APLICANDO TÉCNICAS DE BUSCA MULTIOBJETIVAS NA PRIORIZAÇÃO DE REQUISITOS DE SOFTWARE

Márcia Maria Albuquerque Brasil
Universidade Estadual do Ceará (UECE)
Av. Paranjana, 1700 – 60.740-903 – Fortaleza – CE
marcia.abrasil@gmail.com

Thiago Gomes Nepomuceno da Silva
Universidade Estadual do Ceará (UECE)
Av. Paranjana, 1700 – 60.740-903 – Fortaleza – CE
thi.nepo@gmail.com

Fabrício Gomes de Freitas
Universidade Estadual do Ceará (UECE)
Av. Paranjana, 1700 – 60.740-903 – Fortaleza – CE
fabriciogf.uece@gmail.com

Thiago do Nascimento Ferreira
Universidade Estadual do Ceará (UECE)
Av. Paranjana, 1700 – 60.740-903 – Fortaleza – CE
thiagonascimento.uece@gmail.com

Mariela Inés Cortés
Universidade Estadual do Ceará (UECE)
Av. Paranjana, 1700 – 60.740-903 – Fortaleza – CE
mariela@larces.uece.br

Jerffeson Teixeira de Souza
Universidade Estadual do Ceará (UECE)
Av. Paranjana, 1700 – 60.740-903 – Fortaleza – CE
jeff@larces.uece.br

RESUMO

No planejamento de um projeto de desenvolvimento de software, decidir quais e em que ordem os requisitos devem ser desenvolvidos nem sempre é uma tarefa fácil. Um planejamento eficiente deve atender às expectativas dos clientes e respeitar às restrições existentes. Nesse contexto, este trabalho apresenta uma abordagem baseada em otimização multiobjetivo para o problema de priorização de requisitos de software. A estratégia visa à satisfação de clientes e o gerenciamento de riscos, tratando dependências entre requisitos e limitações de recursos. Experimentos utilizando metaheurísticas multiobjetivas foram realizados.
PALAVARAS CHAVE. Priorização de Requisitos de Software, Engenharia de Software Baseada em Busca, Metaheurísticas Multiobjetivas.

ABSTRACT

During the planning of a software development project, the decision about which requirements to develop and the implementation order is not always an easy task. An efficient planning must address customers' expectations and comply with constraints. In this context, this work presents an approach based on multiobjective optimization to the software requirements prioritization problem. The strategy aims at customer satisfaction and risk management, as well as provides ways for handling requirements dependencies and available resources. Experiments were conducted using multiobjective metaheuristics.
KEYWORDS. Software Requirements Prioritization. Search-Based Software Engineering. Multiobjective Metaheuristics.

1. Introdução

Cada vez mais, as empresas são pressionadas a se manterem competitivas em um ambiente de negócios em constantes mudanças, desenvolvendo para isso novas estratégias que as diferencie de seus concorrentes, tais como a oferta de novos produtos e serviços (AURUM e WOHLIN, 2005). Nesse contexto, o software tornou-se parte integrante de quase todas as operações de negócio (SOMMERVILLE, 2007), sendo assim um ativo estratégico para muitas organizações.

Uma vez que os produtos de software exercem um papel fundamental nos processos de negócios, é imprescindível que funcionem de acordo com as necessidades e objetivos para os quais foram desenvolvidos. Muitas vezes, a qualidade de um produto de software é determinada pela sua capacidade em atender às necessidades dos clientes e usuários (BERANDER e ANDREWS, 2005). Portanto, um importante desafio para as empresas de software durante o desenvolvimento de um sistema é a correta identificação e o entendimento dos principais requisitos de negócio a fim de garantir que os produtos a serem desenvolvidos atendam às expectativas de seus clientes.

No entanto, dado um determinado conjunto de funcionalidades a serem desenvolvidas e considerando o fato de que nem sempre há recursos suficientes, decidir quais requisitos devem ser implementados e disponibilizados aos clientes não é uma tarefa fácil. Outra dificuldade consiste em planejar e definir em que ordem os requisitos serão desenvolvidos considerando que as prioridades de cada cliente devem ser atendidas, que cada requisito tem um risco associado (o qual pode impactar no projeto como um todo) e que os requisitos podem estar relacionados entre si de alguma forma.

Assim, realizar um planejamento adequado é uma atividade importante e complexa que envolve diversos aspectos e decisões que visam atender às necessidades dos clientes e respeitar as restrições existentes. Normalmente, o planejamento é feito de modo *ad hoc* (RUHE e SALIU, 2005) ou informal, sendo realizado unicamente através do conhecimento de gerentes e baseado em experiências anteriores (COLARES et al., 2009).

Neste trabalho, o problema de selecionar e priorizar um conjunto de requisitos de software é modelado como um problema de otimização. Assim, deseja-se selecionar um conjunto ideal de requisitos a serem desenvolvidos de forma a agregar maior valor de negócio à organização e deseja-se, ainda, que os requisitos que com maior prioridade estabelecida pelos clientes e maior risco associado sejam ser implementados primeiro.

Da mesma forma que em outras disciplinas, técnicas de otimização têm sido aplicadas com sucesso na resolução de problemas da Engenharia de Software. O trabalho aqui desenvolvido insere-se no contexto de uma área de pesquisa recente e em crescente desenvolvimento denominada *Search-Based Software Engineering – SBSE* (HARMAN e JONES, 2001), ou Otimização em Engenharia de Software. Trata-se de uma área que tem por finalidade a aplicação de técnicas de otimização, notadamente as metaheurísticas, na resolução de problemas difíceis da Engenharia de Software, muitas vezes caracterizados pela existência de restrições, objetivos conflitantes e grandes espaços de busca.

