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

Yenisei Delgado Verdecia

Expansão de um modelo de avaliação de arquitetura de linha de produto de software

Maringá
2017

Yenisei Delgado Verdecia

Expansão de um modelo de avaliação 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. Thelma Elita Colanzi

Maringá
2017

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

Verdecia, Yenisei Delgado
V483e Expansão de um modelo de avaliação de arquitetura de linha de produto de software/ Yenisei Delgado Verdecia. -- Maringá, 2017.
85 f. il. : figs., color., tabs.

Orientadora: Prof.a. Dr.a. Thelma Elita Colanzi.

Dissertação (mestrado) – Universidade Estadual de Maringá, Centro de Tecnologia, Departamento de Informática, Programa de Pós-Graduação em Ciência da Computação, 2017.

1. Linha de produto de software. 2. Projeto de arquitetura de LPS. 3. Modelo de avaliação. 4. Métricas de software. 5. Engenharia de software baseada em busca. I. Colanzi, Thelma Elita, orient. II. Universidade Estadual de Maringá. Centro de Tecnologia. Departamento de Informática. Programa de Pós-Graduação em Computação. III Título.

CDD 22. ED. 004.22
JLM001623

FOLHA DE APROVAÇÃO

YENISEI DELGADO VERDECIA

Expansão de um modelo de avaliação 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. Thelma Elita Colanzi Lopes
Universidade Estadual de Maringá – DIN/UEM

Profa. Dra. Gislaine Camila Lapasini Leal
Universidade Estadual de Maringá – DIN/UEM

Profa. Dra. Silvia Regina Vergilio
Universidade Federal do Paraná – DInf/UFPR

Aprovada em: 31 de março de 2017.
Local da defesa: Sala 120, Bloco C56, *campus* da Universidade Estadual de Maringá.

DEDICATÓRIA

A minha mãe, por acreditar em mim o tempo todo.
Muito obrigada pela confiança!

AGRADECIMENTOS

"Grandes coisas têm feito o Senhor por mim, por isso me sinto feliz"

Agradeço a Deus por ter me abençoado nesta caminhada e as pessoas que contribuíram na realização deste trabalho. Em especial:

À minha mãe, por todo o apoio e força que me deu. Mãe, obrigada por acreditar em mim o tempo todo!

À minha irmã, por ser a mais crítica e direta e por me aturar nos momentos difíceis;

Ao meu pai pelo carinho, apoio e incentivo;

À minha orientadora Dra. Thelma Elita Colanzi, Chegamos ao fim! Muito obrigada pela oportunidade e saiba que sou eternamente grata por tudo;

À Anita Winkler e ao Edgardo Gómez por terem me acolhido como uma filha. Infinitamente agradecida por ser parte de sua maravilhosa família. Sem eles, minha jornada seria impossível;

À Elenir Voi Xavier pelo amor incondicional! Uns mimos são maravilhosos quando precisamos de força para continuar;

À Florencia Ghiggino por me ouvir, por me apoiar, por me puxar para cima quando estava mais sensível e por me ajudar a ultrapassar todas as barreiras assim como ao seu marido, Gerard, o nosso Gordo, que suportou o nosso "cotorreo";

À Maydelin Azaharez e Eduardo Arteaga. Não importa a que distância, eles sempre estão comigo;

Ao Enrique Monteagudo. Obrigada por compreender o quanto este passo foi importante para mim!

Ao Eduardo Vicente Valdes, Yurdi Cervantes e Laya Rabasa pelo incontável apoio e presença constante;

À minha tia, Miriam Verdecia, e primos, Yunior e Yudel, pela ajuda em todos os aspectos!

A Cleudinete, Akiyochi, Valéria, Claudia, Kerlei, Loia, Jose, Marlete, Alcides e Ana de Freitas, por todo o carinho e amor que me deram. Muito obrigada por se tornar minha família e estarem a meu lado nos melhores e piores momentos destes dois anos! Amo vocês!

Ao Édipo Féderle pela paciência com meu Espanhol e por prontamente me ajudar sempre que o procurei;

À Dra. Camila Lapasini e ao Dr. Donizete C. Bruzarosco pelas valiosas críticas e sugestões na realização da qualificação;

À Dra. Linnyer B. Ruiz Aylon obrigada pelo aquele abraço que me deu forças para seguir adiante;
Ao Dr. Nardênio Almeida Martins, pelos conselhos que me impulsionaram a continuar a caminhada;
À Dra. Aline M. Malachini Miotto Amaral, pela simpatia e gentileza que imprimiu ao nosso contato;
A todos os professores do mestrado pelo prazer em ensinar;
À Dra. Silvia Regina pela disponibilidade e tempo dedicado na leitura da minha tese;
À Mayra Fretes, Marina Aza, Sabrina Magloire e Mirna Aquino, pelas "tardes de chicas";
À Isela Mendoza e Carola Gonzáles, pelas oportunas ligações que me fizeram muito feliz;
À Dayana Montero e Javier Jiménez pelas dicas e conselhos. Obrigada por serem parte desta etapa de minha vida;
Aos demais professores das disciplinas que cursei e aos colegas do mestrado.
À Universidade Estadual de Maringá e ao ILG;
A Cnpq pelo apoio financeiro concedido a este trabalho;
Ao Brasil por me ensinar "**a nunca desistir dos meus sonhos**";
A todos aqueles que contribuíram de forma directa ou indireta;
Muito Obrigada!

Expansão de um modelo de avaliação de arquitetura de linha de produto de software

RESUMO

O objetivo principal de uma Linha de Produto de Software (LPS) é reduzir custos e aumentar a produtividade por meio do reúso de artefatos. Nessa abordagem, o principal artefato é a arquitetura da LPS (*Product Line Architecture* - PLA) porque ela fornece uma solução para uma gama de produtos da LPS. A avaliação do projeto de uma PLA é uma tarefa fundamental e complexa durante o ciclo de vida de LPS, dado que a PLA contem um conjunto de componentes reutilizáveis na LPS. A busca por soluções para problemas da Engenharia de Software tem sido tratada no campo de pesquisa denominado Engenharia de Software Baseada em Busca (*Search Based Software Engineering* - SBSE) por meio de abordagens baseadas em Algoritmos Evolutivos Multiobjetivos (*Multi-Objective Evolutionary Algorithms* - MOEAs), que consideram diferentes fatores e medidas que afetam o problema do projeto de PLA. Diante disso, foi proposta a abordagem *Multiobjective Optimization Approach for Product Line Architecture*-(MOA4PLA), composta por funções objetivo para avaliar projetos de PLA, as quais utilizam diferentes métricas de software para avaliar modularidade de características, extensibilidade de PLA e princípios básicos de projeto. No entanto, outras propriedades arquiteturais podem ser avaliadas. Nesse contexto, o objetivo do presente trabalho foi expandir o modelo de avaliação da MOA4PLA com métricas que avaliem outras propriedades de projeto de PLA. Portanto, foi definido um modelo de qualidade para a MOA4PLA. Logo foram identificadas outras métricas que permitam medir outras propriedades arquiteturais no modelo de qualidade proposto. Tendo em conta as novas métricas selecionadas, novas funções objetivo foram definidas no modelo de avaliação da MOA4PLA e implementadas na ferramenta OPLA-Tool. Em vista de investigar a correlação existente entre as novas funções objetivo, realizou-se um estudo exploratório envolvendo quatro experimentos. Assim, o presente trabalho expande o modelo de avaliação da MOA4PLA, com outras métricas para avaliar projetos de PLA. As principais contribuições do trabalho são: a proposta do modelo de qualidade para SBPD, o uso de métricas para a LPS orientado a serviços aplicadas por primeira vez no contexto de SBPD; e as novas funções objetivo que permitem avaliar outras propriedades arquiteturais no modelo de avaliação da MOA4PLA.

Palavras-chave: Linha de Produto de Software. Projeto de Arquitetura de LPS. Modelo de Avaliação. Métricas de Software. Engenharia de Software baseada em Busca.

Expansion of an evaluation model of software product line architecture

ABSTRACT

The primary purpose of a Software Product Line (LPS) is to reduce costs and increase productivity by reusing artifacts. In this approach, the main artifact is the Product Line Architecture (PLA) because it provides a solution for a range of LPS products. Evaluating the PLA design is a fundamental and complex task during the LPS life cycle, since the PLA specifies a set of reusable components in the LPS. The search for solutions to software engineering problems has been addressed in the search field called Search- based Software Engineering (SBSE) through approaches based on Multi-Objective Evolutionary Algorithms (MOEAs), which consider different factors and measures that affect the PLA design problem. In order to evaluate PLA designs, which use different software metrics to evaluate the modularity of characteristics, the PLA extensibility and the basic principles of design, the Multi-Objective Approach for Product Line Architecture Design (MOA4PLA) was proposed. However, other architectural properties can be evaluated. In this context, the objective of the present work was to expand the MOA4PLA evaluation model with metrics that assess other PLA design properties. Therefore, a quality model was defined for the MOA4PLA. After, other metrics have been identified to measure other architectural properties in the proposed quality model. Taking into account the new selected metrics, new objective functions were defined in the MOA4PLA evaluation model and implemented in the tool OPLA-tool. In order to investigate the correlation between the new objective functions, an exploratory study was carried out involving four experiments. Thus, the present work expands the evaluation model of the MOA4PLA, with other metrics to evaluate PLA designs. The main contributions of this work are: the proposal of the quality model for SBPD, the use of metrics for LPS oriented to services applied for the first time in the context of SBPD; and the new objective functions that let evaluate other architectural properties in the evaluation model of MOA4PLA.

Keywords: Software Product Line. Product Line Architecture Design. Evaluation Model. Software Metrics. Search based Software Engineering.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS E ABREVIATURAS

ACOMP: Acoplamento de Componentes
ACLASS: Acoplamento de Classes
PLA: Arquitetura de Linha de Produto de Software
AGM: *Arcade Game Maker*
CDAC: *Concern Diffusion over Architectural Components*
CDAI: *Concern Diffusion over Architectural Interfaces*
CDAO: *Concern Diffusion over Architectural Operations*
CDepIn: Dependência de Entrada de uma classe
CDepOut: Dependências de Saída de uma Classe
CIBC: *Component-level Interlacing Between Concerns*
CM: Métricas Convencionais
CS:*Component Size*
CV:*Coupling of Variability*
DC: Difusão de Características
DepIn: Dependências de Entrada de um Pacote
DepOut: Dependência de Saída de um Pacote
DC: Difusão de Características
EC: Entrelaçamento de Características
FM: Métricas dirigidas a Características
GA: *Genetic Algorithms*
H: Coesão Relacional
IIBC: *Interface-level Interlacing Between Concerns*
LCC: *Lack of Concern-based Cohesion*
LPS: Linha de Produto de Software
MOA4PLA: *Multi-Objective Approach for Product-Line Architecture Design*
MM: *Mobile Media*
MOEA: *Multi-Objetive Evolutionary Algorithms*
NSGA-II: *Non-dominated Sorting Genetic Algorithm*
NumOps: Número de Operações por interface
OOBC: *Operation-level Overlapping Between Concerns*
OPLA-Tool: *Optimization for PLA Tool*
PAES: *Pareto Archived Evolution Strategy*
RCC: *Relationships Coupling of Components*
SBSE: *Search-based Software Engineering*

SBPD: *Search-based PLA Design*
SD: *Similarity of Design*
SV: *Structure of Variability*
TAM: Tamanho
TV: *Total Variability*
UML: *Unified Modeling Language*

SUMÁRIO

1

Introdução

1.1 Contextualização

A mudança para o desenvolvimento baseado em reuso foi uma resposta às exigências de menores custos de desenvolvimento e manutenção, de sistemas de maior qualidade e de entregas mais rápidas de sistemas (Sommerville, 2011). Diante disso, o processo de desenvolvimento de sistemas de software começa a ter um novo enfoque, diferente do utilizado tradicionalmente, visando ao reuso sistemático e cumprimento de exigências específicas dos clientes. O reuso contínuo de sistemas de software consiste de uma mudança da abordagem de desenvolvimento de sistemas únicos para o desenvolvimento de famílias de sistemas (Linden et al., 2007). Com esse enfoque, busca-se obter maior rendimento na produtividade, produtos mais confiáveis e preço mais acessível.

Uma abordagem de reuso de software tem ganhado atenção tanto da indústria quanto da academia ao longo dos últimos anos. Esta abordagem é conhecida como Linha de Produto de Software (LPS) e é baseada na reutilização sistemática de artefatos de software, por meio da exploração de pontos comuns e a gestão de variabilidade entre os produtos, que são estabelecidos sob uma mesma arquitetura (Pohl et al., 2005). O principal artefato de referência da LPS é a arquitetura da LPS (*Product Line Architecture* - PLA), porque ela é o vínculo entre as necessidades do cliente, o domínio que deve atender a LPS e o novo produto que será instanciado.

Tendo em conta que o objetivo principal de uma LPS é reduzir o tempo, esforço, custo e complexidade da criação e manutenção dos produtos da LPS, a definição da PLA representa uma atividade que demanda grande esforço, porque ela descreve uma arquitetura genérica que fornece uma solução para a gama de produtos da LPS. Deste

modo, a PLA representa a abstração das possíveis arquiteturas de potenciais produtos específicos que a LPS pode gerar.

Nessa perspectiva, a PLA torna-se um dos artefatos mais importantes para gerar produtos específicos com sucesso durante todo o ciclo de vida da LPS. Portanto, quanto mais extensíveis forem os componentes da arquitetura, maior é sua taxa de reusabilidade (OliveiraJr et al., 2013). Sendo assim, é importante analisar a arquitetura para predizer sua qualidade antes da derivação dos produtos, de modo a identificar potenciais riscos e verificar se os requisitos de qualidade estão sendo tratados no projeto (Colanzi e Vergilio, 2012).

Para poder ter uma visão da qualidade de uma PLA, medir é muito importante e quando se trata de avaliação quantitativa, quatro conceitos são fundamentais: medida, medição, métrica e indicador. Nesse ponto de vista, uma medida fornece uma indicação quantitativa da extensão, quantidade, dimensão, capacidade ou tamanho de um atributo de um produto ou de um processo. Medição é o ato de medir, isto é, de determinar uma medida. Já uma métrica procura relacionar medidas individuais com o objetivo de se ter uma ideia da eficácia do processo, do projeto ou do produto sendo medido. Por fim, desenvolvem-se métricas para se obter indicadores, isto é, para se ter uma compreensão do processo, produto ou projeto sendo medido (Pressman, 2011).

Estudos sobre avaliação estrutural de PLA frequentemente consideram um conjunto de métricas para fornecer indicadores sobre diferentes propriedades (Oizumi et al., 2012). Desse modo as métricas podem fornecer indicadores sobre propriedades arquiteturais de interesse do arquiteto. Por conseguinte, métricas para a avaliação de PLA consideram aspectos inerentes a estrutura de arquiteturas de LPS como características variáveis e opcionais.

Encontrar o melhor balanceamento de prioridades (*trade-off*) entre as métricas assim como, escolher quais devem ser priorizadas é uma tarefa difícil que precisa grande esforço e tempo. Nesse ponto de vista a Engenharia de Software Baseada em Busca (*Search Based Software Engineering* - SBSE) (Harman et al., 2012) fornece técnicas para avaliar possíveis alternativas para um projeto de uma PLA, visto que, os problemas de Engenharia de Software podem ser formulados como problemas de otimização a serem resolvidos por meio de técnicas baseadas em busca (Colanzi et al., 2014).

Abordagens baseadas em Algoritmos Evolutivos Multiobjetivos (*Multi-Objective Evolutionary Algorithms* - MOEAs) têm sido aplicadas para otimizar o projeto de software ou otimizar a ordem de refatorações a serem aplicadas ao projeto (Räihä, 2010), dado que eles permitem encontrar uma melhor solução dentro de um conjunto de possíveis soluções. Neste ponto de vista, foi proposta a abordagem de otimização multiobjetivo denominada

Multiobjective Optimization Approach for Product Line Architecture (MOA4PLA) (Colanzi et al., 2014).

A MOA4PLA é uma abordagem de otimização multiobjetivo proposta para identificar automaticamente as melhores alternativas de projeto para uma PLA e contribuir para diminuir o esforço dos arquitetos durante o projeto e avaliação de uma PLA (Colanzi et al., 2014). O principal foco da MOA4PLA é a avaliação e a melhoria do projeto de PLA, considerando múltiplos objetivos (métricas), independentemente do algoritmo de busca adotado, visando facilitar o trabalho do arquiteto no momento da definição do projeto da PLA. Ela inclui um processo para realizar a otimização multiobjetivo de um projeto de PLA, um metamodelo para representar os projetos de PLA, assim como operadores de busca convencionais e para a modularização de características. A abordagem é automatizada mediante uma ferramenta denominada *Optimization for PLA Tool* (OPLA-Tool).

1.2 Motivação e Objetivos

Atualmente o modelo de avaliação da MOA4PLA só mede extensibilidade, elegância, modularização e princípios básicos de projeto como tamanho, coesão e acoplamento outras medidas de interesse para avaliar projeto de PLA. No entanto, existem outras propriedades arquiteturais interessantes ainda não avaliadas no modelo como: ***Similaridade*** e ***Variabilidade***. Conhecer o nível de similaridade e variabilidade permite ter uma visão de como ambas propriedades são atendidas por elementos arquiteturais existente na PLA assim como, da uma visão de como os produtos podem ser construídos a partir de artefatos comuns (Clements e Northrop, 2001).

Isso motivou um estudo cujo principal objetivo é **enriquecer o modelo de avaliação, com novos objetivos a serem otimizados pela abordagem, tendo em conta outras propriedades arquiteturais ainda não avaliadas no modelo**. Nesse sentido, direcionar a avaliação quantitativa de níveis de qualidade no contexto de avaliação de projeto de PLA para SBPD é o ponto de partida para a extensão do modelo em vista de obter um modelo de qualidade para o modelo de avaliação que direcione quais características de qualidade, atributos de qualidade e propriedades arquiteturais estam sendo avaliadas pelas funções objetivo compostas pelas métricas.

No entanto, para se cumprir o objetivo mencionado foram estabelecidos os seguintes objetivos específicos: (i) Propor um modelo de qualidade para o modelo de avaliação da MOA4PLA; (ii) Classificar as métricas das funções objetivo existentes no modelo de avaliação da MOA4PLA, em vista de mapear propriedades arquiteturais que medem com atributos de qualidade; (iii) Selecionar métricas para LPS a fim de medir as propriedades

arquiteturais da PLA ainda não cobertas pela MOA4PLA; (iv) Propor novas funções objetivo baseadas nas métricas selecionadas; (v) Implementar as novas funções propostas na ferramenta OPLA-Tool; e (vi) Planejar e conduzir estudo exploratório para identificar a possível correlação existente entre algumas das novas funções propostas.

1.3 Método de pesquisa

Para atingir o objetivo proposto, uma sequência de atividades, mostradas na Figura - 1.1, foi realizada para a extensão do modelo de avaliação de projeto de PLA:

(i) Definir modelo de qualidade: em vista de direcionar a avaliação quantitativa de níveis de qualidade no contexto de avaliação de projeto de PLA para SBPD o qual segue o meta-modelo, onde as características de qualidade são compostas por atributos de qualidade que contêm propriedades arquiteturais que podem ser medidas por uma ou várias métricas,

(ii) Mapear métricas no modelo de qualidade: em virtude de mapear as métricas existentes no modelo de avaliação atual, no modelo de qualidade proposto tendo em conta as propriedades arquiteturais que medem e dessa forma obter uma visão geral de quais características de qualidade, atributos de qualidade e propriedades arquiteturais estão sendo medidas no modelo,

(iii) Selecionar novas métricas: devido a no modelo de qualidade proposto existem outras propriedades arquiteturais ainda não avaliadas como similaridade e variabilidade que permitem enriquecer o modelo de avaliação, é necessário investigar via mapeamento sistemático métricas que permitam medir as propriedades arquiteturais mencionadas anteriormente assim como realizar uma prova de conceito para averiguar a viabilidade de inserir as métricas selecionadas no modelo de avaliação,

(iv) Propor novas funções objetivo: se definiram novas funções objetivo para o modelo de avaliação, baseadas nas métricas selecionadas,

(v) Implementar novas funções objetivo: as novas funções objetivo propostas foram implementadas na ferramenta OPLA-Tool para o qual foram necessárias algumas alterações nos módulos OPLA-GUI e OPLA-Core, e;

(vi) Realizar estudo exploratório: uma vez estendido o modelo de avaliação com novas funções objetivo, realizou-se um estudo exploratório para investigar a correlação existente entre aquelas funções objetivo que medem propriedades arquiteturais como similaridade, variabilidade e acoplamento. A realização desse estudo exploratório foi organizado tendo em conta as seguintes fases: (i) definição; (ii) planejamento; (iii) operação e; (iv) interpretação e para dar efeito ao objetivo do estudo foram realizados quatro experimentos

usando diferentes combinações de funções objetivo relativo as métricas referentes a similaridade, varabilidade e acoplamento.

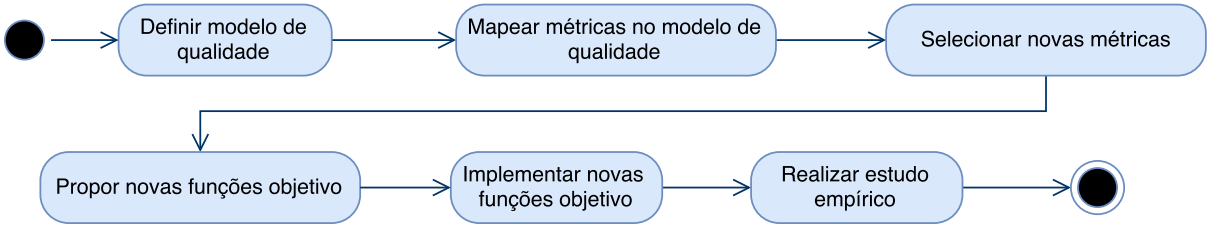

Figura 1.1: Atividades para a extensão do modelo de avaliação de projeto de PLA

1.4 Organização do trabalho

Sendo assim, o próximo capítulo, apresenta os principais conceitos a respeito de LPS, com relevância para projeto de PLA. Ademais são apresentados conceitos referentes a otimização multiobjetivo e algoritmos genéticos, assim como é apresentada a abordagem de otimização multiobjetivo para projeto de PLA, modelos de qualidade e métricas. No Capítulo 3 encaminha-se a extensão do modelo de avaliação incluindo as 5 primeiras atividades apresentadas na Figura - 1.1. No Capítulo 4 aborda-se a realização do estudo exploratório e as ameaças à validade. Finalmente o Capítulo 5 destina-se às conclusões do trabalho.

2

Fundamentação Teórica

Com a finalidade de familiarizar o leitor com os principais conceitos relacionados aos objetivos deste trabalho definidos no Capítulo 1, neste capítulo são apresentados os principais conceitos relacionados com a Linha de Produto de Software (LPS), com ênfase na a arquitetura da LPS (*Product Line Architecture* - PLA), otimização multi-objetivo e algoritmos genéticos. Ademais é apresentada uma abordagem de otimização multiobjetivo para projeto de PLA assim como modelos de qualidade e métricas.

Portanto na Seção 2.1 apresenta-se uma visão geral sobre LPS, destacando a importância da PLA para gerar produtos específicos durante o ciclo de vida da LPS. Na Seção 2.2 aborda-se como a otimização multi-objetivo pode contribuir para melhorar o projeto de arquiteturas de LPS e como os algoritmos genéticos fornecem maior diversidade de soluções no contexto. Na Seção 2.3 mostra-se a abordagem de otimização multiobjetivo para projeto de PLA, a ferramenta que automatiza a MOA4PLA e são descritas as funções objetivo disponíveis do modelo de avaliação. Por último na Seção 2.4 refere-se a importância dos modelos de qualidades e como as métricas contribuem na medição de propriedades arquiteturais.

2.1 Linha de Produto de Software (LPS)

Uma das abordagens mais promissoras para o reuso é a abordagem de LPS (Pohl et al., 2005), como forma de reusar componentes de sistemas de software entre aplicações semelhantes (Clements e Northrop, 2001). A LPS pretende explorar as semelhanças e as diferenças dentro de uma família de sistemas em um campo particular de aplicação e

fornece uma infraestrutura para a derivação de membros comuns dessa família (Mendonça, 2009). O conceito de desenvolvimento de sistemas de software em famílias consiste basicamente na criação e na manutenção de uma LPS como base para o desenvolvimento de um conjunto de sistemas relacionados a partir de artefatos comuns (Clements e Northrop, 2001; Pohl et al., 2005). Utilizando LPS, o processo de desenvolvimento de sistemas de software começa a ter novo enfoque, que promete ganho de produtividade e produto com mais confiabilidade e qualidade, em menor tempo e a preço mais acessível, o que colabora para o aumento considerável na satisfação dos clientes (Dursckil et al., 2004). Assim, o reuso sistemático de sistemas de software consiste em uma mudança da abordagem de construção de sistemas únicos para o desenvolvimento de famílias de sistemas (Pohl et al., 2005).

Essa vantagem é alcançada, pois a PLA é projetada de um modo bem específico: características comuns a um domínio formam o núcleo e características variáveis definem pontos de variação (Jacobson, 2004). Uma característica (do inglês *feature*) é uma capacidade do sistema que é relevante e visível para o usuário final (Kang et al., 1998). Uma característica pode ser de um dos seguintes tipos: obrigatória, opcional ou alternativa. As características obrigatórias devem estar presentes em todos os membros da LPS, as opcionais podem ou não estar presente em um membro da LPS e as alternativas representam os grupos de caraterísticas, uma caraterística alternativa pode agrupar duas ou mais caraterísticas, podendo ser mutuamente exclusivas.

O modelo de características é utilizado para apresentar em alto nível as principais características comuns e variáveis em uma LPS.

"A variabilidade é a capacidade de alterar ou adaptar um sistema. Aumentar a variabilidade de um sistema implica torná-lo mais fácil a certos tipos de alterações " (Gurp et al., 2001).

Na LPS, as variabilidades são modeladas na PLA fazendo uso da representação dos pontos de variabilidade e variantes. Um ponto de variabilidade corresponde a um aspecto de variação funcional num elemento de software base. O ponto de variabilidade define o conjunto de possíveis variantes, o mecanismo de variabilidade a utilizar para os instanciar e o tempo de ativação das variantes, por exemplo instanciação da arquitetura de software do produto, tempo de compilação, execução. Um variante corresponde a uma opção do conjunto de possíveis instâncias de variação que um ponto de variabilidade poderá originar (Linden et al., 2007).

