UNIVERSIDADE ESTADUAL DE MARINGÁ
CENTRO DE TECNOLOGIA
DEPARTAMENTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

CARLOS VINÍCIUS BINDEWALD

Otimização interativa de projeto de Arquitetura de Linha de Produto de Software

Maringá
2020

CARLOS VINÍCIUS BINDEWALD

Otimização interativa de projeto de Arquitetura de Linha de Produto de Software

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Departamento de Informática Centro de Tecnologia da Universidade Estadual de Maringá, como requisito parcial para obtenção do título de Mestre em Ciência da Computação.

Orientadora: Profa. Dra. Aline Maria Malachini Miotto Amaral

Maringá
2020

Dados Internacionais de Catalogação-na-Publicação (CIP)
(Biblioteca Central - UEM, Maringá - PR, Brasil)

Bindewald, Carlos Vinícius

B612o Otimização interativa de projeto de Arquitetura de Linha de Produto de Software / Carlos Vinícius Bindewald. -- Maringá, PR, 2020.
140 f.: il. color., figs., tabs.

Orientadora: Profa. Dra. Aline Maria Malachini Miotto Amaral.
Dissertação (Mestrado) - Universidade Estadual de Maringá, Centro de Tecnologia, Departamento de Informática, Programa de Pós-Graduação em Ciência da Computação, 2020.

1. Linha de produto de software. 2. Search-Based Software Engineering. 3. OPLA-Tool. 4. Interação homem-computador. 5. Aprendizagem de máquina. I. Amaral, Aline Maria Malachini Miotto, orient. II. Universidade Estadual de Maringá. Centro de Tecnologia. Departamento de Informática. Programa de Pós-Graduação em Ciência da Computação. III. Título.

CDD 23.ed. 004.22

Marinalva Aparecida Spolon Almeida - 9/1094

FOLHA DE APROVAÇÃO

CARLOS VINICIUS BINDEWALD

Otimização interativa de projeto de arquitetura de linha de produto de software

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Departamento de Informática, Centro de Tecnologia da Universidade Estadual de Maringá, como requisito parcial para obtenção do título de Mestre em Ciência da Computação pela Banca Examinadora composta pelos membros:

BANCA EXAMINADORA

Profa. Dra. Aline Maria Malachini Miotto Amaral
Universidade Estadual de Maringá – DIN/UEM

Profa. Dra. Thelma Elita Colanzi Lopes
Universidade Estadual de Maringá – DIN/UEM

Prof. Ph.D. Jerffeson Teixeira de Souza
Universidade Estadual do Ceará – PPGI/UTFPR-CP

Aprovada em: 17 de fevereiro de 2020.
Local da defesa: Sala 101, Bloco C56, *campus* da Universidade Estadual de Maringá.

AGRADECIMENTOS

Agradeço primeiramente à minha família, meus pais Carlos Bindewald e Vilma Aparecida Delgado Bindewald e aos meus irmãos Helder e Guilherme pelo apoio durante esses mais de dois anos de desenvolvimento deste trabalho. Agradeço também aos meu sobrinhos Pedro e João que, embora muitas vezes me "atrapalharam" com suas brincadeiras enquanto eu desenvolvia o trabalho, fizeram-me refletir que conciliar trabalho e família é uma parte essencial de nossas vidas.

À minha orientadora Profa. Dra. Aline Maria Malachini Miotto Amaral pela orientação e conhecimentos compartilhados, bem como pelo apoio dado durante todo este período.

À Profa. Dra. Thelma Elita Colanzi por ter sido a minha orientadora em um primeiro momento e, posteriormente, pelo apoio contínuo ao desenvolvimento deste trabalho.

Aos amigos do grupo de pesquisa OtimizES-UEM e demais colegas de mestrado por todo apoio, conhecimentos compartilhados e companheirismo durante esses mais de dois anos. Agradecimento especial ao amigo Willian por toda ajuda dada ao desenvolvimento deste trabalho.

À Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) por todo apoio financeiro concedido para desenvolvimento deste trabalho de mestrado.

Otimização interativa de projeto de Arquitetura de Linha de Produto de Software

RESUMO

A busca constante por redução de custos, melhoria de qualidade e aumento na velocidade de desenvolvimento de softwares tem feito a indústria utilizar cada vez mais o conceito de reúso. Uma das formas de reúso que tem se destacado é a Linha de Produto de Software (LPS). Em LPS, todo o processo de desenvolvimento é feito considerando-se o futuro reúso de partes/componentes. Para um projeto de LPS é fundamental a definição de uma arquitetura central que represente um design de alto nível para todos os produtos de uma LPS, a Arquitetura de Linha de Produto de Software (ALPS). O projeto de uma ALPS pode envolver vários fatores, muitas vezes conflitantes, o que torna o mesmo uma tarefa que demanda grande esforço humano. A abordagem MOA4PLA, com o suporte da ferramenta OPLA-Tool, possibilita, a partir de um processo de otimização multiobjetivo a geração de um conjunto de soluções alternativas a uma ALPS dada como entrada, otimizadas a partir de métricas específicas de LPS. Processos de otimização como este são estudados em um campo de pesquisa denominado *Search Based Software Engineering* (SBSE). Pesquisas em SBSE têm demonstrado que a inclusão da opinião do *Decision Maker* (DM), através de processos interativos, durante o processo de otimização, tem possibilitado o surgimento de soluções que melhor se adéquam a cada DM. Contudo, estas pesquisas apontam também para o surgimento do problema da fadiga. O tratamento deste problema pode ser feito a partir da utilização de algoritmos de Aprendizagem de Máquina (AM). Este trabalho teve como objetivos a inserção da opinião do DM (arquiteto de software) durante o processo de otimização e a incorporação de um modelo de AM na MOA4PLA/OPLA-Tool. Para isso, foram executados três experimentos (Exp 01, Exp 02, Exp 03). Exp 01 visou avaliar a viabilidade do processo interativo. Exp 02 testou três diferentes estratégias para a composição do conjunto de dados de treinamento do modelo de AM. Exp 03 avaliou o Modelo Interativo Proposto (MIP) como um todo (processo interativo + modelo de AM). Estes experimentos permitiram concluir que o MIP possibilita a geração de alternativas de ALPS que melhor atendem as expectativas do arquiteto e, além disso, o modelo de AM adotado foi apto em capturar as preferências de cada arquiteto e pôde substituí-lo com eficiência no processo de avaliação de ALPS.

Palavras-chave: Linha de produto de software. Arquitetura de Linha de Produto de Software. *Search-Based Software Engineering*. MOA4PLA. OPLA-Tool. Interação Homem-computador. Aprendizagem de máquina.

Interactive Optimization of Product Line Architecture Design

ABSTRACT

The constant pursuit of cost reduction, quality improvement and increased development speed has made the industry increasingly use the concept of reuse. One of the forms of reuse that has stood out is the Software Product Line (SPL). In SPL, the entire development process is done considering the future reuse of parts/components. For an SPL project it is critical to define a core architecture that represents a high-level design for all SPL products, the Product Line Architecture (PLA). The design of a PLA can, often, involve several conflicting factors, which makes it a task that requires great human effort. The MOA4PLA approach, supported by the OPLA-Tool, enables, from a multi-objective optimization process, the generation of a set of alternative solutions of an initial PLA, optimized from SPL specific metrics. Optimization processes are studied in a research field called Search-Based Software Engineering (SBSE). Researches in SBSE has shown that the inclusion of Decision Maker (DM) opinion through interactive processes, during the optimization process, has enabled the emergence of more adequate solutions to each DM. However, these researches also points to the emergence of the fatigue problem. The treatment of this problem can be done by using Machine Learning (ML) algorithms. This work aimed to insert the DM's opinion (software architect's) during the optimization process and the incorporation of an ML model in the MOA4PLA/OPLA-Tool. For this, three experiments were performed (Exp 01, Exp 02, Exp 03). Exp 01 aimed to evaluate the viability of the interactive process. Exp 02 tested three different strategies for composing the ML training data set. Exp 03 evaluated the Proposed Interactive Model (PIM) as a whole (interactive process + ML model). These experiments led to the conclusion that PIM enables the generation of PLA alternatives that best meet the architect's expectations and, moreover, the ML model adopted was able to capture the preferences of each architect and could effectively replace it in the interactive process.

Keywords: Software Product Line. Product Line Architecture. Search-Based Software Engineering. MOA4PLA. OPLA-Tool. Human-computer interaction. Machine Learning
.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS E ABREVIATURAS

AGM: *Arcade Game Maker*
ALPS: Arquitetura de Linha de Produto de Software
AM: Aprendizagem de Máquina
DM: *Decision Maker*
ES: Engenharia de Software
HCI: *Human–computer interaction*
IGA: *Interactive Genetic Algorithm*
IHC: Interação Homem-Computador
PLA: *Product Line Architecture*
LPS: Linha de Produto de Software
MA: Modelo Atual
MAE: *Mean Absolute error*
MIP: Modelo Interativo Proposto
ML: *Machine Learning*
MLP: *Multilayer perceptron*
MOA4PLA: *Multi-Objective Optimization Approach for PLA Design*
MS: Mapeamento Sistemático
OPLA-Tool:*Optimization for PLA Tool*
RAE: *Root Absolute Error*
RNA: Redes Neurais Artificiais
RMSE: *Root Mean Square Error*
RRSE: *Root Relative Squared Error*
SBSE: *Search-based Software Engineering*
SPL: *Software Product Lines*
UML: *Unified Modeling Language*

SUMÁRIO

1

Introdução

Com a demanda cada vez mais crescente por produtos e/ou serviços ligados à área tecnológica, faz-se necessário também que o desenvolvimento de softwares ligados a esses produtos/serviços seja feito de maneira eficiente, uma vez que o atraso no desenvolvimento, entre outras coisas, pode levar determinados projetos ao fracasso.

Neste sentido, para acelerar o desenvolvimento de softwares, a indústria tem buscado diminuir o tempo de desenvolvimento utilizando-se do conceito de reúso. Frakes e Kang (2005) definem que "reutilização de software é o uso de software existente ou conhecimento referente a este para construir um novo software". Prado e Lucrédio (2001) apontam para a reutilização como um princípio essencial para a área de engenharia de software, de modo a garantir a redução de esforços e custos no processo de desenvolvimento de software, visto que muitos sistemas apresentam componentes similares ou mesmo idênticos. Dentro desse contexto, a indústria tem adotado a abordagem de Linha de Produto de Software (LPS) como uma forma de sistematizar o reúso de software. De acordo com Van Der Linden (2007), em LPS, o reúso de software se distingue de outras formas de reúso pois, nesse caso, o projeto do software já é feito considerando o futuro reúso de seus componentes. Sendo assim, o desenvolvimento é feito para a reutilização de software.

Para um projeto de LPS, um dos artefatos mais importantes é a ALPS (Arquitetura de Linha de Produto de Software), que define um *design* comum para todos os produtos derivados da LPS, incluindo características (*features*) obrigatórias e variáveis (KANG, et al., 1990). Contudo, como exposto por Colanzi (2014), obter uma ALPS modular, extensível e reusável pode ser uma tarefa não-trivial, que envolve vários fatores (objetivos), muitas vezes conflitantes entre si, demandando-se assim também um grande esforço humano.

Problemas como estes, onde há a necessidade de se encontrar um melhor balanceamento (custo-benefício) entre os diversos fatores que envolvem os mesmos podem ser resolvidos por meio de técnicas de otimização matemática. Estas técnicas possibilitam encontrar diferentes soluções para um mesmo problema e que possuem o melhor custo-benefício entre os diversos fatores considerados. Para estes casos, onde há muitos fatores a serem considerados, os problemas são descritos na literatura como problemas de otimização multi-fatores (multiobjetivos).

De acordo com Ngatchou et al. (2005), o objetivo de uma otimização multiobjetivos é encontrar um conjunto de soluções aceitáveis e apresentá-las aos DM[1] (*Decision Makers*) para que os mesmos possam escolher a(s) solução(ões) que melhor representa(m) os seus objetivos. Soluções para este tipo de problema podem ser encontrados por meio da utilização de algoritmos chamados de "algoritmos de busca", em um campo de pesquisa denominado SBSE (*Search Based Software Engineering*).

Segundo Harman et al. (2012), SBSE é um campo de pesquisa e prática no qual a busca computacional (bem como as técnicas de otimização mais usualmente associadas à Pesquisa Operacional) são usadas para resolver problemas complexos em Engenharia de Software. Vergilio et al. (2011) acrescentam que a utilização de técnicas de SBSE contribui para reduzir custos e esforços associados ao desenvolvimento de software, visto que as soluções encontradas satisfazem restrições que estão normalmente em conflito e, em geral, difíceis de serem obtidas por engenheiros de software.

Dado que o projeto de uma ALPS é um tarefa que envolve vários fatores (objetivos), muitas vezes conflitantes, onde há uma necessidade de se encontrar um melhor *trade-off* (custo benefício) entre os mesmos, faz sentido tratar este problema como um problema de otimização multiobjetivos. Neste contexto, a abordagem MOA4PLA (*Multi-Objective Optimization Approach for PLA Design*) Colanzi et al. (2014), foi proposta e permite dar um tratamento multiobjetivo para o problema do projeto de uma ALPS, baseado em métricas específicas de ALPS.

A abordagem MOA4PLA já foi utilizada em outros estudos (COLANZI e VERGILIO, 2016; CHOMA NETO, et al., 2018) onde foi possível demonstrar sua aplicabilidade. Importante ressaltar que a automação da abordagem MOA4PLA é suportada pela ferramenta OPLA-Tool (FÉDERLE et al., 2015), que disponibiliza aos arquitetos de software as atividades definidas pela referida abordagem.

A abordagem MOA4PLA, bem como a ferramenta OPLA-Tool, demonstram grande aplicabilidade quando os fatores a serem otimizados podem ser modelados matema-

[1]Neste trabalho os DM (*Decision Makers*) são os usuários (arquitetos de software) que interagem com a ferramenta OPLA-Tool, descrita posteriormente.

ticamente. Contudo, muitas vezes, é necessário considerar aspectos mais subjetivos em um processo de otimização, como as preferências individuais de cada DM. Neste sentido, Ferreira et al. (2017) e Simons, Singer e White (2015) indicam que, em alguns casos, as soluções obtidas em SBSE, muitas vezes, são rejeitadas pelos DM pois as mesmas não satisfazem determinados aspectos (particularidades) de cada DM. Buscando por uma forma de encontrar soluções mais adequadas a cada DM ao final de um processo de otimização, alguns trabalhos (DANTAS, et al., 2015; FERREIRA, et al., 2017; RAMIREZ, et al., 2018; SU e ZHANG, 2010) apontam para a incorporação das preferências individuais de cada DM durante o processo de otimização. Para isso, processos interativos são propostos, de modo que o DM possa intervir durante o processo de otimização, com a introdução de suas preferências durante o mesmo (fatores subjetivos) e, assim, "guiar" o algoritmo a encontrar soluções mais satisfatórias a cada DM.

No entanto, em trabalhos de SBSE que utilizam a interação humana durante o processo de otimização, há grande preocupação por parte dos pesquisadores (SHACKELFORD, 2007; RAMIREZ et al., 2018) com relação à quantidade de interações necessárias, pois, em geral, processos repetitivos podem provocar cansaço no DM, a chamada fadiga.

Neste sentido, alguns trabalhos (FERREIRA et. al, 2017; RAMIREZ et al., 2018, ARAÚJO e PAIXÃO,2014) apontam para a utilização de algoritmos de Aprendizagem de Máquina (AM) para o tratamento da fadiga. Com a utilização de algoritmos de AM é possível que o sistema "aprenda" sobre o comportamento do DM e, posteriormente, possa vir a substituí-lo no processo interativo, tornando este processo o menos cansativo quanto possível.

Atualmente, na abordagem MOA4PLA, e por consequência na ferramenta OPLA-Tool, a interação com o arquiteto ocorre apenas *a priori* e *a posteriori*, ou seja, antes e depois do processo de otimização. Desta forma, durante o processo de otimização, a opinião do arquiteto ainda não é considerada. Considerando a vertente de trabalhos em SBSE que buscam incorporar a opinião humana durante o processo de otimização, este trabalho visa, como primeiro objetivo, introduzir este conceito na abordagem MOA4PLA/ferramenta OPLA-Tool.

Além disso, haja visto o problema da fadiga, relatado em outros trabalhos que consideram a interação humana em SBSE, este trabalho tem, como um segundo objetivo, a introdução de um modelo de AM que seja capaz de aprender sobre o comportamento do arquiteto e possa, posteriormente, assumir o papel do mesmo durante o processo interativo.

A Seção 1.1 destaca as questões de pesquisa que este trabalho visa responder e na Seção 1.2 os objetivos que o mesmo deseja cumprir.

1.1 Motivação e justificativa

Como já destacado anteriormente, na abordagem MOA4PLA/ferramenta OPLA-Tool a interação com o arquiteto é feita, atualmente, apenas *a priori* e *a posteriori*. Desta forma, todo o processo de otimização é executado e, ao final do processamento, os resultados são apresentados ao arquiteto. Neste sentido, durante o processo de otimização ainda não é possível que o arquiteto participe do mesmo, incorporando aspectos mais subjetivos (preferências individuais).

Nesse sentido, objetiva-se com este trabalho responder aos seguintes questionamentos:

- **Q1:**Como adaptar a abordagem MOA4PLA e ferramenta-OPLA-Tool para permitir a interação homem-computador durante o processo de otimização?
- **Q2:**Levando-se em consideração que um excessivo número de interações no processo de otimização pode levar à fadiga do arquiteto, como a Aprendizagem de Máquina pode ser utilizada para que, em algum momento, venha a substituir o arquiteto no processo interativo?

1.2 Objetivos

1.2.1 Objetivo Geral

Incorporar as preferências do arquiteto de software em processo de otimização de Arquitetura de Linha de Produto de Software.

1.2.2 Objetivos específicos

- Adaptar a abordagem MOA4PLA e, por conseguinte, a ferramenta OPLA-Tool, para permitir a interação com o arquiteto durante o processo de otimização;
- Desenvolver um modelo de AM que possa ser integrado à abordagem MOA4PLA e à ferramenta OPLA-Tool para lidar com o problema da fadiga;
- Avaliar experimentalmente a incorporação das preferências do arquiteto na abordagem MOA4PLA e na ferramenta OPLA-Tool.

1.3 Organização do trabalho

Este trabalho está organizado da seguinte maneira: No Capítulo 2, é apresentada uma fundamentação teórica sobre os temas que permeiam o desenvolvimento deste trabalho (Linha de Produto de Software; Arquitetura de Linha de Produto de Software; *Search-Based Software Engineering*; abordagem MOA4PLA; ferramenta OPLA-Tool; Interação Humana em SBSE; Fadiga Humana em Processos Interativos de SBSE; Aprendizagem de Máquina).

Na sequência, Capítulo 3, são apresentados os resultados do mapeamento sistemático com os trabalhos encontrados que embasaram o desenvolvimento deste trabalho. No Capítulo 4, são descritos os métodos e materiais utilizados para a elaboração da proposta deste trabalho, bem como para a sua validação. No Capítulo 5 são detalhados como foi introduzida a interação homem-computador no processo de otimização da MOA4PLA/OPLA-Tool e também como foi desenvolvido o modelo de AM. Posteriormente, no Capítulo 6, são descritos e discutidos os resultados dos experimentos realizados para a validação do modelo proposto. Por fim, no Capítulo 7, são apresentadas as conclusões deste trabalho e também os possíveis cenários que podem ser explorados em trabalhos futuros.

2

Revisão bibliográfica

2.1 Considerações Iniciais

Este capítulo apresenta os conceitos teóricos que embasam o desenvolvimento deste trabalho.

2.2 Linha de Produto de Software

Linha de Produto de Software (LPS) surgiu como uma forma de utilizar na Engenharia de Software (ES) técnicas de produção que já eram utilizadas com sucesso em outras áreas da engenharia. No intuito de se aumentar a eficiência produtiva, em geral, uma mesma linha de produção é utilizada para a formação de diferentes produtos, que se distinguem na maneira como os mesmos são montados e configurados. Para isso, além de um meio de produção em comum, estes produtos compartilham também muitas partes/componentes. Partindo-se desta ideia, a abordagem de LPS centra-se no uso de técnicas de engenharia que permitem criar um grupo de sistemas de software similares a partir de um conjunto de especificações de software comuns a todos esses sistemas, usando para tal um meio comum de produção. Para conceituar LPS e fazer um paralelo com a indústria tradicional, Apel et al.(2016) expõem que, na indústria tradicional, os fabricantes planejam e projetam linhas de produtos para cobrir todo um espectro de possíveis produtos e variações, servindo, portanto, às necessidades individuais e desejos de muitos consumidores. Em LPS, com base em um conjunto de partes (componentes) reutilizáveis, um fabricante de software pode gerar um produto de software com base nos requisitos de seu cliente. Desta maneira,

é possível conciliar produção em massa e padronização com customização na engenharia de software.

Para que seja possível alcançar alta produtividade no desenvolvimento de software, o foco deve estar no reuso de software. Em LPS, há uma distinção fundamental de como o modelo de reutilização de software é pensado. De acordo com Van Der Linden (2007), em LPS, o desenvolvimento é feito para a reutilização de software. Desta forma, foca-se no desenvolvimento de ativos relevantes que abrangem todo o ciclo de vida de desenvolvimento de software, desde o estágio de requisitos, até a arquitetura, implementação e teste. Esse conjunto de ativos define a infraestrutura da linha de produtos. Ainda segundo Van Der Linden (2007), uma distinção importante entre a engenharia de LPS e outras abordagens de reutilização é que os vários ativos em si contêm variabilidade explícita. Por exemplo, uma representação dos requisitos pode conter uma descrição explícita de requisitos específicos que se aplicam apenas a um determinado subconjunto dos produtos. De uma forma geral, a engenharia de LPS segue alguns princípios fundamentais:

- **Gerenciamento de variabilidade:** sistemas individuais são considerados variações de um tema comum. Essa variabilidade é explicitada e deve ser gerenciada sistematicamente.
- **Centrado nos negócios:** a engenharia da linha de produtos de software visa conectar completamente a engenharia da linha de produtos à estratégia de longo prazo dos negócios.
- **Centrado na arquitetura:** o lado técnico do software deve ser desenvolvido de uma maneira que permita tirar vantagem das semelhanças entre os sistemas individuais.
- **Abordagem de dois ciclos de vida:** os sistemas individuais são desenvolvidos com base em uma plataforma de software. Esses produtos - assim como a plataforma - devem ser projetados e ter seus ciclos de vida individuais.

Para a definição de uma LPS, deve-se pensar antes em seu projeto. Neste ponto, a arquitetura de Linha de Produto de Software (ALPS) desempenha papel fundamental.

2.2.1 Arquitetura de Linha de Produto de Software

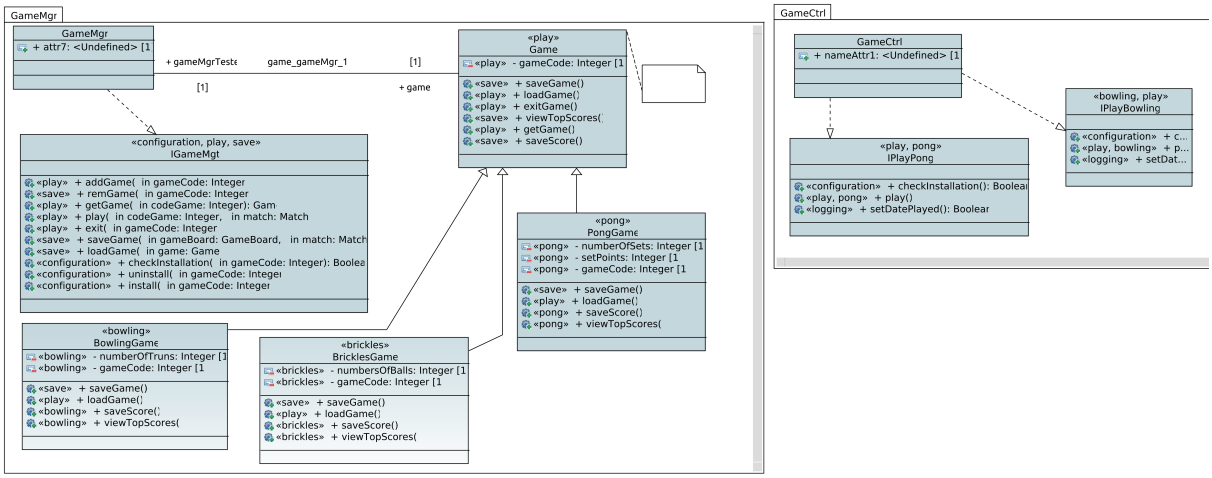
Segundo Schmid (2002) um elemento chave para uma bem-sucedida engenharia de LPS é identificar desde o início uma arquitetura de referência que possa fornecer um plano

para produzir diferentes variantes. A semelhança estrutural entre as variantes, resultante da arquitetura comum, permite que os desenvolvedores reutilizem componentes em uma variedade de produtos diferentes na linha de produtos.

Neste sentido, a definição da Arquitetura de Linha de Produto de Software (ALPS) possui importância fundamental no projeto de LPS. De acordo com Cardoso et al. (2017), a "ALPS é a arquitetura central que representa um *design* de alto nível para todos os produtos de uma LPS, incluindo pontos de variação e variantes".

A Figura - 2.1 representa um exemplo de uma ALPS, sendo um excerto da ALPS AGM.

Figura 2.1: Excerto da ALPS AGM



Fonte: Autoria própria (adaptado de SEI, 2019) .

Cardoso et al. (2017) destacam ainda que a ALPS é um dos mais valiosos ativos de uma LPS, pois contém os seus componentes centrais, bem como os componentes variáveis, em uma estrutura que abrange o comportamento do qual os produtos de software são desenvolvidos.

No entanto, como descrito por Colanzi (2014), um projeto de ALPS é uma tarefa não-trivial e que envolve diversos fatores, muitas vezes ainda conflitantes, o que acarreta em grande esforço humano.

Problemas como este, onde há uma necessidade de se lidar com vários fatores (também chamados objetivos) conjuntamente e que, muitas vezes, podem ainda ser conflitantes, são tratados em um campo de pesquisa denominado Engenharia de Software Baseada em Busca (*Search-Based Software Engeneering*).

2.3 *Search-Based Software Engineering*

Search Based Software Engineering (SBSE), termo cunhado por Harman e Jones (2001), representa um campo de pesquisa no qual métodos de otimização matemática são aplicados à engenharia de software.

Em SBSE, o termo busca (*search*) refere-se à busca por soluções ótimas ou quase ótimas em um espaço de soluções candidatas, guiada por uma função de *fitness*[1] que distingue entre soluções melhores e piores.

De acordo com Harman e Clark (2004) há dois ingredientes principais para a aplicação de técnicas de otimização matemática em Engenharia de Software (ES), sendo estes: a escolha da representação do problema e a definição da função de avaliação. Neste sentido, os mesmos autores indicam que a adoção de técnicas de otimização matemática em ES torna-se uma opção bastante adequada, visto que, normalmente, um engenheiro de software terá uma representação adequada para o problema, pois a solução do problema exige do engenheiro de software uma maneira de representar o problema em questão. Além disso, muitos problemas em ES possuem um conjunto rico e variado de métricas, como por exemplo as métricas relacionadas à ALPS, chamadas métricas arquiteturais. Essas métricas são utilizadas para avaliar alguns princípios de projeto, tais como: baixo acoplamento e alta coesão (CHIDAMBER e KEMERER, 1994), e podem ser bons candidatos iniciais para funções de avaliação (HARMAN e CLARK, 2004).

Problemas em ES são geralmente complexos e difíceis de resolver pois podem envolver vários fatores (objetivos) e, muitas vezes, os mesmos podem ainda ser conflitantes, ou seja, a otimização de um determinado fator pode afetar outro. Nestes casos, o problema de otimização torna-se um problema de otimização multi-fatores, também chamado otimização multiobjetivos.

2.3.1 Otimização multiobjetivos

Problemas de otimização multiobjetivos são problemas impactados por vários fatores, cada um representado por uma função objetivo. Assim, faz-se necessária uma otimização simultânea desses objetivos, que podem ser dependentes, independentes, cooperativos ou concorrentes (COELLO et al., 2007). De acordo com Coello et al. (2007) quando há mais de uma função objetivo, o problema não tem apenas uma solução ótima, mas sim um conjunto diversificado de soluções ótimas. As várias soluções ótimas representam

[1]Neste trabalho, função de *fitness*, função de avaliação e função objetivo dizem respeito ao mesmo conceito

o melhor *trade-off* (custo-benefício) entre os objetivos definidos e compõem o conjunto das soluções não-dominadas, formando a Fronteira de Pareto[2] (PARETO et al., 1927). Define-se que uma solução é não-dominada quando, dado um conjunto de soluções, uma solução A domina uma solução B se o valor de pelo menos um objetivo de A é melhor que o respectivo valor objetivo de B e os demais valores objetivos de A não são piores que os respectivos valores de B. Assim, A é uma solução não-dominada se não for dominada por nenhuma outra solução.

Dentre os problemas de ES que podem ser tratados por meio de um processo de otimização multiobjetivos, encontra-se o problema do projeto de ALPS. Como descrito por Colanzi (2014), em um projeto de ALPS há diversos fatores envolvidos, o que demanda grande esforço humano. Neste sentido, a abordagem MOA4PLA trata o projeto de ALPS como um problema de otimização com vários fatores (objetivos) para os quais pode não existir uma única solução possível, mas sim um conjunto de soluções ótimas que apresentam o melhor *trade-off* entre os objetivos envolvidos.

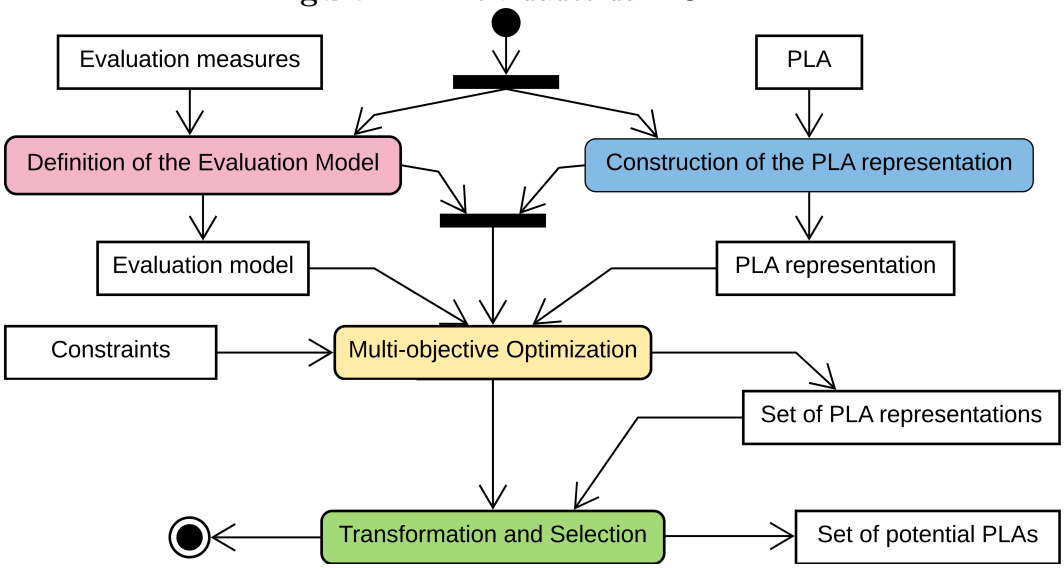
2.4 MOA4PLA

De acordo com Colanzi (2014), MOA4PLA é uma abordagem sistemática e automatizada que utiliza algoritmos de busca multiobjetivos para avaliar e melhorar o projeto de ALPS em termos de princípios básicos de projeto, modularização de características e extensibilidade de LPS.

A Figura - 2.2 apresenta as atividades realizadas pela abordagem MOA4PLA, descritas em maiores detalhes na sequência.

[2]Conceito que define um estado de alocação de recursos em que é impossível realocá-los tal que a situação de qualquer participante seja melhorada sem piorar a situação individual de outro participante. Possui aplicações em variadas campos, tais como: economia, engenharia, informática, e ciências sociais.

Figura 2.2: Atividades da MOA4PLA



Fonte: (COLANZI, 2014) .

- ***Construction of the PLA Representation*** - Esta atividade recebe como entrada o projeto de ALPS (em diagrama de classes) e instancia a solução inicial a ser utilizada pelo algoritmo de busca como ponto de partida para o processo evolutivo. Como saída desta atividade, tem-se a conversão do diagrama de classes inicial em um metamodelo passível de sofrer o processo de otimização;

- ***Definition of the Evaluation Model*** - O objetivo desta atividade é a definição pelo arquiteto de quais medidas (métricas) devem ser utilizadas no modelo de avaliação (*Evaluation Model*) que será utilizado no processo de otimização. Desta forma, o arquiteto pode escolher métricas que melhor atendam as suas necessidades. Estas métricas constituem os objetivos a serem otimizados. (VERDECIA et al., 2017)

- ***Multi-Objective Optimization*** - Nesta atividade o projeto de ALPS original é otimizado durante o processo de otimização multiobjetivo, respeitando as restrições estabelecidas e retornando como saída um conjunto de soluções alternativas em relação à ALPS original;

- ***Transformation and Selection*** - A saída da atividade anterior é dada como entrada para esta atividade. Neste momento o conjunto de soluções é então convertido para uma forma legível para o arquiteto, em um diagrama de classes.

Ainda de acordo com Colanzi (2014), o principal foco da MOA4PLA é a avaliação e a melhoria do projeto de ALPS, considerando múltiplos objetivos (métricas), independentemente do algoritmo de busca adotado. Sendo assim, trata-se de uma abordagem genérica no que diz respeito ao algoritmo multiobjetivo a ser empregado. A MOA4PLA é independente tanto do método adotado para mapear as características nos elementos arquiteturais, como do método usado para representar as variabilidades na ALPS. O foco desta abordagem é a análise estática automatizada de ALPS em nível estrutural representada por diagramas de classes UML estereotipados.

Como seus principais aspectos, a MOA4PLA:

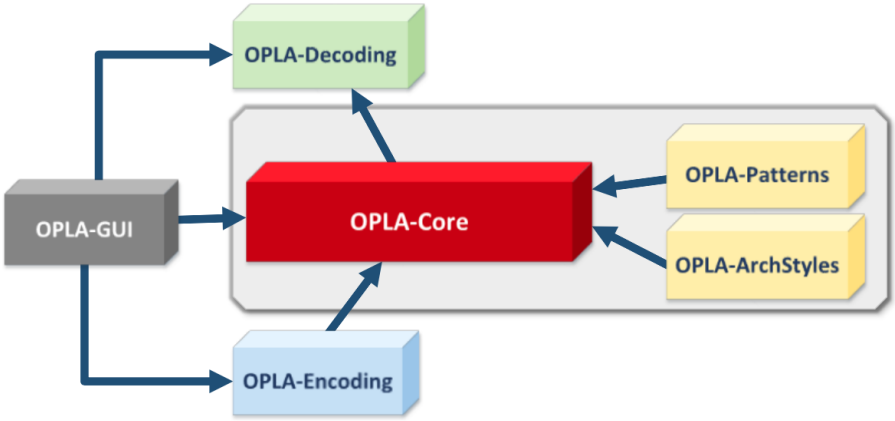
- Engloba um processo para conduzir a otimização do projeto de ALPS por meio de algoritmos multiobjetivos;
- Inclui um metamodelo que representa a ALPS, além de viabilizar a manipulação da mesma por parte dos algoritmos;
- Trata a modularização de características por meio de um novo operador de busca (COLANZI et al., 2014);
- Propõe um tratamento multiobjetivo para o problema de projeto de ALPS, incluindo métricas dirigidas a características (VERDECIA et al., 2017).

Para automatizar a aplicação da abordagem MOA4PLA, foi desenvolvida a ferramenta OPLA-Tool. Na seção a seguir, são apresentados mais detalhes sobre esta ferramenta.

2.5 OPLA-Tool

A ferramenta OPLA-Tool (FÉDERLE et al., 2015) foi projetada para permitir a aplicação da MOA4PLA. A Figura - 2.3 demonstra os módulos que compõem a ferramenta OPLA-Tool, sendo descritos na sequência.

Figura 2.3: OPLA-Tool em seu estágio de desenvolvimento atual



Fonte: (FÉDERLE et al., 2015).

- **OPLA-GUI:** Módulo de interface para interação com o arquiteto, neste módulo são identificados a ALPS que será otimizada (arquivo .UML contendo os diagramas de classes gerados pela ferramenta Papyrus[3]), qual algoritmo multiobjetivo será utilizado, bem como, as funções de *fitness* e operadores de busca disponíveis. Esse módulo se comunica com três outros módulos (OPLA-Encoding, OPLA-Core e OPLA-Decoding);

- **OPLA-Encoding:** Este módulo reconhece a ALPS (modelada na ferramenta Papyrus). O módulo converte o projeto de ALPS em um metamodelo que pode ser utilizado pelo OPLA-Core;

- **OPLA-Core:** Principal módulo da ferramenta. Possui como entrada o metamodelo gerado pelo módulo OPLA-Encoding e na sequência executa o processo evolutivo e de busca por meio de algoritmos de busca multiobjetivos. Os algoritmos cujas implementações estão disponíveis na ferramenta são: NSGAII[4] (DEB et al., 2002) e PAES[5] (KNOWLES e CORNE, 2000). As funções de *fitness* e operadores utilizados já devem ter sido escolhidos no módulo OPLA-GUI. Como saída da otimização, um conjunto de soluções é produzido, ainda representado segundo o metamodelo;

- **OPLA-Decoding:** Este módulo recebe como entrada a saída do módulo OPLA-Core, decodifica as soluções e as converte em uma representação gráfica legível ao arquiteto, modeladas por meio de diagrama de classes;

[3] *https://www.eclipse.org/papyrus/*
[4] Non-dominated Sorting Genetic Algorithm-II
[5] Pareto Archived Evolution Strategy

- **OPLA-Patterns:** Este módulo dispõe de um operador de aplicação de padrões de projeto denominado *design Pattern Mutation Operator*. Este operador é responsável por analisar se uma determinada região da solução apresenta um escopo propício para aplicação de um padrão de projeto e, em caso positivo, realiza a aplicação;
- **OPLA-ArchStyles:** Módulo com operadores de mutação para preservar os estilos arquiteturais aplicados ao projeto de ALPS.

A elaboração dos diversos módulos da OPLA-Tool possibilitou a realização de estudos empíricos que tiveram o intuito de avaliar a efetividade da abordagem MOA4PLA, bem como otimizar o projeto de ALPS (CHOMA et al., 2018; VERDECIA et al., 2017).

Dentro do contexto de SBSE, há uma crescente demanda por incorporar algo que, por envolver aspectos mais subjetivos, é mais difícil de se modelar matematicamente, como as preferências de cada DM. Ramírez et al. (2017) destacam que, em muitos problemas de ES, a avaliação e preferência humanas são cruciais, e, por isso, o número de pesquisas em SBSE que incorporam o DM durante o processo de otimização é cada vez maior.

Atualmente, a abordagem MOA4PLA e a ferramenta OPLA-Tool permitem ao arquiteto interagir *a priori* (antes de inicializar o processo de otimização) e *a posteriori* (após finalizar o processo). No processo a priori, o arquiteto define as configurações usadas no processo evolutivo. O processo a posteriori apresenta as soluções otimizadas para o arquiteto. Dessa maneira, ainda não é possível a interação com o arquiteto durante o processo evolutivo. No intuito de se explorar como pode ser introduzido a interação com o arquiteto durante o processo de otimização realizado pela abordagem MOA4PLA e ferramenta OPLA-Tool, alguns trabalhos foram selecionados, sendo os mesmos descritos a seguir.

2.6 Interação humana em SBSE

Embora muitas vezes os processos de otimização realizados em SBSE tragam bons resultados, há casos também que as soluções podem não agradar ao DM.

Ferreira et al. (2017) e Simons, Singer e White (2015) indicam que as soluções obtidas em SBSE são, em alguns casos, rejeitadas pelos DMs porque muitos aspectos do problema não podem ser matematicamente modelados. Desse modo, percebe-se a necessidade da incorporação em uma otimização multiobjetivos outros aspectos que são inerentes a cada DM, como suas preferências individuais. Neste sentido, há uma vertente de pesquisadores que incluem o DM durante o processo de otimização, como meio de guiar o algoritmo a encontrar soluções mais adequadas a cada um ao final do processo evolutivo.

No trabalho Su e Zhang (2010), por meio de um processo interativo com o DM, a incorporação da opinião do mesmo ocorre em toda nova geração de resultados obtidos por meio de cruzamento e mutação. Os resultados obtidos, quando consistentes com o esperado, são usados como dados de entrada para uma nova iteração.

No mesmo sentido, Ferreira et al. (2017) descrevem que em vários estudos as preferências do DM são também fornecidas interativamente e, em muitos casos, são incorporadas nas funções de *fitness*. Além disso, um aspecto importante é mencionado em relação aos algoritmos utilizados, relatando-se o uso de algumas variações de algoritmos genéticos, denominados genericamente de IGA (*Interactive Genetic Algorithm*) (QUIROZ, et al., 2007).

Mais especificamente em relação aos algoritmos utilizados, Bechikh et al. (2015) descrevem que, para introduzir a interação humana, algumas pesquisas propuseram a modificação de algoritmos evolutivos conhecidos, como o NSGA-II.

Em Ramirez et al. (2018), entre outros aspectos, discute-se sobre a quantidade de interações (intervenções) do DM necessárias durante o processo de otimização. As pesquisas apontam que, em geral, os modelos propostos definem um número fixo de interações. No entanto, em alguns trabalhos, a quantidade de interações a serem feitas é definida pelo DM.

Ainda em relação ao número de interações, vários trabalhos (FERREIRA, et al.,2017; RAMIREZ, et al.,2018; SHACKELFORD, 2017; SHACKELFORD e SIMONS, 2014; BINDEWALD, et al., 2019) apontam para um aspecto importante que deve ser levado em consideração quando o número de interações pode ser elevado, o problema da fadiga. Esse problema é relatado como o problema que mais pode afetar os resultados ao se incorporar a interação humana em SBSE.

2.7 Fadiga humana em processos interativos de SBSE

Ao introduzir interação em SBSE, uma grande preocupação dos pesquisadores é o problema da fadiga (SHACKELFORD, 2007; RAMIREZ et al., 2018). Esse problema pode ocorrer devido, geralmente, a tarefas repetitivas.

Shackelford (2007) lista três pontos principais que podem causar fadiga e afetar os resultados dos processos interativos:

- **Demorado** - O cálculo dos valores de *fitness* para muitos indivíduos e em muitas gerações pode ser custoso em termos de tempo de processamento;

- **Entediante** - Tarefas repetitivas podem levar o DM a perder o interesse rapidamente;
- **Não confiável** - A concentração do DM com o passar do tempo diminui, permitindo a entrada de dados conflitantes e/ou contraditórios.

Para evitar o problema de fadiga, Takagi (1998) sugere algumas técnicas que podem ser utilizadas, tais como:

- Exibir soluções em uma ordem pré-classificada;
- Reduzir o número de soluções;
- Reduzir o número de interações.

Considerando a incorporação de conhecimento do DM no processo de otimização, alguns autores (FERREIRA et. al, 2017; RAMIREZ et al., 2018, ARAÚJO e PAIXÃO, 2014) sugerem o uso de algoritmos de Aprendizagem de Máquina (AM) para evitar o problema de fadiga. Estes algoritmos podem ser capazes de aprender sobre o comportamento do DM e, em algum momento, vir a substituí-lo no processo de interação.

A seguir, há uma breve explicação sobre AM e também sobre os algoritmos de AM utilizados no modelo proposto neste trabalho.

2.8 Aprendizagem de Máquina

De acordo com Mjolsness et al. (2001), Aprendizagem de Máquina (AM) é o estudo de algoritmos capazes de aprender a melhorar o desempenho de uma tarefa com base em sua própria experiência anterior. A partir de dados de amostra, conhecidos como "dados de treinamento", os algoritmos de AM são capazes de construir modelos matemáticos que podem fazer previsões ou decisões sem serem explicitamente programados para esses fins (BISHOP, 2006). Os algoritmos de AM podem ser divididos em dois grupos principais: aprendizagem supervisionada e aprendizagem não-supervisionada.

- **Aprendizagem não-supervisionada:** Nesse caso, o conjunto de dados de treinamento tem apenas entradas. O foco destes algoritmos é encontrar padrões desconhecidos nos dados, agrupando-os com base em atributos compartilhados (Hinton et al., 1999).
- **Aprendizagem supervisionada:** Nesse tipo de aprendizagem, o conjunto de dados de treinamento possui entradas e resultados esperados (saídas) (RUSSELL e

NORVIG, 2016). Assim, a partir de conjunto de dados de treinamento, os algoritmos podem aprender com estes e prever as saídas.

Neste trabalho, dois algoritmos de AM são utilizados: *K-means* e *MultiLayer Perceptron* (MLP).

A utilização do algoritmo K-means tem como objetivo agrupar soluções semelhantes em clusters, a fim de reduzir as soluções a serem avaliadas pelo arquiteto. Em trabalho anterior (FREIRE, et al., 2019), esta abordagem de agrupamento de soluções demonstrou-se ser adequada para o projeto de ALPS. MLP, por sua vez, é utilizado para a construção de um modelo preditivo de AM, que visa aprender as preferências do arquiteto e, posteriormente, substituí-lo na tarefa de avaliar arquiteturas (ALPS).