Assim, este trabalho apresenta as seguintes contribuições:

- Apresenta uma formulação multiobjetiva para resolver o problema de Seleção e Priorização de Requisitos de Software. A abordagem leva em consideração importantes aspectos inerentes ao contexto de projetos reais de software, tais como: satisfação de clientes (baseada nos conceitos de importância e prioridade), valor de negócio, gerenciamento de riscos, recursos disponíveis e interdependências entre requisitos;
- Utiliza metaheurísticas multiobjetivas baseadas em algoritmos genéticos para resolver instâncias do problema;
- Indica o desempenho dos algoritmos, incluindo além das soluções geradas, o tempo de execução; ainda, compara a qualidade das soluções geradas com a qualidade de soluções obtidas através de um algoritmo randômico.

Além desta seção introdutória, o trabalho é organizado conforme segue: Seção 2, que

contextualiza o processo de Engenharia de Requisitos e a área de pesquisa denominada SBSE; Seção 3, onde são descritos alguns trabalhos relacionados à Priorização de Requisitos de Software; Seção 4, a qual apresenta importantes aspectos considerados na modelagem do problema e define formalmente a abordagem proposta; Seção 5, que apresenta alguns conceitos de otimização multiobjetivo e descreve os algoritmos utilizados neste trabalho; Seção 6, que reporta os resultados de uma avaliação inicial conduzida para demonstrar a eficácia dos algoritmos utilizados na resolução do problema; e, Seção 7, onde são feitas as considerações finais do artigo e apresentadas oportunidades para trabalhos futuros.

2. Fundamentação Teórica

2.1. Engenharia de Requisitos

Requisitos são descrições de como um sistema de software deve se comportar. Todo projeto começa com a atividade de levantamento de requisitos. Os requisitos são definidos durante os estágios iniciais do desenvolvimento do sistema e consistem em uma especificação do que deve ser implementado. Portanto, um requisito é um conjunto de necessidades que devem ser atendidas. Para se ter sistemas de software de alta qualidade, desenvolvidos dentro do prazo e do orçamento previstos, é imprescindível que as especificações de requisitos sejam compreensíveis, abrangentes e consistentes, dentro de um processo estruturado e controlado (AURUM e WOHLIN, 2005).

Stakeholders são as pessoas que serão afetadas pelo sistema e que influenciam direta ou indiretamente nos requisitos da aplicação; incluindo clientes, usuários finais, gerentes, líderes de projeto, desenvolvedores, entre outros (SOMMERVILLE e SAWYER, 1997). Desta forma, a participação dos *stakeholders* nas atividades da Engenharia de Requisitos é de grande importância, dado que a correta identificação das suas necessidades determina, em grande parte, a qualidade do produto de software. Portanto, os requisitos devem ser negociados e validados antes de serem documentados e iniciada a sua implementação (AURUM e WOHLIN, 2005).

Engenharia de Requisitos é o termo usado para designar todas as atividades relacionadas à descoberta, documentação e manutenção de um conjunto de requisitos para um sistema (SOMMERVILLE e SAWYER, 1997). O processo de Engenharia de Requisitos é um dos principais fatores para o sucesso de projetos de software em um mercado global competitivo onde o tempo de entrega e o atendimento aos requisitos dos clientes são fatores-chave de sucesso (AURUM e WOHLIN 2005). A Engenharia de Requisitos visa, na sua essência, transformar as necessidades (possivelmente incompletas, inconsistentes e contraditórias) dos *stakeholders* em um completo conjunto de requisitos de alta qualidade (AURUM e WOHLIN, 2005).

2.2. *Search-Based Software Engineering*

Durante o processo de desenvolvimento de software, engenheiros de software normalmente se deparam com problemas difíceis que envolvem o equilíbrio entre restrições concorrentes e o conflito de interesses, em que a melhoria de um determinado aspecto implica em tornar outro pior, obrigando a se fazer uma escolha a partir da análise da situação como um todo. Assim, encontrar soluções ótimas para esses problemas é impossível ou inviável; por outro lado, encontrar soluções próximas de uma solução ótima ou que estejam dentro de uma faixa aceitável é admissível em problemas da Engenharia de Software (HARMAN e JONES, 2001).

O termo *Search-Based Software Engineering* surgiu em 2001 no trabalho de HARMAN e JONES (2001) para designar uma nova e promissora área de pesquisa caracterizada pela utilização de técnicas de otimização na resolução de problemas da Engenharia de Software. Muitos problemas da Engenharia de Software, devido a sua natureza complexa, podem ser reformulados como um problema de busca tornando-se, assim, adequados para a aplicação de algoritmos de otimização, notadamente metaheurísticas como Algoritmos Genéticos (GOLDBERG, 1989), *Simulated Annealing* (LAARHOVEN e AARTS, 1987) e Busca Tabu (GLOVER e LAGUNA, 1997).

A reformulação de um problema da Engenharia de Software como um problema de busca (possibilitando assim a aplicação de técnicas de otimização baseadas em busca) requer

(HARMAN e JONES, 2001) (CLARKE et al., 2003): a escolha de uma representação do problema, que consiste na definição da estrutura de dados a ser utilizada para representar soluções para o problema; e a definição de uma função de *fitness* (ou função objetivo), que consiste na definição de uma função que avalie a qualidade das soluções geradas.

É possível identificar uma vasta aplicação de SBSE em diversas atividades do ciclo de vida de desenvolvimento de software. Dentre essas atividades, destacam-se trabalhos aplicando técnicas de otimização em: Engenharia de Requisitos, Alocação de Recursos e Estimativas de Custo, Garantia da Qualidade, Decisões de Projeto e Código-Fonte, Manutenção e Engenharia Reversa, Testes, dentre outras. A competitividade humana de SBSE é demonstrada em um trabalho recente de SOUZA et al. (2010), através de um consistente e abrangente estudo experimental, onde voluntários especialistas em Engenharia de Software resolveram instâncias de quatro problemas clássicos em SBSE. Uma análise comparativa dos resultados sugere que as soluções obtidas através das metaheurísticas podem ser ditas "humano-competitivas".

