o framework de recomendação do qual é parte integrante, consideram tanto os aspectos técnicos quanto os aspectos não-técnicos.

Em (Barreto, 2005) é apresentado uma abordagem para a alocação de recursos humanos baseado em satisfação de restrições. Esta abordagem procura por uma solução válida baseada nas restrições definidas. De forma similar a proposta aqui apresentada, esta abordagem avalia aspectos técnicos e não-técnicos dos profissionais. No entanto, ao contrário da abordagem aqui proposta, não considera aspectos culturais e de afinidade, mas apenas aspectos temporais. Outro diferencial é o fato desta abordagem alocar pessoas a atividades do projeto, enquanto a presente abordagem aloca equipes distribuídas, considerando as características técnicas e não-técnicas individuais dos profissionais que as compõem.

Lamersdorf e Munch (Lamersdorf e Munch, 2010) apresentaram um método baseada em redes Bayesianas para alocar tarefas a equipes distribuídas. Neste método, aspectos técnicos e não-técnicos das equipes são também considerados, por exemplo, o custo para realizar as atividades e o custo de comunicação entre as localidades. Embora considere aspectos não-técnicos das equipes, esta ferramenta não define mecanismos para coletar informações sobre tais aspectos não-técnicos. Ao contrário da abordagem aqui proposta, cuja coleta de informações é baseada em questionários submetidos aos gerentes de equipes e seus membros. Este método adota apenas tabelas probabilísticas, não refletindo, portanto, a realidade e as especificidades de cada equipe.

### **4. Considerações Finais**

Apesar de existir diversas propostas na literatura para alocação de tarefas a equipes de desenvolvimento, raras são aquelas que consideram aspectos de DDS. Aquelas que consideram, ou tratam apenas aspectos técnicos, e, assim, são incapazes de lidar com os problemas de comunicação decorrentes da dispersão geográfica, ou também tratam os aspectos não-técnicos, mas de uma forma limitada e aproximada, por exemplo, desprezando aspectos culturais e de afinidade e usando informações estatísticas ao invés de informações reais e atualizadas das equipes.

Neste contexto, a abordagem de alocação aqui proposta, embora ainda represente um trabalho em andamento, tem o potencial de mitigar problemas de comunicação decorrentes de DDS, considerando os aspectos técnicos e não-técnicos das equipes. Além disso, o trabalho não negligencia a experiência do gerente de projetos, uma vez que as melhores soluções encontradas são apenas oferecidas ao mesmo, que, com sua experiência, pode realizar uma melhor alocação em função do reduzido conjunto de possibilidades, quando comparado com o espaço de busca. Atualmente, a abordagem está sendo incorporada em uma ferramenta que permitirá realizar a avaliação, refinamento e validação da abordagem com estudos de casos artificiais e também reais. Em uma próxima geração da abordagem e respectiva ferramenta, o custo e o prazo de desenvolvimento dos módulos também serão considerados.

## Referências

- ABNT, N. I. 1. Gestão da Qualidade – Diretrizes para a Qualidade no Gerenciamento de Projetos. Rio de Janeiro, RJ - Brasil: Associação Brasileira de Normas Técnicas, 2000.
- AUDY, J.; PRIKLADNICKI, R. Desenvolvimento Distribuído de Software. Rio de Janeiro: Campos/Elsevier, 2007.
- BARRETO, A. Apoio à Alocação de Recursos Humanos em Projetos de Software: Uma Abordagem Baseada em Satisfação de Restrições. Dissertação de Mestrado. UFRJ. 2005.
- CALLEGARI, D. A.; FOLIATTI, F. L.; BASTOS, R. M. MRES - Ferramenta para Seleção de Recursos para Tarefas de Projetos de Software via Abordagem Difusa e Multicritérios. Simpósio Brasileiro de Engenharia de Software (SBES 2009). Fortaleza. 2009.
- DUGGAN, J.; BYRNE, J.; LYONS, G. J. A Task Allocation Optimizer for Software Construction. IEEE Software. vol. 21, issue 3. 2004. p. 76-82.
- GHEZZI, C.; JAZAYERI, M.; MANDRIOLI, D. Fundamentals of Software Engineering. Upper Saddle River: Prentice Hall, 2002.
- GUMM, D. C. Distribution Dimensions in Software Development Projects: A Taxonomy. IEEE Software, vol 3, issue 5, September & October 2006. pp.45-51.
- HERBSLEB, J. D.; GRINTER, R. E. Splitting the Organization and Integrating the Code: Conway's Law Revisited. Int. Conf. on Soft. Eng. (ICSE'99). Los Angeles. 1999.
- HERBSLEB, J. D.; MOCKUS, A. An Empirical Study of Speed and Communication in Globally-Distributed Software Development. IEEE Trans. Soft. Eng. vol. 29, no. 6, 2003. pp.481-494.
- HUZITA, E. et al. Um Conjunto de Soluções para Apoiar o Desenvolvimento Distribuído de Software. II Workshop de Desenvolvimento Distribuído de Software (WDDS 2008). Campinas. 2008. pp.101-110.
- LAMERSDORF, A.; MÜNCH, J. Studying the Impact of Global Software Development Characteristics on Project Goals: A Causal Model. The Open Software Engineering Journal. 2010. pp.2-13.
- LINDEN, R. Algoritmos Genéticos. Rio de Janeiro: Brasport, 2008.
- PAULISH, D. J. Product Line Engineering for Global Development. International Workshop on Product Line Engineering: The Early Steps. 2003.
- SANTOS, V. et al. Um Framework de Recomendação para Alocação de Equipes de Desenvolvimento em Projetos Distribuídos de Linhas de Produto de Software. IV Workshop de Desenvolvimento Distribuído de Software (WDDS 2010). Salvador. 2010.
- SILVA, M. A. WebAPSEE-Planner: Auxílio à Alocação de Pessoas em Projetos de Software através de Políticas. UFPA. 2007.
- WARREN MCFARLAN, F. Issues in Global Outsourcing. Global Information Technology and Systems Management. Nashua: Ivy League Publishing. 1996.

# Otimização heurística de uma técnica para seleção e priorização de portfólios balanceados de projetos de software

Fábio V. Figueiredo, Márcio de O. Barros

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

{fabio.figueiredo, marcio.barros}@uniriotec.br

**Abstract.** This paper proposes a heuristic approach to allow a Software Project Portfolio Selection Technique to be used on scenarios with a large number of candidate projects to compose the portfolio. While the original technique is based on analyzing all possible project combinations for selecting the best one, we propose using a genetic algorithm to partially explore this search space and select an efficient project combination. The paper presents an experimental study that compares the proposed heuristic with a local search technique (Hill Climbing) and a non-systematic search technique (Random Search). Its results confirm that the project portfolio technique can be used in large scale and the proposed heuristic presents better results than the other search strategies selected to be part of the experimental study.

**Resumo.** Este artigo propõe uma heurística que permite que uma técnica de seleção de projetos de software visando compor um portfólio balanceado seja aplicada em cenários com grande número de projetos candidatos. Enquanto a técnica original se baseia na análise de todas as combinações possíveis dos projetos para posterior seleção da melhor combinação, propomos o uso de um algoritmo genético para explorar parcialmente o conjunto destas combinações e selecionar uma combinação eficiente. O artigo apresenta um estudo experimental que compara a heurística proposta com uma técnica de busca local (Hill Climbing) e uma técnica de busca não-sistêmática (Random Search). Os resultados do estudo confirmam que a técnica de seleção de portfólios de projetos de software pode ser utilizada em larga escala e que a heurística proposta apresenta melhores resultados que as outras estratégias de busca selecionadas para o estudo experimental.

## 1. Introdução

Segundo o Project Management Institute [PMI 2008], um portfólio é uma coleção de projetos, programas, portfólios e outros trabalhos que são agrupados para facilitar a gerência efetiva do trabalho e o alcance dos objetivos estratégicos de negócio. A gestão de um portfólio inclui a identificação, categorização, avaliação, seleção, priorização, autorização, gerência e controle dos projetos, programas e trabalhos relacionados, de acordo com o seu alinhamento com os objetivos e estratégias da organização, trazendo diversos benefícios para as organizações, como auxílio na seleção e priorização de idéias e projetos candidatos para execução, alinhamento dos investimentos aos objetivos políticos e estratégicos da organização, entre outros.

Costa et al. (2010) propõem uma técnica de seleção e priorização de projetos para a composição de portfólios平衡ados de projetos de software. A técnica permite

que o gestor identifique os portfólios que possuem os maiores índices de eficiência e que possam ser compostos a partir de um conjunto de projetos candidatos. A eficiência de um portfólio é calculada através do IEP (Índice de Eficiência do Portfólio), que é calculado dividindo-se o retorno esperado do portfólio por seu risco. O retorno do portfólio é estimado a partir dos retornos esperados para os projetos, enquanto o risco do portfólio é calculado de acordo com a exposição individual de cada projeto a uma série de ameaças e oportunidades relevantes identificadas pelo gestor. Com base nas relações existentes entre os projetos candidatos e estas ameaças/oportunidades, calcula-se o grau de dependência entre os projetos, que permite considerar a diversificação dos riscos no portfólio. A técnica utiliza conceitos da Teoria Moderna do Portfólio [Markowitz 1952] e mostrou bons resultados em testes que foram executados com instâncias controladas de projetos de software.

Embora permita a seleção de portfólios平衡ados, a técnica proposta por Costa et al. (2010) possui complexidade computacional exponencial com relação ao número de projetos candidatos. Isto ocorre porque a técnica se baseia na análise de todas as possíveis combinações entre os projetos candidatos, ou seja, são construídos e avaliados todos os portfólios que podem ser formados por subconjuntos dos projetos candidatos. Sendo assim, o número de projetos candidatos afeta o tempo de execução da técnica, inviabilizando seu uso em larga escala. Por exemplo, ao utilizar a técnica para um cenário com cinco projetos candidatos, um gestor terá que avaliar 32 portfólios. No entanto, se houver 40 projetos candidatos, o número de possíveis combinações passa de um trilhão. Neste cenário, a técnica não pode ser aplicável em tempo aceitável.

Neste artigo, propomos utilizar uma busca heurística baseada em algoritmos genéticos para criar uma implementação da técnica estudada cujo uso seja viável em situações onde o número de projetos candidatos não permita a análise de todas as suas possíveis combinações. Esse tipo de busca percorre as combinações de projetos segundo uma estratégia que tem como objetivo encontrar uma boa solução sem examinar todas as possibilidades, sendo útil quando o número de projetos candidatos a compor o portfólio é muito grande. Para avaliar se a busca heurística é efetivamente necessária e indicada para resolver este problema, projetamos e executamos um estudo experimental que compara os resultados do algoritmo genético proposto com os resultados de uma busca local (Hill Climbing) e de uma busca não-sistemática (Random Search).

Este artigo está dividido em 7 seções. A Seção 1 apresenta esta introdução. A Seção 2 apresenta o problema a ser tratado e descreve o funcionamento da técnica estudada. A Seção 3 é composta da formalização do modelo da técnica de seleção de projetos para a composição de portfólios balanceados de projetos de software. A seção 4 descreve a configuração e a condução do estudo experimental para comparar os resultados do algoritmo genético proposto com as outras estratégias de busca utilizadas neste estudo (busca local e busca não-sistemática). A seção 5 faz uma análise dos resultados apresentados pelo estudo experimental e a seção 6 discute as ameaças que podem afetar a validade do estudo empírico. Por fim, a seção 7 encerra o artigo.

## **2. Seleção de Projetos para a Composição de Portfólios**

Dentre as funções de um gestor de portfólio estão a seleção, priorização e balanceamento do portfólio, visando selecionar os projetos mais vantajosos dentre as várias alternativas, tendo em vista os recursos financeiros disponíveis para esse fim e

outros fatores relevantes, internos e externos à empresa [Kooros e McManis 1998]. A seleção de projetos para compor o portfólio é uma tarefa complexa, pois geralmente envolve projetos com diferentes características, riscos e custos [Borgonovo e Peccati 2006]. Por conta destas dificuldades, os gestores costumam se valer de técnicas que os auxiliam na tomada de decisão.

A técnica proposta por Costa et al. (2010) caracteriza os projetos e os riscos envolvidos na composição do portfólio com as seguintes informações: (i) identificação para cada projeto candidato; (ii) custo necessário para sua execução; (iii) retorno esperado para cada projeto, geralmente expresso como valor presente líquido; (iv) identificação de cada risco que possa afetar um ou mais projetos; (v) probabilidade de ocorrência de cada risco; e (vi) impacto financeiro que cada risco causará em cada projeto afetado por ele. A técnica define também relações entre os projetos, onde pares de projetos podem ser considerados: (a) dependentes, se os dois projetos precisam, obrigatoriamente, fazer parte do mesmo portfólio; (b) mutuamente exclusivos, se os dois projetos não podem estar presentes no mesmo portfólio; ou (c) independentes, se não existir relação entre eles. As relações entre os projetos são tratadas como restrições, impedindo que certas combinações de projetos sejam válidas na seleção do portfólio.

A partir das informações sobre os projetos e riscos, a técnica identifica todos os portfólios que possam ser formados por combinações válidas de projetos candidatos, ou seja, combinações que atendam às restrições impostas pelas relações entre projetos. Em seguida, a técnica identifica todos os possíveis cenários de riscos, com base na combinação dos riscos que foram anteriormente identificados, calculando a probabilidade de cada cenário de acordo com a probabilidade de ocorrência dos riscos que o compõem. Para cada cenário de risco, a técnica calcula o retorno esperado e o risco de cada portfólio. O retorno esperado do portfólio é calculado pela soma dos retornos esperados dos seus projetos, enquanto o risco do portfólio é calculado pela composição dos riscos de seus projetos, considerando as correlações entre eles. As correlações são calculadas de acordo com o impacto dos cenários de risco em cada projeto. Maiores detalhes sobre estes cálculos podem ser encontrados em [Costa 2011].

Finalmente, a técnica constrói uma Fronteira Eficiente, gerando um gráfico de dispersão bidimensional, com o risco agregado e o retorno agregado do portfólio nos eixos. O risco agregado é calculado pela média dos riscos nos diferentes cenários, ponderada pela probabilidade de cada cenário. Um procedimento similar é utilizado para o retorno. A fronteira eficiente é um gráfico côncavo e os portfólios apresentados no limite superior desta fronteira são os mais eficientes, ou seja, aqueles que oferecem o maior retorno por unidade de risco.

Uma limitação da técnica de Costa et al. (2010) é que o seu tempo de execução cresce exponencialmente com o número de projetos candidatos e de riscos identificados. O espaço de busca pode ser aproximado por  $(2^n) * (2^m)$  alternativas, onde  $n$  representa o número de projetos candidatos e  $m$  representa a quantidade de riscos identificados. Portanto, dependendo do número de projetos candidatos ao portfólio, a técnica não conseguirá percorrer todas as alternativas em tempo aceitável. Assim, sugerimos a aplicação de uma busca heurística, que consiga retornar uma boa solução para o problema de seleção de projetos a partir de uma análise parcial do espaço de busca.

### 3. Formalização do Modelo da Técnica de Seleção de Projetos

Seja  $PC$  o conjunto de todos os projetos candidatos a compor o portfólio da empresa. Cada projeto  $p_i \in PC$  representa um projeto candidato e contém o nome do projeto, o custo necessário para sua execução e o retorno esperado para o projeto. Assim, temos  $p_i = (\text{nome}, \text{custo}, \text{retorno})$ , com  $i = 1, \dots, N$ , onde  $N$  é o número total de projetos candidatos do conjunto  $PC$ .

Seja  $RS$  o conjunto de riscos identificados. Cada risco  $r_j \in RS$  representa uma ameaça ou oportunidade que pode afetar um ou mais projetos candidatos. Cada risco  $r_j$  é descrito por seu nome, sua probabilidade de ocorrência e o impacto financeiro que causará em cada projeto afetado por ele. Então, temos  $r_j = (\text{nome}, \text{probabilidade}, \text{impacto}_{ji})$ , com  $j = 0, \dots, R$ , onde  $R$  é o número de riscos identificados pelo gestor;  $r_j.\text{probabilidade} \in \mathbb{R}$ ,  $0 \leq r_j.\text{probabilidade} \leq 100\%$ ; e  $\text{impacto}_{ji} \in \mathbb{R}$  é o impacto que o risco  $r_j$  causará no projeto  $p_i \in PC$ , caso ocorra (zero, se  $p_i$  não é afetado por  $r_j$ ).