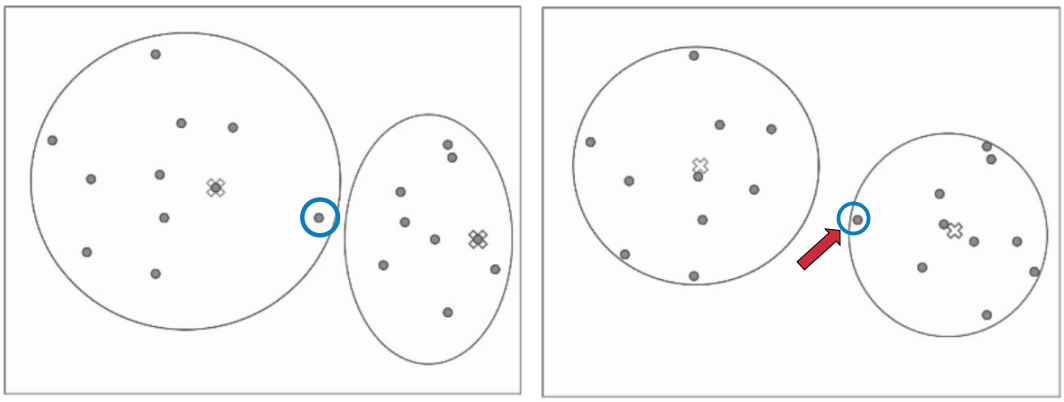
A seguir, uma explicação mais detalhada de cada um dos algoritmos de AM utilizados neste trabalho é apresentada.

K-means clustering

De acordo com Sammut e Webb (2011) *K-means clustering* é um dos métodos de clusterização mais populares e seu funcionamento ocorre da seguinte maneira: Dado um cluster inicial, mas não ideal, realoque cada ponto para o novo centro mais próximo, atualize os centros de cluster calculando a média dos pontos membros e repita o processo de realocação e atualização até que algum critério de convergência (como um número predefinido de iterações) seja atingido.

A Figura - 2.4 mostra um exemplo de agrupamento pelo k-means em um conjunto de pontos, com K = 2 clusters. Os clusters são inicializados selecionando-se aleatoriamente dois pontos como centros, demarcados com "`x`" (Figura - 2.4 (a)). Calcula-se a distância de cada ponto até o centro do cluster, atribuindo-se este ponto ao centro mais próximo. Na sequência, calcula-se o novo centro de cada cluster (Figura - 2.4 (b)). Com os novos centros, calcula-se novamente a distância de cada ponto ao centro do cluster. Caso algum ponto esteja mais próximo do centro do outro cluster, este ponto é então atribuído para o outro cluster. Esta realocação de um ponto é demonstrado na Figura - 2.4 pelo ponto destacado com círculo azul.

Figura 2.4: K-Means clustering, com K = 2 clusters



(a) Inicialização. (b) Reatribuição dos valores.

Fonte: Autoria própria (Adaptado de HAYKIN et al., 2009).

Multilayer Perceptron (MLP)

Segundo Haykin et al. (2009), MLP (Perceptron de múltiplas camadas) surgiu para superar limitações existentes do *Perceptron single layer* (Perceptron de camada única) que se limitava à classificação de padrões linearmente separáveis.

As três características básicas de uma rede MLP são:

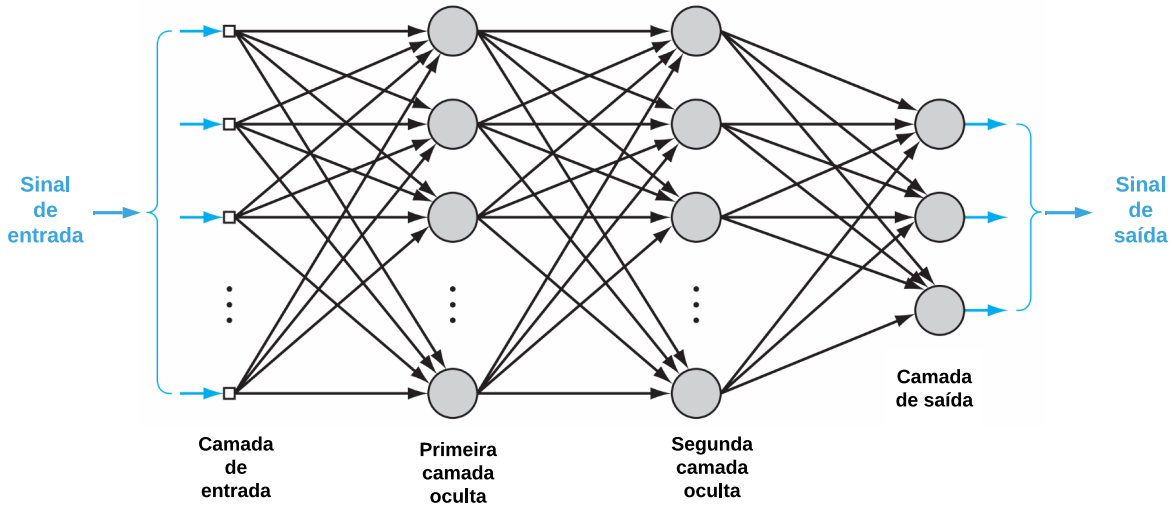
- O modelo de cada neurônio na rede inclui uma função de ativação não linear que é diferenciável;
- A rede contém uma ou mais camadas ocultas dos nós de entrada e saída;
- A rede exibe um alto grau de conectividade, cuja extensão é determinada pelos pesos sinápticos da rede.

Em uma rede MLP os neurônios de saída constituem a camada de saída da rede (*output layer*). Os neurônios restantes constituem camadas ocultas da rede (*hidden layers*). Estes neurônios possuem como função atuarem como “detectores de características”. Desta maneira, à medida que o processo de aprendizado progride, os neurônios ocultos começam a gradualmente "descobrir"as características mais importantes que caracterizam os dados de treinamento. Eles fazem isso executando uma transformação não linear nos dados de entrada em um novo espaço chamado “espaço de característica”. Nesse novo espaço, as classes de interesse em uma tarefa de classificação de padrões, por exemplo, podem ser mais facilmente separadas uma de outra do que seriam no espaço de dados de entrada original.

As camadas ocultas possuem esta denominação pois não fazem parte da saída ou entrada da rede. A primeira camada oculta é alimentada a partir da camada de entrada composta de unidades sensoriais (nós de origem); os resultados resultantes da primeira camada oculta são aplicados à próxima camada oculta; e assim por diante pelo resto da rede.

A Figura - 2.5 apresenta um exemplo de rede MLP.

Figura 2.5: Exemplo - Rede MLP



Fonte: Autoria própria (Adaptado de HAYKIN et al., 2009).

Para o treinamento de uma rede MLP, um método comumente utilizado é o algoritmo *backpropagation* (retropropagação), que funciona em duas fases:

1. Na fase *forward*, os pesos sinápticos da rede são fixos e o sinal de entrada é propagado através da rede, camada por camada, até atingir a saída. Assim, nesta fase, as alterações são delimitadas aos potenciais e saídas de ativação dos neurônios na rede;

2. Na fase *backward*, um sinal de erro é produzido comparando a saída da rede com a resposta desejada. O sinal de erro resultante é propagado pela rede, novamente camada por camada, mas desta vez a propagação é realizada na direção inversa. Nesta segunda fase, ajustes sucessivos são feitos nos pesos sinápticos da rede.

Em uma rede MLP, podem ser identificados dois tipos de sinais, sendo eles:

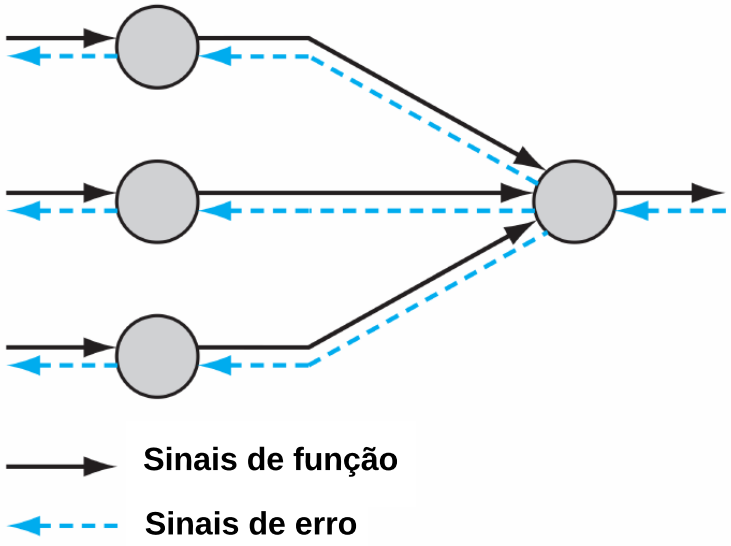
1. **Sinais de Função:** Um sinal de função é um sinal de entrada (estímulo) que chega na extremidade de entrada da rede, propaga-se para frente (fase *forward*), neurônio

por neurônio, através da rede e emerge na extremidade de saída da rede como um sinal de saída.

2. **Sinais de erro:** Um sinal de erro se origina em um neurônio de saída da rede e se propaga para trás (fase *backward*), camada por camada, através da rede.

A Figura - 2.6 exemplifica a propagação dos sinais em uma rede MLP.

Figura 2.6: Propagação dos sinais em uma rede MLP



Fonte: Autoria própria (Adaptado de HAYKIN et al., 2009).

2.9 Considerações finais

Este capítulo apresentou uma fundamentação teórica sobre os diferentes campos de estudos que fazem parte do desenvolvimento deste trabalho.

Assim, em um primeiro momento, o capítulo fez uma introdução sobre Linha de Produto Software (LPS), Arquitetura de Linha de Produto de Software (ALPS) e *Search-based Software Engineering* (SBSE). Na sequência, foram apresentadas a abordagem MOA4PLA e a ferramenta que implementa a mesma, denominada OPLA-Tool.

Por fim, o capítulo apresentou uma discussão sobre processos interativos em SBSE e alguns possíveis problemas que podem ocorrer nestes, como o problema da fadiga. Este problema vem sendo, em alguns casos, tratado por meio da utilização de algoritmos de Aprendizagem de Máquina (AM), cuja introdução sobre este tópico encerra o capítulo.

O próximo capítulo apresenta os resultados do mapeamento sistemático realizado para se ter um maior embasamento para o desenvolvimento deste trabalho.

3

Trabalhos relacionados

Esta seção apresenta um Mapeamento Sistemático (MS) desenvolvido com o intuito de aprofundar os conhecimentos sobre o tema de pesquisa e identificar lacunas que ainda poderiam ser exploradas. Além disso, foram identificadas estratégias para a resolução de problemas de natureza similar ao problema abordado neste trabalho.

Inicialmente, optou-se por fazer a pesquisa com vistas a se encontrar trabalhos que contivessem quatro termos inter-relacionados: "*Software Product Lines*"(SPL) "*Search-based software engineering*"(SBSE), "*Machine Learning*"(ML) e "*Human-computer interaction*"(HCI). Contudo, os resultados foram pouco significativos, uma vez que são escassos os trabalhos que relacionam estes quatro termos. Assim, posteriormente, as *strings* de buscas foram "divididas", ou seja, foram utilizadas strings de buscas menores para que o objeto da busca não fosse tão específico (ver Apêndice A).
As bases utilizadas para o mapeamento sistemático foram: Science Direct, ACM, IEEE, Springer e Research Gate. O objetivo principal do MS foi o de responder às seguintes questões:

- Quais são os problemas de SBSE solucionados utilizando interação homem computador durante o processo de otimização?
- Quais métodos e técnicas são empregados para suportar a interação homem computador no processo de otimização em SBSE?

A seleção dos trabalhos se deu a partir de diferentes etapas, sendo elas:

- **Etapa 01 (resultados da busca)** - Buscou-se pelas palavras chaves e sinônimos nas seções: título, *abstract* e palavras-chaves por meio das ferramentas de filtragem oferecidas pelas bibliotecas digitais selecionadas;

- **Etapa 02 (1ª. Seleção) -** Leitura do título e *abstract* para identificar a relevância dos estudos encontrados no contexto das perguntas do mapeamento sistemático, utilizando-se os critérios de seleção (listados no Apêndice A);

- **Etapa 03 (2ª. Seleção)** - A partir dos trabalhos selecionados na Etapa 02, realizou-se a leitura da introdução e conclusão para se ter uma ideia mais concreta da contribuição que o trabalho poderia trazer;

- **Etapa 04 (seleção final)** -Para esta etapa, os trabalhos considerados como relevantes na Etapa 3 foram lidos na íntegra, buscando-se por informações que viessem a contribuir com o presente trabalho. Para ser considerado relevante, um estudo deveria ter ao menos um critério de inclusão e nenhum critério de exclusão.

O Apêndice A apresenta o protocolo adotado no MS e, na próxima seção, são apresentados os resultados obtidos.

3.1 Resultados do mapeamento sistemático

A execução da busca trouxe um total de 194 trabalhos que estavam de alguma forma relacionados com as strings utilizadas. Após isso, foram lidos título, resumo e palavras-chave destes trabalhos, o que gerou a 1ª. Seleção, diminuindo-se o total para 69 trabalhos. Destes, para uma melhor clareza, foram lidos também a introdução e a conclusão, gerando-se a 2ª. Seleção com 22 trabalhos. Por fim, optou-se pela leitura na íntegra destes 22 trabalhos o que levou à seleção final de 8 trabalhos. A Tabela - 3.1 sumariza os resultados da 1ª. Seleção, 2ª. Seleção e Seleção Final.

Tabela 3.1: Resultados da busca

Bibliotecas	Resultados da Busca	1ª. Seleção	2ª. Seleção	Selecionados
Science Direct	16	14	4	3
ACM	49	13	4	0
IEEE	74	28	9	4
Springer	32	9	3	1
Research Gate	23	5	2	0
Total	194	69	22	8

Fonte: Autoria própria.

A partir da seleção final, constatou-se que os autores desses 8 trabalhos mencionavam também outros trabalhos que poderiam ser úteis aos objetivos aqui apresentados. Assim,

fez-se uma busca manual para se encontrar essas publicações, o que levou à adição de mais 6 trabalhos, que podem ser observadas na Tabela - 3.2.

Tabela 3.2: Encontrados a partir da seleção manual

Bibliotecas	Encontrados a partir da seleção final
Scopus	1
ACM	2
Springer	2
Academic Press	1
Total	6

Fonte: Autoria própria.

A próxima seção apresenta uma breve descrição do conteúdo tratado em cada um dos 14 trabalhos que foram selecionados, levando-se em consideração as questões definidas para o mapeamento sistemático.

3.2 Visão geral dos trabalhos selecionados

Durante o processo de busca, optamos por dividir as *strings* de busca para que as mesmas não fossem tão específicas e assim fosse possível obter um número maior de trabalhos. Deste modo, foram encontrados trabalhos relacionando SBSE com ML, HCI com ML, HCI em SBSE e também com estes três termos relacionados (SBSE, ML e HCI).

Para a discussão dos trabalhos a seguir, as siglas ML e HCI serão utilizadas com as suas respectivas siglas em português, AM (Aprendizagem de Máquina) e IHC (Interação Homem-computador).

Dos trabalhos que relacionam **SBSE** e **AM**, Duro et al. (2014) focam na problemática da redução da quantidade de funções-objetivo, visto que, muitas vezes, há uma falta de critério por parte dos DM que faz com que muitas funções-objetivo tragam resultados parecidos. Assim, a utilização de AM nesse caso, visa "aprender" as preferências do DM para, posteriormente, poder guia-lo no processo de escolher funções-objetivo que evitem redundâncias. Para Zhang et al. (2011) a utilização de AM no campo de SBSE se dá sob cinco diferentes frentes, sendo elas: AM para inicialização de população; AM para avaliação e seleção de *fitness*; AM para reprodução e avaliação populacional; AM para adaptação de algoritmo; e, AM para busca local. Mostra-se assim que, cada vez mais a utilização de AM está sendo considerada em pesquisas na área de SBSE. Uma outra maneira da utilização conjunta de AM e SBSE é relatada em Litao et al.(2012). Neste trabalho, algoritmos genéticos são utilizados para buscar os melhores "pontos de cortes"

para discretizar atributos. O processo de discretização de atributos visa a transformação de atritutos contínuos em discretos. Como exemplo, um atributo contínuo (comprimento), pode precisar ser transformado em um atributo com as categorias discretas: curto, médio ou longo. Com a utilização de algoritmos de busca, o trabalho de Litao et al. (2012) conseguiu um melhor acurácia na classificação de dados quando comparada a outros algoritmos de classificação.

Em outros resultados das buscas, dois trabalhos focam na relação de **IHC** e **AM**. Ware et al.(2001) buscam construir melhores algoritmos classificadores. Para isto, os dados de treinamento são plotados em um gráfico 2D e o DM pode, interativamente, contruir uma árvore de decisão. Singh et al. (2009), por sua vez, focam em como diminuir a fadiga do DM em um modelo interativo baseado em AM. Neste caso, a cada nova geração, o algoritmo agrupa soluções em "grupos de soluções" e questiona o DM para fazer um ranqueamento destas soluções. Deste modo, o algoritmo vai sendo "treinado" e passa a compreender melhor as preferências do DM, e, com isso, a interação tende a ser cada vez menos frequente.

Dos trabalhos selecionados, a maior parte (6 trabalhos) refere-se à **IHC** em **SBSE**. Em Su e Zhang (2010), a interação com o DM ocorre a cada nova geração de resultados obtidos por meio de "crossover" e "mutation". Os resultados obtidos, quando condizentes com o esperado, são utilizados como dados de entrada para uma nova iteração. Ferreira e Vergilio (2017) realizaram uma revisão sistemática no intuito de demonstrar como está o campo de pesquisa que integra IHC em SBSE. Esses autores demonstraram que este é um campo de pesquisa que está em crescimento, com algoritmos evolutivos desenvolvidos com este objetivo, denominados IGA (*Interactive Genetic Algorithm*). Já em outra revisão sistemática, Ramirez et al.(2018) chamam a atenção para a quantidade necessária de interações do DM durante o processamento. Em geral, percebeu-se que os trabalhos continham um número fixo de interações. Entretanto, foram identificados também trabalhos em que a quantidade e o momento das interações ficava à escolha do DM. Importante ressaltar que nos dois últimos trabalhos citados, os autores destacam a necessidade de se evitar a fadiga humana no processo, citando que a utilização de algoritmos de AM poderia ser útil neste caso.

O problema da fadiga é relatado também com mais ênfase em dois outros trabalhos. Shackelford (2007) discute questões de implementação em Computação Interativa Evolucionária. Neste trabalho é descrito como maior problema a fadiga do DM. Neste caso, o autor indica a utilização de perfis e/ou *templates* com as preferências do DM. Assim, pode ser possível que o sistema absorva o conhecimento implícito do DM sem que o mesmo necessite formalizar esta informação. Em outro estudo, mais recente, Shackelford e Simons

(2014) discutem diversas estratégias para lidar com a interação em SBSE, levando-se em consideração a fadiga do DM. Neste contexto o trabalho tem como objetivo descrever as possíveis maneiras de mostrar o mínimo possível de soluções para a avaliação do DM.

O último trabalho que possui como foco **IHC** e **SBSE** detalha como ocorre a utilização das preferências do DM em Computação Evolucionária Multiobjetivos. Bechikh et al.(2015) relatam a existência de algumas estruturas de preferências que os DM costumam utilizar. De todas as citadas, a mais utilizada denomina-se *Reference Point*. Neste tipo de estrutura (também chamada de Meta) o DM fornece para cada objetivo o nível que ele gostaria de alcançar.

Por fim, as buscas retornaram também três trabalhos que englobam conjuntamente **SBSE**, **AM** e **IHC**. Amal et al. (2014) propõem uma técnica para refatoração de software que utiliza algoritmos genéticos em conjunto com Redes Neurais Artificiais (RNA). A partir de algumas iterações do algoritmo, o DM é questionado a avaliar as soluções geradas, que passarão a ser o conjunto de treinamento para a rede neural. Após um certo número de iterações/interações, a rede neural passa a substituir o DM no processo. Interessante fato é apontado neste trabalho também sobre a quantidade de interações. Concluiu-se que 35 interações seria um ponto ótimo para o problema tratado, já que após esta quantidade não se conseguia uma melhora no modelo. Abordagem semelhante é relatada em Pedro e Takahashi (2013). Neste trabalho, utiliza-se o algoritmo denominado iTDEA (*Interactive Territory Defining Evolutionary Algorithm*) que, em determinados pontos do processo evolutivo, questiona o DM sobre o a execução do processamento para que este opine sobre o mesmo. As respostas fornecidas compõem um conjunto de treinamento de um modelo de AM que visa substituir o DM após o padrão de preferências deste ter sido determinado.

O uso de algoritmos de AM também é descrito em Araújo e Paixão (2014). Este trabalho lida com o problema da próxima versão (NRP - *Next Release Problem*). Este problema consiste em selecionar quais requisitos serão implementados em uma próxima versão de um sistema. As abordagens que envolvem SBSE e NRP carecem da capacidade de incluir com eficiência a opinião humana e suas peculiaridades no processo de busca. Assim, é apresentada uma arquitetura para resolver o problema do NRP, incorporando as preferências do DM e, por meio de um modelo de AM, buscando minimizar o problema da fadiga humana. Os resultados experimentais demonstram que essa abordagem é capaz de incorporar com sucesso as preferências do DM na solução final.

As tabelas a seguir (Tabela - 3.3 e Tabela - 3.4) apresentam um panorama geral sobre os trabalhos selecionados e acima brevemente descritos, levando-se em consideração as questões de pesquisa, o problema de SBSE tratado e o método de interação utilizado.

Tabela 3.3: Resumo dos trabalhos selecionados no Mapeamento Sistemático

Trabalho	Problema SBSE	Método de interação	Algoritmo evolutivo	Abordagem AM	String de Busca
Duro, et al (2014	Diminuição da quantidade de funções-objetivo.	Um modelo de AM busca por funções-objetivo que possivelmente levariam a uma mesma Fronteira de Pareto.Questiona-se o DM para escolher entre possíveis redundânciasnas funções-objetivo.	MSOPS-II - Multiple Single Objective Pareto Sampling	Análises -MOSS e k-EMOSS	("Search-based software engineering" AND "Machine Learning")
Zhang et al. (2011	Foca-se em como melhorar algoritmos evolutivos com o auxílio de técnicas de AM	Não há referências específicas sobre processos interativos	Não há um específico. Trata-se de um Mapeamento Sistemático	AM é utilizada para o auxílio em diferentes frentes da computação evolutiva.	("Search-based software engineering" AND "Machine Learning")
Litao, et al. (2012)	Buscar os melhores "pontos de cortes"necessários ao algoritmo classificador de AM	Não se aplica	MOGA (Multi-Objective Genetic Algorithm)	É proposto um algoritmo de AM que se utiliza de técnicas da computação evolutiva para se obter um melhor algoritmo de AM.	("Search-based software engineering" AND "Machine Learning")
Ware, et al. (2001)	Não se aplica	A partir de uma interface gráfica, o DM define para onde a árvore de decisão deve crescer	Não se aplica	Neste trabalho, busca-se construir melhores algoritmos classificadores, com base em opiniões de DM.	("Human-computer interaction" AND "Machine Learning")
Sing, et al. (2009)	Não se aplica	O DM é questionado a ranquear um conjunto de soluções clusterizadas	IMOGA (interactive multiobjective genetic algorithm)	A abordagem aqui descrita utiliza o classificador Naive Bayes.	("Human-computer interaction" AND "Machine Learning")
Su e Zhang (2010)	Inovação na forma de geração de diferentes *designs* para a formação de um novo produto	A cada nova geração os DM são questionados. Quando os resultados estão dentro do esperado, utiliza-se os mesmos como entrada para uma nova interação.	Não há um específico. Apenas, descreve-se genericamente como algoritmo genético.	Não se aplica	("Human-computer interaction" AND "Search-based software engineering")
Ferreira e Vergílio (2017)	Não há um problema específico. Visa-se dar um panorama geral do processo interativo em SBSE	Algoritmos denominados Interactive Genetic Algorithms são os mais utilizados.	Há vários algoritmos mencionados, tais como: DE, IBEA, Biomimetic, HCM, VEGA NSGA-II, IGA, e ACO	Não se aplica	("Human-computer interaction" AND "Search-based software engineering")

Tabela 3.4: Resumo dos trabalhos selecionados no Mapeamento Sistemático

Trabalho	Problema SBSE	Método de interação	Algoritmo evolutivo	Abordagem AM	String de Busca
Ramirez, et al. (2018)	Descobrir se há uma quantidade ideal de interações	Os trabalhos relatam a busca pela opinião do DM em pontos fixos do processamento. Mas há também trabalhos em que o DM escolhe quando intervir no processo.	Vários. Trata-se de uma revisão sistemática.	Não se aplica.	("Human-computer interaction" AND "Search-based software engineering")
Bechikh, et al. (2015)	Não há problema específico. Descreve-se sobre estruturas de preferências utilizadas pelos DMs em processos interativos em SBSE	Existem várias de estruturas de preferências utilizadas. Tais estruturas buscam mostrar ao DM uma parte menor das soluções, para que o mesmo possa decidir entre estas as melhores.	Vários. Destaca-se as variações interativas do algoritmo NSGA-II	Não se aplica.	("Human-computer interaction" AND "Search-based software engineering")
Shackelford (2007)	Como utilizar a interação humana no processo e lidar com o problema da fadiga.	Não há um método de interação específico. Apenas, levanta-se questões sobre o processo interativo. A interação visa a compor perfis e/ou templates que caracterizam cada DM.	Não há um específico. Descreve-se sobre questões relacionadas à computação evolutiva em geral.	Não se aplica.	("Human-computer interaction" AND "Search-based software engineering")
Shackelford e Simons (2014)	Como lidar com a fatiga do DM em processos interativos.	O processo interativo segue estratégias adequadas a cada caso, visando-se a diminuição tanto quanto possível do número de soluções geradas.	Não se descreve sobre um algoritmo especificamente.	Não se aplica	("Human-computer interaction" AND "Search-based software engineering")
Amal (2014)	Refatoração de software	Após algumas iterações, questiona-se o DM a avaliar a solução gerada. As soluções bem avaliadas passam a ser o conjunto de treinamento da rede neural.	Descreve-se genericamente como IGA. Sendo um modelo de IGA que contempla AM.	Utilizou-se neste caso Redes Neurais Artificiais para compor o modelo.	("Search-based software engineering"AND "Machine Learning"AND " Human-computer interaction")
Pedro e Takahashi (2013)	Diminuição da quantidade de questionamentos feitos ao DM em um algoritmo genético interativo	A cada evolução do algoritmo, o DM é questionado sobre a mesma. As soluções bem avaliadas passam a ser o conjunto de treinamento da rede neural.	iTEDA (interactive territory defining evolutionary algorithm	Utilizou-se Redes Neurais Artificiais.	("Search-based software engineering"AND "Machine Learning"AND " Human-computer interaction")
Araújo e Paixão (2014)	Seleção de requisitos para a próxima release de software	Em determinados momentos do processamento, o DM intervém no processo. Essas intervenções visam a fornecer a opinião humana sobre o processo.	Algoritmo Genético Interativo	Utilizou-se Least Means Square, Multilayer Perceptron	("Search-based software engineering"AND "Machine Learning"AND " "Human-computer interaction")

3.3 Considerações finais

Este capítulo apresentou os trabalhos que foram selecionados a partir de um MS e embasam o desenvolvimento deste trabalho. A leitura e análise dos trabalhos selecionados permitiu trazer *insights* de como pode ser utilizada IHC no processo de otimização realizado pela abordagem MOA4PLA e ferramenta OPLA-Tool. Além disso, foi possível perceber que o problema da fadiga é relatado em vários trabalhos e algumas abordagens são propostas para tratá-lo, utilizando-se, muitas vezes, de algoritmos de AM para este tratamento.

Por fim, o MS também demonstrou não haver trabalhos que utilizam otimização interativa no contexto de LPS.

No capítulo a seguir são descritos os métodos de pesquisa e os materiais utilizados para o desenvolvimento deste trabalho.

4

Métodos e Materiais

4.1 Considerações iniciais

Este capítulo descreve os métodos de pesquisa, avaliação e também os materiais utilizados para o desenvolvimento deste trabalho. Neste sentido, na Seção 4.2 são detalhados todos os passos necessários ao desenvolvimento do trabalho, desde a definição do tema de pesquisa até a análise dos resultados obtidos experimentalmente. Na sequência, Seção 4.2.1, são descritos os métodos utilizados para a avaliação experimental da proposta de pesquisa, que conta com análises quantitativas e qualitativas. Posteriormente, a Seção 4.3 aborda os materiais utilizados no desenvolvimento do trabalho, tais como: softwares e ALPS utilizados. Por fim, Seção 4.4, são discutidas algumas ameaças à validade dos experimentos realizados. A seguir, cada uma destas etapas é brevemente descrita.

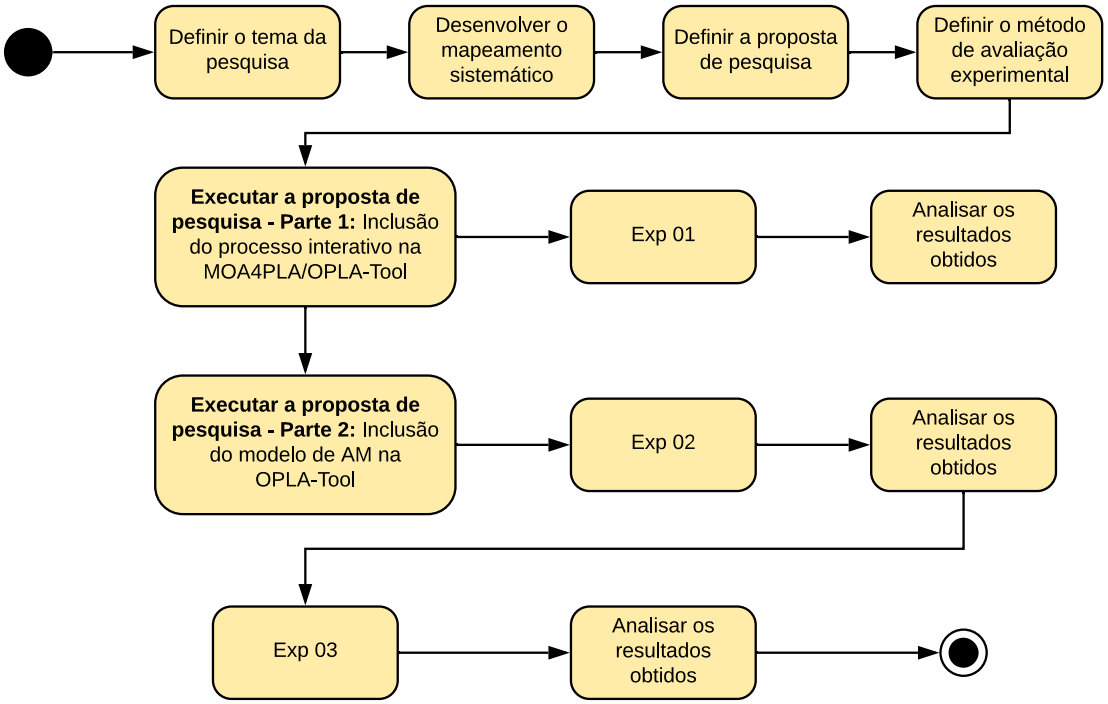
4.2 Método de pesquisa

Para atingir o objetivo proposto, a sequência de atividades apresentadas na Figura - 4.1 foi realizada.

Definir o tema da pesquisa: A utilização da interação homem-computador em processos de otimização tem ganhado cada vez mais atenção em pesquisas em SBSE, visto o seu potencial por trazer resultados mais adequados a cada DM individualmente. Diante disso, propôs-se uma abordagem que incluía o DM (arquiteto) no processo de otimização realizado pela MOA4PLA e, consequentemente, pela ferramenta OPLA-Tool.

Desenvolver o mapeamento sistemático: Para a elaboração da proposta de pesquisa, fez-se necessário o desenvolvimento de um mapeamento sistemático para descobrir

Figura 4.1: Atividades realizadas para o desenvolvimento do trabalho.



Fonte: Autoria própria.

o atual estado da arte no que diz respeito a processos interativos em SBSE. Os resultados apontaram que não havia nenhum trabalho que incluía o arquiteto interativamente no contexto de projeto de ALPs. Além disso, fato relevante apontado pelo mapeamento sistemático diz respeito ao problema da fadiga, ocasionado pelo elevado número de interações com o DM. Neste sentido, alguns estudos apontaram a utilização de algoritmos de AM para lidar com este problema.

Definir a proposta de pesquisa: Com base nos resultados do mapeamento sistemático, a proposta de pesquisa foi definida da seguinte maneira: em um primeiro momento, foi definido que a abordagem MOA4PLA seria modificada para permitir a interação com o arquiteto durante o processo de otimização, com consequente implementação desta modificação na OPLA-Tool, por meio da modificação do algoritmo NSGA-II. Após a introdução do processo interativo, a proposta de pesquisa passou para a segunda etapa. Nesta etapa, um modelo de AM seria incorporado à ferramenta OPLA-Tool para atuar no tratamento do problema da fadiga.

Definir o método de avaliação experimental: Para a avaliação experimental, foi definido que experimentos seriam realizados (três experimentos). O primeiro experimento (Exp 01), teve como objetivo comparar o MIP (definido na proposta de pesquisa) com

o modelo atual (onde não há interação durante o processo de otimização), no intuito de verificar se o MIP era adequado à MOA4PLA e, consequentemente, à OPLA-Tool. Posteriormente, com a introdução de algoritmos de AM na OPLA-Tool, outros experimentos (Exp 02 e Exp 03) foram realizados para descobrir se o modelo de AM suportaria o tratamento do problema da fadiga. Para todos os experimentos, foi necessário, também, definir-se indicadores de qualidade e testes estatísticos que possibilitaram mostrar a validade dos experimentos.

Executar a proposta de pesquisa: A execução da proposta ocorreu em três etapas. Na primeira, o processo interativo foi introduzido na abordagem MOA4PLA e na ferramenta OPLA-Tool (por meio da modificação do algoritmo NSGA-II), permitindo que o arquiteto definisse a partir de qual momento ele iria interagir com o processo de otimização e quantas interações seriam feitas. Na segunda etapa, para compor o modelo de AM, foi introduzido o algoritmo MLP e três diferentes estratégias foram testadas para a escolha de qual destas seria a mais adequada para a formação do conjunto de dados de treinamento do algoritmo. A elaboração destas diferentes estratégias foi necessária para o tratamento de uma deficiência encontrada para o modelo de AM, a pouca quantidade de dados para o conjunto de treinamento. Por fim, na terceira etapa, a melhor estratégia foi avaliada por meio de um experimento (Exp 03) com a participação de arquitetos de software.

Executar as avaliações experimentais: Foram executados três experimentos, sendo dois deles quanti-qualitativos e o outro apenas quantitativo. O primeiro experimento (quanti-qualitativo) envolveu a execução da OPLA-Tool de duas maneiras. A primeira execução (Exec-MA - Execução no modelo atual) foi feita com a ferramenta no seu estágio atual de desenvolvimento. A segunda execução (Exec-MIP - Execução com o modelo interativo proposto) contou com a participação de um especialista no intuito de verificar a eficácia do MIP. O segundo experimento (quantitativo) envolveu a realização de vários testes para verificar qual das estratégias elaboradas para a formação do conjunto de dados de treinamento do modelo de AM trazia os melhores resultados. O último experimento (quanti-qualitativo - Exp 03) contou com a colaboração de especialistas para verificar a eficácia do MIP como um todo, ou seja, foi avaliado se o processo interativo com o modelo de AM proporcionou a geração de ALPS mais adequadas a cada arquiteto e se os participantes sentiram cansaço (fadiga) durante o processo interativo.

Analisar os resultados obtidos: Análises foram realizadas sobre os resultados de cada um dos experimentos. No experimento 1 (Exp 01), a análise quanti-qualitativa permitiu fazer uma comparação entre as soluções obtidas com o modelo proposto e o modelo atual. No experimento 2 (Exp 02), testes estatísticos possibilitaram indicar qual

das estratégias elaboradas para a formação do conjunto de dados de treinamento do modelo de AM atingiu os melhores resultados. Por fim, no experimento 3 (Exp 03), análises quanti-qualitativas permitiram inferir se o MIP, juntamente com o modelo de AM, possibilitaram trazer soluções mais adequadas a cada arquiteto e ao mesmo tempo lidar com o problema da fadiga.

4.2.1 Método de avaliação

Com o objetivo de avaliar o MIP, três experimentos foram definidos.

A Tabela - 4.1 sintetiza o método de avaliação, com as colunas, a partir da primeira, representando respectivamente: nome do experimento; objetivos do experimento; tipo de experimento que foi realizado; os algoritmos utilizados em cada experimento; a ALPS utilizada; funções objetivo utilizadas e, por fim, os dados que foram analisados.

No primeiro experimento (Exp 01), o objetivo foi avaliar a validade do MIP. Desta maneira, foram realizadas duas execuções da ferramenta OPLA-Tool, uma com o modelo atual (sem a participação de um arquiteto de software durante o processo de otimização) e outra com o MIP, com a participação de um arquiteto de software. Posteriormente, os resultados trazidos por ambas execuções foram analisados e comparados. Neste sentido, os valores de *fitness* de ambas execuções foram utilizados para a formação do PF_{true} (conceituado na Subseção 4.2.2). Além disso, os valores de avaliação dados pelo arquiteto para as ALPS avaliadas durante Exec-MIP foram coletados para analisar se os mesmos foram evoluindo com o decorrer do processo de otimização.

Para Exp 02, novos algoritmos foram incorporados à OPLA-Tool, ambos no intuito de tratar o problema da fadiga. Primeiramente, com base em Freire et al. (2019), o algoritmo K-means foi introduzido para que fosse possível a redução da quantidade de ALPS avaliadas. Em Exp 01, o arquiteto necessitava avaliar todas as soluções geradas a cada momento de interação. Com a introdução do K-means, foi possível o agrupamento de soluções em grupos (*clusters*) de soluções semelhantes, possibilitando que o arquiteto avaliasse apenas uma solução em cada grupo de soluções.

Tabela 4.1: Síntese do método de avaliação.

Nome do Experimento	Objetivo do Experimento	Tipo do Experimento	Algoritmos utilizados	ALP utilizada	Funções-objetivo utilizadas	Dados analisados
Exp 01	(i) Avaliar a incorporação da interação homem-computador na OPLA-Tool; (ii) Comparar as implementações da OPLA-Tool com e sem interação.	Quantitativo e Qualitativo	NSGA-II	AGM	FM, COE, ACLASS	Fitness das funções-objetivo; Valores de notas atribuídos pelo arquiteto.
Exp 02	(i) Verificar qual das estratégias elaboradas para o aumento da quantidade de dados do conjunto de treinamento do algoritmo MLP era a mais adequada.	Quantitativo	NSGA-II, K-means e MLP	AGM	FM, COE, ACLASS	Percentual de acertos, de cada estratégia; Dados estatísticos fornecidos pelo Weka.
Exp 03	(i) Avaliar a interação homem-computador, juntamente com os algoritmos de AM na OPLA-Tool, com a participação de arquitetos de software.	Quantitativo e Qualitativo	NSGA-II, K-means e MLP	AGM	FM, COE, ACLASS	Valores de notas atribuídos por cada arquiteto em cada momento de interação.

O segundo algoritmo introduzido foi o MLP. A introdução deste algoritmo teve como objetivo compor o modelo de AM, para aprender sobre o comportamento do arquiteto e, posteriormente, substituí-lo no processo de avaliação de ALPS. Um problema encontrado em relação a introdução deste algoritmo foi a pouca quantidade de dados que seriam gerados para o conjunto de treinamento. Os dados são gerados em tempo de processamento, a partir de cada avaliação de ALPS por parte do arquiteto. Considerando que o processo de avaliação de ALPS é uma tarefa que demanda tempo, não poderiam ser avaliadas muitas ALPS, visto o cansaço (fadiga) do arquiteto que estas avaliações trariam, acarretando em uma pouca quantidade de dados. Neste sentido, três diferentes estratégias foram propostas para tratar a deficiência na quantidade de dados. Assim, Exp 02 teve como objetivo, também, testar as três estratégias propostas para a formação do conjunto de dados de treinamento do algoritmo MLP e verificar qual delas era a mais adequada. Os testes possibilitaram fazer o cálculo da acurácia (com o percentual de acertos de cada estratégia), bem como analisar os cálculos de RAE, RRSE, MAE e RMSE fornecidos pelo software Weka (Seção 4.3).

Por fim, a realização de Exp 03 teve como objetivo avaliar, sob o ponto de vista de um arquiteto de software, o modelo de AM, com a utilização da estratégia selecionada em Exp 02 para a formação do conjunto de dados de treinamento. Foram coletadas durante Exp 03 as notas (avaliações) atribuídas pelos arquitetos durante 3 momentos de interação, sendo avaliadas uma solução por *cluster* de soluções. Além disso, os arquitetos avaliaram as soluções restantes (soluções finais) da execução (sendo uma por *cluster* novamente) para verificar se as mesmas estavam de acordo com suas necessidades. Caso as soluções restantes fossem bem avaliadas, isso seria um indicativo de que o modelo de AM estava apto a capturar as preferências do arquiteto.

4.2.2 Técnicas de análise quantitativa

Esta seção apresenta as técnicas e indicadores que foram utilizados nos três experimentos realizados para a avaliação dos resultados dos mesmos.

Indicadores de Qualidade

Como meio de avaliar os algoritmos em processo de otimização multiobjetivos, costuma-se utilizar os seguintes conjuntos de soluções: PF_{approx} , PF_{known} e PF_{true}. O conjunto de soluções do PF_{approx} é formado pelas soluções não-dominadas dadas por uma execução de um algoritmo. PF_{known}, por sua vez, é o conjunto de soluções formado pelas soluções não-dominadas dentre todas as PF_{approx} obtidas por um dado algoritmo. Em muitos

problemas, Fronteira de Pareto Real (PF_{true}) não é conhecida. Neste caso, considera-se que o PF_{true} é formado pelas soluções não dominadas da união das PF_{known} encontradas por todos os algoritmos avaliados (Yoo e Harman, 2007; Zitzler et al., 2003).

Para avaliar a qualidade de cada solução gerada, foi utilizado o indicador Euclidian Distance, descrito a seguir.

Euclidean Distance to the Ideal Solution

O indicador de *Euclidean Distance to the Ideal Solution* (ED), apesar de não ser considerado um indicador de qualidade, é usado como um meio de auxiliar o tomador de decisão a escolher uma solução entre várias em um conjunto. Neste sentido, o valor de ED indica qual solução está mais próxima de uma chamada "solução ideal". Considerando-se um problema de minimização, tal como o problema modelado pela abordagem MOA4PLA, a solução ideal é uma solução que contempla o menor valor encontrado para cada objetivo selecionado (Cochrane e Zeleny, 1973).

Medidas estatísticas utilizadas

Para a análise dos resultados gerados pelos algoritmos de AM, algumas medidas estatísticas foram utilizadas, sendo estas descritas a seguir:

Estatística Kappa (K): é uma medida de concordância usada em escalas nominais que nos fornece uma ideia do quanto as observações se afastam daquelas esperadas (resultantes do acaso) indicando-nos assim o quão legítimas as interpretações são (Fleiss e Cohen, 1973). A escala K varia de 0 a 1 de acordo com Tabela - 4.2.

Tabela 4.2: Escala estatística kappa

Valor de K	**Interpretação**
$k = 0$	Não existe concordância
$0 < k \leq 0.2$	Concordância mínima
$0.21 < k \leq 0.4$	Concordância razoável
$0.41 < k \leq 0.6$	Concordância moderada
$0.61 < k \leq 0.8$	Concordância substancial
$0.81 < k \leq 1$	Concordância quase perfeita

Fonte: Autoria própria (Adaptado de FLEISS e COHEN, 1973).

Além da estatística Kappa, outras quatro métricas também foram utilizadas: *Mean Absolute error* (MAE), *Root Mean Square Error* (RMSE), *Root Relative Squared Error* (RRSE) e *Root Absolute Error* (RAE) (WITTEN et al., 2016), descritas a seguir:

- **MAE**: A métrica MAE caracteriza-se por ser a média dos erros cometidos pelo modelo de previsão durante uma série de execuções. O seu cálculo é feito subtraindo-se o valor do modelo de previsão ($\widehat{\Theta}$) ao valor verdadeiro (Θ) em cada período de execução. Ao resultado (em módulo), soma-se e divide-se pelo número de valores utilizados para obter a soma (N). O cálculo de MAE é feito pela Equação 4.1.

$$MAE = \frac{1}{N}\sum_{i=1}^{N}\left|\widehat{\Theta_i} - \Theta_i\right| \tag{4.1}$$

- **RMSE**: A métrica RMSE é utilizada também como uma medida para determinar o erro do modelo de previsão. Seu cálculo é feito pela raiz quadrada do quadrado do somatório dos erros de previsão $\left(\widehat{\Theta_i} - \Theta_i\right)$, dividido pelo número de erros (N). A Equação 4.2 é utilizada para o cálculo do RMSE.

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\widehat{\Theta_i} - \Theta_i\right)^2} \tag{4.2}$$

- **RRSE**: A métrica RRSE é utilizada para calcular o erro de previsão em relação ao que seria predito por um preditor simples. O preditor simples em questão prediria apenas a média dos valores reais dados, denotada por $\overline{\Theta}$. Desta maneira, RRSE faz uma normalização, dividindo-se pelo erro quadrado total do indicador padrão. O cálculo de RRSE é feito segundo a Equação 4.3.

$$RRSE = \sqrt{\frac{\sum_{i=1}^{N}\left(\widehat{\Theta_i} - \Theta_i\right)^2}{\sum_{i=1}^{N}\left(\overline{\Theta} - \Theta_i\right)^2}} \tag{4.3}$$

- **RAE:** A métrica RAE calcula o erro absoluto total, sendo um tipo de normalização semelhante a RRSE. Para o RAE os erros são normalizados pelos erros de um preditor simples (que prevê apenas os valores médios). O cálculo de RAE é feito segundo a Equação 4.4.

$$RAE = \frac{\sum_{i=1}^{N}\left|\widehat{\Theta_i} - \Theta_i\right|}{\sum_{i=1}^{N}\left|\overline{\Theta} - \Theta_i\right|} \tag{4.4}$$

4.3 Materiais utilizados

Esta seção apresenta os materiais utilizados nos experimentos realizados. Os três experimentos foram feitos utilizando a ferramenta OPLA-Tool, sendo que, no Exp 01, a ferramenta foi utilizada em sua versão atual e também na versão que suporta o MIP. Além da ferramenta OPLA-Tool, foram utilizados outros materiais, sendo descritos a seguir.

