UMA ABORDAGEM BASEADA EM BUSCA PARA O PROBLEMA DA SELEÇÃO DE REQUISITOS DE SOFTWARE NA PRESENÇA DE SIMILARIDADE

Márcia Maria Albuquerque Brasil, Thiago Gomes Nepomuceno da Silva, Thiago do Nascimento Ferreira, Fabrício Gomes de Freitas, Jerffeson Teixeira de Souza

Grupo de Otimização em Engenharia de Software (GOES.UECE)
Universidade Estadual do Ceará (UECE)
Av. Paranjana, 1700 – 60.740-903 – Fortaleza, Ceará, Brasil

marcia.abrasil@gmail.com, thi.nepo@gmail.com, thiagonascimento.uece@gmail.com, fabriciogf.uece@gmail.com, jeff@larces.uece.br

RESUMO

Durante o planejamento de um projeto de desenvolvimento de software, decidir quais funcionalidades devem ser desenvolvidas e disponibilizadas na próxima versão (ou *release*) do sistema nem sempre é uma tarefa simples. Um planejamento adequado deve atender às expectativas dos clientes, agregar valor ao negócio da organização desenvolvedora e respeitar às restrições existentes. Outro aspecto que também deve ser levado em consideração são as interdependências entre os requisitos, ou seja, a forma como os requisitos se relacionam e interagem entre si. Nesse contexto, este trabalho tem por finalidade apresentar uma abordagem baseada em otimização para o problema da seleção de requisitos considerando a existência de similaridade entre os requisitos, uma forma de interdependência estrutural. A estratégia visa à satisfação de clientes e ao aumento do valor de negócio agregado evitando, contudo, que requisitos similares sejam selecionados. Experimentos realizados, através da aplicação de metaheurísticas, evidenciam a viabilidade da abordagem proposta.

PALAVARAS CHAVE. Seleção de requisitos de software, Engenharia de software baseada em busca, Metaheurísticas.

ABSTRACT

During the planning of a software development project, deciding which features should be developed and released in the next release of the system is not a simple task. Proper planning should meet the customers' expectations, adding value to the developer organization's business and comply with existing restrictions. Another aspect that must also be taken into consideration is the interdependency between requirements, i.e., how the requirements relate and interact. In this context, this paper aims to present an optimization-based approach to the problem of requirements selection taking into account the similarity between the requirements, a form of structural interdependence. The strategy aims both customer satisfaction and increased in business value, avoiding, however, that similar requirements are selected. Experiments performed by the application of metaheuristics demonstrate the viability of the proposed approach.

KEYWORDS. Software requirements selection, Search-based software engineering, Metaheuristics.

1. Introdução

O software é parte integrante de quase todas as operações de negócio (SOMMERVILLE, 2007), sendo considerado, portanto, um ativo estratégico para muitas organizações. Desta forma, é indispensável que funcionem de acordo com os objetivos para os quais foram desenvolvidos e atendam às necessidades e expectativas dos clientes.

Um importante desafio para as empresas desenvolvedoras de software consiste em, dado um conjunto de funcionalidades, determinar quais requisitos devem ser desenvolvidos e disponibilizados na próxima versão – ou *release* – do sistema. Essa dificuldade advém do fato de que nem sempre há recursos suficientes; da necessidade de agregar maior valor ao negócio do cliente e da organização; e da existência de possíveis interdependências entre os requisitos, pois os requisitos podem estar relacionados entre si de diferentes modos. Essas interdependências podem ocorrer de forma estrutural, restritiva e baseada em custo/valor (DAHLSTEDT e PERSSON, 2005). Assim, realizar um planejamento eficiente é uma importante e complexa atividade que envolve diversos aspectos.

Neste trabalho, a similaridade entre requisitos, uma espécie de interdependência estrutural entre funcionalidades, é tratada durante o processo de seleção de requisitos por meio de uma abordagem baseada em otimização. Esse contexto é então modelado matematicamente como um problema de busca onde, dado um conjunto de requisitos, deve-se selecionar um sub-conjunto "ótimo" de requisitos (de forma a se ter um maior valor de negócio, mas respeitando-se o orçamento disponível), evitando-se, contudo a seleção de requisitos similares. Com isso, seria evitado que requisitos similares fossem incluídos no *release* ou que os recursos fossem estimados em duplicidade para uma mesma funcionalidade, o que poderia impedir a inclusão de outros requisitos devido a limitações de recursos.