Seja  $P$  o conjunto dos portfólios que podem ser formados pelos projetos candidatos.  $P$  é um conjunto potência de  $PC$ , ou seja, o conjunto de todos os subconjuntos de  $PC$ ;  $P = \{P_z\}$ ;  $P_z \subseteq PC : P_z = [w_1, \dots, w_N] / w_i = 0 \text{ se } p_i \notin P_z; w_i = 1 \text{ se } p_i \in P_z ; \forall p_i \in PC. z = 1, \dots, 2^N : Z$  representa o número de possíveis portfólios.

Seja  $IEP(P_z)$  o índice de eficiência do portfólio  $P_z$ ,  $P_z \in P$ , calculado pela razão entre o retorno esperado e o risco do portfólio.  $IEP(P_z)$  é um índice que demonstra a quantidade de retorno atribuída a um portfólio por unidade de risco, que é uma medida teórica da eficiência do portfólio. O gestor deseja selecionar o portfólio  $P_z^* \in P$  que maximize  $IEP(P_z^*)$  e cujo custo esteja restrito ao montante financeiro disponível. Formalmente, essa seleção pode ser descrita como: selecionar um portfólio  $P_z^*$  onde  $\nexists P'_z / IEP(P'_z) > IEP(P_z^*) ; \sum(p_i.\text{custo} \in P_z) \leq OD \forall p_i \in PC : OD$  representa o orçamento disponível da empresa para esse fim.

### 4. Estudo Experimental

Para tratar a limitação da técnica de Costa et al. (2010) quanto ao número de projetos candidatos ao portfólio, foi projetado e executado um estudo experimental que compara uma busca heurística baseada em Algoritmos Genéticos [Holland 1962] com uma técnica de busca não-sistemática (Random Search) [Rastrigin 1963] e com uma técnica de busca local (Hill Climbing) [Russell e Norvig 2003].

Acreditamos que uma busca heurística seja necessária para encontrar boas soluções para o problema proposto. Desta forma, compararemos seus resultados com uma busca local e usaremos uma busca aleatória como teste de sanidade para o estudo experimental.

Os detalhes do código-fonte desenvolvido como parte deste estudo e as instâncias geradas para avaliação das técnicas propostas podem ser vistos nas subseções 4.2 e 4.3 deste artigo e estão disponíveis para download em <http://www.uniriotec.br/~marcio.barros/gaportfolio>.

#### 4.1. Questões de pesquisa

Neste estudo, foram consideradas as seguintes questões de pesquisa:

**QP1.** Eficiência como critério de avaliação: É possível utilizar a estratégia descrita na técnica de Costa et al. (2010) em cenários onde o número de projetos candidatos a compor o portfólio não permita a análise de todas as suas possíveis combinações?

Esta questão trata do tempo de execução de cada algoritmo de busca utilizado no estudo, de forma que se comprove que serão apresentadas soluções em tempo aceitável.

**QP2.** Eficácia como critério de avaliação: Dentre os algoritmos utilizados no estudo, qual melhor maximiza a função aptidão que trata do índice de eficiência do portfólio?

Esta questão compara as soluções apresentadas pelos três algoritmos quanto ao valor do índice de eficiência do portfólio, que é a medida que deve ser maximizada na busca pelas melhores soluções. Espera-se que as soluções apresentadas pelo algoritmo genético sejam melhores que as apresentadas pelos outros dois algoritmos.

#### 4.2. Instâncias Utilizadas no Estudo

Para responder às questões colocadas na subseção 4.1, este estudo experimental utilizou um conjunto de 12 instâncias geradas com dados aleatórios através de um gerador de instâncias desenvolvido pelo grupo de pesquisa dos autores deste artigo, variando-se o número de projetos (25, 50 e 75 projetos) e o número de riscos (3, 5, 7 e 10 riscos).

O valor presente dos projetos foi gerado segundo uma distribuição uniforme com limite inferior de 1.000 e limite superior de 1.500. O custo dos projetos também seguiu uma distribuição uniforme com limites entre 15.000 e 30.000.

A probabilidade de ocorrência dos riscos seguiu uma distribuição de 0% a 100% e o impacto que o risco causará em cada projeto seguiu uma distribuição uniforme entre -8.400 e 5.000 (valores negativos representam ameaças, enquanto valores positivos representam oportunidades, ambas incertezas e, portanto, riscos). Estes valores foram estabelecidos por conveniência para que houvesse consistência entre risco e retorno.

#### 4.3. Algoritmos Utilizados no Problema

O estudo experimental executou 30 vezes cada algoritmo sobre cada instância. As informações observadas foram o tempo gasto na execução de cada algoritmo e o índice de desempenho para cada solução. Esses dados foram coletados nas 30 repetições e posteriormente analisados com o teste de Wilcoxon-Mann-Whitney [Feltovich 2003], para que fosse determinado se os resultados apresentados pelo algoritmo genético eram significativamente distintos dos outros dois algoritmos com 95% de certeza.

Os algoritmos de busca foram implementados utilizando o framework jMetal [Durillo et al. 2010], que é uma solução de código reutilizável orientada a objeto, baseada em Java, criada para dar suporte ao desenvolvimento, à experimentação, e ao estudo de algoritmos meta-heurísticos para resolver problemas de otimização.

Foi utilizada uma codificação binária para descrever as soluções para os algoritmos, onde cada solução representa um possível portfólio de projetos e é composta de N bits, onde N representa o número total de projetos candidatos ao portfólio. Por exemplo: para um cenário com os projetos A, B, C, D e E, um portfólio que contenha apenas o primeiro, o terceiro e o quarto projeto, a representação seria  $P_z = [1, 0, 1, 1, 0]$ , representando uma solução da qual somente os projetos A, C e D fazem parte.

O algoritmo genético foi configurado com o tamanho da população igual a N/2, onde N representa o número de projetos candidatos ao portfólio. Elitismo foi utilizado para prevenir a perda de boas soluções na geração de novas populações. Utilizamos *single point crossover* para a operação de cruzamento, com probabilidade de 80% e os indivíduos destinados à geração de novos indivíduos foram selecionados pelo método da roleta. Finalmente, utilizamos *single point mutation* como operador de mutação, com probabilidade de 2%. Estes parâmetros foram escolhidos por similaridade com outros experimentos que utilizam algoritmos genéticos e ainda pretendemos realizar testes de sensibilidade sobre eles.

O Hill Climbing [Russell e Norvig 2003] é um algoritmo iterativo que começa com uma solução aleatória no espaço de busca e prossegue tentando encontrar uma solução melhor através de mudanças incrementais, mudando um único bit da solução corrente. O algoritmo foi modificado para ser reiniciado em um ponto aleatório possivelmente diferente quando encontra um ótimo local, continuando esse reinício aleatório até que atinja o número pré-definido de execuções.

O Random Search [Rastrigin 1963] é um algoritmo que procura soluções para um problema explorando o espaço de busca de forma aleatória, armazenando a melhor solução até o momento, até que um critério de parada seja atendido. Para este estudo experimental o mesmo critério de parada foi dado aos três algoritmos, ou seja, o número de avaliações a serem executadas equivale a N<sup>2</sup>.

## 5. Análise dos Resultados

A tabela 2 apresenta a média aritmética dos índices de eficiência dos portfólios IEP (MAI), o desvio padrão desses índices (DPI) e a média aritmética do tempo gasto (MTG) para a execução do algoritmo, medido em segundos, para as 30 execuções das instâncias por cada um dos algoritmos utilizados no estudo.

**Tabela 2 – Resultados originados das instâncias aleatórias**

INSTÂNCIAS Projetos / Riscos	Genetic Algorithm MAI / DPI / MTG			Hill Climbing MAI / DPI / MTG			Random Search MAI / DPI / MTG		
	MAI	DPI	MTG	MAI	DPI	MTG	MAI	DPI	MTG
025P / 03R	4,85	±3,01	2,00	4,77	±4,25	2,00	0,87	±0,40	2,00
025P / 05R	37,65	±56,41	1,07	8,30	±8,16	1,37	0,92	±0,30	2,00
025P / 07R	42,73	±43,14	1,20	18,70	±34,72	1,00	1,24	±0,24	2,00
025P / 10R	12,41	±7,27	2,33	8,09	±8,42	3,00	0,44	±0,67	4,00
050P / 03R	43,75	±42,28	42,13	7,27	±6,24	40,67	5,52	±5,43	41,33
050P / 05R	26,09	±19,92	49,10	11,15	±10,14	50,83	N/A	-	-
050P / 07R	150,46	±310,86	38,47	22,28	±65,03	38,97	N/A	-	-
050P / 10R	46,14	±39,70	53,20	8,51	±8,41	56,33	N/A	-	-
075P / 03R	38,76	±52,62	255,20	8,48	±11,50	260,03	8,76	±10,31	268,57
075P / 05R	89,59	±64,35	235,73	34,16	±20,07	183,60	0,46	±0,08	213,40
075P / 07R	36,96	±69,23	207,60	11,74	±16,39	147,40	0,98	±0,08	210,83
075P / 10R	51,59	±53,63	260,60	28,41	±37,07	198,87	0,94	±0,08	285,77

Respondendo a primeira questão de pesquisa (QP1), é possível ver que os resultados acima demonstram, empiricamente, que é possível utilizar a técnica descrita por Costa et al. (2010) em cenários onde o número de projetos candidatos a compor o portfólio não permita a análise de todas as suas possíveis combinações em tempo aceitável, pois todas as três estratégias de busca utilizadas neste estudo apresentaram resultados, com uma média aceitável de tempo de execução (MTG).

Nos resultados mostrados na tabela 2 também é possível ver que a segunda questão de pesquisa (QP2) pode ser respondida. O algoritmo genético é o que melhor maximiza a função aptidão, que trata do índice de eficiência do portfólio, pois foi o que apresentou melhores índices em todos os resultados. Algumas vezes o algoritmo de busca não-sistemática não conseguiu encontrar soluções viáveis (estes resultados são representados como "N/A" na Tabela 2).

O computador utilizado para a execução deste estudo experimental possui um processador AMD Athlon II X2 245 2,91 GHz, com 2,00 GB de memória RAM do tipo DDR3 e 160 GB de disco rígido de 7200 rpm SATA, executando Windows 7, versão de 32 bits e foi utilizado em regime de exclusividade.

## **6. Ameaças à Validade do Estudo**

Para este estudo, neste contexto, podem ser consideradas ameaças externas: (i) a falta de uma definição clara das instâncias utilizadas no estudo; (ii) a não apresentação de uma estratégia de seleção das soluções do espaço de busca; e (iii) a falta de instâncias de tamanho e complexidade crescentes para a avaliação da abordagem proposta. Estas ameaças foram tratadas neste estudo ao serem descritos os dados da geração das instâncias na subseção 4.2, as estratégias de seleção adotadas e ao serem utilizadas instâncias de três tamanhos distintos, no que se refere aos projetos candidatos ao portfólio e com quatro diferentes medidas de complexidade, referindo-se ao número de riscos envolvidos no problema.

As ameaças internas incluem: (i) a não discussão do código fonte utilizado no estudo experimental; (ii) a falta de informação quanto aos parâmetros utilizados na execução das técnicas de busca empregadas; (iii) a falta de uma descrição clara de procedimentos para levantamento de dados; e (iv) a falta de exemplos do mundo real. Para o tratamento destas ameaças, foram disponibilizadas para download no endereço eletrônico descrito na seção 4, as instâncias geradas para avaliação das técnicas propostas, comentadas na subseção 4.2 e o código-fonte desenvolvido como parte deste estudo experimental, comentado na subseção 4.3., com sua descrição e a parametrização utilizada. Exemplos do mundo real não foram abordados aqui, mas acredita-se que com as 30 execuções para cada uma das 12 instâncias utilizadas, foi possível responder as questões de pesquisa colocadas na subseção 4.1. Futuramente, espera-se complementar este estudo com tais instâncias do mundo real.

Ameaças à validade da construção de estudos experimentais podem ser: (i) a falta de uma avaliação da métrica de custo, tratada neste artigo como sendo a resposta da primeira questão de pesquisa (QP1), onde foi usada uma métrica aceitável para estimar o custo de cada algoritmo (tempo de execução); (ii) a falta de uma avaliação da validade das métricas de eficácia, discutido na seção 3 com a definição do IEP( $P_z$ ); e (iii) a falta de uma discussão sobre o modelo que suporta o processo de otimização, discutido também na seção 3 deste artigo, ao transformar o problema da seleção de projetos para o portfólio em um problema de otimização.

Por fim, as ameaças às conclusões incluem: (i) a desconsideração da natureza aleatória dos algoritmos de busca heurística; e (ii) a falta de uma base de comparação relevante para avaliar um algoritmo de busca heurística e de testes de inferência estatística. Estas ameaças foram tratadas neste estudo pela execução de vários ciclos para cada configuração do algoritmo (30 execuções), apresentando a média aritmética e

o desvio padrão dos IEP( $P_z$ ), comparando esses valores usando o teste estatístico não-paramétrico de Wilcoxon-Mann-Whitney.

## 7. Conclusão

Neste artigo, foi possível ver o funcionamento da técnica estudada e a estratégia proposta para torná-la aplicável em larga escala quanto ao número de projetos candidatos ao portfólio, sendo possível confirmar, através do estudo experimental, que a técnica de seleção de portfólios de projetos de software estudada pode ser utilizada em cenários com muitos projetos e que a heurística proposta apresenta melhores resultados que as outras estratégias de busca utilizadas na pesquisa. Esta pesquisa possui como foco o tratamento da limitação ligada ao número de projetos apenas, não levando em consideração o número de riscos identificados, o que também poderá tornar a técnica estudada inviável, dependendo da quantidade de riscos envolvidos no problema.

Trabalhos futuros referentes à pesquisa aqui desenvolvida podem englobar o tratamento e/ou modificação do modelo descrito na técnica estudada para torná-la aplicável também em cenários onde a quantidade de riscos inviabilize a sua utilização e tratar ainda a possibilidade de fazer com que a estratégia descrita na técnica estudada possa avaliar projetos cujos retornos e custos não sejam quantificáveis.

## Agradecimentos

Os autores agradecem à FAPERJ e CNPq pelo apoio financeiro provido para este projeto de pesquisa.

## Referências

- Costa, H. R. (2011). "Apoio à Seleção de Portfólio de Projetos de Software baseado na Moderna Teoria do Portfólio", Tese de Doutorado, PESC, COPPE/UFRJ.
- Costa, H. R.; Barros, M. O.; Rocha, A. R. (2010). "Software Project Portfolio Selection. A Modern Portfolio Theory Based Technique". The 22<sup>nd</sup> International Conference on Software Engineering and Knowledge Engineering, San Francisco, USA.
- Durillo, J.J.; Nebro, A.J.; Alba, E. (2010). "The jMetal Framework for Multi-Objective Optimization: Design and Architecture". CEC 2010, pp: 4318-4325
- PMI (2008). "Project Management Institute (PMI): a guide to the project management body of knowledge". Newtown Square.
- Borgonovo, E.; Peccati, L. (2006). "The importance of assumptions in investment evaluation", International Journal of Production Economics, vol.101, p.298.
- Feltovich, N. (2003). "Nonparametric Tests of Differences in Medians: Comparison of the Wilcoxon–Mann–Whitney and Robust Rank-Order Tests. Experimental Economics", Springer Netherlands.
- Russell, S.J.; Norvig, P. (2003). "Artificial Intelligence: A Modern Approach (2nd ed.)", Upper Saddle River, New Jersey: Prentice Hall, pp. 111–114.
- Kooros, S. K.; McManis, B. L. (1998). "A multiattribute optimization model for strategic investment decisions". Canadian Journal of Administrative Sciences, vol.15, n.2, p.152.
- Rastrigin, L.A. (1963). "The convergence of the random search method in the extremal control of a many parameter system". Automation and Remote Control 24 (10): 1337–1342.
- Holland, J. H. (1962). "Outline for a logical theory of adaptive systems", Journal of the ACM (JACM), vol 9, nr. 3, pp. 279–314.
- Markowitz, H. M. (1952). "Portfolio Selection". The Journal of Finance 7 (1): 77–91.

# Uma Abordagem Otimizada para o Problema de Alocação de Equipes e Escalonamento de Tarefas para a Obtenção de Cronogramas Eficientes

Ítalo Mendonça. Rocha<sup>1</sup>, Gerardo Valdisio R. Viana<sup>1</sup>, Jerffeson Teixeira de Souza<sup>1</sup>

<sup>1</sup>Mestrado Acadêmico em Ciência da Computação - Universidade Estadual do Ceará (UECE)

italomr@gmail.com, valdisio@uece.br, prof.jerff@gmail.com

**Abstract.** One of the main activities of project planning is the planning schedule, which includes allocations of teams and task scheduling. The Problem of Human Resource Allocation is to assign responsibility for a task to a set of human resources while the Resource-Constrained Project Scheduling Problem determines the start time of execution of each task. However, these problems are usually dealt with in isolation. This article proposes a hybrid approach that uses optimization techniques to the Scheduling Developing Problem involving the two problems in order to find good solutions within reasonable computational time.

