O PROBLEMA DO PRÓXIMO LANÇAMENTO NA ENGENHARIA DE SOFTWARE[1]

Maria Cecilia Holler [2], Marcelo de Souza [3]

[1] Vinculado ao projeto intitulado “Projeto automático de algoritmos”.
[2]Acadêmico(a) do Curso de Eng. de Software – CEAVI – Bolsista PIVIC.
[3] Orientador(a), Departamento de Eng. de Software – CEAVI – marcelo.desouza@udesc.br

Empresas dedicadas ao desenvolvimento de software enfrentam desafios ao tentar equilibrar custos e selecionar quais requisitos devem ser incluídos na próxima versão do produto. A avaliação criteriosa desses requisitos é fundamental, pois eles provêm de diferentes clientes e possuem prioridades variadas, conforme observado por Morais e Zanin (2020). Decisões equivocadas podem resultar em problemas, como erros em estimativas de esforço ou o não cumprimento de prazos e orçamentos relacionados à nova versão (Harman, Mansouri e Zhang, 2009). Tudo isso gera insatisfação e perda de clientes. Os principais desafios desse cenário são estudados e tratados por uma área de pesquisa chamada *Engenharia de Software Baseada em Busca* (SBSE, do inglês *Search-Based Software Engineering*), que procura resolver problemas de otimização encontrados na engenharia de software, aplicando técnicas de busca (Freitas, Maia, Campos e Souza, 2010).

Na literatura, o problema de decidir quais requisitos serão implementados para a próxima entrega é frequentemente referido como o Problema do Próximo Lançamento, ou *Next Release Problem* (NRP). Trata-se de um problema NP-difícil (Bagnall, Rayward-Smith e Whittley, 2001) e amplamente estudado ao longo das últimas duas décadas (Veerapen et al., 2015). O problema considera um conjunto de n requisitos com custos de desenvolvimento c_i, $i = \{1, 2, \ldots, n\}$, e um conjunto de m clientes, cada qual com sua importância w_j, $j = \{1, 2, \ldots, m\}$. O conjunto F de pares (i, j) define que o requisito j depende do requisito i, ou seja, o requisito i precisa estar implementado para a implementação do requisito j. O conjunto Q de pares (i, k) define que o requisito i foi solicitado pelo cliente k. Deseja-se definir quais clientes serão atendidos na próxima versão do software. Um cliente é atendido quando todos os requisitos por ele solicitados são implementados.

Abaixo é apresentada a formulação matemática do problema. No modelo linear inteiro resultante, a variável de decisão binária $x_i = \{0, 1\}$ define se o requisito i será implementado ou não, enquanto a variável de decisão binária $y_j = \{0, 1\}$ define se o cliente j será atendido ou não. A função objetivo calcula a soma das importâncias dos clientes atendidos, a qual deve ser maximizada. A primeira restrição determina que o custo total dos requisitos a serem implementados não ultrapasse o orçamento b. A segunda restrição assegura que para implementar um requisito j, todos os requisitos i dos quais ele depende sejam também implementados. Finalmente, a terceira restrição define que se um cliente j é atendido, então todos os requisitos i por ele solicitados devem ser implementados.

$$
\begin{aligned}
\text{maximize} \quad & \sum_{i=1}^{m} w_i y_i \\
\text{subject to} \quad & \sum_{i=1}^{n} c_i x_i \leq b \\
& x_i \geq x_j, \forall (i,j) \in P \\
& x_i \geq y_j, \forall (i,j) \in Q \\
& x \in \{0,1\}^n, y \in \{0,1\}^m
\end{aligned}
$$

Neste trabalho, quando um requisito *j* depende de outro requisito *i*, o requisito *i* é adicionado à lista de requisitos de todos os clientes que solicitaram o requisito *j*. Ou seja, para atender um cliente, devem ser implementados todos os requisitos solicitados por ele, bem como aqueles dos quais esses requisitos dependem. Com esse pré-processamento, a segunda restrição pode ser removida do modelo, diminuindo sua complexidade.