Assim como em outras áreas, técnicas de otimização vêm sendo aplicadas com sucesso a problemas da Engenharia de Software, os quais apresentam, muitas vezes, restrições, conflito de objetivos e grandes espaços de busca. Durante o processo de desenvolvimento de software, engenheiros de software normalmente se deparam com problemas que envolvem o atendimento a restrições inerentes ao próprio contexto do problema; e o conflito de interesses, em que a melhoria de um determinado aspecto muitas vezes implica em tornar outro pior, obrigando a se fazer uma escolha a partir da análise da situação como um todo. Portanto, encontrar soluções ótimas para esses problemas pode ser impossível ou inviável; por outro lado, encontrar soluções próximas de uma solução ótima ou que estejam dentro de uma faixa de tolerância aceitável pode ser suficiente em problemas da Engenharia de Software (HARMAN e JONES, 2001). Nessa área de pesquisa, conhecida como *Search-based Software Engineering* – SBSE (HARMAN e JONES, 2001), muitos problemas da Engenharia de Software, por serem de natureza complexa, são reformulados como um problema de busca e resolvidos através da aplicação de algoritmos de otimização, com destaque para as metaheurísticas. A reformulação de um problema da Engenharia de Software como um problema de busca envolve (HARMAN e JONES, 2001) (CLARKE et al., 2003): a escolha de uma representação do problema, que consiste na definição da estrutura a ser utilizada para representar as 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 produzidas pelo algoritmo. 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.

Assim, as principais contribuições desse trabalho compreendem:

- A apresentação de uma formulação matemática para o problema da seleção de requisitos de software considerando a presença de similaridade entre os requisitos;
- A resolução do problema modelado por meio da aplicação de técnicas de busca em instâncias do problema, bem como avaliação do desempenho das metaheurísticas utilizadas.

Além desta seção introdutória, o trabalho é organizado conforme a seguir. Seção 2, onde são apresentados alguns trabalhos relacionados à seleção e à priorização de requisitos de software; Seção 3, a qual define formalmente a abordagem proposta, descrevendo importantes aspectos levados em consideração e apresenta uma modelagem matemática para o problema; Seção 4, onde são apresentadas as metaheurísticas aplicadas neste trabalho; Seção 5, que mostra como o problema modelado pode ser resolvido por meio da aplicação de técnicas de busca mono-objetivas e relata experimentos realizados com a finalidade de demonstrar a aplicabilidade e a viabilidade da abordagem proposta; e, finalmente, Seção 6, onde são feitas as considerações finais do trabalho e apresentadas oportunidades para trabalhos futuros.

2. Trabalhos Relacionados

Uma abordagem baseada em custo/valor para priorização de requisitos foi desenvolvida por KARLSSON e RYAN (1997). No trabalho, o método *Analytic Hierarchy Process* – AHP é utilizado para comparar pares de requisitos e avaliar sua importância relativa baseando-se no valor de cada requisito e no custo de sua implementação. No entanto, o grande número de comparações necessárias torna a abordagem complexa à medida que a quantidade de requisitos aumenta. Uma variante do problema da mochila 0-1 foi apresentada por JUNG (1998) para reduzir a complexidade da abordagem custo/valor.

Na área de SBSE, o problema da seleção de requisitos é conhecido como "*The Next Release Problem*" (NRP) ou Problema do Próximo *Release* e foi inicialmente abordado por BAGNALL, RAYWARD-SMITH e WHITTLEY (2001). No trabalho, uma abordagem baseada em otimização é proposta para selecionar um conjunto ótimo de clientes, cujos requisitos deverão ser atendidos no próximo *release* do sistema. Com uma formulação mono-objetiva, os clientes mais importantes são então priorizados. A estratégia leva em consideração os recursos disponíveis e a precedência técnica entre os requisitos, ou seja, quando a implementação de um requisito pressupõe a implementação prévia ou conjunta de outro requisito. Diversas técnicas foram aplicadas para resolver o problema, dentre elas: Programação Linear Inteira, Algoritmos Gulosos, GRASP, *Simulated Annealing* e *Hill Climbing*.