**Resumo.** Uma das principais atividades do planejamento de projetos é o planejamento de cronograma, que compreende alocar equipes e escalar tarefas. O Problema de Alocação de Recursos Humanos consiste em atribuir as responsabilidades de uma tarefa a um conjunto de recursos humanos, enquanto o Problema de Escalonamento de Tarefas com Restrição de Recursos determina o instante de início de execução de cada tarefa. No entanto, geralmente esses problemas são trabalhados isoladamente. Este artigo propõe uma abordagem híbrida que utiliza técnicas de otimização para o Problema de Elaboração de Cronograma envolvendo os dois problemas a fim de encontrar boas soluções num tempo computacional aceitável.

## 1. Introdução

O Problema de Elaboração de Cronograma (*PEC*) é uma prática fundamental da engenharia de software que define as responsabilidades dos recursos humanos e os instantes de início de cada atividade em um projeto com restrições específicas a fim de alcançar objetivos diversos e muitas vezes conflitantes. Ele compreende a junção de dois conhecidos problemas: Alocações de Recursos Humanos (*HRAP* - *Human Resources Allocation Problem*) e Escalonamento de Tarefas com Restrição de Recursos (*RCPSP* - *Resource-Constrained Project Scheduling Problem*). Apesar destes dois problemas estarem intrinsecamente relacionados, geralmente eles são trabalhados individualmente. Pesquisas que abordam apenas a alocação de equipes consideram previamente estabelecida a ordem de execução das tarefas. Portanto procura-se apenas definir a melhor forma de alocar os recursos para aquela ordem. De forma análoga, problemas de escalonamento de tarefas geralmente consideram as alocações pré-estabelecidas e destinam-se apenas em definir a melhor ordem de execução. No entanto, esses problemas dependem da qualidade da escolha prévia da ordem da execução das atividades ou de quem executará quais tarefas. Logo, o *PEC* procura tanto determinar

a melhor alocação dos recursos como a melhor ordem de execução a fim de buscar cronogramas ótimos ou próximos do ótimo.

Os dois problemas são difíceis de serem resolvidos em sua otimalidade e são classificados como *NP-Completo*s. A junção deles no intuito de desenvolver o cronograma é ainda um problema mais complexo e, portanto, exigiria um tempo muitas vezes inaceitável em projetos de grande porte. Isso justifica o fato desse trabalho adotar heurísticas para o desenvolvimento de cronogramas ótimos ou próximos do ótimo.

**Engenharia de Software Baseada em Buscas (SBSE - Search Based Software Engineering)** é uma área emergente da ciência da computação que utiliza técnicas de otimização matemática, geralmente metaheurísticas, para resolver problemas complexos da engenharia de software.

Esse artigo tem como objetivo propor uma modelagem híbrida para o PEC, compreendendo a junção do HRAP e RCPSP, a fim de gerar bons cronogramas, minimizando a duração e o custo do projeto e maximizando a qualidade da equipe alocada, através da utilização de técnicas computacionais de otimização.

O Algoritmo Genético de Ordenação Não Dominante (*Non-dominated Sorting Genetic Algorithm*) é um algoritmo evolucionário baseado no conceito de **não-dominância**. Inicialmente proposto por Srinivas e Deb (1994) e posteriormente por Deb et al. (2002) em sua versão mais recente, o NSGA-II, base de nossa implementação, é provavelmente a mais utilizada meta-heurística para otimização multiobjetivo.

Dentre os trabalhos relacionados ao HRAP destacam-se Antoniol et al. (2004), Antoniol et al. (2005), Alba and Francisco Chicano (2007) e Colares (2010). Podemos citar como referências à resolução do RCPSP os trabalhos de Alcaraz and Maroto (2001), Simões (2004) e Tchao (2007). Chang et al. (2008) apresentaram ideias e uma modelagem interessante para o PEC utilizando o conceito de linha de tempo. A abordagem utilizada evitou o uso de técnicas de geração de escalonamento, que definem os instantes de início e fim de cada atividade, geralmente utilizadas para resolver o RCPSP.

## 2. Modelagem Proposta

O algoritmo proposto recebe como parâmetros de entrada os recursos humanos, as tarefas e outras propriedades do projeto, tais como o limite diário de horas extras por empregado e seu percentual de acréscimo sob as horas normais. Os recursos podem ser do tipo **Empregado**, que recebe mensalmente e exerce horas extras ou **Consultor**, que ganha a cada hora trabalhada. Os recursos possuem carga horária, salários, níveis de habilidades e disponibilidades individuais. As disponibilidades são registradas através da diferença entre sua carga horária e os períodos de indisponibilidade, onde cada período determina a data de início e fim e quantas horas o recurso estará indisponível.

Cada tarefa contém uma lista de predecessoras cujos tipos ligação podem ser Início-Início (II), Início-Final (IF), Final-Início (FI) e Final-Final (FF). Uma precedência pode existir associada a um tempo de latência, significando os dias de atraso da tarefa sucessora. A tarefa pode ser do tipo **Concreta, Marco e Duração Fixa**. Tarefa Concreta é aquela em que quanto maior o número de recursos alocados, menor é seu tempo de conclusão (desconsiderando o *overhead* de comunicação). Duração Fixa é um tipo de tarefa muito utilizada em reuniões que requer a mesma dedicação para todos os recursos

alocados e mantém fixa sua duração independente do tamanho da equipe. Marcos são atividades que registram eventos significativos e não possuem recursos associados.

Tarefas Concretas e de Duração Fixa exigem suas execuções por pessoas com habilidades específicas. Uma tarefa pode ter **Múltiplos Recursos** alocados e os recursos podem trabalhar **Paralelamente** em mais de uma atividade no mesmo dia. Na modelagem proposta, o gerente tem a opção de limitar a quantidade mínima e máxima de recursos alocados em cada tarefa. Acrescentamos um atraso nas tarefas Concretas referente ao *overhead* de comunicação entre os membros da equipe alocada. A nova duração da tarefa será  $t_{hh} \times \left(1 + ec \times \frac{N(N-1)}{2}\right)$ , onde  $t_{hh}$  é a duração estimada em Homens-Hora (HH),  $N$  é o tamanho da equipe e  $ec$  é o esforço médio de comunicação entre duas pessoas. Para maiores informações sobre *overhead* de comunicação consultar Penta et al. (2007).

A duração em HH de uma tarefa  $t_{hh}$  é calculada previamente pelo gerente e pode ser obtida pela fórmula  $t_{hh} = Esf(PF) \times Prod(HH/PF)$ , onde  $Esf$  é o esforço da tarefa em Pontos de Função (PF) e  $Prod$  é a produtividade média dos recursos nessa atividade. Logo, o algoritmo conhece apenas a estimativa de duração de cada tarefa em HH. A duração real da atividade será descoberta durante a construção do cronograma e vai depender de como forem alocados os recursos.

São três os objetivos propostos: (1) minimizar a duração do projeto (*makespan*); (2) minimizar o custo relativo às horas-extras dos empregados e às horas pagas aos consultores; e (3) maximizar a qualidade das equipes alocadas através do somatório dos níveis de habilidades dos recursos alocados vezes a dedicação do recurso para cada habilidade exigida pela tarefa. As funções são definidas na Equação 1 e sua legenda na Tabela 1. As horas normais pagas aos empregados são desconsiderados na segunda função dado o fato que eles recebem mensalmente, independentemente de quão ocupados eles estejam. Logo, a duração do projeto complementaria essa necessidade, uma vez que quanto maior a duração, maior o custo mensal com empregados.

$$\text{Tempo} = Makespan$$

$$\text{Custo} = \left( \sum_{e=1}^E \left( \sum_{t=1}^T \sum_{d=1}^D HE_{etd} \right) \times SH_e \right) \times pHE + \sum_{c=1}^C \left( \sum_{t=1}^T \sum_{d=1}^D Horas_{ctd} \right) \times SH_c \quad (1)$$

$$\text{Qualidade} = \sum_{r=1}^R \sum_{t=1}^T \sum_{h=1}^H NH_{rh} \times DT_{rt}$$

$d$ = Dia	$c$ = Consultor	$E$ = Total de Empregados	$HE$ = Hora Extra
$t$ = Tarefa	$h$ = Habilidade	$C$ = Total de Consultores	$SH$ = Salário-Hora
$e$ = Empregado	$T$ = Total de Tarefas	$D$ = Total de Dias	$NH$ = Nível Habilidade
$r$ = Recurso	$R$ = Total de Recursos	$H$ = Total de Habilidades	$DT$ = Dedicação Total
$Makespan$ = Dia de conclusão (número real)			$pHE$ = % de acréscimo à hora normal

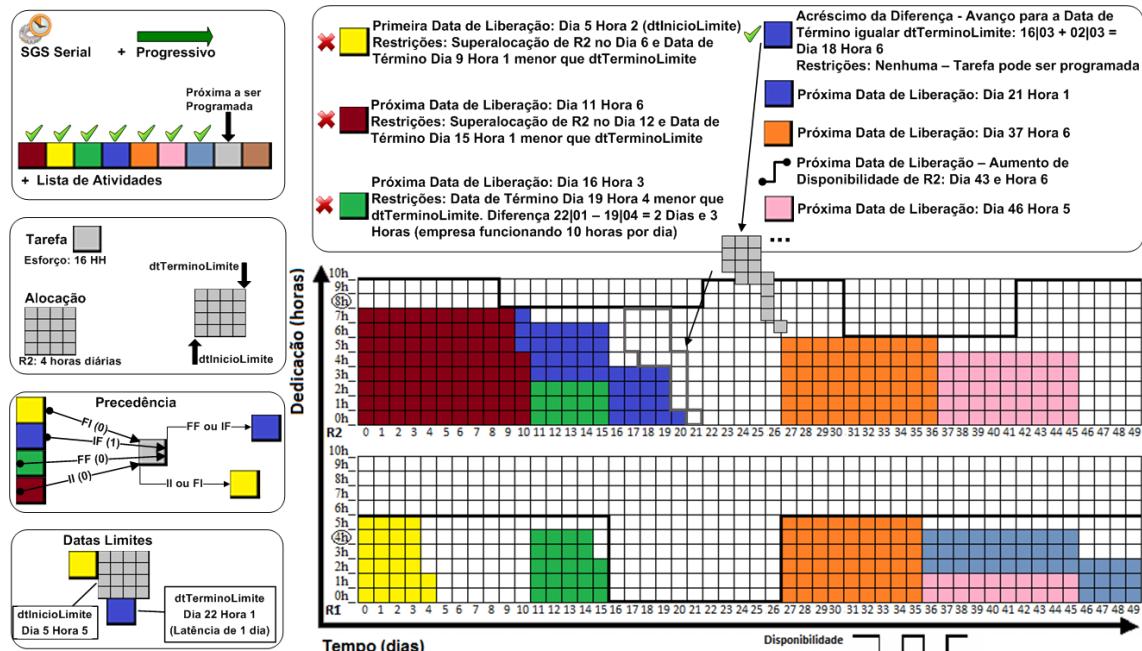
**Tabela 1. Legenda das Funções Objetivo**

A modelagem proposta utiliza como implementação base o NSGA-II. O cromossomo é representado pela **Matriz de Alocação** e pela **Lista de Atividades** dispostas na Figura 2. Uma célula da Matriz de Alocação representa a dedicação do

recurso na tarefa, que pode variar de zero até  $R_{ch} + E_{he}$ , onde  $R_{ch}$  é a carga horária do recurso e  $E_{he}$  é o limite de horas extras permitido pela empresa.

A Lista de Atividades apenas determina a ordem em que as atividades serão escalonadas pelo **Esquema de Geração de Escalonamento (SGS)**. O **SGS Serial** constrói o cronograma definindo o exato instante de início e término das tarefas, programando a atividade selecionada o mais cedo possível respeitando as restrições de recursos e precedências, segundo a ordem da Lista de Atividades definida pela metaheurística. Optamos pelo método SGS Serial pelo fato dele gerar um **Programa Ativo**, classe que contém uma solução ótima de construção de um cromossomo. Para maiores informações sobre SGS Serial consultar Simões (2004) - página 10.

Geralmente é realizada a verificação de cada unidade de tempo, partindo do primeiro dia do cronograma, até que a menor data possível da atividade ser programada seja encontrada. Esse procedimento pode ser muito custoso, principalmente se a unidade de tempo vertical utilizada estiver muito fragmentada, como em minutos, no nosso caso. Criamos então um conceito chamado de **Data de Liberação**, que armazena as datas em que houve conclusão de atividades ou aumento de disponibilidade dos recursos, ordenadas de forma crescente. Cada Data de Liberação contém uma lista de recursos associados. Se houve superalocação numa tentativa de programação de uma atividade numa determinada data, o próximo passo é tentar programá-la na próxima Data de Liberação em que o recurso está associado, pois apenas quando o recurso concluir uma atividade ou tiver sua disponibilidade diária aumentada é que será possível tentar programá-la novamente.



**Figura 1. Proposta para o Esquema de Geração de Escalonamento Serial**

Outra economia de processamento é a definição da primeira data de programação. A atividade deve ser programada pelo menos na data de término mais tarde das predecessoras FF e na data de início mais tarde das predecessoras II. A atividade também pode sofrer restrição pela sua data de término. Ela deve finalizar pelo menos na data de início mais tarde das predecessoras IF e na data de término mais tarde das predecessoras

FF. Quando isso ocorrer, a próxima tentativa de programação será na data em que a data de término da tarefa coincidir com a data da restrição. Uma visão geral do SGS Serial proposto pode ser visualizado na Figura 1.

Aplicamos o **Cruzamento de Dois Pontos** para a Matriz de Alocação, copiando colunas inteiras aos descendentes, evitando com isso geração de descendentes inválidos. Para a Lista de Atividades aplicamos o **Two-Point Precedence Set Crossover** proposto por Alcaraz and Maroto (2001). Esse tipo de cruzamento gera filhos mantendo a lista precedente-factível. Enquanto os cruzamentos são aplicados em pares de indivíduos a uma taxa previamente determinada, as mutações são aplicadas a cada *gene* do cromossomo também a uma taxa previamente definida. Um gene na Matriz de Alocação é uma célula e na Lista de Atividades uma posição no vetor. A mutação na Lista de Atividades desloca a atividade selecionada de forma que sua nova posição no vetor não fique antes que suas predecessoras nem depois que suas sucessoras. Essa operação mantém o sequenciamento precedente-factível. Uma observação importante é que essa modelagem sempre trabalha com soluções válidas.

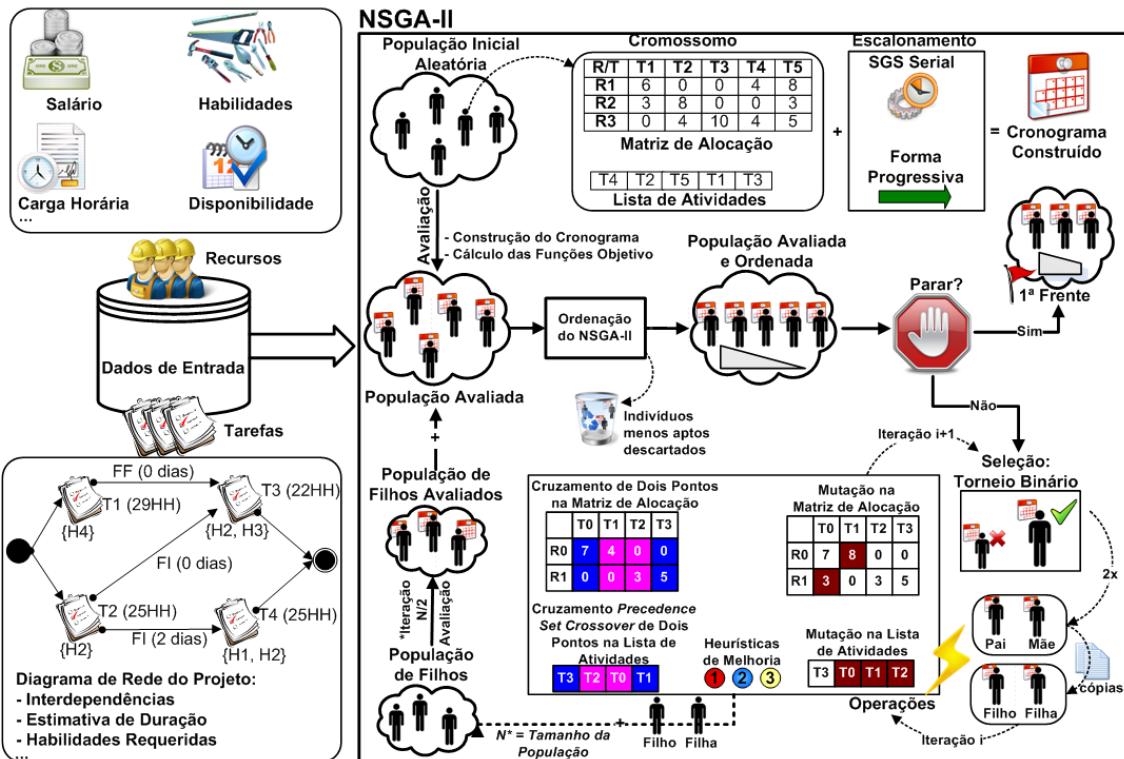
Além das operações de cruzamento e mutação, propomos aplicar as seguintes heurísticas de melhoria: aumentar as dedicações [gerando\não gerando] horas extras; redução de algumas dedicações que geraram horas extras; troca de dedicação entre membros da equipe [gerando\não gerando] horas extras; e probabilidade de não alocação durante a escolha aleatória de dedicação de um recurso. Esses heurísticas gulosas seriam aplicadas a uma taxa determinada *a priori* em cada indivíduo da população a fim de efetuar um ajuste fino na qualidade das soluções.