3. Trabalhos Relacionados

KARLSSON e RYAN (1997) desenvolveram uma abordagem baseada em custo/valor para priorizar requisitos, utilizando o método *Analytic Hierarchy Process* – AHP (SAATY, 1980) para realizar uma comparação em pares entre os requisitos e avaliar sua importância relativa baseado no valor e custo de implementação de cada requisito. Porém, a grande quantidade de comparações necessárias torna a abordagem proibitiva quando o número de requisitos aumenta. JUNG (1998) apresentou uma variante do problema da mochila 0-1 para reduzir a complexidade da abordagem custo/valor. Em KARLSSON, WOHLIN e REGNELL (1998) são descritas e comparadas diversas abordagens para priorização de requisitos, incluindo o método AHP; no entanto, de acordo com o trabalho, tais abordagens apresentam complexidade elevada. Além disso, não contemplam as perspectivas dos *stakeholders* nem as limitações de recursos.

Na área de SBSE, a seleção de requisitos foi tratada inicialmente por BAGNALL, RAYWARD-SMITH e WHITTLEY (2001) no trabalho intitulado "*The Next Release Problem*" (NRP) ou Problema do Próximo *Release*. A abordagem consiste em selecionar um conjunto ótimo de clientes, cujos requisitos deverão ser atendidos no próximo *release* do sistema, priorizando os clientes mais importantes e levando em consideração tanto os recursos disponíveis, como a precedência técnica entre os requisitos (quando a implementação de um requisito pressupõe a implementação prévia ou conjunta de outro requisito). Diversas técnicas foram empregadas na resolução do problema, incluindo Programação Linear Inteira, Algoritmos Gulosos, GRASP, *Simulated Annealing* e *Hill Climbing*. Com uma formulação mono-objetiva, o planejamento não leva em consideração a importância que os requisitos têm para os clientes.

Uma formulação multiobjetiva para o problema do próximo *release* é apresentada por ZHANG, HARMAN e MANSOURI (2007). A satisfação de clientes e o custo total do projeto são os objetivos a serem otimizados. Para selecionar o conjunto ótimo de requisitos, a abordagem considera a importância de cada cliente para a empresa, a importância do requisito para os clientes e o custo de desenvolvimento do requisito. No entanto, a formulação não inclui qualquer forma de interdependência entre requisitos, algo incomum no contexto de projetos reais, uma vez que um requisito pode depender de outro de diferentes formas (CARLSHAMRE et al., 2001). Três metaheurísticas (*Pareto GA*, *Single-Objective – Weighted – GA* e NSGA-II) são aplicadas na resolução do problema e mais um algoritmo de busca randômica. Em DURILLO et al. (2009), é feito um estudo da versão multiobjetiva do NRP, onde as metaheurísticas NSGA-II e MOCell são comparadas através de métricas de desempenho e uma análise das soluções obtidas é realizada em termos de qualidade, quantidade de soluções, número de soluções ótimas e intervalo de soluções cobertas pelas frentes geradas.

Em DEL SAGRADO, DEL ÁGUILA e ORELLANA (2010), o conjunto de técnicas aplicadas ao problema do próximo *release* é ampliado e uma adaptação do algoritmo ACS (*Ant Colony System*) é aplicada ao problema. Além desta metaheurística, Algoritmos Genéticos e *Simulated Annealing* também foram utilizados na resolução do problema através de um estudo comparativo que avaliou a qualidade das soluções obtidas. No trabalho, não foram consideradas

interdependências entre os requisitos.

Ainda, ZHANG et al. (2010) apresentaram um modelo de análise de importância para balancear as necessidades atuais e futuras dos requisitos quando do planejamento do próximo *release* do sistema. Assim, o problema de otimização modelado consiste em três objetivos: satisfazer as necessidades atuais (baseada na importância atual de cada requisito), satisfazer as necessidades futuras (baseada na importância futura de cada requisito) e reduzir o custo total do projeto. O modelo proposto não apresenta restrições, nem considera interdependências entre requisitos. A metaheurística NSGA-II foi utilizada na resolução do problema.

ZHANG, FINKELSTEIN e HARMAN (2008) apresentam uma visão geral sobre SBSE para a Engenharia de Requisitos.

4. Definição do Problema

4.1. Aspectos Considerados

Esta seção descreve importantes aspectos e definições relacionados à abordagem proposta e que foram considerados na formulação matemática do problema.

4.1.1. Requisitos e Interdependências

Seja $R = \{r_i | \ i = 1, 2, \dots, N\}$ o conjunto de requisitos a serem desenvolvidos.

O desenvolvimento de cada requisito r_i demanda certo custo e tempo denotados por $cost_i$ e $time_i$, respectivamente. O custo e o tempo de desenvolvimento normalmente são estimados pela organização desenvolvedora. Fatores que influenciam essa estimativa incluem a complexidade do requisito, documentação e testes necessários, infra-estrutura, entre outros. Aqui, o custo e o tempo de implementação de cada requisito são estimados em uma escala de 10 a 20.

As escalas numéricas aqui utilizadas são apenas uma forma de avaliar valores de importância, prioridade, risco, custo, tempo etc. e permitir a modelagem matemática do problema. Outras escalas podem ser usadas em diferentes contextos sem perda de generalidade.

Cada requisito r_i também possui um risco associado, representado por $risk_i$, que pode ser definido em termos da análise de impacto do risco para o negócio do cliente *versus* a sua probabilidade de ocorrência. Assim, o risco de cada requisito varia em uma escala de 1 (mais baixo risco) a 9 (mais alto risco) e é quantificado conforme a matriz na Tabela 1 a seguir.

Tabela 1 – Quantificação do Risco através da Análise de Impacto *versus* Probabilidade de Ocorrência

Impacto no Negócio	Probabilidade		
	Baixo	Médio	Alto
Baixo	*Baixo-Baixo* **(1)**	*Baixo-Médio* **(2)**	*Baixo-Alto* **(3)**
Médio	*Médio-Baixo* **(4)**	*Médio-Médio* **(5)**	*Médio-Alto* **(6)**
Alto	*Alto-Baixo* **(7)**	*Alto-Médio* **(8)**	*Alto-Alto* **(9)**

Relacionamentos entre requisitos ou interdependências afetam o desenvolvimento de software de várias formas (DAHLSTEDT e PERSSON, 2005). De acordo com um estudo publicado em CARLSHAMRE et al. (2001), 75% das interdependências vêm de aproximadamente 20% dos requisitos. O tipo de relacionamento considerado neste trabalho é a precedência técnica entre requisitos. Essa forma de relacionamento pode ser identificada quando a implementação de certo requisito pressupõe o prévio desenvolvimento de um ou mais requisitos. Assim, se um requisito r_i precede tecnicamente um requisito r_j, o requisito r_i deve ser implementado antes do requisito r_j, ou seja, r_i é pré-requisito para r_j.

