Workshop de Engenharia de Software Baseada em Busca

X WORKSHOP DE ENGENHARIA DE SOFTWARE BASEADA EM BUSCAS

EVENTO ONLINE

12 de JULHO de 2019

Editores:

Márcio de Oliveira Barros

Matheus Henrique Esteves Paixão

Mensagem dos Chairs da edição de 2019

É com satisfação que convidamos os interessados a participar do WESB'19, o X Workshop de Engenharia de Software Baseada em Busca.

A Engenharia de Software Baseada em Busca (SBSE – Search Based Software Engineering) é uma área de pesquisa que visa resolver os problemas da Engenharia de Software utilizando conceitos, técnicas, algoritmos e estratégias de busca. O objetivo da busca é identificar, dentre todas as soluções possíveis de um problema, uma solução que seja suficientemente boa de acordo com uma métrica apropriada. A SBSE permite que problemas previamente resolvidos de forma intensivamente manual e intuitiva possam ser resolvidos, total ou parcialmente, de forma sistemática e automatizada.

O Workshop de Engenharia de Software (WESB) tem o objetivo de ser um fórum nacional para discussão e divulgação de pesquisas em temas relacionados à área de SBSE, assim como um evento regular, que possa desenvolver e ampliar a emergente comunidade nacional, promovendo o crescimento e a consolidação do interesse na área por parte dos pesquisadores e profissionais que atuam tanto em Engenharia de Software quanto em outras áreas tais como otimização, meta-heurísticas, etc.

A edição de 2019 será a primeira a utilizar o formato de encontro online. Tal estratégia tem como objetivo facilitar a participação da comunidade como um todo, promovendo um rico fórum para apresentação e discussão de propostas de pesquisas, pesquisas em andamento e pesquisas finalizadas no âmbito da comunidade nacional de SBSE.

A programação contará com palestras de pesquisadores brasileiros na área de SBSE. Sete trabalhos serão apresentados, incluindo pesquisas finalizadas e em andamento no âmbito da comunidade nacional de Engenharia de Software Baseada em Busca. Os temas incluem arquitetura de software, programação genética, geração automática de código a partir de exemplos, modelagem de tópicos, alocação de recursos e novos algoritmos de busca.

Na posição privilegiada de primeiros *chairs* da versão online, desejamos a todos um excelente evento, cheio de discussões interessantes e em que todos possamos aprender um pouco mais sobre este cativante campo de pesquisa.

Organização

Program Chairs

Márcio de Oliveira Barros
Universidade Federal do Estado do Rio de Janeiro (UNIRIO)

Matheus Henrique Esteves Paixão
Universidade Estadual do Ceará (UECE)

Comitê de Programa

Altino Dantas Basílio Neto
Universidade Federal de Goiás (UFG)

Jerffeson Teixeira de Souza
Universidade Estadual do Ceará (UECE)

Thelma Elita Colanzi Lopes
Universidade Estadual de Maringá (UEM)

Thiago Nascimento Ferreira
Universidade Federal do Paraná (UFPR)

Wesley Kleverton Guez Assunção
Universidade Federal Tecnológica do Paraná (UTFPR)

Artigos selecionados

Programação Genética Aplicada a Estimativas de Projeto de Software

Arthur Lopes
Escola de Informática Aplicada - EIA
Univ. Federal do Estado do Rio de Janeiro - UNIRIO
Rio de Janeiro, Brasil
arthur.lopes@uniriotec.br

Márcio de Oliveira Barros
PPGI - Programa de Pós-Graduação em Informática
Univ. Federal do Estado do Rio de Janeiro - UNIRIO
Rio de Janeiro, Brasil
marcio.barros@uniriotec.br

***Resumo*—Um algoritmo de Programação Genética foi desenvolvido para gerar expressões matemáticas a partir de exemplos de entrada e saída no contexto da estimativa em projetos de software. Os resultados do algoritmo foram comparados com um algoritmo também baseado em Programação Genética, anteriormente apresentado na literatura.**

***Index Terms*—programação genética, estimativas em projetos de software**

I. Introdução

Projetos de desenvolvimento de software são conhecidos pela dificuldade de estimar seus custos e cronogramas. Apesar do cenário vir melhorando ao longo do tempo, os relatórios bienais do The Standish Group [2] mostram que ainda hoje a maioria dos projetos têm seus cronogramas e seus custos desafiados. A complexidade do ambiente sociotécnico em que projetos de software são desenvolvidos, aliado às constantes evoluções da tecnologia, contribuem para a dificuldade de se desenvolver bons modelos de estimativas. No entanto, este é um objetivo que deve ser constantemente perseguido em uma indústria do tamanho e com a capilaridade da indústria de software.

Neste artigo, uma técnica de Programação Genética foi desenvolvida para resolver um problema de otimização onde deseja-se encontrar a expressão matemática que melhor relacione um conjunto de pares de entradas x e y. Este problema é análogo ao desenvolvimento de modelos de estimativas em projetos de software quando temos, de um lado, o tamanho do software ou qualquer medida que possa ser estabelecida antes do seu desenvolvimento e, do outro lado, o esforço necessário para o desenvolvimento. Dolado [1] coletou doze *datasets* com este formato da literatura de Engenharia de Software para comparar um algoritmo baseado em Programação Genética com técnicas de regressão. Os resultados produzidos pela técnica aqui proposta foram comparados com os resultados obtidos pelo algoritmo de Programação Genética de Dolado [1].

As expressões matemáticas geradas pela técnica proposta superaram aquelas geradas pelos modelos de regressão linear e pelo modelo de Dolado na maioria dos casos, considerando uma comparação baseada em uma métrica de aptidão comum aos modelos.

II. A técnica de estimativa proposta

A população inicial do algoritmo de Programação Genética proposto é gerada através do método *Ramped Half-and-Half*, que usa o método *Grow* para gerar a metade da população e o método *Full* para gerar a outra metade. Nenhuma característica específica do problema de geração de expressões de estimativa é injetada na população inicial. Sua formação depende apenas da profundidade máxima usada no sorteio aleatório da população inicial (parâmetro da técnica) e dos conjuntos de funções e de terminais.

O conjunto das funções é composto pelos operadores matemáticos de divisão, multiplicação, subtração e soma e o conjunto dos terminais é composto pelo conjunto dos números reais e pela variável x, que representa o tamanho do software em questão. Na geração de árvores *Grow* e *Full*, o sorteio de terminais se limitou ao subconjunto dos números inteiros. Números reais aparecem como resultado da simplificação de árvores, método que age de forma recursiva para reduzir o tamanho das árvores, solucionando sub-expressões que não dependem da variável x. A simplificação ocorre após gerar cada árvore *Grow* e *Full* e após a ação dos operadores cruzamento e mutação.

A estrutura de representação dos indivíduos que compõem a população manipulada pela Programação Genética foi a de árvores sintáticas binárias, cujos nós são preenchidos com elementos de um conjunto de funções e de terminais. A métrica de aptidão adotada foi a magnitude média do erro relativo (MMRE), calculada comparando o resultado gerado por cada árvore resultante da técnica proposta com o valor y do *dataset* relacionado. A população possui tamanho fixo e evolui ao longo de diversas gerações graças à ação dos operadores genéticos de seleção, cruzamento, mutação e reprodução. O algoritmo é encerrado quando se atinge um certo número de gerações e retorna a árvore com o MMRE mais próximo de zero.

O operador de seleção escolhido foi a seleção por torneio probabilístico. Uma amostra de tamanho k é retirada da população e ordenada de acordo com a aptidão. O i-ésimo indivíduo é eleito para participar do torneio com probabilidade: $p \times (1-p)^i$. O operador de reprodução é acionado antes da seleção e copia os 30% melhores indivíduos,

isto é, indivíduos cujo MMRE mais se aproxima de zero, da população atual para a próxima geração (elitismo). Eles permanecem elegíveis para o torneio, que escolhe os participantes que sofrerão a ação dos operadores cruzamento e mutação. Após a reprodução, a seleção escolhe um destes operadores, baseado na sua probabilidade de ocorrência, para gerar um novo indivíduo, até completar os 70% restantes da população.

O operador de cruzamento escolhido foi o cruzamento de subárvore. A escolha do ponto de cruzamento se restringe aos nós internos das árvores, que contêm os operadores matemáticos. Um nó interno aleatório é sorteado em cada uma das duas árvores-pai vencedoras do torneio e as subárvores com raízes nestes nós são unidas em um novo descendente, que fará parte da próxima geração. O operador de mutação escolhido foi a mutação de subárvore. Um ponto de mutação aleatório é escolhido no vencedor do torneio e substituído por uma árvore gerada aleatoriamente. Optou-se pelo uso do método *Grow* para gerar a árvore que substituirá o ramo que sofreu mutação.

Os parâmetros selecionados para os primeiros estudos experimentais da técnica proposta acima são:

- Tamanho da população: 70 indivíduos.
- Número de gerações: 50 gerações
- Profundidade limite das árvores (*Grow* e *Full*): 4
- Tamanho do torneio (k): 5
- Probabilidade do torneio (p): 0.7
- Taxa de elitismo na reprodução: 30% da população
- Probabilidade de cruzamento: 75%
- Probabilidade de mutação: 25%

Algumas das escolhas de parâmetros foram baseadas na literatura, como o tamanho k e a probabilidade p do torneio. A profundidade limite das árvores iniciais tem um valor propositalmente intermediário. Com a redução dessas árvores, através da simplificação das expressões, há um potencial de geração de árvores muito simples, que dificilmente apareceriam em gerações futuras, dada a natureza expansiva dos operadores de mutação e cruzamento.

Os indivíduos selecionados pelo elitismo permitem a preservação de árvores com alta aptidão, garantindo que bons genes não se percam por consequência do sorteio aleatório do torneio. Os 25% de probabilidade da mutação representam uma alta taxa de ocorrência do operador, trazendo propositalmente um maior potencial de geração de variedade e resgate de diversidade perdida entre as gerações do processo evolutivo.