O algoritmo inicia gerando aleatoriamente a população inicial que é então ordenada pela não-dominância e pela diversidade. É iniciado um laço de  $N/2$  iterações, onde  $N$  é o tamanho da população. O pai é escolhido através do **Método do Torneio Binário**, onde dois indivíduos são escolhidos aleatoriamente e o melhor deles será o pai. A mãe segue o mesmo método de Seleção. Inicialmente o filho e a filha serão, respectivamente, as cópias do pai e da mãe escolhidos nessa iteração. Os pais são submetidos à operação de cruzamento e caso a operação realmente ocorra, os filhos serão alterados. Os filhos são então submetidos à mutação e às heurísticas de melhoria. Pelo conceito de elitismo, eles são adicionados a uma lista temporária. Terminado esse laço, os filhos são adicionados à população corrente e os que sofreram alterações são Avaliados. A população é novamente ordenada. Perceba que o acréscimo dos filhos excederá o limite  $N$  do tamanho da população. Logo, o processo de ordenação exclui os indivíduos menos aptos, eliminando aqueles situados nas últimas frentes e, dentro da última frente eliminada, os que possuem as piores medidas de diversidade. O processo continua até que o ponto de parada seja alcançado e, por fim, a primeira frente da população é retornada. A Figura 2 ilustra uma visão geral da modelagem.

### 3. Resultados Computacionais

A modelagem proposta foi aplicada em uma das iterações do projeto Sigecom, um projeto do SERPRO - Serviço Federal de Processamentos de Dados, a fim de avaliar a eficiência do algoritmo implementado. Nesta empresa, a elaboração do cronograma é realizada baseada no conhecimento do gerente do projeto e nas experiências em projetos anteriores.

Viabilizamos uma interface gráfica para cadastramento das Habilidades, dos



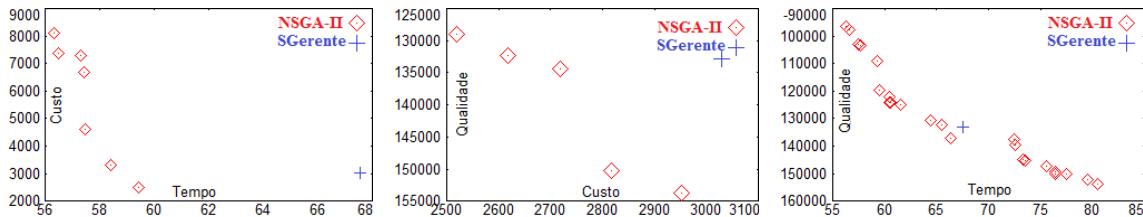
**Figura 2. Visão Geral da Modelagem Proposta**

Recursos e das Tarefas. Alimentamos a instância do projeto a partir de um Cronograma Planejado pelo Gerente (CPG) auxiliada pela ferramenta *MS Project*. A ideia dessa seção é comparar os cronogramas gerados pelo NSGA-II com o cronograma elaborado a partir da configuração (cromossomo) planejada pelo gerente (*SGerente*). A Matriz de Alocação de *SGerente* é obtida a partir da coluna *Recursos* e a Lista de Atividades através da ordem do campo *Data de Início* das atividades do CPG. Se houver atividades que iniciam na mesma data, escolhe a próxima atividade a ser programada com base nas precedências. Se ainda houver empate, escolhe aquela que foi inserida primeira no CPG. O projeto contém 64 Tarefas Concretas, uma de Duração Fixa e 11 Marcos. São 12 empregados, todos com o mesmo salário e mesma carga horária de 7 horas diárias. Dois deles possuem um período de indisponibilidade. Cada empregado possui suas próprias habilidades. O limite mínimo e máximo de recursos alocados foi registrado com base na particularidade de cada tarefa.

Executamos o NSGA-II com os parâmetros dispostos na Figura 3. A Figura 4 ilustra comparativos entre os resultados do NSGA-II e *SGerente*. Os objetivos foram combinados dois a dois para formar gráficos bidimensionais, permitindo uma melhor visualização dos resultados. Para que as melhores soluções entre pares de objetivos do NSGA-II sejam exibidas, novamente ordenamos a Frente de Pareto pelo critério de não-dominância considerando apenas dois objetivos por vez.

Percebemos que o cronograma construído a partir da configuração do gerente (*SGerente*) buscou alcançar valores intermediários em todos os aspectos. O NSGA-II, pela sua própria natureza, gerou resultados bem diversificados, dos quais pelo menos um

Tamanho da População	100	Prob. de Cruzamento da Alocação (%)	0.6
Prob. de Cruzamento do Escalonamento (%)	0.6	Prob. de Mutação da Alocação (%)	0.01
Prob. de Mutação do Escalonamento (%)	0.01	Prob. de Aplicar Heurísticas de Melhoria	0.8
Probabilidade de Não Alocar um Recurso	0.9	Redução de Horas Extra	0.01
Aumento de Dedicação com Horas Extras	0.01	Aumento de Dedicação sem Horas Extra	0.9
Troca de Dedicação com Horas Extra	0.01	Troca de Dedicação sem Horas Extra	0.01
Qtd Máxima de Gerações	500	Quantidade Máxima de Gerações sem Melhoria	200
Tempo de Execução Máximo (Minutos)	30	Qtd Máxima de Minutos Extras por Dia	120
Taxa Salarial Adicional do Minuto Extra (%)	0.5	Taxa de Comunicação Entre Dois Recursos	0.05
Funcionamento Diário da Empresa (Minutos)	600		

**Figura 3. Parâmetros de Entrada****Figura 4. Comparativo entre os Resultados do NSGA-II e SGerente**

deles foi melhor do que *SGerente* entre pares de objetivos. Mesmo quando comparamos os três objetivos simultaneamente, pelo menos uma solução do NSGA-II foi melhor que *SGerente*, como mostra a Tabela 2. Se *SGerente* fosse incorporada no conjunto de soluções geradas pelo NSGA-II, ela seria descartada pelo conceito de dominância.

Por fim, o cronograma completo da solução do NSGA-II da Tabela 2 foi avaliado pelo gerente e considerado mais viável do que o cronograma por ele construído, pelo fato de ter reduzido o tempo e o custo e agrupado os recursos mais habilidosos para cada tarefa. Além disso, a Frente de Pareto foi disponibilizada ao gerente que optou pelo cronograma com tempo de 57,47, apesar do custo ser de 4.600,00 e qualidade de 87.688,00.

Solução	Tempo	Custo	Qualidade
<b>SGerente</b>	67,55	3.094,50	132.874,00
<b>NSGA-II</b>	66,38	3.030,00	137.133,00

**Tabela 2. Exemplo de solução dominante do NSGA-II sobre SGerente**

#### 4. Conclusão e Trabalhos Futuros

Este trabalho propôs a utilização de uma abordagem otimizada híbrida que busque a melhor alocação e o melhor sequenciamento das atividades a fim de elaborar bons cronogramas que minimize o tempo e o custo e maximize a qualidade das equipes.

Foi proposta a utilização de técnicas chamadas de *Datas de Liberação* e *Limits das Datas de Precedência* a fim de evitar um custo muito alto de processamento durante o SGS Serial e permitir a utilização do minuto como unidade de tempo, obtendo uma modelagem mais flexível.

Vimos que, quando os objetivos são comparados em pares, existe pelo menos um resultado do NSGA-II superiores à solução construída com base na configuração

planejada pelo gerente (*SGerente*). Mesmo quando comparados todos os objetivos simultaneamente, existe pelo menos uma solução do NSGA-II que é melhor em todos os objetivos que *SGerente*.

A Elaboração de Cronograma é um problema muito difícil de ser resolvido devido à grande quantidade de características envolvida. Buscamos implementar as principais, tais como alocações em que mais de um recurso pode ser alocado por tarefa e um recurso por trabalhar em mais de uma tarefa no mesmo dia, limites de recursos por tarefa, disponibilidade, carga horária e salário individual, hora extra, *overhead* de comunicação, quatro tipos de ligação de dependência e tipos de tarefas e recursos humanos. No entanto, existem muitas outras características que podem ficar para trabalhos futuros, tais como replanejamentos, experiências profissionais adquiridas e produtividade individual.

## Referências

- E. Alba and J. Francisco Chicano. Software project management with gas. *Inf. Sci.*, 177(11):2380–2401, 2007. ISSN 0020-0255.
- J. Alcaraz and C. Maroto. A robust genetic algorithm for resource allocation in project scheduling. *Annals of Operations Research*, 102:83–109, 2001. ISSN 0254-5330. URL <http://dx.doi.org/10.1023/A:1010949931021>.
- G. Antoniol, M. D. Penta, and M. Harman. A robust search-based approach to project management in the presence of abandonment, rework, error and uncertainty. In *Proceedings of the Software Metrics, 10th International Symposium*, pages 172–183, Washington, DC, USA, 2004. IEEE Computer Society. ISBN 0-7695-2129-0. URL <http://portal.acm.org/citation.cfm?id=1018439.1021904>.
- G. Antoniol, M. D. Penta, and M. Harman. Search-based techniques applied to optimization of project planning for a massive maintenance project. In *In 21 st IEEE International Conference on Software Maintenance*, pages 240–249. IEEE Computer Society Press, 2005.
- C. K. Chang, H.-y. Jiang, Y. Di, D. Zhu, and Y. Ge. Time-line based model for software project scheduling with genetic algorithms. *Inf. Softw. Technol.*, 50:1142–1154, October 2008. ISSN 0950-5849. URL <http://portal.acm.org/citation.cfm?id=1405197.1405315>.
- F. Colares. Alocação de equipes e desenvolvimento de cronogramas em projetos de software utilizando otimização. Mestrado, Dissertação de Mestrado em Ciência da Computação da Universidade Federal de Minas Gerais, 2010.
- K. D. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm : NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, Apr. 2002. ISSN 1089778X. URL <http://dx.doi.org/10.1109/4235.996017>.
- M. D. Penta, M. Harman, G. Antoniol, and F. Qureshi. The effect of communication overhead on software maintenance project staffing: a search-based approach, 2007.
- J. Simões. Algoritmos baseados em busca tabu e busca tabu reativa para problemas generalizados de programação de projetos. Mestrado, Faculdade de Engenharia Elétrica e de Computação, UNICAMP, 2004.
- C. Tchao. Heurísticas para o problema de escalonamento de projetos com restrição de recursos. Mestrado, Dissertação de Mestrado em Ciência da Computação da Universidade Federal Fluminense, 2007.

# Desenvolvendo uma Abordagem Sistemática para Avaliação dos Estudos Experimentais em *Search-Based Software Engineering*

**Márcio de O. Barros<sup>1</sup>, Arilo C. Dias-Neto<sup>2</sup>**

<sup>1</sup> Programa de Pós-graduação em Sistemas de Informação – UNIRIO  
Av. Pasteur 458, Urca – Rio de Janeiro, RJ – Brasil

<sup>2</sup> Departamento de Ciência da Computação – DCC/UFAM  
Programa de Pós-graduação em Informática – PPGI

R. Gen. Rodrigo Octávio Jordão Ramos, 3000 – Setor Norte – Manaus, AM – Brasil  
[marcio.barros@uniriotec.br](mailto:marcio.barros@uniriotec.br), [arilo@dcc.ufam.edu.br](mailto:arilo@dcc.ufam.edu.br)

**Abstract.** In this paper, we present a discussion regarding how empirical studies in the context of Search-Based Software Engineering (SBSE) should be conducted to reduce the potential effects of a set of validity threats. An initial list of threats that can affect the results produced by empirical studies in SBSE is presented and an approach to support the evaluation of these studies is described. The approach was tested in papers published in the first two editions of International Symposium on Search-Based Software Engineering and initial results suggest that several validity threats are not yet properly addressed in the design of many SBSE experiments.

**Resumo.** Este artigo apresenta uma discussão a respeito de como estudos experimentais no contexto de pesquisas em Engenharia de Software Baseada em Buscas devem ser conduzidos visando à redução do impacto de ameaças à validade sobre os resultados observados nos estudos. Propomos uma lista inicial de ameaças à validade que podem afetar esses estudos e uma abordagem para apoiar a avaliação de experimentos na área. A abordagem foi aplicada nos artigos científicos publicados nas duas primeiras edições do International Symposium on Search-Based Software Engineering e os resultados observados sugerem que várias ameaças à validade ainda não são tratadas de forma apropriada no projeto de experimentos.

## 1. Introdução

*Search-based Software Engineering* (SBSE), ou Engenharia de Software Baseada em Buscas, é uma área de pesquisa que aplica algoritmos de otimização baseados em busca para automatizar a construção de soluções para problemas da Engenharia de Software [1][2]. SBSE relaciona as áreas de algoritmos e Engenharia de Software e sua literatura técnica mostra que a perspectiva prática da segunda área se sobrepõe à primeira.

Por conta desta visão prática, observa-se que muitas propostas de pesquisa em SBSE são avaliadas por meio de experimentação ao invés de análises teóricas. No entanto, o uso de experimentos para avaliar e comparar algoritmos é uma tendência recente na área de Computação. O mesmo pode ser dito para a área de Engenharia de Software, onde grande parte dos estudos experimentais foi publicada a menos de vinte anos [3]. Assim, experimentos em SBSE compartilham as limitações provenientes da imaturidade de ambas as áreas de origem. Entre estas limitações, identificamos a falta de uma lista de ameaças que podem afetar a validade dos resultados obtidos através da execução de estudos experimentais em SBSE. Por outro lado, estudos experimentais são

necessários para descobrir que algoritmos e heurísticas funcionam melhor em situações práticas e para avaliar sua eficiência, efetividade e escalabilidade, dentre outros aspectos, em cenários que simulem o tamanho e complexidade dos problemas enfrentados pela Engenharia de Software no mundo real.

Recentemente, Ali *et al.* [6] apresentaram uma revisão sistemática sobre como a investigação experimental tem sido conduzida na área de *search-based software testing* (SBST), uma subárea de SBSE. A revisão cobriu 64 artigos publicados entre 1996 e 2007 e abordou os algoritmos heurísticos mais aplicados e como estudos experimentais foram utilizados para obter evidências sobre a aplicação de tais algoritmos para propor soluções para problemas na área de teste de software. Os autores concluíram que o número de publicações contendo estudos experimentais bem projetados e bem reportados no domínio de SBST é muito pequeno [6]. Os autores também apresentaram uma lista inicial de ameaças à validade de estudos experimentais em SBSE que foi utilizada como base em nossa pesquisa.

Neste trabalho propomos uma lista de ameaças à validade para experimentos em SBSE baseada em um *framework* de experimentação freqüentemente utilizado na área de Engenharia de Software Experimental (ESE) [4]. Apresentamos ainda uma abordagem sistemática para avaliar o quanto um artigo científico na área de SBSE leva estas ameaças em consideração no projeto de seus estudos experimentais. Como mecanismo para avaliação preliminar da abordagem proposta, realizamos um *survey* nos artigos publicados nas duas primeiras edições do *International Symposium on Search-Based Software Engineering* (SSBSE), cobrindo 23 artigos entre 2009 (9 artigos) e 2010 (14 artigos). Nestes artigos, aplicamos a abordagem proposta para avaliar a abrangência e “qualidade” dos resultados obtidos em seus estudos experimentais. Resultados iniciais mostram que as principais limitações do projeto experimental dos estudos descritos nesses artigos estão relacionadas a ameaças à validade interna, externa e de constructo, enquanto que ameaças à validade de conclusão são abordadas de forma adequada em grande parte dos artigos.