Software Weka[1] (Waikato Environment for Knowledge Analysis)

Os algoritmos de AM utilizados neste trabalho fazem parte do pacote de software Weka. Weka é um conjunto de softwares de AM escrito em Java, desenvolvido na Universidade de Waikato, Nova Zelândia.

O software contém um conjunto de ferramentas de visualização e algoritmos para análise de dados e modelagem preditiva, juntamente com interfaces gráficas do usuário para facilitar o acesso a essas funções .

ALPS utilizada

Em todos os experimentos, utilizou-se a ALP Arcade Game Maker (AGM).

AGM é uma ALP que inclui três jogos Brickles, Bowling e Pong, desenvolvida pelo Software Engineering Institute (SEI) (SEI, 2019). A Tabela - 4.3 mostra a quantidade de alguns elementos arquiteturais da ALPS AGM.

Tabela 4.3: Elementos arquiteturais da ALP AGM

ALP	Componentes	Interfaces	Classes	Features	Variabilidades
AGM	9	14	30	11	5

Fonte: Autoria própria (adaptado de FREIRE et al., 2019)

4.4 Ameaças à validade

A principal ameaça encontrada no Exp 01 foi o fato de apenas um arquiteto participar do experimento. Assim, a avaliação das ALPS geradas em cada momento de interação foi feita por apenas um DM, o que faz com que não seja possível generalizar os resultados deste experimento para uma população maior. Além disso, no terceiro momento de interação,

[1]Informações sobre o software Weka, bem como *download* do mesmo, podem ser encontrados em https://www.cs.waikato.ac.nz/ml/weka/

observou-se que o arquiteto já apresentava fadiga, o que pode ter impactado em sua avaliação.

O problema da fadiga também afetou o que foi planejado inicialmente para o experimento. Exp 01 foi planejado para ter cinco interações, mas, quando o arquiteto apresentou fadiga, as duas interações restantes não foram realizadas, afim de evitar impacto na avaliação das soluções.

Em Exp 02, embora os resultados tenham sido promissores, pode-se destacar como ameaça à validade o fato de que o experimento não foi feito com arquitetos de software. Desta maneira, não pode ser atestado que os resultados encontrados também seriam reproduzidos em um experimento com a participação de arquitetos de software.

Por fim, em Exp 03, a principal ameaça à validade reside no fato de que os participantes do experimento, em sua grande maioria, não possuíam conhecimentos avançados em análise de ALPS. Além disso, alguns participantes consideraram também que a quantidade de soluções (ALPS) a serem avaliadas foi excessiva, o que pode ter prejudicado a avaliação final das soluções (ALPS restantes da execução) em decorrência do efeito da fadiga.

4.5 Considerações finais

Este capítulo detalhou os métodos e materiais utilizados para a realização dos três experimentos que objetivaram testar a validade do processo interativo na abordagem MOA4PLA e ferramenta OPLA-Tool, bem como verificar a eficácia do modelo de AM introduzido para o tratamento do problema da fadiga.

O capítulo a seguir descreve todos os passos realizados para a introdução do processo interativo na abordagem MOA4PLA e ferramenta OPLA-Tool, e também para a composição do modelo de AM.

5

Modelo proposto para Otimização Interativa de Projeto de ALPS

5.1 Considerações iniciais

Este capítulo descreve o modelo proposto para incorporar as preferências do arquiteto durante o processo de otimização realizado pela abordagem MOA4PLA e automatizado pela ferramenta OPLA-Tool. A Seção 5.2 explica as modificações que foram feitas para permitir a interação com o arquiteto durante o processo de otimização. Para isso, foi necessário, em um primeiro momento, modificar a abordagem MOA4PLA e, em seguida, implementar estas alterações na ferramenta OPLA-Tool. Como descrito na Seção 2.7, em processos interativos em SBSE, um problema frequentemente encontrado é o da fadiga, que pode ser tratado por meio de algoritmos de AM. Neste sentido, um modelo de AM, descrito na Seção 5.3, que inclui os algoritmos K-means e MLP, foi proposto, com três diferentes estratégias para a formação do conjunto de dados de treinamento do algoritmo MLP.

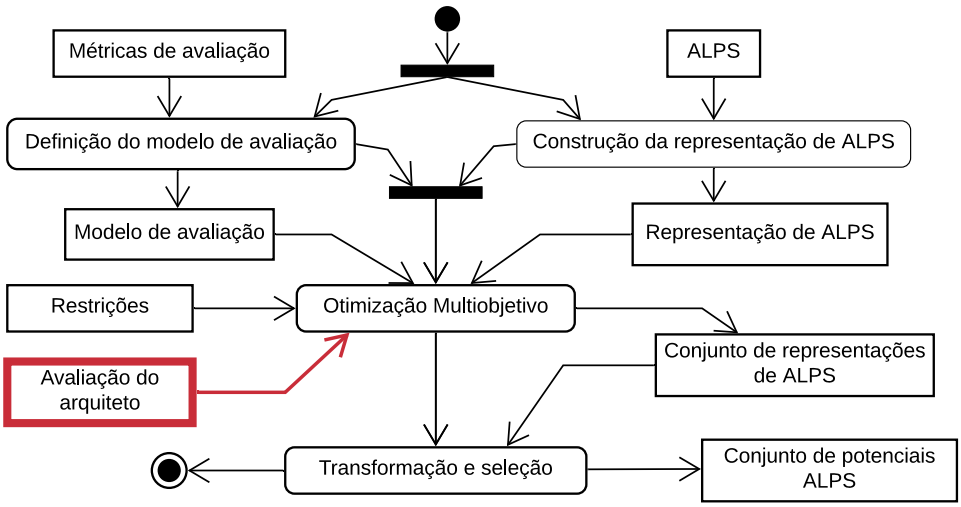
5.2 Modelo inicial para incorporação das preferências do arquiteto na abordagem MOA4PLA e ferramenta OPLA-Tool

Como discutido anteriormente, incluir as preferências do DM durante o processo de otimização pode produzir soluções que se adaptem melhor às necessidades individuais

de cada DM. Esta Seção descreve o Modelo Interativo Proposto (MIP) para a inclusão das preferências do arquiteto na abordagem MOA4PLA e, também, na ferramenta OPLA-Tool.

Para a inclusão das preferências do arquiteto, novos dados devem ser incorporados ao processo de otimização. Esses dados se referem à avaliação subjetiva do arquiteto para as ALPS geradas durante o processamento e são incorporados na abordagem MOA4PLA pela atividade "Avaliação do arquiteto", como demonstrado na Figura - 5.1. Esta atividade fornece novos dados de entrada que afetam diretamente a atividade de Otimização multiobjetivos.

Figura 5.1: Inclusão das preferências do arquiteto na MOA4PLA



Fonte: Autoria própria (Adaptado de COLANZI, 2014).

Como descrito anteriormente (Seção 2.5), a OPLA-Tool fornece dois algoritmos para o processo de otimização de ALPS, sendo eles: NSGA-II e PAES. Na Seção 3 (Tabela - 3.4), foi mencionado que alguns trabalhos relataram modificações no NSGA-II para permitir a interação do DM durante o processamento. A análise dessas modificações mostrou que as mesmas não poderiam ser usadas sem adaptação para o caso específico deste trabalho, visto serem adaptações específicas a determinados contextos. Dessa forma, decidiu-se introduzir modificações no algoritmo original do NSGA-II, a fim de permitir a interação do arquiteto adaptada ao contexto específico deste trabalho.

O modelo proposto em Araújo e Paixão (2014) tem como dados de entrada as configurações utilizadas no processo evolutivo e o número de interações que o DM deseja executar. Após o início do processo evolutivo, o DM fornece uma avaliação subjetiva a cada novo indivíduo (nova solução). Esta avaliação é feita com uma pontuação em um intervalo previamente definido, sendo que a solução recebe a pontuação mais alta quando

satisfaz plenamente as preferências do DM e mínima quando está muito aquém do que ele esperava.

Para o caso específico deste trabalho, no entanto, é necessário também que o DM (arquiteto) defina o momento em que a primeira interação deve ocorrer.

Isso foi necessário pois foi observado que o processo de otimização deve ocorrer por um determinado período para permitir a geração de boas alternativas de projetos de ALPS. Assim, para a introdução do processo interativo na abordagem MOA4PLA foi decidido que, em um primeiro momento, o arquiteto definiria (além das configurações já utilizadas na abordagem) o número de interações que seriam feitas e também o momento da primeira interação. Na sequência, o processo de otimização é iniciado (com a inicialização da população). No momento em que surgir a primeira geração de novos indivíduos (geração de alternativas de ALPS) o sistema verifica se o momento da primeira interação foi atingido. Em caso negativo, o processo de otimização continua até uma nova rodada de soluções geradas, quando então ocorre novamente a verificação em relação ao primeiro momento de interação. Ao se atingir o primeiro momento de interação, as soluções geradas são apresentadas ao arquiteto para que o mesmo possa avaliá-las. Esta avaliação é feita com a atribuição de uma nota (`escore`) no intervalo de 1 a 5, sendo que as:

- Soluções com **escore = 1:** são descartadas do processo evolutivo, pois estão muito aquém das preferências do arquiteto. Desta forma, para a continuidade do processo evolutivo, novas soluções são geradas para completar a quantidade de indivíduos colocada inicialmente como parâmetro de execução;

- Soluções com **escores = 2, 3 ou 4:** retornam ao processo de otimização pois eventualmente podem se tornar soluções com melhores avaliações pelo arquiteto;

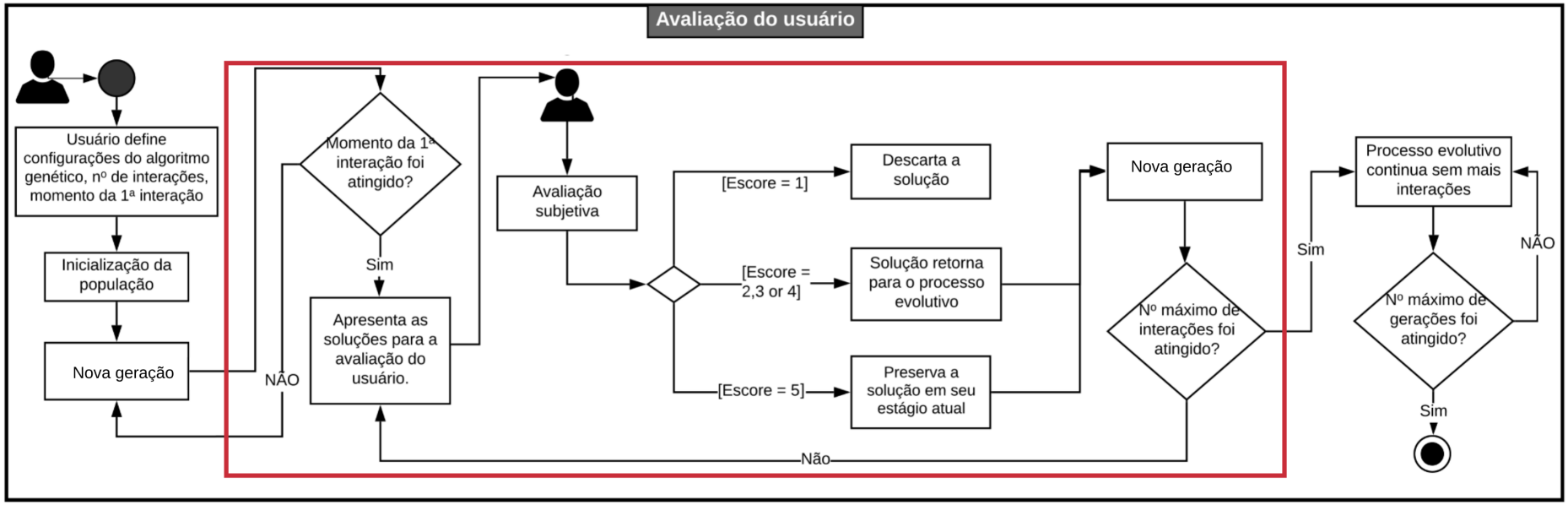
- Soluções com **escore = 5:** são preservadas em seu estágio atual, pois já estão otimizadas suficientemente segundo o ponto de vista deste arquiteto. Neste caso, estas soluções são armazenadas juntamente com as soluções restantes do processo evolutivo. Contudo, soluções com `escore = 5` também retornam ao processo evolutivo pois, a partir destas soluções, podem ser geradas outras soluções que podem receber boas avaliações por parte do arquiteto.

A partir do primeiro momento de interação, a cada nova geração de soluções (novas alternativas de ALPS geradas), o sistema verifica se o número máximo de interações foi atingido. Em caso negativo, o processo de avaliação continua, com a apresentação das soluções geradas para o arquiteto. Em caso afirmativo, o processo interativo com o arquiteto é encerrado e o processo evolutivo continua sem mais interações, até se atingir

o número máximo de gerações estipulado inicialmente, quando então o processo evolutivo também é encerrado.

A Figura - 5.2 demonstra o fluxograma para o processo interativo em seu modelo inicial. O modelo inicial foi feito no intuito de se mostrar a viabilidade do MIP na abordagem MOA4PLA e ferramenta OPLA-Tool. Nesta figura, a parte destacada em "vermelho"refere-se às modificações feitas no algoritmo de otimização (NSGA-II) para permitir a interação com o arquiteto durante o processo de otimização.

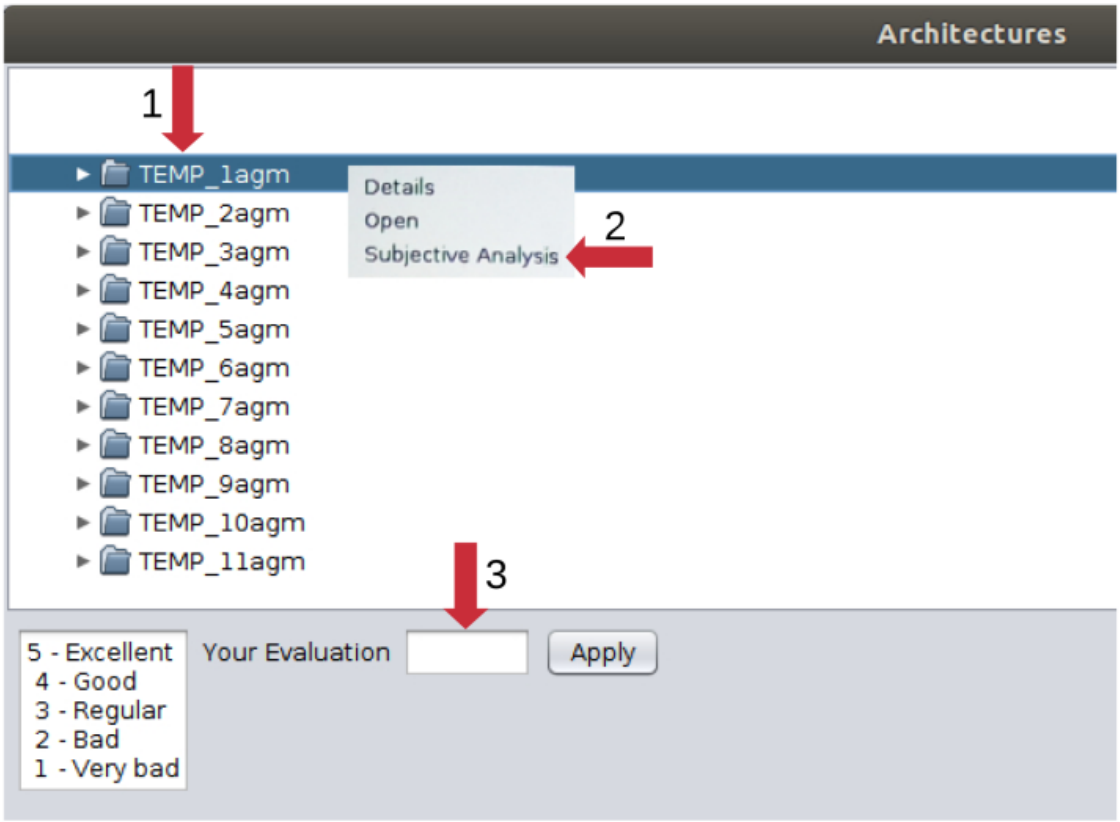
Figura 5.2: Fluxograma - Proposta inicial p/ Interação na MOA4PLA



Fonte: Autoria própria (Adaptado de BINDEWALD et al., 2019).

Para a avaliação do modelo inicial, uma nova interface foi desenvolvida na ferramenta OPLA-Tool. Após um número definido pelo arquiteto para o início do processo interativo (quantidade de gerações até a primeira interação), a tela da Figura - 5.3 aparece para o arquiteto, apresentando a lista de soluções geradas para aquela geração em questão (seta vermelha número 1). É importante destacar que a avaliação de cada solução (arquitetura) é independente e não depende da ordem de sua geração. Ao clicar com o botão direito do mouse nas soluções, são disponibilizadas ao arquiteto três opções: *"Details"*, *"Open"* e *"Subjective Analysis"* (seta vermelha número 2). Em *"Details"*, algumas informações específicas da solução são exibidas para o arquiteto. A opção *"Open"* abre o arquivo .UML [1] na ferramenta Papyrus[2] para que o arquiteto possa avaliar a solução. A última opção, *"Subjective Analysis"*, mostra um campo na parte inferior da tela onde o arquiteto pode fazer a sua avaliação, com um valor no intervalo de 1 a 5 (seta vermelha número 3).

Figura 5.3: Tela para o processo interativo na OPLA-Tool



Fonte: Autoria própria.

[1] Arquivo contendo o diagrama de classes da solução (alternativa de ALPS) gerada em tempo de execução pela OPLA-Tool.

[2] *Plugin* instalado no ambiente de desenvolvimento do software Eclipse (*https://www.eclipse.org/downloads/*) utilizado para a visualização das ALPS geradas em tempo de execução.

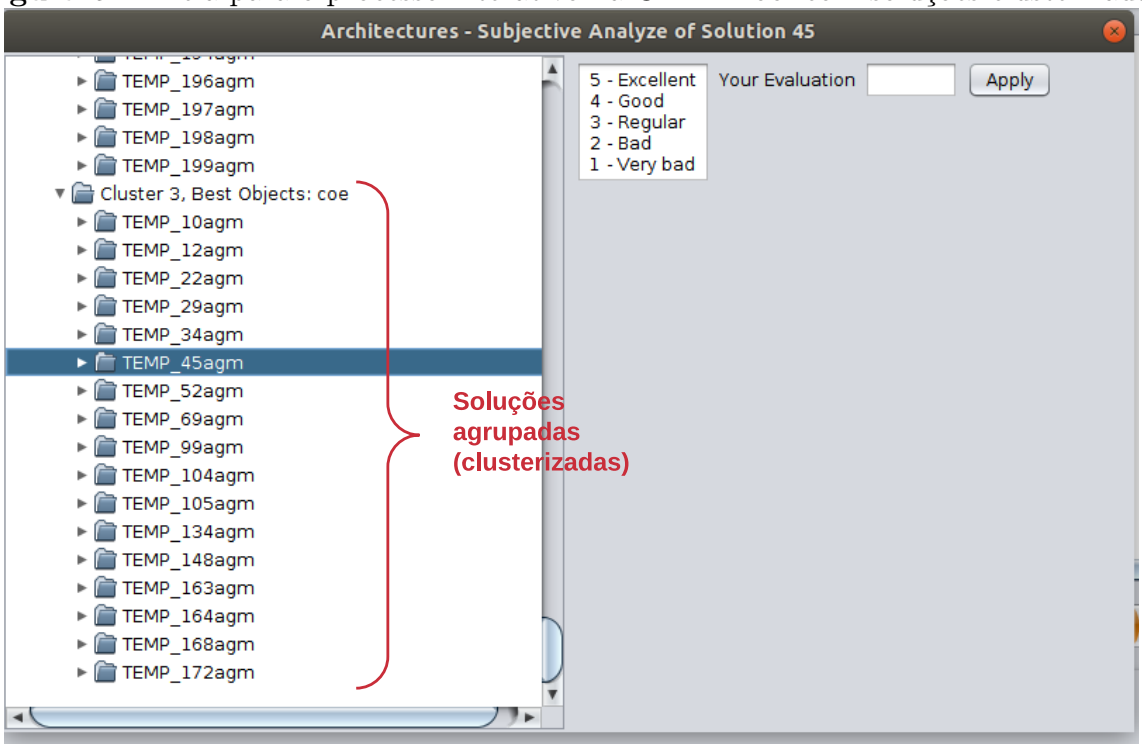
Os resultados obtidos experimentalmente para o modelo interativo inicial (apresentados na Seção 6.2 e publicados em Bindewald et al., 2019), demonstraram ser promissores, além de trazer *insights* de como o modelo poderia ser melhorado com a inclusão de algoritmos de AM.

5.3 Estratégias propostas para a formação do conjunto de dados de treinamento do modelo de AM

A partir de experimento realizado (BINDEWALD, et al., 2019) foi notado que a cada momento de interação, as soluções geradas eram muito semelhantes entre si, de modo que, avaliar todas as soluções em cada momento de interação era um trabalho desnecessário.

Em Freire et al. (2019), foi visto que era possível agrupar soluções (arquiteturas) por semelhança entre as mesmas, utilizando-se o algoritmo K-means para este fim. Baseando-se neste trabalho, o algoritmo K-means foi introduzido na ferramenta OPLA-Tool para que fosse possível uma diminuição na quantidade de soluções a serem avaliadas em cada momento de interação. Assim, foi possível agrupar soluções, utilizando-se como parâmetros para a formação dos *clusters* as funções-objetivo selecionadas e também mais um *cluster* contendo as soluções com o melhor *trade-off*. Desta forma, o número de *clusters* formados será igual ao número de funções-objetivo selecionadas e mais um *cluster* contendo as soluções com melhor *trade-off*. A Figura - 5.4 demonstra um exemplo de clusterização para o objetivo "coe".

Figura 5.4: Tela para o processo interativo na OPLA-Tool com soluções clusterizadas



Fonte: Autoria própria.

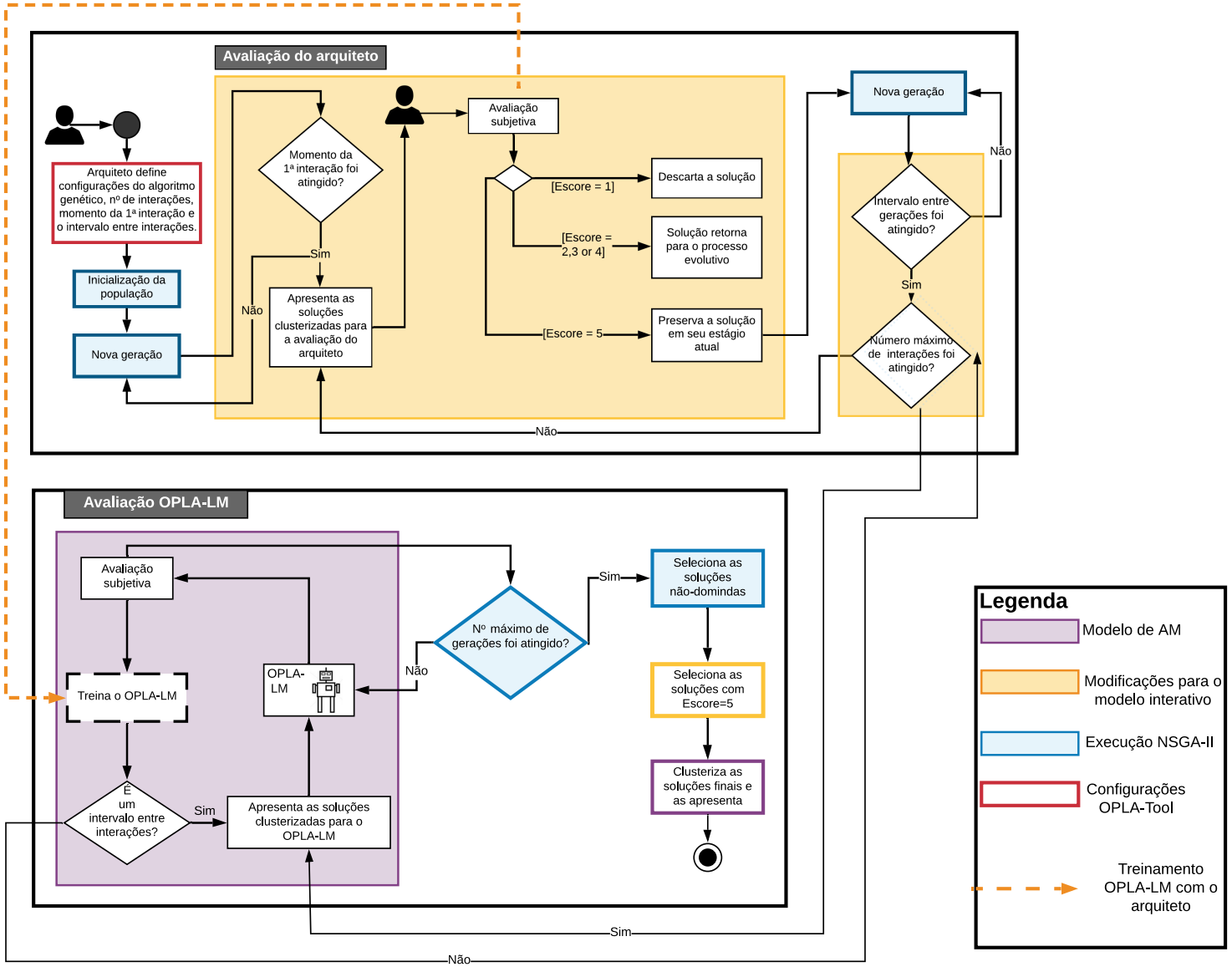
O agrupamento de soluções em *clusters* possibilita que o arquiteto possa escolher uma solução de cada *cluster* para avaliação em vez de avaliar todas as soluções geradas a cada momento de interação, colaborando para o tratamento da fadiga. Contudo, caso o número de funções-objetivo seja alto, a quantidade de soluções a serem avaliadas pode também ainda ser alta. Desta forma, faz-se necessário também um modelo de AM que possa aprender a partir da experiência do arquiteto e, em algum momento, substituí-lo no processo de avaliação de ALPS.

O modelo de AM utiliza o algoritmo K-means para o agrupamento de soluções em *clusters* e o algoritmo MLP para aprender as preferências do arquiteto e predizer as avaliações do mesmo. A Figura - 5.5 é uma adaptação da Figura - 5.2 e demonstra o processo interativo de avaliação de ALPS juntamente com o modelo de AM. O processo de avaliação de ALPS é o mesmo descrito na Seção 5.2. Contudo, há um novo parâmetro a ser colocado nas configurações de execução, o intervalo entre interações. Este intervalo foi necessário pois foi percebido experimentalmente que, entre uma geração e outra, o avaliador não conseguia notar diferenças significativas entre as ALPS. Desta forma, possibilitando um espaçamento entre uma interação e outra, o processo de otimização poderia evoluir por algumas gerações antes de solicitar a avaliação do arquiteto novamente.

Com a inserção de um intervalo entre interações, foi necessária também uma nova verificação pelo sistema durante o processo de otimização. A partir do primeiro momento de interação, a cada nova geração de indivíduos (nova geração de soluções alternativas de ALPS) o sistema verifica se o intervalo entre interações foi atingido, ou seja, se a quantidade de gerações entre uma interação e outra foi atingida. Em caso negativo, o processo evolutivo continua até uma nova geração de soluções. Quando o intervalo entre interações é atingido, o sistema verifica se o número máximo de interações foi atingido. Em caso negativo, as soluções são apresentadas para a avaliação do arquiteto. Em caso afirmativo, o processo de avaliação de ALPS por parte do arquiteto é encerrado. Neste momento o processo de avaliação de ALPS passa para o OPLA-LM (OPLA-Learning Model - Figura - 5.12), que foi treinado durante o processo de interação com o arquiteto. A cada momento de interação, a avaliação das soluções feita pelo arquiteto gera dados para o conjunto de treinamento do OPLA-LM. Importante mencionar que o OPLA-LM é treinado paralelamente ao processo de interação com o arquiteto e, entre uma interação e outra, o processo de treinamento do OPLA-LM ocorre com os dados de treinamento já existentes.

Após o processo de avaliação ter sido encerrado com o arquiteto e iniciado com o OPLA-LM, a cada nova geração de soluções, o sistema verifica se o máximo número de gerações foi atingido. Em caso negativo, o processo de avaliação (pelo OPLA-LM) é executado. Em caso afirmativo, o processo de otimização é encerrado.

Figura 5.5: Fluxograma - Proposta p/ Interação na abordagem MOA4PLA e ferramenta OPLA-Tool com algoritmos de AM



Fonte: Autoria própria.

A Figura - 5.6 apresenta o algoritmo NSGA-II com as modificações feitas para a introdução do processo interativo. Para os dados de entrada do algoritmo (linha 1), três novos parâmetros foram adicionados, sendo estes: *int_ 1*, *intervalo_ int* e *max_ inter*, respectivamente para o "primeiro momento de interação", o "intervalo entre interações"e o "número máximo de interações".

Figura 5.6: Modificação do NSGA-II com a introdução do processo interativo

Algoritmo 1 NSGA-II modificado

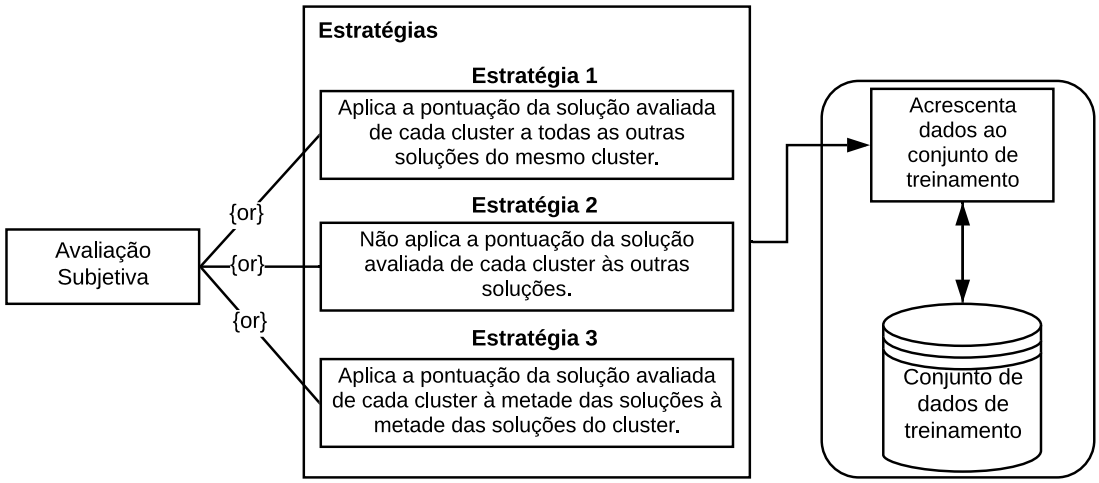
1: **Entrada:** *N', g, $f_k(X)$, int_*1, *intervalo_ inter, max_ inter*
2: Inicializar população *P*
3: Gerar população aleatória - Tamanho *N'*
4: Avaliar valores dos objetivos
5: Atribuir *rank* baseado na dominância de Pareto - Ordenação
6: Gerar população filho
7: Seleção por torneio binário
8: Recombinação e Mutação
9: **para** $i = 1$ **até** g **faça**
10: **se** $i == int\_1$ **então**
11: `Arquiteto avalia a solução`
12: **fim se**
13: **para** cada *Pai* e *Filho* na População **faça**
14: Atribuir *rank* baseado na dominância de Pareto - Ordenação
15: Gerar fronteiras não dominadas
16: Determinar a distância de multidão para cada solução das fronteiras e percorrer todas as fronteiras adicionando para a próxima geração do primeiro ao *N'* indivíduo
17: **fim para**
18: Selecionar indivíduos das melhores fronteiras e com maior distância de multidão
19: Gerar população filho
20: Seleção por torneio binário
21: Cruzamento e Mutação
22: **se** *intervalo_ inter* foi atingido **então**
23: **se** *max_ inter* foi atingido **então**
24: `OPLA-LM avalia as soluções`
25: **senão**
26: `Arquiteto avalia a solução`
27: **fim se**
28: **fim se**
29: **fim para**

Fonte: Autoria própria (Adaptado de Coello et al, 2007).

Para a formação do conjunto de dados de treinamento do algoritmo MLP, em um primeiro momento, seriam utilizados apenas os dados referentes às ALPS avaliadas pelo arquiteto. Contudo, foi constatado que o arquiteto poderia avaliar apenas poucas ALPS, de modo que não seriam gerados dados suficientes para o treinamento do modelo de AM. Desta forma, três diferentes estratégias foram elaboradas. Estas estratégias partiram da premissa de se utilizar todas as ALPS geradas a cada momento de interação para compor o conjunto de dados de treinamento do algoritmo MLP. Assim, ambas estratégias possuem a mesma quantidade de dados, mas se diferenciam em como estes dados são preparados.

A Figura - 5.7 sumariza as estratégias elaboradas para compor o conjunto de dados de treinamento do algoritmo MLP, sendo as mesmas explicadas na sequência.

Figura 5.7: Estratégias para a formação do conjunto de treinamento



Fonte: Autoria própria.

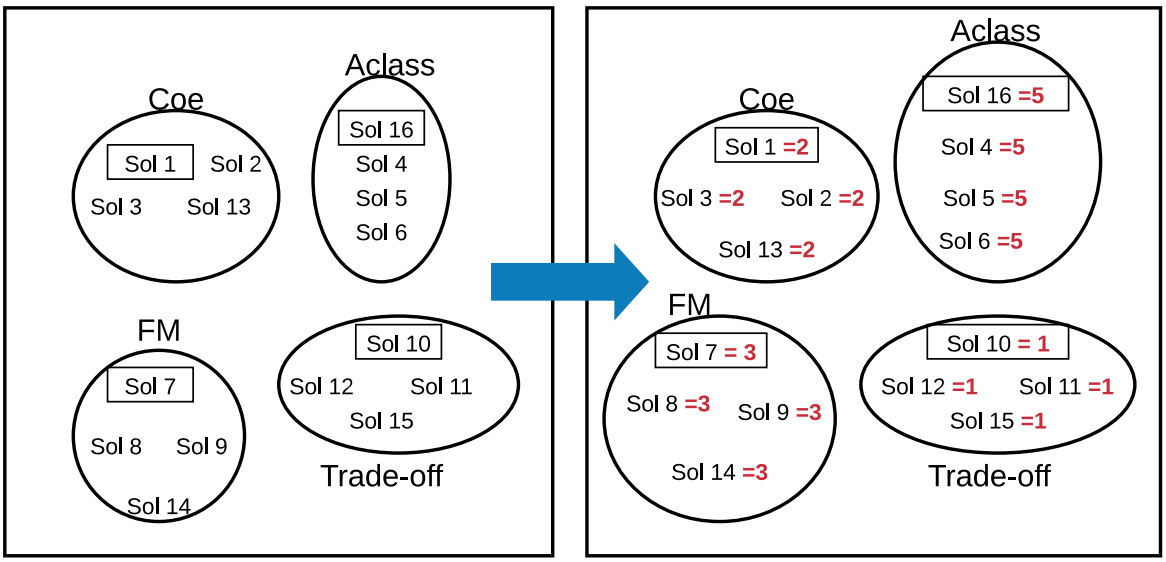
Estratégia 1 - Generaliza a pontuação da solução avaliada de cada *cluster* para todas as soluções do *cluster*.

Essa estratégia realiza a generalização da pontuação no *cluster*. Assim, a pontuação atribuída pelo arquiteto a uma solução é aplicada em todas as outras soluções presentes no mesmo *cluster*. Se mais de uma solução foi avaliada, a média das pontuações atribuídas é calculada e aplicada às outras soluções. No caso de números fracionados, o valor médio é arredondado e depois atribuído às outras soluções.

A Figura - 5.8 demonstra um exemplo da aplicação da Estratégia 1. Este é um exemplo de uma execução da OPLA-Tool com 3 objetivos selecionados (COE, AClass e FM). Desta forma, são gerados 3 *clusters* referentes aos objetivos selecionados e mais um *cluster* que apresenta as soluções com melhor *trade-off*. Neste exemplo, foram selecionadas 1 solução

de cada *cluster* para serem avaliadas, sendo elas: "Sol 1", "Sol 16", "Sol 7"e "Sol 10", respectivamente para os *clusters* "COE", "AClass", "FM"e "Trade-off".

Figura 5.8: Exemplo - Aplicação da Estratégia 1



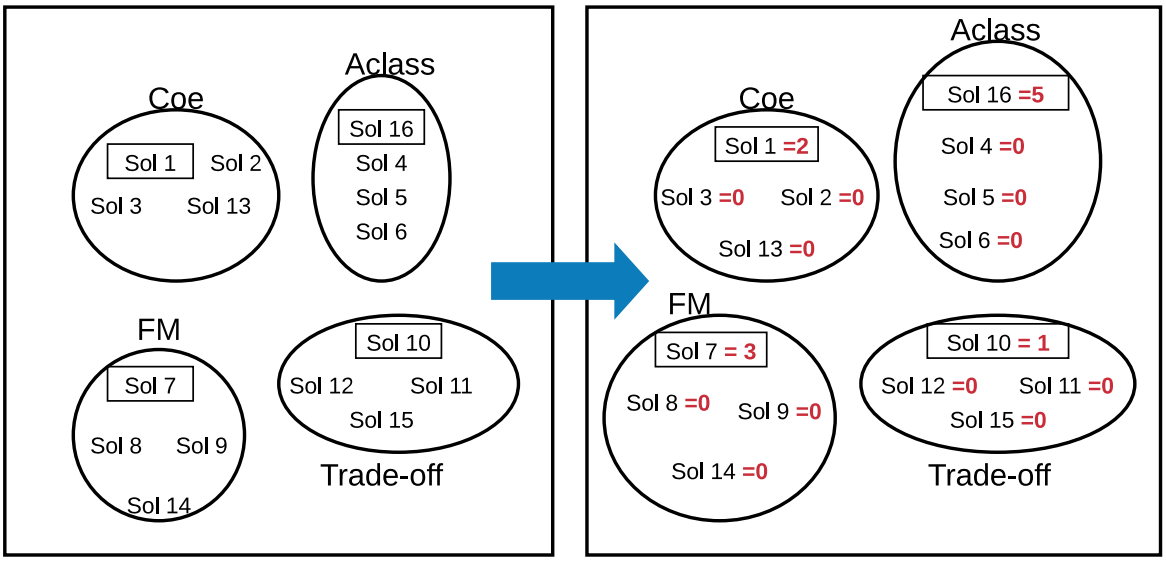
Fonte: Autoria própria.

Estratégia 2 - Não aplica a pontuação da solução avaliada de cada *cluster* às outras soluções.

Um possível problema verificado na primeira estratégia é que o algoritmo de AM não aprende como "não avaliar" algumas soluções, ou seja, o arquiteto vai avaliar uma (ou poucas) soluções por *cluster*, de modo que o modelo de AM deveria aprender as soluções que ele não deveria avaliar. Nesse caso, soluções não avaliadas são mantidas sem avaliação (recebendo um `escore` = 0).

A Figura - 5.9 demonstra um exemplo da aplicação da Estratégia 2. Este é um exemplo de uma execução da OPLA-Tool com 3 objetivos selecionados (COE, AClass e FM). Desta forma, são gerados 3 *clusters* referentes aos objetivos selecionados e mais um *cluster* que apresenta as soluções com melhor *trade-off*. Neste exemplo, foram selecionadas 1 solução de cada *cluster* para serem avaliadas, sendo elas: "Sol 1", "Sol 16", "Sol 7"e "Sol 10", respectivamente para os *clusters* "COE", "AClass", "FM"e "Trade-off".

Figura 5.9: Exemplo - Aplicação da Estratégia 2



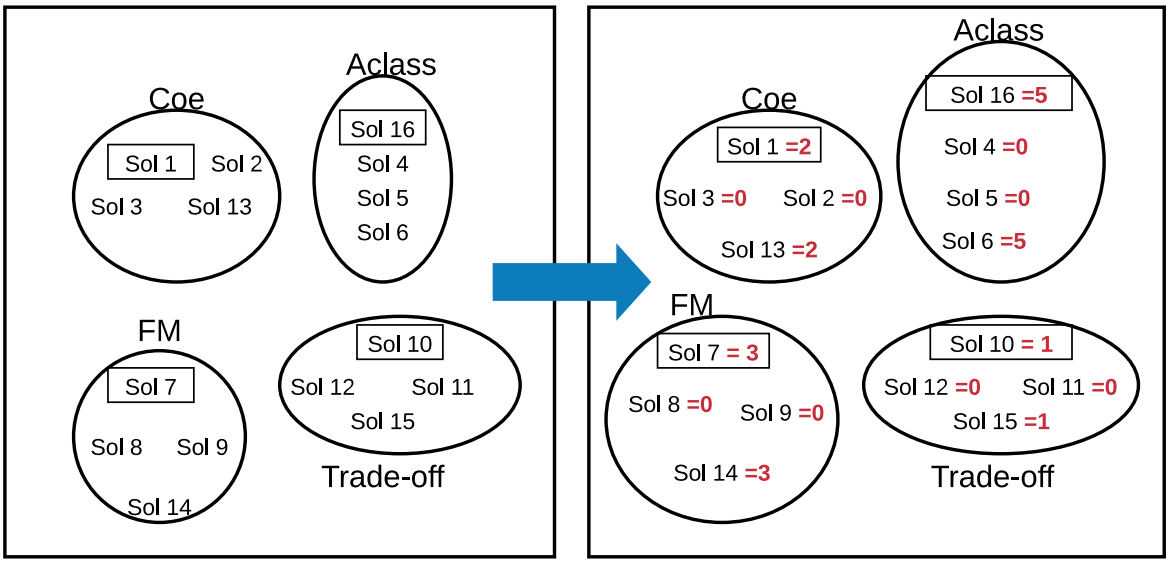
Fonte: Autoria própria.

Estratégia 3 - Aplica a pontuação da solução avaliada de cada *cluster* à metade das soluções.

Esta estratégia busca ser um meio termo entre as estratégias anteriores. Neste caso, a metade das soluções não avaliadas no *cluster* é generalizada de acordo com as soluções avaliadas e a outra metade das soluções é mantida com `escore` = 0.

A Figura - 5.10 demonstra um exemplo da aplicação da Estratégia 3. Este é um exemplo de uma execução da OPLA-Tool com 3 objetivos selecionados (COE, AClass e FM). Desta forma, são gerados 3 *clusters* referentes aos objetivos selecionados e mais um *cluster* que apresenta as soluções com melhor *trade-off*. Neste exemplo, foram selecionadas 1 solução de cada *cluster* para serem avaliadas, sendo elas: "Sol 1", "Sol 16", "Sol 7"e "Sol 10", respectivamente para os *clusters* "COE", "AClass", "FM"e "Trade-off".

Figura 5.10: Exemplo - Aplicação da Estratégia 3



Fonte: Autoria própria.

A Figura - 5.11 apresenta a rede MLP para o modelo de AM. Como entrada para a rede, há 10 atributos, sendo estes: Os três objetivos selecionados (FM, AClass, COE), juntamente com as propriedades arquiteturais seguintes:

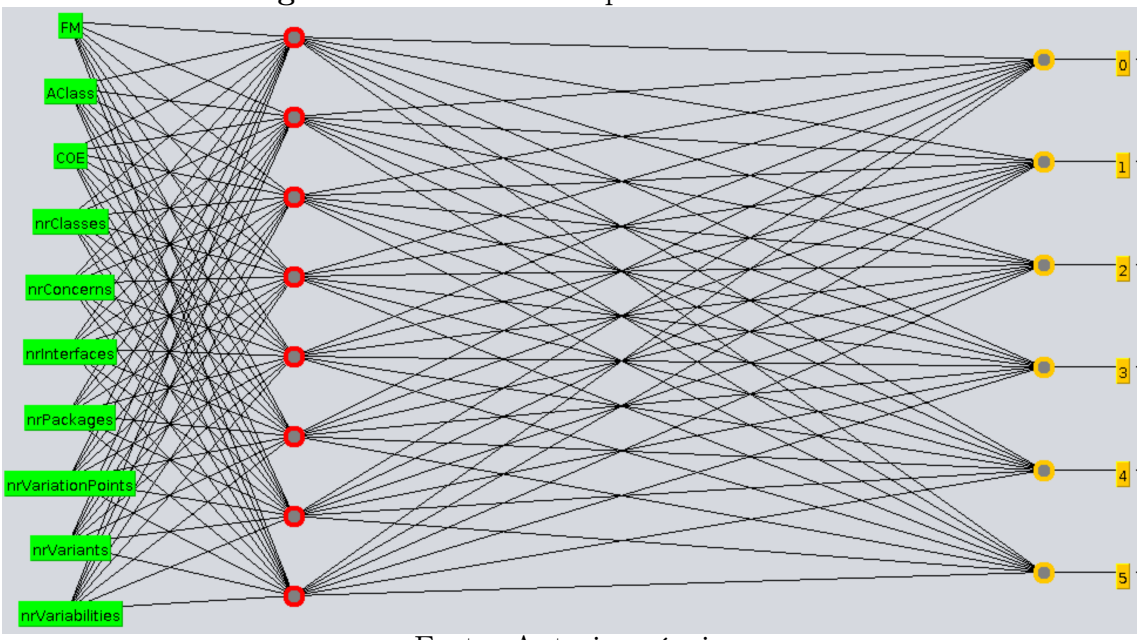
- **nrClasses:** (número de classes): Quantidade de classes da arquitetura dada como entrada para o processo de otimização;
- **nrConcerns** (número de *concerns*): Correspondem a requisitos, funcionais ou não, que devem estar presentes em um sistema de software;
- **nrInterfaces** (número de interfaces): Definem os pontos de acesso ao componente - propriedades visíveis externamente;
- **nrPackages:** (número de pacotes): Quantidade de pacotes da arquitetura dada como entrada para o processo de otimização;
- **nrVariabilities:** (número de variabilidades): Corresponde a um aspecto funcional em um elemento de software base que possui variação. Define um conjunto de possíveis variantes, o mecanismo de variabilidade para instanciar as variantes e o tempo de ativação das variantes;
- **nrVariationPoints:** (número de pontos de variação): Um ponto de variação é um lugar especıfico em um artefato da LPS ao qual uma decisão de projeto está associada. Cada ponto de variação está associado a um conjunto de variantes;

- **nrVariants:** (número de variantes): Corresponde a uma opção do conjunto de possíveis instâncias de variação que um ponto de variabilidade poderá originar;

A escolha por utilizar as propriedades arquiteturais descritas acima, deveu-se ao fato de que a utilização apenas dos valores de *fitness* das funções-objetivo selecionadas poderia não ser suficiente para que o modelo de AM capturasse as preferências do arquiteto. Com a realização do experimento com o modelo interativo inicial (Exp 01 - Seção 6.2) foi percebido que o arquiteto avaliara soluções que continham exatamente os mesmos valores de *fitness* diferentemente. Desta forma, não seria possível para um modelo de AM aprender como deveria avaliar as soluções, visto que, do ponto de vista matemático (utilizando-se apenas os valores de *fitness*) estas soluções seriam iguais. O número de camadas ocultas da rede foi definido a partir de testes empíricos realizados. Ao aumentar o número de camadas, foi percebido que não havia ganho significativo nas predições realizadas pelo algoritmo e o tempo de processamento era maior. Desta forma, optou-se por utilizar apenas uma camada oculta.

