UNIVERSIDADE ESTADUAL DO NORTE DO PARANÁ

***CAMPUS* LUIZ MENEGHEL - CENTRO DE CIÊNCIAS TECNOLÓGICAS**

SISTEMAS DE INFORMAÇÃO

WESLLEY AUGUSTO DE ALMEIDA

UM ESTUDO SOBRE REUTILIZAÇÃO DE SOFTWARE UTILIZANDO LINHA DE PRODUTO DE SOFTWARE E DESENVOLVIMENTO DIRIGIDO A MODELOS

Bandeirantes

2017

WESLLEY AUGUSTO DE ALMEIDA

UM ESTUDO SOBRE REUTILIZAÇÃO DE SOFTWARE UTILIZANDO LINHA DE PRODUTO DE SOFTWARE E DESENVOLVIMENTO DIRIGIDO A MODELOS

Trabalho de Conclusão de Curso submetido à Universidade Estadual do Norte do Paraná, como requisito parcial para obtenção do grau de Bacharel em Sistemas de Informação.

Orientador: Profº. Me. Carlos Eduardo Ribeiro

Bandeirantes

2017

WESLLEY AUGUSTO DE ALMEIDA

UM ESTUDO SOBRE REUTILIZAÇÃO DE SOFTWARE UTILIZANDO LINHA DE PRODUTO DE SOFTWARE E DESENVOLVIMENTO DIRIGIDO A MODELOS

Trabalho de Conclusão de Curso submetido à Universidade Estadual do Norte do Paraná, como requisito parcial para obtenção do grau de Bacharel em Sistemas de Informação.

COMISSÃO EXAMINADORA

_____

Profº. Me. Carlos Eduardo Ribeiro
UENP – Campus Luiz Meneghel

_____

Profº.Dr. André Luis Andrade Menolli
UENP - Campus Luiz Meneghel

_____

Profº. Me. Fábio de Sordi Junior
UENP – Campus Luiz Meneghel

Bandeirantes, 16 de junho de 2017

RESUMO

O desenvolvimento de sistemas de informação sempre foi conhecido como um processo caro e lento, a busca constante para aumentar a produtividade no desenvolvimento de software, reaproveitando o máximo possível das experiências de projetos passados evitando a duplicação do esforço, e reduzindo significativamente custos e tempo de desenvolvimento, encontrou na engenharia de reuso a possibilidade de identificar características genéricas de produtos abrangentes a um contexto, e a partir dessas características essências possa gerar um subproduto mais específico. Novos conceitos e abordagens vêm sendo propostos pela disciplina de Engenharia de Software, para facilitar a construção e reutilização de softwares, a interação das abordagens desenvolvimento dirigido a modelo e linha de produto de software mostra-se como uma abordagem que contribui para melhoria no processo de desenvolvimento de software, sendo interessante analisar a pratica da utilização que envolve os conceitos propostos por estas abordagens nos contextos acadêmicos e organizacional.

Palavras-chave: Reutilização de Software, Linha de Produto de Software, Desenvolvimento Dirigido por Modelos, Arquitetura Dirigida a Modelos.

Abstract

The development of information systems has always been known as an expensive and slow process, the constant quest to increase productivity in software development, reusing as much of experiences the experiences of the past projects as possible, avoiding the effort duplication, and reducing costs and time of development significantly. This process found in the reuse engineering the possibility of identify generic characteristics of products encompassing to a context and from these essential characteristics it can generate a more specific by-product. New concepts and approaches have been proposed by the Software Engineering discipline, to facilitate the construction and reuse of software, the integration of the approaches development directed to the model and software product line is shown as an approach that contributes to improvement in the process of Development, being interesting to analyze the practice of the use that involves the concepts proposed by these approaches in the academic and organizational contexts

Keywords: Software Reuse, Software Product Line, Model Driven Development, Model-Driven Architecture.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIAÇÕES

CIM Computation Independent Model / Modelo Independente de Computação

DSL Domain-Specific Language / Linguagem Específica de Domínio

EA Engenharia de Aplicação

ED Engenharia de Domínio

FAST Family-oriented abstraction, specification and translation

FODA Análise de Domínio Orientada a Features (do inlglês Fearure Oriented Domain Analysis)

LPS Linha de Produto de Software

MDA Model-Driven Architecture / Arquitetura Orientada a Modelos

MDD Model-Driven Development / Desenvolvimento Orientado a Modelos

MDE Model-Driven Engineering / Engenharia Orientada a Modelos

MD* Acrônimo que abrange todas as abordagens de desenvolvimento de software orientadas a modelos

MOF Meta-Object Facility / Infraestrutura de MetaObjetos

OMG Object Management Group / Grupo de Gerenciamento de Objetos

PIM Platform Independent Model / Modelo Independente de Plataforma

PLUS do inlglês *Product Line UML Base Software Engineering*

PSM Platform Specific Model / Modelo Específico de Plataforma

UML Unified Modeling Language / Linguagem de Modelagem Unificada

XMI XML Metadata Interchange / Intercâmbio de Metadados em XML

Sumário

1. INTRODUÇÃO

A Engenharia de Software tem como objetivo melhorar cada vez mais as várias tecnologias e ferramentas para o desenvolvimento de software, com o intuito de diminuir os custos, prazos e aumentar a qualidade do software.