4.1.2. *Stakeholders*

Stakeholders exercem um importante papel na seleção e priorização de requisitos. Seja $C = \{c_m | \ m = 1, 2, \dots, M\}$ o conjunto de *stakeholders* envolvidos com o desenvolvimento do sistema, cujos requisitos deverão ser atendidos. Para cada *stakeholder* c_m, é associado um peso

baseado na sua importância relativa para a organização. Portanto, w_m define a importância de um *stakeholder* para a empresa e é quantificada em uma escala de 1 (baixa importância) a 10 (alta importância). Neste trabalho, o conjunto de *stakeholders* considerado envolve apenas clientes.

4.1.3. Projeto

O desenvolvimento do produto consiste de todos os requisitos a serem selecionados e priorizados durante o planejamento do projeto. Assim, há um limite máximo no orçamento, denotado por $budgetProject$, e na duração, denotada por $timeProject$, totais do projeto, os quais não devem ser excedidos.

4.1.4. Requisitos *versus Stakeholders*

Diferentes *stakeholders* possuem diferentes interesses com a implementação de cada requisito. Assim como em GREER e RUHE (2004) e em RUHE e SALIU (2005), os conceitos de prioridade, em termos de urgência para a implementação de um requisito; e valor, em termos de valor de negócio agregado, são utilizados neste trabalho. Estes conceitos são analisados do ponto de vista de cada cliente e de sua importância para a organização.

Assim, $value(m,i)$ quantifica a importância que um *stakeholder* c_m associa a um requisito r_i, atribuindo um valor que varia de 1 (baixa importância) a 10 (alta importância).

De forma similar, $priority(m,i)$ denota a urgência que um *stakeholder* c_m tem para a implementação de um requisito r_i. Essa prioridade é expressa em valores de 1 (baixa prioridade) a 10 (alta prioridade).

4.2. Abordagem Proposta

Na abordagem proposta, os requisitos são selecionados e ordenados (ou priorizados) de acordo com os objetivos de valor, prioridade e risco; ao mesmo tempo em que devem respeitar a precedência técnica existente entre os requisitos e as limitações de orçamento e duração do projeto. O objetivo dessa estratégia é selecionar para desenvolvimento os requisitos considerados mais importantes do ponto de vista dos clientes mais significativos e estabelecer uma ordem de implementação, onde os requisitos que apresentarem maior risco e os requisitos com maior prioridade devem ser implementados primeiro. Devido às limitações de recursos, é possível que nem todos os requisitos sejam selecionados.

Assim, essa abordagem tem como objetivos:

- Maximizar o valor de negócio agregado, selecionando para desenvolvimento os requisitos considerados mais importantes, analisados do ponto de vista dos clientes mais representativos;
- Maximizar a satisfação de clientes, de forma ponderada, implementando mais cedo os requisitos considerados mais prioritários;
- Minimizar os riscos do projeto como um todo, implementando inicialmente os requisitos com maior risco associado.

E as seguintes restrições:

- Respeitar a precedência técnica existente entre os requisitos;
- Respeitar os limites de tempo e orçamento disponíveis para o projeto.

4.3. Formulação Matemática

A partir do que foi apresentado nas subseções anteriores, o problema é formulado matematicamente com as seguintes funções-objetivo e restrições.

(1) $$Max\ f_{SCORE}(y) = \sum_{i=1}^{N} importance_i . y_i$$

(2) $$Max\ f_{PRIORITY}(x^{Prio}, y) = \sum_{i=1}^{N} prior_i . (N - x^{Prio}{}_i) . y_i$$

(3) $$Min\ f_{RISK}(x^{Prio}) = \sum_{i=1}^{N} \left((risk_i . x^{Prio}{}_i + 1) + 9 . N . (1 - y_i) \right)$$

Sujeito a:

(4) $\sum_{i=1}^{N} cost_i . y_i \leq budgetProject$

(5) $\sum_{i=1}^{N} time_i . y_i \leq timeProject$

(6) $x^{Prio}{}_i < x^{Prio}{}_j$, $sempre\ que\ r_i\ precede\ tecnicamente\ r_j$

A variável $x^{Prio}{}_i$ indica a posição do requisito r_i, sendo um valor em $\{0, 1, 2, \ldots N-1\}$, na ordem para a implementação estabelecida pela priorização, para $i = 1, 2, \ldots N$. A variável y_i indica se o requisito r_i será implementado ($y_i = 1$) ou não ($y_i = 0$), para $i = 1, 2, \ldots N$.

Função (1) – Expressa a satisfação dos clientes pela implementação dos requisitos considerados mais importantes, onde: $importance_i = \sum_{m=1}^{M} w_m . value(m, i)$ representa, de forma ponderada, o valor de negócio agregado pela implementação do requisito r_i, da mesma forma que nos trabalhos de ZHANG, HARMAN e MANSOURI (2007) e ZHANG (2010).

Função (2) – Expressa, de forma ponderada, a satisfação dos clientes pela implementação antecipada dos requisitos considerados mais prioritários, onde: $prior_i = \sum_{m=1}^{M} w_m . priority(m, i)$.

Função (3) – Requisitos com um alto risco associado são mais propensos a ocasionarem problemas no desenvolvimento (SOMMERVILLE e SAWYER, 1997). Assim, da mesma forma que em COLARES et al. (2009), os requisitos que apresentarem maior risco devem ser implementados mais cedo, assegurando um gerenciamento de riscos de forma adequada.