III. Avaliação

Um total de doze *datasets* com medidas de custo de software foram submetidos à técnica proposta. Os resultados foram comparados e, em onze casos, o valor de MMRE calculado pelo algoritmo aqui proposto superou aquele encontrado pelo algoritmo de Dolado [1]. No artigo original, a métrica de aptidão leva em consideração tanto o MMRE quanto a capacidade de predição do modelo. O nível de predição adotado foi 0.25, avaliando se o valor de y encontrado pelo algoritmo está dentro do intervalo $[y \times 0.75,\ y \times 1.25]$, considerando o y registrado no *dataset*.

Tabela I
Comparação dos resultados do algoritmo proposto com base nos *datasets* de Dolado

	Algoritmo de Dolado		Algoritmo proposto	
Dataset	MMRE	PRED(0.25)	MMRE	PRED(0.25)
1	0.2687	**76,19**	**0,2033**	66,67
2	0,6159	**56,52**	**0,4729**	26,0
3	0,2838	**72,22**	**0,2633**	61,11
4	0,8451	28,57	**0,4081**	28,57
5	1,8097	14,29	**0,6006**	**17,46**
6	**0,0896**	94,29	0,0901	94,29
7	0,4321	**45,83**	**0,3618**	35,42
8	1,0000	0,00	**0,3967**	40,00
9	0,5166	**46,81**	**0,3581**	48,94
10	3,8942	0,00	**0,4395**	**50,00**
11	0,6328	50,82	**0,4995**	39,34
12	1,1781	**30,30**	**0,6326**	18,18

Além do MMRE, a métrica de predição PRED(*0.25*) também foi calculada para os resultados gerados pelo algoritmo aqui proposto. Houve somente quatro casos de melhor valor de PRED e dois casos de empate, o que garante ao modelo do Dolado uma melhor capacidade de predição apesar de um pior MMRE. A Tabela I ilustra a comparação dos valores de MMRE e PRED(0.25) dos dois sistemas de Programação Genética em cada *dataset*.

A larga vantagem na comparação do modelo aqui proposto com o modelo de Dolado [1], quando observados os valores de MMRE, permite constatar que o algoritmo proposto é capaz de gerar bons resultados para diversos tipos de *dataset*. Quando comparados aos resultados das regressões lineares, os resultados de MMRE do algoritmo proposto são melhores em 11 de 12 casos, número superior à comparação dos resultados de Dolado com as mesmas regressões. Apesar disso, os valores de PRED do algoritmo proposto também são piores do que os das regressões, com quatro empates, três vitórias e cinco derrotas.

IV. Conclusão

Este artigo apresentou um algoritmo de Programação Genética para a construção de modelos de estimativa para projetos de desenvolvimento de software, representados como expressões matemáticas. O algoritmo proposto produziu melhores resultados do que um algoritmo de Programação Genética encontrado na literatura com relação à magnitude do erro médio das estimativas. Como trabalho futuro, pretendemos modificar a função de aptidão, passando a levar também em consideração os valores calculados de PRED(*0.25*) com o intuito de melhorar os resultados do algoritmo em relação para esta métrica.

Referências

[1] J.J. Dolado. On the problem of the software cost function. *Information and Software Technology*, 43(1):61–72, 2001.

[2] The Standish Group. Chaos report 2015. https://www.standishgroup.com/sample_research_files/CHAOSReport2015-Final.pdf, 2015.

Towards Assisting Architectural Changes During Code Review

Matheus Paixao
State University of Ceara
matheus.paixao@uece.br

I. Introduction

The architecture of a software system serves as the bridge between the system's requirements and its implementation [1]. In modern software systems, the architecture is not expressed by a single artefact or diagram, where different views and perspectives are separately tailored to different stakeholders [1], [2]. In this context, the structural view of a software architecture represents the dependencies between source code files alongside the organisation of the code base in its directory structure. The architecture's structural view is not only the one developers interact the most but it also represents the view practitioners use as the groundwork for the design of the other architectural views [3]. Hence, in this work, we focus our analysis on the structural view of the software's architecture.

Recent studies have empirically shown that bug-prone files are more architecturally connected than clean files [4], and that architectural flaws can lead to increased maintenance effort [5]. In addition, architectural debt has been appointed by practitioners as the most alarming type of technical debt to be accrued during software evolution [6]. In this context, code review is acknowledged as one of the main quality assurance processes adopted in modern software development [7]. In a recent empirical study [8], we have shown that developers are often not aware of the architectural impact of their changes during the process of code review. Hence, given the potential issues caused by poor architectural decisions, and the current lack of awareness regarding architectural changes during code review, we believe developers may greatly benefit from approaches to assist architectural changes during code review.

In recent years, search-based software modularisation has emerged as a promising technique to assist developers in performing architectural changes and improvements [9]. This technique employs optimisation algorithms to propose refactoring operations that lead to a better architecture as measured by quality metrics such as cohesion and coupling. However, in spite of its potential, to the best of our knowledge, we are not aware of any relevant uptake of search-based modularisation by software engineering practitioners.

We argue this is due to two main reasons. First, most search-based modularisation approaches receive the system's entire architecture as input, where the algorithm will propose refactorings to any part of the system's structure that needs improvement. It is widely known that developers are commonly not willing to modify certain parts of the system, such as legacy code and security-sensitive code. Hence, modifications proposed by search-based approaches that adopt this 'shotgun' optimisation style are likely to be ignored by developers.

Second, it has been observed that search-based modularisation approaches tend to suggest refactorings that extensively modify the existing structure of the system, in a phenomenon that has been called 'big-bang' re-modularisation [10]. Since developers tend to resist structural and architectural improvement in favour of familiarity [11], refactorings that greatly change the existing structure are likely to be dismissed.

In this extended abstract, we depict a work-in-progress search-based modularisation approach to assist developers in performing architectural changes during code review. Our approach aims at contributing towards two important research gaps. As previously mentioned, developers are commonly not aware of the architectural impact of their changes during code review [8]. Hence, this approach will be integrated into the code review pipeline to serve as an additional 'reviewer' to assist developers with architectural changes. For each new code change submitted for review, our approach will automatically assess the architectural impact of the proposed change and notify the developers involved in the review regarding the change's architectural ramifications. In addition, our approach will consider the code submitted for review and employ a search-based modularisation technique to suggest alternative code changes that exhibit higher architectural quality than the original code submitted by the developer.

In addition, this approach aims at bridging the gap between search-based modularisation and software engineering practitioners. Since the approach will only focus on the code change currently at review, the approach will not attempt to improve other parts of the system the developers are not keen to modify. Moreover, the approach will consider the original code submitted by the developer so that the search-based technique will try to find alternative code changes with a minimum number of modifications to the original code. In summary, our proposed approach will not suffer from the drawbacks of existing search-based modularisation techniques. Therefore, we believe our approach has the potential to be widely adopted by software engineering practitioners.

II. An Approach to Assist Architectural Changes During Code Review

Our approach to assist architectural changes during code review is composed of five phases, as detailed next.

A. Phase 1: Computing the architectural impact of previous code changes

When first integrated into the code review system of a certain project, our approach will first compute the architectural impact of all the previous code changes in the project.

Hence, for each reviewed code change, our approach will compute the architectural difference between the code base before and after the code change took place. The architectural difference will be computed according to the *a2a* metric [3], described as follows.

$$a2a(A_i, A_j) = (1 - \frac{mto(A_i, A_j)}{aco(A_i) + aco(A_j)}) \times 100\% \quad (1)$$

where, $mto(A_i, A_j)$ indicates the minimum number of operations needed to transform architecture A_i into A_j; and $aco(A_i)$ indicates the number of operations needed to construct architecture A_i from a null architecture $A_\emptyset$.

B. Phase 2: Identifying significant architectural changes

Not all code changes cause an architectural impact. Small changes that are localised in a single file, for example, are not likely to modify the system's structure. Hence, in Phase 2, for each code change submitted for review, our approach will identify whether the code change causes a significant impact on the system's architecture. If that is the case, the tool proceeds into the next phases; otherwise, the code change is ignored.

Code changes are considered to perform significant architectural changes when identified as outliers by Tukey's method, as described in our previous study [8]. This heuristic compares the $a2a$ value of the submitted code change to the distribution of previously submitted changes computed in Phase 1. This procedure guarantees that our approach will only provide assistance and recommendations to code changes that actually impact the system's structure.

C. Phase 3: Pre-processing the code change

When a code change is considered to perform significant architectural changes, our approach will employ static analysis to extract the structural architecture of the system in the form of a Module Dependency Graph (MDG). A MDG is a directed graph $G(C, D)$ where the set of nodes C represents the classes in the system and D represents the dependencies between classes. In our approach, a dependency between two classes is formed when we observe methods' calls and/or variables' access between classes.

D. Phase 4: Generating alternative code changes through search-based modularisation

In Phase 4, we employ the search-based modularisation technique we proposed in a previous work [12] to find alternative code changes that exhibit a tradeoff between architectural quality and modifications to the original code. Hence, we employ structural coupling as a proxy of architectural quality:

$$StrCop(A) = \frac{\sum_{i=1}^{P} FanOut_{p_i}}{P} \quad (2)$$

where, P indicates the number of packages in architecture A, and $FanOut_{p_i}$ indicates that number of classes inside package p_i that depend on classes outside p_i.

In summary, our approach consists of an optimisation problem in which we try to *max* structural coupling ($StrCop$) and *min* architectural change ($a2a$), when compared to the original code change submitted for review. Initially, we plan to employ the Two-Archive Genetic Algorithm as our optimisation engine due to its success when used in multiobjective search-based modularisation approaches in the literature [9], [12].

E. Phase 5: Recommending alternative code changes

The final phase of our approach consists of recommending the alternative code changes computed in Phase 4 as feedback during the reviewing process. Since we employ a multiobjective optimisation procedure, we need to select a subset of solutions from the Pareto front to be recommended to the developers. To do this, we make use of the architectural impact of previous changes as computed in Phase 1. Through a historical analysis of the previous changes, we can identify levels of architectural impact that developers commonly tolerated during the system's evolution. Hence, by filtering the alternative code changes based on the architectural impact developers are already accustomed to, we enhance the chances of providing meaningful recommendations to developers.