À medida que as indústrias de software aumentam a capacidade de produção, cresce também a demanda por sistemas mais complexos e de maior qualidade. Neste contexto, tem-se na engenharia de reuso uma forma de reagir a estas mudanças.

Uma alternativa que vem ganhando força é o uso de Desenvolvimento Dirigido a Modelos, trata se de um novo paradigma de desenvolvimento de software em que os principais artefatos para criação de software são modelos, esta abordagem consiste em transformações que convertem modelos em código-fonte, assim a partir do reuso de modelos podem ser gerados novos produtos de software.

Linha de produto de software também tem se mostrado como uma abordagem importante e viável para o reuso de software, pois não se trata apenas de um reuso em nível de código, mais sim de uma reutilização com o objetivo de fornecer processo, técnicas e ferramentas que permitam o gerenciamento das similaridades e variabilidades que compõem uma família de produtos de software, pertencentes a um mesmo domínio.

Para Paludo (2016) apud (MAGALHÃES et al., 2011), adotar individualmente a abordagem engenharia dirigida a modelos ou linhas de produto de software, já proporciona benefícios significativos para o processo de desenvolvimento de software. Ao combinar a integração entre as duas abordagens ocorre uma melhoria no processo de desenvolvimento.

1.1 OBJETIVO GERAL

O objetivo desse trabalho propõe uma análise comparativa sobre reuso de software, utilizando as abordagens Desenvolvimento Dirigido a Modelos e Linha de Produto de Software em um contexto acadêmico, e reuso de software utilizando a abordagens Desenvolvimento Dirigido a Modelos e Linha de Produto de Software em um contexto organizacional.

1.2 OBJETIVOS ESPECÍFICOS

1) Analisar a prática da utilização dos principais conceitos que envolvem as abordagens de desenvolvimento dirigido a modelos e Linha de Produto de Software no contexto acadêmico.

2) Analisar a prática da utilização dos principais conceitos que envolvem as abordagens de desenvolvimento dirigido a modelos e Linha de Produto de Software no contexto organizacional.

3) Apresentar o estudo comparativo da utilização dos principais conceitos das abordagens desenvolvimento dirigido a modelos e Linha de Produto de Software no contexto acadêmico e no contexto organizacional.

1.3 JUSTIFICATIVA

Dentre as muitas vantagens que o MDD propõe, a reutilização de software se apresenta de forma notável para o desenvolvimento de software, tendo em vista que a partir de um nível elevado de abstração e transformações entre modelos novos modelos compatíveis para plataformas específicas podem ser obtidos sem que haja a necessidade de um novo processo de construção de modelos.

Para Paludo apud (MAGALHÃES et al., 2011), quando se utiliza a abordagem dirigida a Modelos ou a abordagem de linhas de produto de software, individualmente já proporciona benefícios para o processo de desenvolvimento de software, quando ocorre a integração das duas abordagens contribui para melhoria no processo de desenvolvimento de software.

Desta forma é importante analisar a prática da utilização integrada das abordagens linha de produto de software e desenvolvimento dirigido a modelos tanto a nível acadêmico quanto a nível organizacional.

1.4 METODOLOGIA

Nesta pesquisa a abordagem do problema é qualitativa, o presente trabalho caracteriza-se como pesquisa exploratória. Uma pesquisa bibliográfica foi realizada buscando trabalhos acadêmicos direcionados a integração das abordagens desenvolvimento dirigido a modelos e linha de produto de software. Através deles são analisados aspectos referentes a pratica da utilização dos principais conceitos que envolvem cada abordagem.

1.5 ORGANIZAÇÃO DO TRABALHO

Esta seção descreve como o trabalho foi estruturado e organizado. No capítulo 2 é apresentada a fundamentação teórica que aborda os seguintes temas: reutilização de software, linha de produto de software, engenharia dirigida a modelos, desenvolvimento dirigido a modelos, arquitetura dirigida a modelos, integração das abordagens linha de produto de software e desenvolvimento dirigido a modelo. No capítulo 3 é apresentada a análise comparativa comparando os resultados obtidos. No capítulo 4 é apresentada a conclusão.

2. FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são apresentados alguns temas que contribuíram para o aprofundamento e embasamento dos conceitos necessários ao desenvolvimento desta pesquisa. Os primeiros temas abordados são reutilização de software, engenharia dirigida a modelos e desenvolvimento dirigido a modelos. Na sequência é abordado o tema arquitetura dirigida a modelos e linha de produto de software. Por fim, é apresentado o tema integração das abordagens linha de produto de software e desenvolvimento dirigido a modelo.

2.1 REUTILIZAÇÃO DE SOFTWARE

A reutilização de software é uma das áreas da engenharia de software que propõem por meio de técnicas e ferramentas reaproveitar de forma geral ou parcial artefatos utilizados em projetos anteriores para serem implantados em novos projetos.

De acordo com Krueger (1992), reutilização de software é o procedimento de criação de software com base em software já efetivo, assim não sendo necessário construir desde o início.

De acordo com D'Souza e Wills (1999), um artefato de reuso é um item do trabalho capaz de ser usado em mais de um projeto.

O reuso de software se fundamenta na utilização de conceitos, produtos ou soluções antecipadamente ordenadas ou obtidas para concepção de um novo software, visando acréscimo na qualidade e produtividade, impedindo a duplicação do empenho e reutilizando ao máximo de um sistema desenvolvido anteriormente. Aparentemente simples, essa idealização não é assim tão fácil de ser praticada, pois, deve ser realizada de maneira ordenada e controlada.