O problema do próximo *release* é tratado de forma multiobjetiva em ZHANG, HARMAN e MANSOURI (2007). A satisfação de clientes e o custo total do projeto são os objetivos que devem ser otimizados. 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 para selecionar um conjunto ótimo de requisitos. Contudo, a formulação não inclui qualquer forma de interdependência entre requisitos, algo incomum no contexto de projetos reais, uma vez que um requisito pode se relacionar com 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. DURILLO et al. (2009) fazem 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.

O conjunto das técnicas utilizadas para resolver o problema do próximo *release* é ampliado e uma adaptação do algoritmo *Ant Colony System – ACS* é aplicada ao problema no trabalho de DEL SAGRADO, DEL ÁGUILA e ORELLANA (2010). 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. Neste trabalho, não foram consideradas interdependências entre os requisitos.

Ainda, um modelo de análise de importância para balancear as necessidades atuais e futuras dos requisitos quando do planejamento do próximo *release* do sistema é apresentado em ZHANG et al. (2010). 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.

No trabalho de ZHANG e HARMAN (2010), o gerenciamento da interação entre requisitos é abordado através de otimização baseada em busca, onde cinco formas de interdependências entre requisitos são consideradas. Dentre essas interdependências destacam-se a precedência técnica; o conflito entre requisitos, que ocorre quando dois requisitos são conflitantes entre si e não podem co-existir, devendo, portanto ser selecionado somente um dos dois; e o relacionamento custo/valor entre requisitos, que ocorre quando a implementação de um requisito afeta o custo ou o valor de outro requisito. Uma formulação multiobjetiva é utilizada para balancear os objetivos de valor e custo considerando as interdependências citadas na modelagem do problema e a metaheurística NSGA-II é um dos algoritmos aplicados.

Conforme pode ser observado, em nenhum dos trabalhos descritos anteriormente, a identificação e/ou o tratamento da similaridade entre requisitos são contemplados.

3. Problema da Seleção de Requisitos de Software com Similaridade

A presente seção traz uma descrição formal do problema e descreve aspectos importantes para a abordagem proposta, que foram considerados na formulação matemática do problema.

3.1. Apresentação Formal

Requisitos são definições de como um software deve se comportar e representam as funcionalidades que devem ser disponibilizadas por uma aplicação. Os requisitos consistem em uma especificação do que deve ser implementado e são definidos durante os estágios iniciais do desenvolvimento do sistema. 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).

Assim, seja $R = \{r_i | \, i = 1, 2, \dots, N\}$ o conjunto dos requisitos a serem implementados. O desenvolvimento de cada requisito r_i demanda certo custo denotado por $cost_i$. O custo de desenvolvimento normalmente é estimado pela organização desenvolvedora em termos de recursos necessários. Diversos fatores podem influenciar essa estimativa, dentre eles: a complexidade do requisito, a documentação e os testes necessários, a infra-estrutura, entre outros. Aqui, o custo de implementação de cada requisito é estimado em uma escala de 10 a 20. O desenvolvimento de um produto de software consiste de todos os requisitos a serem selecionados para o próximo *release* durante o planejamento do projeto. Assim, há um limite máximo nos recursos disponíveis, denotado por $resourceRelease$, os quais não devem ser excedidos.

Cada requisito r_i possui também um valor associado, representado por $value_i$, que pode ser definido em termos de quão importante esse requisito é para o cliente e para o negócio da organização desenvolvedora do software. Assim, o valor de cada requisito varia em uma escala de 1 (baixa importância) a 9 (alta importância). As escalas numéricas foram definidas apenas como uma forma de avaliar valores de importância e custo e possibilitar a modelagem matemática do problema.