As restrições do problema são expressas em 4, 5 e 6. Desta forma, **(4)** e **(5)** são as restrições que limitam o custo e o tempo de implementação de todos os requisitos ao orçamento disponível e à duração do projeto, respectivamente; e **(6)** é a restrição que expressa relacionamentos de precedência técnica entre requisitos. Se um requisito r_i precede tecnicamente um requisito r_j, então r_i deve ser implementado antes de r_j $\left(x^{Prio}{}_i < x^{Prio}{}_j\right)$.

5. Otimização Multiobjetivo

Uma vez que o problema tratado neste trabalho é modelado como um problema de otimização multiobjetivo, esta seção apresenta alguns conceitos relacionados à otimização multiobjetivo e descreve os algoritmos, NSGA-II e MOCell, usados nos experimentos.

5.1. Frente de Pareto

Em otimização multiobjetivo, o conceito de "melhor, pior ou igual", usado na otimização mono-objetivo para comparar soluções, é substituído pelo conceito de dominância.

Assim, uma solução S_1 domina uma solução S_2 se:

- A solução S_1 é ao menos tão boa quanto à solução S_2 em todos os objetivos (ou, a solução S_1 não é pior que a solução S_2 em todos os objetivos) e;
- A solução S_1 é estritamente melhor que a solução S_2 em ao menos um objetivo.

Soluções que não são dominadas por nenhuma outra solução são chamadas de soluções não-dominadas ou ótimas, ou seja, representam soluções que são igualmente boas, onde nenhuma é melhor que a outra. Assim, o processo de busca não se restringe a encontrar e retornar uma única solução, mas sim o conjunto das soluções ótimas ou não-dominadas.

Cada solução no conjunto de soluções não-dominadas é dita ser um ótimo de Pareto. A representação coletiva de todas essas soluções é denominada frente de Pareto (ou *Pareto front*).

Os principais desafios em otimização multiobjetivo consistem em encontrar a frente de Pareto ideal (ou ótima) e manter a distribuição das soluções a mais diversa possível.

5.2. Metaheurísticas

Nem sempre é possível encontrar a melhor solução de um problema de otimização, em tempo razoável, através de algoritmos exatos. Métodos heurísticos são algoritmos desenvolvidos exclusivamente para resolver certo tipo de problema a um custo computacional razoável, obtendo-se soluções de boa qualidade em um tempo adequado às necessidades da aplicação. As heurísticas de propósito geral, desenvolvidas para resolução de problemas difíceis, são denominadas Metaheurísticas (VIANA, 1998).

Algoritmos evolucionários são metaheurísticas que utilizam seleção natural como seu mecanismo de busca na resolução de problemas (COELLO, 2005) e são caracterizados por uma população de soluções candidatas e um processo iterativo de reprodução (que realiza a combinação de soluções existentes para gerar novas soluções) e avaliação; assim, uma seleção natural determina quais indivíduos da população atual irão participar da nova população. Devido a sua abordagem baseada em população, esse processo permite encontrar várias soluções ótimas em apenas uma única execução. Assim, algoritmos evolucionários multiobjetivos (MOEA – *Multiobjective Evolutionary Algorithms*) caracterizam-se como abordagens ideais para a resolução de problemas de otimização multiobjetivo (DEB, 2009). São apresentadas a seguir duas importantes e bastante utilizadas metaheurísticas, baseadas em algoritmos genéticos evolucionários, para resolução de problemas de otimização multiobjetivo, as quais foram usadas na resolução da abordagem aqui proposta.

5.2.1. NSGA-II – *Non-dominated Sorting Genetic Algorithm II*

NSGA-II (DEB et al., 2000) é um algoritmo que implementa os conceitos de dominância e de elitismo, sendo executado em duas fases, utilizando os mecanismos: *Non-dominated Sorting* – que realiza a busca por soluções próximas à frente de Pareto e *Crowding Distance Sorting* – que realiza a busca por soluções bem distribuídas no espaço. O seu funcionamento começa com uma população inicial, aleatoriamente criada, de tamanho N. A seguir, uma segunda população, também de tamanho N, é gerada com base na primeira a partir de operações genéticas como cruzamento e mutação. Essas duas populações são então combinadas formando uma nova população de tamanho $2N$. A seguir, é atribuído a cada indivíduo um grau de não-dominância em relação a todos os demais indivíduos dessa população. Logo após, ocorre a classificação dos indivíduos em *fronts*, conforme seus valores de não-dominância. Assim, no primeiro *front* (F_1), deverão ficar os indivíduos que não são dominados por nenhuma outra solução. No próximo *front* (F_2), deverão ficar os elementos que não são dominados por nenhuma outra solução, excetuando-se os elementos de F_1. Esse processo segue até que todos os indivíduos estejam classificados em algum *front*. Após esse processo, uma nova população, de tamanho N, deve ser gerada e composta por indivíduos dos diferentes *fronts*. A geração começa com a inclusão dos indivíduos do primeiro *front*, seguida pela inclusão dos indivíduos do segundo *front* e assim sucessivamente. Como a quantidade de indivíduos de todos os *fronts* é $2N$ e o tamanho da nova população é N, não será possível incluir todos os *fronts* na nova população (os quais serão descartados). Ainda, é possível que na inclusão dos indivíduos do último *front* permitido, não haja espaço suficiente para a inclusão de todos os seus elementos e assim, alguns indivíduos desse *front* também precisarão ser descartados. O procedimento para selecionar quais indivíduos desse último *front* deverão compor a nova população é denominado *Crowding Distance* e tem por objetivo classificar e escolher os elementos melhor espalhados dentro desse *front*, através de uma métrica de diversidade, que calcula a distância entre um indivíduo e seus vizinhos levando em consideração todos os objetivos (M). A complexidade do algoritmo é, no máximo, $O(M.N^2)$.

5.2.2. MOCell – *Multiobjective Cellular Genetic Algorithm*

MOCell (NEBRO, et al. 2009) é uma adaptação de um modelo celular de algoritmo genético (cGA – *cellular Genetic Algorithm*) canônico para otimização multiobjetivo.