REFERENCES

[1] N. Rozanski and E. Woods, *Software Systems Architecture: Working With Stakeholders Using Viewpoints and Perspectives*, 2nd ed. Addison Wesley, 2011.

[2] P. B. Kruchten, "The 4+1 view model of architecture," *IEEE software*, vol. 12, no. 6, pp. 42–50, 1995.

[3] T. Lutellier, D. Chollak, J. Garcia, L. Tan, D. Rayside, N. Medvidovic, and R. Kroeger, "Measuring the Impact of Code Dependencies on Software Architecture Recovery Techniques," *IEEE Transactions on Software Engineering*, vol. 44, no. 2, pp. 159–181, feb 2018.

[4] R. Schwanke, L. Xiao, and Y. Cai, "Measuring architecture quality by structure plus history analysis," in *Proceedings of the 35th International Conference on Software Engineering (ICSE '13)*. IEEE, May 2013.

[5] L. Xiao, Y. Cai, R. Kazman, R. Mo, and Q. Feng, "Identifying and quantifying architectural debt," in *Proceedings of the 38th International Conference on Software Engineering (ICSE '16)*. New York, USA: ACM Press, 2016, pp. 488–498.

[6] N. A. Ernst, S. Bellomo, I. Ozkaya, R. L. Nord, and I. Gorton, "Measure it? manage it? ignore it? software practitioners and technical debt," in *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering (ESEC/FSE '15)*. Bergamo, Italy: ACM Press, 2015.

[7] A. Bacchelli and C. Bird, "Expectations, outcomes, and challenges of modern code review," in *2013 35th International Conference on Software Engineering (ICSE)*. IEEE, may 2013, pp. 712–721.

[8] M. Paixao, J. Krinke, D. Han, C. Ragkhitwetsagul, and M. Harman, "The Impact of Code Review on Architectural Changes," *IEEE Transactions on Software Engineering*, 2019.

[9] K. Praditwong, M. Harman, and X. Yao, "Software Module Clustering as a Multi-Objective Search Problem," *IEEE Transactions on Software Engineering*, vol. 37, no. 2, pp. 264–282, mar 2011.

[10] M. Hall, M. A. Khojaye, N. Walkinshaw, and P. McMinn, "Establishing the Source Code Disruption Caused by Automated Remodularisation Tools," in *2014 IEEE International Conference on Software Maintenance and Evolution*. IEEE, sep 2014, pp. 466–470.

[11] M. Wermelinger, Y. Yu, A. Lozano, and A. Capiluppi, "Assessing architectural evolution: a case study," *Empirical Software Engineering*, vol. 16, no. 5, pp. 623–666, oct 2011.

[12] M. Paixao, M. Harman, Y. Zhang, and Y. Yu, "An Empirical Study of Cohesion and Coupling: Balancing Optimization and Disruption," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 3, jun 2018.

Uso de Busca Local no Desafio Regex Golf

André de Almeida Farzat
PPGI - Programa de Pós-Graduação em Informática
Univ. Federal do Estado do Rio de Janeiro - UNIRIO
Rio de Janeiro, Brasil
andre.farzat@uniriotec.br

Márcio de Oliveira Barros
PPGI - Programa de Pós-Graduação em Informática
Univ. Federal do Estado do Rio de Janeiro - UNIRIO
Rio de Janeiro, Brasil
marcio.barros@uniriotec.br

***Resumo*—Regex Golf é um desafio que consiste em encontrar a menor expressão regular possível dado um conjunto de frases para realizar *match* e outro conjunto para não realizar *match*. O desafio é considerado um problema NP-completo e buscas heurísticas são métodos bem aceitos para a resolução de problemas com este nível de complexidade. Neste artigo, apresentaremos um algoritmo de busca local, em conjunto de um algoritmo compactador de expressões regulares, para encontrar soluções válidas para instâncias do desafio Regex Golf.**

***Index Terms*—expressão regular, busca local, prog. automática**

I. Introdução

Expressões regulares são uma tecnologia com uso crescente no desenvolvimento de software, a medida que aumenta o número de aplicações que analisam documentos de texto não estruturados e processam dados disponíveis na Web. Expressões regulares são utilizadas para identificar termos de interesse dentro de um documento, permitindo que estes termos sejam encontrados tanto em suas formas normais quanto em inflexões. Elas também são utilizadas para identificar regiões no código-fonte de páginas Web e outros documentos *online* (como e-mails, páginas de *Wiki*, *blogs*, etc.) de onde possam ser extraídas informações de interesse da aplicação.

Mesmo com o crescente aumento do interesse pelas expressões regulares, ainda existem poucos trabalhos relacionados a automatizar a sua criação [1]. Algumas das técnicas de criação automática de expressões regulares utilizam uma lista com exemplos de trechos de texto que se deseja extrair para garantir a qualidade da expressão regular gerada [2]. Porém, essa abordagem tende a criar expressões muito amplas (por exemplo, `.*`) pela falta de restrições que poderiam ser impostas por contra-exemplos do que se pretende capturar.

Em 2014, Peter Norvig publicou um *blog post* apresentando um desafio chamado *Regex Golf* [3]. Ele é baseado nos desafios do tipo *Code Golf*, onde o objetivo é criar o menor código ou o menor algoritmo possível para resolver um problema. O desafio *Regex Golf* consiste em criar a menor expressão regular que faça *match* em todas as frases de uma lista (*match list*) e **não** faça *match* em nenhuma frase de uma segunda lista (*unmatch list*) [4]. Na Tabela I temos um exemplo de instância do problema: uma *match list* com os nomes dos livros da série "Harry Potter" e uma *unmatch list* com os livros da série "O guia dos mochileiros da galáxia", ambas em inglês.

Tabela I
Exemplo de instância do *Regex Golf*

Match list	**Unmatch list**
the philosopher's stone	the hitchhiker's guide to the galaxy
the chamber of secrets	the restaurant at the end of the universe
the prisoner of azkaban	life, the universe and everything
the goblet of fire	so long, and thanks for all the fish
the order of the phoenix	mostly harmless
the half-blood prince	and another thing
the deathly hallows	

Uma solução dessa instância é a expressão regular `^the\s[^r][^i]` onde `^` é o marcador de início de linha, usado para determinar que a expressão regular deve procurar pela *string* que comece no início da linha, `the` simboliza uma *string* literal, `\s` marca um espaço em branco, `[^r]` indica que o caractere seguinte não pode ser a letra `r` e `[^i]` indica que o próximo caractere não pode ser `i`.

Um algoritmo capaz de vencer o desafio *Regex Golf* é uma contribuição relevante para o desenvolvimento de software de extração de textos de documentos semi-estruturados utilizando expressões regulares, usando trechos de texto de outras partes do documento, como contra-exemplos para criar a expressão regular para capturar a informação de interesse. Além disso, a redução do tamanho da expressão regular tem como vantagens torná-la mais simples de processar pelo computador, acelerando o processamento dos documentos, e mais simples de entender por um desenvolvedor.

Neste artigo, descrevemos um algoritmo de busca local que, combinado a um compactador de expressões regulares cria uma nova heurística para o desafio *Regex Golf*. Realizamos testes em 15 instâncias do desafio e comparamos nossa proposta ao algoritmo exato [3] e a um algoritmo de Programação Genética, ambos criados para o desafio [4]. O algoritmo proposto neste artigo obteve resultados competitivos com o algoritmo exato e próximos (ou melhores) aos do algoritmo de Programação Genética em um menor tempo de execução.

II. Proposta de solução

Na meta-heurística de busca local proposta, uma solução é representada por uma árvore onde os nós terminais são os caracteres (literais) e os nós funções são operadores da expressão regular. A Figura 1 representa a expressão regular `^ab[^c]` no formato de árvore sintática. Os nós terminais são representados pelo caractere que descreve o literal associado. Os nós funções contêm os caracteres que representam o

operador da expressão regular, utilizando o caractere • como um marcador para o local que os caracteres ocuparão quando a expressão regular for convertida para o formato texto. A Tabela II apresenta todos os possíveis nós funções considerados no algoritmo proposto e as suas representações.

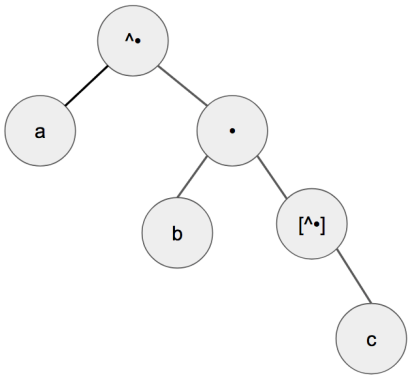

Figura 1. Representação em árvore sintática da expressão regular *^ab[^c]*.

Tabela II
OPERADORES DA EXPRESSÃO REGULAR E SUA REPRESENTAÇÃO

Operador	Representação	Operador	Representação
concatenation	•	alternative	•|•
negation	[^•]	line begin	^•
line ending	•$	zero or more	•*
one or more	•+	repetition	•#
list	[•]	range	[•-•]
optional	•?	group	(•)
any char	.•	backref	\#•

Uma solução é considerada viável quando ela representa uma expressão regular que faz *match* em todas as frases da *match list* e não faz *match* em nenhuma frase da *unmatch list*. Para calcular o *fitness* dessa solução, a expressão regular é executada para cada frase da *match list* e da *unmatch list*. Quando ocorre um *match* em uma frase da *match list*, a solução ganha um ponto. Quando ocorre um *match* em uma frase da *unmatch list*, a solução perde um ponto. A pontuação máxima que uma solução pode atingir é o número de frases na *match list*. Caso duas soluções possuam a mesma pontuação na contagem baseada nas listas, os critérios de desempate são o número de caracteres na solução (quanto menor, melhor), o número de *matches* na *match list* e a ordem de criação (a primeira solução gerada tem prioridade sobre as demais).