Relacionamentos entre requisitos ou interdependências afetam o desenvolvimento de software de diversas formas (DAHLSTEDT e PERSSON, 2005). A interdependência considerada neste trabalho é a similaridade entre requisitos, uma forma de relacionamento estrutural entre funcionalidades. Essa forma de relacionamento descreve situações em que um requisito é semelhante a outro em termos de como ele é expresso ou em termos de uma idéia semelhante daquilo que o sistema deve ser capaz de realizar (DAHLSTEDT e PERSSON, 2005). Medidas de similaridade entre requisitos são discutidas em NATT OCH DAG e GERVASI (2005). A similaridade aqui tratada ocorre entre pares de requisitos e é recíproca. Assim, a similaridade entre dois requisitos r_i e r_j é dada por $similarity(r_i, r_j)$. Se dois requisitos são similares, então $similarity(r_i, r_j) = 1$; caso contrário $similarity(r_i, r_j) = 0$.

3.2. Abordagem Proposta

A finalidade da estratégia aqui proposta é selecionar para desenvolvimento os requisitos considerados mais importantes e evitar que requisitos considerados similares sejam incluídos na próxima versão do sistema. Ainda, devido às limitações de recursos, é possível que nem todos os requisitos sejam selecionados.

Assim, essa abordagem tem como objetivo:

- Maximizar o valor de negócio agregado, selecionando para desenvolvimento os requisitos considerados mais importantes e evitando, ao mesmo tempo, a seleção de requisitos similares;

E a seguinte restrição:

- Respeitar os recursos disponíveis para o *release*. Ou seja, os custos de implementação dos requisitos estão limitados aos recursos disponíveis para o *release*.

3.3. Formulação Matemática

A partir do que foi apresentado anteriormente, o problema é matematicamente formulado conforme a seguir.

$$Max\ f_{BUSINESS\_VALUE}(x) = \alpha.\left(100.\frac{\sum_{i=1}^{N}(value_i.x_i)}{\sum_{i=1}^{N}(value_i)}\right) - \beta.(100.similarityLevel)$$

Sujeito a:

$$\sum_{i=1}^{N} cost_i.x_i \leq resourceRelease$$

A estrutura utilizada para representar a solução do problema é um vetor de N posições (número total de requisitos), onde cada posição, representada pela variável x_i, corresponde a um requisito i e o seu conteúdo (0 ou 1) indica se o requisito i foi selecionado ou não.

Assim, a variável x_i indica se o requisito r_i foi selecionado para compor o *release* ($x_i = 1$) ou não ($x_i = 0$), para $i = 1, 2, \ldots N$. Os pesos, indicados por α e β, representam a importância que se deseja atribuir a cada função.

A função $similarityLevel$ representa a penalidade atribuída à solução pela seleção de requisitos similares, ou seja, o nível de similaridade existente em uma solução. É calculada conforme a seguir.

$$similarityLevel = \frac{penalty}{maxSimilarity}$$

Onde, $penalty$ contabiliza o número de pares de requisitos similares que foram selecionados na solução, sendo dado por:

$$penalty = \sum\left(similarity(r_i, r_j)\right) \mid \{\forall_{i,j}\ 1 \leq i \leq N\ e\ i < j\} \mid \{x_i = x_j = 1\}$$

E, $maxSimilarity$ representa o número máximo de pares de requisitos similares existentes em uma determinada situação ou instância do problema.

4. Algoritmos Baseados em Busca

Metaheurísticas são métodos aproximativos desenvolvidos para a resolução de problemas difíceis. Não garantem encontrar a solução ótima de um problema (a inexistência de uma solução melhor não é assegurada), mas são capazes de obter soluções de boa qualidade a um

custo computacional aceitável, através da redução do espaço de busca do problema. Dependendo do problema em questão, uma solução relativamente boa pode ser suficiente. (VIANA, 1998).

A seguir são apresentadas duas importantes metaheurísticas utilizadas para resolução de problemas de otimização mono-objetivo, as quais foram aplicadas na resolução do problema aqui modelado.

4.1. Algoritmos Genéticos (*Genetic Algorithm – GA*)

Baseados na teoria da evolução natural das espécies, os algoritmos genéticos (GOLDBERG, 1989) partem da idéia de que os "melhores" indivíduos devem sobreviver e gerar descendentes que preservem as suas características genéticas.