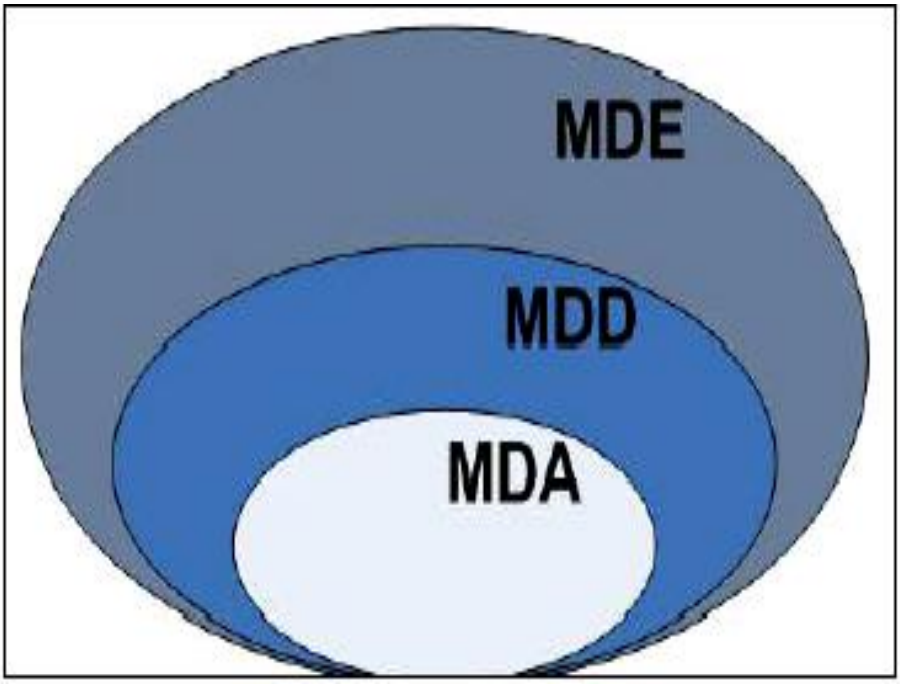
Adotar uma ferramenta ou tecnologia não é satisfatório para que os aditamentos da reutilização sejam adquiridos em sua máxima amplitude, sendo indispensáveis distintos fatores, como uma adequada administração organizacional e infraestrutura focada à reutilização (RINE;SONNEMANN, 1998).

Diferenciar o desenvolvimento "para e com" reutilização é a questão essencial do enfoque de engenharia de software conhecida como Linha de Produtos de Software (CLEMENTS; NORTHROP, 2002;) Linha de Produtos de Software será referida no tópico 2.5.

2.2 ENGENHARIA DIRIGIDA A MODELOS

A definição de MDE expande o conceito de Desenvolvimento Orientado por Modelos MDD, que engloba a definição da Arquitetura Orientada por Modelos MDA (OMG, 2016), conforme exibição da Figura abaixo.

Figura 1 – Abordagens Dirigidas a Modelos.



Fonte: (DEMIR, 2006).

2.2.1 Principais conceitos da engenharia dirigida a modelos

Para Zambiasi (2010), o principal conceito da MDE é o modelo, que pode ser definido como uma abstração de um sistema criado para um objetivo específico.

Ainda de acordo com Zambiasi (2010), um segundo conceito muito importante na MDE é o meta-modelo, que pode ser determinado de forma simples como sendo um modelo de modelo. Além do modelo e meta-modelo, um terceiro conceito deve ser considerado, o meta-meta-modelo. Existe ainda um último conceito fundamental que é a transformação de modelo que se apoiam em tecnologias computacionais para transformá-los em sistemas executáveis.

2.2.2 Metamodelagem

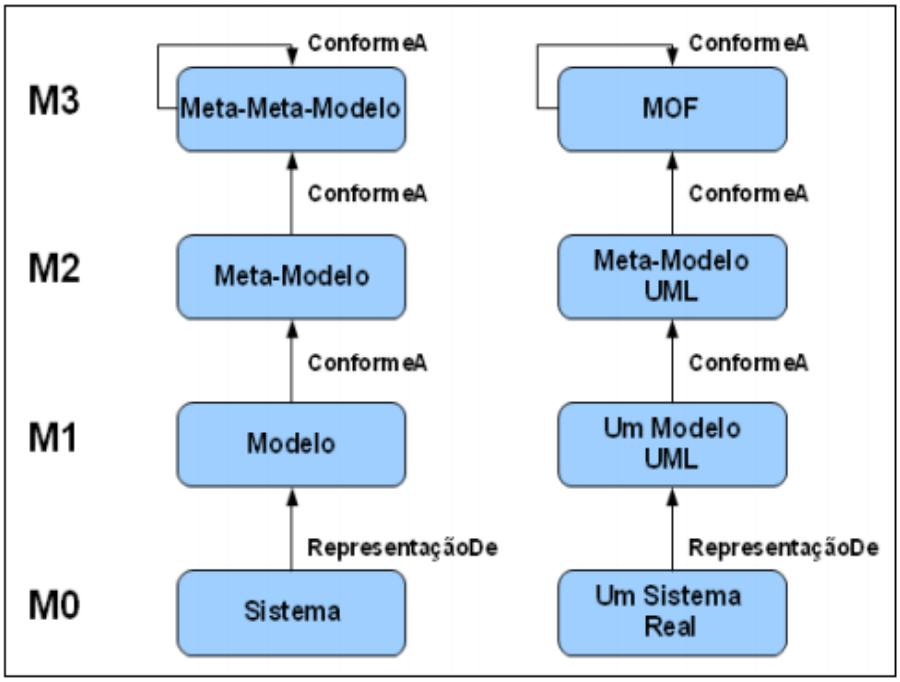
O metamodelo consiste em um modelo que define a linguagem de modelagem utilizada por outro modelo. De acordo com Bézivin (2005), os metamodelos descrevem os diversos tipos de elementos contidos no modelo e a forma como eles são relacionados e restringidos.