A execução do algoritmo é iniciada com um indivíduo gerado aleatoriamente. A vizinhança é construída criando novos indivíduos utilizando nós funções gerados com base nos operadores apresentados na Tabela II, em conjunto com caracteres extraídos das frases da instância do problema. A solução atual é comparada com todas as soluções de sua vizinhança. O algoritmo salta para a melhor solução encontrada na vizinhança e o processo de busca local é reiniciado a partir desta solução. Se a vizinhança não apresentar nenhuma solução melhor do que a solução atual, o algoritmo compactador de expressões regulares é executado sobre o melhor indivíduo encontrado até o momento e o processo de busca local é reiniciado a partir do resultado da compactação.

O algoritmo compactador de expressões regulares tem o intuito de reduzir a quantidade de caracteres de uma expressão regular, porém mantendo o mesmo número de *matches* e *unmatches* nas listas. Este algoritmo aplica reduções em cada nó da árvore sintática e tenta transformar cada ramo em um ramo que realize um *match* equivalente com uma quantidade menor de caracteres. Por exemplo, a expressão regular `abccccdef` por ser convertida para `abc{5}def`.

III. AVALIAÇÃO

O algoritmo proposto se mostrou um bom competidor para o desafio *Regex Golf*, apresentando resultados superiores ao do algoritmo baseado em Programação Genética em 5 das 15 instâncias. Em duas instâncias a busca local vence por utilizar os operadores *one or more* e *zero or more*, ignorados pelo outro algoritmo [4]. Os três algoritmos (o proposto, o exato e a programação genética) empatam nas duas instâncias mais simples. A busca local vence o algoritmo exato em apenas uma instância e ambos algoritmos (genético e busca local) perdem ou empatam nas outras instâncias na comparação com o algoritmo exato. Por utilizar o operador *alternative* com muita frequência, algumas das soluções encontradas pelo algoritmo exato possuem um número de caracteres muito superior às soluções da busca local e do algoritmo genético. Porém, ele sempre encontra uma solução com *match* e *unmatch* nas listas em todas as instâncias.

IV. CONCLUSÃO

Este trabalho demonstrou que algoritmos mais simples, como a busca local, podem gerar resultados competitivos com algoritmos mais complexos, como a Programação Genética, na geração de soluções para o desafio *Regex Golf*. Um algoritmo mais simples pode permitir a inclusão de operadores não usados em trabalhos anteriores e trabalhos futuros podem avaliar operadores que não foram considerados até aqui, como *negative*, *positive lookahead* e repetição *greedy*, *lazy* e com *possessive quantifiers* [5]. Também pretendemos estudar o comportamento do parâmetro que determina o tamanho das árvores geradas no início ou reinício da busca.

REFERÊNCIAS

[1] Y. Li, R. Krishnamurthy, S. Raghavan, S. Vaithyanathan, and H. Jagadish, "Regular expression learning for information extraction," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2008, pp. 21–30.

[2] F. Brauer, R. Rieger, A. Mocan, and W. M. Barczynski, "Enabling information extraction by inference of regular expressions from sample entities," in *Proceedings of the 20th ACM international conference on Information and knowledge management*. ACM, 2011, pp. 1285–1294.

[3] P. Norvig, "Regex golf with peter norvig," https://www.oreilly.com/learning/regex-golf-with-peter-norvig, 2014, visitado em: 26-05-2019.

[4] A. Bartoli, A. De Lorenzo, E. Medvet, and F. Tarlao, "Playing regex golf with genetic programming," in *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*. ACM, 2014, pp. 1063–1070.

[5] J. E. Friedl, *Mastering Regular Expressions: Understand Your Data and Be More Productive*. O'Reilly Media, Inc., 2006.

Algoritmo Genético para Recuperar Arquiteturas de Sistemas Pela Fusão de Diagramas de Classe UML

Wesley Klewerton Guez Assunção*, Jabier Martinez†
*Universidade Tecnológica Federal do Paraná, Toledo, Brasil
Email: wesleyk@utfpr.edu.br
†Tecnalia, Derio, Espanha
Email: jabier.martinez@tecnalia.com

***Resumo*—A necessidade de uma visão arquitetural para manter e evoluir um conjunto de variantes é fundamental. Uma arquitetura pode ser obtida pela fusão de diagramas UML, mas esta é uma atividade complexa, devido a possibilidade de diferentes combinações e a existência de conflitos durante o processo. Este artigo apresenta melhorias em uma abordagem de fusão de diagramas UML baseada em um Algoritmo Genético. As melhorias referem-se a uma melhor estratégia de comparação de diagramas e a implementação concorrente da função objetivo. Além diso, são apresentadas três oportunidades de pesquisa para estender a abordagem descrita.**

I. Introdução

O desenvolvimento de software é uma atividade complexa, custosa, e propensa a falhas. Para contornar estas situações, o reúso de sotware é uma estratégia bem estabelecida [1]. Qualquer tipo de artefato de software pode ser reusado. Na prática, geralmente o reúso é aplicado utilizando-se uma estratégia *ad hoc*, também chamada de "copia e cola" [2]. Nessa estratégia os artefatos são copiados e modificados para atenderem os novos requisitos. A estratégia *ad hoc* é fácil de ser aplicada, não requer grande investimento antecipado, e gera resultados a curto prazo. Contudo, a manutenção e evolução simultânea de um grande número de variantes independentes é uma tarefa complexa. Por exemplo, muitas vezes só um desenvolvedor é especialista em uma parte do software ou as interações entre diferentes funcionalidades implicam em restrições, o que deve ser tratado cuidadosamente [3].

Uma alternativa para lidar com o problema exposto é a recuperação de uma arquitetura global que represente da melhor maneira as variantes do sistema. Uma arquitetura de software é um artefato que provê uma visão de alto nível da estrutura de um sistema e permite sua análise, compreensão, e facilita as decisões de projeto e reúso [4]. A maioria dos trabalhos encontrados na literatura lidam com a recuperação de arquiteturas a partir do código fonte, como por exemplo [5].

Em um trabalho recente foi proposto uma abordagem baseada em Algorimto Genético (AG) para fusão de variantes de diagramas de classe UML para recuperar uma arquitetura inicial [6]. No AG cada indivíduo é uma arquitetura completa, representada por um diagrama de classe UML, e a função objetivo avalia a similaridade do indivíduo em relação a todos os diagramas de classe UML que representam as variantes.

Apesar da abordagem proposta por [6] lidar satisfatoriamente com sistemas com até 32 variantes, observou-se que o processo de cálculo da função objetivo é muito custoso. Além disso, a análise de similaridade entre os diagramas é feita simplismente por meio da comparação de nomes dos elementos. Para contornar essas limitações, o objetivo deste trabalho é apresentar avanços iniciais no uso de uma função objetivo concorrente no processo de fusão e empregar uma comparação estrutural dos elementos. Além disso, apontam-se oportunidades de pesquisa para trabalhos futuros.

II. Fusão de Diagramas de Classe UML

A fusão de diagrams de classes UML proposta por Assunção et al. [6] é baseada em uma AG que busca arquiteturas que possuam a maior similaridade quando comparadas com as variantes existentes. Basicamente o processo evolutivo do AG tem por objetivo construir uma arquitetura que possui a maior quantidade de elementos espalhadas dentre as variantes, contudo, lidando com possíveis conflitos no processo de fusão.

A. Características da Abordagem Original

A seguir são apresentadas brevemente as características da abordagem proposta por Assunção et al. Mais detalhes são encontrados no trabalho original [6].

Cada indivíduo é um diagrama de classes completo, representado utilizando-se a Eclipse EMF UML2[1], descrita por meio do metamodelo Ecore.

O cálculo da função objetivo é obtido através da análise de diferenças existentes, computadas pela comparação pareada da arquitetura candidata para todos os diagramas variantes. A quantidade de diferenças de cada comparação é somada, gerando o valor final de aptidão da solução. A equação abaixo apresenta a função objetivo chamada *Model Similarity (MS)*, que expressa o grau de similaridade de uma arquitetura candidata para um conjunto de diagrama de classes UML variantes. A função *diff* representa a comparação pareada dos diagramas. Para a construção das arquiteturas candidatas são consideradas somente as diferenças que existem na variante *v* mas faltam no *candidate model*.

$$MS = \sum_{v \in Variants} diff(candidate\_model, v)$$

Os operadores evolucionários tomam por base as diferenças encontradas na comparação pareada de duas arquiteturas can-

[1]https://www.eclipse.org/modeling/mdt/?project=uml2

didatadas. Os indivíduos são selecionados para a aplicação dos operadores por meio da estratégia baseada em torneio.

No cruzamento, após a análise de diferenças, dois descententes são gerados. Em um dos filhos todas as diferenças são adicionadas e no outro todas as diferenças são removidas. Na mutação, somente uma das diferenças é selecionada aleatoriamente e aplicada.

A população inicial é gerada pela clonagem dos diagramas variantes. Caso a quantidade de variantes seja menor que o tamanho da população, múltíplos clones são gerados.

B. Melhorias no Processo de Fusão

Nesta seção são apresentadas as melhorias implementas na abordagem original de Assunção et al. [6]. Este é um trabalho em andamento, portanto, o estudo experimental para avaliar a abordagem com as melhorias ainda está em andamento.

No trabalho original, o processo de comparação entre os diagramas de classe considera nomes qualificados dos elementos UML para fazer a análise de diferenças. A primeira diferença deste trabalho refere-se a adição da estrutura hierárquica dos elementos, o que faz com que a comparação entre modelos considere melhor quando dois elementos são similares ou não.

O outro avanço em relação a abordagem original está no processo de cálculo da função objetivo. No trabalho original a função objetivo executa a análise de diferenças mediante uma comparação pareada sequencial. Contudo, quando os diagramas de classe UML são grandes, ou existem muitas variantes, esse processo requer um tempo de execução elevado. Uma versão melhorada da função objetivo executa as comparações seguindo uma estratégia concorrente, usando os benefícos de processadores com múltiplos-núcleos que equipam praticamente todos os computadores atuais. Em análises experimentais iniciais foi possível observar um ganho significativo no tempo de execução.