A organização de uma LPS engloba três atividades essenciais (Linden et al., 2007; SEI, 2016): (i) desenvolvimento do núcleo de artefatos, no qual é desenvolvido o núcleo de artefatos; (ii) desenvolvimento do produto, em que os produtos da LPS são gerados

e; (iii) gerenciamento da LPS. O núcleo de artefatos é constituído dos recursos comuns a LPS e um desses principais recursos é a PLA. A atividade de desenvolvimento do núcleo de artefatos pode ser chamada de engenharia de domínio, assim como a atividade de desenvolvimento do produto pode ser chamada de engenharia de aplicação.

2.1.1 Arquitetura da LPS (PLA)

Dentro da LPS o principal artefato de referência é a arquitetura da LPS (PLA), porque ela é o vínculo entre as necessidades do cliente e o domínio que deve atender a LPS e o novo produto que será instanciado.

"Uma arquitetura de uma linha de produtos de software é uma arquitetura de software que satisfaz as necessidades da linha de produto em geral e os produtos individuais em particular, explicitamente admitindo um conjunto de pontos de variabilidade necessários para suportar o espectro dos produtos no âmbito da linha de produtos" (Clements e Northrop, 2001).

A modularidade, reusabilidade, extensibilidade e estabilidade são fatores que impactam na definição da arquitetura e o arquiteto de LPS deve considerar o impacto sobre a solução. Por conseguinte, o arquiteto desempenha um papel fundamental na concepção da LPS, e é responsável por propor um bom projeto arquitetural, priorizar os elementos que podem entrar em conflito e definir os componentes que representam a PLA.

Além da maximização de propriedades como modularidade, reusabilidade, extensibilidade e estabilidade, a presença de pontos de variação inter-relacionados dificulta a especificação inicial da PLA assim como a sua subsequente evolução (López et al., 2009). Sob outra perspectiva, a similaridade estrutural entre as variantes, resultante da arquitetura comum, favorece o reuso de componentes nos diferentes produtos da LPS.

Nesse ponto de vista, a definição e a avaliação do projeto de PLA é uma atividade importante ao longo do ciclo de vida da LPS (Edwards et al., 2008), e é ainda mais complexa do que o projeto de software único. Portanto, a definição de uma PLA é uma tarefa fundamental para a geração dos produtos da LPS e para facilitar seu gerenciamento. Uma PLA contém todas as variabilidades da LPS, razão pela qual é possível a geração da arquitetura de cada produto individualmente. Reconhecer um bom projeto pode ser fácil para arquitetos, mas por outro lado, é difícil obtê-lo (Colanzi et al., 2014).

As métricas podem fornecer indicadores sobre propriedades arquiteturais de interesse do arquiteto. Para isso, os valores de métricas precisam ser analisados pelos arquitetos durante o desenvolvimento do projeto e antes da geração dos produtos da LPS. No entanto, encontrar o melhor balanceamento de prioridades (*trade-off*) entre as métricas assim

como, escolher quais devem ser priorizadas é uma tarefa difícil que precisa grande esforço e tempo. Nesse ponto de vista a Engenharia de Software baseada em busca (*Search-based Software Engineering* - SBSE) (Harman et al., 2012) fornece técnicas para avaliar possíveis alternativas para o projeto de uma PLA, cujos principais conceitos são apresentados na próxima seção.

2.2 Otimização multi-objetivo

A SBSE é uma área da Engenharia de Software na qual seus problemas são reformulados e modelados como problemas de otimização e, posteriormente, são resolvidos utilizando conceitos, técnicas, algoritmos e estratégias de busca (Harman et al., 2012). O objetivo da busca é identificar, dentre todas as soluções possíveis de um problema da Engenharia de Software, uma solução que seja suficientemente boa de acordo com uma métrica apropriada. A reformulação permite que problemas previamente resolvidos de forma intensivamente manual e intuitiva possam ser resolvidos, total ou parcialmente, de forma sistemática e automatizada.

Os problemas de otimização com duas ou mais funções objetivo são chamados de multi-objetivos e têm como objetivo minimizar ou maximizar uma função ou grupo de fatores. Este tipo de problema exige a otimização simultânea de vários interesses interdependentes e muitas vezes conflitantes e, por isso, não há uma solução única. Assim, para solucionar problemas multi-objetivos, procura-se encontrar soluções que melhor representem o compromisso entre os objetivos (Harman et al., 2009).

As funções objetivo empregadas nos problemas de otimização multi-objetivo são em geral conflitantes entre si. Uma função objetivo f_1 é conflitante com uma outra função f_2 quando não é possível melhorar o valor de f_1 sem piorar o valor da função f_2. A razão entre a quantidade que deve ser aumentada para um objetivo a fim de que haja diminuição de um outro objetivo é denominada de *trade-off*. Os *trade-offs* e as soluções eficientes são dois tipos de informações importantes para que o tomador de decisão possa escolher a solução de melhor compromisso.

Um exemplo prático de um problema com objetivos conflitantes são preço e desempenho na compra de equipamentos, por exemplo, de câmeras digitais reflex. As câmeras digitais reflex de melhor desempenho são as de maior custo usualmente e vice-versa. Dessa forma, em uma compra devem ser analisados vários modelos de câmeras digitais reflex com diversos valores nos objetivos de custo e objetivo reflex. Se ambos objetivos possuem a mesma importância, não é possível afirmar, por exemplo, que certa redução do preço compensa certa perda de desempenho. Igualmente, existe um conjunto de soluções que

possuem vantagens em desempenho, mas que não são melhores em custo e vice-versa. Isto é, existe um conjunto de alternativas ótimas que são não-dominadas entre si nos objetivos custo e reflex.

O conjunto de soluções não-dominadas em um problema de otimização multi-objetivo é chamado de conjunto Pareto-ótimo, o qual representa as soluções ótimas do problema. A fronteira de Pareto é o conjunto de valores das funções objetivo das soluções do conjunto Pareto-ótimo. A Figura - 2.1 mostra os valores de custo e objetivo reflex (de 0 a 200 mm) de várias alternativas para o exemplo de compra de câmeras digitais reflex. Nessa figura os pontos representam modelos de câmaras digitais onde: (i) os modelos 1 e 2 são dominados pelos modelos (3, 4, 5 e 6) e deste modo devem ser descartados; (ii) os modelos 3, 4, 5 e 6 são não dominados, portanto, consideram-se as melhores opções de câmaras reflex, formando o conjunto de Pareto-ótimo ; e (iii) a união de todos os pontos dos modelos não dominados, forma a fronteira de Pareto.

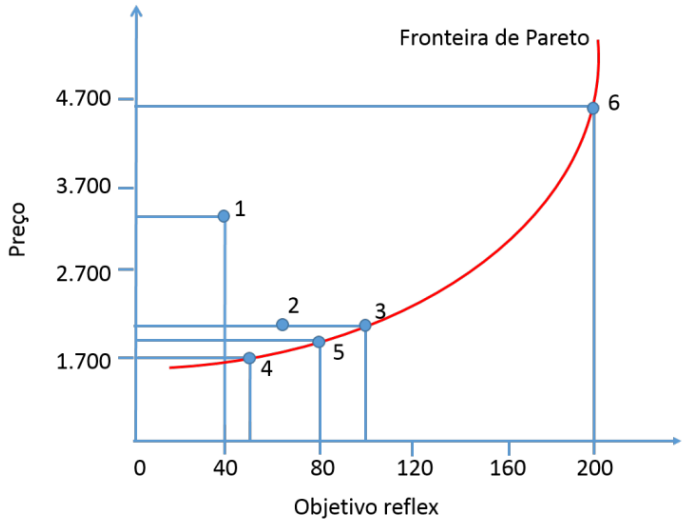

Figura 2.1: Exemplo de problema com 2 objetivos

A possibilidade de se trabalhar com várias soluções simultaneamente, de não precisar de informações adicionais e poder escapar de ótimos locais fazem dos algoritmos genéticos uma técnica promissora a ser empregada nos problemas de otimização multi-objetivo.

2.2.1 Algoritmos genéticos

Os algoritmos evolutivos são inspirados na teoria da seleção natural e evolução biológica dos seres vivos para otimização de processos classificados como complexos (Coello et al., 2007). Um tipo de algoritmo evolutivo são os algoritmos genéticos (*Genetic Algorithm* - GAs). Os GAs podem ser descritos como algoritmos inspirados na teoria da evolução,

no qual o mais adaptado sobrevive. Eles buscam a melhor solução para os problemas de otimização, utilizado um processo iterativo de busca da melhor solução para o seu problema. Onde a busca se dá a partir de uma população inicial, que combinando os melhores representantes desta população, obtém uma nova, que passa a substituir à anterior. A seleção serve para escolher um individuo (reprodução assexuada) ou um par de indivíduos (reprodução sexuada) para gerar descendência. Para definir os indivíduos da geração seguinte, as soluções com maior valor de retorno da função de *fitness* são selecionadas. Uma função de *fitness* é uma função de avaliação, que mede o quão adaptado está o indivíduo ao ambiente. Tendo selecionado os pares, novos indivíduos são gerados por recombinação e mutação (Mitchell, 1998).

Um conjunto de operações é necessário para que, dada uma população, se consiga gerar populações sucessivas que (espera-se) melhorem sua aptidão com o tempo. O princípio básico dos operadores genéticos é transformar a população através de sucessivas gerações, estendendo a busca até chegar a um resultado satisfatório. Os operadores genéticos são necessários para que a população se diversifique e mantenha características de adaptação adquiridas pelas gerações anteriores. O operador de cruzamento tem a função de combinar os cromossomos dos pais, para gerar os cromossomos dos filhos, e o operador de mutação é responsável pela inserção de pequenas mudanças aleatórias nos cromossomos dos filhos. O critério de parada dos algoritmos genéticos pode variar de acordo com a opção do usuário. Uma das formas é definir uma quantidade de gerações que devem ser geradas. Outra forma é aplicá-lo até que se encontre uma população onde os indivíduos tenham a função de avaliação que se deseja alcançar (Mitchell, 1998).

Variantes de GA adaptados para problemas multi-objetivos foram propostos na literatura. Os seguintes MOEAs (*Multi-objetive Evolutionary Algoritmos* - MOEAs) representativos, são variantes dos GA tradicionais: NSGA-II (*Non-dominated Sorting Genetic Algorithm*) (Deb et al., 2002) e PAES (*Pareto Archived Evolution Strategy*) (Knowles e Corne, 1999). Neste trabalho é utilizado NSGA-II para a realização do estudo exploratório definido no Capítulo 4.

O NSGA-II é apresentado em (Deb et al., 2002) para reduzir a complexidade computacional no classificador não-dominado e inserir o elitismo. Ele usa um mecanismo que garante maior extensão de busca no espaço, trazendo mais opções de soluções para alcançar a melhor possível naquele espaço. Assim, a maioria das boas soluções não-dominadas estão incluídas na nova população nos primeiros estágios do procedimento, portanto, as últimas soluções incluídas para preencher a população não têm grande impacto. No entanto, durante os estágios finais, a maioria das soluções podem estar na melhor fronteira não-dominada, ou o número de soluções na primeira fronteira não-dominada pode exceder

N. Neste momento, o *crowding distance* escolhe um conjunto diverso de soluções da fronteira em análise. Quando toda a população converge para a Fronteira Ótima de Pareto, o uso desse algoritmo garantirá uma melhor distribuição entre as soluções (Deb et al., 2002).

De maneira geral pode-se observar que problemas da área SBSE são influenciados por diferentes fatores e podem ser tratados como um problema de otimização multi-objetivo, mediante a definição de funções objetivo baseadas em um conjunto de métricas, que podem fornecer indicadores para avaliar a qualidade do projeto de PLA.

2.3 Abordagem de otimização multiobjetivo para projeto de PLA

Um projeto de PLA pode ser modelado como um problema de otimização multi-objetivo, porque é influenciado por diferentes fatores, tais como modularização de características, extensibilidade e reusabilidade da PLA. Esses fatores podem ser otimizados durante o processo de optimização, mas como eles podem estar em conflito, várias possibilidades de modelagem de um projeto específico de PLA podem ser encontradas. Neste contexto, uma abordagem sistemática que usa algoritmos evolutivos multi-objetivo (MOEAs) foi proposta em (Colanzi et al., 2014) com o objetivo de encontrar automaticamente as melhores alternativas de projeto para uma PLA por meio de algoritmos de busca.

MOA4PLA (*Multi-Objective Approach for Product-Line Architecture Design*) (Colanzi et al., 2014) é uma abordagem sistemática e automatizada que usa algoritmos de busca multiobjetivos para avaliar e melhorar o projeto de PLA. Esta abordagem produz um conjunto de soluções que têm o melhor *trade-off* entre os vários objetivos otimizados.

Ela usa um metamodelo, apresentado na Figura - 2.2, para representar o projeto de PLA e operadores de busca específicos para otimizar a PLA. Cada solução gerada após a aplicação dos operadores de busca é avaliada de acordo com as funções objetivo definidas no modelo de avaliação (Colanzi et al., 2014).

O metamodelo apresentado na Figura - 2.2, está composto por elementos arquiteturais que podem ser componentes, interfaces, operações de interfaces e seus inter-relacionamentos. Ademais os componentes são compostos por classes e relacionamentos interclasses e cada elemento arquitetural é associado à(s) característica(s) por meio de estereótipos. Além disso um elemento pode ser comum a todos os produtos da LPS ou variável, estando presente só em algum(ns) produto(s) da LPS e os elementos variáveis estão associados a pontos de variação e suas respectivas variantes (Colanzi et al., 2014).

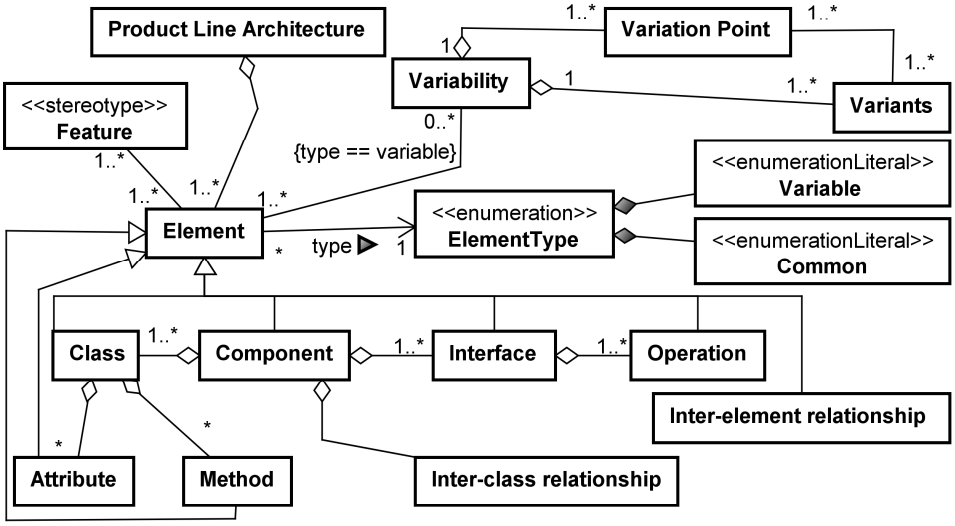

Figura 2.2: Metamodelo da MOA4PLA (Colanzi et al., 2014).

A MOA4PLA abrange quatro atividades apresentadas na Figura - 2.3 e descritas a seguir:

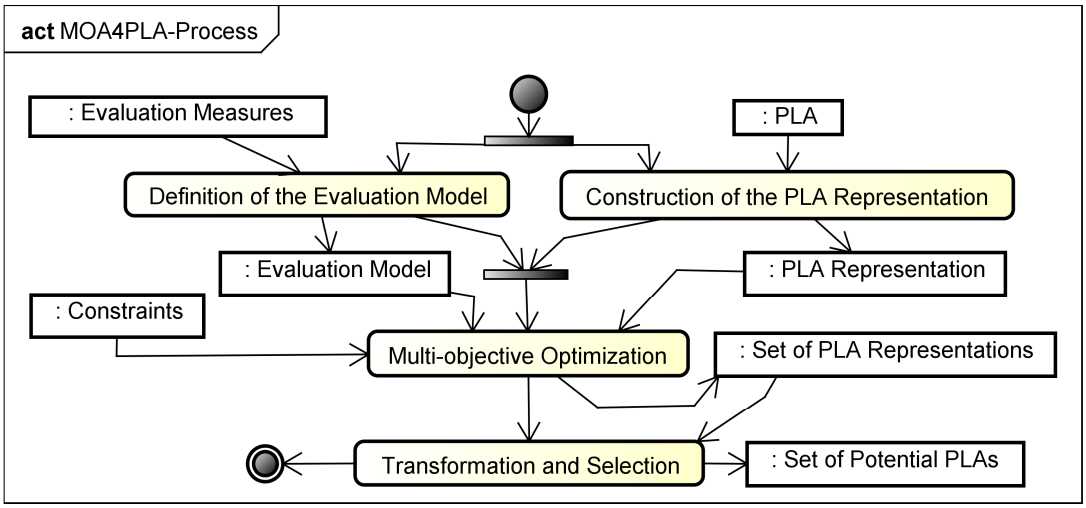

Figura 2.3: Atividades da MOA4PLA (Colanzi et al., 2014).

Construção da representação da PLA (primeira atividade da MOA4PLA): a entrada para esta atividade é o projeto PLA, modelado em um diagrama de classe (estereotipados) UML contendo as variabilidades da LPS incluindo os pontos de variação e seus variantes. Ademais, nesse diagrama, as características devem estar associadas aos elementos arquiteturais utilizando estereótipos, o que permite que os operadores de busca sejam aplicados adequadamente.

Assim, a saída desta atividade é a representação da PLA de acordo com o metamodelo definido na MOA4PLA que contém todos os elementos arquiteturais, seus inter-relacionamentos, todos os pontos de variação, variantes e características associadas aos elementos arquiteturais. Cada elemento é associado com as características por meio de estereótipos UML e pode ser comum a todos os produtos da LPS.

Definição do modelo de avaliação (segunda atividade da MOA4PLA): De acordo com as necessidades da LPS, o arquiteto deve escolher as métricas que devem ser incluídas no modelo de avaliação, que são usadas no processo de otimização para avaliar cada solução obtida (potencial projeto de PLA). Assim, as métricas constituem os objetivos a serem otimizados pela abordagem. É importante ressaltar que o uso de diferentes métricas apoia o arquiteto na análise de *trade-off* entre os diferentes atributos de qualidade e que algumas, dessas métricas podem ser conflitantes. Detalhes sobre o modelo de avaliação de MOA4PLA são apresentados na Seção 2.3.1.

Otimização multi-objetivo (terceira atividade da MOA4PLA): A representação da PLA obtida na primeira atividade é otimizada considerando as restrições fornecidas pelo arquiteto. Cada projeto de PLA obtido é avaliado seguindo o modelo de avaliação definido na atividade anterior. Um conjunto de representações de PLA é gerado como saída.

As restrições originais incluídas na MOA4PLA são as seguintes: (i) componentes, interfaces e classes não devem estar vazias; (ii) classes e interfaces devem estar envolvidas em pelo menos um relacionamento; (iii) um método abstrato não deve ser movido para outra classe se está em uma hierarquia de herança, pois pode haver métodos concretos que os implementam em subclasses; (iv) relacionamentos de generalização entre as classes e as subclasses devem ser mantidos em todas as soluções geradas; (v) os elementos arquiteturais que sofrem a ação de operadores de busca devem permanecer associados com as características originais, embora eles tenham sido movidos para um elemento arquitetural associado com uma característica diferente; e (vi) atributos e métodos de classes, que são pontos de variação ou variações associadas com a característica da variabilidade associada não devem ser movidos.

MOA4PLA inclui operadores de mutação convencionais: MoveMethod, MoveAttribute, AddClass, MoveOperation e AddComponent. AddClass move-se um método ou um atributo para uma nova classe. MoveOperation movem-se as operações entre interfaces. AddComponent cria uma nova interface para um novo componente e em seguida, move uma operação para esta interface.

Essa atividade engloba também operadores de mutação e *crossover*, onde o operador de mutação faz mudanças aleatórias nas características dos cromossomos, com a intenção

de obter maior diversidade genética na população; e o operador *crossover* combina as informações genéticas de dois indivíduos (pais) para gerar novos indivíduos (filhos). Neste trabalho são utilizados os operadores de mutação para a realização do estudo exploratório definido no Capítulo 4.

Transformação e seleção (quarta atividade da MOA4PLA): O conjunto de soluções obtidas na terceira atividade da MOA4PLA é convertido para uma forma legível para o arquiteto: um diagrama de classe que contém o projeto da PLA. Assim, o arquiteto deve selecionar uma alternativa que prioriza algum(ns) objetivo(s), para ser adotado como projeto da PLA, de acordo com as prioridades da LPS.

2.3.1 Funções objetivo do modelo de avaliação

O modelo de avaliação da MOA4PLA, inicialmente foi composto por quatro funções objetivo: CM, FM, Ext, $Eleg$ (Colanzi et al., 2014), as quais utilizam métricas de software para medir: princípios básicos de projeto, modularização de características, extensibilidade e elegância do projeto de PLA, respectivamente. A seguir apresenta-se uma breve descrição das funções objetivo:

1. **CM(pla)**: Tem como objetivo fornecer indicadores sobre princípios básicos de projeto incluindo coesão, acoplamento e tamanho. CM(pla) é formada por uma agregação de várias métricas (DepIn, DepOut, CDepIn, CDepOut, DepPack, NumOps, H) extraídas de (Wüst, 2016). Na Equação 2.1 pode ser vista sua definição, onde c representa o número de componentes, itf representa o número de interfaces e cl representa o número de classes de uma dada PLA.

$$CM(pla) = \sum_{i=1}^{c} DepIn + \sum_{i=1}^{c} DepOut + \sum_{i=1}^{cl} CDepIn + \sum_{i=1}^{cl} CDepOut + \frac{\sum_{i=1}^{c} DepPack}{c} + \frac{\sum_{i=1}^{itf} NumOps}{itf} + \frac{1}{\sum_{i=1}^{c} H} \quad (2.1)$$

2. **FM(pla)**: Tem como objetivo medir a modularização de características e, para isso, é composta por um conjunto de várias métricas dirigidas às características (LCC, CDAC, CDAI, CDAO, CIBC, IIBC, OOBC) propostas em (Sant'Anna, 2008). A seguir a Equação 2.2 proposta por (Colanzi et al., 2014), define a FM(pla), onde i representa o número de interfaces e f representa o número de características de uma dada PLA.

$$FM(pla) = \sum_{i=1}^{c} LCC + \sum_{i=1}^{f} CDAC + \sum_{i=1}^{f} CDAI + \sum_{i=1}^{f} CDAO + \sum_{i=1}^{f} CIBC + \sum_{i=1}^{f} IIBC + \sum_{i=1}^{f} OOBC \quad (2.2)$$

3. **Ext(pla)**: Tem como alvo indicar o grau de extensibilidade da LPS. A Equação 2.3 apresenta o valor invertido da métrica ExtensPLA, onde ExtensPLA é uma agregação das métricas propostas em (OliveiraJr et al., 2013). Ext(pla) é calculada dessa maneira com a intenção de maximizar a extensibilidade da PLA, uma vez que os objetivos são minimizados na MOA4PLA.

$$Ext(pla) = \frac{1}{ExtensPLA} \quad (2.3)$$

4. **Eleg(pla)**: Tem como objetivo fornecer indicadores da elegância do projeto orientado a objetos. Eleg(pla) definida na Equação 2.4 é composta pelo somatório dos valores das métricas (NAC, EC, ATMR) propostas em (Simons e Parmee, 2012).

$$Eleg(pla) = NAC(pla) + EC(pla) + ATMR(pla) \quad (2.4)$$

No entanto, em um estudo realizado por (Guizzo et al., 2014) foi observado que CM e FM envolvem métricas (Tabela - 2.1) referentes a diferentes propriedades arquiteturais, o que prejudica a sensibilidade das funções objetivo em avaliar o projeto de PLA, já que uma métrica com maior grandeza exerce maior influência no valor final da função. Portanto os resultados desse estudo indicaram que o modelo de avaliação de MOA4PLA precisava ser melhorado a fim de obter funções objetivo mais sensíveis às mudanças realizadas no projeto PLA durante o processo de avaliação.