Como saídas das rede MLP, há 6 diferentes saídas, sendo que, os valores de 1 a 5 representam a predição das avaliações feitas pelo arquiteto e a saída "0"(zero) representa o caso de predição para arquiteturas que não deveriam ser avaliadas.

Figura 5.11: Rede MLP para o modelo de AM

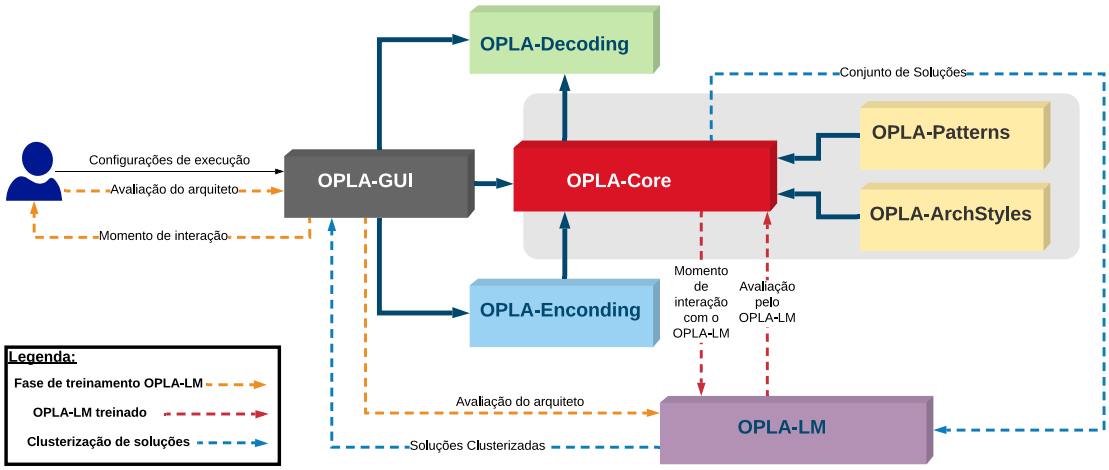


Fonte: Autoria própria.

A Figura - 5.12 apresenta a nova configuração da ferramenta OPLA-Tool com a inclusão do modelo de AM (módulo OPLA-LM). Na Figura podem ser vistos três novos momentos, indicados pelas setas pontilhadas. O módulo OPLA-LM recebe o conjunto de

soluções geradas pelo OPLA-Core para cada momento de interação e as clusteriza (seta pontilhada em azul). Na sequência, o conjunto de soluções clusterizadas é apresentado ao arquiteto para a avaliação do mesmo (seta pontilhada em amarelo). Por fim, com o modelo de AM treinado, o mesmo passa a substituir o arquiteto no processo de avaliação de ALPS (seta pontilhada em vermelho).

Figura 5.12: Inclusão do módulo OPLA-LM



Fonte: Autoria própria (Adaptado de FÉDERLE et al., 2015).

5.4 Considerações finais

Este capítulo descreveu as modificações que foram necessárias para permitir o processo interativo com o arquiteto durante o processo de otimização realizado pela abordagem MOA4PLA e, consequentemente, pela ferramenta OPLA-Tool. Em um primeiro momento, foi proposto um modelo inicial para o processo interativo no intuito de ver a viabilidade da incorporação das preferências do arquiteto. Os resultados (apresentados na Seção 6.2) demonstraram ser promissores, além de possibilitar propor melhorias para o processo interativo e para o tratamento da fadiga. Neste sentido, um modelo de AM foi proposto com a inclusão de dois algoritmos na OPLA-Tool, K-means e MLP. O algoritmo K-means foi introduzido no intuito de se agrupar soluções em *clusters* de soluções semelhantes a fim de que o arquiteto não necessite avaliar todas as soluções, o que ajuda no tratamento da fadiga. Além disso, foi introduzido também o algoritmo MLP para aprender sobre o comportamento do arquiteto e, posteriormente, substituí-lo no processo de avaliação de ALPS.

O capítulo a seguir descreve e discute os resultados obtidos com os experimentos realizados para a avaliação do MIP.

6

Resultados e discussões

6.1 Considerações iniciais

Este capítulo demonstra e discute os resultados alcançados por cada um dos três experimentos realizados. Em um primeiro momento, Experimento 1 (Seção 6.2), foi realizado um experimento com a participação de um arquiteto para avaliar a viabilidade do MIP e também para se ter um maior conhecimento sobre o possível problema da fadiga que poderia ocorrer durante o processo de avaliação de ALPS. Na Seção 6.3, são apresentados os resultados obtidos experimentalmente (Exp 02) por cada uma das estratégias propostas para compor o conjunto de dados de treinamento do modelo de AM. A partir dos resultados do Exp 02 foi possível identificar qual das três estratégias é a mais adequada para compor o modelo de AM final. Com a identificação da melhor Estratégia, foi possível realizar outro experimento (Exp 03 - Seção 6.4), cujo objetivo foi avaliar o MIP na MOA4PLA e OPLA-Tool como um todo. Para isso, este experimento contou com a participação de arquitetos de software para avaliar as soluções (arquiteturas) geradas em tempo de processamento. Assim, foi possível identificar se o processo interativo proporciona a geração de soluções mais adequadas a cada arquiteto e, além disso, comprovar a eficácia do modelo de AM, se o mesmo foi hábil em aprender sobre o comportamento do arquiteto e conseguiu substituí-lo com eficiência no processo interativo.

6.2 Experimento 1 (Exp 01)

Esta seção apresenta e discute os resultados obtidos após a execução do Exp 01, que ocorreu em duas etapas. Em um primeiro momento, foi realizada a execução da ferramenta

OPLA-Tool em seu modelo atual (Exec-MA) e, posteriormente, uma nova execução foi realizada com o MIP (Exec-MIP). Neste caso, contando com a participação de um arquiteto de software para a avaliação de ALPS geradas no processo de otimização. Com os resultados de ambas execuções, foi possível avaliar a viabilidade do MIP. Para isto, foram realizadas análises quanti-qualitativas.

6.2.1 Método de análise quantitativa

O método quantitativo envolveu a execução da ferramenta OPLA-Tool duas vezes. A primeira, Exec-MA (MA - modelo atual), foi realizada com a ferramenta em seu estágio atual de desenvolvimento, no qual o arquiteto não tem a opção de interagir com a ferramenta durante o processo evolutivo. A segunda execução, Exec-MIP (MIP - Modelo Interativo Proposto), foi realizada com o modelo proposto neste trabalho, que permite a interação do arquiteto para a avaliação das ALPS geradas durante o processo evolutivo.

Para a Exec-MIP foram necessárias tarefas adicionais, uma vez que, neste experimento, o arquiteto participa do processo interativo. Assim, as atividades para sua seleção e treinamento foram realizadas conforme descrito abaixo.

1. **Seleção do especialista:** A escolha do arquiteto para participar do experimento levou em consideração o nível educacional e o conhecimento de projeto de ALPS. Assim, o arquiteto escolhido é estudante de doutorado e possui alto conhecimento em análise de projeto de ALPS.

2. **Treinamento do especialista:** Para a execução do experimento, o arquiteto recebeu treinamento para entender como o processo interativo funciona na OPLA-Tool e como avaliar a ALPS;

Nas duas execuções, os valores dos parâmetros de configuração utilizados foram:

- número de execução do algoritmo: 1;
- número de gerações: 20;
- população inicial: 12 indivíduos;
- operador de mutação com probabilidade = 0,9.

Esses valores foram definidos considerando a quantidade de tempo necessária para executar a Exec-MIP e o cansaço (fadiga) do arquiteto, uma vez que avaliar o projeto de ALPS é uma tarefa que exige alto esforço intelectual, e o arquiteto avaliaria todas

as soluções geradas em cada interação. Além disso, para Exec-MIP, foram definidos também o número de interações e o momento da primeira interação. Os novos parâmetros utilizados foram:

- número de interações = 5;
- momento da primeira interação = 5.

Para fazer uma comparação quantitativa entre os resultados obtidos por Exec-MA e Exec-MIP, vários dados foram analisados, tais como:

- **Número de soluções geradas:** Foi analisado se Exec-MIP poderia produzir mais e diferentes soluções em relação a Exec-MA;
- **Valores de *fitness* das soluções:** Foram utilizados para o cálculo do *Fitness* Ideal, para a análise do impacto do processo interativo no cálculo do ED das soluções geradas e também para o cálculo das soluções que compõem o PF_{true};
- **Número de pacotes, classes, dependências, associações e relacionamentos:** Foram utilizados para analisar em detalhes as diferenças entre as soluções geradas em ambas execuções.

Para a análise da Exec-MIP, foi coletada também a pontuação atribuída a cada projeto de ALPS pelo arquiteto.

6.2.2 Método de análise qualitativa

Para complementar a análise quantitativa, foi realizado com o arquiteto participante da Exec-MIP um experimento qualitativo. Este experimento teve como objetivo esclarecer quais aspectos um arquiteto considera importantes ao avaliar uma ALPS e também se as soluções geradas pelo MIP atendiam às suas necessidades. Para realizar este experimento, um questionário foi preparado e aplicado ao arquiteto, contendo os seguintes questionamentos:

1. Apesar de ser uma avaliação subjetiva, existe algum critério específico usado para avaliar a ALPS?
2. Na sua opinião, o Modelo Interativo Proposto pode trazer soluções mais adequadas para cada arquiteto em particular?

3. Nos trabalhos relacionados à interação em SBSE, há uma grande preocupação com o problema da fadiga, causado pelo elevado número de interações do DM. Com base na sua experiência e participação neste experimento, você poderia dizer um número de adequado de ALPS a serem avaliadas antes que o problema da fadiga ocorra?

6.2.3 Resultados quantitativos

Como mencionado na Seção 6.2.1, vários dados foram coletados das duas execuções feitas (Exec-MA e Exec-MIP) para realizar uma análise da eficácia do modelo proposto.

A Tabela - 6.1 mostra, na parte superior, o valor original de *fitness* da ALPS AGM e o valor ideal de *fitness* obtido a partir das soluções geradas em ambas execuções. Na parte inferior da Tabela - 6.1 são mostrados as quantidades originais de algumas propriedades arquiteturais da ALPS AGM, tais como: classes, pacotes, dependências, associações e relacionamentos.

Tabela 6.1: Informações da ALPS AGM original

PLA	***Fitness* Original** (ACLASS, FM, COE)	**Fitness ideal** (ACLASS, FM, COE)
AGM	30.0 , 758.0 , 26.0	12.0 , 674.0 , 25.0

PLA	**Pacotes**	**Classes**	**Dependências**	**Associações**	**Relacionamentos**
AGM	9	30	14	7	47

Fonte: Autoria própria.

A Tabela - 6.2 mostra, na parte superior, os dados referentes aos valores de *fitness* e o cálculo do ED para cada uma das soluções de Exec-MIP. Na parte inferior são mostrados as quantidades de algumas propriedades arquiteturais para as soluções geradas pela mesma execução.

Tabela 6.2: Exec-MA - Resultados

Solução	***Fitness* da Solução** (ACLASS, FM, COE)	**ED**
Solução 1	23.0 , 719.0 , 27.0	46.4
Solução 2	28.0 , 752.0 , 26.0	79.6
Solução 3	16.0 , 694.0 , 29.0	20.8
Solução 4	12.0 , 676.0 , 30.0	5.4

Solução	**Pacotes**	**Classes**	**Dependências**	**Associações**	**Relacionamentos**
Solução 1	11	28	12	8	44
Solução 2	9	28	13	8	46
Solução 3	11	28	13	8	46
Solução 4	12	28	13	8	46

Fonte: Autoria própria.

A Tabela - 6.3 mostra, na parte superior, os dados referentes aos valores de *fitness* e o cálculo do ED para cada uma das soluções de Exec-MIP. Na parte inferior são mostrados as quantidades de algumas propriedades arquiteturais para as soluções geradas pela mesma execução.

Tabela 6.3: Exec-MIP - Resultados

Solução	***Fitness* da Solução** (ACLASS, FM, COE)	**ED**
Solução 1	24.0 , 718.0 , 27.0	45.7
Solução 2	23.0 , 712.0 , 28.0	39.7
Solução 3	16.0 , 692.0 , 29.0	18.9
Solução 4	12.0 , 674.0 , 30.0	5.0
Solução 5	34.0 , 793.0 , 25.0	121.0
Solução 6	28.0 , 747.0 , 26.0	74.7
Solução 7	18.0 , 716.0 , 28.0	42.5

Solução	**Pacotes**	**Classes**	**Dependências**	**Associações**	**Relacionamentos**
Solução 1	10	28	13	8	46
Solução 2	11	28	10	7	40
Solução 3	11	28	13	8	46
Solução 4	12	28	13	8	46
Solução 5	8	28	12	8	44
Solução 6	9	28	12	8	44
Solução 7	11	28	13	7	45

Fonte: Autoria própria.

A análise dos dados (Tabela - 6.2 e Tabela - 6.3), permite observar que Exec-MIP gerou mais soluções que a Exec-MA. Além disso, é possível observar que quase todas as soluções

(para os dois experimentos) têm os valores de *fitness* para ACLASS e FM minimizados, enquanto os valores de *fitness* para COE foram aumentados. A única exceção é a Solução 5 da Exec-MIP, onde os valores *fitness* de COE foram minimizados, enquanto os valores de ACLASS e FM foram aumentados.

A análise dos valores do valor de ED permite identificar algumas diferenças entre os dois experimentos. Os melhores valores de ED para as duas execuções estão destacados em vermelho (Tabela - 6.2 e Tabela - 6.3). É possível verificar que, considerando os três melhores valores de ED, dois deles estão no conjunto Exec-MIP (Soluções 3 e 4). Esse fato pode ser devido ao impacto da interação do arquiteto no processo evolutivo.

Foram analisados também os três objetivos otimizados individualmente. A avaliação dos valores obtidos para ACLASS mostra que os dois experimentos possuem soluções que obtiveram o valor 12 (o menor valor encontrado no conjunto de soluções) e, neste caso, não é possível afirmar que, em relação a esse objetivo , existem diferenças entre as execuções.

A análise dos valores de coesão (COE) mostra que nas soluções da Exec-MIP existe uma solução com o menor valor para esse objetivo (Solução 5 - COE = 25). Além disso, pode-se observar que o maior valor de COE em Exec-MIP não excede o maior valor de COE em Exec-MA.

O último objetivo analisado foi a modularização de características (FM). Em relação à análise dos valores de FM, é possível observar que a solução com o menor valor de FM foi obtida na Exec-MIP (Solução 4 - FM = 674).

Foi solicitado ao arquiteto que avaliasse cada solução com uma pontuação no intervalo de 1 a 5. A Tabela - 6.4 mostra as avaliações dadas pelo arquiteto para as ALPS geradas em tempo de execução na Exec-MIP. Inicialmente, Exec-MIP foi configurada para ter cinco interações com o arquiteto, porém, após três interações e 36 ALPS avaliadas (12 para cada interação), verificou-se que o arquiteto já apresentava fadiga. Assim, a execução do experimento continuou sem mais interações. Desta forma, as avaliações do arquiteto ocorreram nas 6ª, 7ª e 8ª gerações de soluções.

A análise dos Escores dadas pelo arquiteto às soluções de ALPS permite inferir que, para esse arquiteto em particular, outros aspectos são mais importantes que os valores de *fitness*, corroborando com a afirmação de Simons, Singer e White (2015), de que métricas não são suficientes para avaliar a qualidade das soluções geradas. Foi possível verificar no conjunto de soluções avaliadas pelo arquiteto que existem soluções com os mesmos valores de *fitness*, mas com Escores diferentes. Como exemplo, na Interação 1, as soluções TEMP-0 e TEMP-4 possuem os mesmos valores de *fitness*, mas suas avaliações são diferentes (1 e 4, respectivamente). Esta mesma situação ocorre também em outras interações.

Tabela 6.4: Escores dados pelo arquiteto em cada momento de interação

ALPS	Interação 1		Interação 2		Interação 3	
	Escore	*Fitness* da solução (ACLASS, FM, COE)	Escore	*Fitness* da solução (ACLASS, FM, COE)	Escore	*Fitness* da solução (ACLASS, FM, COE)
TEMP−0	1	28.0 , 749.0 , 26.0	3	21.0 , 743.0 , 30.0	3	21.0 , 743.0 , 30.0
TEMP−1	3	21.0 , 743.0 , 26.0	2	28.0 , 749.0 , 26.0	3	28.0 , 749.0 , 26.0
TEMP−2	1	28.0 , 749.0 , 26.0	2	28.0 , 749.0 , 26.0	3	28.0 , 749.0 , 26.0
TEMP−3	2	28.0 , 757.0 , 27.0	3	21.0 , 743.0 , 30.0	3	28.0 , 749.0 , 26.0
TEMP−4	4	28.0 , 749.0 , 26.0	3	21.0 , 743.0 , 30.0	2	34.0 , 793.0 , 25.0
TEMP−5	3	28.0 , 749.0 , 26.0	4	28.0 , 749.0 , 26.0	4	28.0 , 755.0 , 27.0
TEMP−6	2	28.0 , 760.0 , 27.0	4	28.0 , 749.0 , 26.0	3	28.0 , 749.0 , 26.0
TEMP−7	4	28.0 , 749.0 , 26.0	1	28.0 , 749.0 , 26.0	4	28.0 , 753.0 , 27.0
TEMP−8	2	28.0 , 749.0 , 26.0	3	21.0 , 743.0 , 30.0	2	28.0 , 749.0 , 26.0
TEMP−9	3	28.0 , 753.0 , 27.0	3	28.0 , 749.0 , 26.0	2	28.0 , 749.0 , 26.0
TEMP−10	1	24.0 , 719.0 , 27.0	3	28.0 , 749.0 , 26.0	2	28.0 , 749.0 , 26.0
TEMP−11	1	28.0 , 749.0 , 26.0	3	28.0 , 749.0 , 26.0	3	28.0 , 749.0 , 26.0
Mediana	**2**		**3**		**3**	

Fonte: Autoria própria.

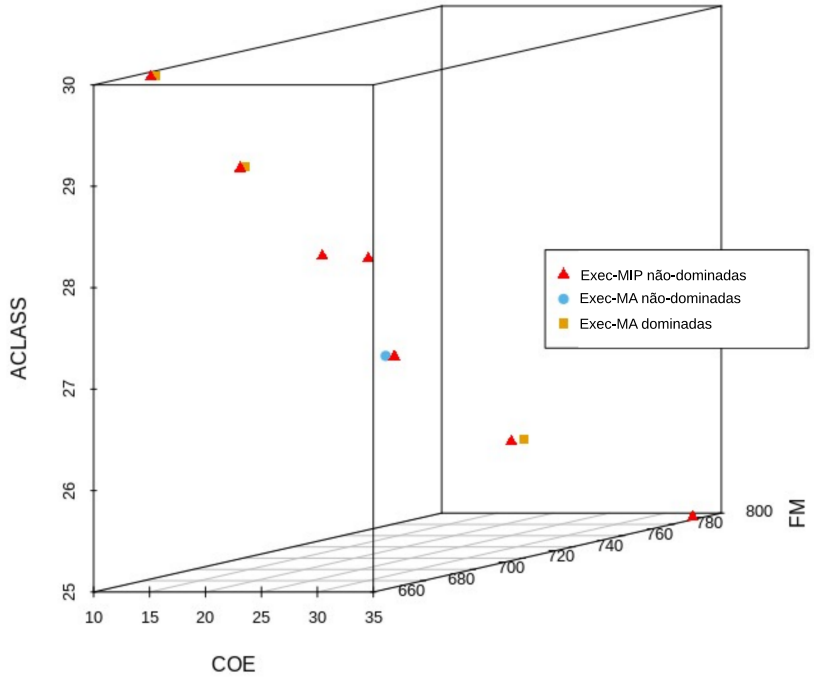
Foi possível verificar ainda que, as soluções com menor avaliação (`escore` = 1) tenderam a diminuir com o passar das interações. Do total de soluções com `escore` = 1 (5 soluções), quatro ocorreram na Interação 1, apenas uma na Interação 2 e nenhuma na Interação 3. Isso pode indicar que a avaliação do arquiteto teve influência sobre a execução, uma vez que o número de soluções consideradas muito ruins diminuiu após o início processo interativo.

Essa melhoria nas avaliações das soluções também pode ser observada pelos valores de medianas dos três conjuntos de soluções avaliados pelo arquiteto. Na Interação 1, o valor da mediana é igual a 2, enquanto nas interações a seguir, esse valor muda para 3. Essa alteração no valor da mediana pode indicar uma melhoria no conjunto de soluções como um todo.

Outro aspecto relevante está relacionado às soluções encontradas por cada execução que compõe o PF_{true}. Para compor o PF_{true}, foram utilizadas todas as soluções geradas nas duas execuções (quatro soluções da Exec-MA e sete soluções da Exec-MIP). Do conjunto total de soluções, oito soluções não-dominadas compõem o PF_{true}. As soluções que compõem o PF_{true} estão destacados em azul (Tabela - 6.2 e Tabela - 6.3).

A Figura - 6.1 demonstra graficamente o PF_{true} com todas as soluções geradas por ambas execuções. É possível verificar no gráfico que apenas uma solução da Exec-MA (símbolo em azul) faz parte do conjunto PF_{true}.

Figura 6.1: PF_{true} com as soluçoes de Exec-MIP e Exec-MA



Fonte: Autoria própria.

Considerando as soluções que compõem o PF_{true}, sete soluções foram geradas a partir da Exec-MIP e apenas uma solução da Exec-MA. Assim, existem evidências de que a interação do arquiteto durante o processo de otimização permitiu obter soluções que melhor exploram o espaço de busca. De uma forma geral, puderam ser identificadas diferenças entre os conjuntos de soluções obtidas por cada execução. Em Exec-MIP, o aspecto que mais chama atenção é que há um número maior de soluções geradas. Além disso, pode-se enfatizar que a solução com o melhor ED e os melhores valores de cada objetivo individualmente são encontrados no conjunto de soluções da Exec-MIP. Esses fatos apontam evidências de que a interação com o arquiteto durante o processo de otimização permitiu o surgimento de soluções com diferentes valores de *fitness* e, em alguns casos, melhores do que os encontrados no modelo atual.

6.2.4 Resultados qualitativos

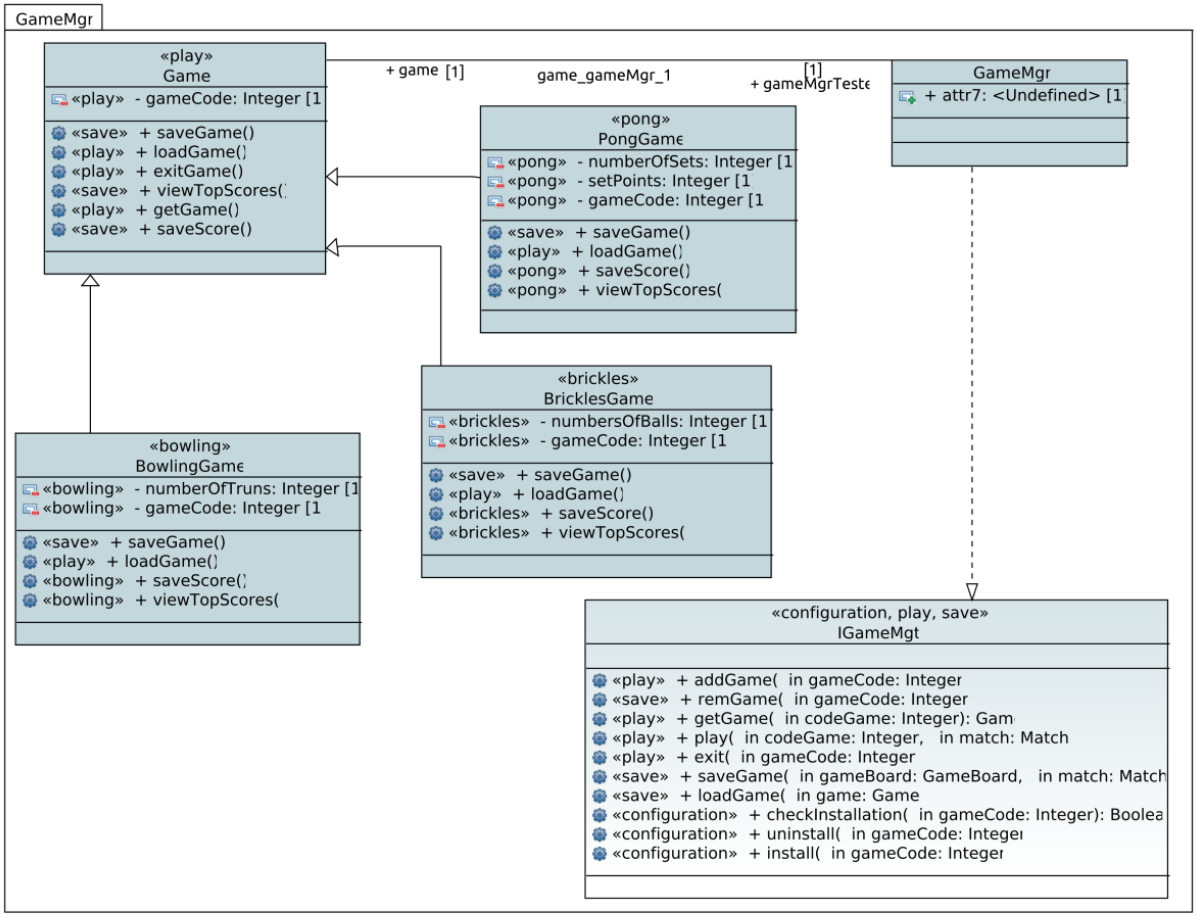
Para identificar o que um arquiteto considera importante ao avaliar uma ALPS e também explorar com melhor clareza o problema da fadiga, ao final da Exec-MIP, algumas perguntas (apresentadas na Subseção 6.2.2) foram feitas ao arquiteto. Suas respostas ajudaram a identificar o que este arquiteto em particular considera importante ao avaliar

uma ALPS. Neste caso, este arquiteto considerou importante ao avaliar uma ALPS: número de pacotes e classes próximos à ALPS original; número de *features* modularizadas em apenas um pacote; e o surgimento ou não de grandes classes.

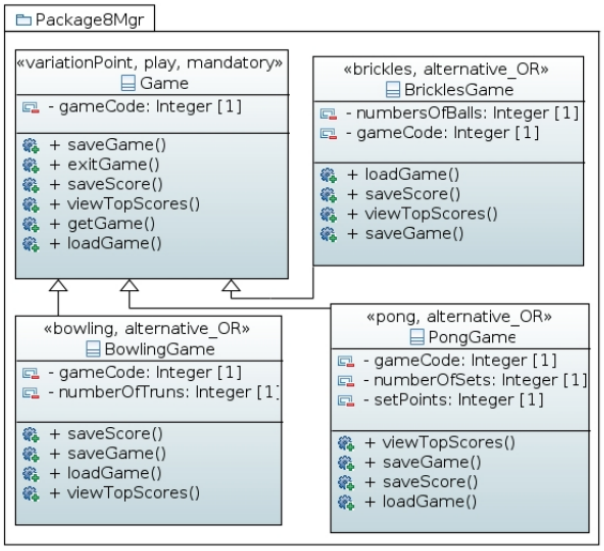
Com base nas respostas dadas pelo arquiteto, foi possível realizar uma investigação mais profunda sobre o modelo proposto. Em relação ao número de pacotes e classes, a ALPS AGM original possui 9 pacotes e 30 classes. A análise das tabelas (Tabela - 6.2 e Tabela - 6.3) mostra variações em relação ao número de pacotes em comparação com a ALPS original da AGM , com todas as soluções tendo um número entre 8 e 12 pacotes e não havendo variação em relação ao número de classes. Todas as soluções (em ambas execuções) têm o número de classes igual a 28. Na ALPS AGM original, há um total de 30 classes, no entanto, existem duas classes (*"point"* e *"display"*) no pacote *"GameBoardCtrl"* que não contêm atributos e métodos e foram removidas durante o processo de otimização, pois uma restrição da abordagem MOA4PLA é não possuir classes vazias. Levando isso em consideração, pode-se observar que todas as soluções contêm o mesmo número de classes que a solução original. Ao analisar o número de pacotes, pode-se observar que existem duas soluções com a mesma quantidade de pacotes da solução original, uma na Exec-MIP (Solução 2) e outra em Exec-MA (Solução 6). No entanto, na execução com interação, também existem duas outras soluções (Soluções 1 e 5) com uma quantidade mais próxima de pacotes em relação ao original.

O segundo aspecto mencionado pelo arquiteto é a modularização de *features*. Tentativas de modularizar *features*, tais como: *"bowling"*, *"brickles"*, *"pong"* e *"save"* podem ser vistas em ambas execuções (Figura - 6.2 e Figura - 6.3).

Figura 6.2: Excerto da ALPS AGM que apresenta a modularização das *features: Bowling*, *Brickles* e *Pong*.



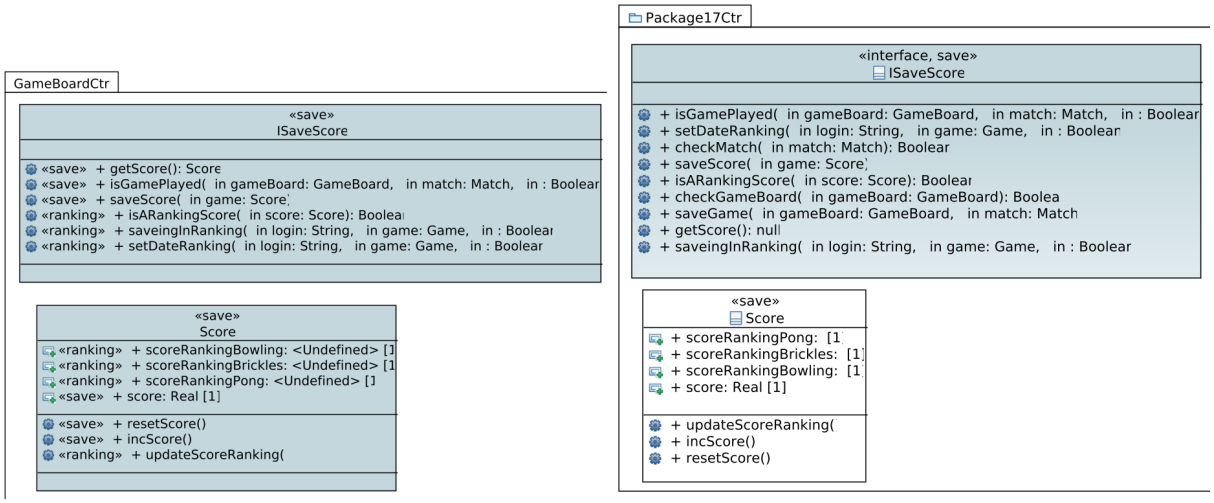
(a) Pacote GameMgr da ALPS original da AGM



(b) Pacote criado na Solução 1 da Exec-MIP

Fonte: Autoria própria.

Figura 6.3: Excerto da ALPS AGM que apresenta a modularização da *feature Save*.



(a) Pacote GameBoardCtrl da ALPS original da AGM

(b) Pacote criado na Solucão 1 da Exec-MA

Fonte: Autoria própria.

Em alguns casos, novos pacotes foram criados para modularizar essas *features*. A Figura - 6.2 apresenta uma melhoria na modularização de *features* da Solução 1 da Exec-MIP (b), considerando a criação de um novo pacote que encapsula os tipos de jogos incluídos na ALPS AGM. Este pacote é composto pelas classes *Game*, *BowlingGame*, *BricklesGame* e *PongGame*. A Figura - 6.3 mostra a modularização da *feature "save"* na Solução 1 - Exec-MA (b). Neste caso, um novo pacote composto apenas por classes relacionadas a essa *feature* foi criado.

O último aspecto descrito como relevante pelo arquiteto foi o surgimento de grandes classes. Neste caso, não foi possível perceber diferenças entre as execuções, pois as maiores classes encontradas na ALPS AGM original permaneceram inalteradas após o processo de otimização. Como exemplo, a maior classe encontrada na ALPS AGM original é a *"GameBoard"* (10 atributos e 23 métodos) e, em todas as soluções, de ambas execuções, a quantidade de atributos e métodos para essa classe permaneceu a mesma.

Em relação às modificações feitas na abordagem MOA4PLA e ferramenta OPLA-Tool para permitir a interação, o arquiteto que participou de Exec-MIP considerou que o MIP pode trazer soluções mais apropriadas para cada arquiteto em particular. No entanto, a configuração relacionada ao número de indivíduos e gerações não foi suficiente para mostrar uma melhoria explícita nas soluções. Por esse motivo, segundo o arquiteto, nenhuma solução recebeu o `escore` = 5. Além disso, foi mencionado também que, entre uma interação e outra era difícil perceber modificações significativas entre as soluções. Como esse experimento foi o primeiro a considerar o processamento interativo

da OPLA-Tool, a opção de não adotar um espaçamento entre gerações foi por não saber com antecedência se era necessário.

Do ponto de vista do arquiteto, o processo evolutivo deveria ainda ser estendido por um período mais longo e com um número maior de indivíduos (cerca de 100 indivíduos), a fim de identificar melhores otimizações. Além disso, o arquiteto sugeriu que, para evitar o problema da fadiga, deveria ser limitado o número de soluções a serem avaliadas em cada momento de interação. Os resultados de Exp 01 foram publicados em (Bindewald et al., 2019).

6.3 Experimento 2 (Exp 02)

Nesta seção são apresentados e analisados os resultados obtidos em Exp 02. Neste experimento, testes foram realizados com as três estratégias propostas para a formação do conjunto de dados de treinamento do modelo de AM (Seção 5.3) no intuito de se verificar qual obteria os melhores resultados.

6.3.1 Método de análise quantitativa

Para este segundo experimento, as configurações de execução da OPLA-Tool foram alteradas para possibilitar um processo evolutivo maior, de modo a tornar mais significativas as otimizações feitas pela ferramenta em cada momento de interação.

Assim, as configurações da OPLA-Tool para Exp 02 foram:

- número de execução do algoritmo: 1;
- número de gerações: 15;
- população inicial: 200 indivíduos;
- operador de mutação com probabilidade = 0,9.
- número de interações = 5;
- momento da primeira interação = 3;
- intervalo entre gerações = 3;

Como mencionado anteriormente (Seção 5.3), a elaboração de diferentes estratégias para compor o conjunto de treinamento do modelo de AM visou tratar uma deficiência encontrada para o modelo de AM que era a pouca quantidade de dados, visto que

os mesmos são gerados em tempo de processamento a partir das ALPS avaliadas pelo arquiteto. Desta forma, três estratégias foram elaboradas contemplando a utilização de todos os dados gerados a cada momento de interação (todas as ALPS), sendo as mesmas descritas, resumidamente, a seguir:

- **Estratégia 1:** A partir da avaliação de uma (ou mais) ALPS, o `escore` da mesma (ou a média, no caso de mais de uma ALPS avaliada) é generalizado para todas as soluções do mesmo *cluster*;
- **Estratégia 2:** As soluções não-avaliadas pelo arquiteto recebem um `escore = 0`, para que o modelo de AM aprenda quais soluções não devem ser avaliadas;
- **Estratégia 3:** Utiliza as duas estratégias anteriores, buscando ser um meio-termo entre as mesmas. Assim, para metade das soluções do *cluster* o `escore` (ou a média) da(s) solução(ões) avaliada(s) é atribuído(a) para as mesmas. As soluções restantes, recebem um `escore = 0`.

Para testar as três estratégias, foram feitos testes utilizando o Software Weka, por meio dos modelos de avaliação: *Training Set* (TS) e *Cross-validation* (CV) com 5 e 10 *folds*. No teste TS, o modelo de aprendizagem é treinado e testado usando o mesmo conjunto de dados. Para o teste CV, o conjunto de dados é dividido em n-grupos (*n-folds*), onde um *fold* é uutilizado para testar o modelo e os *folds* restantes são utilizados para treinar o modelo.

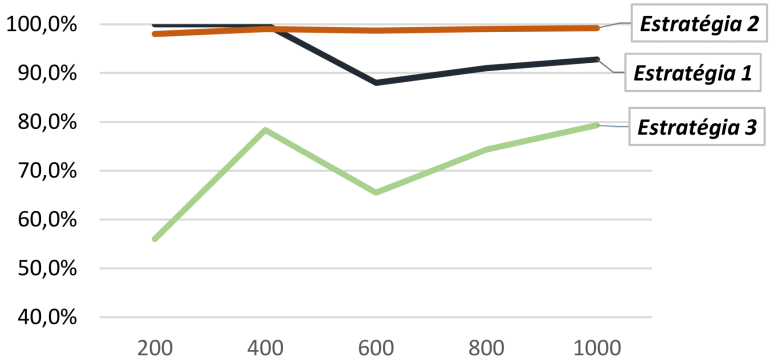
Os testes com TS e CV (5 e 10 *folds*) foram realizados em dois cenários diferentes: com e sem alterações no perfil do arquiteto. Esses cenários são necessários porque o arquiteto pode alterar seus critérios de avaliação de ALPS durante o processo de otimização. Em uma interação, o arquiteto pode avaliar as soluções de ALPS usando o critério de coesão e, em outra interação, o arquiteto pode alterar seu critério de avaliação para o acoplamento de classes, por exemplo. O conjunto de dados foi incrementado após cada momento de interação na OPLA-Tool. Assim, os testes foram realizados com: 200, 400, 600, 800 e 1000 instâncias.

6.3.2 Resultados da análise quantitativa

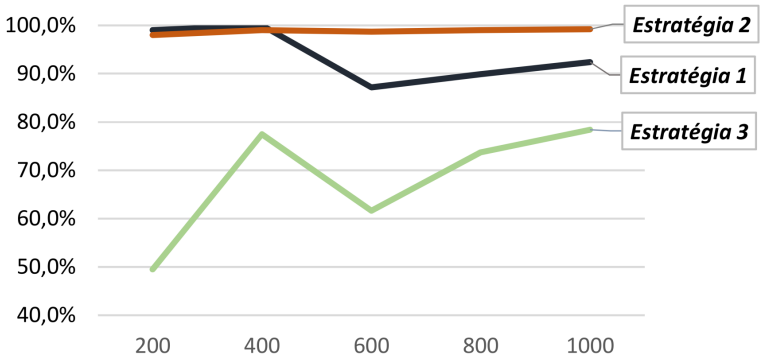
A Figura - 6.4 mostra os resultados dos testes TS e CV (com 5 e 10 *folds*) sem alterações no perfil do arquiteto e a Figura - 6.5 mostra os resultados dos testes TS e CV (com 5 e 10 *folds*) com alterações no perfil do arquiteto. O eixo "X" apresenta o número de instâncias

e o eixo "Y" demonstra a porcentagem de instâncias classificadas corretamente de cada estratégia.

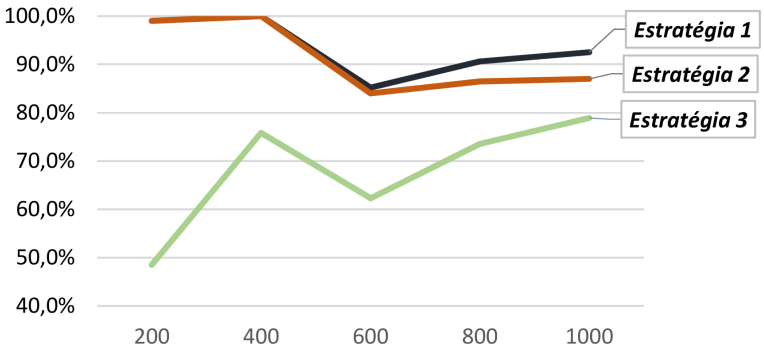
Figura 6.4: Experimentos conduzidos sem alterações no perfil do arquiteto



(a) Teste *Training set*



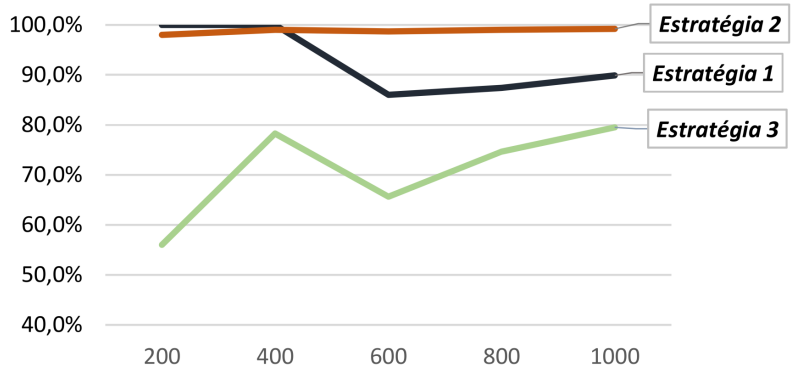
(b) Teste *Cross-validation* com 5 *folds*



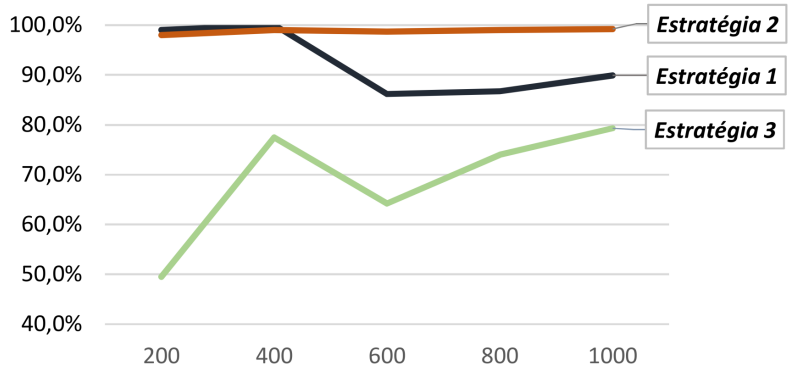
(c) Teste *Cross-validation* com 10 *folds*

Fonte: Autoria própria.

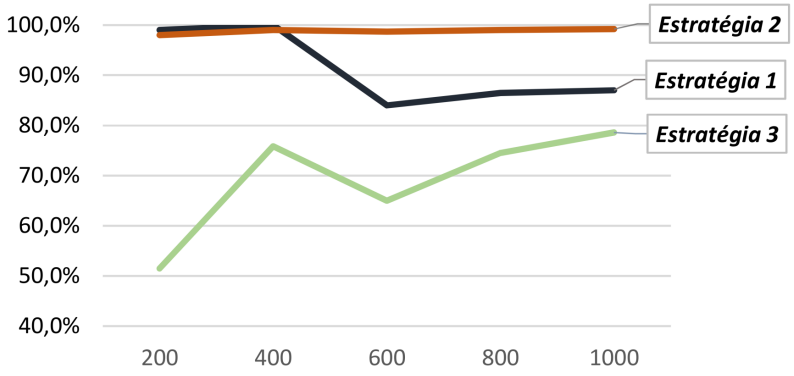
Figura 6.5: Experimentos conduzidos com alterações no perfil do arquiteto



(a) Teste *Training set*



(b) Teste *Cross-validation* com 5 *folds*



(c) Teste *Cross-validation* com 10 *folds*

Fonte: Autoria própria.

A análise dos gráficos permite inferir que a Estratégia 2 apresenta os melhores resultados em quase todos os cenários, exceto em CV com 10 *folds* sem alterações no

perfil do arquiteto. Além disso, a Estratégia 3 apresenta os piores resultados em todos os cenários. No entanto, há uma limitação importante a ser discutida com relação à Estratégia 2. Nesta estratégia, o arquiteto avalia uma solução para cada *cluster* e as instâncias restantes recebem `escore = 0` . Assim, há apenas uma pequena porcentagem de instâncias com avaliações diferentes de "zero". Neste caso, o OPLA-LM "aprende" que instâncias completamente diferentes podem ter a mesma avaliação (escore).

Por esse motivo, quando a rede MLP precisa prever valores de avaliação, todas as instâncias são previstas com `escore = 0`. Essa estratégia foi proposta porque uma ameaça verificada na Estratégia 1 foi de que o OPLA-LM deveria aprender quando não avaliar uma solução. Para isso, soluções não-avaliadas receberiam uma avaliação (`escore = 0`) como indicativo de solução não-avaliada.