Além desta introdução, este artigo é composto por 4 seções. A Seção 2 apresenta uma lista de ameaças à validade para experimentos em SBSE. A Seção 3 apresenta brevemente a proposta de uma abordagem sistemática para avaliar se um artigo científico aborda adequadamente as ameaças à validade identificadas na seção anterior. A Seção 4 apresenta os principais resultados obtidos a partir da aplicação da abordagem proposta sobre os artigos publicados nas duas primeiras edições do SSBSE. Por fim, a Seção 5 apresenta as conclusões e direcionamentos futuros para esta pesquisa.

## **2. Ameaças à Validade em Experimentos em SBSE**

Ameaças à validade são riscos relacionados com o projeto, a execução e a análise de resultados provenientes de estudos experimentais [4]. Essas ameaças podem limitar a capacidade do estudo em gerar resultados confiáveis ou que possam ser generalizados para uma população mais ampla do que a amostra utilizada para executar o experimento.

Nesta seção apresentamos uma lista de ameaças à validade que podem afetar estudos experimentais em SBSE. A lista segue o framework proposto por Wohlin *et al* [4], que divide as ameaças à validade de estudos experimentais em quatro categorias: conclusão, interna, construção e externa. As ameaças listadas nas próximas subseções são baseadas na experiência dos autores na área de Engenharia de Software Experimental, em

pesquisas que abordam a condução de estudos experimentais para avaliar algoritmos de busca (principalmente o trabalho de Johnson [5]) e uma lista anterior de ameaças à validade proposta por Ali *et al.* [6] (novas ameaças são marcadas com [+]). Uma lista detalhada das ameaças à validade pode ser encontrada em <http://www.seer.unirio.br/index.php/monografiasppgi/article/viewFile/1479/1307>.

## 2.1. Ameaças à Validade de Conclusão

Ameaças à validade de conclusão são associadas ao relacionamento entre o tratamento e os resultados. O projeto experimental deve garantir que existe um relacionamento estatístico entre as partes envolvidas. As principais ameaças de conclusão que podem afetar experimentos em SBSE incluem:

- **Não considerar a variação aleatória nos algoritmos:** uma única execução de um estudo experimental para uma dada instância gera resultados que podem carregar os benefícios de uma seleção inicial aleatória favorável ou os problemas de um ponto de partida ruim [5]. Portanto, todos os estudos experimentais devem ser executados várias vezes para cada instância;
- **Ausência de estatísticas descritivas:** visto que os estudos experimentais devem coletar dados a partir da execução de múltiplos ciclos para cada instância, esses dados devem ser sumarizados de forma adequada para permitir que conclusões possam ser traçadas [6]. Neste sentido, os estudos experimentais devem apresentar, ao menos, estatísticas descritivas de média e variância dos dados coletados;
- **Falta de uma base de comparação significativa:** pesquisas na área de SBSE geralmente comparam os resultados obtidos ao se executar dois ou mais algoritmos a fim de encontrar soluções para um determinado problema. Para que conclusões confiáveis possam ser extraídas quanto à necessidade de um mecanismo de busca mais complexo (por exemplo, heurístico) para resolver o problema em questão, o conjunto de algoritmos utilizado na comparação deve incluir estratégias de busca não-heurística e levar em consideração as melhores soluções conhecidas até então;
- **[+] Ausência de hipótese formal e validação por testes estatísticos:** a comparação descrita no parágrafo anterior deve ser baseada em uma hipótese formal (usualmente de que as médias dos resultados coletados pelos diferentes algoritmos são diferentes), que deve ser avaliada por um teste estatístico apropriado, de acordo com a distribuição dos dados observados e propriedades definidas no projeto experimental.

## 2.2. Ameaças à Validade Interna

Se um relacionamento entre o tratamento e os resultados é observado, o projeto do experimento deve garantir que é um relacionamento de causa e não o resultado de um fator sobre o qual o pesquisador não possui controle. As principais ameaças à validade interna que podem afetar experimentos em SBSE incluem:

- **Parâmetros definidos de forma não-sistemática:** os parâmetros selecionados para a técnica de busca proposta ou para os algoritmos utilizados na comparação com esta técnica não são explicitamente apresentados no projeto experimental ou são escolhidos de forma não sistemática;
- **[+] Falta de discussão sobre instrumentação do código:** o código-fonte usado no experimento pode esconder detalhes de implementação que podem favorecer certos algoritmos em detrimento de outros, influenciando nos resultados observados. Sendo

assim, o código-fonte deve ser discutido e, idealmente, disponibilizado para análise dos pesquisadores interessados no artigo;

- [+] **Falta de detalhamento sobre os procedimentos de coleta de dados:** os passos executados para coletar informações usadas na criação de instâncias do mundo real ou para gerar instâncias aleatórias devem ser descritos de forma precisa para garantir que esses aspectos não influenciem os resultados do experimento, por conta da seleção de instâncias favoráveis a certos algoritmos;
- [+] **Ausência de instâncias reais:** instâncias aleatórias podem ser úteis para avaliar o comportamento de uma nova técnica em situações particulares, que não sejam comumente encontradas em instâncias reais. Porém, elas podem carecer de propriedades encontradas em situações reais, possivelmente obscurecendo certos comportamentos dos algoritmos que se revelariam nestes cenários. Portanto, pesquisadores devem usar instâncias aleatórias combinadas com um conjunto razoável de instâncias reais nos estudos experimentais em SBSE [5].

### 2.3. Ameaças à Validade de Construção (*Constructo*)

Tais ameaças estão associadas às relações entre teoria e observação, garantindo que o tratamento reflete a construção da causa e que os resultados refletem a construção do efeito. As ameaças de construção que podem afetar um experimento em SBSE incluem:

- **Falta de avaliação da validade das medidas de custo:** por ser independente do hardware utilizado e da carga do sistema operacional, o número de avaliações da função objetivo (*fitness function*) é geralmente aceito como uma medida de custo adequada [5]. Quando o custo da execução de um algoritmo for medido de outra forma, o pesquisador deve justificar a métrica selecionada;
- **Falta de avaliação da validade das medidas de efetividade:** essas medidas devem ser relevantes para o problema e uma melhoria nos valores deve estar claramente relacionada a uma melhoria na qualidade da solução. Portanto, a seleção de medidas de efetividade apropriadas é relevante para garantir que os resultados observados no estudo experimental sustentam ou refutam a teoria proposta;
- [+] **Falta de discussão sobre o modelo submetido à otimização:** qualquer modelo que descreva um problema relacionado ao desenvolvimento e manutenção de projetos de software é uma simplificação do mundo real. Quando se manipula tais modelos para propor uma teoria, as limitações do modelo devem ser discutidas focalizando em como as simplificações adotadas influenciam sua aplicação prática.

### 2.4. Ameaças à Validade Externa

Essas ameaças estão associadas à generalização dos resultados observados em um experimento para uma população maior, além das instâncias que compõem a amostra usada no estudo. As principais ameaças externas de experimentos em SBSE incluem:

- [+] **Falta de definição das instâncias utilizadas:** nenhuma generalização é possível se os pesquisadores interessados no artigo não puderem entender as instâncias usadas em um estudo experimental. Portanto, qualquer projeto experimental deve prover uma clara definição das instâncias selecionadas;

- [+] **Ausência de uma estratégia de seleção de instâncias:** a pesquisa deve claramente citar como as instâncias usadas no experimento foram selecionadas, projetadas, geradas aleatoriamente ou coletadas a partir de problemas do mundo real;
- [+] **Falta de avaliações para instâncias de diferentes tamanhos e complexidades:** técnicas de Engenharia de Software são projetadas para lidar com sistemas e equipes que podem variar em tamanho e complexidade. Portanto, uma abordagem de SBSE deve ser avaliada para uma grande variedade de instâncias, com diferentes tamanhos e complexidades, a fim de avaliar os limites e escalabilidade da nova técnica.

### **3. Abordagem Sistemática para Avaliação de Ameaças à Validade de Estudos Experimentais em SBSE**

Dada a classificação das ameaças à validade dos resultados de estudos experimentais descritas na seção anterior, podemos avaliar como os artigos científicos lidam com cada classe de ameaça. Para facilitar esta avaliação, propomos um *framework* qualitativo e baseado em questionário. Em sua versão inicial, o questionário é composto por 17 questões, cada qual tratando de uma ameaça à validade. As questões são respondidas qualitativamente, indicando se o artigo provê uma avaliação Limitada (L), Parcial (P) ou Completa (C) da ameaça à validade endereçada pela questão.

A Tabela 1 lista as questões que compõem o questionário. Nem todas as possibilidades de resposta (L, P e C) estão disponíveis para todas as questões. Além disso, não se pode afirmar que essas questões provêm um *framework* completo para lidar com as ameaças à validade que podem afetar um experimento em SBSE: no estado atual desta pesquisa, elas representam a melhor compilação da experiência e problemas reportados nos artigos relacionados, porém sua efetividade ainda precisa ser avaliada. Mais detalhes sobre como estas questões podem ser respondidas estão disponíveis em <http://www.seer.unirio.br/index.php/monografiasppgi/article/viewFile/1479/1307>.

**Tabela 1. Questões que compõem o formulário de avaliação de experimento em SBSE.**

#ID	Questão
Q1	O experimento é executado várias vezes para cada instância?
Q2	Os dados coletados a partir do experimento estão bem organizados e resumidos?
Q3	Existe uma base ( <i>baseline</i> ) de comparação relevante?
Q4	As hipóteses do estudo estão formalmente apresentadas?
Q5	Testes de inferência estatística são usados?
Q6	O artigo discute o modelo submetido à otimização?
Q7	O artigo discute a validade das medidas de custo?
Q8	O artigo discute a validade das medidas de efetividade?
Q9	O artigo discute instrumentações realizadas no código fonte?
Q10	O código fonte usado no estudo está disponível para outros pesquisadores?
Q11	O artigo descreve o procedimento de coleta de dados usado no experimento?
Q12	O artigo apresenta uma clara definição das instâncias de destino?
Q13	O artigo apresenta claramente a estratégia de seleção de instâncias?
Q14	O artigo trata a variação no tamanho das instâncias usadas no experimento?
Q15	O artigo trata a variação na complexidade das instâncias usadas no experimento?
Q16	O artigo usa instâncias do mundo real no experimento?
Q17	O artigo apresenta os valores dos parâmetros usados no experimento?

## 4. Resultados Preliminares da Aplicação da Abordagem Sistemática

Como forma de avaliação da abordagem apresentada, foi feito um levantamento inicial dos artigos científicos publicados nas duas primeiras edições (2009 e 2010) do principal simpósio internacional da área (SSBSE – *International Symposium on Search-Based Software Engineering*). Ao total, 23 artigos foram avaliados, sendo 9 publicados em 2009 e 14 publicados em 2010. O levantamento baseado nestes artigos possibilitou uma avaliação do questionário proposto e forneceu um mapeamento inicial das pesquisas científicas na área de SBSE a ser usada como base para avaliações futuras. Esta seção descreve um resumo dos resultados obtidos.

Inicialmente, os artigos analisados foram classificados por domínio da Engenharia de Software e os resultados estão descritos na Tabela 2. Observa-se que metade dos artigos (50%) propõe algoritmos para apoiar teste de software. Análise de requisitos é o segundo domínio mais frequente (18%). Gerenciamento de projetos (12%), métricas (8%), controle de versão (4%), manutenção (4%) e arquitetura de software (4%) completam a lista dos domínios abordados pelos artigos analisados. É importante observar que a soma da coluna que representa os artigos de 2010 (17) contém um número maior que a quantidade de artigos publicados neste ano (14). Isto ocorre porque dois artigos abordam mais do que um domínio simultaneamente.

**Tabela 2. Domínios da Engenharia de Software Abordados nos artigos do SSBSE.**

Domínio da Engenharia de Software	2009	2010	Total	%
Controle de Versão	0	1	1	4%
Gerenciamento de Projetos	0	3	3	12%
Manutenção	0	1	1	4%
Análise de Requisitos	1	4	5	18%
Métricas	2	0	2	8%
Arquitetura de Software	1	0	1	4%
Teste de Software	5	8	13	50%
<b>TOTAL</b>	<b>9</b>	<b>17*</b>	<b>26</b>	<b>100%</b>

A Tabela 3 apresenta o número de artigos para cada categoria (Limitada, Parcial ou Completa) e cada tipo de ameaça à validade.

Em relação às ameaças à validade de conclusão, observamos que a maioria dos artigos publicados no SSBSE considera a variação aleatória em algoritmos de busca heurística, fornece um bom resumo dos dados coletados durante os experimentos usando estatísticas descritivas adequadas e usa algoritmos relevantes como base de comparação com as novas técnicas propostas. A maioria dos artigos que são exceções a estas regras, especialmente com relação a levar em consideração a natureza aleatória dos algoritmos de busca heurística, justifica um número reduzido de ciclos de execução dos experimentos pelo alto custo de execução de cada ciclo, tendo o algoritmo de interagir com seres humanos ou dispositivos de resposta lenta. Por outro lado, a proposição formal de hipóteses e sua avaliação através de testes de inferência estatística é uma área onde os experimentos em SBSE ainda podem ser aprimorados.

Analizando as ameaças à validade interna, observa-se que as principais limitações encontradas nos artigos publicados no SSBSE estão relacionadas à falta de uma discussão apropriada sobre a instrumentação do código-fonte, dos procedimentos de coleta dos dados e à ausência de instâncias reais nos experimentos. Sobre o código fonte e procedimento de coleta de dados, os artigos foram classificados como L

(avaliação limitada) quando nenhuma discussão sobre a implementação do algoritmo foi apresentada, P (parcial) quando o (pseudo-) código dos algoritmos foi apresentado, e C (completo) quando é código foi disponibilizado para outros pesquisadores. Sobre o uso de instâncias reais, os artigos que não usaram tais instâncias em seus estudos foram classificados como L; aqueles que usaram poucas instâncias reais (2 foi usado como patamar mínimo) foram classificados como P e os demais foram classificados como C. Como pode ser observado na Tabela 3, o uso de instâncias reais em experimentos em SBSE ainda é bastante limitado.

**Tabela 3. Avaliação das Ameaças à Validade em Experimentos em SBSE.**

Grupo de Ameaça	Item de Ameaça à Validade	L	P	C
Conclusão	Não considerar a variação aleatória nos algoritmos	6 (26%)	n/a	17 (64%)
	Ausência de estatísticas descritivas	2 (9%)	8 (35%)	13 (56%)
	Falta de uma base de comparação significativa	2 (9%)	4 (17%)	17 (74%)
	Ausência de hipótese formal e testes estatísticos	8 (35%)	3 (13%)	12 (52%)
Interna	Parâmetros definidos de forma não-sistemática	2 (9%)	0 (0%)	21 (91%)
	Falta de discussão sobre instrumentação do código	9 (39%)	5 (22%)	9 (39%)
	Falta de descrição de procedimentos de coleta de dados	4 (17%)	12 (52%)	7 (30%)
	Ausência de instâncias reais	9 (39%)	9 (39%)	5 (22%)
Construção	Falta de avaliação de validade de medidas de custo	10 (43%)	4 (17%)	9 (39%)
	Falta de avaliação de validade de medidas de efetividade	7 (30%)	8 (35%)	8 (35%)
	Falta de discussão do modelo submetido à otimização	2 (9%)	14 (61%)	7 (30%)
Externa	Falta de definição de instâncias utilizadas	1 (4%)	10 (43%)	12 (52%)
	Ausência de uma estratégia de seleção de instâncias	10 (43%)	0 (0%)	13 (57%)
	Falta de instâncias de diferentes tamanhos	9 (39%)	4 (17%)	10 (43%)
	Falta de instâncias de diferentes complexidades	12 (52%)	7 (30%)	4 (17%)

Analizando as ameaças à validade de construção, os principais problemas são relacionados à medida de custo selecionada e a descrição do modelo submetido à otimização. Em relação à medida de custo, artigos que utilizaram o número de avaliações da função objetivo foram classificados como C; artigos que justificaram o uso de outras medidas foram classificados como P; os demais foram classificados como L. Em relação ao modelo submetido à otimização, artigos relatando pesquisas em um domínio inexplorado por outros artigos da área de Engenharia de Software Baseada em Buscas e sem uma discussão sobre as limitações do modelo utilizado na otimização foram classificados como L; artigos que apresentaram uma solução para um problema recorrente sem discutir o modelo utilizado na otimização foram classificados como P; por fim, artigos foram classificados como C quando apresentaram um modelo do problema que abordam e discutiram suas limitações práticas.