Lucrédio (2009) define a arquitetura de meta modelagem em quatro níveis:

O primeiro nível (M0) representa os dados. O segundo nível (M1) representa os metadados, ou modelo, que coresponde aos dados que descrevem os dados. O terceiro nível (M2) é o metamodelo, utilizado para a definição de modelos. O quarto nível (M3) é utilizado para definir metamodelos, ou seja, um meta-meta-modelo.

Ainda de açodo com Lucrédio (2009), a base da MDA é o MOF (Meta-Object Facility) (OMG, 2006b), o meta-metamodelo que serve de base para a definição de todas as linguagens de modelagem. O MOF consiste em um padrão orientado a objetos e possibilita a definição de classes com atributos e relacionamentos, o MOF é semelhante ao diagrama de classes da UML. Também apresenta uma interface padrão para acesso aos dados dos modelos, proporcionando métodos para manipulação dos dados e dos metadados, é devido ao MOF que as ferramentas de modelagem e transformação podem trabalhar em conjunto.

Figura 2 – Hierarquia de Modelos.



Fonte: (ZAMBIASI, 2010).

2.2.3 Transformação

Na visão da MDE os modelos são artefatos principais para o desenvolvimento de software, e a partir de tecnologias computacionais específicas os desenvolvedores podem transformá-los em sistemas executáveis.

A transformação de modelos é a geração automática de um modelo destino a partir de um modelo origem utilizando um conjunto de regras (MAIA. *et al* apud KLEPPE *et al*. 2006).

2.3 DESENVOLVIMENTO DIRIGIDO A MODELOS

O Desenvolvimento Dirigido por Modelos (Model Driven Development - MDD) é uma abordagem para desenvolvimento de software, que tem como foco principal elevar o nível de abstração do problema a ser modelado, para construir modelos, a partir dos quais são gerados o código e outros artefatos.

Esses modelos de software são tipicamente expressos a partir de linguagens de modelagens, podendo ser utilizada como por exemplo a UML *Unified Modeling Language*, controlada pela (*Object Management Group* -OMG), não se trata de uma metodologia, mais sim de uma linguagem para construção de diagramas que especificam e documentam modelos de sistemas de software, inicialmente foi criada para desenvolvimento de software Orientado à Objeto, e atualmente é utilizada também pela MDA – *Model Driven Architecture*, como linguagem para criar modelos, para serem transformados automaticamente em código fonte, por meio do uso de ferramentas case.

O MDD considera que no desenvolvimento do software a modelagem do problema é muito mais significativa do que o desenvolvimento do código fonte. Desse modo, amortiza a importância do código e as cautelas ficam mais centralizadas no que verdadeiramente importa: construir uma aplicação final que trabalhe de acordo com as condições especificadas e fornecidas pelo cliente e usuários (COSTA *et al*., 2008).

No Desenvolvimento Dirigido por Modelos, a construção do código é executada por meio de ferramentas específicas que concretizam a geração automática da estrutura e de itens do código, facilitando o desenvolvimento do software (COSTA, *et al*., 2008).

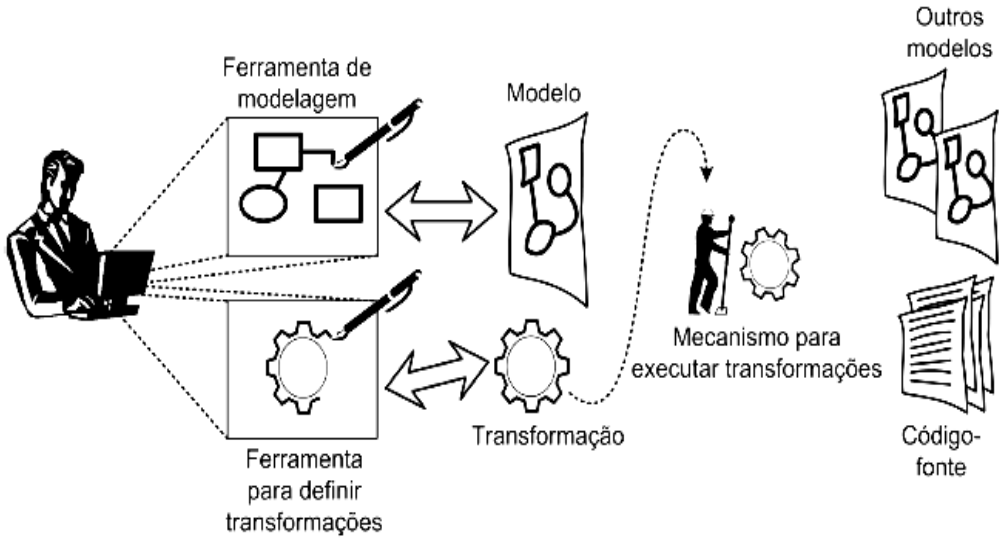
Os elementos principais do MDD são identificados por Booch et al. (2005) como sendo:

• Uso de um nível maior de abstração em relação a outras metodologias. Tanto para a especificação do problema, como para a especificação da solução correspondente;

• Uso maior de mecanismos de automatização por meio de mecanismos computacionais para realizar as fases de análise, projeto e implementação;

• Uso de padrões propostos e/ou adotados pela indústria para facilitar a comunicação entre seus usufrutuários e também melhorar a interoperabilidade entre diferentes aplicações e produtos.

Figura 3 – Principais Elementos do MDD.



Fonte: (LUCRÉDIO, 2009).

2.3.1 Ferramentas para apoiar o desenvolvimento dirigido a modelos

No contexto do desenvolvimento dirigido a modelos, é necessária a utilização de ferramentas para apoiar o desenvolvimento de aplicações, neste contexto existem tanto ferramentas livres como AndroMDA, quanto ferramentas comerciais como ArcStyler e OptimalJ.

2.3.1.1 ArcStyler

A ArcStyler (ARCSTYLER, 2006) é uma ferramenta comercial que utiliza a UML como linguagem de modelagem ela possui apenas um modelo, conhecido

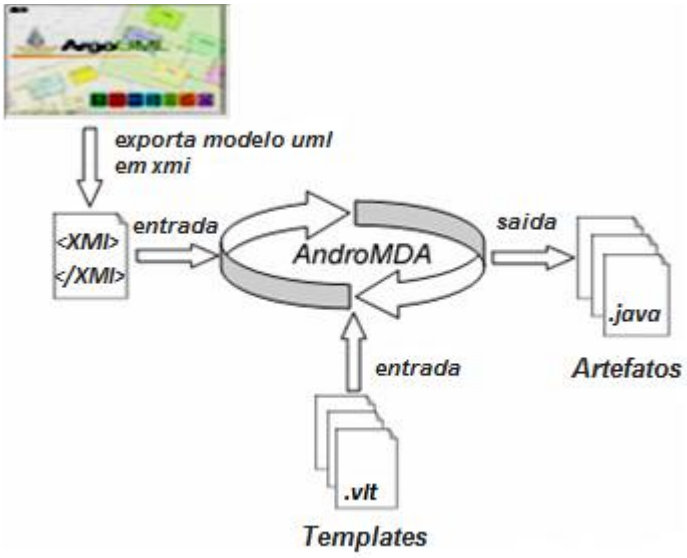
como modelo de domínio utilizado como entrada para as transformações e geração direta de código (CALIARI 2007).

Ainda segundo Caliari (2007), a ferramenta ArcStyler realiza somente a transformação do modelo independente de plataforma para o código ou para outro modelo, deixando de lado conceitos como registro de transformação ou modelo independente de computação.

2.3.1.2 AndroMDA

O AndroMDA ferramenta de código aberto é um gerador de código que recebe um modelo UML como entrada, em formato XMI, e gera um código-fonte como saída (COSTA 2008) *apud*, (ANDROMDA, 2008). Esses modelos utilizados pela ferramenta AndroMDA devem ser independentes de plataforma, portanto não deve ser inserido no modelo características que sejam específicas de alguma linguagem ou plataforma.

Figura 4 - Funcionamento do AndroMDA.



Fonte: COSTA (2008).

2.3.1.3 OptimalJ

O OptimalJ é uma ferramenta comercial que utiliza um subconjunto da UML como linguagem de modelagem e o MOF como meta-meta-modelagem. A OptimalJ divide o desenvolvimento do sistemas em três etapas; modelo de domínio, modelo de aplicação e modelo de código. O modelo de domínio define a funcionalidade e a estrutura, sem detalhes específicos de tecnologia, o modelo de aplicação descreve a

funcionalidade baseado em uma determinada tecnologia e o modelo de código que prove uma implementação do sistema descrito nos modelos anteriores (CALIARI 2007).

2.3.2 Transformação de Artefatos de Software

De acordo com Corrêa *et al*. (2011), transformações é o método utilizado para obter o refinamento entre os modelos, uma transformação pode ser compreendida como uma função que recebe um ou mais modelos como entrada, e levando em consideração um conjunto de regras ou instruções, gera outros artefatos. Embora a possibilidade das transformações serem executadas manualmente, o propósito é realizar as transformações de maneira automática ou pelo menos semi-automática, com auxílio do desenvolvedor. Assim, as transformações automatizam as tarefas que os desenvolvedores deveriam realizar manualmente, reduzindo o tempo de desenvolvimento e evitando a inserção de defeitos.

2.3.3 Transformações de modelo

As transformações de modelos podem ser classificadas em dois tipos, transformação modelo-modelo e transformação modelo-texto. A transformação modelo-modelo é a geração de um modelo a partir de outro modelo, a transformação modelo-texto é a geração de artefatos de software que são considerados textuais, como, por exemplo, código fonte.