Nesse sentido, Santos et al. (2015) refinaram o modelo originalmente proposto em (Colanzi et al., 2014), mediante a proposta de sete novas funções ($ACLASS$, $ACOMP$, TAM, COE, DC, EC e LCC) as quais foram acrescentadas ao modelo de avaliação original da MOA4PLA e são descritas a seguir:

1. **ACLASS(pla)**: Tem como objetivo medir o número de elementos de entrada e saída em que uma classe depende de outra. ACLASS(pla) é definida mediante a Equação 2.5 na qual é composta pelo somatório das métricas que medem as

Tabela 2.1: Métricas do modelo de avaliação atual

Propriedades Arquiteturais	**Métricas**	**Descrição**
Feature Diffusion	CDAC - *Concern Diffusion over Architectural Components*	Mede a modularização de características em nível de pacotes (Sant'Anna, 2008).
	CDAI -*Concern Diffusion over Architectural Interfaces*	Mede número de interfaces que contribuem para a realização de um dado interesse (Sant'Anna, 2008).
	CDAO - *Concern Diffusion over Architectural Operations*	Mede o número de operações que contribuem para a realização de um dado interesse (Sant'Anna, 2008).
Feature Interaction	CIBC - *Component-level Interlacing Between Concerns*	Mede o número de características com os quais um dado interesse compartilha ao menos um pacote (Sant'Anna, 2008).
	IIBC - *Interface-level Interlacing Between Concerns*	Mede o número de características com os quais um dado interesse compartilha ao menos uma interface (Sant'Anna, 2008).
	OOBC - *Operation-level Overlapping Between Concerns*	Mede o número de características com os quais um dado interesse compartilha ao menos uma operação (Sant'Anna, 2008).
Feature-based Cohesion	LCC - *Lack of Concern*	Mede o número de características com os quais um dado pacote está associado (Sant'Anna, 2008).
Cohesion	H - *Relational Cohesion*	Mede a coesão relacional do projeto contando o número médio de relacionamentos entre classes e interfaces dentro de um pacote (Wüst, 2014).
Coupling	DepPack - *Dependency of Packages*	Mede a dependência de pacotes entre classes e interfaces (Wüst, 2014).
	DepIn - *ClassDependencyIn*	Mede o número de dependências UML onde o pacote é o fornecedor (Wüst, 2014).
	DepOut - *ClassDependencyOut*	Mede o número de dependências UML onde o pacote é o cliente (Wüst, 2014).
	CDepIn - *DependencyIn*	Mede o número de elementos arquiteturais que dependem dessa classe (Wüst, 2014).
	CDepOut - *DependencyOut*	Mede o número de elementos arquiteturais dos quais essa classe depende (Wüst, 2014).
Extensibility	Ext - *ExtensPLA*	Mede a capacidade de extensão de cada um dos pacotes da PLA (OliveiraJr et al., 2013)
Size	NumOps - *Number of Operations by Interface*	Mede o número de operação por interface (Wüst, 2014)
Elegance	NAC - *Numbers Among Classes*	Mede a elegância de números entre classes. Desvio padrão dos números de atributos e métodos entre as classes de um projeto (Simons e Parmee, 2012).
	EC - *External Couples*	Mede a elegância de acoplamentos externos das classes de um projeto (Simons e Parmee, 2012).
	ATMR - *Attributes To Methods Ratio*	Desvio padrão da razão entre atributos e métodos dentro das classes de um projeto (Simons e Parmee, 2012).

dependências de entrada DepIn e de saída DepOut e c representa o número de pacotes da PLA.

$$ACLASS(pla) = \sum_{i=1}^{c} CDepIn + \sum_{i=1}^{c} CDepOut \tag{2.5}$$

2. **ACOMP(pla)**: Tem como objetivo medir o acoplamento entre os pacotes de uma arquitetura. ACOMP(pla) é composta pelo somatório das métricas de dependência de entrada DepIn e saída DepOut de um pacote e é definida mediante Equação 2.6, onde c representa o número de componentes da PLA.

$$ACOMP(pla) = \sum_{i=1}^{c} DepIn + \sum_{i=1}^{c} DepOut \tag{2.6}$$

3. **TAM(pla)**: Tem como objetivo medir o número de operações de uma interface. TAM(pla) é definida pela Equação 2.7 a qual é composta pela métrica NumOps e *itf* representa o número de interfaces.
$$TAM(pla) = \frac{\sum_{i=1}^{itf} NumOps}{itf} \tag{2.7}$$

4. **COE(pla)**: Tem como objetivo medir a coesão relacional entre as classes. COE(pla) é representada pela Equação 2.8.
$$COE(pla) = H \tag{2.8}$$

5. **DC(pla)**: Tem como objetivo medir a difusão de caraterísticas, somando os números de elementos arquiteturais (componentes, interfaces e operações do projeto) que contribui para a realização das características da LPS. *DC* função é definida na Equação 2.9 onde i representa o número de interfaces e f o número de características de uma dada PLA.
$$DC(pla) = \sum_{i=1}^{f} CDAI + \sum_{i=1}^{f} CDAO + \sum_{i=1}^{f} CDAC \tag{2.9}$$

6. **EC(pla)**: Tem como objetivo medir o entrelaçamento de características em uma PLA, somando os números das características com o qual um interesse compartilha pelo menos com um elemento arquitetônico, como componente, interface e operação, a Equação 2.10 define *EC*.
$$EC(pla) = \sum_{i=1}^{f} CIBC + \sum_{i=1}^{f} IIBC + \sum_{i=1}^{f} OOBC \tag{2.10}$$

7. **LCC(pla)**: Tem como objetivo medir a coesão baseada em características por meio do número de características com as quais um dado pacote está associado. *EC* é definida pela Equação 2.10.
$$LCC(pla) = LCC \tag{2.11}$$

2.3.2 OPLA-Tool

OPLA-Tool (*Optimization for PLA Tool*) (Féderle et al., 2015) é a ferramenta que automatiza a MOA4PLA. Esta ferramenta fornece suporte automatizado para todas as atividades da MOA4PLA. OPLA-Tool está composta por seis módulos apresentados na Figura - 2.4 que permitem: (i) a codificação e decodificação do projeto da PLA em um

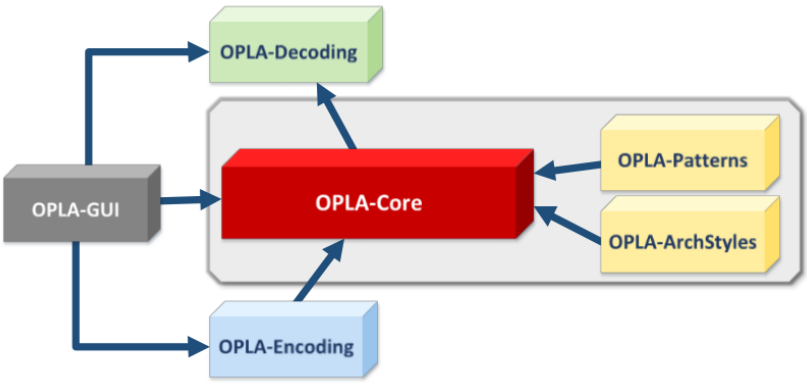

Figura 2.4: Módulo da OPLA-Tool (Colanzi et al., 2014)

formato utilizável pelos algoritmos de pesquisa (OPLA-Encoding e OPLA-Decoding), (ii) o uso de operadores de busca para aplicar padrões de projeto e estilos arquitetônicos nos projetos de PLAs (OPLA-Patterns e OPLA-ArchStyles), (iii) a comunicação com o usuário da ferramenta (OPLA-GUI) e, (iv) a otimização do projeto por meio de algoritmos de busca e funções objetivo composto por métricas para projeto de PLA (OPLA-Core). Na Figura - 2.4 os módulos foram coloridos de forma a relacioná-los com as atividades da MOA4PLA (Figura - 2.3) que cada um deles apóia. A seguir são descritos brevemente cada módulo:

- OPLA-GUI: Uma interface gráfica com o usuário, permitindo a interação com a abordagem MOA4PLA. Este módulo permite que o arquiteto informe qual PLA deseja fornecer como entrada para a abordagem, qual MOEA, quais as métricas e operadores de busca se deseja utilizar. Também permite a visualização dos resultados obtidos e a execução dos experimentos;

- OPLA-Encoding: Converte uma PLA em uma representação que segue um meta-modelo e é passível de ser utilizada pelos algoritmos de busca;

- OPLA-Core: Realiza a otimização com algoritmos de busca retornando um conjunto de PLAs. É aqui que os MOEAs são aplicados usando os operadores de busca da MOA4PLA para otimizar um projeto de PLA fornecido como entrada. Dois MOEAs são implementados NSGA-II e PAES;

- OPLA-Decoding: Traduz o conjunto de PLAs retornadas pelo módulo OPLA-Core em um formato que seja legível para o arquiteto, tanto no formato textual quanto no formato gráfico (diagrama de classes);

- OPLA-Patterns: Contém operadores de mutação para a aplicação de padrões de projeto; e
- OPLA-ArchStyles: Contém operadores para a aplicação de estilos arquiteturais.

2.4 Modelos de qualidade de software

Os modelos de qualidade são definidos para apoiar o aumento no nível de qualidade do gerenciamento, aquisição e manutenção de software (Dromey, 1995). Segundo Kitchenham et al. (1995) na etapa de desenvolvimento, os modelos de qualidade servem como base para a modelagem e implementação. Ademais promovem recomendações diretas sobre a modelagem do projeto e elaboram abordagens construtivas para alcançar o alto nível de qualidade (Dromey, 1995).

Os modelos de qualidade descrevem as características de qualidade através de conjuntos de atributos, que são definidos com base no domínio especifico, onde os fatores de qualidade podem ser classificados como: (i) atributos internos que são referentes ao nível de projeto e desenvolvimento, e são obtidos sem a execução do software (ISO/IEC, 2001), como por exemplo: coesão e acoplamento entre os elementos; complexidade (número de pontos de decisão do código ou do modelo); tamanho do projeto (número de linhas de código ou número de classes do modelo); ou, a dependência entre os módulos do software; e (ii) atributos externos de qualidade, que somente podem ser obtidos com a execução do software (ISO/IEC, 2001).

Segundo Meyer (1988), são os atributos internos (métricas internas) que asseguram a qualidade dos atributos externos, por meio de técnicas de avaliação aplicadas aos atributos internos, no entanto os atributos externos têm importância para o produto final. Por conseguinte, os atributos internos e externos estão estritamente relacionados. Por exemplo, os atributos externos confiabilidade, manutenibilidade, usabilidade, integridade, dentre outros, estão ligados com métricas internas que estão associadas com o código, documentação e outros artefatos que dão apoio ao desenvolvimento (Souza, 2015).

Consequentemente, existem alguns modelos de qualidade encontrados na literatura que influenciaram na definição de modelos específicos (Souza, 2015) como por exemplo a norma ISO/IEC 9126 e a norma ISO/IEC 25010.

A norma ISO/IEC 9126 (ISO/IEC, 2001) é um padrão internacional, desenvolvido pelo Subcomitê de Software (SC7) do Comitê Técnico Conjunto (JTC1) da ISO e IEC. A ISO/IEC 9126 divide-se em quatro relatórios técnicos, no qual cada um assume responsabilidades sobre a avaliação do produto de software. Na Figura - 2.5 mostra-se

como os atributos desse modelo de qualidade de produto de software são classificados numa estrutura hierárquica, tendo em conta características e subcaracterísticas. A aplicação da norma ISO/IEC 9126-1 permite que a avaliação da qualidade do produto de software seja especificada e avaliada pelos envolvidos com aquisição, requisitos, projeto, desenvolvimento, uso, avaliação, apoio, manutenção, garantia de qualidade e auditoria de software (Souza, 2015).

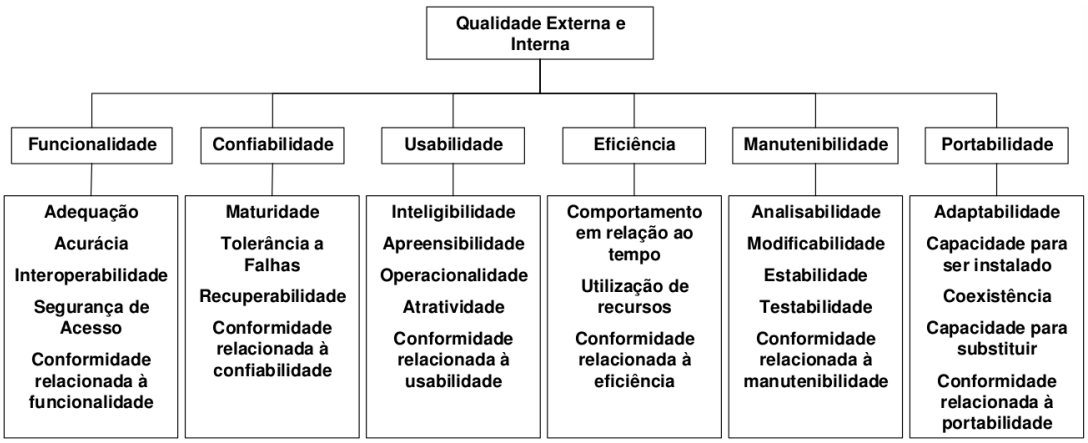

Figura 2.5: Modelo de Qualidade ISO/IEC 9126 - adaptado de (ISO/IEC, 2001)

A norma ISO/IEC 25010, trata do modelo de qualidade para produtos de software e é uma reestruturação da norma ISO/IEC - 9126. O modelo da norma está baseado em um conjunto de atributos de qualidade que um produto de software deve apresentar, o qual também é organizado hierarquicamente em subatributos (ISO/IEC, 2011). A Figura - 2.6 apresenta os oito atributos (que são subdivididos em subatributos) do modelo de qualidade, onde atributos de qualidade relacionam se com propriedades tanto estáticas quanto dinâmicas do software (ISO/IEC, 2011).

Anteriormente os modelos de qualidade expostos são propostos para atender aos critérios de qualidade de âmbito geral, mas na literatura existe um amplo conjunto de modelos de qualidade para aplicação de avaliações específicas, por exemplo: modelos de qualidade para componentes de software (Sharma et al., 2008), modelos de qualidade para projetos OO (Bansiya e Davis, 2002) e modelos de qualidade para Sistemas Embarcados (Ahrens et al., 2013),(Hu et al., 2012), (Jeong e Kim, 2012), (Carvalho et al., 2009). Este tipo de modelos de qualidade adaptados ou elaborados com base nos modelos apresentados, são propostos de acordo com as necessidades dos domínios e linguagem a serem aplicadas na avaliação (Souza, 2015).

No contexto de Sistemas Embarcados, pode-se citar o trabalho realizado por Ahrens et al. (2013), que contempla um conjunto de sete características de qualidade, cada uma

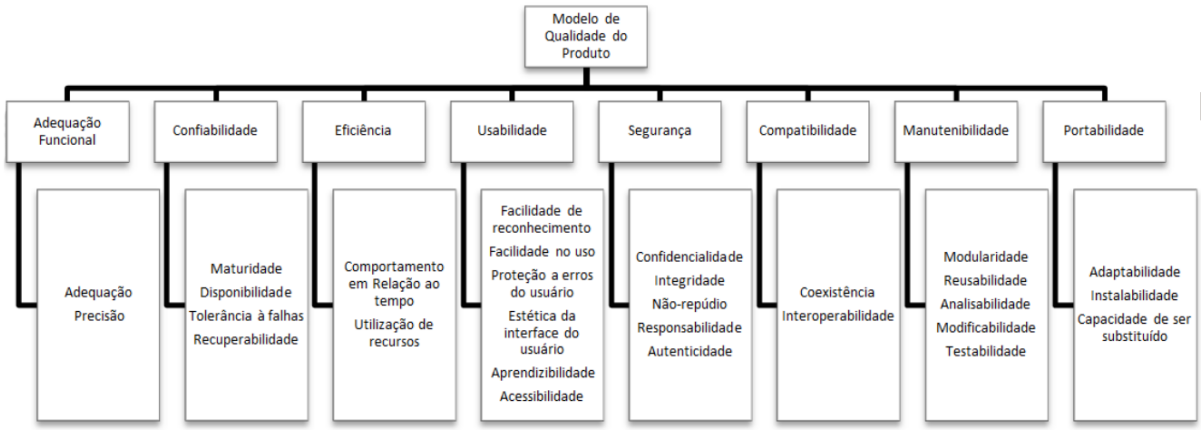

Figura 2.6: Modelo de Qualidade ISO/IEC 25010 - adaptado de (ISO/IEC-25010, 2005)

contendo vários atributos de qualidade, exceto a característica **Conformidade**, onde os atributos definidos são considerados atributos atômicos, podendo ser diretamente associados a métricas internas. Na Figura - 2.7 apresentam-se os (sub)atributos do modelo de qualidade estabelecido por Ahrens et al. (2013), onde a **Reusabilidade** é definida como capacidade de usar certas partes do sistema em mais de um caso. Esta caraterística está associada com o atributo de qualidade **Modularidade** o qual define como o sistema é decomposto em componentes independentes e auto-contidos. Além disso, a **Mutabilidade** refere-se a como um sistema de software mostra a possibilidade de ser alterado durante o ciclo de vida e esta caraterística está associada com os atributos de qualidade **Compreensibilidade, Extensibilidade e Modularidade**; onde a Compreensibilidade descreve a capacidade de um sistema de software para tornar a sua estrutura clara para os *stakeholders*, e a Extensibilidade define-se como a possibilidade de incluir requisitos suplementares após a implementação inicial, como por exemplo, a adição de novas funcionalidades, a adição de *cross-linkings* para sistemas externos bem como variantes específicas para o projeto.

Como os modelos de qualidade estão relacionados com qualidade de software, as métricas estão incluídas em modelos de qualidade para a medição e associação com os atributos de qualidade. Uma métrica procura relacionar medidas individuais com o objetivo de se ter uma ideia da eficácia do processo, do projeto ou do produto sendo medido (Pressman, 2011). Segundo Rocha et al. (2001), o nível de qualidade de um produto ou de um processo de software pode ser obtido por meio da utilização de métricas, que auxiliam na organização, no monitoramento, na identificação e na prevenção de falhas.

Na área de desenvolvimento de software, um projeto é uma atividade que envolve trabalho profundo para que os critérios de qualidade estabelecidos satisfaçam aos requisitos

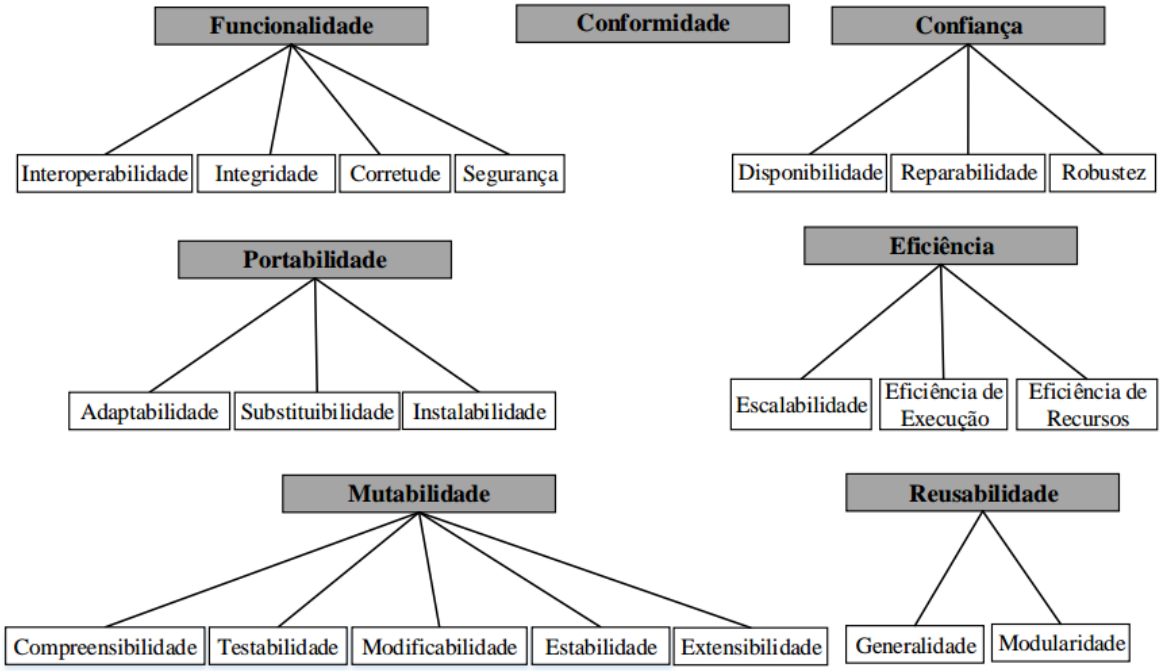

Figura 2.7: Modelo de Qualidade definido por Ahrens et al. (2013)

dos clientes ou usuários. Nesse sentido, as métricas podem auxiliar na avaliação tanto dos processos quanto dos produtos finais (Rocha et al., 2001). Portanto, pode-se dizer, que as métricas servem para administração e controle do processo de desenvolvimento para determinar os resultados obtidos, onde a medição de atributos de produto é essencial para o aprimoramento do software (Humphrey e Kellner, 1989).

Mas a dificuldade fundamental da medição é justamente saber o que medir. Medir o impacto da integração de *features* para ter controle sobre a variação da qualidade interna do sistema ao adicionar ou retirar características é necessário na reutilização de artefatos (Fenton e Bieman, 2014). Esta medição pode ser realizada utilizando métricas de produto (Chidamber e Kemerer, 1994), já que as métricas podem apoiar a definição e avaliação de uma arquitetura de software. Nessa perspectiva, no próxima seção aborda trabalhos relacionados que se relacionam ou contexto.

2.5 Trabalhos relacionados

Recentemente foi realizado um mapeamento sistemático com o objetivo de enriquecer o modelo de avaliação da MOA4PLA (Delgado et al., 2016). A questão de pesquisa desse mapeamento foi: *Qual é o estado da arte das métricas para Linha de Produto*

de Software?. Para responder essa questão foi executada uma busca automática nas fontes de pesquisa: IEEExplore Digital Library, ACM Digital Library, Elsevier Scopus e SpringerLink. Como resultado dessa busca, foram obtidos 588 trabalhos, onde na primeira seleção foram excluídos 566 trabalhos, restando 22. Após da segunda seleção, aplicando os critérios de inclusão e exclusão definidos em (Delgado et al., 2016), o número de trabalhos foi reduzido para 20 e ao realizar a leitura integral deles, centrada na busca de métricas existentes para PLA, foram excluídos 15 trabalhos; restando ao final um total de 5 trabalhos brevemente descritos a seguir.

De acordo com (Sant'Anna et al., 2003), medir atributos particulares internos, tais como coesão, é útil se está relacionado a uma medida de alguns atributos externos do objeto de estudo, por exemplo, reutilização ou manutenção. Nesta perspectiva, (Ribeiro et al., 2010) desenvolvem um *framework* de medição para capturar o entendimento do tamanho, acoplamento, coesão, instabilidade e separação de interesses em termos dos atributos de qualidade de manutenção e reutilização. O *framework* é composto por seis métricas de código agrupadas de acordo com os atributos que medem, por exemplo: tamanho, acoplamento, coesão, instabilidade e separação de interesses, onde a descrição de cada métrica enfatiza como ele satisfaz os requisitos de medição definidos.

Zhang et al. (2008) reafirmam que é importante que a PLA seja medida na engenharia de domínio, por que ajuda a capturar aspectos importantes da PLA anteriormente na fase de concepção do domínio, para que possa tomar ações corretivas mais cedo e poder impedir a propagação de erros, em componentes e arquiteturas de produtos da LPS e oferecer o maior potencial de retorno sobre investimento. Nesse trabalho eles abordam o estudo de métricas para PLA tem recebido pouca atenção apesar de a arquitetura de linha de produto definir os conceitos necessários para atingir a variação em características dos produtos variantes e alcançar o máximo compartilhamento de artefatos na implementação. Nesse sentido eles propõem métricas para medir a similaridade, variabilidade, reusabilidade e complexidade da arquitetura de linha de produto em vista de ajudar a analisar e promover a qualidade da PLA.

O trabalho de Her et al. (2007) referencia que reutilização do núcleo de artefatos determina em grande parte o sucesso da LPS. Nesse sentido eles propõem um *framework* para avaliar a capacidade de reutilização da PLA onde primeiro identificam as principais características do núcleo de artefatos e derivam um conjunto de atributos de qualidade que caracteriza a capacidade de reutilização de núcleo de artefatos, e logo são definidas as métricas para cada atributo de qualidade.

O trabalho de Aldekoa et al. (2008) refere-se sobre o fato de quantificar a capacidade de manutenção de software, tendo em conta que a manutenibilidade é a facilidade com que um

sistema pode ser modificado para corrigir falhas, melhorar o desempenho ou se adaptar a um ambiente em mudança. A capacidade de manutenção do programa é calculada através do índice de manutenção, que usa uma combinação de medidas amplamente utilizadas e comumente disponíveis. Nesse sentido Aldekoa et al. (2008) propõem o uso do índice de manutenção para tomar decisões de projeto que podem melhorar a manutenção global da LPS.

Em outra iniciativa no contexto da avaliação de LPS, Torres et al. (2010) comparam produtos de diferentes abordagens. Essa comparação foi realizada através de uma análise quantitativa onde foram medidos os atributos modularidade, complexidade e estabilidade ao longo da derivação de artefatos produzidos. Nesse trabalho foram utilizadas diferentes métricas para quantificar esses atributos em termos de difusão, entrelaçamento, número de *tokens* e quantidade de expressões de sentença CK (*Configuration Knowledge*) e número de expressões de CK adicionados/removidos/alterados.

A continuação, na Tabela - 2.2 mostra-se as métricas encontradas nos trabalhos mencionados anteriormente, onde é apresentado de cada trabalho, as métricas utilizadas e as propriedades arquiteturais a serem medidos por cada métrica. Na quarta coluna, especificam-se quais métricas podem ser ou não utilizadas no modelo de avaliação da MOA4PLA, tendo em conta os seguentes critérios de seleção: (i) que seja possível obter as informações necessárias para aplicar as métricas em um diagrama de classes UML, dado que o modelo de avaliação baseia-se em um metamodelo que instancia um diagrama de classes UML; e (ii) as métricas não estejam implementadas no modelo de avaliação atual. Por ultimo, na coluna Observações argumenta-se brevemente somente as métricas não podem ser utilizadas no modelo.

Tabela 2.2: Métricas encontradas nos trabalhos relacionados

Trabalho	Métrica	Propriedade Arquitetural	Pode ser utilizadas?	Observações
Ribeiro et al. (2010)	Weighted Operations per Component or Service (WOCS)	Tamanho	Sim	
	Cyclomatic Complexity (CC)	Tamanho	Não	Métrica orientada a Lines of Code (LOC)
	Concern Diffusion over Components or Services (CDCS)	Separação de interesses	Não	Métrica jé se encontra implementada na MOA4PLA
	Instability Metric for Service or Component (IMSC)	Instabilidade	Não	O cálculo desta métrica precisa de diagramas de comportamento
	Coupling Between Components or Services (CBCS)	Acoplamento	Sim	
	Lack of Cohesion over Operations (LCOO)	Coesão	Não	O cálculo desta métrica precisa do nome dos atributos e não é possível garantir que o nome dos atributos seja igual em todas as classes da PLA
Zhang et al. (2008)	Structure Similarity Coefficient (SSC)	Semelhança	Sim	
	Strong Coupling Coefficient (SCC)	Acoplamento	Sim	
	Weak Coupling Coefficient (WCC)	Acoplamento	Não	Não é possível medir as condições de guarda existentes na PLA
	Structure Variability Coefficient (SVC)	Variabilidade	Sim	
	Architecture variability (AV)	Variabilidade	Sim	
	Component reuse rate (CRR)	Reusabilidade	Não	Não se tem produtos gerados pela LPS
	Reuse benefit rate (RBR)	Reusabilidade	Não	O cálculo desta métrica precisa de LOC
	Interior information flow complexity (IIFC)	Complexidade	Não	Metrica para vADL (Product Line Architecture Description Languages)
	Exterior information flow complexity (EIFC)	Complexidade	Não	Metrica para vADL
	PLA-IFG vertex complexity (PVC)	Complexidade	Não	Métrica para vADL
	PLA-IFG information flow complexity (PIFC)	Complexidade	Não	Metrica para vADL
	PLA-IFG total complexity (PTC)	Complexidade	Não	Métrica para vADL
	PLA-IFG cyclomatic complexity (PCC)	Complexidade	Não	Métrica para vADL
Her et al. (2007)	Overall Understandability (OU)	Compreensibilidade	Não	Métrica precisa da descrição do núcleo de artefatos (onde a descrição do núcleo de artefatos inclui a especificação, manual do usuário e o documento que descreve o mesmo)
	Component Compliance (CC)	Subtilidade	Não	O cálculo desta métrica precisa do número de componentes substituíveis
	Functional Coverage (FC)	Semelhança funcional	Não	O cálculo desta métrica requer conhecer o número de aplicativos que usam o recurso funcional
	Non-Functional Commonality (NFC)	Semelhança Não-funcional	Não	O cálculo desta métrica requer conhecer o número de aplicativos que usam o recurso funcional
	Cumulative Applicability (CA)	Aplicabilidade	Não	Tem dependencia com as métricas FC e NFC
	Cover age off Variability (CV)	Riqueza de variabilidade	Não	Não se tem produtos gerados pela LPS
	Tailorability (TL)	Tailorability	Não	O cálculo desta métrica precisa do uso de listas de verificação para decidir se cada ponto de variação é usado
Aldekoa et al. (2008)	Maintainability Index (MI)	Maintainability	Não	O cálculo da métrica precisa de LOC
Torres et al. (2010)	Stability metric	Estabilidade	Não	Métrica é calculada em termos de sentença CK (configuração de conhecimento)
	Number of tokens in CK specificatio	Complexity	Não	Métrica é de CK
	Number of CK sentence expressions	Complexity	Não	Métrica é de CK
	Modularity metrics	Modularidade	Não	Métrica é de CK

Os trabalhos mencionados anteriormente, em geral abordam a avaliação quantitativa de LPS e utilizam métricas a fim de avaliar atributos de qualidade de projeto de PLA. Como resultado do mapeamento sistemático foram selecionadas várias métricas a serem utilizadas no modelo de avaliação da MOA4PLA, que serão descritas detalhadamente no próximo capítulo. A partir dos resultados do mapeamento sistemático foi realizado uma prova de conceito para averiguar a possibilidade de uso das métricas selecionadas no modelo de avaliação. Isso resultou em uma publicação (Delgado et al., 2016) a qual encontra-se no Apêndice B.