O modelo matemático proposto foi implementado em Python usando a biblioteca pyomo, e resolvido usando o *solver* GLPK (GNU Linear Programming Kit). Foram adotadas as instâncias propostas por Xuan et al. (2012), que incluem instâncias criadas artificialmente e instâncias reais extraídas de projetos de software de código aberto. O orçamento é determinado como um percentual do custo total dos requisitos. São adotados os percentuais *{0,3, 0,5, 0,7}*, também chamados coeficientes.

Tabela 1. Resultados obtidos na solução do modelo matemático.

Instância	Clientes	Coeficiente	Resultado
nrp1	100	0,3	1.204
nrp1	100	0,5	1.840
nrp1	100	0,7	2.507
nrp2	500	0,3	4.970
nrp2	500	0,5	8.065
nrp2	500	0,7	11316
nrp3	500	0,3	7.488
nrp3	500	0,5	11.159
nrp3	500	0,7	14.196
nrp4	750	0,3	10.690
nrp4	750	0,5	15.985
nrp4	750	0,7	20.913
nrp5	1000	0,3	18.510
nrp5	1000	0,5	24.701
nrp5	1000	0,7	28.912
nrp-e1	536	0,3	7.919
nrp-e1	536	0,5	11.071
nrp-e2	491	0,3	7.446
nrp-e2	491	0,5	10.381
nrp-e3	456	0,3	6.666
nrp-e3	456	0,5	9.362
nrp-e4	399	0,3	5.814

nrp-e4	399	0,5	8.174
nrp-g1	445	0,3	6.130
nrp-g1	445	0,5	8.897
nrp-g2	315	0,3	4.580
nrp-g2	315	0,5	6.553
nrp-g3	423	0,3	5.932
nrp-g3	423	0,5	8501

A Tabela 1 apresenta cada instância, o número de clientes (que define o tamanho da instância e, por consequência, sua complexidade), o coeficiente usado para determinar o orçamento e o resultado obtido pelo *solver*, i.e. o valor da solução ótima. Os resultados demonstram a capacidade do modelo de programação linear inteira (PLI) em identificar as soluções ótimas para o problema do próximo lançamento, considerando restrições de custo e dependências entre requisitos.

A abordagem de PLI aplicada neste contexto demonstra como técnicas de otimização podem ser integradas na prática de engenharia de software, permitindo decisões bem informadas e mais acertadas na seleção de funcionalidades para novos lançamentos de produtos de software. A abordagem estudada não só maximiza a importância dos clientes atendidos e, por consequência, a satisfação desses clientes, mas também garante a viabilidade financeira e técnica das decisões tomadas pelas equipes de desenvolvimento.

Palavras-chave: Problema da próxima entrega. Otimização combinatória. Programação Linear Inteira.

MORAIS, Izabelly S.; ZANIN, Aline. Engenharia de software. Grupo A, 2020. E-book. ISBN 9788595022539. Disponível em: https://app.minhabiblioteca.com.br/#/books/9788595022539/. Acesso em: 10 nov. 2023.

HARMAN, Mark; MANSOURI, S. Afshin; ZHANG, Yuanyuan. Search based software engineering: A comprehensive analysis and review of trends techniques and applications. Department of Computer Science, Kings College London, Tech. Rep. 2009.

FREITAS, Fabrício Gomes de; MAIA, Camila Loiola Brito; CAMPOS, Gustavo Augusto Lima de; SOUZA, Jerffeson Teixeira de. Otimização em teste de software com aplicação de metaheurísticas. Revista de Sistemas de Informação da FSMA, n. 5, 2010.

BAGNALL, Anthony J.; RAYWARD-SMITH, Victor J.; WHITTLEY, Ian M. The next release problem. Information and Software Technology, 2001.

VEERAPEN, Nadarajen et al. An integer linear programming approach to the single and bi-objective next release problem. Information and Software Technology, v. 65, p. 1-13, 2015.

XUAN, Jifeng et al. Solving the large scale next release problem with a backbone-based multilevel algorithm. IEEE Transactions on Software Engineering, v. 38, n. 5, p. 1195-1212, 2012.