O funcionamento de um algoritmo genético começa com uma população inicial de indivíduos, que pode ser gerada de forma aleatória ou através de alguma heurística de construção. A partir daí, é feita uma avaliação de cada indivíduo e os melhores são selecionados. A seleção é feita de acordo com o desempenho em relação à função de satisfação. Em seguida, manipulações genéticas, como cruzamento e mutação, são aplicadas a esses indivíduos com o objetivo de se gerar uma nova população.

A operação de cruzamento permite uma recombinação do material genético dos indivíduos da população, a fim de promover uma diversificação. Através de um operador de *crossover*, dois pais são escolhidos aleatoriamente e têm parte de seu material genético trocado. Dois novos filhos são então gerados em substituição a seus pais. Já a operação de mutação realiza uma alteração na estrutura do indivíduo gerando um novo indivíduo. Para que o processo não gere necessariamente as mesmas soluções e para que vários espaços de busca possam ser explorados, a operação de mutação é aplicada com uma pequena taxa sobre as soluções. Assim, como novos indivíduos (soluções) são gerados a partir de soluções que foram selecionadas, o processo evolui com o intuito de gerar soluções cada vez melhores.

Todo esse processo se repete por um determinado número de gerações, caracterizando um critério de parada do algoritmo. O melhor indivíduo da última população gerada representa a solução do problema.

4.2. Têmpera Simulada (*Simulated Annealing – SA*)

A metaheurística Têmpera Simulada ou Arrefecimento Simulado (LAARHOVEN e AARTS, 1987) é baseada em um processo físico da Termodinâmica para têmpera de alguns materiais. Trata-se de um processo de resfriamento que tem por finalidade atribuir rigidez e consistência a certos materiais. Tais materiais são inicialmente submetidos a altas temperaturas. Essas temperaturas são então reduzidas gradualmente até que se alcance, por meio de elevações e reduções do estado de energia, o equilíbrio térmico (VIANA, 1998). O algoritmo de busca local Têmpera Simulada é uma analogia a esse processo de evolução do equilíbrio térmico dos materiais.

Inicialmente, tem-se uma solução qualquer, gerada aleatoriamente e a temperatura inicial assume um valor elevado. A cada iteração, uma nova solução, vizinha da solução atual é gerada. É feita então uma verificação da alteração do estado de energia (valor da função-objetivo). A solução atual é substituída de acordo com essa variação. Se houver uma redução de energia, o novo estado é aceito porque é melhor. Se não houver variação de energia (estabilidade), a aceitação do novo estado é indiferente. Se houver um aumento de energia, a aceitação do novo estado depende de uma probabilidade baseada no valor da temperatura. No início há uma maior probabilidade para aceitação de soluções piores. Essa probabilidade vai então diminuindo conforme a temperatura vai decrescendo. Todo esse processo é repetido iterativamente, diminuindo-se gradualmente a temperatura, até que se aproxime de zero.

5. Avaliação Empírica

Uma avaliação empírica foi conduzida com o objetivo de demonstrar a viabilidade da abordagem proposta e a eficiência dos algoritmos na resolução do problema.

5.1. Projeto dos Experimentos

Esta subseção descreve o contexto utilizado na avaliação empírica.

5.1.1. Descrição das Instâncias

Seis instâncias de problemas foram geradas e utilizadas para avaliar a abordagem em contextos distintos. A Tabela 1 apresenta as características de cada instância.

Tabela 1 – Descrição das Instâncias

Instância	*Número de Requisitos*	*Nível de Similaridade entre Requisitos*	*Orçamento Disponível* ($resourceRelease$)
InstA_050_05_80	50	5%	80%
InstB_050_20_80	50	20%	80%
InstC_200_05_80	200	5%	80%
InstD_200_20_80	200	20%	80%
InstE_500_05_80	500	5%	80%
InstF_500_20_80	500	20%	80%

Os valores para $cost_i$ e $value(i)$ foram gerados aleatoriamente de acordo com as escalas definidas anteriormente. Para o atributo $resourceRelease$ (coluna Orçamento Disponível) foi considerado 80% dos recursos necessários para implementar todos os requisitos.