2.6 Considerações finais

De maneira geral a avaliação e melhoria automática de arquiteturas foram resolvidos com sucesso no campo da SBSE (Harman et al., 2014; Räihä, 2010), sendo a otimização do projeto de PLA modelada como um problema multi-objetivo para ser resolvido por técnicas baseadas em busca, que fornecem apoio automatizado para avaliar alternativas de PLA, facilitando as decisões do arquiteto (Colanzi et al., 2014). Neste capítulo, apresentou-se uma conceituação sobre a otimização do projeto de PLA, a MOA4PLA, modelos de qualidades existentes e mencionaram-se trabalhos relacionados a métricas para a PLA.

Apesar de o modelo de avaliação proposto por Colanzi et al. (2014) ter sido refinado por Santos et al. (2015), é importante notar que existem outras propriedades arquiteturais a ter em conta no momento de otimizar projeto da PLA, que ainda não são avaliadas no modelo de avaliação da MOA4PLA. Nesse sentido, uma contribuição do presente capítulo foi a identificação de métricas para la LPS descobertas no mapeamento sistemático realizado. Em geral o estudo dos modelos de qualidade e das métricas encontradas deu subsídio para extensão do modelo de avaliação proposto no próximo capítulo.

3

Extensão do Modelo de Avaliação de Projeto da PLA

A qualidade da PLA pode ser refletida por meio de uma série de atributos de qualidade, que geralmente são agregados em um modelo de qualidade. Como foi mencionado anteriormente; o modelo de avaliação original baseia-se em funções objetivo que avaliam princípios básicos de projeto, modularização de características, elegância e extensibilidade da LPS. Dado que o objetivo do presente trabalho é a extensão do modelo de avaliação de projeto de PLA da MOA4PLA, neste capítulo abordam-se as atividades inerentes a esta extensão.

Nesse sentido o presente capítulo está estruturado da seguinte forma: na Seção 3.1 é proposto o modelo de qualidade para o modelo de avaliação da MOA4PLA. Na Seção 3.2 são apresentadas as novas funções objetivo propostas para o modelo de avaliação e na Seção 3.3 explicam-se os aspectos de implementação das novas funções objetivo na OPLA-Tool.

3.1 Modelo de qualidade

Segundo Dromey (1995) para possibilitar a avaliação quantitativa de níveis de qualidade para um determinado artefato de software tendo como base, métricas; é necessário definir um modelo de qualidade factível para um determinado domínio e associar seus atributos de qualidade com métricas computáveis.

A definição de um modelo capaz de capturar a qualidade do projeto de PLA para *Search-based PLA Design* (SBPD) ainda é uma questão de investigação aberta. Como a

definição do modelo de avaliação para SBPD abrange a construção de funções objetivo baseadas em um conjunto de métricas, é necessário desenvolver um modelo de qualidade baseado em características de qualidade que contenham atributos de qualidade que são importantes para projeto de PLA e que inclua propriedades arquiteturais medidas por métricas. Então, a definição do modelo de qualidade, com base em métricas existentes, precede a definição do modelo de avaliação. Desde essa perspectiva, é necessário definir um modelo de qualidade que abrange as características de qualidade que se deseja avaliar, tendo em conta os atributos de qualidade e as propriedades arquiteturais que são avaliadas pelas funções objetivo do modelo de avaliação atual.

Portanto, foi realizado um estudo sobre os modelos de qualidade, apresentados no Capítulo 2, que visam a apoiar o desenvolvimento de software no âmbito geral, especificamente para garantir a qualidade dos produtos de software. Dos modelos citados, o modelo de Ahrens et al. (2013) definido para o contexto de Sistemas Embarcados, é mais abrangente que a norma ISO/IEC 9126 e ISO/IEC 25010 (mencionadas no Capítulo 2), pois tem em conta: (i) as características de qualidade ***Reusabilidade*** e ***Mutabilidade***, as quais são muito importantes no contexto da LPS; devido à necessidade de reúso de artefatos e da evolução da mesma e (ii) que o modelo especifica cada subatributo para as necessidades individuais da avaliação da arquitetura.

Assim, para a definição do modelo de qualidade da MOA4PLA, optou-se por seguir o modelo de qualidade de Ahrens et al. (2013), considerando especificamente as características de qualidade ***Reusabilidade*** e ***Mutabilidade*** e exclusivamente os atributos de qualidade ***Modularidade, Modificabilidade, Extensibilidade e Compreensibilidade***, levando em consideração que as funções objetivo do modelo de avaliação original são baseadas em métricas que medem propriedades arquiteturais relacionadas com os atributos de qualidade mencionados anteriormente.

Como os modelos de qualidade especificam características de qualidade por meio de conjuntos de atributos de qualidade, que são definidos com base no domínio específico; onde: (i) as características de qualidade representam características que um software deve apresentar, e (ii) os atributos de qualidade são os critérios de qualidade, que caracterizam a característica associada; no contexto de avaliação de projeto de PLA para SBPD é necessário mapear atributos de qualidade com propriedades arquiteturais medidas por métricas. Nesse sentido, o modelo de qualidade proposto segue o meta-modelo apresentado na Figura - 3.1, onde as características de qualidade são compostas por atributos de qualidade que contêm propriedades arquiteturais que podem ser medidas por uma ou várias métricas.

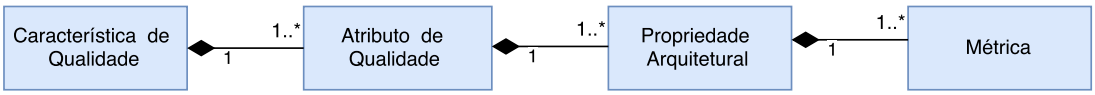

Figura 3.1: Meta-model do Modelo de Qualidade

Definido o meta-modelo e tendo em conta os (sub)atributos do modelo de qualidade proposto por Ahrens et al. (2013), foram classificadas todas as métricas das funções existentes no modelo de avaliação da MOA4PLA, em vista de mapear propriedades arquiteturais considerando os atributos de qualidade. A Tabela - 3.1 apresenta o modelo de qualidade para SBPD, seguindo a classificação de características e atributos de qualidade propostos por Ahrens et al. (2013). Portanto, define-se o modelo de qualidade tendo em consideração duas características de qualidade (***Reusabilidade*** *e* ***Mutabilidade***) as quais são compostas pelos atributos de qualidade ***Modularidade, Compreensibilidade e Extensibilidade*** cujas propriedades arquiteturais são mostradas na Tabela - 3.1, onde cada propriedade arquitetural pode ser medida por uma ou várias métricas definidas na Tabela - 2.1.

Tabela 3.1: Modelo de qualidade para o modelo de avaliação

Características de Qualidade	**Atributos de Qualidade**	**Propriedades Arquiteturais**	**Métricas**
Reusabilidade	Modularidade	Difusão de Características	CDAC
			CDAI
			CDAO
		Entrelaçamento de Características	CIBC
			IIBC
			OOBC
		Coesão baseada em Características	LCC
		Coesão relacional	H
		Acoplamento	DepPack
			DepIn
			DepOut
			CDepIn
			CDepOut
Mutabilidade	Extensibilidade	-	Ext
	Compreensibilidade	Tamanho	NumOps
		Elegância	NAC
			EC
			ATMR

Definido o modelo de qualidade, pode-se observar que: (i) o atributo de qualidade ***Modificabilidade*** ainda não é avaliado pelo modelo de avaliação e (ii) existem outras propriedades arquiteturais de projeto relevantes na concepção da PLA, que também não foram incluídas no modelo de qualidade proposto; tais como: *complexidade da PLA*, *adaptabilidade* e *similaridade de um projeto de PLA*. Em consequência, na próxima seção

apresentam-se as métricas selecionadas para estender o modelo de qualidade proposto, em vista de enriquecer o modelo de avaliação atual.

3.1.1 Métricas para a extensão do modelo de avaliação

Como as métricas consideram aspectos inerentes à estrutura de arquiteturas de LPS, considerando o trabalho de (Delgado et al., 2016), foram selecionadas varias métricas a serem utilizadas no modelo de avaliação da MOA4PLA, as quais são descritas brevemente a seguir.

M1: WOCS (*Weighted Operations per Component or Service*) (Ribeiro et al., 2010). WOCS mede a complexidade do serviço ou componente, em termos de suas operações (métodos) que serão necessárias para outros serviços ou componentes; onde serviço é definido como uma função sem um estado, independente, que aceita uma(s) chamada(s) e retorna uma resposta(s) através de uma interface bem definida.

A métrica indica que as operações com muitos parâmetros formais são mais prováveis de serem complexas do que as operações que exigem menos parâmetros.

M2: CBCS (*Coupling Between Components or Services*) (Ribeiro et al., 2010) é calculada com base no número de relacionamentos entre um serviço A, por exemplo e outros serviços. Ela pode ser descrita matematicamente da seguinte forma:

$$\mathbf{CBCS} = \sum_{i \neq j = 1 K_n}^{n} AiBj \tag{3.1}$$

Onde n é o número de serviços; AiBj = 0 se AiBj não estão relacionados e AiBj = 1 se Ai se relaciona com Bj (Ribeiro et al., 2010) .

Para um serviço A, quanto maior o valor da métrica do CBCS, quer dizer que maior é a dependência de A com outros (Quynh, 2009). Esta métrica é baseada no acoplamento entre de objetos (CBO) de Chidamber e Kemerer (1994) e o acoplamento entre componentes (CBC) definido em (Sant'Anna et al., 2003).

M3: SSC (*Structure Similarity Coefficient*) (Zhang et al., 2008) é proposta para medir a similaridade da PLA mediante a seguinte equação:

$$\mathbf{SSC} = \frac{|Cc|}{|Cc| + |Cv|} \tag{3.2}$$

Na equação, Cc é o número de componentes comuns na PLA e Cv é o número de componentes variáveis na PLA. De acordo com a fórmula, a PLA que tem mais componentes comuns e menos componentes variáveis possui mais benefícios de reutilização.

M4: SCC (*Strong Coupling Coefficient*) (Zhang et al., 2008) é proposta para medir o forte acoplamento de variabilidade mediante a seguinte equação:

$$\mathbf{SCC} = 1 - \frac{|IVP|}{|VP|} \tag{3.3}$$

Na equação, VP é o número de pontos de variabilidade nas interfaces da PLA e IVP é o número de pontos de variabilidade independentes, que não têm relações de dependência com os outros. A PLA que tem mais pontos de variabilidade é mais complexa e difícil de ser projetada (Zhang et al., 2008).

M5: SVC (*Structure Variability Coefficient*) (Zhang et al., 2008) é definida para medir a variabilidade da estrutura da PLA mediante a Equação 3.4. SVC também pode ser usada para medir a variabilidade da estrutura de componentes compostos (Zhang et al., 2008).

$$\mathbf{SVC} = \frac{|Cv|}{|Cc| + |Cv|} \tag{3.4}$$

Na equação, Cv é o número de componentes variáveis na PLA e Cc é o número de componentes comuns na PLA .

M6: AV (*Architecture variability*) (Zhang et al., 2008) mede a variabilidade total da PLA. AV é definida na equação a seguir:

$$\mathbf{AV} = |Cv| + \sum_{i} AV(Ci) \tag{3.5}$$

Onde: $|Cv|$: O número de componentes variáveis, $AV(Ci)$ é a variabilidade do interior do componente Ci. Se Ci é um componente composto, $AV(Ci)$ pode ser calculado mediante a Equação 3.5. Se Ci é o componente básico e tem variabilidade interior, $AV(Ci)$= 1, se não $AV(Ci) = 0$ (Zhang et al., 2008).

3.1.2 Mapeamento das métricas selecionadas no modelo de qualidade

Dado que as métricas mencionadas anteriormente permitem enriquecer o modelo de avaliação, procede-se a inclusão delas, no modelo de qualidade proposto. A Tabela - 3.2 apresenta o mapeamento das métricas selecionadas as quais são classificadas tendo em conta a propriedade arquitetural que ela mede. Na tabela, a propriedade arquitetural *Variabilidade*, assim como *Similaridade* e *Acoplamento de Variabilidade* são novas propriedades arquiteturais a serem avaliadas no modelo.

Tabela 3.2: Mapeamento das métricas selecionadas no modelo de qualidade

Característica de Qualidade	Atributo de Qualidade	Propriedade Arquitetural	Métrica	Descrição
Reusabilidade	Modularidade	Acoplamento	CBSC	Mede o número de relacionamentos entre serviços fornecidos por componentes (Ribeiro et al., 2010).
		Similaridade	SSC	Mede a similaridade da PLA tendo en conta o número total de componentes comuns da PLA (Zhang et al., 2008).
		Acoplamento de Variabilidade	SCC	Mede o acoplamento entre os pontos de variabilidade da PLA (Zhang et al., 2008).
Mutabilidade	Compreensibilidade	Tamanho	WOCS	Mede a complexidade dos componentes em termos de operações (métodos) que podem ser requeridos por outros componentes (Ribeiro et al., 2010).
		Variabilidade	SVC	Mede a variabilidade da estrutura da PLA, tendo em conta o número de componentes variaveis da PLA (Zhang et al., 2008).
			AV	Mede a variabilidade total da estrutura da PLA, tendo em conta número de componentes compostos e variáveis da PLA (Zhang et al., 2008).

De maneira geral, na próxima seção é apresentado o modelo de qualidade para o modelo de avaliação da MOA4PLA.

3.1.3 Modelo de qualidade para o modelo de avaliação da MOA4PLA

A seguir apresenta-se o modelo de qualidade definido para SBPD, seguindo a classificação de caraterísticas e atributos de qualidade propostos por (Ahrens et al., 2013) e as métricas mencionadas na seção anterior. Na Figura - 3.2, define-se o modelo de qualidade tendo em consideração as características de qualidade ***Reusabilidade*** e ***Mutabilidade***, as quais são compostas por atributos de qualidade ***Modularidade***, ***Extensibilidade***, ***Compreensibilidade*** e ***Modificabilidade*** cujas propriedades arquiteturais são mostradas na figura.

Cada propriedade arquitetural pode ser medida por uma ou várias métricas. As métricas adicionadas no modelo foram: **WOCS** (*Weighted Operations per Component or Service*), **CBCS** (*Coupling Between Components or Services*), **SSC** (*Structure Similarity Coefficient*), **SCC** (*Strong Coupling Coefficient*), **SVC** (*Structure Variability Coefficient*), **AV** (*Architecture variability*) definidas na Tabela - 2.1. Nesse sentido, as propriedades arquiteturais ***Similaridade***,***Acoplamento de Variabilidade*** e ***Variabilidade*** relacionadas com os atributos de qualidade ***Modularidade*** e ***Compreensibilidade*** como é mostrado na Figura - 3.2, agora podem ser medidas graças à inclusão das novas métricas no modelo.

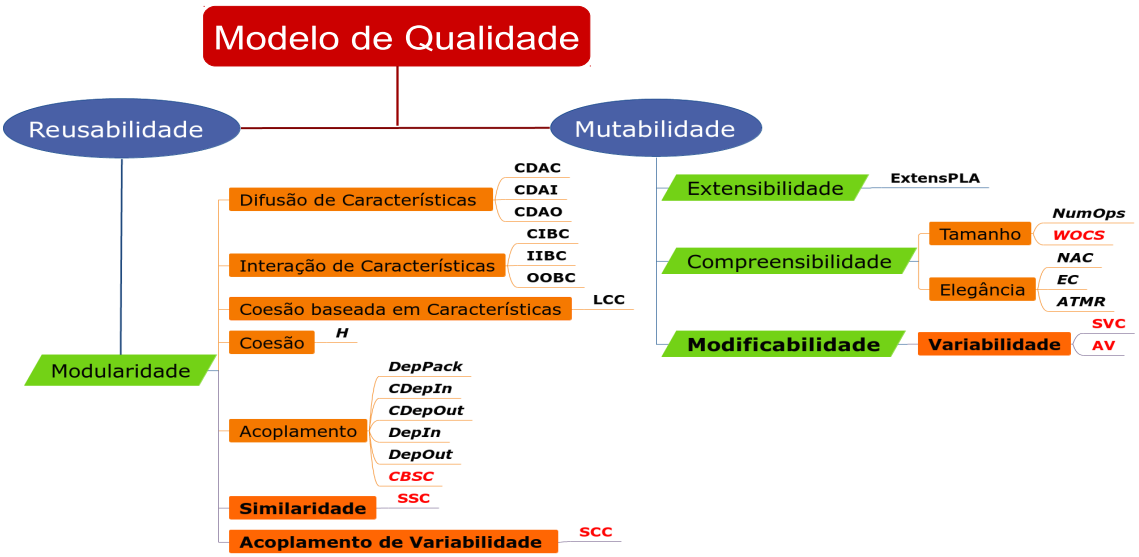

Figura 3.2: Modelo de Qualidade da MOA4PLA

Uma vez apresentado o modelo de qualidade para o modelo de avaliação da MOA4PLA, na próxima seção são apresentadas as novas funções objetivo baseadas nas métricas selecionadas e incluídas no modelo de qualidade proposto.

3.2 Novas funções objetivo para o modelo de avaliação da MOA4PLA

Em vista de enriquecer o modelo de avaliação da MOA4PLA com métricas para a LPS que permitam avaliar outras propriedades arquiteturais como similaridade e variabilidade da PLA, a Tabela - 3.3 apresenta para cada uma das novas funções objetivo, a métrica na qual é baseada, assim como a propriedade arquitetural que a mesma avalia e logo são definidas as novas funções objetivo.

Component Size - CS(pla): Tem como objetivo avaliar a quantidade de operações existentes em classes e interfaces da PLA. CS(pla) é definida pela Equação 3.6, onde **op** simboliza o número de operações.

$$\mathbf{CS(pla)} = \sum_{i=1}^{op} WOCS \tag{3.6}$$

Tabela 3.3: Breve descrição das novas funções objetivo

Função Objetivo		**Métrica**	**Propriedade Arquitetural**
Nome	**Sigla**		
Component Size	**CS(pla)**	WOCS	Tamanho
Relationship Coupling of Components	**RCC(pla)**	CBCS	Acoplamento
Similarity of Design	**SD(pla)**	SSC	Similaridade
Coupling of Variability	**CV(pla)**	SCC	Acoplamento de variabilidade
Structure of Variability	**SV(pla)**	SVC	Variabilidade
Total Variability	**TV(pla)**	AV	Variabilidade

Relationship Coupling of Components - RCC(pla): Tem como objetivo avaliar o acoplamento existente entre os componentes da PLA. RCC(pla) é definida pela Equação 3.7, onde **itf** simboliza o número de interfaces.

$$\mathbf{RCC(pla)} = \sum_{i=1}^{itf} CBCS \tag{3.7}$$

Similarity of Design - SD(pla): Tem como objetivo avaliar o nível de similaridade da PLA mediante o número de componentes variáveis. SD(pla) é definida pela Equação 3.8 e calculada como a inversa da métrica SSC em vista de maximizar o número de componentes variáveis da PLA.

$$\mathbf{SD(pla)} = \frac{1}{SSC} \tag{3.8}$$

Coupling of Variability - CV(pla): Tem como objetivo avaliar o nível de acoplamento de variabilidade considerando as dependências existentes nos pontos de variabilidade da PLA. CV(pla) é definida pela Equação 3.9.

$$\mathbf{CV(pla)} = SCC \tag{3.9}$$

Structure of Variability - SV(pla): Tem como objetivo avaliar a estructura da variabilidade da PLA em termos de componentes variáveis existentes. SV(pla) é definida pela Equação 3.10.

$$\mathbf{SV(pla)} = SVC \tag{3.10}$$

Total Variability - TV(pla): Tem como objetivo avaliar a variabilidade total existente na PLA, tendo en conta os componentes variáveis e compostos da PLA. TV(pla) é definida pela Equação 3.11.

$$\mathbf{TV(pla)} = AV \tag{3.11}$$

Tendo em conta o modelo de qualidade para SBPD, as equações de todas as funções objetivo do modelo de avaliação da MOA4PLA são resumidas na Tabela - 3.4 seguindo a ordem da Figura - 3.2.

Tabela 3.4: Equações das funções objetivo do modelo de avaliação

Equações
1) **FM(pla)** $= \sum_{i=1}^{c} LCC + \sum_{i=1}^{f} CDAC + \sum_{i=1}^{f} CDAI + \sum_{i=1}^{f} CDAO + \sum_{i=1}^{f} CIBC + \sum_{i=1}^{f} IIBC + \sum_{i=1}^{f} OOBC$
2) **CM(pla)** $= \sum_{i=1}^{c} DepIn + \sum_{i=1}^{c} DepOut + \sum_{i=1}^{cl} CDepIn + \sum_{i=1}^{cl} CDepOut + \frac{\sum_{i=1}^{c} DepPack}{c} + \frac{\sum_{i=1}^{itf} NumOps}{itf} + \frac{1}{\sum_{i=1}^{c} H}$
3) **DC(pla)** $= \sum_{i=1}^{f} CDAI + \sum_{i=1}^{f} CDAO + \sum_{i=1}^{f} CDAC$
4) **EC(pla)** $= \sum_{i=1}^{f} CIBC + \sum_{i=1}^{f} IIBC + \sum_{i=1}^{f} OOBC$
5) **LCC(pla)** $= \sum_{i=1}^{c} LCC$
6) **ACOMP(pla)** $= \sum_{i=1}^{c} DepIn + \sum_{i=1}^{c} DepOut$
7) **ACLASS(pla)** $= \sum_{i=1}^{cl} CDepIn + \sum_{i=1}^{cl} CDepOut$
8) **COE(pla)** $= \sum_{i=1}^{cl} H$
9) **CS(pla)** $= \sum_{i=1}^{op} WOCS$
10) **SD(pla)** $= \frac{1}{SSC}$
11) **CV(pla)** $= SCC$
12) **Ext(pla)** $= \frac{1}{ExtensPLA}$
13) **TAM(pla)** $= \frac{\sum_{i=1}^{itf} NumOps}{itf}$
14) **RCC(pla)** $= \sum_{i=1}^{itf} CBCS$
15) **Eleg(pla)** $= NAC(pla) + EC(pla) + ATMR(pla)$
16) **SV(pla)** $= SVC$
17) **TV(pla)** $= AV$

3.3 Implementação das novas funções objetivo

A implementação da ferramenta OPLA-Tool, proposta por (Colanzi et al., 2014), é composta por quatro módulos principais, os quais são relacionados como se apresenta na

Figura - 2.4 da Seção 2.3.2. Nesse sentido, para a implementação das novas funções objetivo foram necessárias algumas alterações nos módulos: i) OPLA-GUI, e ii) OPLA-Core. A seguir, nas Subseções 3.3.1 e 3.3.2 explica-se detalhadamente cada uma das alterações realizadas. Uma visão geral de todas as classes alteradas e acrescentadas pode ser vista no diagrama de classes apresentado no Anexo A.3.

3.3.1 Módulo OPLA-GUI da OPLA-Tool

O módulo OPLA-GUI fornece uma interface gráfica com o usuário, para facilitar o uso da abordagem MOA4PLA. Sendo assim, este módulo é responsável por fornecer ao usuário acesso à abordagem MOA4PLA (Colanzi et al., 2014) de forma fácil e produtiva. Nessa perspectiva, para a implementação das novas funções objetivo foi necessário mudar algumas classes, acrescentando métodos. A Figura - 3.3 representa os pacotes que sofreram alterações e a Tabela - 1.2 e Tabela - 1.1 apresentada no Anexo A.1 mostram as classes e métodos acrescentados ou mudados durante a implementação. Para desenvolvimento desse módulo foi usada a tecnologia Java Swing.

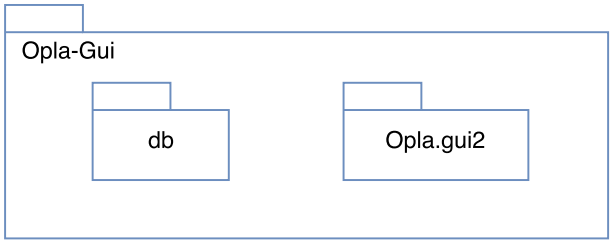

Figura 3.3: Pacotes alterados no Pacote OPLA-Gui - Implementação de novas métricas

3.3.2 Módulo OPLA-Core da OPLA-Tool

O módulo OPLA-Core implementado por (Féderle et al., 2015) estende a implementação padrão do *framework* jMetal (Durillo et al., 2010), adicionando ao jMetal novas métricas para avaliação de PLAs. A Figura - 3.4 apresenta os pacotes que sofreram alterações e no Anexo A.2 são apresentadas as classes e métodos acrescentados ou mudados durante a implementação das novas funções objetivo.

3.4 Considerações finais

Neste capítulo foi apresentada a metodologia aplicada para a extensão do modelo de avaliação, foi definido o modelo de qualidade para a MOA4PLA, foram definidas novas

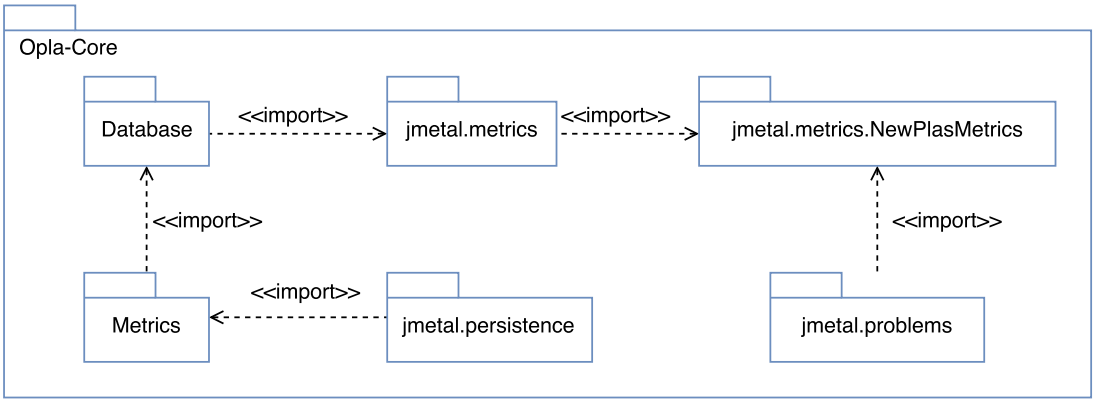

Figura 3.4: Pacotes alterados no Pacote OPLA-Core - Implementação de novas métricas

funções objetivo, assim como as alterações realizadas para a implementação das mesmas. De maneira geral, o presente capítulo contém as seguintes contribuições: (i) extensão do modelo de avaliação permitindo medir novos atributos de qualidade e propriedades arquiteturais, (ii) formalização de um modelo de qualidade para SBPD (concretização que ainda não havia sido feita em estudos anteriores) e; (iii) proposta de funções objetivo para uso das métricas incluídas no modelo, o que permite o uso das mesmas pela primeira vez no contexto de SBPD. No próximo capítulo apresentam-se o estudo exploratório realizado com o propósito de analisar a correlação existente entres as novas funções objetivo propostas.