III. Oportunidades de Pesquisa

Além das melhorias mencionadas na Seção II-B, durante a condução das atividades deste trabalho, foram identificadas possibilidades de trabalhos futuros, conforme segue:

- *Mapeamento entre elementos da arquitetura candidata e as variantes que os possuem*: o objetivo aqui é manter a rastreabilidade entre os elementos que estão na arquitetura candidata e as variantes, ou variante única, de onde esses elementos vieram, ou que possuem estes elementos. Esta rastreabilidade poderá dar suporte a manutenção das variantes, planejamento mais elaborado de evoluções do sistema, e ser uma fonte de informação para a modernização do sistema, como por exemplo, a migração para uma linha de produtos de software [7].
- *Incorparação da abordagem definida em uma ferramenta industrial*: Para viabilizar e motivar a utilização prática da abordagem descrita neste trabalho, observa-se a necessidade de incluí-lá em uma ferramenta que possibilite uma fácil utilização, levando-se em consideração aspectos de usabilidade. Uma possibilidade é a inclusão desta abordagem na ferramenta de código-aberto But4Reuse[2], que fornece um ambiente unificado para mineração de artefatos de software em variantes de sistemas [8].
- *Eliminação de elementos que introduzem "sujeira" na arquitetura*: algumas variantes de sistema podem ter sido profundamente modificadas, tornando-se muito diferentes de seus predecessores. Essas variantes são chamadas de *"Outliers"*. Estudos apontam que a inclusão dessas variantes podem prejudar o processo de recuperação de arquiteturas [9]. Uma oportunidade de pesquisa é identificar e elimitar elementos *outliers* durante o processo evolutivo de fusão de diagramas de classes UML.

IV. Considerações Finais

Este artigo apresenta avanços iniciais no processo de fusão de diagramas de classes UML para a obtenção de arquiteturas de sistemas. A partir de um trabalho relacionado duas melhorias são propostas: (i) considerar a estrutura herárquica no processo de comparação de diagramas e (ii) executar as comparações na função objetivo utilizando-se processamento concorrente. Estas duas melhorias estão implementadas e em processo de avaliação experimental.

Além das melhorias já implementadas, três oportunidades de pesquisa para estender a abordagem atual são apresentadas. Essas oportunidades de pesquisa referem-se a manter a rastreabilidade entre variantes e arquitetura candidata, incorporação da abordagem em uma ferramenta industrial, e eliminação de "sugeira" durante o processo evolutivo.

Referências

[1] C. W. Krueger, "Software reuse," *ACM Computing Surveys*, vol. 24, no. 2, pp. 131–183, 1992.

[2] C. Riva and C. Del Rosso, "Experiences with software product family evolution," in *International Workshop on Principles of Software Evolution*, 2003, pp. 161–169.

[3] D. Faust and C. Verhoef, "Software product line migration and deployment," *Software: Practice and Experience*, vol. 33, no. 10, pp. 933–955, 2003.

[4] L. Dobrica and E. Niemela, "A survey on software architecture analysis methods," *IEEE Transactions on Software Engineering*, vol. 28, no. 7, pp. 638–653, 2002.

[5] K. Jeet and R. Dhir, "Software architecture recovery using genetic black hole algorithm," *ACM SIGSOFT Software Engineering Notes*, vol. 40, no. 1, pp. 1–5, 2015.

[6] W. K. G. Assunção, S. R. Vergilio, and R. E. Lopez-Herrejon, "Discovering Software Architectures with Search-Based Merge of UML Model Variants," in *Mastering Scale and Complexity in Software Reuse: 16th International Conference on Software Reuse (ICSR)*, G. Botterweck and C. Werner, Eds. Springer, 2017, pp. 95–111.

[7] W. K. G. Assunção, R. E. Lopez-Herrejon, L. Linsbauer, S. R. Vergilio, and A. Egyed, "Reengineering legacy applications into software product lines: a systematic mapping," *Empirical Software Engineering*, vol. 22, no. 6, pp. 2972–3016, Dec 2017.

[8] J. Martinez, T. Ziadi, T. F. Bissyandé, J. Klein, and Y. L. Traon, "Bottom-up technologies for reuse: Automated extractive adoption of software product lines," in *39th International Conference on Software Engineering Companion*, ser. ICSE-C '17. Piscataway, NJ, USA: IEEE Press, 2017, pp. 67–70.

[9] C. Lima, W. K. G. Assunção, J. Martinez, I. do Carmo Machado, C. von Flach G. Chavez, and W. D. F. Mendonça, "Towards an automated product line architecture recovery: The apo-games case study," in *VII Brazilian Symposium on Software Components, Architectures, and Reuse*, ser. SBCARS '18. New York, NY, USA: ACM, 2018, pp. 33–42.

[2]https://but4reuse.github.io/

iMOCell: Uma Proposta de Algoritmo Evolucionário Multiobjetivo Baseado em Otimização Interativa e Aprendizado Supervisionado

Denis Sousa[1], Pamella Soares[1], Allysson Allex Araújo[2], Raphael Saraiva[1], e Jerffeson Souza[1]

[1]Universidade Estadual do Ceará, Fortaleza, Brasil
[2]Universidade Federal do Ceará, Crateús, Brasil
Grupo de Otimização em Engenharia de Software (GOES)

***Resumo*—Diversos problemas explorados pela Engenharia de Software Baseada em Busca lidam com questões as quais são inerentemente subjetivas e demandam influência humana. Tal intervenção se faz necessário em decorrência da capacidade que os engenheiros de software têm em identificar elementos subjetivos os quais são difíceis de serem previstos ou modelados matematicamente. Nesse sentido, abordagens baseadas em *Interactive Evolutionary Computation* vêm sendo propostas para retornar soluções que correspondam a um *trade-off* adequado entre as métricas próprias do problema e as preferências humanas. Entretanto, tais propostas enfrentam desafios associados ao desenvolvimento estrategias de interação mais eficientes e, inclusive, questões relacionadas ao excesso de intervenção do usuário o qual ocasiona no problema da fadiga humana. Diante destas motivações, o presente trabalho objetiva propor uma adaptação ao algoritmo MOCell, denominado iMOCell, que incorpora as preferências do tomador de decisão durante o processo de avaliação de dominância de soluções. A fim de contornar o problema de fadiga humana, a presente proposta adota um modelo de aprendizado de máquina para substituir o usuário após um determinado número de interações.**

I. Introdução

Durante o desenvolvimento de softwares, diversos desafios ocorrem devido aos variados critérios conflitantes existentes. Assim, os *stakeholders* podem deparar-se com problemas considerados inerentemente complexos por possuírem mais de um objetivo. Neste contexto, tem-se a Engenharia de Software Baseada em Busca (em inglês, *Search Based Software Engineering* - SBSE) como a área de estudos responsável por modelar problemas da Engenharia de Software (ES) como problemas de otimização. Tal área tem sido pertinente ao propor abordagens que se utilizam de Algoritmos Evolucionários Multiobjetivos na resolução de problemas da ES.

Por tratar-se de problemas multiobjetivos, uma considerável quantidade de soluções não dominadas podem ser retornadas ao Tomador de Decisão (TD). Essas soluções compõem a frente de Pareto, podendo ou não corresponder às preferências subjetivas do TD. A escolha da "melhor" solução da frente de Pareto exige um alto esforço cognitivo por parte deste, o que se torna crítico à medida que aumenta-se o número de objetivos [5]. Há, ainda, um desafio frequente relacionado a modelagem matemática das preferencias humanas as quais apresentam um caráter subjetivo, dinâmico e igualmente relevante para guiar o processo de busca [9]. Assim, faz-se necessário a investigação de abordagens que se utilizam das preferências humanas e intuição a fim de influenciar a busca [3].

Segundo Miettinen [7], as preferências do TD podem ser adicionadas antes do processo de busca iniciar (*a priori*), após (*a posteriori*) ou durante (interativamente). Em especial, abordagens interativas permitem que o TD participe ativamente no processo de busca guiando as soluções conforme seus critérios. A área denominada *Interactive Evolutionary Computation* (IEC), por exemplo, explora o uso das avaliações humanas para guiar o processo de busca em abordagens evolucionárias [10]. Fundamentados em tal área, Tonella *et al.* [11] e Ghannem *et al.* [4] propõem, respectivamente, o uso de *Interactive Genetic Algorithm* para os problemas de priorização de requisitos e sugestões de refatoração. Todavia, abordagens interativas podem enfrentar o problema da fadiga humana, o qual ocorre quando se demanda em excesso intervenções do TD [10]. Para lidar com a fadiga humana, Kamalian *et al.* [6] sugerem o uso de preditor baseado em fuzzy, enquanto Wang *et al.* [12] avaliam a criação de uma escala absoluta para melhorar a previsão de avaliações humanas em IEC. Finalmente, Araújo *et al.* [1] propõem um modelo de aprendizado responsável por aprender as avaliações subjetivas do TD em relação a qualidade de uma *release* de software.

Diante de tais motivações, o presente trabalho objetiva propor uma adaptação ao Algoritmo Evolucionário Multiobjetivo MOCell [8], denominado iMOCell (*Interactive* MOCell), que permitirá ao TD incorporar interativamente suas preferências durante a avaliação da dominância de soluções. Paralelo a esse processo, a fim de reduzir o problema de fadiga humana, propõe-se utilizar um modelo de aprendizado de máquina que será treinado para simular o usuário com o objetivo de substituí-lo na avaliação subjetiva das soluções apresentadas interativamente, conforme explorado em [1]. Entretanto, vale destacar que tanto o processo de avaliação de soluções, quanto o de aprendizado serão diferentes da referida referência.