De acordo com Corrêa (2011), o processo de transformação modelo-modelo utiliza o CIM e especificações de transformação para gerar um PIM. Depois se aplica outra transformação modelo-modelo onde utiliza o PIM e outras especificações de transformações direcionadas para uma plataforma específica para gerar o PSM. Por ultimo o PSM é usado para transformação modelo-texto para gerar o código fonte.

2.4 ARQUITETURA DIRIGIDA A MODELOS

Arquitetura Dirigida a Modelos (*Model Driven Architecture* – MDA) destaca a possibilidade de abordar o paradigma MDD por meio de duas maneiras, quando se deseja uma modelagem de propósito geral ou com propósito específico, em ambos os casos propõem elevar o nível de abstração, criando modelos para ser transformado em código executável.

A MDA divide a complexidade do sistema em três tipos de modelos: primeiro o CIM, que recai sobre os requisitos do sistema, não considerando sua estrutura ou processamento. O segundo o PIM, mostra a lógica do negócio do sistema fora do contexto de uma plataforma específica. No terceiro PSM são relacionados ao sistema para uma plataforma específica (OLIVEIRA, 2012).

CIM - O Modelo Independente de computação é uma visão do sistema a partir de uma perspectiva independente de seus detalhes computacionais, algumas vezes ele é chamado de modelo de domínio, este modelo não revela a estrutura pela qual o sistema é construído. (MAIA, 2006).

Para Lucrédio (2009), a transformação entre um modelo independente de computação e um modelo independente de plataforma é menos passível de automação, pois envolve mais decisões e maiores possibilidades de interpretação dos requisitos. Este modelo está mais próximo do especialista do domínio.

PIM - Modelo de alto nível de abstração que descreve o sistema, independente da plataforma que será usada na implementação assim um único modelo é necessário quando se pretende variar de uma plataforma para outra, contribuindo de forma significativa para a portabilidade, pois o mesmo PIM pode ser transformado em vários PSM de diferentes plataformas. Para Lucrédio (2009), o PIM é um modelo conceitual do sistema, de forma que o mesmo modelo possa ser reaproveitado em diferentes plataformas.

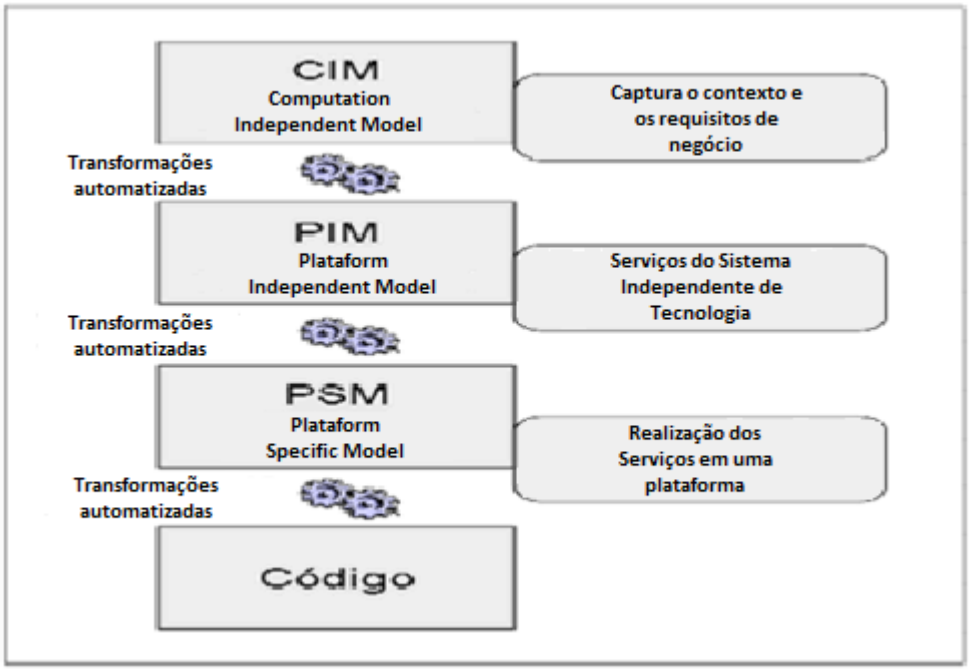
De acordo com Paludo (2016), os modelos independentes de plataforma representam uma visão da aplicação sem estar associada diretamente com as características de uma plataforma que irá abrigar a aplicação, assim e possível ter um modelo que seja adequado para diversas plataformas de tipos semelhantes (OMG, 2003). Esses modelos não mostram detalhes da tecnologia que será utilizada nas etapas posteriores do desenvolvimento.

PSM - Modelo construído do ponto de vista de uma plataforma específica, ou seja, descreve um sistema com o conhecimento em um tipo particular de plataforma. Segundo Oliveira (2012), neste modelo detalhes da implementação são descritos em uma plataforma junto com as informações do PMI. O PSM é gerado através de transformações de modelos.

Lucrédio (2009), compreende que a transformação entre CIM e PIM exige esforço maior para obter automação, pois envolve mais decisões e maiores possibilidades de interpretação dos requisitos. Entretanto para transformação entre

PSM e código-fonte a automação é obtida com maior facilidade, já que o PSM está fortemente ligado com a plataforma de implementação.

Figura 5 - Visões da Arquitetura MDA.



Fonte: Medeiros (2012).