Finalmente, analisando as ameaças à validade externa, os principais problemas estão relacionados à ausência de variações significativas de tamanho e complexidade nas instâncias utilizadas nos estudos. Como SBSE é uma área de pesquisa voltada para problemas de larga escala, onde é complexo encontrar soluções satisfatórias, analisar a escalabilidade de uma solução proposta é uma característica importante e desejada nos artigos científicos da área. No entanto, observa-se que poucos estudos reportam resultados para mais que 2 instâncias com diferentes tamanhos (43%) e complexidade (17%). Além disso, nem todos os estudos apresentam uma descrição clara das instâncias

selecionadas e da estratégia utilizada para medir seu tamanho e complexidade, de modo que se possa observar uma possível influência nos resultados coletados.

## 5. Conclusões e Trabalhos Futuros

Este artigo apresentou uma discussão preliminar a respeito da avaliação experimental de pesquisas na área de SBSE e o tratamento das ameaças à validade nos estudos experimentais que avaliam tais tecnologias. Foi apresentada uma versão inicial de uma lista de ameaças à validade que podem provocar impacto nos resultados da avaliação de um estudo experimental em SBSE, gerada a partir dos conceitos de Engenharia de Software Experimental. Também foi apresentada uma proposta de abordagem sistemática para avaliação da abrangência das descrições de ameaças à validade em estudos experimentais na área de SBSE. Esta abordagem foi aplicada na avaliação dos artigos publicados nas duas edições iniciais do *International Symposium on Search-Based Software Engineering* (2009 e 2010). Os resultados desta aplicação indicam limitações na descrição das ameaças à validade nestas pesquisas.

Este trabalho está em um estágio inicial e seus resultados não são conclusivos. Como trabalho futuro, será executada uma revisão sistemática para identificação e avaliação de artigos na área de SBSE com o propósito de expandir esta análise inicial e prover um amadurecimento da abordagem proposta. Além disso, serão investigadas novas ameaças à validade que podem afetar experimentos na área de SBSE para serem adicionadas ao conjunto inicial e, com isso, prover uma análise mais completa dos estudos a serem analisados.

## Agradecimentos

Os autores agradecem à FAPERJ, CNPq, INCT-SEC e FAPEAM pelo apoio financeiro provido para este projeto de pesquisa.

## Referências Bibliográficas

- [1] M. Harman, B.F. Jones, “Search-based Software Engineering”, *Information and Software Technology*, 2001, pp. 833-839
- [2] M. Harman, “The Current State and Future of Search Based Software Engineering”, *Future of Software Engineering (FOSE'07)*, 2007.
- [3] D.I.K. Sjoberg, J.E. Hannay, O. Hansen, V.B. Kampenes, A. Karahasanovic, N. Liborg, A.C. Rekdal. “A Survey of Controlled Experiments in Software Engineering”. In: *IEEE Transaction on Software Engineering*, Vol. 31, 9, 2005, pp. 733-753.
- [4] C. Wohlin, P. Runeson, M. Host, M.C. Ohlsson, B. Regnell, A. Wesslen, *Experimentation in Software Engineering: An Introduction*, Kluwer Academic Publishers, 2000.
- [5] D. S. Johnson, “A Theoretician's Guide to the Experimental Analysis of Algorithms”, *Data Structures, Near Neighbor Searches, and Methodology: Proceedings of the Fifth and Sixth DIMACS Implementation Challenges*, American Mathematical Society, Providence, 2002, 215-250
- [6] S. Ali, L. C. Briand, H. Hemmati, and R. K. Panesar-Walawege, “A Systematic Review of the Application and Empirical Investigation of Search-Based Test Case Generation”, *IEEE Transactions on Software Engineering*, vol. 36 (6), November/December 2010, 2010, pp. 742-762.

# Uma Proposta de *Framework* de Apoio à Seleção de Tecnologias de Software aplicando Estratégias de Busca

**Ariolo Claudio Dias Neto, Rosiane de Freitas Rodrigues**

Programa de Pós-Graduação em Informática – PPGI  
 Departamento de Ciência da Computação – DCC  
 Universidade Federal do Amazonas – UFAM  
 Manaus – Amazonas, Brasil

`{ariolo,rosiane}@dcc.ufam.edu.br`

**Abstract.** *The software technologies selection may affect the final product quality. Usually, several technologies are available to be applied in distinct software development activities. This scenario makes feasible the adoption of Combinatorial Optimization strategies as mechanism to support the generation of feasible solutions for this problem. This paper presents a proposal of modeling of software technologies selection problem as a Combinatorial Optimization problem (minimum dominating set) aiming attending different real scenarios in Software Engineering. This represents an initial step aiming the development of a framework supporting the software technologies.*

**Resumo.** *A seleção de tecnologias para projeto de software pode impactar diretamente na qualidade do produto final. Geralmente, diversas tecnologias estão disponíveis para serem aplicadas em diferentes atividades do processo de desenvolvimento, o que viabiliza a adoção de estratégias de Otimização Combinatória como apoio à geração de soluções para este problema. Este artigo apresenta uma proposta de modelagem do problema de seleção de tecnologias de software como um problema de Otimização Combinatória (menor conjunto dominante) visando atender a diversos cenários reais de Engenharia de Software. Este representa um passo inicial visando à construção de um framework de apoio à seleção de tecnologias de software.*

## 1. Introdução

A seleção de tecnologias a serem aplicadas em determinadas atividades do processo de desenvolvimento de software consiste em uma importante tarefa que pode afetar diretamente a qualidade do produto final. Esta tomada de decisão pode estar relacionada a tarefas de gerenciamento de projeto [Aranda *et al.*, 2006; Birk, 2007], de análise de requisitos [Maiden e Rugg, 1996], projeto arquitetural, de qualidade do software [Vegas e Basili 2005; Wojcicki e Strooper, 2007; Dias-Neto e Travassos, 2009], dentre outras. Em alguns casos, o número de opções de tecnologias possíveis de serem aplicadas no contexto de um projeto de software pode ser demasiadamente grande, dificultando a análise por um engenheiro de software de todas as possíveis soluções de forma a respeitar as restrições de tempo, custo e recursos em um projeto de software.

Como possível instrumento de apoio a este cenário, nos últimos 10 anos tem crescido bastante a adoção de técnicas de otimização combinatória como mecanismo de apoio às tomadas de decisões em determinados problemas de engenharia de software nos quais o número de opções disponíveis seja grande demais e difícil de ser analisado por engenheiros de software respeitando tais restrições de projeto. Esta área foi denominada de “Engenharia de Software Baseada em Buscas” (ou SBSE, do inglês *Search-Based Software Engineering*) [Harman, 2001].

O problema de seleção de tecnologias para projeto de software atende às duas características apresentadas por Harman (2001) para sua inclusão no cenário de SBSE:

(1) ele possui aspectos matemáticos que possibilitam a busca pela melhor solução e (2) existe uma grande quantidade de possibilidades para escolha. Em Dias-Neto *et al.* (2011), foi apresentada a modelagem do problema da seleção de técnicas de teste baseado em modelos para projetos de software como um problema em grafos conhecido como o problema do menor conjunto dominante (*minimum dominating set problem*) [Bondy e Murty, 2008; Garey e Johnson, 1979]. No entanto, esta modelagem poderia ser aplicada na seleção de tecnologias para diversos cenários da engenharia de software (ex: técnicas de teste, técnicas de elicitação de requisitos, metodologias de desenvolvimento de software, dentre outros) a partir da representação de características comuns a todos os cenários e a possibilidade de especialização de tal modelagem de forma a representar as especificidades de cada cenário.

Baseado neste contexto, este artigo apresenta uma proposta de modelagem do problema da seleção de tecnologias de software em geral como um problema de otimização combinatória. Esta modelagem representa um primeiro passo visando ao desenvolvimento de *framework* de apoio à seleção de tecnologias de software a partir da aplicação de técnicas de SBSE. Os benefícios esperados com a construção deste framework estão relacionados a dois cenários:

- (1) Apoiar às tomadas de decisões de engenheiros de software a respeito da seleção de tecnologias para projetos de software reais;
- (2) Apoiar pesquisas na área de SBSE e suas avaliações experimentais, provendo uma infraestrutura para instanciação de estratégias de seleção de tecnologias de software. Isso possibilitará avaliações das estratégias propostas a partir da aplicação de diferentes algoritmos de busca utilizando instâncias reais obtidas a partir da literatura técnica, reduzindo as ameaças às validades das avaliações de tais estratégias quando comparadas ao uso de dados fictícios.

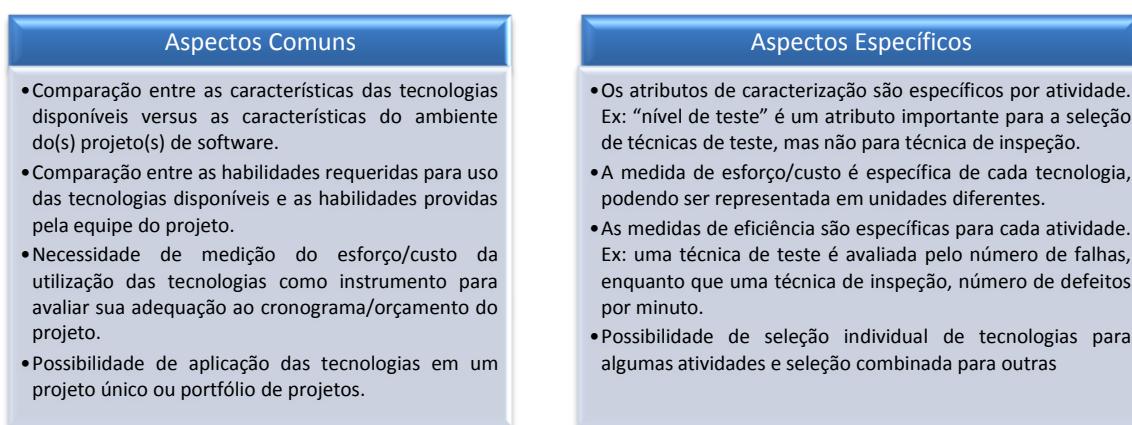
O presente artigo está estruturado da seguinte forma: a Seção 2 apresenta o problema da seleção de tecnologias de software de forma geral. A Seção 3 apresenta o meta-modelo desenvolvido para o problema de seleção de tecnologias, modelando-o como um problema de otimização combinatória. A Seção 4 apresenta a estrutura inicial de um *framework* de apoio à seleção de tecnologias de software. Por fim, na Seção 5 são feitas as considerações finais e próximos passos deste trabalho.

## **2. Seleção de Tecnologias de Software**

A seleção de tecnologias (ex: processos, técnicas, métodos, ferramentas) a serem usadas em um projeto de software é uma tarefa complexa que pode influenciar diretamente na efetividade do processo e qualidade do produto final [Vegas e Basili, 2005]. O principal desafio relacionado a este problema é entender e decidir pela seleção de qual(is) tecnologia(s) de software melhor se adéqua(m) a uma tarefa específica de um projeto.

O problema da seleção de tecnologias pode ser definido como a composição de dois problemas: (1) a completude do conjunto de tecnologias na qual a seleção será baseada e (2) a identificação apropriada das características que representam simultaneamente as tecnologias e o ambiente do projeto de software.

No processo de seleção de tecnologias, alguns aspectos são comuns ao processo de tomada de decisão independentemente da atividade de engenharia de software onde esta tomada de decisão está sendo requerida e outros são específicos da atividade em questão, conforme citado na Figura 1. Assim, um desafio para um framework de apoio à seleção de tecnologia de software seria prover as características comuns de seleção previstas em qualquer atividade do processo de desenvolvimento, além de possibilitar as especializações necessárias em cada atividade.



**Figura 1. Aspectos Comuns x Específicos da Seleção de Tecnologias.**

### 3. Meta-Modelo para o Problema de Seleção de Tecnologias representado como um Problema de Otimização Combinatória

O problema geral da seleção de tecnologias de software pode ser modelado como um problema de otimização combinatória cujo objetivo consiste na seleção da melhor tecnologia, ou o menor subconjunto possível dentre todas as tecnologias em consideração, que melhor atenda o(s) projeto(s) em questão, de acordo com suas características. Um ou mais critérios podem ser considerados para que este ‘melhor’ atendimento seja o mais representativo possível.

Além destes dois agentes principais – tecnologias e projetos de software -, os critérios a serem considerados para estimar a adequabilidade de aplicação da(s) tecnologia(s) ao(s) projeto(s) é um fator também relevante na qualidade e representatividade da solução proposta. Sendo assim, nesta seção são definidos os pontos principais a serem considerados na modelagem do problema, bem como apresentado um meta-modelo teórico que permite a representação adequada dos diversos cenários de seleção de tecnologias no contexto de engenharia de software.

#### 3.1. Identificação das Principais Dimensões a Considerar

No processo de modelagem do problema, três dimensões foram identificadas inicialmente como sendo relacionadas ao problema da seleção de tecnologias de software. Tais dimensões estão descritas na Figura 2, e as possíveis combinações resultarão em especializações do problema de seleção de tecnologias de software para diferentes (e até para um mesmo) cenários de engenharia de software.

##### 3.1.1. Modelagem para a Dimensão Origem (Individual x Combinada)

A **dimensão origem** aborda a questão de como as tecnologias disponíveis serão consideradas para serem selecionadas e aplicadas ao projeto ou portfólio de projetos em questão. Neste caso, há duas grandes opções a serem tratadas:

- **DOrig1 (seleção individual):** as tecnologias serão consideradas individualmente, e somente uma delas, baseando-se em suas características, será selecionada para aplicação no projeto/portfólio de projetos (dimensão destino);
- **DOrig2 (seleção combinada):** as tecnologias serão consideradas de maneira combinada de tal forma que mais de uma delas podem ser selecionadas, segundo suas características e adequação, para serem aplicadas simultaneamente no projeto/portfólio de projetos (dimensão destino).

### ► Origem (Seleção Individual x Seleção Combinada)

- Analisa a origem do processo de seleção, ou seja, as tecnologias disponíveis para serem selecionadas em um projeto/portfólio. Possui as seguintes opções:
  - Seleção individual: apenas uma tecnologia pode ser selecionada para cada projeto;
  - Seleção combinada: mais de uma tecnologia pode ser selecionada para um mesmo projeto.

### ► Destino (Projeto Individual x Protótipo de Projeto)

- Analisa o destino do processo de seleção, ou seja, o(s) projeto(s) de software que utilizarão as tecnologias selecionadas. Possui as seguintes opções:
  - Projeto individual: a seleção será destinada apenas a um projeto de software.
  - Portfólio de projetos: a seleção será destinada a um grupo de projetos, atendendo-os simultaneamente.

### ► Objetivo (Mono-objetivo x Multiobjetivo)

- Analisa o tipo de função objetivo a ser aplicado no problema da seleção de tecnologias para um cenário específico. Possui as seguintes opções:
  - Mono-objetivo: apresenta um único objetivo a ser otimizado, cuja função pode ser fruto de um único critério ou de diversos critérios combinados;
  - Multiobjetivo: apresenta mais de um a função-objetivo a serem otimizadas, de acordo com os critérios do problema (podendo ser conflitantes).

**Figura 2. Dimensões envolvidas no problema da seleção de tecnologias de software.**

#### 3.1.2. Modelagem para a Dimensão Destino (Projeto Individual x Portfólio)

A dimensão destino aborda a questão do cenário de aplicação da(s) tecnologia(s): se serão consideradas as características de um projeto único ou se serão considerados múltiplos projetos, cada projeto de maneira individual ou com identificação de características similares entre eles. Neste caso, há três grandes opções a serem tratadas:

- **DDest1 (projeto único):** as tecnologias serão selecionadas (dimensão origem) para serem aplicadas em um projeto único, baseando-se em suas características;
- **DDest2 (portfólio de projeto considerando cada projeto separadamente):** as tecnologias serão selecionadas (dimensão origem) para serem aplicadas em um portfólio de projetos a serem analisados separadamente.
- **DDest3 (portfólio de projeto considerando sobreposição de projetos - identificação de características similares entre projetos):** as tecnologias serão selecionadas (dimensão origem) para serem aplicadas em um portfólio de projetos a serem analisados a partir da sobreposição de suas características, ou seja, considerando que o conjunto de elementos que representa a dimensão destino seria formado pela interseção das características dos projetos de software que formam o portfólio de projetos.

#### 3.1.3. Modelagem para a Dimensão Objetivo (Mono-objetivo e Multiobjetivo)

A dimensão objetivo aborda a questão dos critérios a serem considerados para otimização, resultando em duas grandes abordagens de modelagem:

- **DObj1 (mono-objetivo):** apresenta um único objetivo a ser otimizado, cuja única função pode ser fruto de um único critério ou de múltiplos critérios combinados de forma ponderada a serem considerados na seleção de tecnologias aplicadas a projeto(s), resultando na determinação de uma solução ótima (ou melhor possível) segundo a função objetivo estipulada;
- **DObj2 (multiobjetivo):** apresenta mais de uma função objetivo a ser otimizada, de acordo com os critérios do problema (podendo ser conflitantes). Cada critério é representado por uma função objetivo específica e assim considerada separadamente, o que resulta na determinação não mais de uma solução ótima

(ou melhor possível), mas sim no conjunto de soluções de compromisso (não dominadas) entre todos os critérios (funções objetivo) envolvidos.