Matrizes de Similaridade entre requisitos também foram geradas randomicamente obedecendo aos percentuais estabelecidos na coluna Nível de Similaridade entre Requisitos da tabela. Assim, essa coluna indica um percentual de similaridade entre pares de requisitos considerando todas as combinações possíveis. Exemplo: Para uma instância com 50 requisitos, o número máximo de pares de requisitos similares é 1225, que é dado por $\frac{N.(N-1)}{2}$, assumindo que todos os requisitos serão similares entre si. Considerando o percentual de 20% sobre esse valor, tem-se que a instância possui um nível de similaridade entre requisitos de 20%; neste caso, 245 pares de requisitos similares.

Os parâmetros α e β foram aqui utilizados como o mesmo valor para que as duas funções tivessem a mesma importância.

5.1.2. Configuração dos Parâmetros

Para a resolução do problema formulado anteriormente, as metaheurísticas Algoritmo Genético e Têmpera Simulada foram aplicadas a cada uma das instâncias apresentadas anteriormente. Os parâmetros utilizados para cada metaheurística foram configurados a partir da execução de testes preliminares nessas instâncias.

Assim, para o Algoritmo Genético foram utilizados: tamanho da população igual a 100 indivíduos (a população inicial foi gerada aleatoriamente), número máximo de avaliações igual a 1000000 resultando em 10000 gerações, taxa de cruzamento igual a 0,9 utilizando o operador *SinglePointCrossover*, taxa de mutação igual a 0,3 utilizando o operador *BitFlipMutation* e seleção utilizando o método torneio binário. Para a Têmpera Simulada foram utilizados: taxa de decréscimo da temperatura T igual a 0,9999 e critério de parada com temperatura T igual a 0,0001. A solução inicial foi gerada de forma aleatória.

5.1.3. Algoritmo de Busca Randômica

Em uma busca randômica, a resolução do problema ocorre através da geração aleatória e sem critério das soluções. Um algoritmo assim pode ser utilizado como um referencial. Em SBSE, para que possa ser aplicada com sucesso, uma metaheurística deve conseguir superar um algoritmo de busca puramente randômico; ou seja, deve produzir soluções melhores (HARMAN e JONES, 2001). Um algoritmo de busca randômica também foi implementado neste trabalho para permitir que os resultados obtidos pelas metaheurísticas pudessem ser comparados e validados.

5.1.4. Implementação

Os algoritmos descritos anteriormente, bem como o problema apresentado foram implementados na linguagem Java.

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

5.2. Apresentação dos Resultados

Para este trabalho, foram calculados a média e o desvio-padrão de dez execuções de cada algoritmo em cada instância para definir o comportamento da função-objetivo do problema e do tempo de execução (em milissegundos). Os resultados obtidos para cada instância são apresentados nas tabelas a seguir.

Tabela 2 – Valor da função $BUSINESS\_VALUE$

Instância	*Algoritmo Genético*	*Têmpera Simulada*	*Algoritmo Randômico*
InstA_050_05_80	68,5081 ± 0,9119	66,2118 ± 1,5265	52,6236 ± 1,5371
InstB_050_20_80	50,8393 ± 0,2986	49,4701 ± 0,8146	41,7165 ± 0,4991
InstC_200_05_80	41,8426 ± 0,7974	39,8752 ± 0,6190	33,8892 ± 0,5300
InstD_200_20_80	39,3072 ± 0,5486	37,9886 ± 0,5475	32,8406 ± 0,6392
InstE_500_05_80	34,5993 ± 0,2722	33,3455 ± 0,4314	29,8733 ± 0,2402
InstF_500_20_80	34,7394 ± 0,3907	33,1773 ± 0,2930	29,7783 ± 0,3405

Tabela 3 – Tempo de Execução (em milissegundos)

Instância	*Algoritmo Genético*	*Têmpera Simulada*	*Algoritmo Randômico*
InstA_050_05_80	17013,50 ± 32,94	3632,90 ± 100,05	321,30 ± 19,87
InstB_050_20_80	18291,50 ± 49,54	3883,50 ± 109,66	352,60 ± 10,72
InstC_200_05_80	122885,80 ± 719,16	32137,70 ± 1198,93	2606,60 ± 23,88
InstD_200_20_80	146614,90 ± 469,61	39093,20 ± 1643,15	3202,70 ± 14,64
InstE_500_05_80	671880,80 ± 1417,32	193494,00 ± 6792,67	14952,60 ± 50,37
InstF_500_20_80	802782,20 ± 4827,87	231507,10 ± 9085,36	18244,10 ± 230,39

5.3. Análise dos Resultados

Comparando-se o desempenho dos algoritmos através da Tabela 2, verifica-se que as duas metaheurísticas apresentaram um resultado bem superior ao do algoritmo randômico, o qual obteve os menores valores para a função-objetivo (menor média) em todas as instâncias.