4

Estudo exploratório

Após a extensão do modelo de avaliação com novas funções objetivo, o modelo contém um total 17 de funções objetivo, como isto é um número elevado de objetivos a serem otimizados simultaneamente por meio de algoritmos de busca; o arquiteto da LPS deve selecionar quais objetivos quer priorizar durante a otimização. Nesse sentido, informações sobre a possível correlação entre as funções objetivo são importantes para minimizar o número de objetivos a ser selecionado para otimização.

Este capítulo está organizado da seguinte forma: a Seção 4.1 apresenta a sequência de atividades realizadas para o planejamento e execução dos experimentos definidos bem como a análise dos resultados e lições aprendidas.Na Seção 4.2 são apresentadas as ameaças à validade dos experimentos.

4.1 Planejamento e execução do estudo exploratório

Esta seção descreve o planejamento e execução do estudo exploratório realizados. A Figura - 4.1 mostra a sequência de atividades realizadas, tendo em conta as fases de experimentação definidas em (Basili et al., 1986) as quais são: (i) definição, (ii) planejamento, (iii) operação e (iv) interpretação. A primeira atividade representa a definição do experimento que contém a motivação, objeto, finalidade e perspectiva do estudo. A segunda atividade define o planejamento do estudo. A operação do estudo é a terceira atividade e consiste em preparar e executar o experimento, bem como coletar dados e analisar a distribuição de dados, onde diferentes testes de correlação são aplicados de acordo com o tipo de distribuição destes. Na última atividade (Interpretação), os

resultados são analisados e discutidos. As subseções a seguir descrevem cada atividade mencionada.

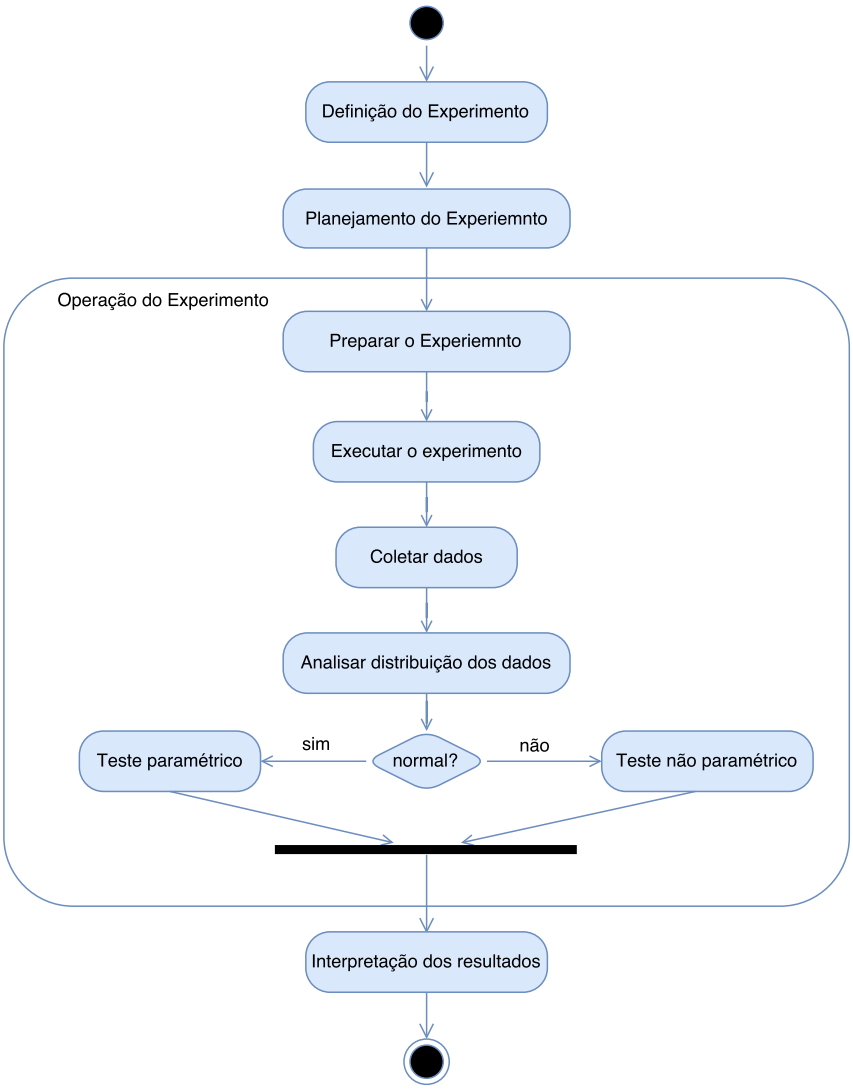

Figura 4.1: Sequência de atividades do processo de experimentação

4.1.1 Definição dos experimentos

Tendo em conta a primeira fase do processo de experimentação proposto por (Basili et al., 1986), a definição do estudo realizado é apresentada na Tabela - 4.1.

O teste de correlação foi realizado por meio de combinações de duas funções objetivo. Então, o estudo é dividido em quatro experimentos, onde cada experimento é chamado como segue: Experimento I que envolve as funções objetivo SV(pla) e SD(pla), Experimento II que envolve as funções objetivo SV(pla) e TV(pla), Experimento III que envolve as funções objetivo SD(pla) e TV(pla) e Experimento IV que envolve as funções objetivo

Tabela 4.1: Definição do estudo exploratório

Definição	
Motivação	Investigar a correlação entre métricas das funções objetivo propostas
Propósito	de caracterizar
Objeto	possível correlação existente entre cada par de funções objetivo
Perspectiva	do pesquisador

RCC(pla) e ACOMP(pla). Todos os experimentos foram executados usando a ferramenta OPLA-Tool (Féderle et al., 2015).

Os experimentos foram realizados para investigar a correlação entre pares de métricas, porque a compensação estabelecida entre os objetivos varia de acordo com as métricas. Para cada experimento, conjunto de amostra diferentes e independentes foram obtidas porque as soluções são obtidas de acordo com os objetivos selecionados para ser otimizado.

4.1.2 Planejamento dos experimentos

Os experimentos foram realizados em um ambiente acadêmico. Para cada experimento, a variável independente o projeto de PLA e as variáveis dependentes variam de acordo com o experimento: (i) para o Experimento I são os valores das funções objetivo SV(pla) e SD(pla), (ii) para o Experimento II são os valores das funções objetivo SV(pla) e TV(pla), (iii) para o Experimento III são os valores das funções objetivo SD(pla) e TV(pla); e para o Experimento IV são os valores das funções objetivo RCC(pla) e ACOMP(pla); conforme apresentado na Tabela - 4.2. A correlação é medida utilizando os *fitness* das soluções obtidas pelo processo de otimização, onde *fitness* de uma solução é o valor de cada métrica para o projeto de PLA, por exemplo, considerando uma solução obtida no Experimento I, seu *fitness* é um par de valores para (SV(pla), SD(pla)).

É importante destacar que o pares das funções objetivo foram selecionadas em vista de avaliar as funções objetivo propostas, igualmente os pares foram montados de acordo com as hipóteses extraídas da prova de conceito, tendo em consideração: (i) o análise dos resultados das métricas incluídas nas funções objetivo SV(pla) e SD(pla), mostra que são conflitantes, levando-se em conta que o número de componentes comuns e o número de componentes variáveis influenciam o valor das funções objetivo SV(pla) e SD(pla), conforme exposto nas Equações 3.9 e 3.10; isto significa que quanto maior for o número de componentes variáveis e menor o número de componentes comuns dentro da PLA, melhor será o valor de SV(pla). Caso contrário, o valor de SV(pla) será melhor, (ii) as funções objetivo SV(pla) e TV(pla) medem a propriedade arquitetural ***Variabilidade*** considerando o número de componentes variáveis existentes na PLA, por tanto conhecer

o comportamento do número de componentes variáveis uma vez executadas as funções da uma visão de qual das duas funções objetivo seria melhor selecionar como objetivo para o processo de avaliação, (iii) as funções objetivo SD(pla) e TV(pla) medem propriedades arquiteturais diferentes (***Similaridade*** e ***Variabilidade***) mas consideram o número de componentes comuns e o número de componentes variáveis existentes na PLA os quais podem ser conflitantes, e; (iv) as funções objetivo RCC(pla) e ACOMP(pla) utilizam métricas que medem a propriedade arquitetural ***Acoplamento***, no entanto a primeira está focada no acoplamento entre interfaces e a segunda no acoplamento entre componentes.

Tabela 4.2: Planejamento dos experimentos

Experimento	**Variáveis independentes (cada PLA)**	**Variáveis dependentes**
I	AGM, MM, Bank, BET	SV e SD
II	AGM, MM, Bank, BET	SV e TV
II	AGM, MM, Bank, BET	SD e TV
IV	AGM, MM, Bank, BET	RCC e ACOMP

O estudo envolveu quatro experimentos. Para cada experimento, duas hipóteses foram definidas, sendo (i) a hipótese nula H_0 que representa que não há correlação significativa entre as métricas envolvidas no experimento e (ii) a hipótese alternativa H_1, que representam a existência de correlação significativa entre as métricas envolvidas no experimento.

4.1.3 Operação dos experimentos

Uma vez planejados os experimentos procede-se a executá-los, para isso são definidas as seguintes atividades:

Preparação dos experimentos: Os experimentos envolvem o uso dos projetos de PLAs: AGM, MM, Bank e BET, cujas principais informações são apresentadas na Tabela - 4.3. Todas as PLAs são baseadas em componentes e seguem o estilo em camadas: (i) AGM (*Arcade Game Maker*) é uma LPS criada pelo SEI que inclui três jogos de arcade: Brickles, Bowling e Pong (SEI, 2016), (ii) MM (*Mobile Media*) é uma LPS que apoia o gerenciamento de diferentes tipos de mídias para dispositivos móveis, incluindo música, vídeos e fotos (Contieri Junior et al., 2011), (iii) Bank(*System Banking*) é uma LPS que suporta o gerenciamento de sistemas bancários (Gomaa, 2011); e (iv) BET (*Bilhetes Eletrônicos em Transporte Urbano*) é uma LPS utilizada para o gerenciamento de transporte urbano, principalmente comercial, possuindo várias características pertinentes

a pagamento, itinerário, gerenciamento de bilhetes, dentre outros (Donegan e Masiero, 2007).

Tabela 4.3: Características das PLAs

PLA	# Componentes	# Interfaces	# Classes	# Features	# Variabilidades
AGM	9	14	30	11	5
MM	8	15	14	14	7
Bank	4	5	25	16	3
BET	56	30	115	18	8

Execução dos experimentos: A ferramenta OPLA-Tool foi usada para executar os experimentos. A Figura - 4.2 mostra a sequência das atividades desenvolvidas para cada experimento. Cada experimento foi executado com o NSGA-II e os parâmetros do algoritmo foram ajustados de acordo com trabalhos anteriores (Colanzi et al., 2014; Féderle et al., 2015; Guizzo et al., 2014) onde o tamanho da população foi igual a 100 indivíduos, o número de avaliação de *fitness* igual a 30000, foram aplicados todos os operadores de mutação do MOA4PLA com uma probabilidade de mutação igual a 0,9. Cada experimento foi executado 30 rodadas para cada PLA de acordo com o sugerido por (Arcuri e Briand, 2014). O número de avaliações de *fitness* foi usado como critério de parada.

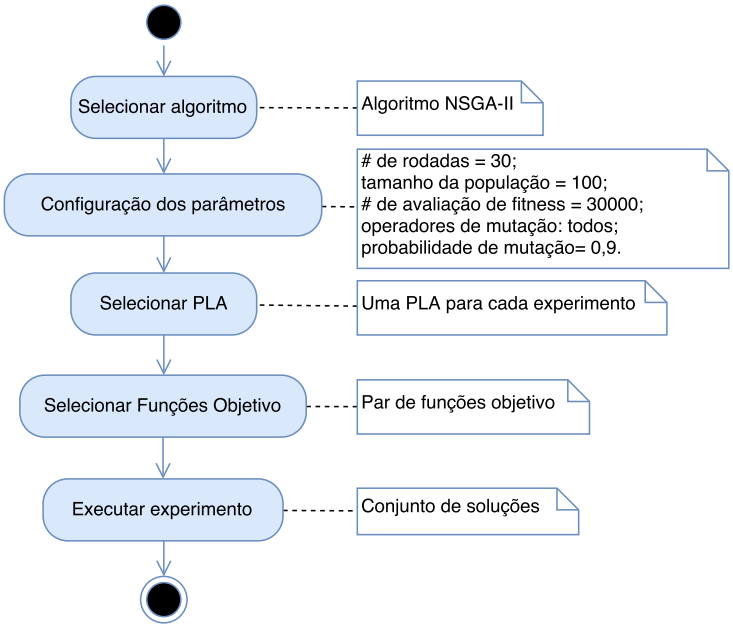

Figura 4.2: Configuração dos experimentos

Coleta dos dados: No final para cada execução dos experimentos os resultados das funções objetivo foram coletados para posterior análise.

Análise dos dados: Uma avaliação de normalidade dos dados é um pré-requisito para muitos testes estatísticos, porque dados normais são uma suposição subjacente em testes paramétricos. Existem dois principais métodos de avaliação de normalidade: numericamente e graficamente. O *Quantile-Quantile* ou Q-Q plot fornece uma comparação gráfica da amostra, neste sentido, é importante também ter em conta o tamanho da amostra quando julgar quão perto são os pontos da linha reta no Q-Q plot. O teste de Shapiro-Wilk para normalidade (Shapiro e Wilk, 1965) permite confirmar os resultados gráficos. O teste rejeita a hipótese de normalidade quando o p-value é menor ou igual a 0,05.

Tendo em conta a distribuição dos dados são aplicados teste estatísticos. Nesse sentido o teste não paramétrico de correlação de Spearman's (Hauke, 2011) foi aplicado para verificar a correlação entre as métricas cujos dados apresentam distribuição não-normal. Por outro lado, o teste paramétrico de correlação de Pearson (Hauke, 2011) foi aplicado para dados com distribuição normal. Os resultados obtidos dos testes de correlação são mostrados na tabela Tabela - 4.14.

A análise do coeficiente de correlação para ambos os testes considera os valores de -1 a + 1. Um valor de + 1 mostra que as variáveis são perfeitamente lineares, relacionadas por um relacionamento crescente e um valor de -1 mostra que as variáveis são perfeitamente lineares, relacionados por uma relação decrescente. Mesmo assim, um valor de 0 mostra que as variáveis não são linearmente relacionadas. Ademais é considerada uma correlação forte se o coeficiente de correlação é maior que 0,8 e uma correlação fraca se o coeficiente de correlação é inferior a 0,5, conforme mostrado na Figura - 4.3.

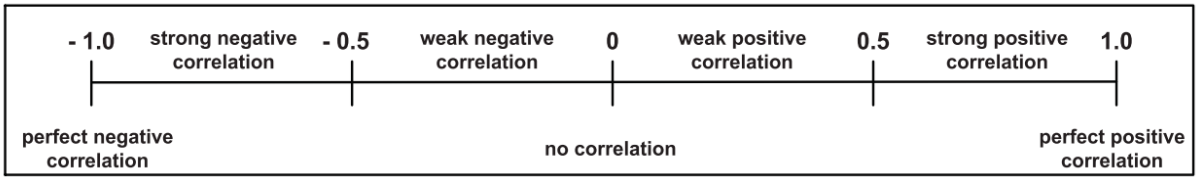

Figura 4.3: Escala de correlação

A seguir são definidas para cada experimento as hipóteses, os resultados do teste de normalidade aplicado, o análise do comportamento dos dados assim como, uma breve discussão sobre os resultados e lições aprendidas.

4.1.4 Experimento I

Objetivo: Este experimento foi definido em vista de investigar a possível correlação entre SV(pla) e SD(pla). Apesar que SV(pla) mede a propriedade arquitetural *Variabilidade* e SD(pla) mede a propriedade arquitetural *Similaridade* ambas funções objetivo estão baseadas em métricas que tem em conta o número de componentes comuns e o número de componentes variáveis existentes na PLA. Nesse sentido, é importante conhecer o comportamento do número componentes comuns e do número componentes variáveis uma vez executadas as funções objetivo.

Hipóteses

H_{sv-sd0} : não existe correlação significativa entre SV(pla) e SD(pla);
H_{sv-sd1} : existe significativa correlação entre SVC e SSC.

Resultados do teste de normalidade: A Tabela - 4.4 apresenta os resultados do teste de normalidade do Experimento I. A Figura - 4.4 mostra o Q-Q plot das amostras (SD(pla) e SV(pla)) obtidas para AGM no Experimento I. Pode-se observar que os pontos não estão perto da linha reta, indicando que os dados não são normalmente distribuídos. Este resultado é compatível com o resultado obtido pelo teste de normalidade de Shapiro-Wilk mostrado na Tabela - 4.4. Os dados obtidos para BET no Experimento I também não seguem uma distribuição normal, apesar de ter uma representação gráfica diferente (ver Figura - 4.5). Os resultados obtidos para MM seguem uma distribuição normal (p-value > 0.05).

Tabela 4.4: Resultados do Teste de Normalidade-Experimento I

PLA	Experimento I			
	SV p-value	**Distribuição**	**SD p-value**	**Distribuição**
AGM	**0.02831**	Não-normal	**0.02693**	Não-normal
MM	0.7392	Normal	0.7398	Normal
Bank	Valores idênticos	-	Valores idênticos	-
BET	**2.2e-16**	Não-normal	**2.2e-16**	Não-normal

Os resultados obtidos para Bank foram os mesmos para todas as funções objetivo em todos os experimentos realizados. Isto significa que a solução encontrada no processo de otimização tem sempre o mesmo *fitness*. Assim, não foi possível aplicar qualquer teste estatístico nos dados da LPS Bank devido à falta de diversidade de dados.

Análise do comportamento dos dados: De acordo com os resultados apresentados na Tabela - 4.5 no Experimento I para AGM existe correlação positiva perfeita (rho =

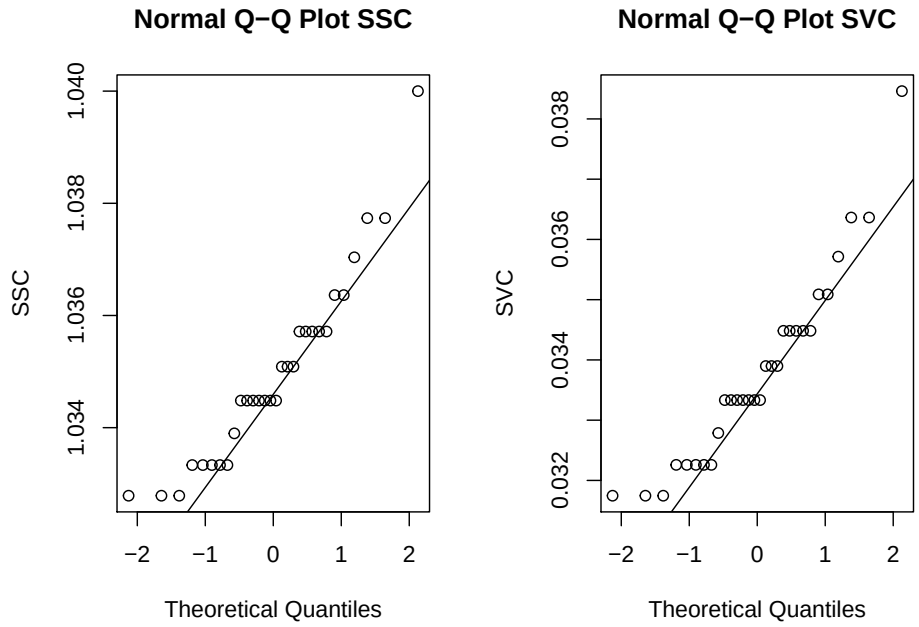

Figura 4.4: Q-Q Plots do Experimento I - AGM

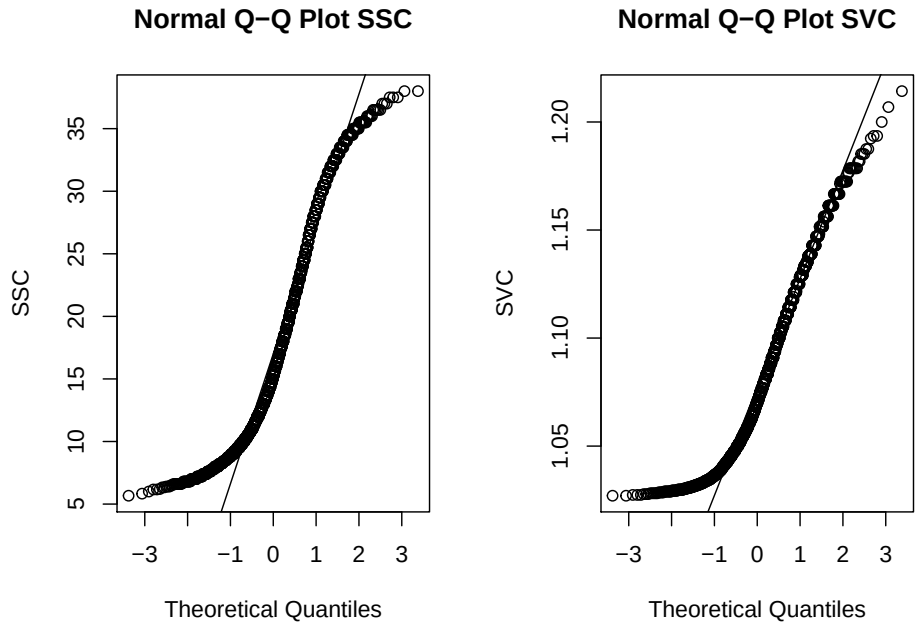

Figura 4.5: Q-Q Plots do Experimento I - BET

1) entre as funções objetivo SV(pla) e SD(pla), tendo em conta a escala de correlação apresentada na Figura - 4.3. Para MM também existe correlação positiva forte (cor = 0,99) entre as funções objetivo SV(pla) e SD(pla) baseada na escala da mesma figura. Em cada rodada, obteve-se apenas uma solução para estas PLAs. Isto corrobora com os

resultados do teste de correlação, porque, se as funções objetivo estivessem em conflito, seriam encontradas várias soluções com possíveis *trade-offs* diferentes entre as mesmas.

Tabela 4.5: Testes de correlação do Experimento I

Experimento I			
PLA	**Teste aplicado**	**p-value**	**Nível de correlação**
AGM	Spearman's	< 2.2e-16	1
MM	Pearson's	< 2.2e-16	0.9999
BET	Spearman's	< 2.2e-16	-1

No entanto, para BET os resultados de correlação foram diferentes (rho =-1). Seguindo a mesma escala de correlação (Figura - 4.3) existe uma forte correlação negativa entre as funções objetivo SV(pla) e SD(pla). Realmente, estes foram os resultados esperados, porque considerando a natureza das funções objetivo SD(pla) (Similaridade) e SV(pla) (Variabilidade), quanto maior o número de componentes comuns na PLA, menor o número de componentes variáveis na PLA e vice-versa.

Discussão sobre os resultados: Para apoiar a análise, Tabela - 4.6 mostra o valor de SD(pla) e SV(pla), antes de ser otimizado (coluna nomeada *Fitness* Original), o melhor valor de SD(pla) (coluna nomeada Melhor SD), o melhor valor de SV(pla) (coluna nomeada Melhor SV) e o melhor *trade-off* enre SD(pla) e SV(pla) (coluna nomeada Melhor *Trade-off*) depois da otimização.

Tabela 4.6: Características das PLAs para o Experimento I

PLAs	**Fitness Original (SD,SV)**	**Melhor SD (SD,SV)**	**Melhor SV (SD,SV)**	**Melhor Trade-off (SD,SV)**
AGM	(1.2857,0.2223)	(1.0328,0.0317)	(1.0328,0.0317)	(1.0328,0.0317)
MM	(1.1428,0.125)	(1.0309,0.0231)	(1.0309,0.0231)	(1.0309,0.0231)
Bank	(1.3333,0.25)	(1.1111,0.10)	(1.1111,0.10)	(1.1111,0.10)
BET	(1.0566,0.5353)	(1.0219, 0.0215)	(1.0219,0.0215)	(1.0638,0.0588)

De acordo com os resultados pode-se notar que:

Do ponto de vista de SD(pla): o número de componentes comuns aumentou em AGM, MM e Bank, portanto o valor do SD(pla) diminuiu consideravelmente nestas PLAs. No caso da BET, o valor de SD(pla) aumentou após a otimização do projeto, provavelmente devido ao aumento do número de componentes variáveis.

Do ponto de vista de SV(pla): os valores de SD(pla) para todos os projetos de PLA diminuíram após a otimização. Isto é devido às alterações nos valores dos componentes comuns mencionados anteriormente, tendo em conta que o número de componentes

comuns influencia sobre o valor de SV(pla), conforme mostrado na Equação 3.10. O número de componentes variáveis existentes nas PLAs foi mantido para AGM e Bank. No caso da BET, inicialmente, houve três componentes variáveis (PassageiroMgr, ViacaoMgr e CartaoMgr) e após a otimização, as variabilidades da PLA foram distribuídas em cinco componentes (PassageiroMgr, ViacaoMgr, CartaoMgr, LinhaMgr e PagamentoCartaoMgr), levando a um aumento do número dos componentes variáveis.
Para este experimento, fica claro que a correlação depende do projeto de PLA fornecido como entrada para o processo de otimização. Para AGM e MM, os resultados apontaram uma correlação positiva. Para ambas as PLAs, as características da LPS estão difusas em vários componentes diminuindo a modularização de caraterísticas. Então, durante o processo de otimização, vários componentes são criados para modularizar características, levando ao aumento do número de componentes comuns. Por outro lado, na BET as características da LPS estão bem modularizadas o que impediu o aumento do número de componentes comuns. Isto pode justificar a correlação negativa atestada pelo teste de correlação. Neste contexto, não é possível definir a existência de correlação entres as métricas definidas neste experimento. Estudos com outros projetos de PLA devem ser realizados para melhorar as evidências sobre a correlação entre as funções objetivo SD(pla) e SV(pla).