### 3.2. O problema da seleção de tecnologias como o problema do conjunto dominante mínimo e variações

O problema geral de seleção de tecnologias de software pode ser modelado como o problema da determinação do menor conjunto dominante em um grafo bipartido e (multi)ponderado.

**Modelo Geral: Menor Conjunto Dominante (grafo bipartido e (multi)ponderado):** dado um grafo  $G=(V,E)$  com a bipartição  $V_1, V_2 \in V$  e com pesos  $w_{ei}$ ,  $1 \leq i \leq m$ , associados a cada aresta  $e=\{v_1, v_2\}$ , onde  $v_1 \in V_1$  e  $v_2 \in V_2$ , determinar o menor conjunto dominante  $D \in V_1$  em  $G$ , tal que para todo vértice  $v \in V_2$  exista um vértice  $u \in D$  para o qual a aresta  $\{u, v\} \in E$  e que os pesos  $w_{ei}$  associados a cada aresta  $\{u, v\} \in E$  seja de melhor compromisso possível.

Neste modelo geral, cada uma das duas partições do grafo constitui um conjunto independente (os vértices da mesma partição não possuem arestas entre si). O processo de seleção também é geral, ou seja, deseja-se selecionar um subconjunto de vértices (um ou no máximo  $|V_1|$ ) que domine todo o conjunto de vértices da segunda partição.

Com base na identificação das dimensões do problema da seleção de tecnologias, que representam diferentes cenários neste processo, variações do modelo teórico geral acima são também propostas. Tais variações de problemas são discutidas a seguir.

Em Dias-Neto *et al.* (2011), é apresentada a modelagem do problema de seleção de técnicas de teste baseado em modelos (um subproblema da seleção de tecnologias de software) baseando-se no problema da determinação do menor conjunto dominante em um grafo bipartido e biponderado (considerando múltiplos objetivos divididos em dois grandes grupos de objetivos – versão biobjetivo). O problema consiste em encontrar um subconjunto  $D$  de tecnologias de software que “domine” ou “cubra” todos os atributos do projeto em avaliação, tal que este subconjunto de tecnologias possua o melhor compromisso entre a máxima cobertura dos atributos e mínimo esforço de construção/adaptação dos modelos a serem usados para geração dos testes.

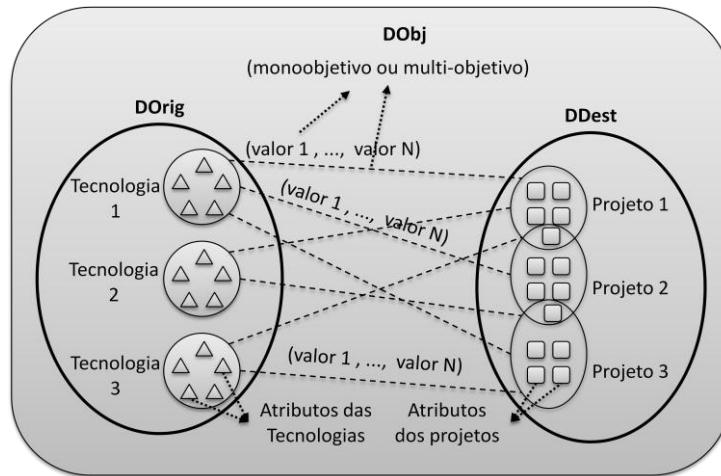
Tal modelagem proposta em Dias-Neto *et al.* (2011) trata os casos onde a dimensão origem é do tipo  $DOrig2$ , a dimensão destino é do tipo  $DDest1$  e a dimensão objetivo é do tipo  $DObj2$ , ou seja, modela o problema como um grafo bipartido e biponderado, onde a primeira partição é constituída por vértices que são componentes conexas (técnicas de teste) e a segunda partição é constituída por vértices que representam as características do projeto em consideração. Como visto, os vértices do modelo geral podem representar conjuntos de elementos (no caso, de características de tecnologias ou de projetos) e, neste caso, pode haver sobreposição entre tais conjuntos, que representam características similares entre as tecnologias ou projetos em consideração. Com base nestas diferenças, são propostos os modelos a seguir.

A Figura 3 a seguir apresenta o meta-modelo de onde podemos instanciar as diferentes combinações entre dimensões identificadas para o problema da seleção de tecnologias. Para instanciar os modelos considerando a dimensão  $DOrig1$  (primeira partição), que representa o caso onde apenas uma tecnologia de software é selecionada no processo, a dimensão destino pode ser do tipo  $DDest1$  (cada vértice é uma característica de um único projeto),  $DDest2$  (cada vértice representa um projeto, sendo portanto múltiplos projetos com suas características) e  $DDest3$  (cada vértice representa um componente conexo, podendo haver sobreposição entre tais componentes indicando características similares entre os projetos).

Ainda na Figura 3 podemos instanciar os modelos considerando a dimensão  $DOrig2$  (primeira partição), que representa o caso onde um subconjunto de tecnologias

pode ser selecionado no processo. Neste caso, o modelo deve permitir que na primeira partição arestas provenientes de mais de uma tecnologia possam se relacionar com os vértices da segunda partição. Assim como no caso anterior, neste caso a dimensão destino pode ser do tipo  $DDest1$ ,  $DDest2$  ou  $DDest3$ .

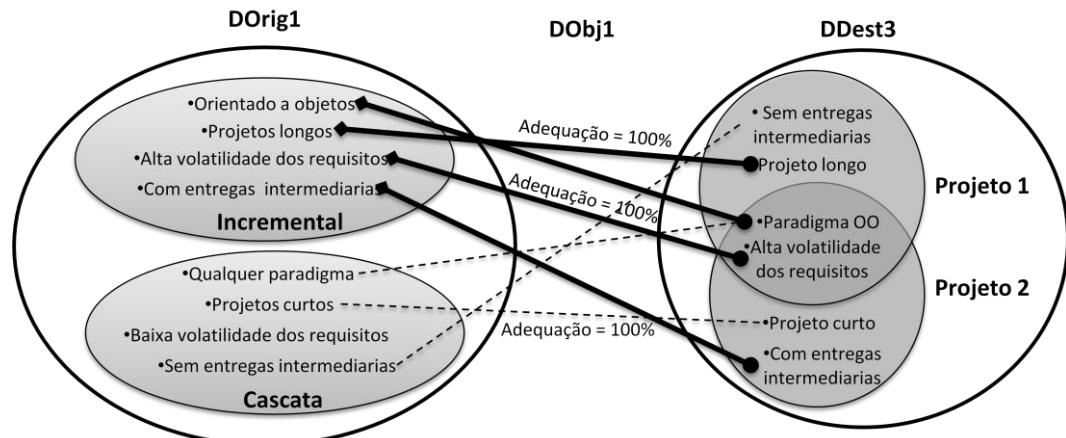
A Figura 3 ainda apresenta as considerações sobre a dimensão objetivo ( $DObj$ ), onde existem duas possibilidades:  $DObj1$  e  $DObj2$ . A dimensão  $DObj1$  representa o caso onde há apenas um peso associado a cada aresta do grafo bipartido, ou seja, apenas uma função objetivo é considerada. A dimensão  $DObj2$  representa o caso onde cada aresta é multiponderada, ou seja, cada critério que se deseja considerar durante a seleção de tecnologias é mapeado em uma função objetivo independente e terá seu valor representado nas arestas.



**Figura 3. Grafo bipartido representando o meta-modelo para o problema da seleção de tecnologias considerando as dimensões DOrig, DDest e DObj.**

### 3.3. Exemplo de Modelagem para a Seleção de Modelos de Ciclo de Vida

Esta seção descreve um exemplo simplificado de instânciação do meta-modelo apresentado na Figura 3 para o problema da seleção de modelos de ciclo de vida para um projeto de software. Para este exemplo, serão usadas as seguintes instâncias das 3 dimensões apresentadas:  $DOrig1$  (apenas 1 modelo de ciclo de vida será selecionado),  $DDest3$  (eles serão aplicados a 2 projetos de software) e  $DObj1$  (teremos apenas uma função objetivo indicando a adequação dos modelos de ciclo de vida ao projeto). A Figura 4 apresenta a instância do modelo para o exemplo descrito acima.



**Figura 4. Exemplo de modelo para o problema da seleção de modelos de ciclo de vida.**

Podemos observar que o ciclo de vida incremental (partição à esquerda) poderia ser o mais adequado por cobrir um maior número de atributos dos 2 projetos de software (partição à direita), podendo ser medido através do peso das arestas que os conectam.

A próxima seção descreve a proposta de *framework* que visa implementar esta modelagem utilizando diversos algoritmos de busca de forma a prover apoio à seleção de tecnologias em projetos de software.

#### **4. Proposta de *Framework* de Apoio à Seleção de Tecnologias de Software**

O *framework* a ser proposto tem sido projetado para viabilizar a aplicação de estratégias de busca como apoio ao problema da seleção de tecnologias de software. Assim, ele deve possibilitar a instanciação de uma infraestrutura computacional para atender diversos cenários da engenharia de software que requerem a seleção de tecnologias de software atendendo à modelagem apresentada na Seção 3.

A próxima subseção descreve os requisitos a serem implementados neste *framework*, como forma de viabilizar a seleção de tecnologias em projetos de software.

##### **4.1. Requisitos do *Framework* Projetado**

O *framework* proposto deve atender como requisitos aos itens:

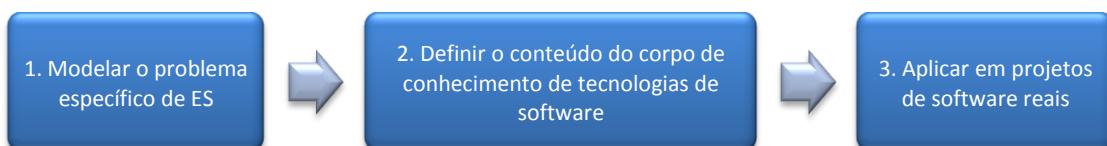
- prover uma modelagem comum ao problema da seleção de tecnologias de software seguindo a solução descrita na Seção 3.
- permitir a caracterização de tecnologias de software, indicando quais são os atributos usados como parâmetro para o processo de tomada de decisão e qual o peso de tais atributos neste processo.
- prover mecanismos para integração da implementação da modelagem do problema de seleção de tecnologias com os principais algoritmos de busca utilizados na área de SBSE, tais como: algoritmo genético (e suas variações), arrefecimento simulado (*simulated annealing*), busca tabu, subida da montanha (*hill climbing*), colônia de formigas, dentre outros. Assim, o *framework* pode ser evoluído com o tempo com a adição de novos algoritmos de busca.

A próxima subseção descreve o processo previsto para a aplicação do *framework* proposto em um cenário real de seleção de tecnologias em projetos de software.

##### **4.2. Processo para Instanciação de Infraestruturas**

O processo para instanciação de infraestruturas computacionais de apoio à seleção de tecnologias de software seguirá uma abordagem dividida em etapas, onde o *framework* proposto funcionará como um “meta-modelador” do processo de seleção de tecnologias e possibilitará a sua aplicação em diferentes cenários da engenharia de software.

Foram definidos três passos para esta instanciação, conforme a Figura 5:



**Figura 5. Representação Gráfica do Processo de Instanciação do *Framework* Proposto.**

- (1) **Modelar o problema específico de Engenharia de Software:** deverão ser definidos os atributos que caracterizam tanto as tecnologias de software a serem selecionadas e o projeto ou portfólio de projetos de software; serão informados os tipos de seleção a serem aplicados de acordo com as dimensões apresentadas na Seção 3: Objetivo, Origem e Destino.

- (2) **Prover o repositório de tecnologias:** as tecnologias a serem disponibilizadas para seleção em projetos devem ser providas a partir da literatura técnica, e o *framework* deve possibilitar a criação e manutenção deste repositório.
- (3) **Aplicar em projetos de software reais:** concluídos os passos anteriores, o *framework* será instanciado para aplicação em projetos reais informando os algoritmos a serem usados no problema e as características do(s) projeto(s) onde serão aplicadas as tecnologias.

## 5. Conclusões

Este trabalho abordou uma proposta de modelagem para o problema da seleção de tecnologias de software, como um problema de Otimização Combinatória baseado no menor conjunto dominante em um grafo bipartido e (multi)ponderado. Tal proposta representa um passo inicial visando ao desenvolvimento de um *framework* de apoio à seleção de tecnologias de software que implementa diversos algoritmos de busca de forma a ser instanciado para diferentes cenários da engenharia de software. A modelagem proposta aborda 3 diferentes dimensões que podem diferenciar a seleção de tecnologias em diferentes atividades da engenharia de software: origem, destino e objetivo.

Esta pesquisa está em fase inicial, e como próximos passos para sua evolução, já em andamento, está prevista a instanciação da modelagem proposta para cada cenário combinando as três dimensões apresentadas com o propósito de avaliar a adequação do modelo a estes cenários. Em seguida, será iniciada a construção do *framework* proposto implementando os requisitos e processo de instanciação citados na Seção 4.

## 6. Agradecimentos

Os autores agradecem ao INCT-SEC, CNPq e FAPEAM pelo apoio financeiro provido para realização desta pesquisa.

## 7. Referências Bibliográficas

- Aranda, G. N., Vizcaino, A., Cechich, A., Piattini, M. (2006), “Technology Selection to Improve Global Collaboration”, In: International Conference on Global Software Engineering (ICGSE), October, pp. 223-232.
- Birk, A. (1997), “Modeling the application domains of software engineering technologies”, In: International Conference on Automated Software Engineering (ASE). Lake Tahoe, CA, November.
- Bondy, J., Murty, U.(2008), Graph Theory: Graduated Text in Mathematics. Springer-Verlag.
- Dias-Neto, A.C.; Travassos, G.H. (2009). “Model-based Testing Approaches Selection for Software Projects”, In: Information and Software Technology, v. 51, p. 1487-1504.
- Dias Neto, A.C.; Rodrigues, R.F.; Travassos, G. H. (2011), Porantim-Opt: Optimizing the Combined Selection of Model-based Testing Techniques, In: 4th International Workshop on Search-Based Software Testing (SBST), March, Berlin, Germany.
- Garey, M., Johnson, D. (1979). Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman.
- Harman, M., Jones, B.F., Search-based software engineering, Information and Software Technology, 2001, pp. 833-839.
- Maiden, N. A. M.; Rugg, G. (1996), “ACRE: Selecting methods for requirements acquisition”, Software Engineering Journal 11(3): 183-192.
- Vegas, S.; Basili, V. (2005). “A Characterization Schema for Software Testing Techniques”, Journal of Empirical Software Engineering, v.10 n.4, p.437-466.
- Wojcicki, M. A.; Strooper, P. (2007). “An Iterative Empirical Strategy for the Systematic Selection of a Combination of Verification and Validation Technologies”. In: international Workshop on Software Quality (May 20 - 26, 2007).

# Utilizando Algoritmo de Otimização para Recomendação de Módulos em Projetos de Linhas de Produto de Software

Thaís Alves Burity Pereira, Gledson Elias

COMPOSE, Departamento de Informática  
Universidade Federal da Paraíba (UFPB) – Brasil

[thais@compose.ufpb.br](mailto:thais@compose.ufpb.br), [gledson@di.ufpb.br](mailto:gledson@di.ufpb.br)

**Abstract.** In order to demonstrate the diversity of scenarios to which techniques of Search Based Software Engineering could be applied, this paper describes a case study on the use of an approach to recommend modules in distributed Software Product Line projects. In such projects, dependencies between components can compromise the work of development teams. Thus, the clustering of tightly coupled components into modules to be developed by dispersed teams is an alternative to make their work more efficient.

**Resumo.** A fim de demonstrar a diversidade de cenários aos quais se aplicam técnicas de Engenharia de Software Baseada em Buscas, o presente artigo descreve um estudo de caso realizado sobre o uso de uma abordagem para recomendar módulos em projetos distribuídos de Linhas de Produto de Software. Em projetos dessa natureza, dependências entre componentes podem dificultar o trabalho das equipes de desenvolvimento. Dessa forma, o agrupamento de componentes fortemente acoplados em módulos para serem desenvolvidos por equipes remotas é uma alternativa para tornar mais eficiente o trabalho das mesmas.