O algoritmo utiliza um arquivo externo para armazenar as soluções não-dominadas encontradas durante a busca e um mecanismo de *feedback* no qual as soluções deste arquivo substituem, randomicamente, indivíduos existentes na população depois de cada iteração. Para gerenciar a inserção de soluções na frente de Pareto, para que se possa obter um conjunto diverso, um estimador de densidade (baseado no método *Crowding Distance*) é utilizado. Esse mecanismo também é utilizado para remover soluções do arquivo quando este se tornar cheio (o arquivo externo possui um tamanho máximo). O algoritmo inicia com a criação de uma frente de Pareto vazia. Os indivíduos são organizados em uma grade bidimensional e operadores genéticos são sucessivamente aplicados a eles até que uma condição de parada seja atingida. Portanto, para cada indivíduo, o algoritmo seleciona dois pais de sua vizinhança, faz uma recombinação entre

eles a fim de obter um descendente, realiza uma mutação e avalia o indivíduo resultante. Este indivíduo é inserido tanto na população auxiliar, se não for dominado pelo indivíduo atual, como na frente de Pareto. Depois de cada iteração, a população antiga é substituída pela auxiliar e um procedimento de *feedback* é acionado para substituir um número fixo de indivíduos da população, selecionados randomicamente, por soluções do arquivo.

6. Avaliação

Uma avaliação foi conduzida para demonstrar a viabilidade da abordagem proposta e a eficiência dos algoritmos utilizados na resolução do problema.

6.1. Experimentos

Esta subseção descreve o contexto utilizado na avaliação empírica.

6.1.1. Descrição das Instâncias

Nove instâncias de problemas foram geradas e utilizadas para analisar a abordagem proposta em diferentes contextos. A Tabela 2 apresenta os atributos de cada instância.

Tabela 2: Descrição das Instâncias

Instância	Número Requisitos	Número Clientes	Orçamento Disponível Projeto	Tempo Disponível Projeto	Precedência Técnica Requisitos	Dens. Prec.Técn. Requisitos
A1_I.50.3.10.80.80	50	3	80%	80%	10%	5%
A1_I.50.5.10.80.80	50	5	80%	80%	10%	5%
A1_I.50.8.10.80.80	50	8	80%	80%	10%	5%
A1_I.100.3.10.80.80	100	3	80%	80%	10%	5%
A1_I.100.5.10.80.80	100	5	80%	80%	10%	5%
A1_I.100.8.10.80.80	100	8	80%	80%	10%	5%
A1_I.200.3.10.80.80	200	3	80%	80%	10%	5%
A1_I.200.5.10.80.80	200	5	80%	80%	10%	5%
A1_I.200.8.10.80.80	200	8	80%	80%	10%	5%

Os valores para $risk_i$, $cost_i$, $time_i$, w_m, $value(m,i)$ e $priority(m,i)$ foram gerados aleatoriamente de acordo com as escalas definidas anteriormente. Para os atributos $budgetProject$ e $timeProject$ (colunas Orçamento Disponível Projeto e Tempo Disponível Projeto, respectivamente) foram considerados 80% do orçamento e do tempo totais necessários para implementar todos os requisitos. Matrizes de Precedência Técnica entre requisitos também foram geradas randomicamente obedecendo aos percentuais estabelecidos nas duas últimas colunas da tabela. Assim, a coluna Precedência Técnica Requisitos indica o percentual de requisitos que possuem Precedência Técnica com outros requisitos (ex.: para uma instância com 100 requisitos, 10 requisitos apresentarão precedência técnica). Da mesma forma, a coluna Dens. Prec. Técn. Requisitos indica o percentual de requisitos com os quais um requisito apresenta Precedência Técnica.

6.1.2. Configuração dos Parâmetros

Para a resolução do problema formulado anteriormente, as metaheurísticas NSGA-II e MOCell foram aplicadas a cada instância. Os parâmetros utilizados para cada metaheurística foram configurados a partir da execução de testes preliminares.

Assim, para o NSGA-II foram utilizados: tamanho da população igual a 250 indivíduos (a população inicial foi gerada aleatoriamente), número máximo de avaliações igual a 33000 resultando em 132 gerações, taxa de cruzamento igual a 0,9 utilizando o operador *TwoPointsCrossover*, taxa de mutação igual a $1/N$ utilizando o operador *SwapMutation* e seleção utilizando o método torneio binário. Para o MOCell foram utilizados: tamanho da população igual a 256 indivíduos (a população inicial também foi gerada aleatoriamente), tamanho do arquivo externo igual a 256, número máximo de avaliações igual a 32768 resultando em 128 gerações, mecanismo de *feedback* igual a 20, taxa de cruzamento igual a 0,9 utilizando o operador *TwoPointsCrossover*, taxa de mutação igual a $1/N$ utilizando o operador *SwapMutation*

e seleção utilizando o método torneio binário.

6.1.3. Algoritmo de Busca Randômica

Qualquer metaheurística deve ser capaz de superar um algoritmo de busca puramente randômico para que possa ser aplicada com sucesso em SBSE; ou seja, deve encontrar soluções melhores (HARMAN e JONES, 2001). Na busca randômica, como o próprio nome sugere, o problema é resolvido obtendo-se uma solução gerada de forma aleatória. Normalmente, um algoritmo assim é utilizado como um referencial.

Neste trabalho, um algoritmo de busca randômica também foi utilizado para permitir que os resultados obtidos pelas metaheurísticas pudessem ser comparados e validados.

6.1.4. Implementação

A implementação da abordagem proposta e sua resolução utilizando os algoritmos descritos foi realizada através do *framework* *j*Metal (DURILLO et al., 2006), o qual disponibiliza diversas metaheurísticas para otimização multiobjetivo, incluindo NSGA-II e MOCell.

Os experimentos foram executados em um computador com a seguinte configuração de: Hardware – Processador Intel Core 2 Duo T6400, 2 GHz e memória RAM de 4GB; e Software – Sistema Operacional *Windows Vista Home Premium Service Pack 1* (32 bits).

6.2. Resultados e Análise

Para este trabalho, foram calculados a média e o desvio-padrão de dez execuções de cada algoritmo em cada instância para definir o comportamento do tempo de execução (em milissegundos). Os resultados obtidos em cada instância são apresentados na tabela abaixo.