Lições aprendidas: Projetos de PLA com características difusas podem influenciar nos resultados obtidos, impactando na correlação das funções objetivo como pode ser observado no Experimento I. Apesar de não poder atestar o tipo de correlação entre as funções objetivo SD(pla) e SV(pla) é melhor otimizar uma de cada vez porque a similaridade e a variabilidade são dois conceitos naturalmente opostos.

4.1.5 Experimento II

Objetivo: Este experimento foi definido em vista de investigar a possível correlação entre SV(pla) e TV(pla), considerando que ambas funções objetivo medem a propriedade arquitetural *Variabilidade*. Nesse sentido, o foco é conhecer o comportamento do número componentes variáveis uma vez executadas as funções objetivo.

Hipóteses

H_{sv-tv0} : não existe correlação significativa entre SV(pla) e TV(pla);
H_{sv-tv1} : existe significativa correlação entre SV(pla) e TV(pla).

Resultados do teste de normalidade: Considerando os valores de SV(pla) obtidos no Experimento II, pode-se observar que estes têm distribuição normal para AGM e MM. No entanto os resultados obtidos para BET, têm distribuição Não-Normal, resultado afirmado no teste de Shapiro-Wilk (Tabela - 4.7) e como pode ser visto na Figura - 4.6 que representa o Q-Q Plot para as amostras de SV(pla) e TV(pla).

Tabela 4.7: Resultados do Teste de Normalidade-Experimento II

PLA	Experimento II			
	SV p-value	**Distribuição**	**TV p-value**	**Distribuição**
AGM	0.7727	Normal	Valores idênticos	-
MM	0.297	Normal	Valores idênticos	-
Bank	Valores idênticos	-	Valores idênticos	-
BET	**6.14 e-09**	Não-Normal	**1.77 e-08**	Não-Normal

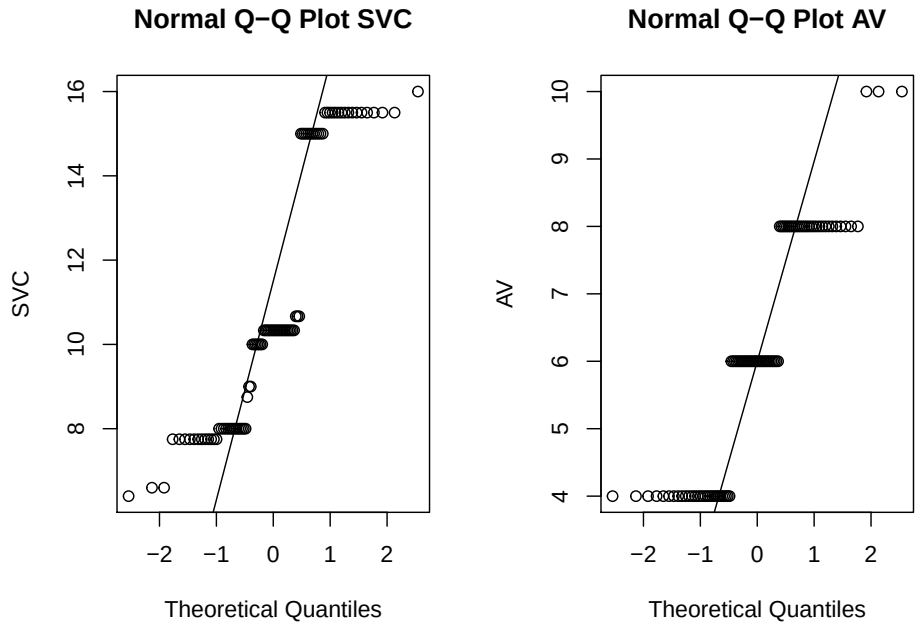

Figura 4.6: Q-Q Plots do Experimento II - BET

Análise do comportamento dos dados: Neste experimento, foi possível aplicar o teste de correlação somente para a BET, onde os resultados apontam uma forte correlação negativa (rho =-0.96) (Tabela - 4.8). Devido à falta de diversidade de dados da LPS Bank e que os valores de TV(pla) são idênticos (Tabela - 4.7), não foi possível aplicar os testes de normalidade e correlação nesses caso.

Tabela 4.8: Testes de correlação do Experimento II

Experimento II			
PLA	**Teste aplicado**	**p-value**	**Nível de correlação**
BET	Spearman's	< 2.2e-16	-0.960358

Discussão sobre os resultados: A Tabela - 4.9 mostra os valores originais de SV(pla) e TV(pla) (coluna nomeada *Fitness* Original), bem como o melhor valor de SV(pla) (coluna nomeada Melhor SV), melhor valor de TV(pla) (coluna nomeada Melhor TV) e o melhor *trade-off* entre TV(pla) (coluna nomeada Melhor TV) após a otimização.

Tabela 4.9: Características das PLAs para o Experimento II

PLAs	**Fitness Original (SV,TV)**	**Melhor SV (SV,TV)**	**Melhor TV (SV,TV)**	**Melhor Trade-off (SV,TV)**
AGM	(0.2223,4)	(0.0307,4)	(0.0307,4)	(0.0307,4)
MM	(0.125,2)	(0.0266,4)	(0.0266,4)	(0.0266,4)
Bank	(0.25,2)	(0.1666,2)	(0.1666,2)	(0.1666,2)
BET	(0.5353,6)	(0.1333,8)	(0.0666,4)	(0.1333,8)

Analisando os resultados pode ser observado que:

Do ponto de vista de SV(pla): os valores desta métrica diminuíram após a otimização para todos os projetos de PLA devido às mudanças no número de componentes comuns como foi justificado no Experimento I.

Do ponto de vista de TV(pla): os resultados de TV(pla) em MM e BET aumentaram devido ao aumento do número de componentes variáveis. Para MM, no projeto original todas as variabilidades estavam concentradas em um só componente variável denominado MediaMgr e, após a otimização, o número de componentes variáveis aumentou porque as variabilidades foram distribuídas nos componentes MediaMgr e EntryMgr. No caso da BET, o número original de componentes variáveis era três (PassageiroMgr, ViacaoMgr e CartaoMgr), o qual aumentou depois da otimização para quatro (NumCartoesMgr, CartaoMgr, LimitePassagensMgr e PassageiroMgr).

Neste experimento também não foi possível chegar a uma conclusão sobre a correlação entre SV(pla) e TV(pla), pois como mencionado antes os valores de TV(pla) são os mesmos para cada solução obtida. Um fator que pode influenciar no valor de TV(pla), é se o projeto de PLA contiver componentes compostos, como pode ser visto na Equação 3.11. Os quatro projetos de PLA usados nos experimentos não contêm componentes compostos.

Lições aprendidas: Apesar de não poder atestar a existência de correlação entre as funções objetivo SV(pla) e TV(pla), foi observado que, para projetos de PLA sem componentes compostos, o aumento do número de componentes variáveis leva a valores mais altos de SV(pla) e TV(pla). Assim, parece suficiente selecionar uma dessas duas métricas como objetivo para o processo de otimização. Nesse sentido se o arquiteto quer priorizar a otimização da variabilidade da LPS e o projeto de PLA não contém componentes compostos, não faz sentido selecionar a métrica TV(pla). Neste caso, é melhor selecionar a métrica SV(pla) como objetivo para o processo de avaliação.

4.1.6 Experimento III

Objetivo: O experimento foi definido em vista de investigar a possível correlação entre SD(pla) e TV(pla). Como as funções objetivo estão baseadas em métricas que medem propriedade arquitetural *Variabilidade* (TV(pla)) e propriedade arquitetural *Similaridade* (SD(pla)), é importante conhecer o comportamento do número componentes comuns e do número componentes variáveis existentes como também foi mencionado no Experimento I.

Hipóteses
H_{sd-tv0} : Não existe correlação significativa entre SD(pla) e TV(pla);
H_{sd-tv1} : existe significativa correlação entre SD(pla) e TV(pla).

Resultados do teste de normalidade: Neste experimento pode-se observar que os valores de SD(pla) obtidos para AGM e MM têm distribuição normal. Considerando que os demais resultados são idênticos (Tabela - 4.10), não foi possível aplicar testes de normalidade.

Tabela 4.10: Resultados do Teste de Normalidade

PLA	Experimento III			
	SD p-value	Distribuição	TV p-value	Distribuição
AGM	0.07186	Normal	Valores idênticos	-
MM	0.2304	Normal	Valores idênticos	-
Bank	Valores idênticos	-	Valores idênticos	-
BET	Valores idênticos	-	Valores idênticos	-

Análise do comportamento dos dados: Como a mesma situação aconteceu com os resultados do TV(pla) no Experimento III para todos os projetos de PLA (Tabela - 4.10), o teste de correlação também não foi aplicado.

Discussão sobre os resultados: A Tabela - 4.11 mostra os resultados do valor de SD(pla) e TV(pla) antes de ser otimizado (coluna nomeada *Fitness* Original), assim como o melhor valor de SD(pla) (coluna nomeada Melhor SD), o melhor valor de TV(pla) (coluna nomeada Melhor TV) e o melhor *trade-off* entre SD(pla) e TV(pla) (coluna nomeada Melhor *Trade-off*), após a otimização.

Tabela 4.11: Características das PLAs para o Experimento III

PLAs	Fitness Original (SD,TV)	Melhor SD (SD,TV)	Melhor TV (SD,TV)	Melhor Trade-off (SD,TV)
AGM	(1.2857,4)	(1.0323,4)	(1.0323,4)	(1.0323,4)
MM	(1.1428,2)	(1.0274,4)	(1.0274,4)	(1.0274,4)
Bank	(1.3333,2)	(1.1251,2)	(1.1251,2)	(1.1251,2)
BET	(1.0566,6)	(1.06,6)	(1.0975,6)	(1.0975,6)

Do ponto de vista de SD(pla): com respeito os valores originais, diminuíram os valores de SD(pla) nas soluções obtidas, exceto para BET no qual o valor aumentou apesar de o número de componentes variáveis manter-se igual.
Do ponto de vista de TV(pla): aos resultados de TV(pla) em MM aumentaram devido ao aumento dos componentes variáveis onde as variabilidades estão distribuídas nos componentes MediaMgr e EntryMgr, considerando que originalmente se concentravam apenas no componente MediaMgr. Assim como no Experimento II, no Experimento III também não foi possível chegar a uma conclusão sobre a correlação existente entre TV(pla) e SD(pla).

Lições aprendidas: Apesar de TV(pla) avaliar o nível de variabilidade existente na PLA, é importante ter em conta que somente deve ser usada se a PLA em analise contém componentes compostos. Em caso de não existirem componentes compostos na PLA será melhor selecionar a função objetivo SV(pla) para avaliar o nível de variabilidade no processo de avaliação, como foi mencionado no experimento anterior.

4.1.7 Experimento IV

Objetivo: O experimento foi definido em vista de investigar a possível correlação entre RCC(pla) e ACOMP(pla), devido a que ambas funções objetivo estão baseadas em métricas que medem a propriedade arquitetural *Acoplamento*.

Hipóteses

$H_{rcc-acomp0}$: Não existe correlação significativa entre RCC(pla) e ACOMP(pla);

$H_{rcc-acomp1}$: existe significativa correlação entre RCC(pla) e ACOMP(pla).

Resultados do teste de normalidade: No Experimento IV, pode-se observar na Figura - 4.7 (que apresenta o Q-Q Plot de AGM, para as amostras de RCC(pla) e ACOMP(pla)) que os pontos não está perto da linha reta, o qual indica que os dados não são normalmente distribuídos, resultado compatível com o resultado obtido pelo teste de normalidade de Shapiro-Wilk mostrado na Tabela - 4.12, o mesmo comportamento acontece com BET. No caso do MM, na Figura - 4.8 pode-se observar que o Q-Q Plot para as amostras de RCC(pla) têm distribuição normal, no entanto para ACOMP(pla) seguem uma distribuição não normal; o teste de Shapiro-Wilk confirma os resultados (Tabela - 4.12). Como os resultados obtidos para Bank foram os mesmos, não foi possível aplicar teste estatístico nos dados da LPS Bank devido à falta de diversidade destes.

Tabela 4.12: Resultados do Teste de Normalidade

PLA	Experimento IV			
	RCC p-value	Distribuição	ACOMP p-value	Distribuição
AGM	0.001785	Não-Normal	0.01278	Não-Normal
MM	0.1399	Normal	0.01341	Não-Normal
Bank	Valores idênticos	-	Valores idênticos	-
BET	0.0014	Não-Normal	0.03256	Não-Normal

Análise do comportamento dos dados: No Experimento IV, o teste de correlação aplicado para AGM, mostra uma fraca correlação negativa (rho =-0.15), no entanto para MM os resultados apresentam uma forte correlação positiva (rho =0.54). No caso do Bank e da BET, não foi possível aplicar a análise de correlação porque os resultados de RCC(pla) assim como de ACOMP(pla) foram os mesmos para cada solução.

Discussão sobre os resultados: A Tabela - 4.13 mostra os resultados do valor de RCC(pla) e ACOMP(pla) antes e depois de ser otimizada.

Tabela 4.13: Características das PLAs para o Experimento IV

PLAs	Fitness Original (RCC,ACOMP)	Melhor RCC (RCC,ACOMP)	Melhor ACOMP (RCC,ACOMP)	Melhor Trade-off (RCC,ACOMP)
AGM	(28.0,66.0)	(0.0,70.0)	(0.0,68.0)	(0.0,70.0)
MM	(29.0,47.0)	(0.0,68.0)	(0.0,64.0)	(0.0,68.0)
Bank	(16.0,48.0)	(0.0,56.0)	(0.0,56.0)	(0.0,56.0)
BET	(159.0,363.0)	(0.0,384.0)	(0.0,382.0)	(0.0,382.0)

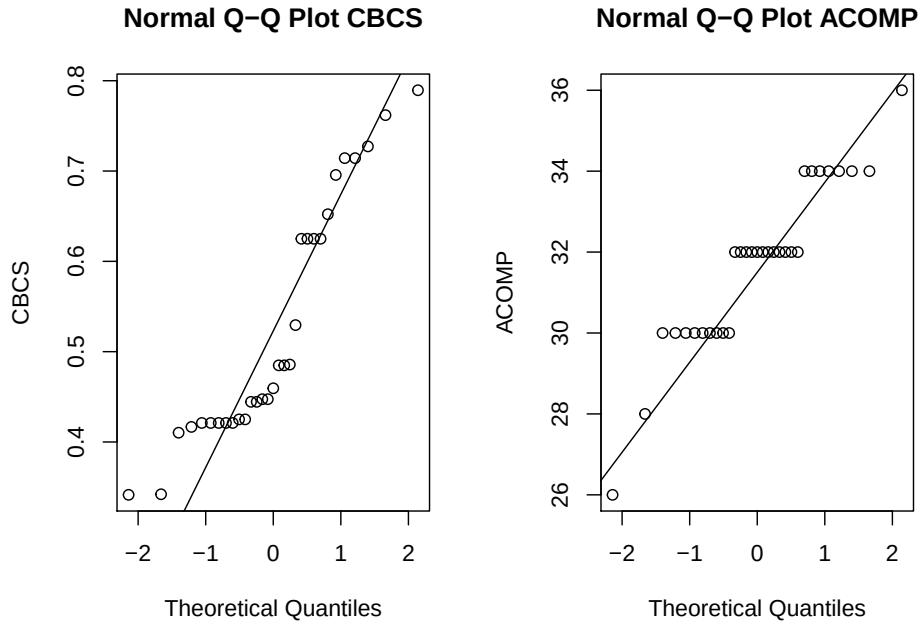

Figura 4.7: Q-Q Plots do Experimento IV - AGM

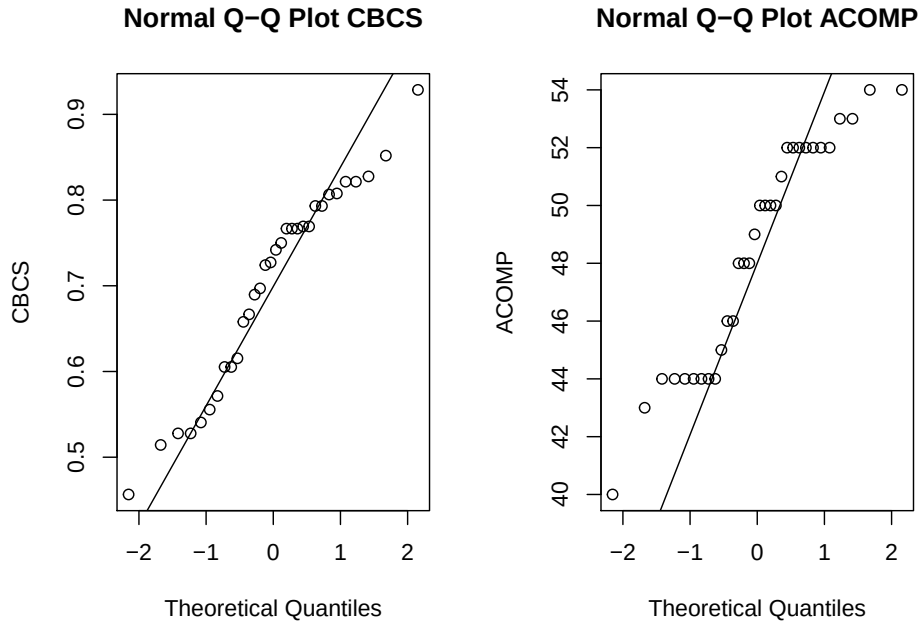

Figura 4.8: Q-Q Plots do Experimento IV - MM

Do ponto de vista de RCC(pla): com respeito os valores originais, pode-se observar como o valor de RCC(pla) para todos os projetos de PLA diminuiu ao máximo. Isto é

devido às alterações nos valores do número de interfaces após a otimização das PLAs. Por exemplo, na PLA AGM o número de interfaces original era 14 (Tabela - 4.3), no entanto, depois de ser otimizada o número de interfaces foi reduzido a zero, mas o número de classes aumentou de 30 (valor original) para 71 (valor otimizado). Esse mesmo comportamento aconteceu nos outros projetos de PLA.

Do ponto de vista de ACOMP(pla): pode-se observar que os valores de ACOMP(pla) aumentaram para todos os projetos de PLA. Esse comportamento é devido a que o número de componentes aumentou para todos os projetos de PLAs, portanto o relacionamento existente entre eles é também influenciado. No caso da PLA AGM por exemplo, o número de componentes antes da otimização era 9 aumentando notavelmente para 36 depois da otimização.
Neste experimento também não foi possível chegar a uma conclusão sobre a correlação existente entre RCC(pla) e ACOMP(pla) porque os resultados para MM e BET apontaram uma correlação positiva, mas para AGM o comportamento é diferente como é apresentado na Tabela - 4.14. Para melhorar as evidências sobre a correlação entre estas funções objetivo estudos com outros projetos de PLA devem ser realizados.

Lições aprendidas: Independentemente de RCC(pla) e ACOMP(pla) serem funções objetivo que permitem avaliar o nível de acoplamento existente na PLA, se o objetivo está focado nas interfaces é melhor selecionar a função objetivo RCC(pla) para diminuir o nível de acoplamento entre interfaces no processo de avaliação, e no caso de o foco estar centrado nos componentes pode ser escolhida ACOMP(pla) como função objetivo no processo de otimização.

Tabela 4.14: Testes de correlação dos Experimento IV

Experimento IV			
PLA	**Teste aplicado**	**p-value**	**Nível de correlação**
AGM	Spearman's	0.4232	-0.149503
MM	Pearson's	0.001576	0.5357674
BET	Spearman's	2.125e-15	0.6241649

4.2 Ameaças à validade

As ameaças à validade consideradas no estudo exploratório foram agrupadas em quatro categorias principais (Wohlin et al., 2000): interna, externa, de conclusão e de construção.

Validade Interna e externa: ameaças à validade interna e externa estão relacionadas ao conjunto de projetos de PLA otimizados. Para atenuar essa ameaça, usamos PLAs de domínios diferentes e com diferentes tamanhos. Três dos quatro projetos de PLA são acadêmicos. O quarto projeto de PLA (BET) é uma LPS real. Portanto, os resultados fornecem análises sobre a possível correlação existente entre as métricas avaliadas nos estudos, levando em consideração o contexto das PLAs usadas.

Validade de Conclusão: A principal ameaça para a validade de conclusão é o número de projetos de PLA. PLAs como AGM, MM e Bank são menores que BET. O número de PLAs existentes é reduzido porque, infelizmente, não é fácil encontrar projetos de PLA para realizar experimentos, portanto não foi possível generalizar os resultados. Estudos envolvendo diversos projetos de PLA devem ser realizados no futuro.

Validade de Construção: essa ameaça está relacionada com a configuração dos experimentos. A respeito das métricas usadas, a ameaça é garantida por sua validação anterior e aplicação bem sucedida em estudos anteriores (Zhang et al., 2008). Apesar de serem não-comercial, os projetos de PLA usados nos experimentos foram objeto dos estudos de outros pesquisadores, por exemplo, AGM (Clements et al., 2006; Contieri Junior et al., 2011; Marcolino et al., 2013), MM (Contieri Junior et al., 2011; Figueiredo et al., 2008; da Silva et al., 2011) BET (Donegan e Masiero, 2007; de Moraes et al., 2010) e Bank (Martinez et al., 2015). A adoção do mesmo tamanho da população e o mesmo número de gerações, independentemente do tamanho da PLA são outras ameaças. Valores diferentes para ambos os parâmetros poderiam acarretar resultados diferentes, especialmente para PLA maiores. Nesse sentido será necessário realizar mais estudos envolvendo diferentes projetos de PLA e diferentes parâmetros de ajuste.

4.3 Considerações finais

Neste capítulo, descreveu-se um estudo exploratório para investigar a possível correlação entre métricas relacionadas ao acoplamento, similaridade e variabilidade para a concepção de um projeto de PLA. Portanto foram realizados quatro experimentos envolvendo quatro projetos de PLA. De forma geral, é muito cedo para considerar estes resultados como definitivos. Os resultados obtidos não permitiram caracterizar a possível correlação entre as funções objetivo, apesar disso, foram aprendidas algumas lições sobre o uso destas métricas no contexto da otimização de projeto de PLA pela abordagem MOA4PLA. Essas lições representam uma importante contribuição, visto que fornecem uma visão mais ampla sobre as funções objetivo analisadas e sua utilização no contexto da otimização de

projeto de PLA, assim como, podem ser úteis para o arquiteto de LPS no momento da seleção das funções objetivo a serem otimizadas.

Como foi mencionado anteriormente, mais estudos exploratórios são necessários, incluindo a replicação dos experimentos previamente realizados. Além disso, é também necessário selecionar PLAs com componentes compostos, para ter evidências reais que provem que a função objetivo **TV(pla)** pode ser usada para medir o total de variabilidade existente em uma PLA.

5

Conclusão

O objetivo do presente trabalho foi expandir o modelo de avaliação da MOA4PLA com métricas para avaliar outras propriedades arquiteturais de projeto de PLA. Tendo em conta que a definição do modelo de avaliação para SBPD abrange a construção de funções objetivo baseadas em um conjunto de métricas, para dar cumprimento ao objetivo traçado primeiramente foi definido um modelo de qualidade para a MOA4PLA. O modelo de qualidade proposto está baseado em características de qualidade que contêm atributos de qualidade que são importantes para o projeto de PLA e que por sua vez incluem propriedades arquiteturais medidas por métricas.

Uma vez definido o modelo de qualidade foi realizado um mapeamento sistemático onde foram encontradas outras métricas ainda não incluídas no modelo de avaliação. A fim de explorar o comportamento dessas métricas foi realizada uma prova de conceito onde pode-se comprovar que as métricas poderiam ser utilizadas no modelo. Deste modo foram propostas seis novas funções objetivo baseadas nas métricas selecionadas, as quais foram implementadas na ferramenta OPLA-Tool. Com a inclusão das novas funções objetivo, o número de funções objetivo do modelo de avaliação da MOA4PLA aumentou, contendo um total 17 funções objetivo, representando um elevado número de objetivos a serem otimizados simultaneamente. Portanto, realizou-se um estudo exploratório em vista de analisar e investigar a possível correlação entre métricas selecionadas. Os resultados obtidos foram inconclusivos porque não foi possível definir o nível de correlação existente entre as métricas. Apesar disso, algumas lições aprendidas podem guiar o arquiteto da PLA na seleção de funções objetivo.

A seguir destacam-se as principais contribuições do presente trabalho e na Seção 5.2 são direcionados novos trabalhos a serem realizados.

5.1 Contribuições

As principais contribuições do presente trabalho são: (i) a proposta do modelo de qualidade para SBPD, baseado em características de qualidade que contêm atributos de qualidade e que inclui propriedades arquiteturais medidas por métricas; (ii) uso de métricas para a LPS, aplicadas por primeira vez no contexto de SBPD; e (iii) proposta de novas funções objetivo que permitem avaliar outras propriedades arquiteturais no modelo de avaliação da MOA4PLA. Cada uma das contribuições é apresentada a seguir.

- **Proposta do modelo de qualidade para SBPD**

  O modelo de qualidade proposto para o modelo de avaliação da MOA4PLA permite capturar as qualidades de projeto de PLA a avaliar. Nesse sentido, o modelo de qualidade define quais características de qualidade, atributos de qualidade, propriedades arquiteturais e métricas podem ser avaliadas pela MOA4PLA, na otimização de projeto de PLA. É essencial mencionar que o modelo de qualidade proposto segue um meta-modelo definido, onde as características de qualidade são compostas por atributos de qualidade que contêm propriedades arquiteturais que podem ser medidas por uma ou várias métricas como foi apresentado na Figura - 3.1. Tendo em conta o modelo de qualidade, foram classificadas todas as métricas das funções existentes, assim como, das novas funções objetivo propostas no modelo de avaliação da MOA4PLA, em vista de mapear propriedades arquiteturais que medem com atributos de qualidade. Além disso é importante especificar que até o presente trabalho não foi encontrado na literatura nenhum modelo de avaliação para SBPD.

- **Uso de métricas para a LPS**

  As métricas selecionadas permitem avaliar propriedade arquiteturais como ***Variabilidade, Similaridade*** e ***Acoplamento de variabilidade***, acrescentando novas propriedades arquiteturais a serem avaliadas no modelo. No caso das métricas WOCS e CBCS, é interessante ressaltar, que foram utilizadas pela primeira vez no contexto de SBPD e que fornecem outro ponto de vista a ter em conta quando se deseja avaliar ***Acoplamento*** e ***Tamanho***, respectivamente.