## 1. Introdução

Linhas de Produto de Software (LPS) tem ganhado bastante espaço na indústria de software nos últimos anos, principalmente por promover o reuso de maneira sistemática e previsível, e oferecer apoio ao desenvolvimento de produtos para mercados globais [Paulish 2003]. No entanto, o desenvolvimento de LPS exige das organizações grande investimento inicial e a participação de profissionais qualificados, os quais nem sempre estão disponíveis localmente. Nesse cenário, o Desenvolvimento Distribuído de Software (DDS) pode ser empregado para encontrar especialistas do domínio e equipes mais qualificadas para o desenvolvimento de LPS. Além disso, essa forma de trabalho reforça alguns dos benefícios já oferecidos por LPS, tais como a redução de custo de desenvolvimento e o aumento da qualidade dos produtos. Apesar disso, abordagens de DDS também têm suas limitações, relacionadas principalmente à comunicação entre as equipes participantes de um mesmo projeto.

Partindo da premissa que dependências entre componentes exercem influência sobre a necessidade de comunicação entre suas respectivas equipes de desenvolvimento, em [Pereira 2011], propomos uma abordagem para identificar candidatos a módulos a ser desenvolvidos de forma (parcialmente) independente por equipes remotas, sendo um módulo um agrupamento de componentes. Para tal, a abordagem adota um algoritmo baseado na metaheurística de arrefecimento simulado (*simulated annealing*), tratando o agrupamento de componentes como um problema de otimização.

O presente artigo tem por objetivo apresentar um estudo de caso realizado para avaliar a referida abordagem, além de demonstrar o seu uso, e, dessa forma, introduzir

um cenário diferente ao qual se aplica o uso de técnicas da Engenharia de Software Baseada em Buscas. O restante do artigo está organizado da seguinte maneira. A Seção 2 apresenta os trabalhos relacionados. A Seção 3 descreve de forma geral a abordagem para recomendação de módulos. A Seção 4 detalha o estudo de caso desenvolvido em um projeto real de LPS. Por fim, a Seção 5 apresenta as considerações finais e os trabalhos futuros.

## 2. Trabalhos Relacionados

Nos últimos anos, diversas abordagens que lidam com o agrupamento de componentes foram propostas na literatura. Em sua maioria, tais abordagens consideram a dependência entre componentes com diferentes objetivos, por exemplo: (i) refinar um projeto de arquitetura de domínio [Blois *et al.* 2005]; (ii) minimizar a necessidade de comunicação entre equipes remotas e dedicadas à atividade de manutenção de software [Mockus e Weiss 2001]; e (iii) recuperar a arquitetura de software de um determinado sistema [Harman *et al.* 2005].

Para alcançar seus objetivos, tais abordagens se baseiam em informações de dependência extraídas a partir da implementação dos componentes, mas sem prover detalhes sobre como efetivamente realizar essa tarefa. Diferentemente, a abordagem aqui avaliada considera dependências a nível arquitetural e define métricas para esse fim. A maioria das abordagens também emprega algoritmos de otimização, sendo predominantemente usadas as metaheurísticas de subida da montanha (*hill climbing*), algoritmos genéticos e arrefecimento simulado. Dado que a metaheurística de subida da montanha produz resultados de melhor qualidade em comparação a algoritmos genéticos [Mancoridis *et al.* 1999], mas, ao mesmo tempo, tende a convergir prematuramente para um ótimo local e depende do ponto de partida da busca, a metaheurística de arrefecimento simulado tem se mostrado mais adequada, sendo a mesma adotada na abordagem avaliada no presente artigo.

Por fim, a função de custo adotada pelas abordagens de agrupamento de componentes baseadas em metaheurísticas apresenta várias limitações que as tornam inadequadas para o contexto em questão, como: admitir a existência de dependência entre um componente e ele próprio; basear-se em grafos não-ponderados, significando que a dependência entre componentes é pobramente avaliada; não considerar o tamanho dos agrupamentos, que é um aspecto importante uma vez que estes representam carga de trabalho para as equipes de desenvolvimento; e fixar previamente o número de agrupamentos que devem ser gerados, que é desconhecido no contexto em questão.

## 3. Abordagem para Recomendação de Módulos

A abordagem para recomendação de módulos possui três objetivos fundamentais: (i) mensurar o grau de dependência em nível arquitetural entre componentes de um projeto de LPS; (ii) identificar módulos que possam ser implementados e integrados por equipes remotas com reduzida necessidade de comunicação, a partir do agrupamento de componentes dependentes; e (iii) mensurar o grau de dependência entre módulos.

A fim de alcançar os objetivos especificados, a abordagem está organizada em três etapas, conforme ilustrado pela Figura 1. A etapa de *medição de dependências entre componentes* lida com a tarefa de orientar o projeto de uma DSM (*Design Structure Matrix*), que deve refletir a arquitetura do projeto com base nas dependências entre componentes. Nesse sentido, essa etapa define as métricas baseadas em coesão e acoplamento denominadas *dependência por interfaces*, *por operações* e *por features*. As duas primeiras métricas estão mais relacionadas ao acoplamento entre componentes,

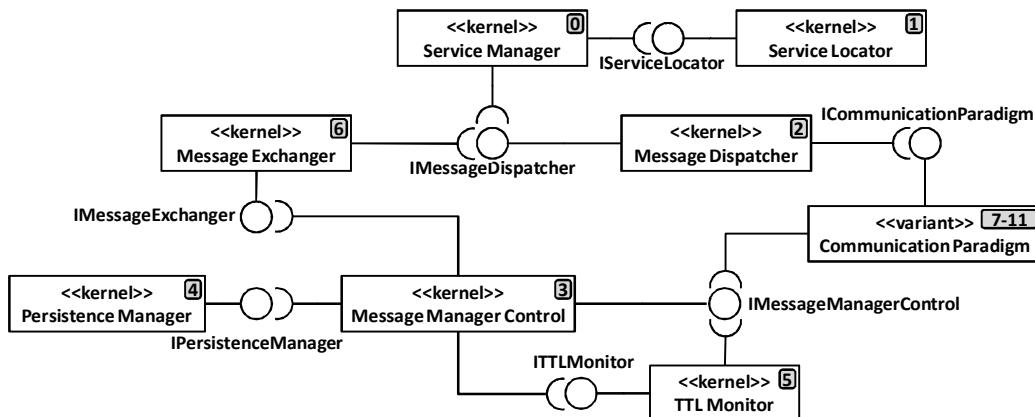


as equipes com maiores facilidades de comunicação são alocadas aos módulos com maiores interdependências.

#### 4. Estudo de Caso: Multi-MOM

Nesta seção é apresentado um estudo de caso realizado com o projeto Multi-MOM [Morais e Elias 2010], cujo propósito é avaliar o uso a referida abordagem. O Multi-MOM é uma LPS de *middleware* para comunicação em computação móvel. A

Figura 3 descreve a arquitetura de domínio desta LPS, na qual os componentes estão enumerados para que possam ser identificados nos artefatos. O componente *Communication Paradigm* consiste em um ponto de variação para o qual há cinco componentes variantes, que não são excludentes entre si ( $c_7-c_{11}$ ). Essa representação genérica abstrai as diferenças entre os componentes variantes, sendo adotada apenas para prover uma representação simplificada da arquitetura.



**Figura 3. Arquitetura de domínio do Multi-MOM.**

Considerando as interfaces providas e requeridas, mostradas na

Figura 3, foi realizado o cálculo da métrica de dependência por interfaces. E, com base na distribuição de *features*, mostrada na Tabela 1, foi calculada a métrica de dependência por *features*. O uso combinado dessas métricas resultou na construção da DSM mostrada na Figura 4. A solução de agrupamento que foi encontrada é descrita pela CMM da Figura 5a e pela DSM reordenada da Figura 5b.

**Tabela 1. Relação de *features* por componente definida pelo arquiteto do projeto.**

Índice	Componente	Feature
$c_0$	Service Manager	Localização de serviços
$c_1$	Service Locator	Localização de serviços
$c_2$	Message Dispatcher	Reflexão
$c_3$	Message Manager Control	Buffer de envio
$c_4$	Persistence Manager	Persistência de mensagens
$c_5$	TTL Monitor	Monitor de TTL
$c_6$	Message Exchanger	Troca de mensagens pela rede
$c_7$	Communication Paradigm – Publish/Subscribe	Publish/Subscribe Buffer de envio
$c_8$	Communication Paradigm – Message Queue	Message Queue Buffer de envio
$c_9$	Communication Paradigm – Tuple Space	Tuple Space Buffer de envio
$c_{10}$	Communication Paradigm – PTP Notification	PTP Notification Buffer de envio
$c_{11}$	Communication Paradigm – Synchronous	Synchronous Buffer de envio



Pela solução do arquiteto, o agrupamento dos componentes *Service Manager* ( $c_0$ ) e *Service Locator* ( $c_1$ ) é confirmado como sendo adequado pela definição do módulo  $M_2$  na Figura 6b. Também se mostrou adequado o agrupamento no módulo  $M_0$  do componente *Message Dispatcher* ( $c_2$ ) e todas as variantes do componente *Communication Paradigm – Publish/Subscribe* ( $c_7$ ), *Message Queue* ( $c_8$ ), *Tuple Space* ( $c_9$ ), *PTP Notification* ( $c_{10}$ ) e *Synchronous* ( $c_{11}$ ). Em ambas as soluções, o componente *Message Exchanger* ( $c_6$ ) manteve-se em um módulo individual.

No entanto, os componentes *Message Manager Control* ( $c_3$ ), *Persistence Manager* ( $c_4$ ) e *TTL Monitor* ( $c_5$ ), que constituem o componente composto *Message Manager*, e que na solução do arquiteto são agrupados juntos formando o módulo  $M_1$ , foram mantidos separados na solução encontrada pelo algoritmo, o que não constitui uma boa solução, já que componentes que constituem um componente composto tendem a ser fortemente dependentes entre si.

A fim de compreender o resultado obtido pelo algoritmo, foi feita uma análise mais detalhada da DSM do projeto (Figura 4), sendo identificado que esta não expressa uma forte dependência entre os componentes  $c_3$ ,  $c_4$  e  $c_5$ :  $DSM_{c_3, c_4} = DSM_{c_4, c_5} = DSM_{c_5, c_4} = 0$ ,  $DSM_{c_4, c_3} = DSM_{c_5, c_3} = 0,22$  e  $DSM_{c_3, c_5} = 0,67$ . Na solução encontrada pelo algoritmo, o componente  $c_3$  é agrupado com o componente *Message Dispatcher* ( $c_2$ ) e com os componentes variantes ( $c_7 - c_{11}$ ), dado os elevados níveis de dependência entre  $c_3$  e os componentes variantes:  $DSM_{c_3, c_7} = DSM_{c_3, c_8} = DSM_{c_3, c_9} = DSM_{c_3, c_{10}} = DSM_{c_3, c_{11}} = 0,83$  e  $DSM_{c_7, c_3} = DSM_{c_8, c_3} = DSM_{c_9, c_3} = DSM_{c_{10}, c_3} = DSM_{c_{11}, c_3} = 0,11$ . Isso sugere que houve uma falha no projeto arquitetural, especificamente no mapeamento de *features* para os componentes, que implicou em uma baixa dependência entre os componentes na DSM.

Diante dessa questão, foi proposta uma avaliação do projeto arquitetural, onde foram confirmadas as falhas no mapeamento das *features*. Como resultado da correção do projeto arquitetural, o componente *Message Manager Control* ( $c_3$ ) passou a incluir as *features* *Persistência de mensagens* e *Monitor de TTL*, já que os componentes *Message Manager Control* ( $c_3$ ), *Persistence Manager* ( $c_4$ ) e *TTL Monitor* ( $c_5$ ) constituem o componente composto *Message Manager* e o componente  $c_3$  é responsável por controlar a funcionalidade desse componente composto. Além disso, cada componente variante do tipo *Communication Paradigm* passou a incluir a *feature* *Paradigma de comunicação*, denotando que tais componentes estão relacionados à mesma funcionalidade geral.

Após a correção do projeto arquitetural, a DSM foi reconstruída e o algoritmo foi reaplicado. A partir da nova DSM, foi encontrada a solução mostrada na DSM reordenada da Figura 7. Por sua vez, a solução do arquiteto de acordo com a DSM corrigida é ilustrada na Figura 8. Conforme pode ser observado, a nova solução mostra-se bem mais coerente com a solução definida pelo arquiteto, exceto pelo fato do componente *Message Exchanger* ( $c_6$ ) também ser agrupado com os componentes *Message Manager Control* ( $c_3$ ), *Persistence Manager* ( $c_4$ ) e *TTL Monitor* ( $c_5$ ), enquanto que na solução do arquiteto, o componente  $c_6$  deve ficar isolado em um módulo. Dado que o componente  $c_6$  possui relação de dependência apenas com os componentes *Message Dispatcher* ( $c_2$ ) e  $c_3$ , e que o módulo  $M_0$  que contém  $c_2$  é grande para ainda incluir  $c_6$ , pelo valor do CTC foi detectado que seria melhor manter  $c_6$  agrupado com  $c_3$ ,  $c_4$  e  $c_5$  do que mantê-lo isolado em um módulo individual.

Dado que o valor de CTC referente à solução do algoritmo é menor, é possível concluir que na perspectiva do objetivo de maximizar as dependências internas aos módulos e minimizar as dependências externas, a solução encontrada pelo algoritmo é



## 5. Considerações Finais

Considerando os resultados obtidos no estudo de caso relatado, o uso da abordagem se mostrou bastante útil, uma vez que oferece uma solução alternativa melhor, atendendo ao seu objetivo fundamental de prover recomendações e assim, auxiliar na tomada de decisão. Além disso, de forma inesperada, a abordagem também se mostrou adequada para identificar falhas de projeto, tal como no caso descrito, que resultou na reavaliação do mapeamento das *features* para os componentes.

Vale ressaltar que a alocação das equipes de desenvolvimento aos módulos recomendados não é o propósito da abordagem avaliada. A alocação é suportada pelo *framework* de recomendações para alocação de equipes de desenvolvimento em projetos distribuídos de LPS, apresentado em [Pereira *et al.* 2010], sendo a abordagem aqui avaliada apenas a primeira etapa do *framework*.

Como trabalho futuro é planejado o desenvolvimento de uma ferramenta para dar suporte ao uso da abordagem, que deve permitir que o usuário crie suas próprias soluções, bem como faça modificações nas soluções recomendadas, sendo informado sobre a qualidade das soluções construídas e das modificações realizadas. Em seguida, é planejada a realização de mais estudos de caso, com projetos de maior complexidade e tamanho, a fim de realizar uma avaliação mais completa da abordagem.

**Agradecimentos.** Este trabalho foi apoiado pelo Instituto Nacional de Ciência e Tecnologia para Engenharia de Software (INES)<sup>1</sup> e financiado pelo CNPq, processo 573964/2008-4.

## Referências

- Blois, A. P., et al. (2005) “Towards a components grouping technique within a domain engineering process”, 31th Euromicro Conference, Component-based Software Engineering Track, Portugal, pp.18-25.
- Harman, M., et al. (2005) “An empirical study of the robustness of two module clustering fitness functions”, Conference on Genetic and Evolutionary Computation, Hans-Georg Beyer (Ed.), pp. 1029-1036.
- Mancoridis, S., et al. (1999) “*Bunch: a clustering tool for the recovery and maintenance of software system structures*”, IEEE International Conference on Software Maintenance. IEEE Computer Society, pp. 50-59.
- Mockus, A., Weiss, D. M. (2001) “Globalization by chunking: a quantitative approach”, IEEE Software. Vol. 18, Issue 2, pp. 30-37.
- Morais, Y., Elias, G. (2010) “Integrating communication paradigms in a mobile middleware product line”, 9th International Conference on Networks, IEEE Computer Society, pp. 255-261.
- Paulish, D. J. (2003) “Product line engineering for global development”, 3rd International Workshop on Product Line Engineering The Early Steps: Planning, Modeling, and Managing, pp. 17-22.
- Pereira, T. A. B., et al. (2010) “A recommendation framework for allocating global software teams in software product line projects”, 2nd International Workshop on Recommendation Systems for Software Engineering, pp. 36-40.
- Pereira, T. A. B. (2011) “Uma abordagem para recomendação de módulos para projetos de desenvolvimento distribuído de linhas de produto de software”, Dissertação de Mestrado, Universidade Federal da Paraíba.

---

<sup>1</sup> <http://www.ines.org.br/>

ISSN: 2178-6097



XXV Simpósio  
Brasileiro de  
Engenharia de  
Software

XV Simpósio  
Brasileiro de  
Linguagens de  
Programação

XIV Simpósio  
Brasileiro de  
Métodos  
Formais

V Simpósio  
Brasileiro de  
Componentes,  
Arquiteturas e  
Reutilização de  
Software

Promoção



Realização



Patrocínio



Ministério da  
Educação



Organização



[www.each.usp.br/cbsoft2011](http://www.each.usp.br/cbsoft2011)