Contudo, no caso da Estratégia 2, a rede MLP acaba entende que praticamente todas as soluções devem receber um `escore = 0`. Assim, em um cenário real, as preferências do arquiteto não seriam capturadas pelo OPLA-LM.

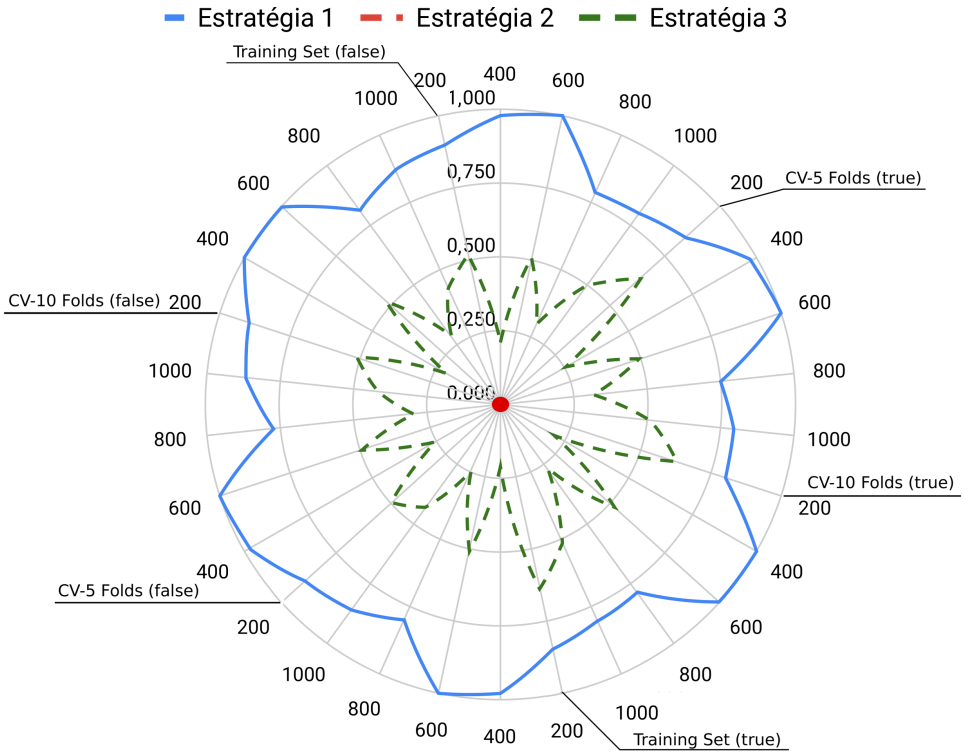
A Estratégia 1 começa com maior precisão, quando há apenas 200 instâncias para o treinamento do algoritmo ML. Considerando 600 instâncias no conjunto de dados, essa estratégia apresenta redução na precisão nos dois testes com e sem alteração de perfil do arquiteto. No entanto, ao final do treinamento do OPLA-LM, essa estratégia se estabiliza com resultados promissores (mais de 80 % de precisão).

A Estratégia 3 apresenta um aumento na precisão ao final do treinamento. Contudo esta estratégia também considera que soluções não-avaliadas devem receber um valor de `escore = 0`. Neste caso, um outro problema que se nota é que não há um critério objetivo que indique qual solução seria avaliada ou não pelo arquiteto. Desta forma, problema semelhante ao que ocorre na Estratégia 2 poderia ocorrer, ou seja, soluções muito diferentes poderiam obter o mesmo valor de avaliação (`escore = 0`), não capturando com eficiência as preferências do arquiteto.

A Subseção 4.2.2 apresentou as medidas de avaliação utilizadas para avaliar os resultados obtidos nos 3 experimentos realizados (Exp 01, Exp 02 e Exp 03). A Figura - 6.6 mostra os resultados com a aplicação da métrica Kappa, que indica se a previsão de dados é confiável. Deve-se ressaltar que as notações "*true*"e "*false*"no gráfico simbolizam "com" e "sem" alterações no perfil do arquiteto.

As divisões na Figura - 6.6 representam a confiabilidade da previsão em determinados momentos. Existem 5 momentos para cada um dos testes feitos: 200, 400, 600, 800 e 1000. Estes momentos representam as instâncias usadas para o treinamento. A análise permite observar que a Estratégia 1 obteve valores de Kappa sempre variando entre 0,75 e 1; a Estratégia 3 obteve valores sempre entre 0,25 e 0,5 e, por fim, a Estratégia 2 sempre

Figura 6.6: Medida Kappa para os experimentos com e sem alterações no perfil do arquiteto.



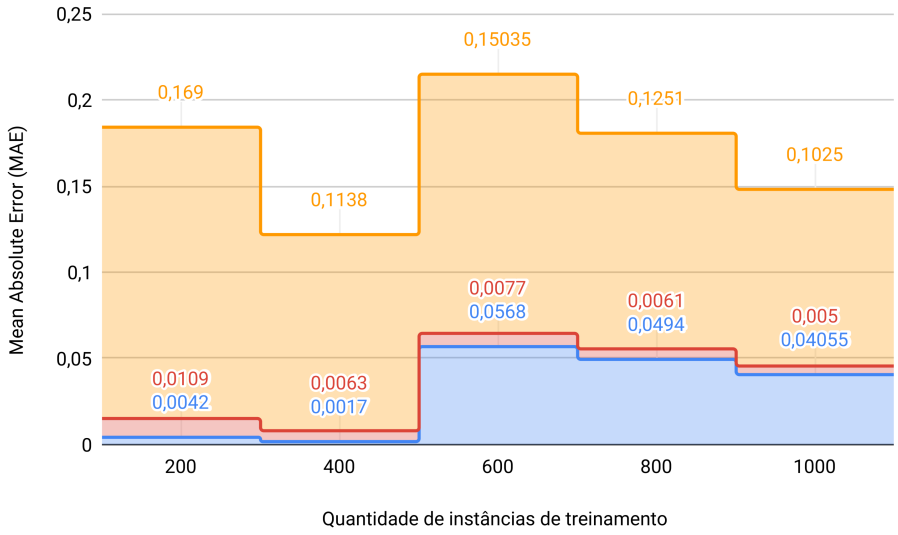
Fonte: Autoria própria.

obteve o valor de "zero". De acordo com a Tabela Tabela - 4.2, pode-se inferir que a Estratégia 1 possui concordância de "substancial"a "quase perfeita"; a Estratégia 3 possui concordância de "razoável"à "moderada"e, para a Estratégia 2, não há concordância. Neste sentido, ao se considerar os valores de Kappa obtidos por cada uma das estratégias, conclui-se que a Estratégia 1 é a que possui a maior confiabilidade em relação as predições feitas pelo algoritmo MLP.

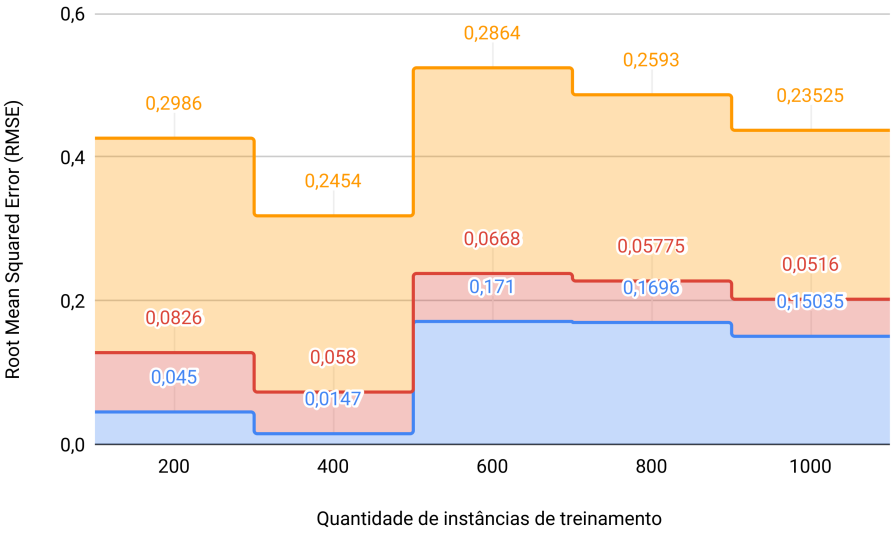
Além da métrica Kappa, para definir qual a melhor Estratégia, foram feitos também os cálculos referentes às métricas de erro (RAE, RRSE, RMSE e MAE). A Figura - 6.7 demonstra graficamente os valores atingidos pelas métricas MAE e RMSE. Já a Figura - 6.8 demonstra graficamente os valores atingidos pelas métricas RAE e RRSE. Nos gráficos (Figura - 6.7 e Figura - 6.8) as três Estratégias são representadas pelas cores "azul", "laranja"e "vermelha", respectivamente para as Estratégias 1, 2 e 3. Os gráficos foram construídos como uma composição de áreas de modo a fazer um comparativo visual dos valores de erros obtidos por cada uma das Estratégias em cada quantidade de instâncias de treinamento consideradas (200, 400, 600, 800 e 1000). Além disso, para a composição

dos gráficos, foram considerados os valores de erros atingidos em cada um dos testes realizados (CV com 5 e 10 *folds* com e sem alterações no perfil de arquiteto; *training-set* com e sem alterações no perfil do arquiteto). Os rótulos indicados nos gráficos representam a mediana dos erros de cada um dos testes, para cada estratégia em cada quantidade de instâncias consideradas. Assim, como exemplo, o valor 0,169 (área laranja) na Figura - 6.7 (a) representa o valor de mediana do MAE para os testes feitos com a Estratégia 3 com 200 instâncias de treinamento.

Figura 6.7: Gráficos de áreas empilhadas para MAE e RMSE referente às Estratégias 1, 2 e 3



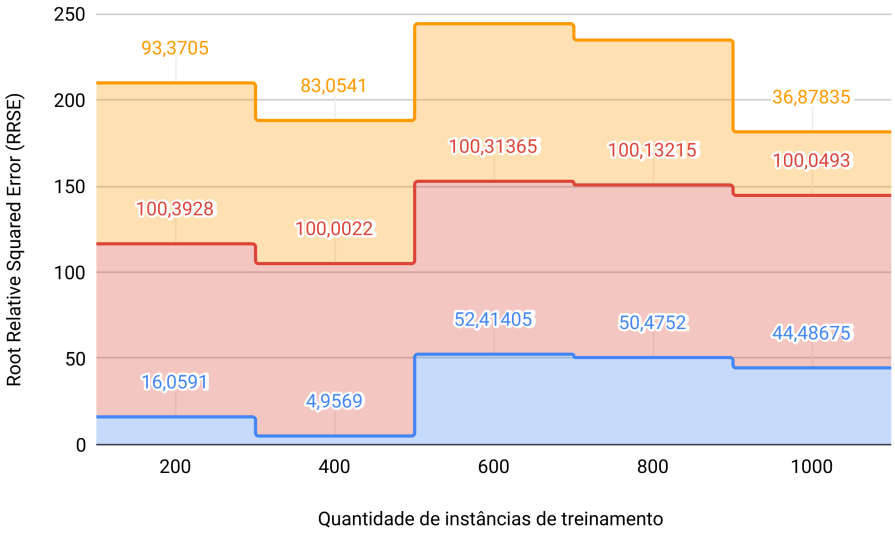
(a) *Mean Absolute Error* (MAE)



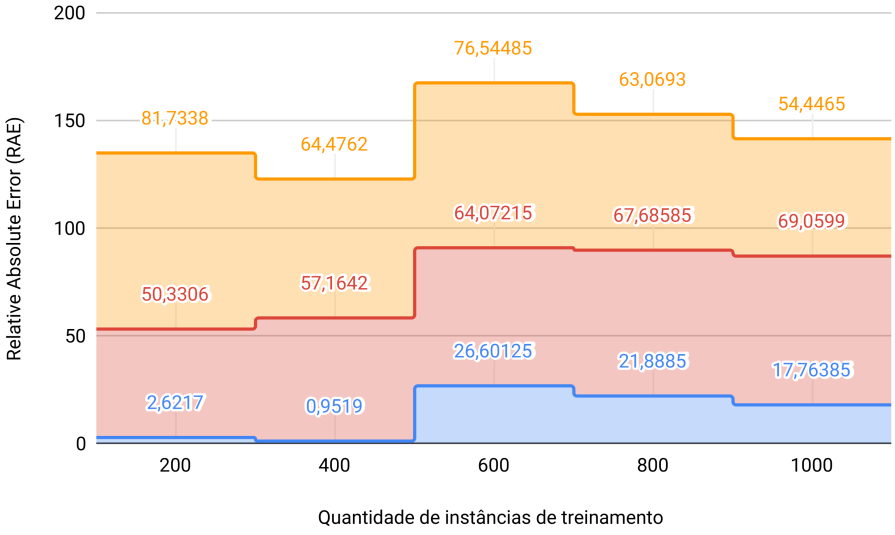
(b) *Root Mean Squared Error* (RMSE)

Fonte: Autoria própria.

Figura 6.8: Gráficos de áreas empilhadas para RRSE e RAE referente às Estratégias 1, 2 e 3



(a) *Root Relative Squared Error* (RRSE)



(b) *Relative Absolute Error* (RAE)

Fonte: Autoria própria.

A análise dos gráficos (Figura - 6.7 e Figura - 6.8) permite inferir que a Estratégia 2 não obteve os menores índices de erros em nenhuma quantidade de instâncias consideradas.

A Estratégia 3 obteve os menores índices de erros para MAE e RMSE, considerando-se 600, 800 e 1000 instâncias de treinamento.

A Estratégia 1, por sua vez, obteve os menores índices de erro para RAE e RRSE (em qualquer quantidade de instâncias de treinamento) e também para MAE e RMSE, ao se considerar 200 e 400 instâncias de treinamento.

De uma forma geral, embora a Estratégia 2 tenha obtido melhores resultados quando se analisou a acurácia, as outras análises (Kappa, RAE, RRSE, RMSE e MAE) permitiram dizer que a Estratégia 1 se sobressaiu sobre a Estratégia 2. Como mencionado anteriormente, a Estratégia 2 foi proposta para incluir pontuações "zero" no conjunto de treinamento, para que o OPLA-LM aprendesse as soluções que não deveriam ser avaliadas. Essa estratégia obteve melhores resultados em quase todos os testes considerando as instâncias classificadas corretamente. No entanto, ao analisar as pontuações atribuídas, foi possível observar que esta estratégia sempre atribui a avaliação "zero" a todas instâncias, inclusive para soluções que não deveriam ser avaliadas com a `escore = 0`.

As medidas Kappa, RAE, RRSE, RMSE e MAE demonstraram que existe um viés nesses resultados. Para buscar um meio-termo entre as Estratégias 1 e 2, foi proposta então a Estratégia 3 para mesclar as Estratégias 1 e 2.

No entanto, a análise dos resultados da Estratégia 3 demonstrou que a mesma superou a Estratégia 1 apenas ao se analisar as métricas MAE e RMSE com 600, 800 e 1000 instâncias utilizadas para o treinamento do modelo de AM. Para as outras métricas (Kappa, RAE e RRSE), bem como para MAE e RMSE (com 200 e 400 instâncias) a Estratégia 3 não superou a Estratégia 1. Além disso, no caso da Estratégia 3, não há um critério objetivo para se dizer quais soluções que deveriam ser avaliadas ou não. Desta forma, a Estratégia 1 demonstrou-se ser a alternativa mais adequada para a formação do conjunto de dados de treinamento do modelo de AM.

6.4 Experimento 3 (Exp 03)

A partir dos resultados trazidos pelo Exp 02, a Estratégia 1 foi incorporada definitivamente no modelo de AM e um novo experimento (Exp 03) foi realizado. Este experimento teve como objetivo testar as modificações feitas na abordagem MOA4PLA e ferramenta OPLA-Tool como um todo, ou seja, o processo interativo juntamente com o modelo de AM. Para isso, foram selecionados 10 arquitetos de software para avaliação das arquiteturas geradas em tempo de execução. As análises quanti-qualitativas feitas a partir de Exp 03 permitiram avaliar a adequação das arquiteturas geradas para cada arquiteto em particular, bem como avaliar o funcionamento do modelo de AM.

As configurações da OPLA-Tool para a execução deste experimento foram:

- número de execução do algoritmo: 1;
- número de gerações: 15;

- população inicial: 200 indivíduos;
- operador de mutação com probabilidade = 0,9.
- número de interações = 3;
- momento da primeira interação = 6;
- intervalo entre gerações = 3;

Para este experimento, além dos 3 momentos de interação, nos quais os arquitetos avaliaram as arquiteturas geradas em tempo de execução, houve também um quarto momento de avaliação de soluções. Esta avaliação final, refere-se à avaliação das soluções finais, ou seja, soluções restantes do processo de otimização.

6.4.1 Método de análise qualitativa

Para a realização do experimento, antes da execução da OPLA-Tool, foi feito um treinamento prévio com os participantes. Este treinamento visou explicar como seria realizado o experimento. Desta forma, os participantes receberam informações referentes às configurações de execução da ferramenta, bem como de como seriam feitas as avaliações das arquiteturas, visto que estas seriam visualizadas em uma outra ferramenta (ferramenta *Papyrus*) e alguns ajustes ainda deveriam ser feitos para a visualização. Além disso, foram explicados alguns conceitos como Coesão, Acoplamento de Classes e Modularização de *features* para auxiliar os participantes em suas análises.

Para a análise qualitativa, foi pedido aos participantes de Exp 03 que respondessem um questionário. Este questionário visou, em um primeiro momento, identificar o perfil dos participantes com os seguintes questionamentos:

- "Qual o seu nível de escolaridade?";
- "No momento é estudante de Pós-graduação?";
- "Trabalha em desenvolvimento de software?";
- "Qual o seu nível de conhecimento em UML (Unified Modeling Language)?";
- "Qual o seu o nível de conhecimento em ALPS (Arquitetura de Linha de Produto de Software)?";

Além disso, para avaliar o modelo interativo na abordagem MOA4PLA e ferramenta OPLA-Tool, os participantes responderam também aos seguintes questionamentos:

- "Apesar de se tratar de uma avaliação subjetiva, há algum critério específico que foi utilizado para a avaliação das ALPS?";
- "De uma forma geral, as soluções geradas pelo modelo interativo da OPLA-Tool estão adequadas ao seu perfil?";
- "Nos trabalhos relacionados a interação em SBSE, há uma grande preocupação por parte dos pesquisadores em relação ao problema da fadiga, ocasionado pelo elevado número de interações do DM. Neste experimento, a quantidade mínima de soluções a serem avaliadas foi de uma solução por *cluster* em cada momento interação. Você considera que esta quantidade foi: pouca, suficiente ou excessiva?".

Por fim, ao final do questionário, os participantes puderam deixar também um *feedback* em relação ao que poderia ser melhorado no processo interativo.

6.4.2 Método de análise quantitativa

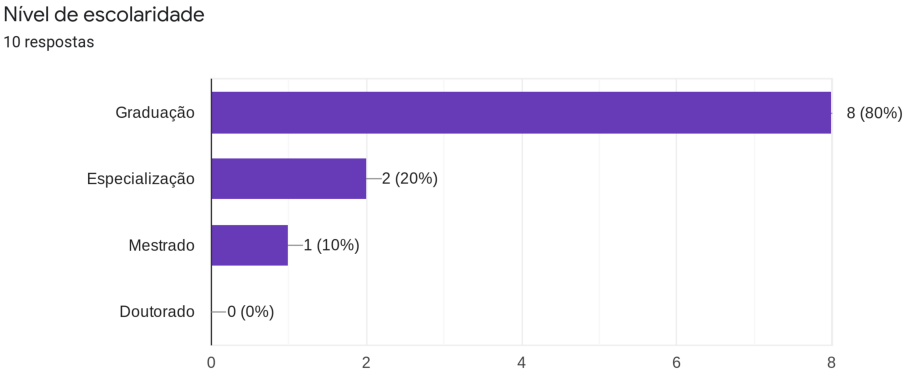
A análise quantitativa focou na variação das avaliações (escores) dadas por cada arquiteto durante todo o processo de interação. Desta maneira, foram coletadas todos os escores dados a cada uma das soluções (ALPS) avaliadas em cada momento de interação e foi feito o cálculo da mediana destes escores. A escolha da utilização da mediana foi em função da mesma fornecer uma tendência central dos valores.

Em um primeiro momento, o cálculo das medianas foi feito para cada arquiteto individualmente. Posteriormente, foram feitos os cálculos das medianas também por grupos, da seguinte maneira: grupos de participantes por nível de conhecimento em UML e ALPS (básico, intermediário ou avançado). Dentro destes grupos, foi verificado se nos mesmos existiam participantes que notaram fadiga durante Exp 03. Esta verificação foi necessária para identificar se a fadiga pode ter afetado o resultado do grupo como um todo.

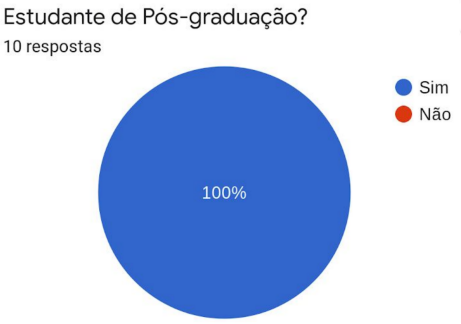
6.4.3 Resultados da análise qualitativa

Os resultados aos questionamentos feitos em relação ao perfil dos participantes indicaram que todos possuem nível superior de escolaridade e atualmente são estudantes de pós-graduação. Além disso, a maioria possui nível intermediário em UML e ALPS e não trabalham com desenvolvimento de software. A Figura - 6.9 demonstra o quadro geral do perfil dos participantes do Exp 03.

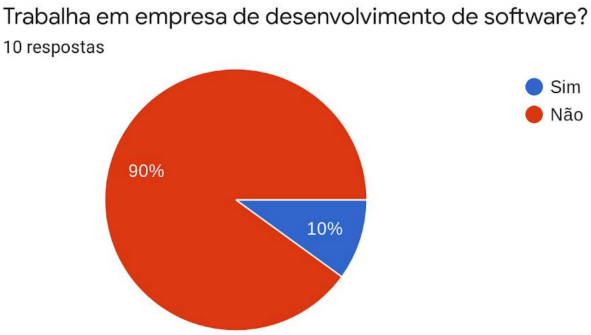
Figura 6.9: Perfil dos participantes do Exp 03



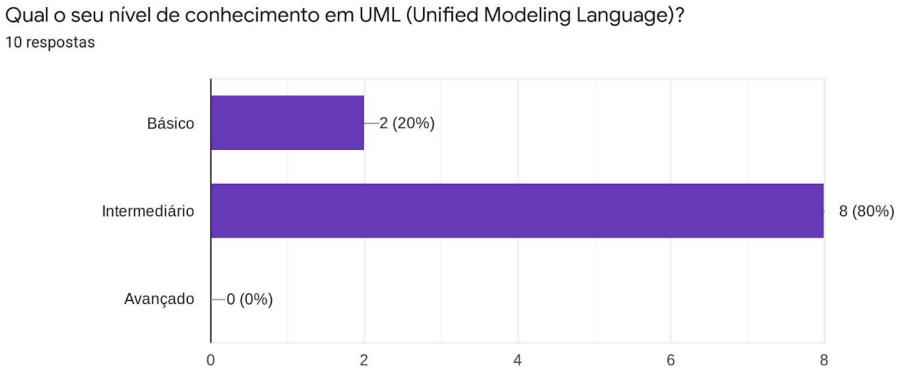
(a) Nível de escolaridade



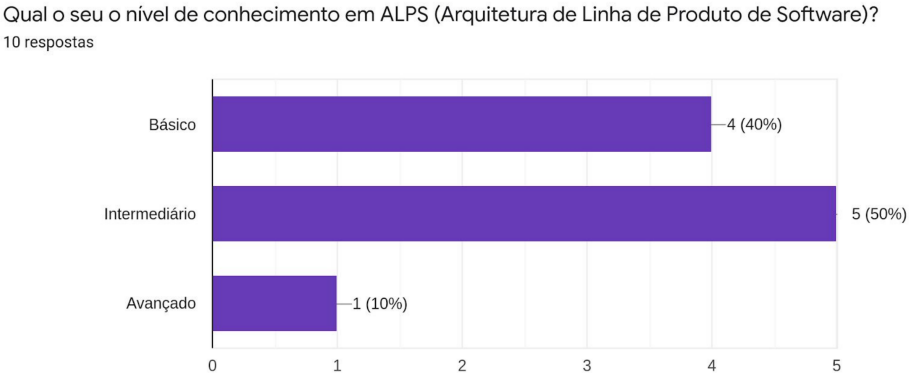
(b) Quantidade de estudantes de Pós-graduação



(c) Trabalham com desenvolvimento de software



(d) Nível de conhecimento em UML



(e) Nível de conhecimento em ALPS

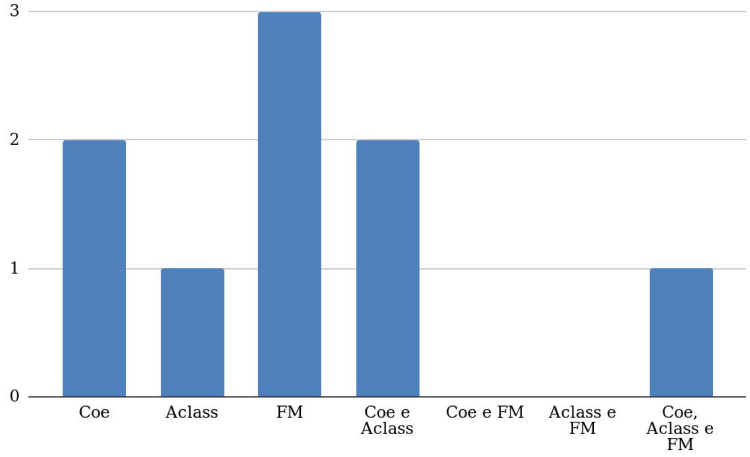
Fonte: Autoria própria.

Em relação aos questionamentos feitos para avaliar o processo interativo na abordagem MOA4PLA e ,ferramenta OPLA-Tool, as respostas indicaram que, de uma forma geral, os participantes possuem uma tendência em analisar a modularização de características (FM) e, além disso, que as soluções geradas estavam adequadas a cada participante e que os mesmos não sentiram cansaço (fadiga) durante a avaliação das ALPS. Estes resultados são apresentados e discutidos em maior profundidade na sequência.

Questionamento 1 (Q1): "Apesar de se tratar de uma avaliação subjetiva, há algum critério específico que foi utilizado para a avaliação das ALPS?"

Os resultados de Q1 podem ser vistos na Figura - 6.10, que indica uma maior tendência a se analisar uma ALPS sob o ponto de vista da modularização de características (FM).

Figura 6.10: Critérios avaliados por cada participante.

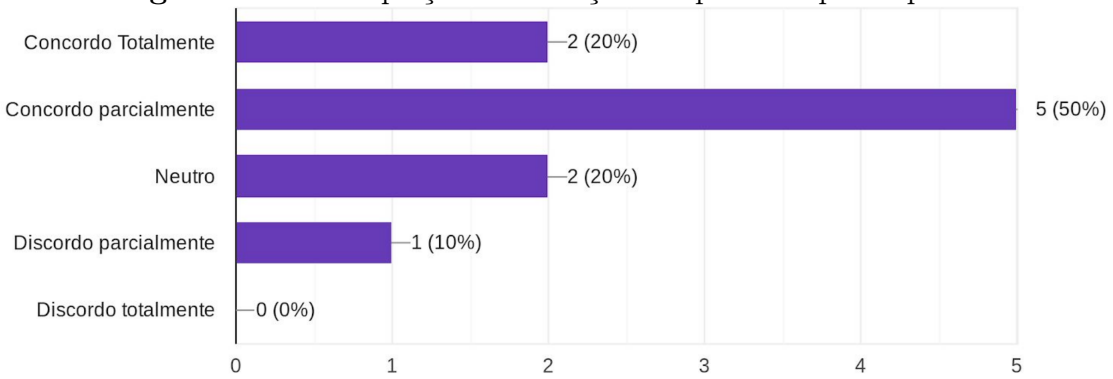


Fonte: Autoria própria.

Questionamento 2 (Q2): "De uma forma geral, as soluções geradas pelo modelo interativo da OPLA-Tool estão adequadas ao seu perfil?"

Este questionamento buscou saber, sob a perspectiva do arquiteto, se o processo interativo conseguiu trazer ao final soluções que estavam conforme o que cada participante esperava, o que pode indicar que o processo interativo, junto com o modelo de AM, foram hábeis a capturar as preferências de cada arquiteto. Os resultados, que podem ser vistos na Figura - 6.11, indicam que, em geral, os participantes do experimento consideram que o modelo interativo possibilitou a geração de alternativas de ALPS que estavam de acordo com as suas expectativas (50% de concordância parcial, 20% de concordância total). Este fato pode ser comprovado também pelas análises feitas a partir do cálculo das medianas dos escores dados por cada participante (vide Subseção 6.4.4).

Figura 6.11: Adequação das soluções ao perfil do participante.



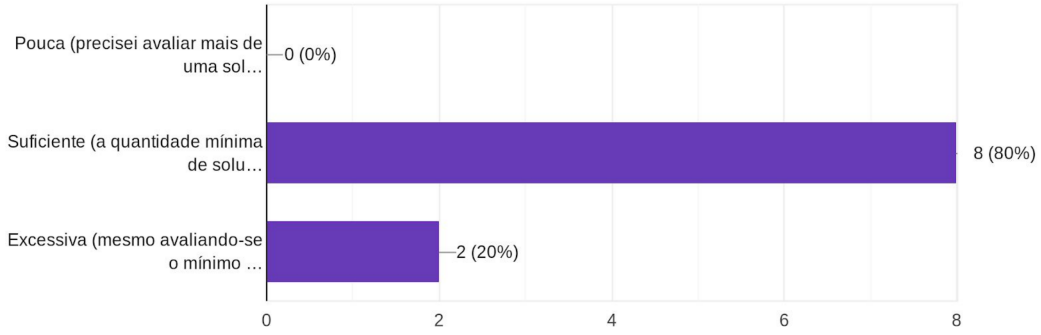
Fonte: Autoria própria.

Questionamento 3 (Q3): "Dos trabalhos relacionados a interação em SBSE, há uma grande preocupação por parte dos pesquisadores em relação ao problema da fadiga, ocasionado pelo elevado número de interações do DM. Neste experimento, a quantidade mínima de soluções a serem avaliadas foi de uma solução por *cluster* em cada momento de interação. Você considera que esta quantidade foi:

- **Pouca** (precisei avaliar mais de uma solução por *cluster*);
- **Suficiente** (a quantidade mínima de soluções a serem avaliadas, possibilitou-me inserir adequadamente as minhas preferências sobre o processo evolutivo);
- **Excessiva** (mesmo avaliando-se o mínimo possível de soluções, uma por *cluster*, notei cansaço durante o processo interativo)."

As respostas do Q3 possibilitaram dizer que a quantidade de soluções avaliadas foi suficiente, não gerando fadiga para a maioria dos participantes. A quantidade de soluções avaliadas foi de 1 por *cluster* em 3 momentos de interação, totalizando 12 soluções avaliadas. Além disso, foram avaliadas mais 4 soluções (1 por *cluster*) das soluções restantes ao final do processo de otimização. A Figura - 6.12 mostra os resultados para Q3.

Figura 6.12: Percepção sobre a quantidade de soluções avaliadas.



Fonte: Autoria própria.

Por fim, foi pedido também que, se possível, os participantes fornecessem um *feedback* (como uma questão aberta) sobre o que pode ser melhorado no processo de interativo da OPLA-Tool. Neste caso, apenas 3 participantes responderam.

Duas das respostas indicaram que deve ser melhorado a forma como a solução é apresentada na ferramenta Papyrus. Esta ferramenta é utilizada atualmente como suporte ao processo interativo. No entanto, para a avaliação das ALPS, é necessário fazer alguns ajustes na arquitetura gerada para permitir a sua visualização, visto que pode haver uma sobreposição de pacotes e/ou classes.

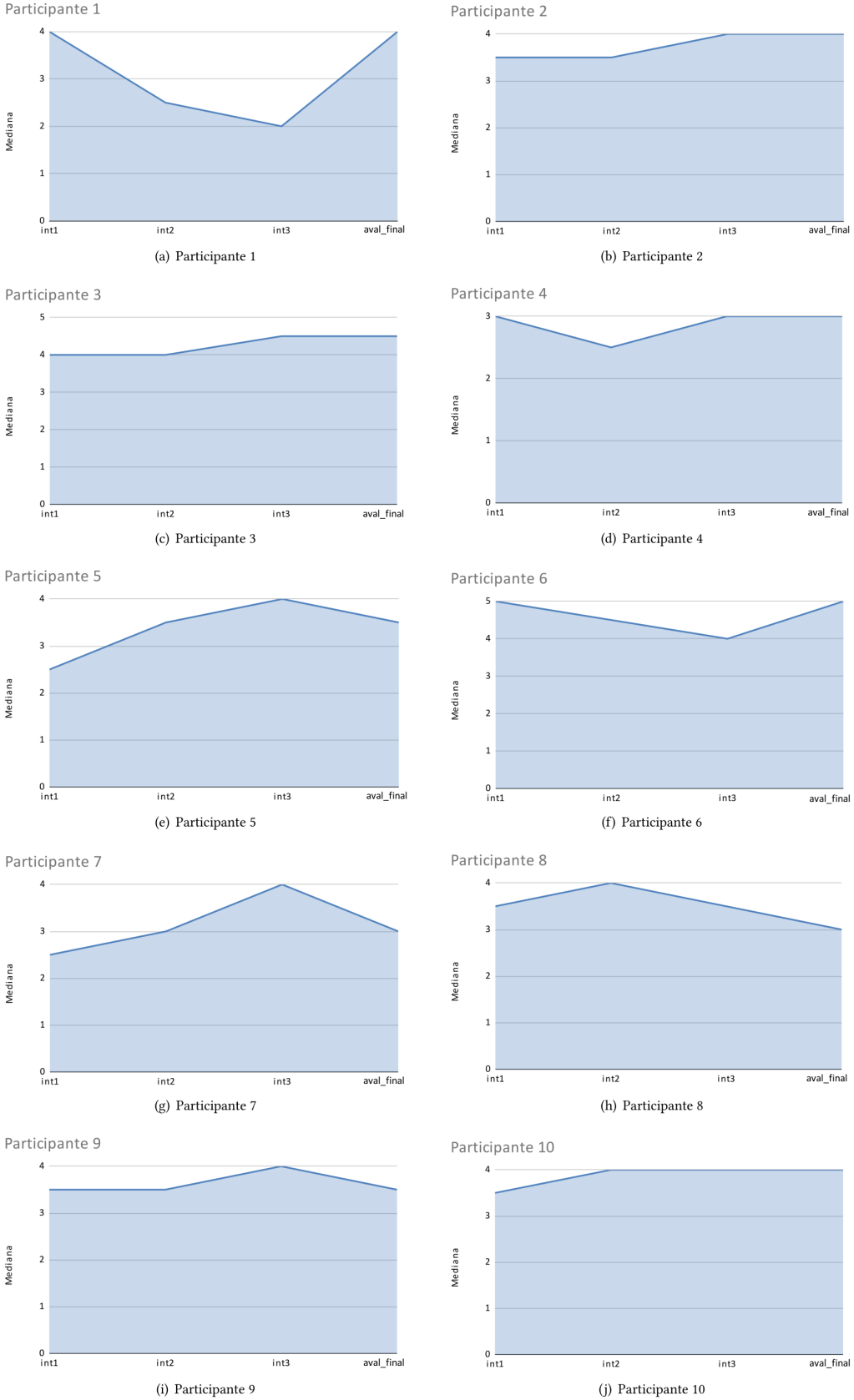
O outro participante que forneceu um *feedback* indicou sentir cansaço ao avaliar as soluções, pois se tornou um processo repetitivo, de modo que a quantidade de soluções avaliadas deveria ser menor, ou seja, a avaliação de uma solução por *cluster* foi excessiva para este participante.

6.4.4 Resultados da análise quantitativa

Os resultados da análise quantitativa a partir do cálculo das medianas dos escores indicaram que, de uma forma geral, as soluções estavam adequadas à cada participante (corroborando com os resultados da análise qualitativa). Além disso, foi possível perceber que, em alguns casos, a fadiga pode ter influenciado a avaliação das soluções finais (soluções restantes do processo de otimização). As análises feitas para cada participante individualmente e por grupos são detalhadas na sequência.

A Figura - 6.13 mostra a variação das medianas das notas dadas por cada participante em cada momento de interação, denotados por int1, int2, int3 e aval-final, respectivamente para interação 1, interação 2, interação 3 e avaliação das soluções restantes)

Figura 6.13: Mediana dos escores dados por cada participante



Fonte: Autoria própria.

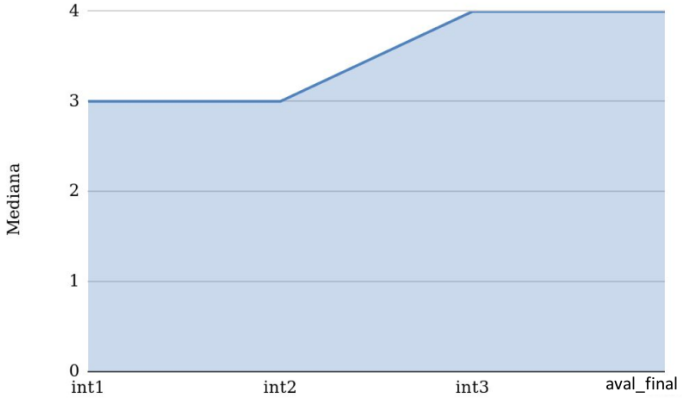
A análise dos gráficos (apresentados na Figura - 6.13) permite afirmar que, entre os 10 participantes, em 6 casos (participantes 1, 2, 3, 4, 6 e 10) as avaliações tenderam a uma crescente ao final, o que pode indicar que, para estes casos, as soluções foram melhor se adequando ao perfil destes participantes (arquitetos).

Ao analisar os 4 participantes nos quais as avaliações diminuíram ao final, pode-se constatar que um deles (participante 7), estava nos 20% (ver Figura - 6.12) de participantes que consideraram excessiva a quantidade de soluções avaliadas. Para este participante, a cada momento de interação, as avaliações estavam sempre crescentes. Contudo, na avaliação final houve uma queda. Como a avaliação final foi feita logo após cada um dos momentos de interação, a fadiga pode ter afetado o julgamento das soluções finais para este participante.

Outras análises foram feitas também por grupos de participantes. Assim, a análise foi dividida por nível de conhecimento em UML e ALPS de cada participante.

Do total de participantes, 20% disseram ter conhecimento básico em UML e o restante possuem conhecimento intermediário. A mediana dos escores dados pelos participantes com nível básico de UML se manteve no mesmo nível do começo ao fim do experimento. Em cada uma das interações o valor de mediana foi 4 e na avaliação das soluções finais esse valor se manteve. Já para o grupo com o nível intermediário em UML, houve uma variação para cima no valor de avaliação, passando de um valor 3 no primeiro momento de interação, indo para 4 e se mantendo em 4. A Figura - 6.14 mostra essa evolução.

Figura 6.14: Mediana dos escores para o conjunto de participantes com nível intermediário de conhecimento em UML



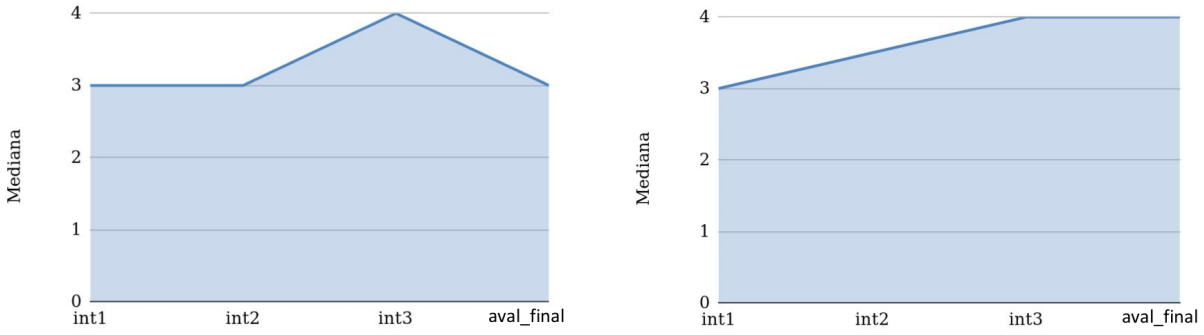
Fonte: Autoria própria.

A análise por conhecimento em ALPS foi dividida em 3 grupos, visto um participante ter conhecimento avançado em ALPS. Assim, do total de participantes, 40% possuem

conhecimento básico em ALPS, 50% conhecimento intermediário e 10% conhecimento avançado.

Do conjunto de participantes com conhecimento básico, o resultado do cálculo das medianas por cada momento de interação foi o mesmo do grupo de conhecimento básico em UML, começando em um valor 4 e se mantendo até o fim neste mesmo patamar. Para o grupo com conhecimento intermediário, fato interessante foi notado. A mediana dos escores dados por estes participantes tendia a uma crescente até o terceiro momento de interação. Contudo, na avaliação final, o valor decaiu. Ao se analisar os participantes deste grupo, pôde ser percebido que entre eles estavam os participantes que consideram excessiva a quantidade de soluções avaliadas, de modo que o efeito da fadiga pode ter afetado a avaliação das soluções finais destes participantes. Ao se retirar estes participantes e fazer o cálculo das medianas do conjunto com os participantes restantes, o valor de mediana se manteve na avaliação das soluções finais. A Figura - 6.15 mostra o gráfico das medianas dos escores para o grupo intermediário com e sem os participantes que informaram ser excessiva a quantidade de soluções avaliadas.

Figura 6.15: Mediana dos escores para o conjunto de participantes com nível intermediário de conhecimento em ALPS



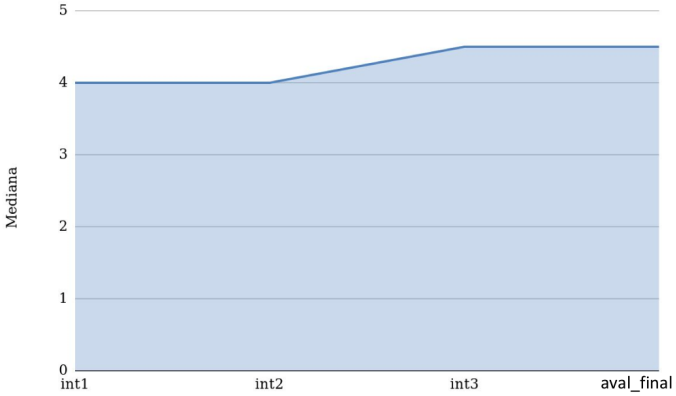
(a) Todos do conjunto de nível intermediário

(b) Conjunto de nível intermediário apenas com os que não sentiram cansaço

Fonte: Autoria própria.

Para o participante com conhecimento avançado ALPS, o valor da mediana começou em um nível 4, crescendo para 4,5 (Interação 3) e se mantendo neste nível na avaliação final. A Figura - 6.16 mostra o gráfico das medianas dos escores para este caso.

Figura 6.16: Mediana dos escores para o participante com nível avançado de conhecimento em ALPS



Fonte: Autoria própria.

6.5 Ameaças à validade

Os três experimentos realizados mostraram que o processo interativo na abordagem MOA4PLA e ferramenta OPLA-Tool pode proporcionar resultados mais adequados a cada arquiteto em particular. Contudo, algumas ameaças ao estudo podem ser destacadas:

1. Os experimentos foram realizados utilizando apenas a ALPS AGM, que possui um tempo de processamento relativamente curto pela OPLA-Tool. Assim, não é possível dizer que os bons resultados alcançados em relação ao problema da fadiga seriam também em outras ALPS, com tempos de processamento mais longos;

2. Outro fator que pode ser impactante no tempo de processamento e na fadiga do arquiteto é quantidade de funções-objetivo selecionadas. Para este trabalho, em todos os experimentos, foram utilizadas apenas três funções-objetivo (COE, AClass, FM). Assim, caso uma maior quantidade de funções-objetivo fossem selecionadas, o tempo de processamento poderia ser maior e afetar o processo interativo;

3. Uma outra ameaça foi encontrada também em relação à seleção dos participantes (arquitetos) para o estudo. Dentre todos os participantes, apenas um relatou ter conhecimentos avançados em ALPS. Desta forma, a avaliação das soluções (arquiteturas) durante os processos interativos pode ter sofrido o impacto dessa falta de conhecimento pela maioria dos participantes.

6.6 Lições aprendidas

Como já previsto em outros trabalhos que lidam com interação em processos de otimização, muitas vezes, a análise das soluções geradas, apenas pelo ponto de vista matemático, não permite dizer se as mesmas estão adequadas a cada DM em particular, pois certas características de uma solução estão relacionadas às preferências individuais de cada DM. No caso deste trabalho, foi possível perceber que este fato também ocorre, visto que soluções com exatos mesmos valores de *fitness* recebem, algumas vezes, avaliações diferentes.

No intuito de se encontrar soluções mais adequadas a cada DM ao final de um processo de otimização, foi constatado que pesquisas apontavam como estratégia a interferência do ser humano no processo de otimização como meio do mesmo opinar sobre o processo e, assim, guiá-lo a encontrar soluções mais promissoras.

Neste sentido, com este trabalho foi possível demonstrar que a interferência do ser humano (arquiteto) no processo de otimização realizado pela abordagem MOA4PLA e ferramenta OPLA-Tool é viável e promissor. Contudo, aspectos relacionados principalmente à fadiga do arquiteto, causada pela excessiva quantidade de soluções (arquiteturas) avaliadas, devem ser melhor explorados para se encontrar uma quantidade ideal de avaliação de soluções. Desta forma, pode ser possível um processo de otimização interativa mais eficiente.