Para MDA o foco na modelagem é muito importante, pois o desenvolvedor tem sua atenção voltada para criar modelos com níveis de abstração muito altos, esses modelos propostos pela MDA são criados independentes de plataforma, e através de transformações geram modelos para plataformas específicas, e a partir de tais modelos permite a geração automática de código.

A abordagem MDA, define que o processo de software seja direcionado pela atividade de modelagem do sistema, no nível conceitual, desconsiderando implementação ou qualquer tipo de plataforma, e que utilizando técnicas de transformações, este modelo conceitual evolua para níveis cada vez mais específicos e ligados à implementação automaticamente, a partir do modelo gerado inicialmente (SOUZA, 2011 *apud* AGUIAR, 2012).

Segundo Belix (2006), MDA é empregada para que seja capaz de realizar a separação da especificação e da implementação de um sistema em uma plataforma.

Uma das essências da MDA está no conceito de empregar a linguagem de modelagem como linguagem de programação (BELIX, 2006). Para isso, no

desenvolvimento de um software a modelagem representa o projeto, ou seja, como o software tem que ser feito, para diminuir a complexidade durante a elaboração do mesmo.

A interoperabilidade entre ferramentas é importante para a viabilidade do MDA. Ela apoiará o desenvolvimento de modelos e a transformação em código (PARVIAINEN, P. et al., 2009).

Por ser orientada a modelos significa que os artefatos fundamentais do processo de desenvolvimento são os modelos, e eles precisam nortear o entrosamento, o projeto, a construção, a operação, a manutenção e as alterações do sistema (CALIARI, 2007).

2.5 LINHA DE PRODUTO DE SOFTWARE

Segundo os autores Pohl *et al* (2005), o conceito de desenvolvimento de software em famílias consiste basicamente na criação e manutenção de uma LPS que servirá como base para o desenvolvimento de um conjunto de sistemas relacionados a partir de artefatos comuns.

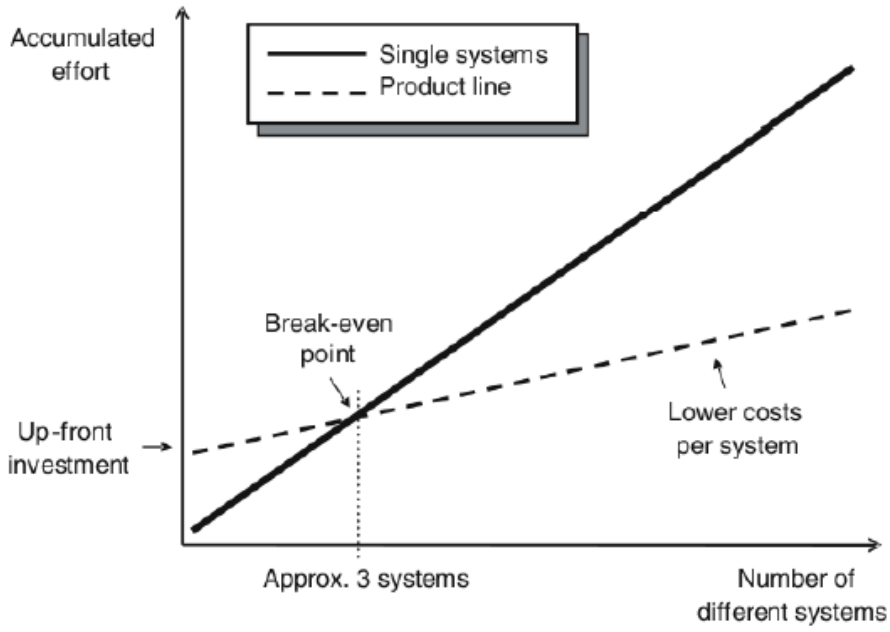
Para diferenciar o processo de desenvolvimento de software tradicional e o de uma linha de produto de software se deve focar na visão ardilosa de uma fração de mercado, ao contrário de criar determinado produto individualizado para atender um algum contrato. Distintos pretextos induzem as empresas a seguir métodos, técnicas e práticas de engenharia de LPS, entre eles destaca-se o custo e o tempo, acarretando redução nos custos e no tempo de entrega do produto e acrescentando qualidade ao produto, e ainda maior credibilidade. A redução dos custos e tempo de entrega do produto ao mercado está profundamente relacionada à abordagem de reutilização em larga escala durante o desenvolvimento de software.

De acordo com Poulin (1997), a reutilização ocasionada pela LPS pode representar 90% do software como um todo. O reuso acarreta maior custo-benefício se for confrontado com o desenvolvimento sob demanda.

Assim sendo, a engenharia de linha de produto pode reduzir, tanto o custo de desenvolvimento como o tempo de entrega do produto. Porém, este incremento não é concedido gratuitamente, é necessário um investimento inicial maior do que a estratégia tradicional, entretanto, em muitos casos, é análogo à aquisição de desenvolvimento de três produtos individuais (LINDEN et al., 2007), como ilustrado

na Figura 5. Partindo desse pressuposto, a tática de engenharia de linhas de produto de software pode torna-se de maneira econômica mais lucrativa.

Figura 6 - Economia da Engenharia de Linha de Produto de Software.



Fonte: (LINDEN *et al.,* 2007).