- **Proposta de novas funções objetivo para o modelo de avaliação da MOA4PLA**

  Foram propostas seis novas funções objetivo as quais foram incluídas na OPLA-Tool. As mesmas estão baseadas nas métricas selecionadas e que podem ser utilizadas pelo

arquiteto durante o processo de otimização. As novas funções objetivo, enriquecem o modelo de avaliação da MOA4PLA pois permitem avaliar outras propriedades arquiteturais que não eram avaliadas pelo modelo.

5.2 Trabalhos futuros

Para aprofundar os resultados obtidos no estudo exploratório, devem ser realizados novos experimentos com outros projetos de PLA para:

(i) Confirmar o comportamento de SD(pla) e SV(pla) sobre PLAs com características difusas;

(ii) Avaliar o impacto de TV(pla) em PLAs que contêm componentes compostos;

(iii) Aumentar o conhecimento sobre a correlação entre as métricas SD(pla), SV(pla) e TV(pla) no contexto de otimização de projeto de PLA.

Outros trabalhos que podem ser realizados a partir deste são:

(i) Alterar o meta-modelo da MOA4PLA em vista de poder usar a métrica *SCC*;

(ii) Realizar experimentos para investigar a correlação de *SCC* com outras métricas do modelo de avaliação e;

(iii) Realizar experimentos para investigar o uso dessas novas funções objetivo na otimização de projeto de PLA com o modelo original.

REFERÊNCIAS

Ahrens, D.; Frey, A.; Pfeiffer, A.; Bertram, T. Objective evaluation of software architectures in driver assistance systems. *Computer Science - Research and Development*, v. 28, n. 1, p. 23–43, 2013.
Disponível em `http://dx.doi.org/10.1007/s00450-011-0185-x`

Aldekoa, G.; Trujillo, S.; Sagardui, G.; Diaz, O. Quantifying maintainability in feature oriented product lines. In: *12th European Conference on Software Maintenance and Reengineering, 2008. CSMR 2008.*, 2008, p. 243–247.

Arcuri, A.; Briand, L. A hitchhiker's guide to statistical tests for assessing randomized algorithms in software engineering. *Software Testing, Verification and Reliability*, v. 24, n. 3, p. 219–250, 2014.

Bansiya, J.; Davis, C. G. A hierarchical model for object-oriented design quality assessment. *IEEE Trans. Softw. Eng.*, v. 28, n. 1, p. 4–17, 2002.
Disponível em `http://dx.doi.org/10.1109/32.979986`

Basili, V. R.; Selby, R. W.; Hutchens, D. H. Experimentation in software engineering. *IEEE Trans. Softw. Eng.*, v. 12, n. 7, p. 733–743, 1986.
Disponível em `http://dl.acm.org/citation.cfm?id=9775.9777`

Carvalho, F.; Meira, S. R. L.; Xavier, E.; Eulino, J. An embedded software component maturity model. In: *2009 Ninth International Conference on Quality Software*, 2009, p. 426–431.

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.

Clements, P.; Northrop, L. *Software product lines: Practices and patterns*. Addison-Wesley Longman Publishing Co., Inc., 2001.

Clements, P. C.; Jones, L. G.; McGregor, J. D.; Northrop, L. M. Getting there from here: A roadmap for software product line adoption. *Commun. ACM*, v. 49, n. 12, p. 33–36, 2006.

Coello, C. A.; Lamont, G. B.; Veldhuizen, D. A. V. *Evolutionary algorithms for solving multi-objective problems*, v. 5. Springer, 2007.

Colanzi, T. E.; Vergilio, S. R. Applying search based optimization to software product line architectures: Lessons learned. In: *Proceedings of the 4th Symposium on Search Based Software Engineering (SSBSE)*, 2012, p. 259–266.

Colanzi, T. E.; Vergilio, S. R.; Gimenes, I. M. S.; Oizumi, W. N. A search-based approach for software product line design. In: *Proceedings of the 18th International Software Product Line Conference (SPLC 2014)*, 2014, p. 237–241.

Contieri Junior, A. C.; de S. Gimenes, G. G. C. I. M.; Junior., E. A. O.; Colanzi, T. E.; Ferrari, S.; Masiero, P. C.; Garcia, A. F. Extending UML components to develop software product-line architectures: Lessons learned. In: *5th European Conference on Software Architecture (ECSA 2011)*, 2011, p. 130–138.

Deb, K.; Pratap, A.; Agarwal, S.; Meyarivan, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, v. 6, n. 2, p. 182–197, 2002.

Delgado, Y.; ; Colanzi, T. E. Towards the extension of an evaluation model for product line architecture design. *VII Workshop de Engenharia de Software Baseada em Busca (WESB)*, 2016.

Donegan, P. M.; Masiero, P. C. Design issues in a component-based software product line. In: *Proc. of Brazilian Symposium on Software Components, Architectures and Reuse (SBCARS)*, 2007, p. 3–16.

Dromey, R. G. A model for software product quality. *IEEE Trans. Softw. Eng.*, v. 21, n. 2, p. 146–162, 1995.
Disponível em `http://dx.doi.org/10.1109/32.345830`

Durillo, J. J.; Nebro, A. J.; Alba, E. The jmetal framework for multi-objective optimization: Design and architecture. In: *IEEE Congress on Evolutionary Computation*, 2010, p. 1–8.

Dursckil, R.; Spinolal, M.; Burnett, R.; Reinehr, S. Linhas de produto de software: riscos e vantagens de sua implantação. *VI Simpósio Internacional de Melhoria de Processos de Software*, 2004.

Edwards, G.; Seo, C.; Medvidovic, N. Model interpreter frameworks:a foundation for the analysis of domain-specific software architectures. *Journal of Universal Computer Science*, v. 14, n. 8, p. 1182–1206, 2008.
Disponível em `http://www.jucs.org/jucs_14_8/model_interpreter_frameworks_a`

Féderle, E. L.; do T. Ferreira; 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*, SPLC 15, New York, NY, USA: ACM, 2015, p. 370–373 (*SPLC 15*,).

Fenton, N.; Bieman, J. *Software metrics: A rigorous and practical approach*. 3 ed. CRC Press, 2014.

Figueiredo, E.; Cacho, N.; Sant'Anna, C.; Monteiro, M.; Kulesza, U.; Garcia, A.; Soares, S.; Ferrari, F.; Khan, S.; Castor Filho, F.; Dantas, F. Evolving software product lines with aspects: an empirical study on design stability. In: *Proceedings of the 30th International Conference on Software Engineering (ICSE'08)*, New York, NY, USA: ACM, 2008, p. 261–270.

Gomaa, H. *Software modeling and design: Uml, use cases, patterns, and software architectures*. Cambridge University Press, 2011.

Guizzo, G.; Colanzi, T. E.; Vergilio, S. R. A Pattern-Driven Mutation Operator for Search-Based Product Line Architecture Design. In: *Proceedings of the 6th Symposium of Search Based Software Engineering (SSBSE)*, Fortaleza: Springer, 2014.

Gurp, J. V.; Bosch, J.; Svahnberg, M. On the notion of variability in software product lines. In: *Proceedings of the Working IEEE/IFIP Conference on Software Architecture*, WICSA '01, Washington, DC, USA: IEEE Computer Society, 2001, p. 45– (*WICSA '01*,).
Disponível em `http://dx.doi.org/10.1109/WICSA.2001.948406`

Harman, M.; Jia, Y.; Krinke, J.; Langdon, W. B.; Petke, J.; Zhang, Y. Search based software engineering for software product line engineering: A survey and directions for future work. In: *Proc. of SPLC 2014*, 2014, p. 5–18.

HARMAN, M.; MANSOURI, S. A.; ZHANG, Y. *Search based software engineering: A comprehensive analysis and review of trends techniques and applications.* Relatório Técnico TR-09-03, Department of Computer Science, King's College London, 2009.
Disponível em `http://www.dcs.kcl.ac.uk/technical-reports/papers/TR-09-03.pdf`

HARMAN, M.; MANSOURI, S. A.; ZHANG, Y. Search-based software engineering: Trends, techniques and applications. *ACM Computing Surveys*, v. 45, n. 1, p. 11:1–11:61, 2012.
Disponível em `http://doi.acm.org/10.1145/2379776.2379787`

HAUKE, JAN; KOSSOWSKI, T. Comparison of values of pearson's and spearman's correlation coefficients on the same sets of data. *Quaestiones Geographicae*, v. 30, n. 2, 2011.
Disponível em `http://www.degruyter.com/view/j/quageo.2011.30.issue-2/v10117-011-0021-1/v10117-011-0021-1.xmlhttp://geoinfo.amu.edu.pl/qg/archives/2011/QG302{_}087-093.pdf`

HER, J. S.; KIM, J. H.; OH, S. H.; RHEW, S. Y.; KIM, S. D. A framework for evaluating reusability of core asset in product line engineering. *Information and Software Technology*, v. 49, n. 7, p. 740–760, 2007.
Disponível em `http://dx.doi.org/10.1016/j.infsof.2006.08.008`

HU, W.; LOEFFLER, T.; WEGENER, J. Quality model based on iso/iec 9126 for internal quality of matlab/simulink/stateflow models. In: *2012 IEEE International Conference on Industrial Technology*, 2012, p. 325–330.

HUMPHREY, W.; KELLNER, M. *Software process modeling: Principles of entity process models.* Relatório Técnico CMU/SEI-89-TR-002, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 1989.
Disponível em `http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=10859`

ISO/IEC Iso/iec 9126 - information technology - product quality - part1: Quality model. international organization for standardization - iso. [*On-line*], https://www.iso.org. Accessed on 10/05/2016.

ISO/IEC-25010 Iso/iec 25010 - systems and software engineering - systems and software quality requirements and evaluation (square): System and software quality

models. [*On-line*], https://www.iso.org/obp/ui/iso:std:iso-iec:25010:ed-1:v1:en. Accessed on 10/05/2016.

Jacobson, I. *Object-oriented software engineering: A use case driven approach*. Addison Wesley Longman Publishing Co., Inc., 2004.

Jeong, H. Y.; Kim, Y. H. A quality model of lightweight component for embedded system. In: *Frontiers of Manufacturing and Design Science II*, Trans Tech Publications, 2012, p. 4907–4911 (*Applied Mechanics and Materials*, v.121).

Kang, K. C.; Kim, S.; Lee, J.; Kim, K.; Shin, E.; Huh, M. Form: A feature-oriented reuse method with domain-specific reference architectures. *Annals of Software Engineering*, v. 5, p. 143–168, 1998.
Disponível em `http://dl.acm.org/citation.cfm?id=590631.590645`

Kitchenham, B.; Pickard, L.; Pfleeger, S. L. Case studies for method and tool evaluation. *IEEE Software*, v. 12, n. 4, p. 52–62, 1995.
Disponível em `http://dx.doi.org/10.1109/52.391832`

Knowles, J.; Corne, D. The pareto archived evolution strategy: a new baseline algorithm for pareto multiobjective optimisation. In: *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, 1999, p. 105 Vol. 1.

Linden, F. J. v. d.; Schmid, K.; Rommes, E. *Software product lines in action: The best industrial practice in product line engineering*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2007.

López, N.; Casallas, R.; van der Hoek, A. Issues in mapping change-based product line architectures to configuration management systems. In: *Proceedings of the 13th International Software Product Line Conference*, SPLC '09, Pittsburgh, PA, USA: Carnegie Mellon University, 2009, p. 21–30 (*SPLC '09*,).
Disponível em `http://dl.acm.org/citation.cfm?id=1753235.1753239`

Marcolino, A.; OliveiraJr, E.; Gimenes, I.; Conte, T. U. Towards validating complexity-based metrics for software product line architectures. In: *Proc. of Brazilian Symposium on Software Components, Architectures and Reuse (SBCARS)*, 2013, p. 69–79.

Martinez, J.; Ziadi, T.; Bissyandé, T. F.; Klein, J.; l. Traon, Y. Automating the extraction of model-based software product lines from model variants. In: *30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2015, p. 396–406.

Mendonça, M. *Efficient reasoning techniques for large scale feature models.* Tese de doutorado, University of Waterloo, Canadá, 2009.

Meyer, B. *Object-oriented software construction.* 1st ed. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1988.

Mitchell, M. *An introduction to genetic algorithms.* Cambridge, MA, USA: MIT Press, 1998.

de Moraes, A. L. S.; de C. Brito, R.; Contieri Junior, A. C.; Ramos, M. C.; Colanzi, T. E.; de S. Gimenes, I. M.; Masiero, P. C. Using aspects and the Spring framework to implement variabilities in a software product line. In: *Proceedings of the 29th International Conference of the Chilean Science Society (SCCC 2010)*, 2010, p. 71–80.

Oizumi, W.; Contieri, A.; Correia, G.; Colanzi, T.; Ferrari, S.; Gimenes, I.; OliveiraJr, E.; Garcia, A.; Masiero, P. On the proactive design of product-line architectures with aspects: An exploratory study. In: *2012 IEEE 36th Annual Computer Software and Applications Conference (COMPSAC)*, 2012, p. 273–278.

OliveiraJr, E. A.; Gimenes, I. M. S.; Maldonado, J. C.; Masiero, P. C.; Barroca, L. Systematic evaluation of software product line architectures. *Journal of Universal Computer Science*, v. 19, n. 1, p. 25–52, 2013.
Disponível em `http://www.jucs.org/jucs_19_1/systematic_evaluation_of_software`

Pohl, K.; Bockle, G.; Linden, F. J. v. d. *Software product line engineering: Foundations, principles and techniques.* Springer-Verlag New York, Inc., 2005.

Pressman, R. S. *Engenharia de software: Uma abordagem profissional.* Bookman, 780 p., 2011.

Quynh, P. T.; Thang, H. Q. Dynamic coupling metrics for service? oriented software. *International Journal of Computer Science and Engineering*, v. 1, 2009.

Ribeiro, H. B. G.; d. L. Meira, S. R.; d. Almeida, E. S.; Lucredio, D.; Alvaro, A.; Alves, V.; Garcia, V. C. An assessment on technologies for implementing core assets in service-oriented product lines. In: *2010 Fourth Brazilian Symposium on Software Components, Architectures and Reuse (SBCARS)*, 2010, p. 90–99.

Rocha, A.; Maldonado, J. C.; Weber, K. C. *Qualidade de software: Teoria e prática.* Prentice Hall, 303 p., 2001.

Räihä, O. Survey: A survey on search-based software design. *Computer Science Review*, v. 4, n. 4, p. 203–249, 2010.
Disponível em `http://dx.doi.org/10.1016/j.cosrev.2010.06.001`

Sant'Anna, C.; Garcia, A.; Chavez, C. Lucena, C.; von Staa, A. On the reuse and maintenance of aspect-oriented software: An assessment framework. *in Proceedings of the 7th Brazilian Symposium on Software Engineering*, 2003.

Sant'Anna, C. N. *On the modularity of aspect-oriented design: A concern-driven measurement approach.* Tese de Doutoramento, PUC-Rio, Rio de Janeiro/RJ, Brasil, 2008.

Santos, M. C. B.; Moschetta, W.; Colanzi, T. E.; OliveiraJr, E. A. Otimizando o projeto de arquitetura de linha de produto de software com muitos objetivos: um estudo exploratório. *VI Workshop de Engenharia de Software Baseada em Busca (WESB)*, 2015.

SEI Arcade Game Maker pedagogical product line. [*On-line*], http://www.sei.cmu.edu/productlines/ppl/. Accessed on 10/05/2016.

Shapiro, S. S.; Wilk, M. B. An analysis of variance test for normality (complete samples). *Biometrika*, p. 591–611, 1965.

Sharma, A.; Kumar, R.; Grover, P. S. Estimation of quality for software components: An empirical approach. *SIGSOFT Softw. Eng. Notes*, v. 33, n. 6, p. 1–10, 2008.
Disponível em `http://doi.acm.org/10.1145/1449603.1449613`

da Silva, B. C.; Sant'Anna, C.; Chavez, C. Concern-based cohesion as change proneness indicator: An initial empirical study. In: *Proceedings of the 2Nd International Workshop on Emerging Trends in Software Metrics*, WETSoM '11, New York, NY, USA: ACM, 2011, p. 52–58 (*WETSoM '11*,).
Disponível em `http://doi.acm.org/10.1145/1985374.1985387`

Simons, C.; Parmee, I. Elegant object-oriented software design via interactive, evolutionary computation. *IEEE Trans. on Systems, Man, and Cybernetics*, v. 42, n. 6, p. 1797 –1805, 2012.

Sommerville, I. *Engenharia de software*. 9 ed. Pearson, 298 p., 2011.

Souza, O. M. d. *Avaliação de modelos de software baseada em métricas internas e atributos de qualidade*. Dissertação de mestrado, Universidade Federal de São Carlos, 2015.

Torres, M.; Kulesza, U.; Sousa, M.; Batista, T.; Teixeira, L.; Borba, P.; Cirilo, E.; Lucena, C.; Braga, R.; Masiero, P. Assessment of product derivation tools in the evolution of software product lines: An empirical study. In: *Proceedings of the 2Nd International Workshop on Feature-Oriented Software Development*, FOSD '10, New York, NY, USA: ACM, 2010, p. 10–17 (*FOSD '10*,).
Disponível em `http://doi.acm.org/10.1145/1868688.1868691`

Wohlin, C.; Runeson, P.; Höst, M.; Ohlsson, M.; Regnell, B.; Wesslén, A. *Experimentation in software engineering: An introduction*. International Series in Engineering and Computer Science. Kluwer Academic, 2000.
Disponível em `https://books.google.co.uk/books?id=nG2UShV0wAEC`

Wüst, J. SDMetrics. [*On-line*], http://www.sdmetrics.com/. Accessed on 05/05/2016.
Disponível em `http://www.sdmetrics.com/`

Zhang, T.; Deng, L.; Wu, J.; Zhou, Q.; Ma, C. Some metrics for accessing quality of product line architecture. In: *2008 International Conference on Computer Science and Software Engineering*, 2008, p. 500–503.

Apêndice A

Neste apêndice são apresentadas todas as classes e métodos acrescentados ou mudados durante a implementação das novas funções objetivo definidas no Capítulo 3. Nesse sentido o presente apêndice está estruturado da seguinte forma: A Seção A.1 apresenta as alterações no componente OPLA-GUI a Seção A.2 apresenta as alterações no componente OPLA-Core e a Seção A.3 mostra o diagrama de classes, das classes alteradas e acrescentadas na OPLA-Tool.

A.1 Módulo OPLA-GUI

Tabela 1.1: Implementações realizadas no Pacote opla.gui2

Pacote: opla.gui2		
Classe: main.java		
Método	Tipo	**Observações**
initComponents()	Atualizado	Foram adicionados os check box associados as novas funções objetivo na tela de Configurações do experimento.
checkWocsClassActionPerformed(java.awt.event.ActionEvent evt)	Adicionado	Métodos definidos para visualizar os check box das novas funções objetivo.
checkWocsInterfaceActionPerformed(java.awt.event.ActionEvent evt)	Adicionado	
checkCbcsActionPerformed(java.awt.event.ActionEvent evt)	Adicionado	
checkSvcActionPerformed(java.awt.event.ActionEvent evt)	Adicionado	
checkSscActionPerformed(java.awt.event.ActionEvent evt)	Adicionado	
checkAvActionPerformed(java.awt.event.ActionEvent evt)	Adicionado	
comboMetricsItemStateChanged(java.awt.event.ItemEvent evt)	Atualizado	Foram adicionados os items assim como os elses if associados as novas funções objetivo .

Tabela 1.2: Implementações realizadas no Pacote OPLA-Gui

Pacote: db		
Classe: BestSolutionBySelectedFitness.java		
Método	Tipo	**Observações**
CalculateBestWocsC(String experimentId)	Adicionado	Métodos definidos para obter o melhor resultado de cada função objetivo
CalculateBestWocsI(String experimentId)	Adicionado	
CalculateBestCbcs(String experimentId)	Adicionado	
CalculateBestSvc(String experimentId)	Adicionado	
CalculateBestSsc(String experimentId)	Adicionado	
CalculateBestAv(String experimentId)	Adicionado	
Classe: Database.java		
Método	Tipo	**Observações**
getWocsclassMetricsForSolution(String idsolution,String experimentId)	Adicionado	Métodos definidos para obter o valor do idsolution de cada função objetivo.
getWocsinterfaceMetricsForSolution(String idsolution,String experimentId)	Adicionado	
getCbcsMetricsForSolution(String idsolution,String experimentId)	Adicionado	
getSvcMetricsForSolution(String idsolution,String experimentId)	Adicionado	
getSscMetricsForSolution(String idsolution,String experimentId)	Adicionado	
getAvMetricsForSolution(String idsolution,String experimentId)	Adicionado	
getAllWoclassMetricsForSolution(String idsolution,String experimentId)	Adicionado	Métodos definidos para obter todos os resultados das funções objetivo nas execuções.
getAllWocsinterfaceMetricsForSolution(String idsolution,String experimentId)	Adicionado	
getAllCbcsMetricsForSolution(String idsolution,String experimentId)	Adicionado	
getAllSvcMetricsForSolution(String idsolution,String experimentId)	Adicionado	
getAllSscMetricsForSolution(String idsolution,String experimentId)	Adicionado	
getAllAvMetricsForSolution(String idsolution,String experimentId)	Adicionado	

A.2 Módulo OPLA-Core

Tabela 1.3: Implementações realizadas no Pacote Database da OPLA-Core

Pacote: Database		
Classe: Result.java		
Método	Tipo	**Observações**
getMetrics(...)	Atualizado	Foram adicionados os if que dão tratamento para as novas funções objetivo
buildWocsclassMetrics(...)	Adicionado	Métodos definidos para guardar o resultado de cada função objetivo
buildWocsinterfaceMetrics(...)	Adicionado	
buildCbcsMetrics(...)	Adicionado	
buildSscMetrics(...)	Adicionado	
buildSVCMetrics(...)	Adicionado	
buildAvMetrics(...)	Adicionado	

Tabela 1.4: Implementações realizadas no Pacote jmetal.Problems da OPLA-Core

Classe: OPLA.java		
Método	Tipo	**Observações**
evaluate(Solution solution)	Atualizado	Foram adicionados os tratamentos case para as novas funções objetivo
evaluateWocsI(Architecture architecture)	Adicionado	Métodos definidos para obter os resultados de cada função objetivo
evaluateWocsC(Architecture architecture)	Adicionado	
evaluateCbcs(Architecture architecture)	Adicionado	
evaluateSsc(Architecture architecture)	Adicionado	
evaluateSvc(Architecture architecture)	Adicionado	
evaluateAv(Architecture architecture)	Adicionado	

Tabela 1.5: Implementações realizadas no Pacote jmetal.metrics da OPLA-Core

Classe: MetricsEvaluation.java		
Método	Tipo	**Observações**
evaluateWocsInterface(Architecture architecture)	Adicionado	Métodos definidos para instanciar os resultados de cada função objetivo
evaluateWocsClass(Architecture architecture)	Adicionado	
evaluateCbcs(Architecture architecture)	Adicionado	
evaluateSsc(Architecture architecture)	Adicionado	
evaluateSvc(Architecture architecture)	Adicionado	
evaluateAv(Architecture architecture)	Adicionado	

Tabela 1.6: Implementação do Pacote jmetal.newPlasMetrcis da OPLA-Core

Class	**Descrição**
$WOCS_Class.java$	Implementa a função objetivo CS(pla) desde o ponto de vista de classe.
$WOCS_Inteface.java$	Implementa a função objetivo CS(pla) desde o ponto de vista de interface.
$CBCS.java$	Implementa a função objetivo RCC(pla).
$SSC.java$	Implementa a função objetivo SD(pla).
$SVC.java$	Implementa a função objetivo SV(pla).
$AV.java$	Implementa a função objetivo TV(pla).

Tabela 1.7: Novas classes implementadas no Pacote Metrics da OPLA-Core

Class	**Descrição**
$WOCS_Class.java$	Define set e get da função objetivo CS(pla) desde o ponto de vista de classe.
$WOCS_Inteface.java$	Define set e get da função objetivo CS(pla) desde o ponto de vista de interface.
$CBCS.java$	Define set e get da função objetivo RCC(pla).
$SSC.java$	Define set e get da função objetivo SD(pla).
$SVC.java$	Define set e get da função objetivo SV(pla).
$AV.java$	Define set e get da função objetivo TV(pla).

Tabela 1.8: Novas classes implementadas no Pacote jmetal.persistence da OPLA-Core

Class	**Descrição**
$WOCSclassPersistence.java$	Implementa a persistência dos dados da função objetivo CS(pla) desde o punto de vista classes na base de dados.
$WOCSintefacePersistence.java$	Implementa a persistência dos dados da função objetivo CS(pla) desde o punto de vista interface na base de dados.
$CbcsPersistence.java$	Implementa a persistência dos dados da função objetivo RCC(pla) na base de dados.
$SscPersistence.java$	Implementa a persistência dos dados da função objetivo SD(pla) na base de dados.
$SvcPersistence.java$	Implementa a persistência dos dados da função objetivo SV(pla) na base de dados.
$AvPersistence.java$	Implementa a persistência dos dados da função objetivo TV(pla) na base de dados.