Com a introdução do processo interativo, algumas configurações relacionadas ao mesmo ficaram a cargo do arquiteto, tais como: o momento da primeira interação, a quantidade de interações a serem feitas e também o intervalo entre gerações. Para os experimentos realizados neste trabalho, as configurações de execução do processo de otimização foram as mesmas para todos os participantes. Contudo, muitas vezes o arquiteto pode não ter por onde se basear para a definição dos parâmetros relativos ao processo interativo. Neste sentido, estes parâmetros podem ser colocados automaticamente pela ferramenta OPLA-Tool, baseado na quantidade de gerações estipuladas pelo arquiteto, deixando a cargo do mesmo a alteração posterior destes parâmetros. Como exemplo, poderia ser utilizada uma escala logarítmica, aonde quanto maior o número de gerações, mais tarde será o primeiro momento de interação, segundo a Equação 6.1:

$$Primeira\ interação = \log_2 X \tag{6.1}$$

Onde, X é o número de gerações.

O resultado da Equação 6.1 deve ser arredondado para que se tenha um número inteiro.

Em relação à quantidade de interações, foi visto experimentalmente (Exp 03) que 3 momentos de interação foi uma quantidade que trouxe bons resultados, tanto em relação à geração de alternativas mais adequadas de projetos de ALPS como também em relação ao problema da fadiga (a maioria dos participantes não indicou sentir cansaço durante o experimento). Desta maneira, a quantidade de 3 interações pode ser utilizada como padrão. Assim, o intervalo entre gerações seria a quantidade de gerações dividida por 3 momentos de interação.

6.7 Respostas às questões de pesquisa

Esta seção tem por objetivo responder aos questionamentos de pesquisa que embasaram o desenvolvimento deste trabalho.

- ***Q1:****Como adaptar a abordagem MOA4PLA e ferramenta-OPLA-Tool para permitir a interação homem-computador durante o processo de otimização?*

  A adaptação da abordagem MOA4PLA ocorreu a partir da inserção de novos dados na atividade de otimização multi-objetivo. Estes dados advêm da avaliação, por parte de um arquiteto, de arquiteturas geradas em tempo de processamento. Para isso, foi feita também a introdução do processo interativo com o arquiteto na ferramenta OPLA-Tool por meio da modificação do algoritmo NSGA-II, permitindo algumas pausas durante o processo de otimização para que o arquiteto pudesse avaliar arquiteturas geradas em tempo de processamento.

- ***Q2:****Levando-se em consideração que um excessivo número de interações no processo de otimização pode levar à fadiga do arquiteto, como a Aprendizagem de Máquina pode ser utilizada para que, em algum momento, venha a substituir o arquiteto no processo interativo?*

  Para o tratamento da fadiga do arquiteto, foram utilizados dois algoritmos de Aprendizagem de Máquina (AM): K-means e Multilayer Perceptron(MLP). O algoritmo K-means possibilitou uma diminuição da quantidade de soluções a serem avaliadas em cada momento de interação, pelo agrupamento de soluções semelhantes. O algoritmo MLP, por sua vez, possibilitou entender o comportamento do arquiteto (como o mesmo estava avaliando as arquiteturas) e assim, posteriormente, foi possível que o processo de otimização interativa continuasse sem a participação do arquiteto, com as soluções (arquiteturas) sendo avaliadas pelo algoritmo MLP.

6.8 Considerações finais

As análises quantitativa e qualitativa do Exp 03 permitiram dizer que, no geral, o processo interativo na OPLA-Tool pôde trazer soluções que estavam dentro das expectativas de cada arquiteto. Além disso, o modelo de AM foi hábil em capturar as preferências de cada participante, visto que a análise das medianas dos escores das soluções na avaliação final indicou que, na maioria dos casos, não houve queda nas mesmas. Ainda, pode-se dizer também que o modelo de AM possibilitou o tratamento da fadiga, pois apenas 20% dos participantes indicaram cansaço durante a realização do experimento.

O próximo capítulo descreve de uma forma geral as conclusões que foram possíveis com a realização deste trabalho e aponta também para futuros trabalhos que podem ser produzidos a partir dos resultados deste.

7

Conclusão

Muitos problemas complexos de Engenharia de Software têm sido tratados com sucesso por meio da utilização de técnicas de SBSE. Contudo, há casos em que ao final do processo de otimização, as soluções geradas não satisfazem o DM. Para estes casos, foi percebido que, muitas vezes, esta falta de concordância do DM em relação às soluções estava associada a aspectos mais subjetivos, como as preferências individuais de cada um, não sendo passíveis de serem modeladas matematicamente. Neste sentido, estudos têm obtido sucesso ao incorporar as preferências do DM durante o processo de otimização. Desta forma, o algoritmo pode receber informações do DM durante o processo evolutivo e, então, ser "guiado" a encontrar soluções mais promissoras.

A abordagem MOA4PLA, implementada pela ferramenta OPLA-Tool, modela o problema do projeto de uma ALPS como um problema de otimização multiobjetivos. Ao final do processo evolutivo, obtém-se um conjunto de soluções alternativas a uma ALPS original, com determinados objetivos otimizados. Contudo, atualmente a abordagem ainda não contempla a interação com o DM (arquiteto de software) durante o processo de otimização, sendo que arquiteto participa deste processo apenas antes (*a priori*) e depois (*a posteriori*) do mesmo. Neste sentido, e buscando seguir a vertente de pesquisas em SBSE que incluem o DM interativamente no processo evolutivo, foi feito um mapeamento sistemático (MS), o qual indicou não haver trabalhos que incluíam o DM interativamente em projetos de otimização de ALPS. Assim, este trabalho teve como primeiro objetivo propor uma modificação na abordagem MOA4PLA e ferramenta OPLA-Tool para permitir a interação com o arquiteto durante o processo de otimização. Além disso, a análise dos trabalhos resultantes do MS demonstrou que um problema frequentemente encontrado em processos interativos é o problema da fadiga, causado pelo

elevado número de interações. Para o tratamento deste problema, alguns autores indicam que a utilização de algoritmos de Aprendizagem de Máquina (AM) pode obter sucesso, visto o seu potencial de "aprender" sobre o comportamento do DM e, assim, substituí-lo no processo interativo.

Neste sentido, este trabalho teve também como objetivo a incorporação de um modelo de AM na ferramenta OPLA-Tool para o tratamento do possível problema da fadiga, já previsto em outros trabalhos que utilizam interação em SBSE.

Para a validação das modificações realizadas na abordagem MOA4PLA e ferramenta OPLA-Tool, foram realizados três experimentos (Exp 01, Exp 02, Exp 03).

O primeiro experimento (Exp 1) objetivou fazer uma análise comparativa entre o Modelo Atual (MA - sem interação com o arquiteto) e o Modelo Interativo Proposto (MIP). Para tanto, a ferramenta OPLA-Tool foi executada duas vezes (uma vez com cada modelo). Como resultados foi possível verificar que o modelo interativo era promissor, pois proporcionou o surgimento de soluções que, em sua maioria, superavam matematicamente (dominavam) as soluções do modelo atual. Além disso, a análise das avaliações (`escores`) dadas pelo arquiteto possibilitou verificar que as mesmas foram crescendo com o passar dos momentos interativos, sendo um indicativo de uma maior adequação das soluções para este arquiteto em particular. Com Exp 01, foi possível também aprender algumas lições em relação ao processo interativo. A quantidade de soluções avaliadas pelo arquiteto (36 soluções) foi considerada alta e, além disso, deveria haver um espaçamento maior entre uma interação e outra, ou seja, as soluções deveriam evoluir por algumas gerações antes de um novo momento interativo. Outro importante aprendizado com este experimento foi em relação à semelhança entre as soluções, de modo que a análise de todas as soluções em cada momento interativo era um esforço desnecessário.

Neste sentido, em Exp 02 um modelo de AM foi adicionado com dois objetivos. O primeiro deles foi o de possibilitar uma menor quantidade de soluções a serem avaliadas. O segundo, foi o de aprender a partir da experiência do arquiteto e, posteriormente, substituí-lo no processo de avaliação de ALPS.

Com a adição do algoritmo K-means, foi possível o agrupamento de soluções por semelhança, o que permitiu que o arquiteto pudesse avaliar uma solução de cada grupo (*cluster*) em vez de avaliar todas as soluções geradas.

Além disso foi introduzido também o algoritmo Multilayer perceptron (MLP) para que o mesmo pudesse, em tempo de processamento, aprender como o arquiteto avaliava as ALPS e, posteriormente, substituí-lo no processo de avaliação. Um problema encontrado para o funcionamento do algoritmo MLP foi em relação à pouca quantidade de dados que iriam compor o conjunto de treinamento do mesmo, visto que os dados seriam gerados

em tempo de processamento a partir de poucas ALPS avaliadas pelo arquiteto. Assim, três diferentes estratégias foram propostas com o intuito de se utilizar todos os dados (todas as ALPS) gerados em cada momento de interação. A análise quantitativa do Exp 02 possibilitou verificar que a Estratégia 1 obteve os melhores resultados, sendo então adotada definitivamente para o modelo de AM. Nesta Estratégia, a avaliação (`escore`) dada a uma ALPS pelo arquiteto é generalizada para todas as outras soluções do mesmo *cluster*.

O último experimento realizado (Exp 3) teve como objetivo testar MIP como um todo (interação com o arquiteto juntamente com o modelo de AM). Assim, o modelo interativo foi testado com a participação de 10 arquitetos de software para verificar a eficácia, tanto do processo interativo, como também do modelo de AM no que tange ao tratamento da fadiga e a capacidade do mesmo em substituir o arquiteto no processo de avaliação de ALPS.

Os resultados da análise quantitativa do Exp 03, feita a partir do cálculo das medianas dos `escores` dados por cada arquiteto em cada momento de interação, permitiu afirmar que o modelo de AM obteve bons resultados ao substituir o arquiteto no processo de avaliação. Isto pôde ser constatado pois a mediana dos `escores` das soluções restantes do processo de otimização, em geral, obtiveram um valor no mesmo nível (ou maior) em comparação com o último momento de interação.

A análise qualitativa do Exp 03 permitiu inferir que o modelo de AM foi apto em tratar o problema da fadiga, visto que apenas 20% dos participantes relataram sentir cansaço durante a realização do experimento. Além disso, em relação a adequação das soluções restantes às preferências de cada arquiteto, pôde ser constatado que a maioria dos participantes (50% concordância parcial e 20% concordância total) considerou que as soluções restantes do processo de otimização estavam adequadas aos seus perfis, corroborando com a análise quantitativa.

De uma forma geral, a análise dos três experimentos permitiu inferir que o processo interativo na abordagem MOA4PLA e ferramenta OPLA-Tool não só é viável, como também promissor. Além disso, pode-se destacar o fato de que, com os resultados trazidos por este trabalho, o estado da arte no que tange a processos interativos em SBSE pôde ser expandido, colaborando com a vertente de pesquisas que incluem o DM em processos de otimização. Ainda, haja visto os bons resultados trazidos pelo modelo de Aprendizagem de Máquina utilizado neste trabalho, o mesmo pode ser adaptado a outros contextos, visto o problema da fadiga ser algo recorrente em processos interativos.

7.1 Trabalhos futuros

Este foi o primeiro trabalho a contemplar a interação durante o processo de otimização realizado pela abordagem MOA4PLA e ferramenta OPLA-Tool. Desta forma, algumas lacunas de pesquisas surgiram a partir do desenvolvimento deste.

Na Seção anterior foram apresentadas algumas ameaças à validade do estudo, como por exemplo, a utilização apenas da ALPS AGM com três objetivos a serem otimizados. Neste sentido, novos estudos podem ser feitos utilizando-se outras ALPS, bem como selecionando-se uma maior quantidade de funções objetivo. Estas novas configurações podem ter impacto tanto no tempo de processamento como também na percepção de fadiga no arquiteto.

Como um primeiro estudo, o modelo de AM utilizou o algoritmo MLP para a predição das avaliações (escores dados pelo arquiteto). Assim, outros algoritmos podem ser testados para fazer um comparativo com o modelo utilizando MLP.

Além disso, ao se selecionar uma maior quantidade de funções objetivo, a quantidade de atributos da rede MLP também será aumentada e, consequentemente, o seu tempo de processamento. Este fato também pode ser considerado para avaliar a utilização de outros algoritmos de AM.

REFERÊNCIAS

AMAL, B., KESSENTINI, M., BECHIKH, S., DEA, J., SAID, L. B. On the use of machine learning and search-based software engineering for Ill-defined fitness function: a case study on software refactoring. In: International Symposium on Search Based Software Engineering. Springer, Cham, 2014. p. 31-45.

APEL, S., BATORY, D., KÄSTNER, C., SAAKE, G. Feature-oriented software product lines . Springer- Verlag Berlin An, 2016.

ARAÚJO, A. A., PAIXÃO, M. Machine learning for user modeling in an interactive genetic algorithm for the next release problem. In: International Symposium on Search Based Software Engineering. Springer, Cham, 2014. p. 228-233.

BECHIKH, S., KESSENTINI, M., SAID, L. B., GHÉDIRA, K. Preference incorporation in evolutionary multiobjective optimization: a survey of the state-of-the-art. In: Advances in Computers. Elsevier, 2015. p. 141-207.

BINDEWALD, C. V., FREIRE, W. M., AMARAL, A. M., COLANZI, T. E. Towards the support of user preferences in search-based product line architecture design: an exploratory study. In: Proceedings of the XXXIII Brazilian Symposium on Software Engineering. ACM, 2019. p. 387- 396.

BISHOP, C. M. Pattern recognition and machine learning. springer, 2006.

CARDOSO, M. P. S., LIMA, C., DE ALMEIDA, E. S., DO CARMO MACHADO, I., VON FLACH G CHAVEZ, C. Investigating the variability impact on the recovery of software product line architectures: an exploratory study. In: Proceedings of the 11th Brazilian Symposium on Software Components, Architectures, and Reuse. ACM, 2017. p. 12.

CHIDAMBER, S. R., KEMERER, C. F. A metrics suite for object oriented design. IEEE Transactions on software engineering, v. 20, n. 6, p. 476-493, 1994.

CHOMA NETO, J., GAIESKI, T., AMARAL, A. M., COLANZI, T. E. Quanti-Qualitative Analysis of a Memetic Algorithm to Optimize Product Line Architecture Design. In: 2018 IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI). IEEE, 2018. p. 498-505.

COCHRANE, J. L.; ZELENY, M. Multiple criteria decision making. Univ of South Carolina Pr, 1973.

COELLO, C. A. C., LAMONT, G. B., VAN VELDHUIZEN, D. A. Evolutionary algorithms for solving multi-objective problems. New York: Springer, 2007.

COLANZI, T. E. Uma abordagem de otimização multiobjetivo para projeto arquitetural de linha de produto de software. 2014. 215 f. Tese (Doutorado em Ciências da Computação) – Universidade Federal do Paraná, Curitiba.

COLANZI, T. E., VERGILIO, S. R. A feature-driven crossover operator for multi-objective and evolutionary optimization of product line architectures. Journal of Systems and Software, v. 121, p. 126-143, 2016.

COLANZI, T. E., VERGILIO, S. R. Representation of software product line architectures for search-based design. In: Proceedings of the 1st International Workshop on Combining Modelling and Search-Based Software Engineering. IEEE Press, 2013. p. 28-33.

COLANZI, T. E., VERGILIO, S. R., GIMENES, I., OIZUMI, W. N. A search-based approach for software product line design. In: Proceedings of the 18th International Software Product Line Conference-Volume 1. ACM, 2014. p. 237-241.

COSTA, G. C. B . Uma abordagem para linha de produtos de software científico baseada em ontologia e workflow. 2013. 105 f.. Dissertação (Mestrado em Ciências da Computação) – Universidade Federal de Juiz de Fora, Juiz de Fora.

DANTAS, A., YELSTSIN, I., ARAÚJO, A. A., Souza, J. (2015, September). Interactive

software release planning with preferences base. In International Symposium on Search Based Software Engineering (pp. 341-346). Springer, Cham.

DEB, K., PRATAP, A., AGARWAL, S., MEYARIVAN, T. A. M. T. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE transactions on evolutionary computation, v. 6, n. 2, p. 182- 197, 2002.

DURO, J. A., SAXENA, D. K., DEB, K., ZHANG, Q. Machine learning based decision support for many-objective optimization problems. Neurocomputing, v. 146, p. 30-47, 2014.

FÉDERLE, É. L., DO NASCIMENTO FERREIRA, T., COLANZI, T. E., VERGILIO, S. R. OPLA-Tool: a support tool for search-based product line architecture design. In: Proceedings of the 19th International Conference on Software Product Line. ACM, 2015. p. 370-373. Curitiba.

FERREIRA, T. N., VERGILIO, S. R., DE SOUZA, J. T. Incorporating user preferences in search- based software engineering: A systematic mapping study. Information and Software Technology, v. 90, p. 55-69, 2017.

FLEISS, J. L., COHEN, J. The equivalence of weighted kappa and the intraclass correlation coefficient as measures of reliability. Educational and psychological measurement, v. 33, n. 3, p. 613-619, 1973.

FONG, R. C., SCHEIRER, W. J., COX, D. D. Using human brain activity to guide machine learning. Scientific reports, v. 8, n. 1, p. 5397, 2018.

FRAKES, W. B., KANG, K. Software reuse research: Status and future. IEEE Transactions on Software Engineering, v. 31, n. 7, p. 529-536, 2005.

FREIRE, W. M., BINDEWALD, C. V., AMARAL, A. M. M., COLANZI, T. E. Supporting decision makers in search-based product line architecture design using clustering. In: 2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC). IEEE, 2019. p. 139-148.

HARMAN, M., CLARK, J. Metrics are fitness functions too. In: 10th International

Symposium on Software Metrics, 2004. Proceedings. Ieee, 2004. p. 58-69.

HARMAN, M. Search based software engineering for program comprehension. In: 15th IEEE International Conference on Program Comprehension (ICPC'07). IEEE, 2007. p. 3-13.

HARMAN, M., BURKE, E., CLARK, J., YAO, X. Dynamic adaptive search based software engineering. In: Proceedings of the ACM-IEEE international symposium on Empirical software engineering and measurement. ACM, 2012. p. 1-8.

HARMAN, M., JONEW, B. F. Search-based software engineering. Information and software Technology, v. 43, n. 14, p. 833-839, 2001.

HAYKIN, S. S. Neural networks and learning machines/Simon Haykin. New York: Prentice Hall,, 2009.

HINTON, G. E., SEJNOWSKI, T. J., POGGIO, T. A. (Ed.). Unsupervised learning: foundations of neural computation. MIT press, 1999.

KANG, K. C., COHEN, S. G., HESS, J. A., NOVAK, W. E., PETERSON, A. S. Feature-oriented domain analysis (FODA) feasibility study. Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst, 1990.

KNOWLES, J. D., CORNE, D. W. Approximating the nondominated front using the Pareto archived evolution strategy. Evolutionary computation, v. 8, n. 2, p. 149-172, 2000.

LITAO, Z., TIEJUN, W., XI, J., JIN, J. The machine learning classifier based on Multi-Objective Genetic Algorithm. In: Computing and Convergence Technology (ICCCT), 2012 7th International Conference on. IEEE, 2012. p. 405-409.

LOPEZ-HERREJON, R. E., ILLESCAS, S., EGYED, A. A systematic mapping study of information visualization for software product line engineering. Journal of Software: Evolution and Process, v. 30, n. 2, p. e1912, 2018.

MCGREGOR, J. D., NORTHROP, L. M., JARRAD, S., POHL, K. Initiating software

product lines. IEEE Software, v. 19, n. 4, p. 24-27, 2002.

MITCHELL, T. M. Machine learning. 1997. Burr Ridge, IL: McGraw Hill, v. 45, n. 37, p. 870-877, 1997.

MJOLSNESS, E., DECOSTE, D. Machine learning for science: state of the art and future prospects. science, v. 293, n. 5537, p. 2051-2055, 2001.

NGATCHOU, P., ZAREI, A., EL-SHARKAWI, A. Pareto multi-objective optimization. In: Intelligent Systems Application to Power Systems, 2005. Proceedings of the 13th International Conference on. IEEE, 2005. p. 84-91.

PARETO, V.; Politique, M. D.; Press, A. Paris. 1927.

PEDRO, L. R., TAKAHASHI, R. H. Decision-maker preference modeling in interactive multiobjective optimization. In: International Conference on Evolutionary Multi-Criterion Optimization. Springer, Berlin, Heidelberg, 2013. p. 811-824.

PRADO, A. D., LUCRÉDIO, D. Ferramenta MVCASE-Estágio Atual: Especificação, Projeto e Construção de Componentes. Sessão de ferramentas do XV Simpósio Brasileiro de Engenharia de Software (SBES 2001), p. 368-373, 2001.

QUEIROZ, PAULO GABRIEL GADELHA. Uma abordagem de desenvolvimento de linha de produtos com uma arquitetura orientada a serviços. 2009. 140 f.. Dissertação (Mestrado em Ciências da Computação e Matemática Computacional) – ICMC/USP, São Carlos.

QUIROZ, J. C., LOUIS, S. J., SHANKAR, A., DASCALU, S. M. Interactive genetic algorithms for user interface design. In: 2007 IEEE congress on evolutionary computation. IEEE, 2007. p. 1366- 1373.

RAMIREZ, A., ROMERO, J. R., SIMONS, C. A systematic review of interaction in search-based software engineering. IEEE Transactions on Software Engineering, 2018.

RUSSELL, S. J.; NORVIG, P. Artificial intelligence: a modern approach. Malaysia; Pearson Education Limited, 2016.

SAID, L. B., BECHIKH, S., GHÉDIRA, K. The r-dominance: a new dominance relation for interactive evolutionary multicriteria decision making. IEEE Transactions on Evolutionary Computation, v. 14, n. 5, p. 801-818, 2010.

SAMMUT, C., WEBB, G. I. (Ed.). Encyclopedia of machine learning. Springer Science Business Media, 2011.

SCHMID, K., VERLAGE, M. The economic impact of product line adoption and evolution. IEEE software, v. 19, n. 4, p. 50-57, 2002.

SEI Software Engineering Institute. Arcade Game Maker pedagogical product line, 2019. https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=485941.

SHACKELFORD, M. Implementation issues for an interactive evolutionary computation system. In: Proceedings of the 9th annual conference companion on Genetic and evolutionary computation. ACM, 2007. p. 2933-2936.

SHACKELFORD, M., SIMONS, C. L. Metaheuristic design pattern: Interactive solution presentation. In: Proceedings of the Companion Publication of the 2014 Annual Conference on Genetic and Evolutionary Computation. ACM, 2014. p. 1431-1434.

SIMONS, C., SINGER, J., WHITE, D. R. Search-based refactoring: Metrics are not enough. In: International Symposium on Search Based Software Engineering. Springer, Cham, 2015. p. 47- 61.

SINGH, A., MINSKER, B. S., BAJCSY, P. Image-based machine learning for reduction of user fatigue in an interactive model calibration system. Journal of Computing in Civil Engineering, v. 24, n. 3, p. 241-251, 2009.

SU, J., ZHANG, S. Research on product shape innovation design method with human-computer interaction through genetic algorithm. In: Computer-Aided Industrial Design Conceptual Design (CAIDCD), 2010 IEEE 11th International Conference on. IEEE, 2010. p. 301- 305.

TAKAGI, H. Interactive evolutionary computation: System optimization based on human

subjective evaluation. In: IEEE Int. Conf. on Intelligent Engineering Systems (INES'98). 1998. p. 17-19.

TAKAGI, H. Active user intervention in an EC search. In: Proceedings of the Fifth Joint Conference on Information Sciences, JCIS 2000. 2000. p. 995-998.

TAKAGI, H. Interactive evolutionary computation: Fusion of the capabilities of EC optimization and human evaluation. Proceedings of the IEEE, v. 89, n. 9, p. 1275-1296, 2001.

VAN DER LINDEN, F. J., SCHMID, K., ROMMES, E. Software product lines in action: the best industrial practice in product line engineering. Springer Science Business Media, 2007.

VERDECIA, Y. D., COLANZI, T. E. Correlation between Similarity and Variability Metrics in Search-based Product Line Architecture: Experimental Study and Lessons Learned. In: ICEIS (2). 2017. p. 533-541.

VERGILIO, S. R., COLANZI, T. E., POZO, A. T. R., ASSUNCAO, W. K. G. Search based software engineering: A review from the Brazilian symposium on software engineering. In: 25th Brazilian Symposium on Software Engineering. IEEE, 2011. p. 50-55.

WARE, M., FRANK, E., HOLMES, G., HALL, M., WITTEN, I. H. Interactive machine learning: letting users build classifiers. International Journal of Human-Computer Studies, v. 55, n. 3, p. 281-292, 2001.

WITTEN, I. H., FRANK, E., HALL, M. A., PAL, C. J. (2016). Data Mining: Practical machine learning tools and techniques. Morgan Kaufmann.

YOO, S., HARMAN, MARK. Pareto efficient multi-objective test case selection. In: Proceedings of the 2007 International Symposium on Software testing and analysis. ACM, 2007. p. 140-150.

ZHANG, J., ZHAN, Z. H., LIN, Y., CHEN, N., GONG, Y. J., ZHONG, J. H.,SHI, Y. H. Evolutionary computation meets machine learning: A survey. IEEE Computational

Intelligence Magazine, v. 6, n. 4, p. 68-75, 2011.

ZITZLER, E., THIELE, L., LAUMANNS, M., FONSECA, C. M., DA FONSECA, V. G. Performance assessment of multiobjective optimizers: An analysis and review. IEEE Transactions on evolutionary computation, v. 7, n. 2, p. 117-132, 2003.

A

Protocolo de Mapeamento Sistemático

A.1 Planejamento

Questões principais:

- Quais são os problemas de SBSE solucionados utilizando interação homem computador durante o processo de otimização?
- Quais métodos e técnicas são empregados para suportar a interação homem computador no processo de otimização em SBSE?

População: Engenharia de Software e Otimização de ALPS.

Intervenção: Aplicação de métodos e técnicas empregados para suportar a Interação Homem Computador no processo de otimização utilizando-se SBSE em ALPS.

Controle: Em SBSE a utilização da Interaçõe Homem-Computador no processo de otimização de ALPS é algo um tanto novo, de modo que não se encontram trabalhos específicos nesta área para que se possam ser utilizados como conjuntos de dados iniciais. Assim, visa-se com este trabalho clarear como pode se dar a interação homem-computador para a otimização de projetos de ALPS.

Resultados: Com o término da revisão sistemática, espera-se encontrar trabalhos que

possam dar indícios de como poderia ser incorporada Interação Homem-Computador em processos de otimização em SBSE. Assim, de uma forma geral, com a revisão sistemática pretende-se:

- Elucidar o modo como é realizada a Interação Homem-Computador no contexto de SBSE;
- Compreender o processo de otimização em projetos de ALPS;
- Extrair informações úteis do processo de Interação Homem-Computador em processos de otimização;
- Aprofundar teoricamente sobre a temática exposta nas questões (seção 1) para que possa servir como base para o desenvolvimento da dissertação.

Critérios de seleção para definição de Fontes: Para se definir as bases de dados de onde serão extraídos os trabalhos que servirão como base para o presente estudo, utilizou-se como critério bibliotecas em que se encontram a maior parte dos trabalhos relevantes dentro da Ciência da Computação.

Estudo nas fontes de pesquisa: A pesquisa foi executada por meio de buscas, utilizando-se as ferramentas de pesquisa oferecidas em cada biblioteca, além da seleção e classificação dos estudos retornados em cada base de dados. Dessa forma, a seleção dos trabalhos se deu a partir de diferentes etapas, sendo elas:

- **Etapa 01 (resultados da busca)** - Buscou-se pelas palavras chaves e sinônimos nas seções: título, abstract e palavras-chaves por meio das ferramentas de filtragem oferecidas pelas bibliotecas digitais selecionadas.
- **Etapa 02 (1ª. Seleção) -** Leitura do título e abstract para identificar a relevância dos estudos encontrados no contexto das perguntas do mapeamento sistemático, utilizando-se os critérios de seleção.
- **Etapa 03 (2ª. Seleção)** - A partir dos trabalhos selecionados na Etapa 02, realizou-se a leitura da introdução e conclusão para se ter uma ideia mais concreta da utilidade (contribuição) que o trabalho poderia trazer.
- **Etapa 04 (seleção final)** -Para esta etapa, os trabalhos considerados como relevantes na Etapa 3 foram lidos em sua íntegra, buscando-se por informações que viessem a ajudar no presente trabalho. Para ser considerado relevante, um

estudo deveria ter ao menos um critério de inclusão e nenhum critério de exclusão.

Palavras-chave:

- *Software product line*
- *Search-based Software Engineering*
- *Machine learning*
- *Human-computer interaction*

Critérios de Seleção:
1 - Inclusão: Trabalhos publicados no período de 2005 a 2015.
2 - Inclusão: Utilização de interação homem-computador em processos de otimização
3 - Inclusão: Utilização de interação homem-computador em processos de otimização com base em SBSE
4 – Inclusão: Aprendizagem de máquina com a utilização de interação homem-computador
6 - Exclusão: Trabalhos que não estiverem em sua íntegra
7 - Exclusão: Trabalhos que não estejam disponibilizados em inglês ou português.

A.2 Execução

De acordo com os critérios estabelecidos na seção "Critérios de Seleção", foram selecionadas as seguintes bibliotecas digitais para busca dos trabalhos:

- Science Direct
- ACM
- Springer
- IEEE
- Research Gate

Para a construção das strings de busca, em um primeiro momento, utilizou-se as palavras-chave conjuntamente. Contudo, como os resultados foram escassos, optou-se então pela "quebra" das strings, gerando-se assim as seguintes strings:

- *("search-based software engineering"AND "human–computer interaction")*

- *("search-based software engineering"AND "software product line")*
- *("search-based software engineering"AND "machine learning")*
- *("human–computer interaction"AND "software product line")*
- *("human–computer interaction"AND "machine learning")*
- *("software product line"AND "machine learning")*
- *("human–computer interaction"AND "machine learning"AND "search-based software engineering"*

Tipos de estudos: O desenvolvimento de leitura e análise dos trabalhos selecionados como relevantes pela pesquisa encontrou: artigos científicos, monografias, dissertações, teses e relatórios técnicos.

Seleção inicial: Para documentar os estudos encontrados nas buscas realizadas nas bibliotecas digitais, inicialmente foram guardados os seguintes dados: autores, título, palavras-chave, abstract, ano e local de publicação, e posteriormente, para os trabalhos selecionados, a obra completa foi armazenada. Os trabalhos retornados nas buscas foram avaliados por duas pessoas. Primeiramente as decisões de inclusão e exclusão foram tomadas e justificadas pelo autor do mapeamento e, em um segundo momento, tais decisões foram validadas pela supervisora da pesquisa, professora Dra. Aline Maria Malachini Miotto Amaral

Avaliação da qualidade dos estudos: A qualidade dos estudos selecionados foi medida pelos critérios de inclusão e exclusão. Os trabalhos selecionados devem conter no mínimo um critério de inclusão e nenhum critério de exclusão.

Extração de dados: Inicialmente somente os dados de identificação dos estudos retornados pelas bibliotecas digitais foram armazenados, tais como: autores, título, palavras-chave, abstract, ano e local de publicação. A partir desses dados, com a utilização dos critérios de inclusão/exclusão, chegou-se aos trabalhos que deveriam ser salvos por completo. Do conjunto de trabalhos, os dados iniciais servirão de base para se compor a análise quantitativa do mapeamento, ao passo que os demais (trabalhos completos) serão utilizados para a análise qualitativa.

Resultados esperados e sumarização: Ao fim do mapeamento sistemático, espera-se

realizar tanto uma classificação quantitativa, apresentando as perspectivas relacionadas a quantidade de estudos encontrados, períodos de maior publicação, autores relacionados a esse foco de estudo, referências mais utilizadas, entre outras classificações, e, também, realizar uma classificação qualitativa dos resultados encontrados, podendo-se levar a encontrar as respostas aos questionamentos propostos ou não, bem como, trazendo-se conceitos que auxiliarão no entendimento do atual estado da arte para a temática em foco.

B

Artigo publicado no SBES 2019

Towards the support of user preferences in search-based product line architecture design: an exploratory study

Carlos Vinicius Bindewald, Willian M. Freire, Aline M. M. Miotto Amaral, Thelma Elita Colanzi
Informatics Department, State University of Maringa (UEM)
Maringa, Parana, Brazil
viniciusbindewald@hotmail.com,willianmarquesfreire@gmail.com,ammmamaral@uem.br,thelma@din.uem.br

ABSTRACT

Software Product Lines (SPLs) is a reuse approach in which a family of products is generalized in a common architecture that can be adapted to different clients. The Product Line Architecture (PLA) is one of the most important artifacts of a SPL. PLA design requires great human effort as it involves several factors that are usually in conflict. To ease this task, PLA design can be formulated as an optimization problem with many factors, i.e, as a multi-objective optimization problem. In this context, the MOA4PLA approach was proposed to optimize PLA design using search algorithms and metrics specific to the context. This approach supported by OPLA-Tool has already been used in several works demonstrating its applicability. However, MOA4PLA does not take into account aspects that are subjective, such as the preferences of a particular Decision Maker (DM). To do so, this paper presents a proposal to incorporate the user preferences in the optimization process performed by MOA4PLA, through an interactive process in which the DM subjectively evaluates the solutions in processing time. Thus, the solutions generated can be better suited to the DM's needs or preferences. In order to allow the user interaction, modifications were made in MOA4PLA and implemented in the OPLA-Tool. Aiming at an initial validation of the proposal, an exploratory study was carried out, composed of two experiments: a qualitative and a quantitative. These experiments were realized with the participation of a software architect. Empirical results pointed out that the proposed interactive process enables the generation of PLAs that are in accordance with the architect's preferences. Another significant contribution are the lessons learned on how to improve the interactive process.

CCS CONCEPTS

• **Software and its engineering → Search-based software engineering**; **Software product lines**; • **Applied computing → Multi-criterion optimization and decision-making**.

KEYWORDS

Product Line Architecture, Multi-Objective Optimization, Human-computer interaction.

ACM ISBN 978-1-4503-7651-8/19/09…$15.00
https://doi.org/10.1145/3350768.3351993

ACM Reference Format:
Carlos Vinicius Bindewald, Willian M. Freire, Aline M. M. Miotto Amaral, Thelma Elita Colanzi. 2019. Towards the support of user preferences in search-based product line architecture design: an exploratory study. In *XXXIII Brazilian Symposium on Software Engineering (SBES 2019), 2019, Salvador, Brazil.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3350768.3351993

1 INTRODUCTION

Search Based Software Engineering (SBSE) is a research field and practice in which computational search, as well as the optimization techniques most commonly associated with operational research, are used to solve complex problems in software engineering [12].

The techniques used in SBSE contribute to the reduction of development costs and efforts, since the solutions found satisfy constraints that are usually in conflict and, in general, are very difficult to obtain by software engineers [15].

In Software Product Line (SPL) engineering, thinking about many of the problems as optimization problems is natural, because the wide variability of different products expressed by their feature models, creates a large and rich search space in which we can seek optimal (or near optimal) choices of products [13]. In this context, several problems related to SPL Engineering have also been solved by SBSE techniques [13, 17], such as SPL testing, feature model selection and reverse engineering of feature models. Some other research topics are emergent and have open questions, such as Product Line Architecture (PLA) design [8] and re-engineering of legacy applications into SPL [3].

A PLA design encompasses the components realizing all the mandatory and varying features in a domain [26]. It is often attractive and advantageous to interpose a PLA in-between the feature model and the products constructed from it, because the PLA captures implementation concerns, with traceability links to the feature model and products [13]. A PLA is useful if it is designed by architects in a proactive approach or even when it is discovered from software variants (extractive approach).

However, obtaining a modular, extensible and reusable PLA can be a non-trivial task [8] because it involves several factors (objectives), often conflicting with each other, thus requiring a great human effort. In this context, the Multi-Objective Approach for PLA Design (MOA4PLA) [8] was proposed to optimize architectural properties of a PLA design using SBSE techniques and metrics specific for SPL. Metrics are used in the objective functions to evaluate each solution generated by MOA4PLA. This approach and OPLA-Tool [10], which provides its automation, have already been used in several studies demonstrating their applicability [7, 18]. Besides, they can be used either in a proactive approach or to optimize properties of a PLA design extracted from products variants.

The MOA4PLA approach does not take into account aspects that are subjective, such as the preferences of a particular Decision Maker (DM). Ferreira et al. [11] and Simons, Singer and White [24] indicate that, in some cases, the obtained solutions in SBSE are rejected by the DM because many aspects of the problem cannot be mathematically modeled. In this context, Bechikh et al. [4] point out that the main objective in a multi-objective optimization is to find an approximation with good Pareto Front convergence and distribution, and from which the DM can subsequently select his/her preferred alternative. However, in general, DMs are not interested in discovering the whole Pareto Front, but rather the region that best meets their preferences, that is, their region of interest [4]. Thus, it is noticed that it is necessary to incorporate in a multi-objective optimization other aspects that are inherent to the DM, such as their individual preferences.

Considering that the DM's preferences were not explored in the PLA design optimization, this work presents a proposal to incorporate DM preferences into the MOA4PLA approach, so that, at the end of the optimization process, the PLA architect[1] can obtain PLA solutions that are better suited to his/her preferences. In this way, this study aims to implement a human-computer interactive process for PLA design, seeking to answer the following research questions:

RQ1: *Are the best architect-evaluated PLAs the best PLAs according to their fitness?*

RQ2: *Does the proposed interactive model for search-based PLA design enable the generation of PLA design alternatives that satisfy the architect's preferences?*

The interactive model proposed in the MOA4PLA approach was implemented in the OPLA-Tool, and then an exploratory study was carried out in order to validate the proposal. Such exploratory study includes two experiments performed for the same original PLA design. The first experiment was designed using the current approach, without interaction. The second one used the proposed interactive model, in which the optimization process was paused in certain moments and the solutions generated until that moment were shown to the architect for evaluation.

The main contributions of this work are: (i) the discovered evidence that the proposed interactive model enables the generation of PLAs that are likely to the architect individual preferences, (ii) empirical results show that the statement of Simons, Singer and White [24] that metrics are not enough to evaluate the obtained solutions is also valid to the PLA design context, (iii) lessons learned about how the proposed interactive model can be improved. While we have proposed the interactive model for MOA4PLA and implemented it using OPLA-Tool, the contributions are useful for different search-based PLA design approaches because they are related to the interactive process and the self-sufficiency of metrics to evaluate PLA design.

This paper is organized as follows: Section 2 presents the main concepts involved in this work. Section 3 addresses related work to this research subject. Section 4 introduces the proposed interactive model. Section 5 describes the exploratory study performed. Sections 6, 7 and 8 present the obtained results, the answers to the research questions and the lessons learned, respectively. Finally, Section 9 concludes the paper and points out to future works.

[1] In this work the PLA architect plays the Decision Maker role.

2 BACKGROUND

This section presents some basic concepts related to SBSE, as well addresses the MOA4PLA approach and OPLA-Tool.

2.1 Search Based Software Engineering

Search Based Software Engineering, a term coined by Harman and Jones [14] that represents a research field in which search-based optimization is applied to Software Engineering, has proved to be very successful and generic. The problems addressed in SBSE are usually complex and difficult to solve because they involve several variables. SBSE seeks to formulate the problems of Software Engineering as optimization problems [14, 15]. The term optimization here refers to the search for optimal or almost ideal solutions in a search space of candidate solutions, guided by an evaluation function (also called as objective function or fitness function) that distinguishes between better and worse solutions.

Next, we describe some concepts related to SBSE and that are mentioned in this work, such as: genetic algorithms and multi-objective optimization.

Genetic algorithms (GAs) are part of evolutionary computation and are inspired by the theory of natural selection and genetic evolution [6]. GAs are considered an efficient search method based on natural and genetic selection principles, such as selection, crossover, and mutation operators to develop a population. According to Harman et al. [15] [13], GAs have been used successfully to find solutions to difficult problems in Software Engineering, such as software testing, refactoring and PLA design. Some of these problems involve finding solutions to problems involving more than one goal, called multi-objective. For these cases some GAs have undergone adaptations. In this work, one of these GA adaptations is used, called Non-dominated Sorting Genetic Algorithm II (NSGA-II) [9].

Multi-objective optimization problems are impacted by several factors, each one represented by an objective function. Thus, a simultaneous optimization of these objectives is required, which may be dependent, independent, cooperative or competing [6]. According to Coello et al. [6] when there is more than one objective function, the problem has not only one optimal solution, but a diverse set of optimal solutions.

The several optmal solutions represent the trade-off between the defined objectives. They are called non-dominated and form the Pareto front [19]. Hence, an optimization problem consists in finding a set of solutions ($PFapprox$) that is the nearest to the Pareto front. $PFapprox$ is composed of different non-dominated solutions. Given a solution set, a solution A dominates a solution B if the value of at least one objective of A is better than the respective objective value of B and the remaining objective values of A are not worse than the respective values of B. A is non-dominated if it is not dominated by any other solution.

Multi-objective problems in Software Engineering can often be affected by competing or conflicting factors, that is, the optimization of one factor can harm the other. In this way, the aim of a multi-objective optimization is to find a balance of interests, a better trade-off, among the considered factors [15].

According to Bechick et al. [4], the main goal of multi-objective optimization is to find a set of solutions with good convergence

and good distribution in the Pareto Front, and from which the DM choose his/her preferred alternative.

2.2 MOA4PLA Approach

MOA4PLA approach was developed to support the work of the PLA architect (Figure 1). Due to space limitations, we already present in Figure 1 an adaptation of MOA4PLA approach including the architect's evaluation input. This input was included in the approach in order to support the interactive optimization process. Thus, the architect's evaluation input will be explained in Section 4, which discusses the MOA4PLA proposed modifications. In a PLA design there are several factors involved, so this activity demands great human effort. In this context, the approach deals with the PLA design as an optimization problem with several objectives from which there may not be a single possible solution, but rather a set of solutions that present the best compromise among the objectives involved. Thus, MOA4PLA is a systematic approach that uses multi-objective search algorithms to evaluate and improve PLA design considering feature modularization, SPL extensibility, understandability, changeability and conventional architectural properties, such as coupling and cohesion [8, 27].

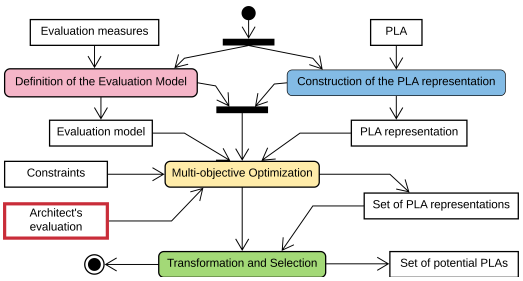


Figure 1: MOA4PLA approach, adapted from [8]

MOA4PLA performs the following activities:

-**Construction of the PLA Representation** - This activity receives as input the PLA design (modeled in an UML stereotyped class diagram containing the SPL variabilities) used by the search algorithm as a starting point for the evolutionary process. Due to the lack of space we do not include a complete example of PLA, but excerpts from PLA AGM used in the experiments of this work are presented in figures 5 and 6. The output of this activity is the PLA representation according to MOA4PLA metamodel [8].

-**Definition of the Evaluation Model** - The definition of the measures that should to be used in the evaluation model for the optimization process provided by the architect.

-**Multi-Objective Optimization** - the PLA representation obtained in the first step is optimized according to the constraints provided by the architect. Each obtained alternative PLA is evaluated following the evaluation model defined in the previous activity. A set of PLA representations is generated as output. Different search-based algorithms can be used in this activity, such as GAs and MOEAs. The constraints and search operators included in MOA4PLA are described in [8]. -**Transformation and Selection** - Finally, each PLA representation, from the set of solutions obtained in the last step, is converted to a class diagram containing the PLA design. So, the architect must select one alternative solution, according to the his/her priorities or goals.

For the definition of the evaluation model activity, the architect, in accordance with the SPL needs, choose objective functions (which are composed of software metrics) that meet these needs. These functions are the goals to be optimized by the approach. Currently, the evaluation model of MOA4PLA has 17 objective functions to measure architectural properties related to modularity, extensibility, changeability and understandability [27].

In previous study [18] three objective functions were considered important by the architects. In this way, these three objective functions were chosen to be used in the exploratory study carried out in this work, which are described below [27]:

-**FM:** it provides indicator about feature modularization of a PLA design in terms of feature scattering, feature interlacing and feature-driven cohesion.

-**ACLASS:** it measures the number of architectural elements that depend on other classes of the design plus the number of elements on which each class depends.

-**COE:** it measures the cohesion of the PLA design in terms of internal relationship between classes of the PLA design.

According to Colanzi et al. [8], the main focus of MOA4PLA is the evaluation and improvement of PLA design, considering multiple objectives (metrics), regardless of the search algorithm adopted. Therefore, it is a generic approach with regard to the multi-objective algorithm to be employed. Nowadays, the MOA4PLA allows the architect interaction *a priori* (before initializing the process) and *a posteriori* (after finalizing the process). In the *a priori* process, the architect defines the settings used in the evolutionary process. The process *a posteriori* represents the presentation of the optimized solutions for the architect. In this way, the approach does not allows the architect interaction during the evolutionary process. In this context, the proposal described in this paper has the aim to allow the architect interaction also during the evolutionary process.

2.3 OPLA-Tool

OPLA-Tool [10] was developed to automate the application of MOA4PLA using multi-objective evolutionary algorithms. It is composed by six modules. These modules are briefly explained next.

-**OPLA-GUI** - Interface module for architect interaction. In this module the architect selects the PLA that should be optimized; the multi-objective algorithm that will be used; as well as the fitness functions and search operators;

-**OPLA-Encoding** - This module receives the original PLA design and converts this design in an initial solution (PLA representation) to be optimized by OPLA-Core;

-**OPLA-Core** - It is the main OPLA-tool module. It has as input the PLA representation generated by the OPLA-Encoding module and in sequence executes the evolutionary process through multi-objective algorithms. Two multi-objective evolutionary algorithms are implemented in the OPLA-Tool: NSGA-II [9] and PAES [16]. As an output of the optimization, a set of solutions is provided;

-**OPLA-Decoding** - This module receives as input the output of the OPLA-Core module, decoding the solutions and converting them into a graphical representation readable to the architect, modeled in class diagrams;

-**OPLA-Patterns**- This module has a design pattern application operator named Design Pattern Mutation Operator. This operator is responsible for analyzing whether a particular region of the

solution has a suitable scope for applying a design pattern and, if so, performs the application;
-**OPLA-ArchStyles-** Module with mutation operators to identify and preserve the architectural styles applied to the PLA design.