Tabela 3: Tempo de Execução (em milisegundos)

Instância	NSGA-II	MOCell	Randômico
A1_I.50.3.10.80.80	8530.5 ± 137.6099	2084.6 ± 57.0247	417.8 ± 10.1521
A1_I.50.5.10.80.80	8593.65 ± 146.6604	2178.5 ± 122.2402	448.05 ± 31.9382
A1_I.50.8.10.80.80	8692.2 ± 196.4478	2236.2 ± 139.2391	494.1 ± 71.1682
A1_I.100.3.10.80.80	8818.975 ± 290.9820	2803.425 ± 1002.5009	573.325 ± 151.9342
A1_I.100.5.10.80.80	8904.1 ± 325.7497	3170.72 ± 1162.3342	649.34 ± 204.8559
A1_I.100.8.10.80.80	8989.15 ± 358.7906	3461.7833 ± 1246.2322	736.0833 ± 270.4139
A1_I.200.3.10.80.80	9079.5714 ± 411.1121	4755.9857 ± 3394.6476	867.2142 ± 408.9294
A1_I.200.5.10.80.80	9171.0625 ± 456.9806	5771.9625 ± 4170.8852	1000.3375 ± 521.2338
A1_I.200.8.10.80.80	9275.9555 ± 524.2612	6596.9222 ± 4576.9661	1151.8111 ± 653.2882

Com relação ao tempo de execução, o algoritmo randômico apresentou o melhor desempenho (menor média e menor desvio-padrão) em todas as instâncias, justificável pelo fato de não utilizar nenhum critério na geração das soluções. Quando se compara somente NSGA-II e MOCell, o algoritmo MOCell apresentou menor tempo de execução em todas as instâncias.

Os gráficos da Figura 1 apresentam os resultados obtidos na primeira execução dos três algoritmos para algumas instâncias selecionadas. Devido a limitações de espaço, apenas três instâncias (uma considerada pequena, outra média e outra grande, em termos de quantidade de requisitos) apresentam seus resultados graficamente. Os resultados comparam o desempenho dos algoritmos através de conceitos da frente de Pareto, onde uma frente gerada por cada metaheurística em cada instância selecionada foi utilizada na construção dos gráficos para representar as soluções não-dominadas.

Em todas as instâncias, a qualidade das soluções obtidas pelo algoritmo de busca randômica foi superada pela das metaheurísticas, visto que encontrou soluções bem distantes da frente de Pareto (distância menor nas instâncias pequenas, mas foi aumentando para as instâncias maiores). Para a menor das instâncias (com 50 requisitos), o desempenho das metaheurísticas pode ser considerado semelhante, com a definição de frentes similares, embora o algoritmo MOCell tenha uma melhor distribuição (ou espalhamento) das soluções. Na instância média (com 100 requisitos), o MOCell apresentou melhores resultados. Já para a maior instância (com 200 requisitos), ocorreu o inverso e o NSGA-II apresentou soluções mais próximas da frente. No

entanto, em ambas, as soluções não estão muito bem distribuídas no espaço.

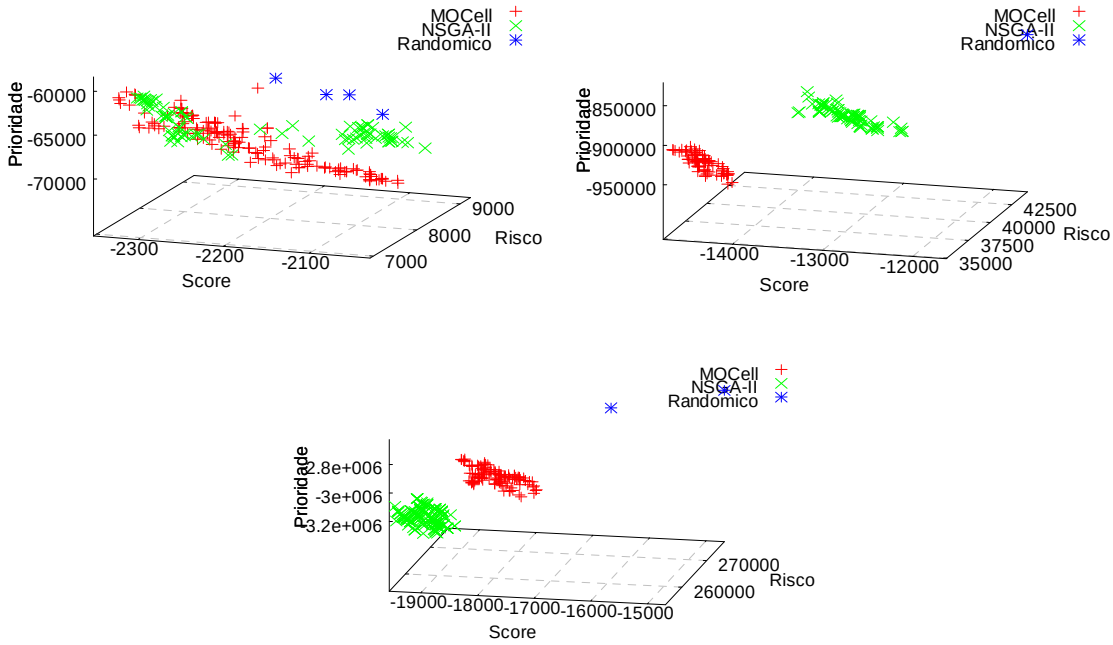

Figura 1: Resultados para as instâncias A1_I.50.3.10.80.80, A1_I.100.5.10.80.80 (superiores) e A1_I.200.8.10.80.80 (inferior).

7. Considerações Finais

Devido à quantidade de aspectos envolvidos, a seleção e a priorização de requisitos em um projeto de desenvolvimento de software apresentam uma complexidade tal que tornam o problema adequado para resolução por meio da aplicação de técnicas automatizadas na tentativa de se obter bons resultados. Neste trabalho, o problema é modelado como um problema de otimização. A abordagem proposta é consistente por considerar diversos aspectos que influenciam a tomada de decisões em projetos reais, visando à satisfação de clientes e o gerenciamento de riscos de forma adequada, bem como buscando respeitar os recursos disponíveis e o relacionamento entre requisitos. Metaheurísticas multiobjetivas bastante utilizadas foram aplicadas e se mostraram eficientes na resolução do problema.