A.3 Diagrama de classes das classes alteradas e acrescentadas na implementação das funções objetivo

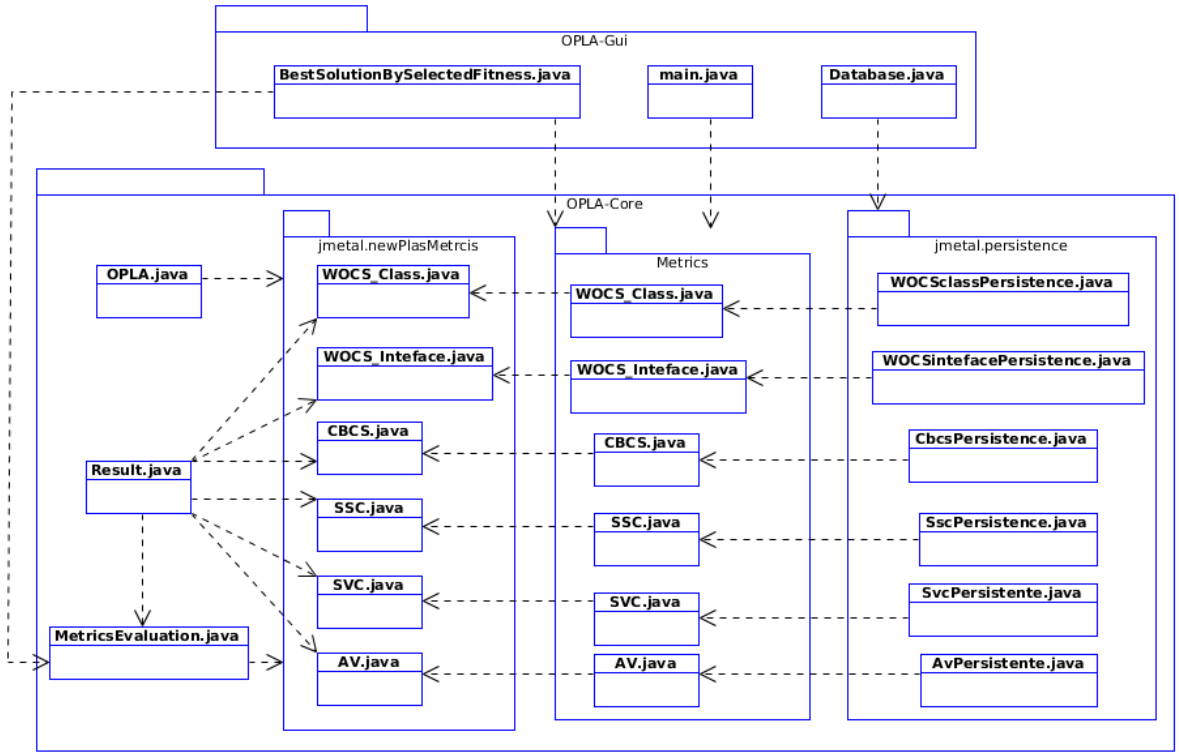

Figura 1.1: Diagrama de classes das classes alteradas e acrescentadas na OPLA-Tool

Apêndice B

ARTIGO PUBLICADO NO WESB 2016

Towards the Extension of an Evaluation Model for Product Line Architecture Design

Yenisei D. Verdecia, Thelma E. Colanzi

[1]Department of Informatics – State University of Maringa (UEM)
Caixa Postal 15.064 – 91.501-970 – Maringa – PR – Brazil

yni6.slim@gmail.com, thelma@din.uem.br

Abstract. *Product Line Architecture (PLA) is the most important artifact for Software Product Lines (SPL) Engineering. Studies about PLA design evaluation often consider a set of metrics that provide indicators about different factors such as design stability, feature modularization and PLA extensibility. A systematic and automated approach that uses search-based algorithms to optimize a PLA design, named MOA4PLA (Multi-Objective Approach for Product -Line Architecture Design), was recently proposed. MOA4PLA aims at optimizing basic design principles, feature modularization, design elegance and SPL extensibility. Currently, the evaluation model of MOA4PLA has one objective function for each one of those optimization goals. However, there are other design properties important to PLA design, what motivates a study for extending the MOA4PLA evaluation model. In this sense, the objective of this paper is to enrich the evaluation model with metrics related to other properties. Our study was guided by the Goal-Question-Metrics approach followed by a Proof of Concept performed with four PLA designs. Preliminary results provide evidences that the identified metrics can contribute to the evaluation model extension.*

1. Introduction

Software Product Line (SPL) approach is based on the systematic reuse of software artifacts, through the holding of common features and management of variability between products, which are established under a same architecture [Pohl et al. 2005]. A Product Line Architecture (PLA) is the main artifact of an SPL and provides the design that is common to all the products of the SPL. Hence it describes all the mandatory and varying features of its SPL domain [Contieri et al. 2011].

PLA design is an important activity throughout the life cycle of the SPL [Oizumi et al. 2012]. It is even more complex than traditional software architecting because, in order to maximize the reuse, some quality requirements, such as modularity, stability and reusability, represent an important role [Colanzi and Vergilio 2016]. The more extensible are the architecture components, the greater is their reusability rate [OliveiraJr et al. 2013]. Therefore, it is important to analyze the architecture to predict their quality before the products generation, in order to identify potential risks and verify that the quality requirements are being treated in the design [Pohl et al. 2005].

Studies on structural evaluation of PLA often consider a set of metrics to provide indicators on different properties [Oizumi et al. 2012]. Find the best balance of priorities (trade-off) between the metrics as well as choose which should be prioritized is a difficult task that needs great effort and time. In this point of view, the Search Based

Software Engineering (SBSE) [Harman et al. 2012], provides techniques to obtain and evaluate possible alternatives for a PLA design. As the problems of Software Engineering can be formulated as optimization problems to be solved through search-based techniques [Colanzi et al. 2014]. Approaches based on Multi-Objective Evolutionary Algorithms (MOEAs) have been applied to optimize the software design or optimize the order of refactoring to be applied to the design [Räihä 2010], whereas they allow finding a better solution in a set of possible solutions. In this view, the multi-objective optimization approach called Multi-Objective Approach for Product-Line Architecture Design (MOA4PLA) [Colanzi et al. 2014] was recently proposed.

The goal of MOA4PLA is to identify automatically the best alternatives for a PLA design, optimizing basic design principles, feature modularization, design elegance and SPL extensibility. Currently, the maximum number of goals of MOA4PLA is limited to four objective functions available, one for each MOA4PLA goal. However, there are other design properties relevant to PLA design, which were not included in the current evaluation model, such as PLA complexity, adaptability and similarity levels of a PLA design. This fact motivates a study for extending the MOA4PLA evaluation model.

In this sense, the objective of this paper is to enrich the evaluation model with metrics related to other properties. To this end, we proceeded the following methodology: 1) the Goal-Question-Metrics (GQM) approach was applied in the definition phase of the study to identify whether the complexity of developed components as well as the level of coupling, the level of similarity and adaptability of a PLA, are goals that can be measured in the evaluation model of MOA4PLA, and 2) a Proof of Concept (PoC) was carried out to explore the behavior of the metrics identified in the GQM. A set of preliminary results provide evidences that the identified metrics can contribute to the evaluation model extension and provide some perspectives to be considered. So, the main contribution of the work is the identification and selection of metric to enrich the MOA4PLA evaluation model.

This paper is organized as follows. Section 2 briefly describes MOA4PLA and its evaluation model. Section 3 presents the GQM defined in order to find new goals to be evaluated in the approach. Section 4 explores the behavior of identifying metrics, presenting the results of measurement as well as the perspectives identified. Finally, Section 5 concludes the paper and directs further work.

2. Characterizing MOA4PLA

MOA4PLA is a multi-objective optimization approach proposed to identify automatically the best alternatives of PLA design, regardless of the search algorithm adopted. It contributes to the decrease the effort of architects during the design and evaluation of a PLA [Colanzi et al. 2014]. MOA4PLA encompasses four main activities: (1) Construction of the PLA Representation, (2) Definition of the Evaluation Model, (3) Multi-objective Optimization and (4) Transformation and Selection. The approach receives as input the PLA design, modeled in a UML class diagram, where each architectural element is associated with the feature(s) that it realizes. The output is the set of PLA alternative designs that have the best trade-off among the defined objectives to be optimized.

The Multi-objective Optimization activity of MOA4PLA includes conventional mutation operators: MoveMethod, MoveAttribute, AddClass, MoveOperation and Ad-

dComponent [Colanzi et al. 2014]. AddClass moves a method or an attribute to a new class. MoveOperation moves operations between interfaces. AddComponent creates a new interface to a new component, and then moves an operation to this interface. That activity also encompasses mutation and crossover operators to improve feature modularization: Feature-Driven Mutation [Colanzi et al. 2014] and Feature-Driven Crossover [Colanzi and Vergilio 2016]. The first one aims at modularizing a feature tangled with others in a component, considering that a feature usually is solved by a group of architectural elements [Colanzi et al. 2014]. The second one selects a feature at random and then creates children by swapping the architectural elements (classes, interfaces, operations, etc.) that realize the selected feature. In this way, children might combine groups of architectural elements that better modularize some feature(s) inherited from their parents [Colanzi and Vergilio 2016].

The evaluation model of MOA4PLA is composed of four objective functions, related to the main goals of the approach, to evaluate every solution generated in the optimization process. The objective functions include metrics to measure basic design principles, features modularization, SPL extensibility and design elegance [Colanzi et al. 2014]. The objective functions are briefly described below:

CM(pla)[1]: Aims at providing indicators on basic design principles including cohesion, coupling and size. This objective function is composed of an aggregation of various metrics extracted from [Wüst 2013].

FM(pla)[1]: Aims at measuring the features modularization. It aggregates several feature-driven metrics proposed in [Sant'Anna 2008] to measure feature-based cohesion, feature diffusion and feature interaction over architectural elements.

Ext(pla): Aims at indicating the degree of extensibility of SPL, where the extensibility is measured by means of PLA abstraction [OliveiraJr et al. 2013].

Eleg(pla): Aims at providing indicators about the elegance of a object-oriented software design. It is measured by the sum of the metrics defined in [Simons and Parmee 2012].

3. Identifying metrics to expand the evaluation model

Given the present work motivation, we define our evaluation goal using the Goal-Question-Metric (GQM) approach [Basili and Rombach 1988]. Based on the GQM template [Basili and Rombach 1988], the **goal** is presented as follows:

Analyze *PLAs*
For the purpose to *expand the evaluation model of MOA4PLA*
With respect to *architectural design properties*
From the point of view of *software architects*
In the context of *SPL.*

The *questions* and *metrics* that must be answered and interpreted are:

Q1: *How is the complexity of components designed in a PLA?* The architecture has a direct impact on the software complexity [Pohl et al. 2005]. The more dependencies

[1]Currently on review because added metrics with different grandeurs [Santos et al. 2015].

between the components that make up the PLA, the greater their complexity, because a high number of dependencies makes any changes cause major impacts on the related components. In this regard the following metric helps answer the question, in spite of it was originally applied in the context of Service-Oriented SPL:

M1: Weighted Operations per Component or Service (WOCS) [Ribeiro et al. 2010]. WOCS will measure the complexity of the service or component, in terms of its operations (methods) that will be required by other services or components. Hence, consider a Component or Service C with operations O1, ... , On. Let c1, ... , cn be the complexity of the operations. Then: WOCS = c1 +...+ cn. The metric indicates that operations with many formal parameters are more likely to be complex than operations in which require less parameter. In this sense, the complexity of an operation ck can be defined as follows: $ck = \alpha + 1$ where αk denotes the number of formal parameters of operations (Ok) [Ribeiro et al. 2010].

Q2: *The components contained in the PLA has low coupling?* One of the expectations of software architects is to know the level of coupling between the components inside of a PLA, because if that component have least possible dependencies, it will be easier to develop, test, and maintain. The following metric was also applied in the context of Service-Oriented SPL, but it is interesting for SPL in general and satisfies the question:

M2: Coupling Between Components or Services (CBCS) [Ribeiro et al. 2010]. This metric is calculated based on the number of relationships between one service A, for example, and other services in an application. It can be mathematically described in Equation 1 as follows:

$$CBCS = \sum_{i \neq j = 1 K_n}^{n} AiBj \tag{1}$$

In which: n is the number of services in application; $AiBj$ = 0 if Ai does not connect to Bj and $AiBj$ = 1 if Ai connects to Bj [Ribeiro et al. 2010].

For a service A, the larger the value of CBCS metric, the tighter the relationship with other services is. In other words, service A depends much more on others [Quynh 2009]. CBCS goal can be focused in the context of Product Line, based on the number of relationships between interfaces.

Q3: *What is the level of similarity existing in the PLAs?* The identification of common components and variables of a PLA, gives insight into the level of reuse that it can exist in a PLA. To know the level of similarity existing in the PLAs the following metric answers that question:

M3: Structure Similarity Coefficient (SSC) method [Zhang et al. 2008], is proposed to measure similarity of PLA as Equation 2 following:

$$SSC = \frac{|Cc|}{|Cc| + |Cv|} \tag{2}$$

In Equation 2, Cc is number of common components in PLA and Cv is number of variable components in PLA. According to Equation 2, PLA has more common components and less variable components, leads to architectures of products that are more similar. Then the SPL will gain more benefits of reuse [Zhang et al. 2008].

Q4: *What is the level of adaptability of a PLA?* The variability is constantly evolving, with the extension of scope and currency exchange in applications [Peng et al. 2011]. The greatest variability, more possibilities to adapt the product to customer's tastes. In this regard the following metrics respond to the question:

M4: Strong Coupling Coefficient (SCC) method [Zhang et al. 2008], is proposed to measure strong coupling of variability as follows in Equation 3:

$$SCC = 1 - \frac{|IVP|}{|VP|} \tag{3}$$

In Equation 3 , VP is the variability point number and IVP is the number of independent variability points of PLA, which have no dependence relations with others. PLA has more variability points, more number of product architectures can be derived by instantiating the PLA. However, PLA is more complex and difficult to be designed [Zhang et al. 2008].

M5: Structure Variability Coefficient (SVC) method [Zhang et al. 2008] is defined to measure structure variability of PLA in Equation 4. Cc is number of common components in PLA and Cv is number of variable components in PLA . PLA has more variable components, and there is more structure variability. SVC also can be used to measure structure variability of compound components [Zhang et al. 2008].

$$SVC = \frac{|Cv|}{|Cc| + |Cv|} \tag{4}$$

M6: Architecture variability (AV) [Zhang et al. 2008], is total variability of PLA. AV is defined as following Equation 5:

$$AV = |Cv| + \sum_{i} AV(Ci) \tag{5}$$

Where: $|Cv|$: The number of variable components in PLA, $AV(Ci)$: Variability of interior component Ci . If Ci is compound component, then $AV(Ci)$ can be calculated with Equation 5. If C is basic component and has interior variability, $AV(Ci)$=1, else $AV(Ci)$=0 [Zhang et al. 2008].

To find the metrics, that respond the questions defined in GQM, we used the following search engines for our reviews: ACM Digital Library, IEEE Xplore, Elsevier Scopus, and SpringerLink. Our specific research question was: *What is the state of art the metrics to Software Product Line?*. The selection of metrics is based on the following criteria: (1) it is possible to obtain the information needed to apply the metrics of a UML class diagram, because the evaluation model refers to a representation of the PLA which is based on a metamodel that instantiates a UML class diagram, and (2) the metrics were not implemented in the current evaluation model. In general, the identified metrics in GQM, except CBCS, measure architecture design properties that not evaluated in evaluation model. Even so, CBCS measures different coupling type that the coupling measures currently applied in MOA4PLA.

4. Validating the identified metrics for the context of MOA4PLA

A Proof of Concept (PoC) is a demonstration aimed at verifying that certain concepts or theories has potential of being used. A prototype is designed to determine feasibility, but does not represent the final deliverable, because PoC studies seek to determine whether

or not a product idea will be viable [Gallant 2006]. For the present work, the objective of the PoC is twofold. Firstly, validate the application of metrics in the context of PLA design, aiming at its use in the evaluation model of MOA4PLA. Secondly, identify new perspectives to be analyzed in approach. For this end, four PLA designs were used in the PoC, whose main information are presented in Table 1. The metrics identified in the GQM were calculated for each PLA design[2].

PLAs selected are based on components and follow the layered style. They are briefly described below. Arcade Game Maker (AGM) is a SPL created by the Software Engineering Institute (SEI) that encompasses three arcade games: Brickles, Bowling, and Pong [SEI 2016]. System Banking (Bank) supports the managing of banking systems [Gomaa 2011]. BET supports the bus city transport management. It offers features such as the use of an electronic card for transport payment; automatic toll gate opening; and unified traveling payment. It was conceived based on three existing transportation products of Brazilian cities [Donegan and Masiero 2007]. Mobile Media (MM) is a SPL composed of features that handle music, videos, and photo for portable devices. It provides support for managing different types of media [Contieri et al. 2011].

Table 1. PLA Information

ALP	# Components	# Interfaces	# Classes	# Features	#Variabilities
AGM	9	14	30	11	5
Bank	4	5	25	16	3
BET	56	30	115	18	8
MM	8	15	14	14	7

Following section address the behavior analysis of the obtained results from the metrics application and the perspectives identified during the analysis.

4.1. Behavior Analysis of the Results

This section presents the behavior analysis of each metric centering on the objectives of the metrics identified in GQM (Section 3).

M1: The value of WOCS [Ribeiro et al. 2010] was calculated for each package of the PLAs selected. Due to the number of interfaces in each package of PLAs, the results obtained from WOCS were extensive. Even so, was analyzed each results of WOCS. To illustrate the behavior of WOCS, one package of the PLAs AGM and BET with the same amount of interfaces was selected, as can be observed in Table 2.

Table 2. Results of WOCS in GameBoardCtrl and LinhaMgr packages

PLAs					
AGM			**BET**		
Package	Interface	WOCS	Package	Interface	WOCS
GameBoardCtrl	ICheckScore	9	LinhaMgr	IAtualizarCorrida	1
	IGameBoardData	2		IRegistrarArrecadacao	2
	ISaveScore	19		ILinhaMgt	23

As can be seen in the Table 2, the value of WOCS for each interface of GameBoardCtrl package of AGM, the values of complexity of interfaces are not high. The

[2]The entire results of the measurement as well as the PLA designs can be obtained in https://github.com/yni6delgado/Test

same behavior occurred for the LinhaMgr package of BET. But through this analysis, we observe that had classes with a high number of operations in the PLAs. In this sense, it is interesting to analyze the number of operations by classes, with the objective of reducing them, therefore emerged two new views defined in Section 4.2.

M2: CBCS metric [Ribeiro et al. 2010] can be focused to the context of Product Line in general by analyzing the number of relationships between an interface A and other interfaces of the PLA. Taking into account this focus, CBCS was calculated for each PLA. According to [Ribeiro et al. 2010], a CBCS value greater than 9 is considered a too high value, in our context we follow the same criteria. Overall the results show a view of the behavior of metric, because they allow to see the level of coupling among the interfaces of the PLAs. Values greater than 9 where achieved only for BET in the following interfaces: IProcessarViagem (10), IRegistrarArrecadacao (10), ILinhaMgt (12), IPassageiroMgt (12), ICartaoMgt (27), IViacaoMgt (11) and ITempoMgt (10).

M3: SSC [Zhang et al. 2008] was calculated for all PLAs. The values obtained of SSC range from 0 to 1, are shown in Table 3. It is possible to notice that the number of common components in PLAs is greater than the number of variable components, what is good for maximizing the PLA reusability.

M4, M5, M6: For the variability analysis, were selected the following metrics: SCC, SVC and AV [Zhang et al. 2008]. Each of the metrics allows analyzing different views: 1) coupling between points of variability (SCC), 2) structure variability (SVC), and 3) architecture variability (AV).

SCC metric allows knowing if the points of variability in PLAs have dependency relationships with other points of variability, and in this way can be analyzed the level of coupling between them. However, the value of SCC for all PLAs is 1 because there is no dependency relationship between points of variability in those designs (see Table 3). Furthermore, this metric cannot be calculated actually in MOA4PLA because the metamodel defined for the approach does not provide this level of relationship. In this sense, to apply such metric in the evaluation model, it is necessary to adapt the metamodel used in MOA4PLA.

The values of SVC range from 0 to 1, the results of SVC shown in Table 3, indicate that the number of variable components inside of PLAs is down.

AV metric is set to understand the architecture variability, through the identification of variability within compound and basic components. Analyzing the behavior of AV in the PLAs, taking into account Equation 5, it is noticed that, there are not compound components inside the selected PLAs. Table 3 shows the results of AV.

In general, we conclude that the values of metrics identified in GQM are consistent with the PLA design properties which they are supposed to measure. With regards to the feasibility of the metrics application in the evaluation model of MOA4PLA, we also analyzed if the metrics values could be influenced by the search operators of MOA4PLA during the optimization process. Table 4 shows what mutation operators can influence on the metrics values during the optimization of PLA design solutions. Thus, that is other evidence of the feasibility application of the identified metric on the evaluation model, as every metric value can be changed during the optimization process.

Table 3. Results of metrics

PLAs	#Cc	#Cv	Metrics			
			SSC	SCC	SVC	AV
AGM	7	2	0.77	1	0.22	4
Bank	3	1	0.75	1	0.25	2
BET	53	3	0.95	1	0.54	6
MobileMedia	7	1	0.88	1	0.13	2

Table 4. Relationship metric X mutation operator

Metrics	Search Operator
WOCS	Move Operation Mutation
CBCS	All operators
SSC	Feature-driven Mutation and Add Package
SCC	Feature-driven Mutation
SVC	Feature-driven Mutation and Add Package
AV	Feature-driven Mutation

4.2. Perspectives

Looking at the preliminary results and considering the PLA design optimization, we can highlight the following perspectives when using the identified metrics to extend the MOA4PLA evaluation model:

P_1: The intention will be to decrease the values of CBCS in search to soften the level of coupling between interfaces. In this sense the results of CBCS to BET will be a point of observation to be taken into account when the metric is implemented in MOA4PLA.

P_2: Analyzing the behavior of WOCS, could be seen that had classes with a high number of operations in the PLAs. In this sense classes with a high number of operations are to be analyzed from the following viewpoints [Chidamber and Kemerer 1994]: 1) the larger numbers of methods in a class, the greater the potential impact on children, since children will inherit all the methods defined in the class and 2) classes with large numbers of methods limit the possibility of reuse. Therefore the intention will be to decrease the values WOCS from the point of view of interfaces and class.

P_3: A SSC value close to 1 represents a greater number of common components within the PLA. In this sense the intention will be to maximize the SSC value taking into account the existing variability in PLA.

P_4: In the context of SCC, decreasing the level of coupling between the existing variability points a PLA, provides greater flexibility in time for the instantiating PLA. Accordingly, increasing the number of IVP from a PLA promotes the level of reusability of the PLA.

P_5: A SVC value close to 1, demonstrates the presence of a greater number of variable components in the PLAs. This metric can be applied to composite components of the PLA, with the intention of increasing the number of variable components within the PLA.

P_6: The greater variability within the compound component, the PLA is more complex and difficult to be designed. In this sense to decrease the number of composite components increases the level of reusability of the PLA.

P_7: Analysis results of SVC and SSC point out that they are conflicting metrics.

5. Conclusions and future works

In this paper we presented a study of metrics guided by GQM and based on PoC to identify and validate metrics to extend the evaluation model of MOA4PLA. Through the GQM were identified metrics to provide indicators about the complexity and coupling among components of the PLA design as well as the level of similarity and adaptability of the PLA to be included as new goals to be optimized by the approach.

The PoC points out the metrics values are consistent with the PLA design properties under measurement and that the values can be suffer the action of the search operators during the optimization process, leading to an optimization of the informed PLA design.

As future works we intend: 1) to propose new objective functions for the evaluation model, 2) to perform experiments to investigate the possible correlation between the proposed objective functions and between them and the objective functions existing in the current evaluation model and 3) perform empirical studies to optimize PLA designs according to the new goals of MOA4PLA.

6. Acknowledgment

We would like to thank CNPq for financial support.

References

Basili, V. R. and Rombach, H. (1988). The tame project: Towards improvement-oriented software environments. *IEEE Transactions on Software Engineering*, 14(6):758–773.

Chidamber, S. R. and Kemerer, C. F. (1994). A metrics suite for object oriented design. *IEEE Transactions on Software Engineering*, 20(6):476–493.

Colanzi, T. E. and Vergilio, S. R. (2016). A feature-driven crossover operator for multi-objective and evolutionary optimization of product line architectures. *Journal of Systems and Software*. In press.

Colanzi, T. E., Vergilio, S. R., Gimenes, I. M. S., and Oizumi, W. N. (2014). A search-based approach for software product line design. In *Proceedings of the 18th International Software Product Line Conference (SPLC 2014)*, volume 1, pages 237–241.

Contieri, Jr., A. C., Correia, G. G., Colanzi, T. E., Gimenes, I. M. S., OliveiraJr., E. A., Ferrari, S., Masiero, P. C., and Garcia, A. F. (2011). Extending uml components to develop software product-line architectures: Lessons learned. In *Proceedings of the 5th European Conference on Software Architecture*, ECSA'11, pages 130–138, Berlin, Heidelberg. Springer-Verlag.

Donegan, P. M. and Masiero, P. C. (2007). Design issues in a component-based software product line. In *Proc. of Brazilian Symposium on Software Components, Architectures and Reuse (SBCARS)*, pages 3–16.

Gallant, M. (2006). An ethnography of communication approach to mobile product testing. *Personal Ubiquitous Comput.*, 10(5):325–332.

Gomaa, H. (2011). *Software modeling and design: UML, use cases, patterns, and software architectures*. Cambridge University Press.

Harman, M., Mansouri, S. A., and Zhang, Y. (2012). Search-based software engineering: Trends, techniques and applications. *ACM Computing Surveys*, 45(1):11:1–11:61.

Oizumi, W., Contieri, A., Correia, G., Colanzi, T., Ferrari, S., Gimenes, I., OliveiraJr, E., Garcia, A., and Masiero, P. (2012). On the proactive design of product-line architectures with aspects: An exploratory study. In *2012 IEEE 36th Annual Computer Software and Applications Conference (COMPSAC)*, pages 273–278.

OliveiraJr, E. A., Gimenes, I. M. S., Maldonado, J. C., Masiero, P. C., and Barroca, L. (2013). Systematic evaluation of software product line architectures. *Journal of Universal Computer Science*, 19(1):25–52.

Peng, X., Yu, Y., and Zhao, W. (2011). Analyzing evolution of variability in a software product line: From contexts and requirements to features. *Information and Software Technology*, 53(7):707 – 721.

Pohl, K., Bockle, G., and Linden, F. J. v. d. (2005). *Software Product Line Engineering: Foundations, Principles and Techniques*. Springer-Verlag New York, Inc.

Quynh, P. T.; Thang, H. Q. (2009). Dynamic coupling metrics for service – oriented software. *International Journal of Computer Science and Engineering*, 1.

Ribeiro, H. B. G., d. L. Meira, S. R., d. Almeida, E. S., Lucredio, D., Alvaro, A., Alves, V., and Garcia, V. C. (2010). An assessment on technologies for implementing core assets in service-oriented product lines. In *2010 Fourth Brazilian Symposium on Software Components, Architectures and Reuse (SBCARS)*, pages 90–99.

Räihä, O. (2010). Survey: A survey on search-based software design. *Computer Science Review*, 4(4):203–249.

Sant'Anna, C. N. (2008). *On the Modularity of Aspect-Oriented Design: A Concern-Driven Measurement Approach*. PhD thesis, PUC-Rio, Rio de Janeiro/RJ, Brasil.

Santos, M. C. B., Moschetta, W., Colanzi, T. E., and OliveiraJr, E. A. (2015). Otimizando o projeto de arquitetura de linha de produto de software com muitos objetivos: um estudo exploratório. *VI Workshop de Engenharia de Software Baseada em Busca (WESB)*.

SEI (2016). Arcade Game Maker pedagogical product line. http://www.sei.cmu.edu/productlines/ppl/. Accessed on 10/05/2016.

Simons, C. and Parmee, I. (2012). Elegant object-oriented software design via interactive, evolutionary computation. *IEEE Transactions on Systems, Man, and Cybernetics*, 42(6):1797 –1805.

Wüst, J. (2013). SDMetrics. http://www.sdmetrics.com/. Accessed on 05/05/2016.

Zhang, T., Deng, L., Wu, J., Zhou, Q., and Ma, C. (2008). Some metrics for accessing quality of product line architecture. In *2008 International Conference on Computer Science and Software Engineering*, volume 2, pages 500–503.