The development of OPLA-Tool made it possible to evaluate the effectiveness of the MOA4PLA approach. Nowadays, OPLA-Tool allows the same interaction moments described in Section 2.2 (*a priori* and *a posteriori*). Thus, OPLA-Tool also should be modified in order to support the modifications proposed in this work aiming to allow the architect interaction during the evolutionary process.

3 RELATED WORK

To propose the model in this work a systematic review was realized [1]. Thus, some papers contributed to build the model presented in the in the Section 4.

Ramirez et al. [20] highlights the amount of interactions as an important aspect regarding the interaction in SBSE. In general, the models presents a fixed number of interactions. However, there are also works in which the amount of interactions and timing are selected by the DM. In Araujo and Paixao [2], the model was proposed in this way, with the DM choosing the amount of interactions. In order to follow the ideas proposed in the papers, in the model presented in the Section 4, the architect can choose the amount of interactions and also the moment of the first interaction.

Another aspect found in the works is related to sequency of interactions. In Su and Zhang [25], DM interaction occurs in every new generation of results obtained by means of "crossover" and "mutation". This is similar with the Araujo and Paixao [2] paper's, that is, with each new generations of individuals, the interactive process happens. In this way, as a first experiment, we puts the architect to interact with the system in each new generations of individuals found.

In order to introduce the interactive process in SBSE, some papers report the utilization of some variations of genetic algorithms, called IGA (Interactive Genetic Algorithm) [11]. More especifically, Bechikh et al. [4] report the modification of a well-known evolutionary algorithms such as NSGA-II. As described previous, the OPLA-Tool currently uses two algorithms, being one of them, NSGA-II. In this way, as a first experiment, we chosen to introduce some modifications in this algorithm, adapted for our especific case.

The works selected in the systematic review, [2, 11, 20, 22, 23], also mentions about an important aspect in interactive processes, the fatigue problem, cause by many interactions with the systems. Considering this, the experiment configurations, Section 5, was chosen in order to avoid the mentioned problem.

It is important to note that, the analysis of the different mentioned works allowed to verify that the human interaction in evolutionary computation is not yet used in the PLA design context. As mentioned before, the PLA design is a non-trivial task that involves several factors and require great human effort. In this way, it is important to explore human interaction in this context as well.

4 PROPOSED MODEL FOR INTERACTIVE SEARCH-BASED PLA DESIGN

As previously discussed, to include the user preferences during the optimization process can produce solutions that better suite to their needs. In this context, this work proposes an interactive model for Search Based PLA Design. This model, presented in this section, is an adaptation of the MOA4PLA approach.

MOA4PLA approach and hence the OPLA-Tool tool do not currently allow the DM interaction during the evolutionary process. It is important to emphasize that, in PLA design, the DMs play the role of architects and hereafter they are called architect. In order to support the architect interaction, new data must be incorporated into the optimization process. These data refer to subjective assessment by the architect for the PLAs generated during processing, and were introduced in the MOA4PLA approach by the Architect's evaluation input data (Figure 1). This entry provides a new input data which directly impacts the Multi-objective Optimization activity.

To implement the modifications incorporated in MOA4PLA approach, OPLA-Tool should be also modified. As described previously, OPLA-Tool provides two algorithms for the PLAs optimization process, NSGA-II and PAES. In previous section was mentioned that some works reported modifications in NSGA-II to allow architect interaction during processing. The analysis of these NSGA-II modifications showed that these modification could not be used without adaptation for our specific case. In this way, and based on the NSGA-II modifications proposed in [2], it was decided to introduce modifications in the original NSGA-II algorithm in order to allow the architect interaction adapted for our specific context.

The model proposed by Araujo and Paixao [2] has as input data the settings used in the evolutionary process and the number of interactions that the DM wishes to execute. After beginning the evolutionary process, with each new generation, the best individuals are selected and the DM provides a score, that is, a subjective evaluation in a numerical interval previously defined. The solution under evaluation receives the highest score when fully satisfies the DM's preferences and minimum when it is completely different from what he/she expected.

For our specific case the process described above is incorporated in OPLA-Tool. However, another input data is also required from the architect, the moment in which the first interaction must happen, that is, only after a determined number of generations chosen by the architect the first interaction begins. This is necessary because it was observed that in the early generations it were not obtained good PLA designs.

In this way, it is important to highlight that besides the input data (configurations) already used in the original version of OPLA-Tool two new data are necessary to support the interactive model, the number of interactions and the moment of the first interaction.

The configuration parameters of the genetic algorithm (GA), the number of interactions and the moment of the first interaction are inputs provided by the user and all are independent of each other.

Furthermore, was decided that the worst evaluated solutions (score = 1) would be discarded, because it is far away of the architect's preferences and it would not be worth continuing its evolution and, also, the best evaluated solutions (score = 5) would be preserved in their current stage, because those solutions would already meet the architect's preference. The solutions with other scores (score = 2, 3 or 4) returned to the optimization process. Figure 2 demonstrates the flow chart for the interactive process proposed.

In this figure can be seen the following steps:

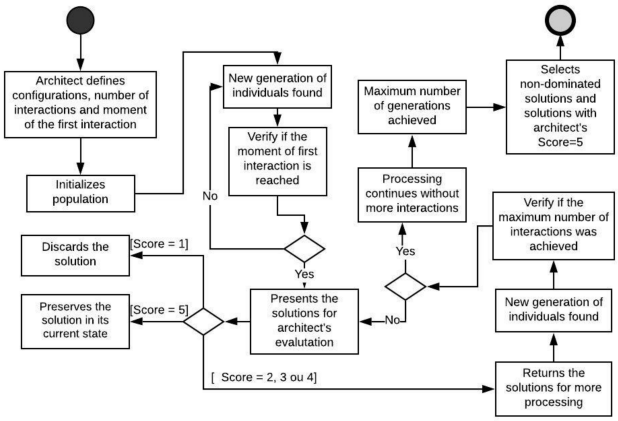


Figure 2: Flow chart for the interactive process

- **At the beginning**, the architect defines the MOA4PLA standard configurations, number of interactions and also moment of the first interaction;

- **After the algorithm starts** its execution with the initial population, following with each new generation of individuals, it is verified if the number of generations chosen by the architect for the first interaction has already been reached. In affirmative case, the solutions generated until that moment are shown to the architect to be evaluated. Solutions with score = 1 are discarded, solutions with score = 5 are preserved and solutions with score = 2, 3 or 4 return to the processing to continue their evolutionary processes;

- **In the sequence, after a new generation of individuals**, it is checked whether the number of interactions has already been reached. In negative case, the new generation of solutions is presented to the architect to be evaluated. In affirmative case, the evolutionary process continues without interaction until the number of generations established initially is reached. All solutions presented to the architect had the same number of GA evolution and started the interaction from the same moment.

- **Finally**, the non-dominated solutions and the solutions with score = 5 are selected, saved and presented to the architect. This occurs as soon as the number of fitness ratings stipulated by the architect is reached.

Figure 3 shows the new interface created in OPLA-Tool to allow the evaluation of PLAs solutions by the architects.

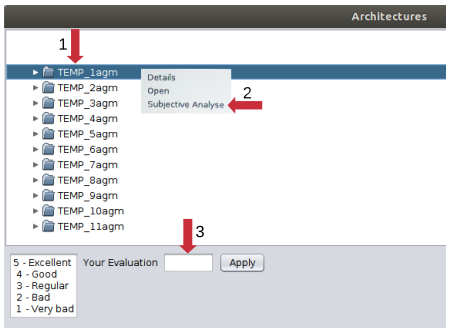


Figure 3: Interface created in the OPLA-Tool

After a specified number of generations (moment of the first interaction), this interface appears to the architect presenting the list of solutions generated until that moment (red arrow number 1). It's important to highlight that each solution evaluation is independent and does not depend on the order of their generation. Thus, on right click in the solutions, the architect has three options: Detail, Open and Subjective Analysis (red arrow number 2). In "Details", some solution-specific information is displayed to the architect. "Open" option open up the .UML file in the Papyrus tool[2] so that the architect can evaluate the solution. The last option, "Subjective Analyze", shows a field at the bottom of the screen where the architect can place his/her evaluation, with a value in the range 1-5 (red arrow number 3).

The computational complexity of the proposed model is related to the complexity of multi-objective GAs. The OPLA-Tool has already been used to optimize PLAs in real contexts, since experiments with the real SPL BET, for example, have already been successfully performed [7, 8, 18]. In an earlier study [8], the execution time for academic PLAs, such as AGM, was about 2 min and for BET less than 10 min.

5 EXPLORATORY STUDY DEFINITION

As an initial study, we performed an exploratory study using one case study - the SPL Arcade Game Maker (AGM,[21]). AGM is an SPL that includes three games: Brickles, Bowling and Pong, developed by the Software Engineering Institute (SEI). The rationale for choosing this case study was due to its relatively short processing time by the OPLA-Tool (around 5 minutes), which could facilitate the interactive process.

Figure 4 presents the protocol of the exploratory study realized. This protocol is composed by two different analysis method. These methods are described in details in Sections 5.1 and 5.2.

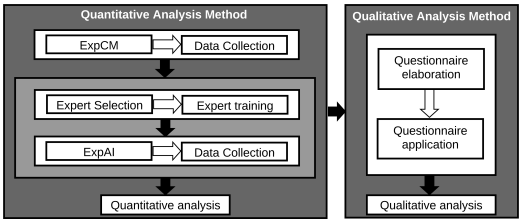


Figure 4: Exploratory study protocol

5.1 Quantitative Analysis Method

The quantitative method involved the execution of two experiments (ExpCM and ExpAI). The first one, ExpCM (CM- Current Model), was performed with OPLA-Tool in its current development stage, in which the architect does not have the option to interact with the tool during the evolutionary process.

The second experiment, ExpAI (AI - Architect Interaction), was carried out with the model proposed in this work (Section 4), which allows the interaction of the architect for the evaluation of PLAs generated during the evolutionary process. In ExpAI was necessary additional tasks, since in this experiment the architect participates in the interactive process, thus activities for his/her selection and training were realized as described below.

Expert selection: The choice of the architect to participate in the experiment took into consideration his/her educational level and PLA design knowledge. Thus, the architect chosen is a PhD candidate and has high knowledge in PLA design analysis;

[2]https://www.eclipse.org/papyrus/

Expert training: For the execution of the experiment, the architect received training to understand how the interactive process works in OPLA-Tool and how to put the PLA score;

For both experiments the values of the configuration parameters used were: number of the algorithm execution: 1; number of generations: 20; initial population: 12 individuals; and mutation operator with probability = 0.9. Additionally for ExpAI the number of interactions an moment of first interaction must be defined. These value was defined considering the amount of time necessary to execute ExpAI. Besides, the architect fatigue was also taking into account, since evaluate PLA design is a task that demands high intellectual effort, and the architect must evaluate all the generated solutions in each interaction. Thus the new parameters used were: number of interactions = 5 and moment of the first interaction = 5.

In order to compare quantitatively the results obtained in ExpCM and ExpAI several data were analysed, such as: the fitness of the obtained solutions; the set of solutions that composes the (PF_{true}) [28]; and the Euclidean distance to ideal solution (ED) [5] of each obtained solution.

The reference front (PF_{true}) is the set of all non-dominated solutions obtained in both experiments and the ED measures the distance between a solution *n* and the ideal solution in the solutions space. For this, the ideal solution has the minimum value obtained for each objective, considering a minimization problem. Solutions with low ED are better because they have the best trade-off between the optimized objectives.

Several quantitative data were collected for both experiments, such as: number of solutions generated, fitness values of the solutions; and for each solution generated was identified: number of packages and classes, dependencies, associations and relationships. Besides, during ExpAI the score given to each PLA design by the architect was collected for further analysis.

The number of solutions generated in each experiment was necessary to verify if ExpAI can produce more and different solutions from ExpCM. The solutions fitness values were used to analyse the impact of the interactive process in calculating the ED of the generated solutions and also to produce the solutions that are part of (PF_{true}) in each experiment. The remaining data was necessary to analyse in details the differences between the solutions generated in both experiments.

5.2 Qualitative Analysis Method

To complement the quantitative analysis was carried out with the architect that participates in ExpAI a qualitative experiment. This experiment aimed to clarify which aspects an architect considers important when evaluating a PLA, and if the generated PLA solutions meet their needs. To conduct this experiment a questionnaire (presented above) was prepared and applied to the architect. His/her responses were collected for further analysis. These activities are described bellow.

Questionnaire elaboration and Questionnaire application: Elaboration of a questionnaire to clarify which aspects an architect considers important when evaluating a PLA. In this way, the architect participated in the interactive process (assigning a score for the PLAs generated), and later answered the following questions related to the interactive model proposed and about PLAs design.
(1) Despite the subjective evaluation, is there any specific criterion that was used to evaluate the PLA?
(2) In your opinion, can the interactive model proposed bring more appropriate solutions to each individual architect?
(3) In the works related to interaction in SBSE, there is great concern regarding the problem of fatigue, caused by the high number of interactions. Based on your experience and participation in this experiment, could you tell what would be a number of PLAs to be evaluated before the problem of fatigue occurs?

6 RESULTS AND DISCUSSION

This section presents and discusses the obtained results after the execution of ExpCM[3], ExpAI and the qualitative experiment. Thus, Subsection 6.1 describes the quantitative results, and Subsection 6.2 the qualitative results. Finally Subsection 6.3 presents some threats to validity of this work.

6.1 Quantitative Results

As mentioned in Section 5.1, several data were collect from ExpCM and ExpAI in order to perform an analysis of the proposed model's effectiveness. Table 1 shows the original data of PLA AGM, such as: original fitness value and number of its component, including classes, dependencies, associations and relationships. Also, Table I presents the ideal fitness values, obtained from the solutions generated in ExpAI and ExpCM. Tables 2 and 3 show the resulting data of these experiments and also the ED values.

Analysing Tables 2 and 3, we can observe that ExpAI generated more solutions than ExpCM. Also, its possible to note that almost all solutions (for both experiments) have the fitness values for ACLASS and FM minimized while the fitness values for COE were increased. The only exception is Solution 5 of ExpAI where the fitness values of COE was minimized whereas the values of ACLASS and FM were increased.

The analysis of ED values allows to identify some differences between both experiments. In Tables 2 and 3, the best ED values for the two experiments are highlighted. It is possible to verify that, considering the three best ED value, two of them are in ExpAI set (Solutions 3 and 4). This fact can be due to the impact of architect interaction on the evolutionary process.

We also analyzed the three optimized objectives individually (Tables 2 and 3). The evaluation of the values obtained in the ACLASS objective shows that both experiments have solutions which obtained the value 12 (the lowest value found in the set of solutions), and in this case it is not possible to affirm that, in relation to this objective, there are differences between the experiments.

The analysis of cohesion values (COE) shows that in the solutions of ExpAI there is a solution with the lowest value for this objective (Solution 5 - COE = 25). Also, it can be observed that the highest COE value in ExpAI not exceeds the highest COE value in ExpCM.

The last objective analyzed was feature modularization (FM). In relation to the analysis of FM values, it is possible to observe that the solution with the lowest value of FM was obtained in ExpAI (Solution 4 - FM = 674). Besides, ExpAI achieved several optimized values for FM.

[3]The experimental package for ExpCM and ExpAI, containing the solutions generated (PLAs) and the analysis about them can be found in https://mega.nz/#!cTISxaiQ!YSrv3TuK6OKD41dVO7bSp_wx2heZOUxjBNtyHsAXlAo

Table 1: Information of the original PLA design for AGM

PLA	Original Fitness (ACLASS, FM, COE)	Ideal Fitness (ACLASS, FM, COE)	Packages	Classes	Dependencies	Associations	Relationships
AGM	30.0 , 758.0 , 26.0	12.0 , 674.0 , 25.0	9	30	14	7	47

Table 2: ExpCM - Results

Solution	Solution Fitness (ACLASS, FM, COE)	ED	Packages	Classes	Dependencies	Associations	Relationships
Solution 1	23.0 , 719.0 , 27.0	46.4	11	28	12	8	44
Solution 2	28.0 , 752.0 , 26.0	79.6	9	28	13	8	46
Solution 3	16.0 , 694.0 , 29.0	20.8	11	28	13	8	46
Solution 4	12.0 , 676.0 , 30.0	**5.4**	12	28	13	8	46

Table 3: ExpAI - Results

2*Solution	Solution Fitness (ACLASS, FM, COE)	ED	Packages	Classes	Dependencies	Associations	Relationships
Solution 1	24.0 , 718.0 , 27.0	45.7	10	28	13	8	46
Solution 2	23.0 , 712.0 , 28.0	39.7	11	28	10	7	40
Solution 3	16.0 , 692.0 , 29.0	18.9	11	28	13	8	46
Solution 4	12.0 , 674.0 , 30.0	**5.0**	12	28	13	8	46
Solution 5	34.0 , 793.0 , 25.0	121.0	8	28	12	8	44
Solution 6	28.0 , 747.0 , 26.0	74.7	9	28	12	8	44
Solution 7	18.0 , 716.0 , 28.0	42.5	11	28	13	7	45

Initially, ExpAI was configured to have five interactions with the architect. However, after three interactions and 36 PLAs evaluated (12 for each generation), it was verified that the architect already showed fatigue. Thus, the execution of the experiment continued without interactions. The architect's evaluation happened in the 6th, 7th and 8th generations of solutions.

The architect was asked to evaluate the solution with a score in the range 1 to 5. Table 4 shows the architect's evaluations of the PLAs generated in ExpAI.

Analysing the scores given by the architect to the PLA solutions allows us to infer that, for this particular architect, other aspects are more important than the fitness values, corroborating the statement of Simons, Singer and White[24]. There are solutions with the same fitness values, but with different scores. As an example, in Interaction 1, the solutions TEMP-0 and TEMP-4 have the same fitness values, but their score are different (1 and 4, respectively). Moreover, in other interactions, this same situation happens.

Also, its possible to verify that there are five solutions with score=1. Four of these solutions occur in Interaction 1, only one in Interaction 2 and none in Interaction 3. This may indicate that the architect's evaluation had influence, since the number of solutions considered very bad diminished after the beginning of the architect interaction. This improvement in the evaluations of the solutions can also be observed by the median values of the three solutions set evaluated by the architect. In Interaction 1, the median value is equal to 2, whereas in the following interactions, this value changes to 3. This change in the value of the median may indicate an improvement in the set of solutions as a whole.

Another relevant aspect is related to the solutions found by each experiment that compose the (PF_{true}). To compose the (PF_{true}) of this case study, all solutions generated from both experiments (four solutions from ExpCM and seven solutions from ExpAI) were used. From all solutions, eight non-dominated solutions compose the (PF_{true}). The names of the solutions that compose (PF_{true}) are highlighted in bold in Tables 2 and 3.

Considering the total of eight solutions that compose the (PF_{true}), seven solutions were generated from ExpAI and only one solution from ExpCM. Thus, there are evidences that the architect interaction during the optimization process allowed obtaining solutions that better explore the search space.

Finally, it was also identified differences between the sets of solutions obtained by each experiment. In ExpAI the most striking aspect is that there is a larger number of solutions generated. In addition, it can be emphasized that the solution with the best ED and the best values of each objective individually are found in the set of solutions of ExpAI. These facts point out to evidences that architect interaction during the optimization process allowed the emergence of solutions with different fitness values and, in some cases, better than those found in the current model.

6.2 Qualitative Results

In order to identify what an architect considers important when evaluating a PLA and also to verify from how many interactions the fatigue problem occurs, at the end of ExpAI execution some questions (presented in previous section) were made to the architect. Their answers helped us to identify metrics that this particular architect considers important when evaluating a PLA, such as: *number of packages and classes close to the original*; *number of features modularized in only one package*; and *Emergence or not of large classes*. Based on the answers given by the architect, it was possible carry out a deeper investigation about the model proposed.

One aspect considered by the architect is the number of packages and classes close to the original solution. The original PLA has 9 packages and 30 classes. The analysis of Tables 2 and 3 shows variations in relation to the number of packages compared to the original AGM PLA (all solutions have a number between 8 and 12 packages) and there is no variation in relation to the number of

Table 4: Scores given by the architect

PLA	Interaction 1		Interaction 2		Interaction 3	
	Score	Solution Fitness (ACLASS, FM, COE)	Score	Solution Fitness (ACLASS, FM, COE)	Score	Solution Fitness (ACLASS, FM, COE)
TEMP−0	1	28.0 , 749.0 , 26.0	3	21.0 , 743.0 , 30.0	3	21.0 , 743.0 , 30.0
TEMP−1	3	21.0 , 743.0 , 26.0	2	28.0 , 749.0 , 26.0	3	28.0 , 749.0 , 26.0
TEMP−2	1	28.0 , 749.0 , 26.0	2	28.0 , 749.0 , 26.0	3	28.0 , 749.0 , 26.0
TEMP−3	2	28.0 , 757.0 , 27.0	3	21.0 , 743.0 , 30.0	3	28.0 , 749.0 , 26.0
TEMP−4	4	28.0 , 749.0 , 26.0	3	21.0 , 743.0 , 30.0	2	34.0 , 793.0 , 25.0
TEMP−5	3	28.0 , 749.0 , 26.0	4	28.0 , 749.0 , 26.0	4	28.0 , 755.0 , 27.0
TEMP−6	2	28.0 , 760.0 , 27.0	4	28.0 , 749.0 , 26.0	3	28.0 , 749.0 , 26.0
TEMP−7	4	28.0 , 749.0 , 26.0	1	28.0 , 749.0 , 26.0	4	28.0 , 753.0 , 27.0
TEMP−8	2	28.0 , 749.0 , 26.0	3	21.0 , 743.0 , 30.0	2	28.0 , 749.0 , 26.0
TEMP−9	3	28.0 , 753.0 , 27.0	3	28.0 , 749.0 , 26.0	2	28.0 , 749.0 , 26.0
TEMP−10	1	24.0 , 719.0 , 27.0	3	28.0 , 749.0 , 26.0	2	28.0 , 749.0 , 26.0
TEMP−11	1	28.0 , 749.0 , 26.0	3	28.0 , 749.0 , 26.0	3	28.0 , 749.0 , 26.0
Median	**2**		**3**		**3**	

classes. All solutions (both experiments) have the number of classes equal to 28. In the original PLA AGM, there is a total of 30 classes, however, there are two classes ("point" and "display") in the "GameBoardCtrl" package that do not contain attributes and methods and were removed during the optimization process, because one constraint of MOA4PLA is not having empty classes. Taking this into account, it can be observed that all solutions contain the same number of classes as the original solution. When analyzing the number of packages, it can be observed that there are two solutions with the same quantity of packages of the original solution, one in ExpAI (Solution 2) and another in ExpCM (Solution 6). However, in the experiment with interaction, there are also two other solutions (Solutions 1 and 5) with a closer amount of packages in relation to the original. The second aspect mentioned by the architect is the feature modularization. Attempts to modularize features, such as: "bowling", "brickles", "pong" and "save" can be seen in both experiments (Figures 5 and 6).

In some cases, new packages were created to modularize these features. Figure 5 presents an improvement in the feature modularization of Solution 1 in ExpAI (b), considering a creation of new a package which encapsulates the kinds of games included in AGM PLA. This package is composed of classes Game, BowlingGame, BricklesGame and PongGame. Besides, Figure 6 shows FM considering the feature save in Solution 1 - ExpCM (b). In this context, a new package composed only by classes related to this feature was created . It is important to highlight that Figure 6 (a) presents an excerpt of the original package GameBoardCtrl in which this feature was originally included.

The last aspect described as relevant by the architect was the emergence of large classes. In this case, it was not possible to perceive differences between the experiments, since the largest classes found in the original PLA AGM remained unchanged after the optimization process. As an example, the largest class found in the original PLA AGM is the "GameBoard" (10 attributes and 23 methods), and in all solutions, from both experiments, the amount of attributes and methods for this class remained the same.

In relation to the modifications presented in the interactive proposed model can bring more appropriate solutions to each particular architect. However, in the experiment carried out, the configuration related to the number of individuals and generations was not enough to show an explicit improvement in the solutions. For this motive, according the architect, no solution received the score = 5.

Since this experiment was the first considering the OPLA-Tool interactive processing, the option of not adopting a cross-generational spacing is because we did not know in advance if it was necessary. From the architect's point of view, the evolutionary process should be extended over a longer period of time and with a larger number of individuals (around 100 individuals) in order to identify better optimizations. Besides, the architects suggests, in order to avoid the fatigue problem, to limit the number of generated solutions to be evaluated in each interaction.

6.3 Threats to Validity

This initial study was done using only one architect to evaluate the PLAs generated at each interaction time. Thus, it is not possible to conclude that this architect's opinions can be generalized to a larger population. Furthermore, in the third interaction, was noted that the architect presented fatigue. This problem is due to the fact that the PLA evaluation is a complex task that evolves several aspects. In this way, the fatigue problem can have impacted the evaluation of PLAs solution in the interaction process.

The fatigue problem also affected what was initially planned for the experiment. The experiment was planned to have five interactions, but, when the architect presented fatigue, the two remaining interactions were not performed.

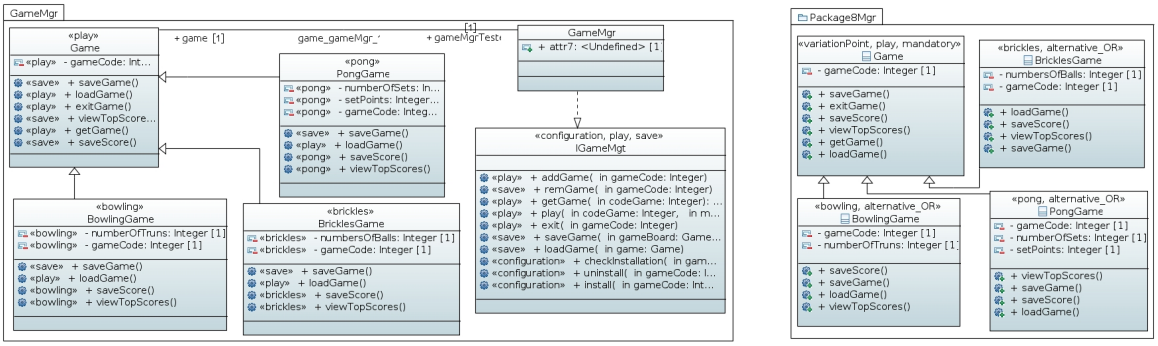
Another threat to observe is that, in the architect's opinion, the evolutionary process needs a larger number of generations to produce solutions with "excellent" scores. Thus, it was not possible to demonstrate clearly this fact, since a relative small number of interactions was experimentally realized.

7 ANSWERING THE RESEARCH QUESTIONS

The research questions posed in Section 1 are answered in this section.

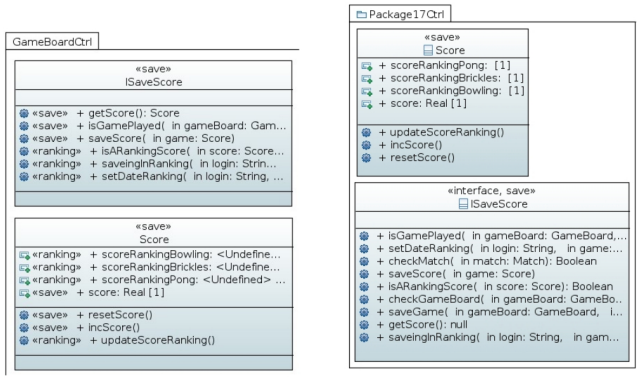
RQ1: Are the best architect-evaluated PLAs the best PLAs according to their fitness?

As mentioned in the previous section, none evaluated solution received the maximum score. Some solutions, however, received score= 4, indicating that these are good solutions for the architect,



(a) GameMgr Package from the original PLA (b) Package created in Solution 1 of ExpAI

Figure 5: Excerpt of the AGM PLA which presents the modularization of Bowling, Brickles and Pong features.



(a) GameBoardCtrl Package from the original PLA (b) Package created in Solution 1 of ExpCM

Figure 6: Excerpt of the AGM PLA which presents the Save feature modularization.

with characteristics very close to what he/she expected. These solutions continued in the optimization process, because they received score = 4, resulting in other solutions at the end of processing.

Although there was no PLA design with an excellent evaluation, it can be seen that, for this particular architect, the evaluation had no relation to fitness values, since solutions with exactly the same fitness values had different scores, with grades ranging from 1 to 4 (inclusive). In this way, it cannot be concluded that the best fitness solutions were the best evaluated. Also, such fact points out that an architect consider some characteristics of a PLA design more important than the fitness values brought by the metrics.

RQ2: Does the proposed interactive model for search-based PLA design enable the generation of PLA design alternatives that satisfy the architect's preferences?
Tables 2 and 3 allow to infer that the results of ExpAI and ExpCM have differences. Notably, the experiment done with the architect interaction resulted in more solutions at the end of the processing, which may indicate that, somehow, the architect's opinion had an influence on the optimization process. The scores assigned by the architect changed over generations (Table 4). In the Interaction 1, he assigned a score median of 2. In the Interaction 3, the score median was 3. Such a difference points out that the solutions obtained in the third interaction are more likely to the architect's preferences. Hence, in spite of the experiment was done with only one architect, there are initial evidences that the proposed interactive model enables the generation of PLAs that are in accordance with the architect's preferences.

The qualitative analysis conducted with the architect interaction allowed to bring insights to what an architect considers important in a PLA design, as well as the most appropriate moment and amount of interactions to be done to avoid the fatigue problem.

8 LESSONS LEARNED

This work presents an exploratory study about how to incorporate user preferences in PLA Design. Thus, considering the proposed model and the experiments carried out some lessons learned are presented above. These lessons represent points that can be improved in both the proposed interactive model and in the design of further experiments. The first three lesson are related to interactive processes in SBSE and are independent from our proposed model. The last two are related specifically to the experiments realized considering optimizing PLA Design. They are:

(1) Keep the number of PLA evaluations as low as possible. Architect's fatigue, mentioned in works related to interactive processes, was evident in the third interaction (after the evaluation of 24 PLA designs). Thus, it was possible to note that the number of PLAs presented to the architect should be decreased so that the evaluation is not impaired;

(2) Use machine learning algorithms to support the user preferences. Also, in relation to fatigue problem, some studies reported that the use of Machine Learning Algorithms can be effective in dealing with this problem. The use of these algorithms can be possible. Therefore, after some interactions, the system "learns" about the preferences of that particular architect and, can replace it in the interactive process;

(3) Fitness is not enough. The evaluations made by the architect allow us to realize that there are more relevant aspects in a PLA design than just the fitness values of the solutions, corroborating the Simons, Singer and White's statement [24].

(4) Adequate the configurations for new experiments. The settings used in the experiments must be modified. With a greater number of generations and individuals, the modifications made by OPLA-Tool in the PLA can become more evident and the architect evaluation more effective;

(5) Improve the interactive model with the architect's feedback. The interactive model must be modified so that it is not necessary for the architect to follow all the processing done by OPLA-Tool. As suggested by the architect, some breakpoints may

be placed in the processing and, in these moments, an alert can be sent (by e-mail, for example) to the architect so that it evaluates the solutions generated up to that moment. Furthermore, was mentioned by the architect that the solutions generated between one interaction and another had minor differences. This may be due to the fact that, from the first interaction, each subsequent interaction was made with the solutions of the next generation, that is, the solutions only evolved for one generation to be evaluated again. Thus, modifications must be made so that the interval between interactions is made after a larger number of generations.

9 CONCLUDING REMARKS

The MOA4PLA approach, implemented by OPLA-Tool, enables SPL architects to generate different optimized PLA designs, thus they can choose one that is closest to their preferences. However, after the optimization process, may be achieved solutions that are not matching with the architect's preferences. This fact is also pointed out in other problems treated in SBSE. In this context, there is a growing trend of researches that seek to include DM preferences in the optimization process.

To do so, the objective of this work was to propose a model to include the architect interaction during the optimization process performed by MOA4PLA via OPLA-Tool, in order to allow the evaluation of the generated solutions during the evolutionary process. Then, the solutions (PLA design alternatives) obtained as output tend to be more suitable to the architect's preferences.

The experiments carried out using the current model and the proposed one showed significant differences between them. The interactive model brought more solutions in the end of the process. Furthermore, the analysis of the Pareto front showed that, almost all solutions generated by the current model were dominated by those generated by the proposed interactive model. Another significant result from the proposed model is that, to evaluate PLA solutions, metrics are not enough. Some solutions with the same values for the metrics were evaluated differently by the architect. In this way, other aspects, such as the architect's preferences, should be considered.

The experiments also made it possible to point out some aspects that need to be improved in the interactive model. The problem of fatigue, reported in other works that uses interactive models, was evident in the experiment carried out with the proposed model. In this way, for future works, we will look for: ways to decrease the amount of solutions to be evaluated by the architect; performing other experiments to determine the best moment for the architect interaction, as well as the amount of interactions to be done before the fatigue problem arises; performing an in-depth analysis (with greater number of architects) about how to represent the architect's preferences in the search-based PLA design; using machine learning algorithms to reduce the amount of interactions that must be made by an architect.

REFERENCES

[1] 2015. Guidelines for conducting systematic mapping studies in software engineering: An update. *Information and Software Technology* 64 (2015), 1–18.

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

[3] Wesley K. G. Assunção, Silvia R. Vergilio, and Roberto E. Lopez-Herrejon. 2017. 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 2017, Salvador, Brazil, May 29-31, 2017, Proceedings*. 95–111.

[4] Slim Bechikh, Marouane Kessentini, Lamjed Ben Said, and Khaled Ghédira. 2015. Preference incorporation in evolutionary multiobjective optimization: a survey of the state-of-the-art. In *Advances in Computers*. Vol. 98. Elsevier, 141–207.

[5] James L Cochrane and Milan Zeleny. 1973. Multiple Criteria Decision Making.

[6] Carlos A Coello Coello, Gary B Lamont, David A Van Veldhuizen, et al. 2007. *Evolutionary algorithms for solving multi-objective problems*. Vol. 5. Springer.

[7] Thelma Elita Colanzi and Silvia Regina Vergilio. 2016. A feature-driven crossover operator for multi-objective and evolutionary optimization of product line architectures. *Journal of Systems and Software* 121 (2016), 126–143.

[8] Thelma Elita Colanzi, Silvia Regina Vergilio, Itana Gimenes, and Willian Nalepa Oizumi. 2014. A search-based approach for software product line design. In *Proceedings of the 18th International SPL Conference-Volume 1*. ACM, 237–241.

[9] Kalyanmoy Deb, Samir Agrawal, Amrit Pratap, and Tanaka Meyarivan. 2000. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In *International conference on parallel problem solving from nature*. Springer, 849–858.

[10] Édipo Luis Féderle, Thiago do Nascimento Ferreira, Thelma Elita Colanzi, and Silvia Regina Vergilio. 2015. OPLA-Tool: a support tool for search-based product line architecture design. In *Proceedings of the 19th International Conference on Software Product Line*. ACM, 370–373.

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

[12] Mark Harman, Edmund Burke, John Clark, and Xin Yao. 2012. Dynamic adaptive search based software engineering. In *Proceedings of the ACM-IEEE international symposium on Empirical software engineering and measurement*. ACM, 1–8.

[13] M. Harman, Y. Jia, J. Krinke, W. B. Langdon, J. Petke, and Y. Zhang. 2014. Search Based Software Engineering for Software Product Line Engineering: A Survey and Directions for Future Work. In *Proceedings of the 18th International Software Product Line Conference - Volume 1 (SPLC '14)*. 5–18.

[14] Mark Harman and Bryan F Jones. 2001. Search-based software engineering. *Information and software Technology* 43, 14 (2001), 833–839.

[15] Mark Harman, Phil McMinn, Jerffeson Teixeira De Souza, and Shin Yoo. 2012. Search based software engineering: Techniques, taxonomy, tutorial. In *Empirical software engineering and verification*. Springer, 1–59.

[16] Joshua Knowles and David Corne. 1999. The pareto archived evolution strategy: A new baseline algorithm for pareto multiobjective optimisation. In *Congress on Evolutionary Computation (CEC99)*, Vol. 1. 98–105.

[17] Roberto E. Lopez-Herrejon, Lukas Linsbauer, and Alexander Egyed. 2015. A systematic mapping study of search-based software engineering for software product lines. *Information & Software Technology* 61 (2015), 33–51.

[18] João Choma Neto, Tatiane Gaieski, Aline M. M. Miotto Amaral, and Thelma Elita Colanzi. 2018. Quanti-Qualitative Analysis of a Memetic Algorithm to Optimize Product Line Architecture Design. In *IEEE 30th International Conference on Tools with Artificial Intelligence, ICTAI 2018, 5-7 November 2018, Volos, Greece*. 498–505.

[19] Vilfredo Pareto. 1927. *Manuel d'économie politique*. Paris, M. Giard.

[20] Aurora Ramirez, Jose Raul Romero, and Christopher Simons. 2018. A systematic review of interaction in search-based software engineering. *IEEE Transactions on Software Engineering* (2018).

[21] Software Engineering Institute (SEI). 2009. The Arcade Game Maker Pedagogical Product Line. https://resources.sei.cmu.edu/library/asset-view.cfm?assetID=485941. Accessed in 2019, March.

[22] Mark Shackelford. 2007. Implementation issues for an interactive evolutionary computation system. In *Proceedings of the 9th annual conference companion on Genetic and evolutionary computation*. ACM, 2933–2936.

[23] Mark RN Shackelford and Christopher L Simons. [n. d.]. Metaheuristic design pattern: interactive solution presentation.. In *GECCO (Companion)*.

[24] Chris Simons, Jeremy Singer, and David R. White. 2015. Search-Based Refactoring: Metrics Are Not Enough. In *Search-Based Software Engineering*, Márcio Barros and Yvan Labiche (Eds.). 47–61.

[25] Jianning Su and Shutao Zhang. 2010. Research on product shape innovation design method with human-computer interaction through genetic algorithm. In *2010 IEEE 11th International Conference on Computer-Aided Industrial Design & Conceptual Design 1*, Vol. 1. IEEE, 301–305.

[26] Frank J Van der Linden, Klaus Schmid, and Eelco Rommes. 2007. *Software product lines in action: the best industrial practice in product line engineering*. Springer Science & Business Media.

[27] Yenisei Delgado Verdecia, Thelma E. Colanzi, Silvia R. Vergilio, and Marcelo C. Benitez Santos. 2017. An Enhanced Evaluation Model for Search-Based Product Line Architecture Design. In *XX Ibero-American Conference on Software Engineering (CIbSE - ICSE 2017)*.

[28] Eckart Zitzler, Lothar Thiele, Marco Laumanns, Carlos M Fonseca, and Viviane Da Fonseca Grunert. 2002. Performance assessment of multiobjective optimizers: An analysis and review. *TIK-Report* 139 (2002).

C

Artigo publicado no COMPSAC 2019

Supporting decision makers in search-based product line architecture design using clustering

1st Willian Marques Freire
Informatics Department, State University of Maringa (UEM)
State University of Maringá
Maringa, Brazil
willianmarquesfreire@gmail.com

2nd Carlos Vinícius Bindewald
Informatics Department, State University of Maringa (UEM)
State University of Maringá
Maringa, Brazil
viniciusbindewald@hotmail.com

3rd Aline M. M. Miotto Amaral
Informatics Department, State University of Maringa (UEM)
State University of Maringá
Maringa, Brazil
ammmamaral@uem.br

4th Thelma Elita Colanzi
Informatics Department, State University of Maringa (UEM)
State University of Maringá
Maringa, Brazil
thelma@din.uem.br

***Abstract*—The Product Line Architecture (PLA) is one of the most important artifacts of a Software Product Line (SPL). PLA design can be formulated as an optimization problem with many factors. In this context, the MOA4PLA approach was proposed to optimize PLA design using search algorithms and metrics specific to the context. MOA4PLA treats the PLA design as a multi-objective optimization problem. At the end of the search process several PLA design alternatives are presented to the decision maker, difficulting the decision about which PLA alternative best fits with his/her needs. In this sense, this work proposes the usage of clustering algorithms to group PLAs design alternatives, according to their characteristics and assist the decision maker in the choice of one PLA design for the SPL. For this purpose, an empirical study was carried out, involving quantitative and qualitative experiments. Such an study integrated the K-Means++ and DBSCAN clustering algorithms in the MOA4PLA approach. The results of the experiments were promising, since an appropriate grouping of the solutions can be quantitatively observed, and also, qualitatively, the suitability of the solutions to the decision makers needs was verified.**

***Index Terms*—product line architecture, multi-objective optimization, clustering algorithm, K-Means++, DBSCAN**

I. Introduction

Software Product Line (SPL) is an approach that provides artifact reuse based on a common infrastructure composed by core assets, which contains artifacts with commonalities and variabilities [37]. Several products can be derived from a SPL infrastructure. So, there exists several different architectures, one for each specific product. Every product architecture is instantiated from the Product Line Architecture (PLA) [37]. A PLA contains elements such as variation points, variabilities and variants [33], making possible the anticipation of design decisions regarding customization, extension and adaptation of software systems for different contexts [27]. Hence, a PLA needs to be generic and flexible to allow the configuration of the various products that derive from a SPL. To comply those needs and evaluate design alternatives for a PLA, some works often taken into account a set of quality attributes metrics, such as design stability and feature modularization [29].

Focusing on architectural properties, architects need to develop PLA design considering different and conflicting architectural measures, for instance with high feature modularization and low feature scattering. This makes PLA design a people-intensive task for which there is not a single solution due to the number of factors that influence the task. Techniques of Search-Based Software Engineering (SBSE) [15] can be used to optimize PLA design to automatically obtain several near-optimal design alternatives according to the selected objectives. Metrics to evaluate those factors are used as objectives. The set of obtained solutions is presented to the architect, who selects the desired solution. In the PLA design context, the architect is the Decision Maker (DM).

SBSE includes approaches for optimization using metaheuristics, and has aroused interest on the part of the researches, because it has supported Software Engineering activities that demand great cognitive effort [16]. There are three moments when users can interact with SBSE techniques [28]: (i) a priori: before the execution of the search algorithm, (ii) during the execution of the search algorithm and (iii) a posteriori: after the search algorithm achieved the solutions.

In this context, the approach named Multi-objective Optimization Approach for PLA design (MOA4PLA) [8] was proposed to optimize PLA design using search algorithms and architectural metrics specific to the context. In addition to supporting the qualitative analysis of PLAs, these architectural metrics are used in software design to obtain indicators which aid the measurement of quality attributes and evaluation of the PLAs logical view [24]. It is important to deatch, that there are also other views of software architecture, such as implementation, process and deployment which adress other architectural properties, such as reliability, confiability and cost.

MOA4PLA treats the PLA design as a multi-objective optimization problem, that allows the simultaneous optimization of several architectural properties (objectives). From an original

PLA design, multi-objective evolutionary algorithms employ search operators specific for PLA design in order to obtain descendant solutions. Each solution obtained in the search is evaluated and ranked by the objective functions. At the end of the search, several design alternatives may be automatically achieved according to the selected objectives.

In MOA4PLA, the DM interacts in two moments: a priori and a posteriori. In the a priori interaction, DM must to select the objectives to be optimized, to set the algorithms parameters and to provide the original PLA design to be optimized. After the search process (a posteriori interaction), several PLA design alternatives (solutions) are presented to DM, who needs to select one of them to be adopted as the PLA to be used in the SPL under development. The number of PLA design alternatives is variable and dependent on the original PLA provided as input. A high number of solutions can cause human fatigue during that interaction.

In the literature there are works that deal with the problem related to many solutions obtained through the optimization algorithms [3] [17] [32]. In those cases, the solutions generated do not require complex analysis from DM; hence, ways to facilitate this analysis are not explored yet. On the other hand, in the context of PLA design, the analysis of each PLA design alternative before choosing one specific alternative is more complex because a lot of architectural elements and their respective relationships should be analyzed. Examples of architectural elements are packages, classes, interfaces, variabilities, variants and so on. In this work, we are interested in investigating the usage of clustering algorithms as a way to avoid human fatigue because a clustering algorithm is able to group similar solutions reducing the number of PLA design alternatives to be analyzed.

In short, clustering algorithms categorize solutions taking into account attributes pertaining to them [2] [12]. In the search algorithm, objective function values are used to indicate the solution optimization level for each objective [9]. Therefore, those values can be used as attributes to cluster the solutions. After the solutions clustering, it is possible to verify the distribution of solutions by cluster. There may be clusters grouping the most cost-effective solutions (considering the objectives) and even clusters grouping solutions with the best values for only one of the optimized objectives. This categorization may help the DM in the decision making of what solution to adopt, because depending on the context, several PLA alternatives can be generated, and the analysis of solution by solution to find the best alternative can be laborious. Thus, if the solutions are pre-organized in such a way that important aspects for the DM are considered in the clustering, the DM can analyze only the cluster that contains such aspect, reducing the solutions number to analyze.

In this sense, the main objective of this work is to propose the usage of clustering algorithms to reduce the amount of solutions to be presented to DM and to facilitate the analysis and the choice of an PLA alternative after the search algorithm execution. In order to test the applicability of the proposal, we carried out a quantitative experiment where two PLA designs were optimized and the obtained solutions (PLA design alternatives) were grouped through two clustering algorithms. In addition to the quantitative analysis, a qualitative study involving four decision makers was performed to verify the quality of the clusters obtained by the clustering algorithms. From the experiments results, the present work intends to answer the following questions:

- RQ1: ***Which clustering algorithm best fits to support search-based PLA design?*** We consider important to experiment different clustering algorithms and obtain preliminary evidence on which algorithm fits better for the PLA design context because, from the best of our knowledge, this is the first study where clustering algorithms are used to this goal. In our study, the clustering algorithms used are K-Means++ [2] and DBSCAN [12]. The first one aims at minimizing the golden ratio between the solutions forming circular centroids, whereas the second one calculates the density of each one, being able to form clusters of any geometric form.