Trabalhos futuros incluem realizar estudos com mais instâncias avaliando o comportamento das metaheurísticas de modo mais aprofundado; comparar o desempenho das metaheurísticas através de métricas com *Hypervolume* e *Spread* (DEB, 2009) e utilizar outras metaheurísticas multiobjetivas.

Referências

Aurum, A. e Wohlin, C. (2005), Requirements Engineering: Setting the Context, *Engineering and Managing Software Requirements*, Springer.

Bagnall, A. J., Rayward-Smith, V. J. e Whittley, I. M. (2001), The Next Release Problem, *Information and Software Technology*, 43(14), 883–890.

Berander, P. e Andrews, A. (2005), Requirements Prioritization, *Engineering and Managing Software Requirements*, 69-94.

Carlshamre, P., Sandahl , K., Lindvall, M., Regnell, B. e Dag, J. N. (2001), An Industrial Survey of Requirements Interdependencies in Software Product Release Planning, *Proceedings of the Fifth IEEE International Symposium on Requirements Engineering*, 84-91, Toronto, Canada, IEEE Computer Society.

Clarke, J., Dolado, J. J., Harman, M., Hierons, R., Jones, B., Lumkin, M., Mitchell, B., Mancoridis, S., Rees, K., Roper, M., Shepperd, M. (2003), Reformulating Software Engineering as a Search Problem, *IEE ProceedingsSoftware*, 161-175.

Coello, C.A.C. (2005), Recent Trends in Evolutionary Multiobjective Optimization, *Evolutionary Multiobjective Optimization Theoretical Advances and Applications*, 7-32, Springer.
Colares, F., Souza, J., Carmo R., Pádua, C. e Mateus, G. R. (2009), A New Approach to the Software Release Planning, *Proceedings of the XXIII Brazilian Symposium on Software Engineering - SBES '09*, 207-215.
Dahlstedt, A. G., Persson, A. (2005), Requirements Interdependencies: State of the Art and Future Challenges, *Engineering and Managing Software Requirements*, Springer, 95-116.
Deb, K., *Multi-Objective Optimization Using Evolutionary Algorithms*, Wiley, 2009.
Deb, K., Pratap, A., Agarwal, S. e Meyarivan, T. (2000), A fast and elitist multiobjective genetic algorithm: NSGA-II, *IEEE Transactions on Evolutionary Computation*, 6(2), 182–197.
del Sagrado, J., del Águilla, I. M. e Orellana, F. J. (2010), Ant Colony Optimization for the Next Release Problem: A Comparative Study, *Proceedings of the 2nd International Symposium on Search Based Software Engineering – SSBSE '10*, 67-76, IEEE Computer Society.
Durillo, J. J., Nebro, A. J., Luna, F., Dorronsoro, B. e Alba, E., jMetal: a Java Framework for Developing Multi-Objective Optimization Metaheuristics, *Technical Report: ITI 2006-10*, University of Málaga, 2006.
Glover, F. e Laguna, M., *Tabu Search*, Kluwer Academic Publishers, 1997.
Goldberg, D. E., *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, 1989.
Greer D. e Ruhe, G. (2004), Software Release Planning: An Evolutionary and Iterative *Approach, Information and Software Technology*, 46(4), 243-253.
Harman, M. e Jones, B. F. (2001), Search-Based Software Engineering, *Information & Software Technology*, 43(14), 833-839.
Jung, H.-W. (1998), Optimizing Value and Cost in Requirements Analysis. *IEEE Software*, 15(4), 74-78.
Karlsson, J. e Ryan, K. (1997), A Cost-Value Approach for Prioritizing Requirements, *IEEE Software*, 14(5), 67-74.
Laarhoven, P. J. M. van e Aarts, E. H. L., *Simulated Annealing: Theory and Applications*, Kluwer Academic Publishers, 1987.
Nebro, A. J., Durillo, J. J., Luna, F., Dorronsoro, B. e Alba, E. (2009), MOCell: A Cellular Genetic Algorithm for Multiobjective Optimization, *International Journal of Intelligent Systems*, 24, 726-746.
Ruhe, G. e Saliu, O. (2005), The Art and Science of Software Release Planning, *IEEE Software*, 22(6), 47-53.
Saaty, T. L., *The Analytic Hierarchy Process*, McGraw-Hill, 1980.
Sommerville, I. e Sawyer, P., *Requirements Engineering: A Good Practice Guide*, Wiley, 1997.
Sommerville, I., Engenharia de Software, Pearson Addison-Wesley, São Paulo, 2007.
Souza, J. T., Maia, C. L., Freitas, F. G. e Coutinho, D. P. (2010), The Human Competitiveness of Search Based Software Engineering, *Proceedings of the 2nd International Symposium on Search Based Software Engineering (SSBSE '10)*, 143-152, Benevento, Italy. IEEE.
Viana, G. V. R., *Meta-Heurísticas e Programação Paralela em Otimização Combinatória*, Edições UFC, 1998.
Zhang, Y., Alba, E., Durillo, J. J., Eldh, S. e Harman, M. (2010), Today/Future Importance Analysis, *Proceedings of the 12th annual Conference on Genetic and Evolutionary Computation (GECCO '10)*, 1357–1364, Portland, Oregon, USA. ACM.
Zhang, Y., Finkelstein , A. e Harman, M. (2008), Search Based Requirements Optimisation: Existing Work and Challenges, *Requirements Engineering: Foundation for Software Quality*, Lecture Notes in Computer Science, 5025, 88-94.
Zhang Y., Harman, M. e Mansouri, S. A. (2007), The Multi-Objective Next Release Problem, *Proceedings of the 9th annual Conference on Genetic and Evolutionary Computation (GECCO '07)*, 1129–1137, London, UK. ACM.