D. M. Sousa, Universidade Estadual do Ceará, Fortaleza, Ceará, Brazil denis.sousa@aluno.uece.br.

II. Abordagem Proposta

A Figura 1 apresenta um fluxograma da adaptação proposta neste trabalho. Inicialmente, têm-se os passos originais do algoritmo MOCell [8]. Uma população inicial é criada e seus indivíduos são organizados em uma estrutura toroidal. Em seguida, para cada indivíduo, ocorre o processo de Seleção dos pais (Passo 1), sendo estes escolhidos entre seus oito vizinhos. Em seguida, realiza-se o processo de Cruzamento e Mutação (Passo 2) para geração do novo indivíduo.

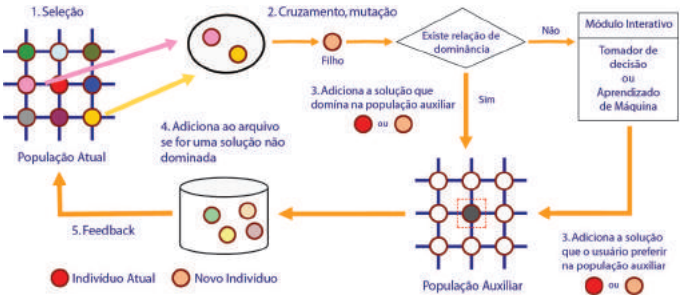

Figura 1. Fluxograma do iMOCell.

Após aplicações dos operadores genéticos, avalia-se a relação de dominância entre o indivíduo atual (indivíduo que passou pela seleção dos pais) e o novo indivíduo gerado pela recombinação dos pais. Após essa avaliação, ocorrerá o processo de inserção do indivíduo em uma população auxiliar (Passo 3). Caso exista uma relação de dominância, o indivíduo dominante será inserido na população auxiliar, na mesma posição que o individuo corrente. Originalmente, quando as soluções não possuem uma relação de dominância, leva-se em consideração o espalhamento dos indivíduos da vizinhança do indivíduo atual, ou seja, se o novo indivíduo tiver o melhor *Crowding Distance*, ele substituirá o indivíduo atual. Pode-se considerar a adaptação do algoritmo a partir desse passo, onde propõe-se substituir o critério de espalhamento pela preferência do TD. Nesse momento, o algoritmo apresentará as duas soluções não dominadas ao TD e o mesmo escolherá entre uma delas, substituindo, assim, o indivíduo atual. O número de interações do usuário será determinado *a priori*, representando a quantidade de vezes que ele deseja interagir no processo de busca. Dessa forma, para que as preferências do usuário sejam incorporadas de forma efetiva no processo de busca, será necessário uma considerável quantidade de interações, o que poderá se tornar um processo exaustivo ao usuário. A fim de contornar esse problema, propõe-se utilizar um modelo de aprendizado de máquina supervisionado que será treinado a partir do conjunto de soluções escolhidas e rejeitadas previamente pelo TD para que, quando o mesmo parar de interagir, o modelo substitua-o. Diferentemente da proposta em [1], onde o TD avalia a solução através de uma nota, este trabalho considera que o TD escolhe entre duas soluções conforme seus critérios.

Após o processo de inserção do indivíduo na população auxiliar, o mesmo também poderá ser adicionado a estrutura denominada *arquivo* (Passo 4), que armazena as soluções com melhor *fitness* no decorrer de todas as gerações das populações e, os ranqueiam de acordo com o *Crowding Distance*. Portanto, o indivíduo é adicionado ao *arquivo* se dominar as soluções da estrutura ou se não existir uma relação de dominância entre ele e as soluções do *arquivo*. No momento em que toda população auxiliar for gerada, a mesma substituirá a população atual e o processo de *feedback* (Passo 5) será invocado, que seguirá conforme o algoritmo MOCell.

III. Considerações Finais

O uso de abordagens interativas em SBSE têm se demonstrado bastante pertinente [3]. Porém, até o momento, ainda não se identificou propostas que permitam ao MOCell uma configuração interativa, apesar do mesmo ser reconhecido como um dos algoritmos mais utilizados em SBSE. Dessa forma, a presente proposta visa usufruir da IEC em SBSE propondo um algoritmo denominado iMOCell que incorpora as preferências do TD durante a avaliação das soluções não dominadas durante o processo evolucionário. Um modelo de aprendizado supervisionado será utilizado para substituir o TD quando este atingir o número de interações previamente estipulado. Como perspectivas futuras, pretende-se validar a proposta com o *Next Release Problem* e realizar comparações com outras abordagens interativas. A opção pelo NRP se deve ao fato do mesmo ser um problema intrínsecamente subjetivo e computacionalmente complexo [2].

Referências

[1] Allysson Allex Araújo, Matheus Paixao, Italo Yeltsin, Altino Dantas, and Jerffeson Souza. An architecture based on interactive optimization and machine learning applied to the next release problem. *Automated Software Engineering*, 24(3):623–671, 2017.

[2] Anthony J. Bagnall, Victor J. Rayward-Smith, and Ian M Whittley. The next release problem. *Information and software technology*, 43(14):883–890, 2001.

[3] Thiago Nascimento Ferreira, Silvia Regina Vergilio, and Jerffeson Teixeira de Souza. Incorporating user preferences in search-based software engineering: A systematic mapping study. *Information and Software Technology*, 90:55–69, 2017.

[4] Adnane Ghannem, Ghizlane El Boussaidi, and Marouane Kessentini. Model refactoring using interactive genetic algorithm. In *Proceedings of the 5th International Symposium on Search Based Software Engineering (SSBSE '13)*, volume 8084, pages 96–110, St. Petersburg, Russia, 24-26 August 2013. Springer.

[5] Hisao Ishibuchi, Hiroyuki Masuda, and Yusuke Nojima. Selecting a small number of non-dominated solutions to be presented to the decision maker. In *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 3816–3821. IEEE, 2014.

[6] Raffi Kamalian, Eric Yeh, Ying Zhang, Alice M Agogino, and Hideyuki Takagi. Reducing human fatigue in interactive evolutionary computation through fuzzy systems and machine learning systems. In *Fuzzy Systems, 2006 IEEE International Conference on*, pages 678–684. IEEE, 2006.

[7] Kaisa Miettinen. *Nonlinear multiobjective optimization*, volume 12. Springer Science & Business Media, 2012.

[8] Antonio J Nebro, Juan J Durillo, Francisco Luna, Bernabé Dorronsoro, and Enrique Alba. Mocell: A cellular genetic algorithm for multiobjective optimization. *International Journal of Intelligent Systems*, 24(7):726–746, 2009.

[9] Christopher L Simons, Jim Smith, and Paul White. Interactive ant colony optimization (iaco) for early lifecycle software design. *Swarm Intelligence*, 8(2):139–157, 2014.

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

[11] Paolo Tonella, Angelo Susi, and Francis Palma. Interactive requirements prioritization using a genetic algorithm. *Information and Software Technology*, 55(1):173–187, January 2013.

[12] Shangfei Wang, Xufa Wang, and Hideyuki Takagi. User fatigue reduction by an absolute rating data-trained predictor in iec. In *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, pages 2195–2200. IEEE, 2006.

Genetic Improved Topic Extraction from StackOverflow Discussions

1st Alan Bandeira
State University of Ceara
alan.bandeira@aluno.uece.br

2nd Lucas Aguiar
State University of Ceara
lucas.aguiar@aluno.uece.br

3rd Matheus Paixao
State University of Ceara
matheus.paixao@uece.br

4th Paulo Maia
State University of Ceara
pauloh.maia@uece.br

I. Introduction

Microservices are a novel architectural style in which an application is built as a collection of services that cooperate to deliver business functionalities [1]. In a microservices-based system, a service represents an independent, small and decoupled component responsible for a single functionality, usually associated to a business domain. Components communicate through lightweight web protocols, such as HTTP, and run in a single process on top of container technology [2].

The advantages of microservices-based architectures are highlighted when compared to traditional monolithic architectures. A monolithic application characterises a software system as a single logic unit, responsible for providing all business functionalities. The services provided by a monolithic application share computational resources and are scaled together, which can cause inefficiency in resource usage, and affect the systems' end users [1], [3].

In contrast, a microservices application distributes its functionalities among small services that are developed and maintained independently. Due to its flexibility, microservices favour horizontal scaling, ensuring an efficient form of computational resource management on cloud environments [4].

With the emergence of microservices as a promising architectural style, relevant members of the industrial tech scenario such as Netflix, Microsoft and Amazon, have adopted and developed technologies towards this new approach, e.g., Netflix Eureka, Azure Service Fabric and Amazon SNS, respectively.

While new technologies arise, the developers' community delve into new concepts and patterns, where Q&A platforms are commonly employed to debate over technical and conceptual questionings that permeate their work and studies.

These platforms tend to aggregate valuable information about software development in the shape of text and code snippets [5], which could be leveraged by software engineering developers and researchers [6]. In this context, StackOverflow is the most prominent Q&A website, where a plethora of work has leveraged its content to perform empirical studies such as Barua et al. [7], Baltes et al. [8] and Ahasanuzzaman et al. [9].

In a previous work [10], we investigated what subjects microservices developers discuss on StackOverflow, and, also, which ones had received the most attention. The study was conducted aiming at unveiling which topics had caught the attention of the microservice's developer community. The results can assist researchers and developers of any level understand the theory behind microservices and the main related technologies.

We classified the discussions in a semi-automated fashion. First, we employed manual analysis to identify three subgroups, namely technical, conceptual and non-related. Second, we employed Latent Dirichlet Allocation (LDA) [11] to automatically extract discussion's subjects from technical and conceptual subgroups. As a result, we presented a categorisation derived from the topics analysis, and the most relevant topics based on popularity metrics from StackOverflow.

The categorisation realised in the aforementioned study heavily depends on the results of the LDA topic modelling. The LDA algorithm belongs to a class of machine learning techniques, known as unsupervised learning, and its results depend on parameter calibration such as the number of topics, the number of words per topic and the number of iterations.

LDA-based topic modelling has been commonly employed as an information retrieval (IR) technique for software engineering tasks [12]. The results provided by machine learning algorithms and optimisation techniques can suffer from poor parameter configuration and biased assumptions of the problem's nature [12], [13]. For example, previous studies indicate that statistical models that work with corpus data tend to capture local regularities with a higher frequency [14], which could explain the need of specific calibrations for different types of SE data.