- RQ2: ***Are the solutions best evaluated by the DM grouped into the best performing cluster?*** The answer for **RQ2** can point the effectiveness of the clustering algorithms to reduce the number of solutions to be analyzed by the DM during a posteriori interaction.

The main contributions of this work are: (i) to assist DM in choosing the best PLA by grouping PLA design alternatives using clustering algorithms, (ii) to compare the clustering algorithms used in this work for the PLA design context, and (iii) to present a way to integrate clustering algorithms with search algorithms in the SBSE field.

This paper is organized as follows. Section II presents background and Section III describes the method employed to perform the empirical study, which involves a quantitative and a qualitative experiments. Results and discussion are presented in Section IV. The research questions are answered in Section V. Section VI concludes the paper.

II. Background

This section addresses the main concepts involved in the PLA design optimization, including software product line, genetic algorithms, multi-objective optimization, search-based PLA design and clustering algorithms.

Software Product Line (SPL) is an approach derived from other engineering areas that employs reuse techniques for a product family in a given domain. Aiming at costs reducing and productivity increasing, this approach allows to create and delivery products that are variants of an initial product, inheriting common features and receiving new properties [33]. To this end, the Product Line Architecture (PLA) defines a common design for all products derived from the SPL, including mandatory and variable features. Features are user-visible aspects or characteristics of the domain [20].

PLA design is influenced by several factors, such as, feature modularization, PLA extensibility, reusability level, coupling and cohesion. Some of them can be conflicting factors leading to several possible designs for a specific SPL. Optimizing all those factors is a non-trivial task and a high-demand activity

for a software engineer [8]. In this context, PLA design can be treated as a multi-objective problem. Such kind of problems depends on diverse factors (objectives) that are in conflict and, in general, does not have a single solution [6]. Hence, diverse good solutions exist and these solutions are named non-dominated and form the Pareto front [31]. In most applications, the search for the Pareto optimal is NP-hard, then the optimization problem focuses on finding an approximation set, as close as possible to the Pareto front.

Software Engineering hard problems, such as PLA design, are explored in the SBSE (Search-Based Software Engineering) field [15], in which problems are solved using metaheuristics as Genetic Algorithms (GAs). A GA is a heuristic inspired by the theory of natural selection and genetic evolution [6]. From an initial population (candidate solutions), basic operators are applied to evolve the population, generation by generation. Through the selection operator more copies of those individuals with the best objective function values are selected to be parent. So the best individuals will survive in the next population. The crossover operator combines parts of two parent solutions to create a new one. The mutation operator randomly modifies a solution. The offspring population created from the selection, crossover and mutation replaces the parent population. Variants of GAs adapted to multi-objective problems are called Multi-Objective Evolutionary Algorithms (MOEAs). The most used MOEA is NSGA-II (Non-dominated Sorting Genetic Algorithm II) [9].

In the SBSE field, three moments of interaction with DM are found in different contexts [28]. The first interaction occurs prior to the execution of the optimization process (a priori) to configure parameters such as input, objective functions and other algorithm parameters. The second moment consists on the inclusion of the DM in the process, in order to incorporate their preferences during the solutions optimization in an interactive way. Finally, the last one is the interaction with DM at the end of the process (a posteriori), to help in the evaluation and choice of one solution.

Given the basic concepts involved in search-based PLA design, next subsection addresses the optimization approach used in this work.

A. Search-Based PLA Design

Colanzi *et al.* [8] proposed the MOA4PLA approach to automate the optimization of PLA designs using MOEAs. After the search process, it returns a set of solutions that have the best trade-off among various optimized objectives, such as feature modularization, SPL extensibility and coupling.

The first activity of MOA4PLA is to instantiate an initial solution, from the original PLA design, to be used by the algorithm as the starting point of the optimization process. The input for this activity is the PLA design modeled in an UML stereotype class diagram containing the SPL variabilities. The output of this activity is the PLA representation that contains architectural elements such as components, interfaces, operations and their inter-relationships. Each element is associated with feature(s) by using UML stereotypes and can be either common to all SPL products or variable being present only in some product(s). Variable elements are associated with variabilities that have variation points and respective variants.

The PLA for the SPL Arcade Game Maker (AGM) [36], presented in Figure 1, was used in the present work. In this PLA design there are five variabilities detailed in UML notes attached to the classes that realize the variabilities. Variation points, alternative variants, optional and mandatory features are shown by the stereotypes $variationPoint, alternative\_OR, optional$ and $mandatory$, respectively, according to the SMarty approach [30]. Features, such as ranking, movement, play, and save are also tagged with stereotypes. Each architectural component was represented by a package. Attributes, operations, GUI components and some details of variabilities were omitted in order to improve readability.

To start the optimization process, the architect must choose which MOEA will be used along with the execution parameters. After the initial configuration, the PLA received as input is optimized in the search process according to the search operators defined by MOA4PLA. These operators perform GAs basic movements: operator of individuals crossover [7] and operators to mutate [8] the new generated individuals. Each solution obtained during the optimization process is evaluated according to the evaluation model. MOA4PLA contains an evaluation model specific for PLA design [38]. Such model has objective functions based on software metrics, which provide indicators on architectural properties. Some of them are feature modularization, SPL extensibility, variability, cohesion and coupling. The DM must choose a subset of objective functions related to the properties that he/she wants to optimize during the search. In the present work, three objective functions are used: Feature Modularization (FM) measures the degree of feature modularization in terms of feature diffusion, feature-based cohesion and feature interaction over architectural elements; Cohesion (COE) indicates how much the class responsibilities are satisfactorily separated; and Class Coupling (ACLASS) measures the classes dependency. Therefore, the selection of objective functions to be optimized defines how the fitness of each solution is calculated during the search. In other words, the fitness defines the quality of each PLA design alternative, which relates directly to the objectives selected by the DM from the evaluation model.

After search algorithm execution, a set of PLA design alternatives is generated as output. Such alternatives have the best trade-off among the optimized objectives. The DM needs to select one of the obtained solutions to adopt as the PLA for a given SPL. Such a selection depends on the DM preferences or priorities.

In order to automate the application of MOA4PLA the tool named OPLA-Tool [13] was developed. It provides implementation of: (i) two MOEAs (NSGA-II [9] and PAES (Pareto Archived Evolution Strategy) [21]), (ii) specific search operators proposed to PLA design and (iii) the whole objective functions encompassed in the MOA4PLA evaluation model. OPLA-Tool allows the selection of objective functions, MOEA

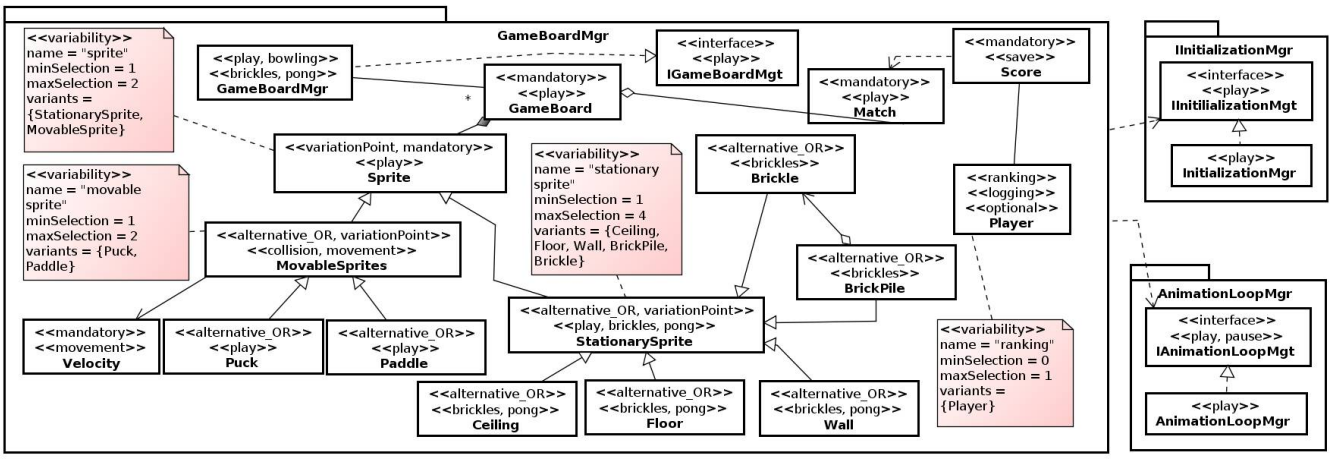


Fig. 1. Excerpt of the original PLA for AGM

and search operators to be used in the optimization process, as well as the visualization of the solutions (PLA design alternatives). However, the greater the number of architectural elements and variabilities, the greater the number of PLA design alternatives obtained during the optimization process. Thus, in spite of having support of OPLA-Tool to visualize the solutions and their respective fitness values, when there are many alternative solutions, the choice of one alternative demands a great effort of the DM to analyze each design alternative what can cause human fatigue. In those cases, clustering algorithms could be used to assist the decision making by grouping similar solutions and reducing the number of alternatives to be analyzed. The next subsection presents the clustering algorithms used in this work with that goal.

B. Clustering Algorithms

The goal of clustering algorithms is to detect and group data with similar characteristics. The result of clustering algorithms are sets of grouped data, named clusters [18]. Clustering algorithms were used in the present work to support the choice of a PLA design at the end of the MOA4PLA optimization process (a posteriori interaction). To do so, two clustering algorithms were applied in order to compare the results: K-Means++ [2] and DBSCAN [12]. K-Means++ is a partitional clustering algorithm that was used in the SBSE context [3]. Due to the fact that it was necessary to use one more clustering algorithm to compare results, we also used the density-based algorithm named DBSCAN. DBSCAN usually achieves good results for clustering sparse data [19], what usually happens in the results obtained for search-based PLA design.

The original K-Means, aims to minimize the average square distance between points in the same cluster. Although it does not have great accuracy, it is considered simple, with O(log k) complexity. The classical NP-hard clustering problem comes from computational geometry, that in a given data set, the objective is to choose k centers in order to minimize the golden ratio, that is, the total square of the distance between each point and its respective centroid [25]. However, the algorithm is started with k arbitrary centers. Calculations are made in the solutions with respect to the centroids, modifying them until the solutions do not change anymore. K-Means++ [2], is an improved variant of K-means, proposed to randomly select the data points centers, assigning weight to the points according to their square distance to the nearest center already chosen.

DBSCAN [12] takes into account the clusters density, considering that requirements such as domain knowledge arise to determine the clustering centroids number for large databases. This algorithm requires some input parameters and assists DM in their proper choice. There are the Epsilon and the number of points required to form a dense region. The average execution complexity of the algorithm is O(n log n). In the literature it is possible to find advantages of DBSCAN in relation to k-means [23], such as:

- It does not require a specific number of clusters a priori;
- It can find clusters forms besides circular ones, since in the K-Means circular clusters are formed;
- Identifies noises and outliers.

However, the following disadvantages can be found:

- It is not entirely deterministic [4];
- The quality of the clusters depends on the distance measure used in the region query function. The most commonly used is the Euclidean distance;
- It cannot group data sets with large densities differences very well, since its input parameters cannot be appropriately chosen for all clusters.

In this section we have presented aspects that differentiate the two clustering algorithms used in this work. Finally, it is necessary to configure existing parameters for both algorithms, and such configuration is detailed in Section III.

III. Empirical Study Method

This section addresses the definition of the empirical study carried out in the present work. Figure 2 shows the method used to perform the empirical study, which is composed by a quantitative experiment followed by a qualitative experiment. The participants of the qualitative experiment play the role of "Decision Makers" and hereafter they are called "users".

The goal of the quantitative experiment is to analyze the clusters formed by the clustering algorithms. In addition, as

the objective of this work is to assist the user to choose one of the PLA design alternatives, a qualitative experiment was conducted in order to verify the quality of the clusters from the user´s point of view. For the accomplishment of the experiments, several steps were performed. In this sense, each step is described in the next subsections according to the experiment type (quantitative or qualitative).

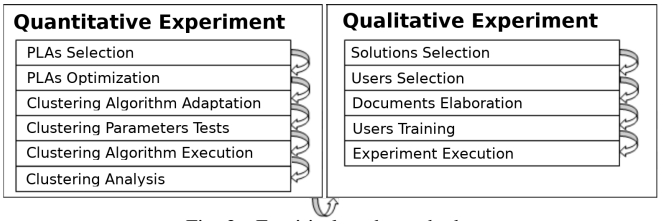


Fig. 2. Empirical study method

A. Quantitative Experiment

The quantitative experiment is composed of the following steps:

PLAs Selection: Two PLA designs were optimized in the quantitative experiment. The first one is for the SPL Arcade Game Maker (AGM), which includes three arcade games [36]. The second one is Electronic Tickets in Urban Transportation (BET), a real SPL developed to manage urban transport [10]. Table I shows information about both PLAs, including architectural elements numbers, such as: packages, variabilities, interfaces, classes and features, and their original and ideal fitness. The ideal fitness consists on the best values reached for each optimized objective, which were obtained in the optimization process presented in Section IV-A.

PLAs Optimization: The original PLA designs for AGM and BET were optimized using the algorithm NSGA-II implemented in OPLA-Tool [13]. The objective functions selected for optimization were: Feature Modularization (FM), Cohesion (COE) and Class Coupling (ACLASS) (Section II-A). The NSGA-II parameters were configured using the same settings adopted in [5]. All MOA4PLA mutation operators were used, under a mutation probability equals to 0.9. The population size was 200 individuals and the number of fitness evaluations was 3,000, what was used as the algorithm stop criterion.

Clustering Algorithm Adaptation: As explained in the Section II-B, clustering algorithms were used to aid the choice of PLA solutions presented at the end of the optimization process (a posteriori interaction). After the PLAs optimization, the selected clustering algorithm will group the solutions. To do so, Algorithm 1 presents the pseudocode for the usage of clustering algorithms in the experiment. The input $solutions$ is the set of optimized solutions returned from the step PLAs Optimization. The variable $numClusters$ is the input to K-Means++ and it is obtained from Equation 1. Variables $epsilon$ and $minPoint$ are input for DBSCAN. Their values were empirically adjusted by the procedures explained in the next step. The instruction *if-then-else* contains the calls for the clustering algorithms according to the user´s selection. In the last line the clusters returned by the clustering algorithm are assigned to one of the optimized objectives in the PLAs Optimization step. Such assignment is done in order to categorize the clusters what enables reducing the number of solutions to be analyzed by the user.

Algorithm 1 Clustering Algorithm Adaptation

```
Require: solutions
  numClusters ← Equation 1
  epsilon ← x
  minPoints ← y
  if selectedClusterAlgorithm = K-Means++ then
    clusters ← KMeans++(solutions, numClusters)
  else
    clusters ← DBSCAN(solutions, epsilon, minPoints)
  end if
  categorizedSolutions ← clustersByObjective(clusters)
```

Clustering Parameters Tests: As shown in Algorithm 1, it was necessary to configure parameters for both clustering algorithms. Therefore, the clustering algorithms were executed several times to define the parameters values effectively. As discussed in Section II-B, it is necessary to define the number of centroids that K-Means++ will form. In this work, it was empirically defined the Equation (1) for calculating it, based on equations used in existing works [14] [22] [26] , where k indicates the centroids number and n indicates the amount of data, that for the PLA design context consists on the number of PLA solutions. During the calibration of this parameter, we have realized that the standard power value of 0.5 generated a lower clusters number leading to bad-formed clusters. The clusters were well-formed when the clusters number approached the objectives number for both PLA designs.

$$k = (n/2)^{0.6} \quad (1)$$

DBSCAN does not need to configure a clusters number, but there are two other parameters to configure. The *epsilon* (ε) and the minimum number of points to form a dense region (*minPts*). When the algorithm starts the clustering process, an arbitrary starting point is selected, and epsilon is used in neighbors computation, so if many points are obtained, a cluster is started. Otherwise, the point is labeled as noise.

As a general rule, the minPts parameter can be defined as the number of dataset dimensions. In the experiments, three objectives were optimized, then the `minPts=3`. The epsilon value can be defined using a k-distance graph [11], searching for the distance of neighbors (`minPts-1`). However, in present work it was defined empirically. Several clustering executions were carried out, aiming to find a better clustering algorithms parameters configuration.

In order to find the best DBSCAN parameter setting, it were tested for *epsilon* parameter all values possibilities, ranging from 0.1 to 1, and for *minPoints* parameter all values ranging from 1 to 10. After the parameter setting tests, the best values were set as *epsilon*=0.3 and *minPoints*=3, that are input to the variables x and y in Algorithm 1.

Clustering Algorithms Execution: Due to the fact that three objective functions were used (ACLASS, COE and FM), it was possible to create three-dimensional graphs to analyze the solutions layout in the search space. The obtained clusters were verified and it is possible to identify that three of them

TABLE I
PLAs INFORMATION

PLA	Original Fitness (FM, ACLASS, COE)	Ideal Fitness (FM, ACLASS, COE)	Packages	Variabilities	Interfaces	Classes	Features
AGM	(758.0 , 26.0 , 30.0)	(667.0 , 12.0 , 25.0)	9	5	14	30	11
BET	(1486.0 , 100.0 , 122.0)	(1423.0 , 115.0 , 79.0)	56	8	30	115	18

present the best solutions for each optimized objective, and the first one also presents the solutions with the best trade-off, i.e, with the best solutions.

K-Means++ has centroids, which enables them to calculate their distance from the ideal solution, this is possible because each centroid has its own value. However, in DBSCAN the solutions are grouped by density, not having centroids, which makes it necessary to verify which best optimized objective is present in each cluster.

Clustering Analysis: After obtaining the clusters of solutions it was verified the PLA solutions grouped in each cluster, and if such solutions were correctly grouped. To validate the clusters formed by clustering algorithms we could use measures such as the Rand index [34] or the Silhouette metric [35]. The validation performed by such measures indicate how is the similarity between the clusters formed by the algorithms and the clusters that should be detected, usually defined by an oracle. However, as the goal of the present work is to assist the DM in choosing PLAs solutions, we performed a qualitative analysis to verify if the PLAs solutions were correctly grouped from the user's point of view. It is important to emphasize that this paper presents preliminary results about the usage of clustering algorithms in PLA design, thus the validation of clusters using quantitative measures, such as those presented in [35] and [34], will be performed in future works.

In this sense, in this step it was analyzed if the solutions grouped in a given cluster are close to the solution with the best value for a given objective, making it possible to categorize the clusters by objective, such as: cluster with the best FM, cluster with best ACLASS and cluster with the best COE. It was also identified a cluster which present the solutions with the best trade-off, considering it as the cluster of the best performance (best cluster). In DBSCAN it is possible to have solutions marked as noise, so such solutions were added in the clusters closest to the objective function values.

The distance between solutions was measured using the Euclidean distance to the ideal solution (ED). Considering a minimization problem [39], the ideal solution has the minimum value of each objective and it is calculated according to Equation (2). It is represented by min[i], where min is the minimum for each objective i. These minimum values are obtained from the Pareto front approximation (Section II). The ED has the goal of finding the closest solutions to the best objectives, i.e., the ideal solution. It is useful because decision makers usually prefer the solution with the best trade-off among the optimized objectives [39].

$$\sum_{k=1}^{N}(objective[i] - min[i])^2 \quad (2)$$

In addition, it is important to highlight that DBSCAN maintains clusters ordered by density, whereas K-Means++ does not maintain any cluster order. Thus, the centroids of K-Means++ were ordered using the Euclidean distance (Equation (2)) to calculate the distance between the solution fitness value and the fitness of the ideal solution. ED was also used to find the best cluster to group solutions marked as noise.

B. Qualitative Experiment

The quantitative experiment involves the following steps:

Solutions Selection: In this step the solutions obtained in the PLA Optimization step were selected to the qualitative experiment execution. We have decided to use only the AGM solutions because a higher number of solutions was generated for BET which demands great human effort to analyze them. Also, BET has more architectural elements than AGM (Table I), which also could cause a user cognitive stress during the solutions evaluation. Considering that the objective functions optimized are related to feature modularization (FM), class coupling (ACLASS) and cohesion (COE), the AGM solutions were separated in groups, according to the clusters categorization performed by the last line of Algorithm 1. Also, solutions with the best trade-off among the objectives were grouped, they are in the best cluster.

Users Selection: Four users were selected to participate in the qualitative experiment. The users were randomly assigned to one architectural property (feature modularization, class coupling, cohesion or best trade-off among the objectives). Each user evaluated only the solutions of one group. They received only information about the architectural property that they would have to evaluate, without knowing that such solutions were clustered.

Document Elaboration: All users received two supporting documents: a document containing the user profile questionnaire; and a document containing the evaluation form. All documents are available in the experimental package [1].

The questionnaire for obtaining the user profile contained questions to characterize: (i) educational level, (ii) the software development acting sector (academic or industrial), (iii) UML knowledge level, and (iv) PLA design knowledge level.

In addition to the questionnaire, the second document was a form to obtain the user evaluations for the PLAs solutions, containing: (i) Solution ID, (ii) Score, and (iii) Justification. The score should be assigned into the range [1-5].

Users Training: Each user received a training regarding the architectural property that he/she should evaluate. Users also received a complementary document, which is included in the experimental package [1].

Experiment Execution: In this step, each user received the PLA solutions and the documents mentioned in the Document Elaboration step. As mentioned above, this experiment was performed using only the AGM PLA, because it has fewer components, classes and interfaces than BET.

TABLE II
USERS CHARACTERIZATION DATA

ID	Academic Education	Acting Sector	Experience with UML	Experience with PLA design
User 1	Master's Degree student	Academic (student)	moderate	moderate
User 2	Master's Degree student	Academic (student)	moderate	moderate
User 3	Master's Degree student	Academic (student)	moderate	moderate
User 4	Graduate	Industrial (business)	moderate	moderate

Table II presents information about the experiment participants (users). All of them had intermediate knowledge in UML and PLA design. Three users are master´s degree students and only user works in a software development company. The qualitative experiment participants attributed score from 1 to 5 and the score justification for every PLA solution of the group related to the architectural property that he/she was assigned.

IV. RESULTS AND DISCUSSION

In this section we present and discuss the obtained results. The experimental package containing raw data and the PLA solutions is available in [1].

A. Quantitative Results

Figures 3 and 4 show the obtained solutions in the search space for AGM and BET, respectively. In both pictures is possible to see that solutions are sparse. As expected, some solutions has one objective better optimized than the others.

Tables III and IV present detailed information about the obtained solutions for AGM and BET, respectively. The second column present the solution fitness according to the objective functions in the format (FM, ACLASS, COE). The ED (Euclidean Distance to the ideal solution) is presented in the third column. The fitness of the ideal solution is presented in Table I. The last columns present the cluster number where each solution was grouped by the clustering algorithms K-Means++ and DBSCAN, respectively.

As shown in Table III, when grouping the AGM solutions using the K-Means++ algorithm, three clusters were created. The cluster number 0 contains 4 solutions, they have the best trade-off (ED values) and the best values for the objective functions FM and ACLASS. In this cluster, FM and ACLASS were optimized compromising the COE values. The cluster number 2 contains the solution with the best value for the function COE (Solution 7), such a solution compromised the values of FM and ACLASS when consider the original fitness

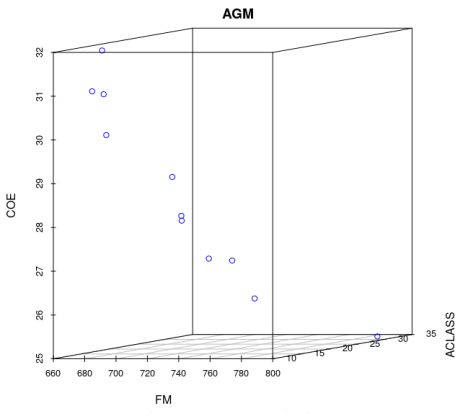


Fig. 3. AGM Solutions

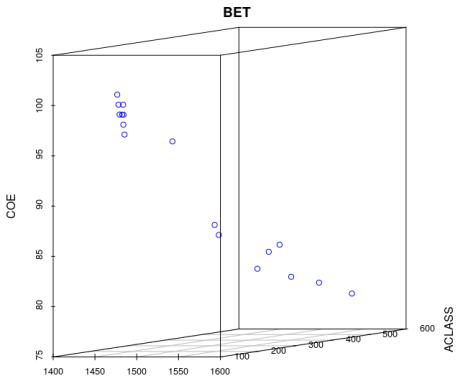


Fig. 4. BET Solutions

(Table I. The cluster number 1 contains the remaining solutions (6 solutions), which have fitness with lower values than the original one for all objective functions.

On the other hand, the DBSCAN algorithm creates only two clusters to group AGM solutions. Clusters 0 and 1 are formed by the same solutions encompassed by the Clusters 0 and 1 created by K-Means++. However, Solution 7, which has the best cohesion, was marked as noise by DBSCAN. Hoever, during the Clustering Analysis step, such a solution was grouped in Cluster 1 because indeed it is not a noise. Thus, it is possible to verify that the clustering algorithms have similar behaviour. The single difference is that K-Means++ separated the solution with the best cohesion in a cluster whereas the DBSCAN identified it as noise.

Regarding the solutions obtained for BET, the clusters for both clustering algorithms can be observed in Table IV. K-Means++ grouped the solutions with the best values for COE (5 solutions) and FM (3 solutions) in the Clusters 1 and 2, respectively. Cluster 0 has 9 solutions, which have the lowest values for ACLASS and the lowest values of ED. DBSCAN created only two clusters to BET solutions. Differently from K-Means++, Cluster 1 of DBSCAN contains the solutions with the best values for COE and FM (Clusters 1 and 2 of K-Means++). Cluster 0 of DBSCAN contains the same solutions of Cluster 0 of K-Means++, as occurred for AGM (Table

TABLE III
AGM SOLUTIONS BY CLUSTER

Solution ID	Solution Fitness (FM, ACLASS, COE)	ED	Cluster Number	
			K-Means	DBSCAN
Solution 1	(684.0 , 12.0 , 32.0)	18.39	0	0
Solution 2	(685.0 , 12.0 , 31.0)	18.97	0	0
Solution 3	(676.0 , 15.0 , 30.0)	10.72	0	0
Solution 4	(667.0 , 15.0 , 31.0)	6.71	0	0
Solution 5	(713.0 , 23.0 , 27.0)	47.34	1	1
Solution 6	(717.0 , 17.0 , 28.0)	50.34	1	1
Solution 7	(785.0 , 33.0 , 25.0)	119.85	2	1
Solution 8	(728.0 , 27.0 , 26.0)	62.83	1	1
Solution 9	(735.0 , 21.0 , 27.0)	68.62	1	1
Solution 10	(711.0 , 17.0 , 29.0)	44.46	1	1
Solution 11	(699.0 , 22.0 , 28.0)	33.67	1	1

TABLE IV
BET SOLUTIONS BY CLUSTER

Solution ID	Solution Fitness (FM, ACLASS, COE)	ED	Cluster Number	
			K-Means	DBSCAN
Solution 1	(1475.0 , 117.0 , 99.0)	55.75	0	0
Solution 2	(1527.0 , 530.0 , 80.0)	427.83	1	1
Solution 3	(1527.0 , 455.0 , 81.0)	355.56	1	1
Solution 4	(1470.0 , 115.0 , 101.0)	51.89	0	0
Solution 5	(1503.0 , 418.0 , 82.0)	313.40	1	1
Solution 6	(1508.0 , 178.0 , 96.0)	107.16	0	0
Solution 7	(1477.0 , 115.0 , 100.0)	57.94	0	0
Solution 8	(1477.0 , 116.0 , 99.0)	57.59	0	0
Solution 9	(1471.0 , 119.0 , 99.0)	52.15	0	0
Solution 10	(1476.0, 118.0 , 98.0)	56.38	0	0
Solution 11	(1476.0 , 121.0 , 97.0)	56.29	0	0
Solution 12	(1461.0 , 543.0 , 83.0)	429.70	2	1
Solution 13	(1498.0 , 489.0 , 84.0)	381.48	1	1
Solution 14	(1423.0 , 483.0 , 86.0)	368.07	2	1
Solution 15	(1573.0 , 515.0 , 79.0)	427.20	1	1
Solution 16	(1428.0 , 483.0 , 85.0)	368.08	2	1
Solution 17	(1471.0 , 116.0 , 100.0)	52.40	0	0

III). In general, those solutions have the best trade-off among the objective functions (the lowest ED values). In almost all solutions, their fitness is better than the original fitness what shows that those solutions have the best compromise among the objectives. DBSCAN has marked Solution 15 as noise. Such a solution has the lowest value of COE but increased the original value of FM and ACLASS. During the Clustering Analysis, such a solution was grouped in Cluster 1.

As mentioned before, both clustering algorithms group the solutions that have the best compromise (trade-off) among the objectives in Cluster 0, which is considered the best performing cluster. In this sense, the inclusion of a clustering algorithm in the a posteriori interaction could reduce the number of solutions to be analyzed from 11 to 4 for AGM and from 17 to 9 for BET. It is interesting to highlight that decision makers usually prefer solutions with the best trade-off among the objectives [39]. However, when the decision maker prefers the solutions with the lowest values of one specific objective, solutions of one of the other clusters should be analyzed. Even so, the number of solutions to be analyzed would be reduced.

In order to corroborate the quantitative analysis, it is necessary to obtain a qualitative analysis of the solutions, which is presented in the next subsection.

B. Qualitative Results

Results of the quantitative experiment show that both clustering algorithms satisfactorily grouped the optimized solutions. However, it is important to assess if such solutions were properly grouped taking into account the user's opinion. To do so, as mentioned in Section III-B, in the qualitative experiment the solutions obtained for AGM were separated for 4 users with the purpose of each one analyzes a specific architectural property of those solutions. Clusters with well-evaluated solutions may be considered well-formed because they really have solutions related to the architectural properties evaluated by users.

Table V shows the solutions used in the qualitative analysis, with their respective ranking, fitness and ED value. Solutions whose ranking is "Best FM and ACLASS" (Solutions 1 to 4) are those ones that have the lowest values for the objective

TABLE V
SOLUTIONS USED IN THE QUALITATIVE ANALYSIS

Solution ID	Ranking	Solution Fitness (FM, ACLASS, COE)	ED
Solution 1	Best FM and ACLASS	(684.0 , 12.0 , 32.0)	18.39
Solution 2	Best FM and ACLASS	(685.0 , 12.0 , 31.0)	18.97
Solution 3	Best FM and ACLASS	(676.0 , 15.0 , 30.0)	10.72
Solution 4	Best FM and ACLASS	(667.0 , 15.0 , 31.0)	6.71
Solution 5	Best COE	(713.0 , 23.0 , 27.0)	47.34
Solution 6	Best COE	(717.0 , 17.0 , 28.0)	50.34
Solution 7	Best COE	(785.0 , 33.0 , 25.0)	119.85
Solution 8	Best COE	(728.0 , 27.0 , 26.0)	62.83
Solution 9	Best COE	(735.0 , 21.0 , 27.0)	68.62
Solution 10	Best COE	(711.0 , 17.0 , 29.0)	44.46
Solution 11	Best COE	(699.0 , 22.0 , 28.0)	33.67

TABLE VI
AGM PLA - SOLUTIONS BY ANALYSIS

PLA ID	FM	ACLASS	COE	Trade-off	Cluster Number	
	User1	User2	User3	User4	DBSCAN	K-Means
Solution 1	X	X		X	0	0
Solution 2	X	X		X	0	0
Solution 3	X	X		X	0	0
Solution 4	X	X		X	0	0
Solution 5			X		1	1
Solution 6			X		1	1
Solution 7			X		-1	2
Solution 8			X		1	1
Solution 9			X		1	1
Solution 10			X		1	1
Solution 11			X		1	1

functions FM and ACLASS. They also represent the solutions with the best trade-off among the objectives, what can be verified by analyzing their ED value. The ranking of the remaining solutions (Solution 5 to 11) is "Best COE", meaning that those solutions have the lowest values of cohesion. Those solutions have greater ED value and worse fitness values for FM and ACLASS than solutions ranked as "Best FM and ACLASS", however, they are design alternatives to be considered depending on what the user aims to prioritize.

Table VI shows which solutions were evaluated by which user. The top of the second to the fifth columns show which architectural property each user has evaluated. This table also presents for each solution its cluster number by clustering algorithm. 11 PLA solutions were qualitatively evaluated. Four of them were in the cluster of the best value for ACLASS, FM and trade-off (Cluster 0), and the others were grouped in the cluster formed by solutions with the best COE values.

Table VII presents the solutions with their respective scores for each user. Table VIII presents the statistical measures on all solutions scores. Only one solution was evaluated with a score equals to 2. The others have scores equal to or greater than 3. In addition, analyzing the medians, it is noticed that the users are partially satisfied with the solutions. It is possible to observe that the user who analyzed the trade-off was the one that better evaluated the solutions (Table VII). We infer that this happened because he considered more than one evaluation criterion, which makes difficult a detailed analysis of each architectural property (objective).

To better understand the scores, the justifications given by each user were taken under consideration [1]. It is possible to observe that User 1 was not so satisfied with the feature modularization. He highlighted, for example, that there are

TABLE VII
QUALITATIVE ANALYSIS: SOLUTIONS SCORE

Solution ID	User 1 (FM)	User 2 (ACLASS)	User 3 (COE)	User 4 (Trade-off)
Solution 1	3	4	-	5
Solution 2	4	3	-	5
Solution 3	2	4	-	4
Solution 4	3	4	-	4
Solution 5	-	-	3	-
Solution 6	-	-	4	-
Solution 7	-	-	4	-
Solution 8	-	-	4	-
Solution 9	-	-	4	-
Solution 10	-	-	3	-
Solution 11	-	-	4	-

TABLE VIII
STATISTICAL MEASURES ON THE SOLUTIONS SCORES

Evaluation	Standard Deviaton	Variance	Median
All	0.7519039	0.5653595	4
User 1	0.8164966	0.6666667	3
User 2	0.5	0.25	4
User 3	0.48795	0.2380952	4
User 4	0.5773503	0.3333333	4.5

packages assigned to more than one feature (e.g. game, ranking and save features). User 2 has analyzed class coupling of the solutions. Some aspects identified were the existence of some unused interfaces (IGameMgt and IAnimationLoop), large number of inheritances and dependencies, and classes with none method. User 3 evaluated cycles between classes, disconnected classes, and the distance between classes and interfaces that are dependent on each other. Finally, User 4, responsible for the trade-off evaluation, observed problems related to partially independent classes and modularized features.

Users were partially satisfied with the obtained solutions, because they assigned good scores to the analyzed solutions. When using MOEAs, the obtained solutions are considered sub-optimal [9]. Hence, the aim of adopting a SBSE technique is to obtain near-optimal solutions. Scores given by users, as well the median of all scores (Table VIII), match with this aim. Considering that the solutions were subjectively analyzed without the users having information about the clusters and the values of the objective functions, the solutions in general were well-evaluated (good scores) what allow to state that, for the analyzed architectural properties, the obtained solutions were properly grouped by the clustering algorithms.

The great variance in the scores can be explained by the fact that the analysis of optimized solutions was completely subjective. The mere fact that a class association is in a different place than the one desired by the user can influence the user's opinion on the solutions. More importantly, we could observe that several justifications about scores match with the solutions fitness, what corroborates that the use of the fitness values to cluster the solutions is appropriate.

V. ANSWERING THE RESEARCH QUESTIONS

After performing the quantitative and the qualitative experiments, it is interesting to discuss some additional points before answering the research questions.

As can be observed in Table VII, the user who analyzed cohesion scored 4 for 5 solutions, including the solution with the best fitness value for cohesion (Solution 7). By observing the clusters formed by both clustering algorithms, it is possible to verify that in DBSCAN those 5 solutions were grouped into the same cluster, but in K-Means++, the solution with the best cohesion was separated in another cluster. Considering the possibility of the user analyze only the solutions that have the best values for cohesion, the cluster formed by DBSCAN contains more design alternatives to show to the user. On the other hand, an important aspect observed is that K-Means++ separated features that DBSCAN did not separate. In Tables III and IV it can be seen that in DBSCAN the PLA solutions presented in only one cluster, were divided into two clusters by K-Means++, one for each objective.

In this context, the following questions can be answered:

RQ1: Which clustering algorithm best fits to support search-based PLA design? The two clustering algorithms presented similar results, with some peculiarities. The clusters with the solutions with best value for trade-off has the same solutions for both algorithms. However, as can be seen in Table IV, DBSCAN considered the solution 15 as noise, and formed the second cluster with the best FM and COE solutions, while in K-Means++ two additional clusters were formed, one for FM and another for COE. When analyzing the ED of the solutions, K-Means++ obtained better results when separating features, as can be observed for BET PLA, in which the best FM and COE solutions were separated in two clusters, whereas DBSCAN kept them in only a cluster. However, when grouped the AGM solutions, DBSCAN obtained better results from the user´s point of view, since it grouped the solution of best COE with solutions that were separated in K-Means++. Therefore, DBSCAN group important solutions from user's point of view that the K-Means++ exclude, but the K-Means++ achieves better results in the features separation. Observing the experiment data, we noticed that DBSCAN fits better when the user is interested in solutions with the best trade-off among the optimized objectives and that K-Means++ fits better when the user is interested in the best value of a specific objective. Such an observation needs to be confirmed in broader and future experiments.

RQ2: Are the solutions best evaluated by the DM grouped into the best performing cluster? The median of scores assigned to solutions of the best performing cluster is 4. Only one solution had score 2. This indicates a good acceptance of thoses PLA solutions by the users. The overall median was good considering that those data come from a subjective evaluation, and that the solutions are sub-optimal. It is important to highlight that the best performing cluster contains the solutions of best trade-off, FM and ACLASS.

VI. CONCLUDING REMARKS

The present work proposed the usage of clustering algorithms (K-Means++ and DBSCAN) to support decision makers in choosing optimized PLA alternatives. Qualitative and quantitative experiments were performed in optimized solutions for BET and AGM PLA designs in order to verify the applicability of this proposal.

It was possible to analyze quantitatively that K-Means++ separated PLA solutions better when considering their objec-

tives, since solutions that were in one DBSCAN cluster were separated in two clusters in K-Means++. However, through the qualitative experiment, it was possible to observe that the solutions presented in DBSCAN cluster were interesting from the user´s point of view. Finally, after analyzing the collected data, it was possible to verify the applicability of the proposal in supporting decision makers in its choices, since in both clustering algorithms, the best PLA solutions were grouped in a single cluster to be evaluated, reducing the number of PLA design alternatives to be analyzed.

Other contributions of this work are a comparison of the algorithms K-Means++ and DBSCAN to cluster the optimized PLA design alternatives and the first joint usage of clustering algorithms and search algorithms in the search-based PLA design. Since this work presents preliminary results, experiments with different values for the input parameters of the clustering algorithms must be conducted in order to verify if changing those values can improve the PLAs solutions clustering.

Two limitations of this work are the reduced number of users which participated in the qualitative experiment and the number of SPLs used in the experiments. Other experiment, with a greater number of users and different SPLs have already been planned and will be conducted. This is important because with different SPLs is possible to obtain a greater diversity of solutions with their particularities, which contributes to increase the amount of evidence about the proposal feasibility.

ACKNOWLEDGMENT

The authors acknowledge CNPq (Process 428994/2018-0) - for the financial support provided to this research.

REFERENCES

[1] (2019) Experimental package. [Online]. Available: https://github.com/wmfsubmissions/experimental-packages-compsac.git

[2] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," in *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035.

[3] C. Birtolo, P. Pagano, and L. Troiano, "Evolving colors in user interfaces by interactive genetic algorithm," in *Nature & Biologically Inspired Computing. NaBIC. World Congress on*. IEEE, 2009, pp. 349–355.

[4] R. J. Campello, D. Moulavi, A. Zimek, and J. Sander, "Hierarchical density estimates for data clustering, visualization, and outlier detection," *ACM Trans. on Knowledge Discovery from Data*, vol. 10.

[5] J. Choma Neto, T. Gaieski, A. M. Amaral, and T. E. Colanzi, "Quanti-qualitative analysis of a memetic algorithm to optimize product line architecture design," in *2018 IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, 2018, pp. 498–505.

[6] C. A. C. Coello, G. B. Lamont, D. A. Van Veldhuizen *et al.*, *Evolutionary algorithms for solving multi-objective problems*. Springer, 2007, vol. 5.

[7] T. E. Colanzi and S. R. Vergilio, "A feature-driven crossover operator for product line architecture design optimization," in *Computer Software and Applications Conference (COMPSAC), IEEE 38th*, 2014, pp. 43–52.

[8] T. E. Colanzi, S. R. Vergilio, I. Gimenes, and W. N. Oizumi, "A search-based approach for software product line design," in *Proceedings of the 18th International Software Product Line Conference-Volume 1*. ACM, 2014, pp. 237–241.

[9] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii," in *International Conference on Parallel Problem Solving From Nature*. Springer, 2000, pp. 849–858.

[10] P. M. Donegan and P. C. Masiero, "Design issues in a component-based software product line." in *SBCARS*. Citeseer, 2007, pp. 3–16.

[11] D. Eppstein, M. S. Paterson, and F. F. Yao, "On nearest-neighbor graphs," *Discrete & Computational Geometry*, vol. 17, no. 3.

[12] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *Kdd*, vol. 96, no. 34, 1996, pp. 226–231.

[13] É. L. Féderle, T. do Nascimento Ferreira, T. E. Colanzi, and S. R. Vergilio, "Opla-tool: a support tool for search-based product line architecture design," in *Proceedings of the 19th International Conference on Software Product Line*. ACM, 2015, pp. 370–373.

[14] A. L. Fred and A. K. Jain, "Data clustering using evidence accumulation," in *null*. IEEE, 2002, p. 40276.

[15] M. Harman and B. F. Jones, "Search-based software engineering," *Information and Soft. Technology*, vol. 43, no. 14, pp. 833–839, 2001.

[16] M. Harman, S. A. Mansouri, and Y. Zhang, "Search-based software engineering: Trends, techniques and applications," *ACM Computing Surveys (CSUR)*, vol. 45, no. 1, p. 11, 2012.

[17] S. M. H. Hasheminejad and S. Jalili, "An evolutionary approach to identify logical components," *Journal of Systems and Software*, vol. 96.

[18] A. K. Jain and R. C. Dubes, *Algorithms for clustering data*. Prentice-Hall, 1988.

[19] H. K. Kanagala and V. J. R. Krishnaiah, "A comparative study of k-means, dbscan and optics," in *Computer Communication and Informatics (ICCCI), 2016 International Conference on*. IEEE, 2016, pp. 1–6.

[20] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, and A. S. Peterson, "Feature-oriented domain analysis (foda) feasibility study," DTIC Document, Tech. Rep., 1990.

[21] J. D. Knowles and D. W. Corne, "Approximating the nondominated front using the pareto archived evolution strategy," *Evolutionary computation*, vol. 8, no. 2, pp. 149–172, 2000.

[22] T. M. Kodinariya and P. R. Makwana, "Review on determining number of cluster in k-means clustering," *International Journal*, vol. 1, no. 6.

[23] H.-P. Kriegel, P. Kröger, J. Sander, and A. Zimek, "Density-based clustering," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 1, no. 3, pp. 231–240, 2011.

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

[25] S. Lloyd, "Least squares quantization in pcm," *IEEE transactions on information theory*, vol. 28, no. 2, pp. 129–137, 1982.

[26] T. S. Madhulatha, "An overview on clustering methods," *arXiv preprint arXiv:1205.1117*, 2012.

[27] A. Marcolino, E. Oliveira, I. Gimenes, and E. F. Barbosa, "Empirically based evolution of a variability management approach at uml class level," in *2014 IEEE 38th Annual Computer Software and Applications Conference (COMPSAC)*. IEEE, 2014, pp. 354–363.

[28] D. Meignan, S. Knust, J.-M. Frayret, G. Pesant, and N. Gaud, "A review and taxonomy of interactive optimization methods in operations research," *ACM Transactions on IIS*, vol. 5, no. 3, p. 17, 2015.

[29] C. Nunes, U. Kulesza, C. Sant'Anna, I. Nunes, A. Garcia, and C. Lucena, "Assessment of the design modularity and stability of multi-agent system product lines," *Journal of Universal Computer Science*, vol. 15, no. 11, pp. 2254–2283, jun 2009.

[30] E. OliveiraJr, I. M. S. Gimenes, and J. C. Maldonado, "Systematic Management of Variability in UML-based Software Product Lines," *Journal of Universal Computer Science*, 2010.

[31] V. Pareto, *Manuel d'économie politique*. Paris, M. Giard, 1927.

[32] A. M. Pitangueira, "Incorporating preferences from multiple stakeholders in software requirements selection an interactive search-based approach," in *Requirements Engineering Conference (RE), 2015 IEEE 23rd International*. IEEE, 2015, pp. 382–387.

[33] K. Pohl, G. Böckle, and F. J. van Der Linden, *Software product line engineering: foundations, principles and techniques*. Springer, 2005.

[34] W. M. Rand, "Objective criteria for the evaluation of clustering methods," *Journal of the American Statistical Association*, vol. 66.

[35] P. J. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *Journal of computational and applied mathematics*, vol. 20, pp. 53–65, 1987.

[36] SEI. (2009) Software Engineering Institute - the Arcade Game Maker pedagogical product line. Https://resources.sei.cmu.edu/library/asset-view.cfm?assetID=485941. Accessed in 2018 August.

[37] F. Van der Linden, K. Schmid, and E. Rommes, "The product line engineering approach," in *SPL in Action*. Springer, 2007, pp. 3–20.

[38] Y. D. Verdecia, T. E. Colanzi, S. R. Vergilio, and M. dos Santos, "An enhanced evaluation model for search-based product line architecture design." in *CIbSE*, 2017, pp. 155–168.

[39] M. Zeleny and J. L. Cochrane, *Multiple criteria decision making*. University of South Carolina Press, 1973.