Analisando o comportamento somente das metaheurísticas, o Algoritmo Genético apresentou melhor resultado em todas as instâncias do problema, com soluções de melhor qualidade, obtendo maior média em sua função-objetivo. O algoritmo também apresentou um menor desvio-padrão em metade das instâncias. No entanto, é importante destacar que os valores alcançados pela Têmpera Simulada ficaram próximos aos obtidos pelo GA.

Com relação ao tempo de execução (Tabela 3), o algoritmo randômico apresentou o melhor desempenho, com menor média e menor desvio-padrão em todas as instâncias. Tal fato é justificado por não ser aplicado nenhum critério para a geração das soluções. Quando se compara o tempo de execução somente das metaheurísticas, verifica-se que a Têmpera Simulada apresentou melhor desempenho (apesar de um desvio-padrão maior) em todas as instâncias, sendo bem mais rápida (aproximadamente 2,5 vezes) que o Algoritmo Genético.

Considerando o contexto de projetos reais, em que a decisão sobre quais requisitos implementar levando em conta diversos aspectos representa um desafio para Gerentes de Projetos, a modelagem aqui apresentada e as metaheurísticas aplicadas se apresentam como ferramentas úteis durante o processo decisório.

6. Considerações Finais

Este trabalho teve como objetivo abordar o problema da seleção de requisitos de software para o próximo *release* do sistema considerando a existência de similaridade entre as funcionalidades a serem desenvolvidas. Devido à quantidade e à natureza dos aspectos envolvidos, a seleção de requisitos em um projeto de desenvolvimento de software apresenta uma complexidade inerente que torna o problema adequado para resolução por meio da aplicação de técnicas automatizadas na tentativa de se alcançar bons resultados.

Aqui, mais um aspecto foi adicionado ao problema: a presença de similaridade entre requisitos. Esse aspecto torna o problema mais difícil, pois, ao mesmo tempo em que se deve selecionar os requisitos que agreguem maior valor, deve-se evitar a seleção de requisitos que apresentem uma relação de similaridade entre si, para que os recursos não sejam estimados em duplicidade e para que outros requisitos possam ser acrescentados. Desta forma, o problema foi modelado como um problema de otimização, onde uma formulação mono-objetiva foi apresentada e foi resolvido através de algoritmos baseados em busca, tais como Algoritmos Genéticos e Têmpera Simulada. As metaheurísticas aplicadas se mostraram eficientes na resolução do problema.

Trabalhos futuros incluem: o tratamento do grau de similaridade entre os requisitos (intensidade com que os requisitos estão relacionados entre si); a abordagem do problema de forma multi-objetiva, considerando objetivos relacionados a valor, custo e similaridade; considerar a perspectiva de *stakeholders* (envolvidos com o desenvolvimento do sistema) na modelagem do problema; a realização de experimentos com variações no peso de cada função; a aplicação de outras metaheurísticas na resolução do problema.

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.

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.

Dahlstedt, A. G., Persson, A. (2005), Requirements Interdependencies: State of the Art and Future Challenges, *Engineering and Managing Software Requirements*, Springer, 95-116.

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.

Goldberg, D. E., *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, 1989.

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.

Natt och Dag, Johan e Gervasi, Vincenzo. (2005), Managing Large Repositories of Natural Language Requirements, *Engineering and Managing Software Requirements*, Springer, 219-244.

Sommerville, I., Engenharia de Software, Pearson Addison-Wesley, São Paulo, 2007.

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., 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.

Zhang Y. e Harman, M. (2010), Search Based Optimization of Requirements Interaction Management, *Proceedings of the 2nd International Symposium on Search Based Software Engineering (SSBSE '10)*, 47-56, Washington, DC, USA. IEEE Computer Society.