The LDA technique can be seen as a clustering algorithm, capable of distributing a corpus of documents in clusters by grouping them based on the similarities between topics' distributions [12]. In this context, a cluster can be defined as a group of documents that shares a dominant topic, where a dominant topic is the most similar topic to every document in the cluster [12].

In our previous work [10], the LDA technique was applied through the Mallet tool [15]. The selected configuration was empirically inferred from tests with the curated dataset, following the best practices indicated in the literature [16]. Although the LDA results were acceptable to our goal, we still observed points for improvement. For instance, one of the conceptual topics could not be clearly associated with any microservices' conceptual subject, therefore excluding a subset of discussions from the classification.

Hence, to tackle the aforementioned problem of low per-

formance in our application of LDA to identify subjects of microservices discussion on StackOverflow, we propose the application of a Genetic Algorithm to improve the LDA parameter calibration. To achieve this, we will leverage the curated dataset SOTorrent [5] and the results available in the paper's supporting website [17].

II. Improving LDA for topic extraction on StackOverflow Discussions

Our proposed approach to calibrate the LDA parameters for topic extraction in our corpus is largely inspired by the work of Panichella et al. [12] and Agrawal et al. [13]. We detail the approach next.

A. Previous Results Evaluation

First, we intend to perform a sensitivity analysis of our previous work and identify which parameter is mainly responsible for the LDA results in our particular corpus. This phase will be responsible for helping on understanding how the LDA implementation, provided by the Mallet tool, can be affected by different parameter configurations. To assess the quality of the clustering provided by LDA for different parameters, we will employ the Silhouete Coefficient:

$$s(d_i) = \frac{b(d_i) - a(d_i)}{max(b(d_i), a(d_i))} \tag{1}$$

The coefficient is calculated using topic cluster centroids as its main factor, where a cluster centroid represents the mean vector of all documents (d_i) within that cluster (C):

$$Centroid(C) = \frac{\sum_{d_i \in C} d_i}{|C|} \tag{2}$$

Following this definition, the Silhouete Coefficient is computed by the following steps:

I For each document d_i, calculate the distance from the farther document, within the cluster. This value is represented as $a(d_i)$.

II For each document d_i, calculate the distance from d_i to the centroid of the closer clusters, which d_i is not member. This value is represented by $b(d_i)$.

III For each document d_i, calculate the silhouette $s(d_i)$ as defined in equation 1.

With values ranging from -1 to 1 negative coefficients are unwanted, as they indicate that a document is closer to documents in other cluster than from its own cluster.

The quality of the LDA results will be evaluated using the mean Silhouette Coefficient and the evaluation will be applied to different parameter calibrations and the previous study configuration.

B. Experiments with the Genetic Improved LDA

Second, considering the mean Silhouette coefficient as a quality metric to assess the LDA results, we will employ a genetic algorithm approach to find the configurations of the parameters identified in the previous phase, that maximises our fitness function. For the GA setting, we will employ a 0.6 percent chance of crossover probability, 0.01 of mutation probability and a population of 100 individuals with a 2 elitism, as proposed by Panichella et al. [12]. The evolution process will cease when the population converge after 10 consecutive iterations or after 100 generations.

C. Internal and External Evaluation

With the results from the previous work and from the experiments conducted, the internal evaluation will compare the two approaches to identify if the improvement was possible in terms of mean Silhouette coefficient and if the improvement is significant. For the external validation, the topics extracted using the genetic improved LDA will be interpreted by specialists, aiming to identify the uncategorised topic and improve the ones that were categorised.

References

[1] M. Fowler and J. Lewis, "Microservices," 2014.

[2] S. Newman, *Building Microservices*. O'Reilly Media, Inc., 1st ed., 2015.

[3] N. Dragoni, S. Giallorenzo, A. L. Lafuente, M. Mazzara, F. Montesi, R. Mustafin, and L. Safina, *Microservices: Yesterday, Today, and Tomorrow*, pp. 195–216. Cham: Springer International Publishing, 2017.

[4] A. Balalaie, A. Heydarnoori, and P. Jamshidi, "Microservices architecture enables devops: Migration to a cloud-native architecture," *IEEE Softw.*, vol. 33, pp. 42–52, May 2016.

[5] S. Baltes, L. Dumani, C. Treude, and S. Diehl, "SOTorrent: Reconstructing and Analyzing the Evolution of Stack Overflow Posts," *Proceedings of the 15th International Conference on Mining Software Repositories - MSR '18*, pp. 319–330, mar 2018.

[6] A. E. Hassan, "The road ahead for mining software repositories," in *Frontiers of Software Maintenance, 2008. FoSM 2008.*, pp. 48–57, IEEE, 2008.

[7] A. Barua, S. W. Thomas, and A. E. Hassan, "What are developers talking about? an analysis of topics and trends in stack overflow," *Empirical Software Engineering*, vol. 19, no. 3, pp. 619–654, 2014.

[8] S. Baltes, C. Treude, and S. Diehl, "Sotorrent: Studying the origin, evolution, and usage of stack overflow code snippets," in *Proceedings of the 16th International Conference on Mining Software Repositories (MSR 2019)*, 2019.

[9] M. Ahasanuzzaman, M. Asaduzzaman, C. K. Roy, and K. A. Schneider, "Mining duplicate questions in stack overflow," in *Proceedings of the 13th International Workshop on Mining Software Repositories - MSR '16*, (New York, New York, USA), pp. 402–412, ACM Press, 2016.

[10] A. Bandeira, C. Medeiros, M. Paixao, and P. H. Maia, "We need to talk about microservices: an analysis from the discussions on stackoverflow," in *Proceedings of the 16th International Conference on Mining Software Repositories (MSR 2019)*, 2019.

[11] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of machine Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003.

[12] A. Panichella, B. Dit, R. Oliveto, M. Di Penta, D. Poshynanyk, and A. De Lucia, "How to effectively use topic models for software engineering tasks? An approach based on Genetic Algorithms," *Proceedings - International Conference on Software Engineering*, pp. 522–531, 2013.

[13] A. Agrawal, W. Fu, and T. Menzies, "What is wrong with topic modeling? And how to fix it using search-based software engineering," *Information and Software Technology*, vol. 98, pp. 74–88, jun 2018.

[14] A. Hindle, E. T. Barr, Z. Su, M. Gabel, and P. Devanbu, "On the naturalness of software," in *2012 34th International Conference on Software Engineering (ICSE)*, pp. 837–847, IEEE, 2012.

[15] A. K. McCallum, "Mallet: A machine learning for language toolkit." http://mallet.cs.umass.edu, 2002.

[16] S. W. Thomas, A. E. Hassan, and D. Blostein, "Mining Unstructured Software Repositories," in *Evolving Software Systems* (T. Mens, A. Serebrenik, and A. Cleve, eds.), pp. 139–162, Berlin, Heidelberg: Springer Berlin Heidelberg, 2014.

[17] A. Bandeira, C. Alberto, M. Paixao, and P. H. Maia, "Replication package and supporting webpage for the paper: We need to talk about microservices."

Explorando Perfis Técnicos e de Personalidades na Seleção e Alocação de Pessoas

Jorcyane Araújo Lima
Instituto Federal da Paraíba - IFPB
João Pessoa, Brasil
jorcyane.lima@ifpb.edu.br

Gledson Elias
Universidade Federal da Paraíba - UFPB
João Pessoa, Brasil
gledson@ci.ufpb.br

***Resumo*—Em projetos de software, os membros da equipe de desenvolvimento desempenham diferentes atividades, agrupadas em variados papéis funcionais. Neste contexto, cada papel funcional requer perfis técnicos e de personalidades específicos, que, caso não considerados, expõem o projeto de software a sérios riscos, uma vez que os membros da equipe podem desempenhar inadequadamente suas atribuições por falta de habilidades técnicas ou traços de personalidades. É importante destacar que o processo de seleção e alocação de pessoas a papéis funcionais é bastante complexo, especialmente quando realizado apenas com base na experiência do gerente de projeto, pois o número de possíveis soluções cresce exponencialmente em relação ao número de pessoas e papéis. Neste contexto, este artigo introduz uma proposta de abordagem multiobjetiva para seleção e alocação de pessoas tecnicamente qualificadas e psicologicamente adequadas para cada papel funcional do projeto de software, proporcionando ao gerente de projetos uma ferramenta de apoio à tomada de decisão.**

Palavras-chave*—*engenharia de software baseada em buscas, seleção e alocação de pessoas, tipologias de personalidades.

I. INTRODUÇÃO

Em projetos de software, as pessoas devem trabalhar em equipes, com divisão de atividades organizadas em papéis funcionais, pois um software de melhor qualidade resulta dos esforços combinados de várias perspectivas, processos mentais e valores [1]. Portanto, diferentes habilidades técnicas e traços de personalidades são requeridos para resolver os problemas associados ao desenvolvimento de software. Assim sendo, as questões mais relevantes da Engenharia de Software estão intrinsicamente relacionadas às pessoas, que são a parte mais significativa do custo de um projeto de software [2].

No que se refere aos perfis técnicos, os papéis funcionais requerem diferentes conhecimentos, habilidades e experiências, sendo imperativo selecionar e alocar pessoas com as qualificações requeridas. Note que alocar pessoas menos qualificadas pode acarretar atrasos, menor qualidade ou não conclusão de atividades. Já alocar pessoas mais qualificadas pode frustrá-las ou aumentar o custo [3].

Em relação aos perfis de personalidades, as pessoas são diferentes entre si em diversos aspectos, por exemplo, na busca de soluções para problemas, na aprendizagem, nos relacionamentos interpessoais e no processo de tomada de decisões. Estas diferenças influenciam no desempenho do profissional e da equipe, pois repercutem na motivação individual e coletiva, impactando no sucesso ou fracasso do projeto de software [4].

Neste contexto, explorando conceitos e técnicas da Engenharia de Software Baseada em Buscas, do inglês *Search Based Software Engineering* (SBSE), este artigo introduz uma proposta de abordagem multiobjetiva para seleção e alocação de pessoas tecnicamente qualificadas e psicologicamente adequadas para cada papel funcional do projeto de software, proporcionando ao gerente de projetos uma ferramenta de apoio à tomada de decisão.

II. ABORDAGEM PROPOSTA

A Fig. 1 apresenta as principais etapas envolvidas na abordagem proposta e ilustra os possíveis tipos de informações de entrada e saída de cada etapa. Adotando uma formulação multiobjetiva, as soluções recomendadas representam o conjunto de alocações não-dominadas de pessoas a papéis funcionais.

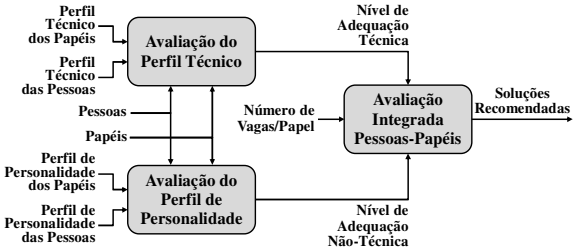

Fig. 1. Etapas da abordagem proposta

A. Avaliação do Perfil Técnico

Esta etapa realiza a avaliação dos diferentes perfis técnicos requeridos pelos papéis funcionais e os perfis técnicos dominados pelas pessoas, contrastando as habilidades, conhecimentos e experiências das pessoas em relação aos papéis. As entradas são representadas pelo **perfil técnico dos papéis** e **perfil técnico das pessoas**, gerando como resultado o **nível de adequação técnica** das pessoas aos papéis.

A modelagem matemática desta etapa será baseada em adaptações da proposta de Souza e Elias [5], que originalmente propõem mecanismos e estratégias para caracterizar aspectos técnicos de módulos de software e equipes de desenvolvimento, como por exemplo sistemas operacionais, plataformas, ferramentas, linguagens de programação e processos de desenvolvimento de software.

Como em Souza e Elias [5], na abordagem proposta, o **perfil técnico dos papéis** deve ser caracterizado adotando *tabelas de implementação*. Sendo assim, cada papel deve ter uma tabela, na qual: *(i)* linhas identificam requisitos técnicos necessários à execução do papel; e *(ii)* colunas identificam os níveis de conhecimentos requeridos. Dada a complexidade de quantificar níveis de conhecimentos, assim como Souza e Elias [5], a abordagem proposta deve representá-los com termos *fuzzy*, por exemplo: *nenhum*, *baixo*, *médio* e *alto*.

Também de forma similar a Souza e Elias [5], na abordagem proposta, o **perfil técnico das pessoas** deve ser caracterizado adotando o conceito de *formulários de avaliação*. Desta forma, cada pessoa deve preencher um formulário onde três eixos são avaliados: *anos de experiência*, *número de projetos desenvolvidos* e *número de certificações*.

Posteriormente, as tabelas de implementação e os formulários de avaliação devem ser matematicamente mapeados visando mensurar o **nível de adequação técnica** de cada pessoa em relação a cada papel. Para tal, adotando uma

política de seleção baseada em qualificação equivalente, a abordagem proposta deve favorecer soluções candidatas que aproximam os níveis de conhecimentos requeridos pelos papéis funcionais e possuídos pelas pessoas. Portanto, quanto mais próximos são esses níveis de conhecimentos, maiores devem ser as pontuações obtidas pelas soluções candidatas.

B. Avaliação do Perfil de Personalidade

Esta etapa realiza a avaliação dos diferentes perfis de personalidades requeridos pelos papéis funcionais e os perfis de personalidades que caracterizam as pessoas, contrastando os traços psicológicos das pessoas em relação aos papéis. As entradas são representadas pelo **perfil de personalidade dos papéis** e **perfil de personalidade das pessoas**, produzindo como resultado o **nível de adequação não-técnica** das pessoas aos papéis funcionais.

Na abordagem proposta, a caracterização dos perfis de personalidades dos papéis e das pessoas deve ser realizada adotando tipologias consolidadas na literatura da área de Psicologia, tais como: *Myers-Briggs Type Indicator* (MBTI) [6] e *Five Factor Model* (FFM) [7]. Dado que não existe consenso na literatura sobre a melhor tipologia, a abordagem proposta deve suportar ambas de forma customizável, ficando a critério do gerente de projeto a escolha daquela que melhor se adequa à organização ou ao projeto de software.

Na literatura, existem propostas com mapeamentos de profissões ou papéis funcionais da engenharia de software para traços de personalidades mais requeridos ou marcantes para desempenhar as respectivas atividades. Neste contexto, na abordagem proposta, o **perfil de personalidade dos papéis** adotará as classificações de Capretz e Ahmed [8] e Rehman *et al.* [9] para as tipologias MBTI e FFM, respectivamente.

Em relação ao **perfil de personalidade das pessoas**, existem diversos testes de personalidade consolidados na literatura e disponíveis na Internet. Portanto, na abordagem proposta, é possível adotar qualquer teste de personalidade das tipologias MBTI ou FFM, optando pelo mais adequado à organização ou projeto de software. Consequentemente, está fora do escopo da abordagem proposta a concepção de mapeamentos de personalidades para papéis funcionais, bem como a elaboração de testes de personalidade.

Posteriormente, os perfis de personalidade de papéis e pessoas devem ser matematicamente mapeados visando mensurar o **nível de adequação não-técnica** de cada pessoa em relação a cada papel. Neste mapeamento, a abordagem proposta deve favorecer soluções candidatas nas quais as pessoas alocadas possuem ou não os traços de personalidade requeridos ou não pelos papéis, respectivamente.

C. Avaliação Integrada Pessoas-Papéis

Nesta etapa, adotando como entrada o **nível de adequação técnica** e o **nível de adequação não-técnica** das pessoas aos papéis funcionais, juntamente com a restrição do **número de vagas/papel**, a abordagem proposta deve adotar algoritmos evolucionários multiobjetivos para buscar e recomendar soluções não-dominadas que idealmente representam ou estão próximas da fronteira de Pareto.

A representação de cada solução (cromossomo) será codificada na forma de uma cadeia de inteiros (genes), segmentados por papéis funcionais de acordo com o número de vagas por papel, cujos possíveis valores são os identificadores das pessoas candidatas (alelos). Cada vaga deve ser alocada para uma única pessoa, e, como restrições, cada pessoa somente pode ser alocada para uma única vaga e todas as vagas devem ser preenchidas.

Na formulação multiobjetiva, a abordagem proposta deve definir duas funções objetivos: (*i*) maximização da adequação técnica das pessoas aos papéis; e (*ii*) maximização da adequação não-técnica das pessoas aos papéis. Porém, por serem objetivos conflitantes, provavelmente, não deve haver uma única solução não-dominada, vencedora em ambos objetivos. Assim, o gerente de projetos deve selecionar uma solução boa o suficiente entre as soluções recomendadas, por um lado, favorecendo o objetivo de maior interesse, mas por outro lado, penalizando o outro objetivo de menor interesse.

III. CONSIDERAÇÕES FINAIS

Uma vez elaborada a formulação matemática da abordagem proposta, estudos de casos devem ser conduzidos para avaliar e validar a abordagem junto a uma equipe de desenvolvimento com projetos de software reais. Neste sentido, contatos já estão sendo realizados com o gerente de projetos de uma instituição pública de ensino, denominada Instituto Federal da Paraíba (IFPB), onde o acesso já está liberado para observações diárias *in loco*, entrevistas e posterior análise de resultados. Esta organização foi escolhida por ter ambiente favorável e acessível à execução dos estudos de casos e por ter um portfólio de projetos que podem ser beneficiados com a adoção da abordagem proposta.

Em relação aos trabalhos relacionados, é possível encontrar propostas que, de forma isolada, parcial ou pontual, contemplam alguns aspectos técnicos ou não-técnicos relativos às tarefas do projeto e pessoas candidatas. No entanto, em sua maioria, tais propostas não apresentam quaisquer proposições de técnicas, métodos ou metodologias para identificação e coleta de informações que viabilizem a medição e a valoração de parâmetros ou atributos adotados.

Assim sendo, a principal contribuição da abordagem proposta é a adoção da análise integrada de perfis técnicos e de personalidades, relacionados aos papéis funcionais e às pessoas, incluindo estratégias de identificação, coleta e medição dos dados requeridos, fornecendo informações de valor agregado a tomadores de decisões. Nos trabalhos relacionados, tais estratégias não são sinalizadas, impondo ao gerente de projeto a complexidade e o esforço de concebê-las.

REFERÊNCIAS

[1] L.F. Capretz, "Personality types in software engineering", Int. J. of Human-Computer Studies, 58(2):207-214, 2003.

[2] L.C. Silva and A.P.C.S. Costa, "Decision model for allocating human resources in information system projects". Int. J. of Project Manag., 31(1):100-108, 2013.

[3] A.S. Barreto *et al.*, "Apoio à alocação de recursos humanos em projetos de software: uma abordagem baseada em satisfação de restrições", IV Simp. Brasil. de Qual. de Softw., 2005.

[4] F.Q.B. da Silva and A.C.F. César, "An experimental research on the relationships between preferences for technical activities and behavioural profile in software development", XXIII Simp. Brasil. de Eng. de Softw., 2009.

[5] V. Souza and G. Elias, "A fuzzy-based approach for selecting technically qualified distributed software development teams", 16th Mexican Int. Conf. on Artif. Intell., 2018.

[6] I.B. Myers *et al.*, "Manual: guide to the development and use of the Myers-Briggs type indicator", Consulting Psychologists Press, 1998.

[7] J.M. Digman, "Personality structure: emergence of the five-factor model", Annual Review of Psychology, 41:417-440.

[8] L. F. Capretz and F. Ahmed, "Making sense of software development and personality types", IT Professional 12.1, 2010.

[9] M. Rehman *et al.*, "Mapping job requirements of software engineers to big five personality traits", Comput. & Inf. Sci., 2:1115-1122, 2012.