Métodos de engenharia de linhas de produto de software propõem uma distinção fundamental entre o desenvolvimento de software para reuso e o desenvolvimento de software com reuso, como mostrado na Figura 5, na engenharia de domínio (desenvolvimento para reuso) a base é fornecida para o desenvolvimento de produtos individuais.

Contraditório a diversas abordagens tradicionais de reutilização focalizadas em artefatos de código, a base da linha de produto conglomera todos artefatos que são relevantes ao ciclo de desenvolvimento. Estes artefatos vão desde os requisitos à arquitetura até a implementação aos testes. Este aglomerado de artefatos determina a infraestrutura da LPS. Outra diferença do modelo tradicional é que os múltiplos artefatos contêm variabilidades evidentes. Por exemplo, a representação dos requisitos precisa conter descrições explícitas das variabilidades que se utilizam exclusivamente aos subconjuntos dos produtos.

Alguns princípios são essenciais para a ocorrência de uma abordagem de engenharia de linha de produto de software. Os fundamentais são referidos a seguir (LIDEN et al, 2007):

• Gerenciamento de Variabilidades: sistemas individualizados são analisados como variações de um tema comum. A variabilidade é nítida e deve ser sistematicamente desenvolvida;

• Centrado no Negócio: a engenharia de linhas de produto de software deve estar interligada com a tática de longo prazo do negócio;

• Centrado na Arquitetura: a composição do software deve ser desenvolvida de forma a consentir alcançar vantagens das similaridades entre os sistemas individuais;

• Abordagem de Ciclo de Vida Duplo: Os sistemas individualizados desenvolvidos são fundamentados em plataforma de software. Tanto os produtos como a plataforma, necessitam ser desenvolvidos e ter seus períodos de vidas particularizados.

2.5.1 Atividades da Engenharia de Linha de Produtos de Software

A LPS possui três atividades essenciais (SEI, 2008), essas atividades são conhecidas como: Engenharia de Domínio (ED), Engenharia de Aplicação (EA), e o Gerenciamento da Linha de Produto de Software.

Figura 7 - Atividades da Engenharia de Linha de Produto.

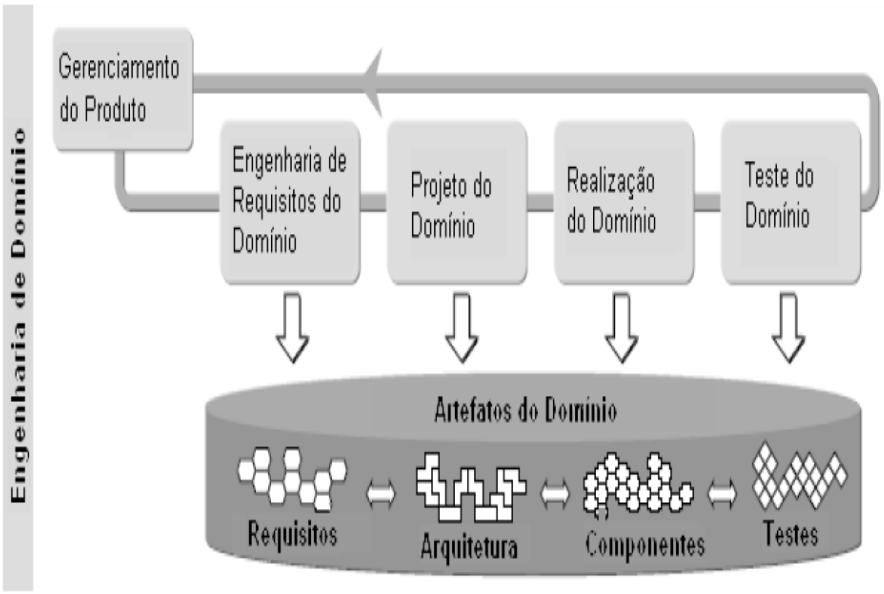
Fonte: (SEI, 2008).

As Atividades da Engenharia de Linha de Produtos de Software são interligadas e interativas, os artefatos gerados no desenvolvimento de produtos, podem conter revisões dos artefatos ou até mesmo incluir novos artefatos. Conforme os produtos vão sendo desenvolvidos, o núcleo de artefatos pode evoluir. Um

exemplo disso ocorre quando novos requisitos que a principio não faziam parte das especificações de requisito da LP são identificados. Assim, essas especificações evoluem, incorporando o novo requisito identificado e o núcleo de artefatos é atualizado.

Engenharia de Domínio - O desenvolvimento do núcleo do produto que corresponde à engenharia de domínio é o processo responsável pelo desenvolvimento de artefatos reutilizáveis, essa atividade é realizada de forma interativa composta por algumas subatividades, sendo elas, o gerenciamento do produto, projeto do domínio, realização de domínio e teste do domínio, seu objetivo é encontrar semelhanças e variabilidades da Linha de produto. A atividade de engenharia de domínio esta subdividida como mostra a figura 8.

Figura 8 - Atividades que Compõem a Engenharia de Domínio.



Fonte: (BURGARELI 2009).

De acordo com Burgareli (2009), a descrição de cada atividade da Engenharia de Domínio é apresentada a seguir:

•Gerenciamento do produto: descreve o quê pertence ou não ao escopo da linha de produto.

• Engenharia de requisito de domínio: identifica o escopo do domínio, apontando os produtos que a linha de produtos é capaz de incluir.

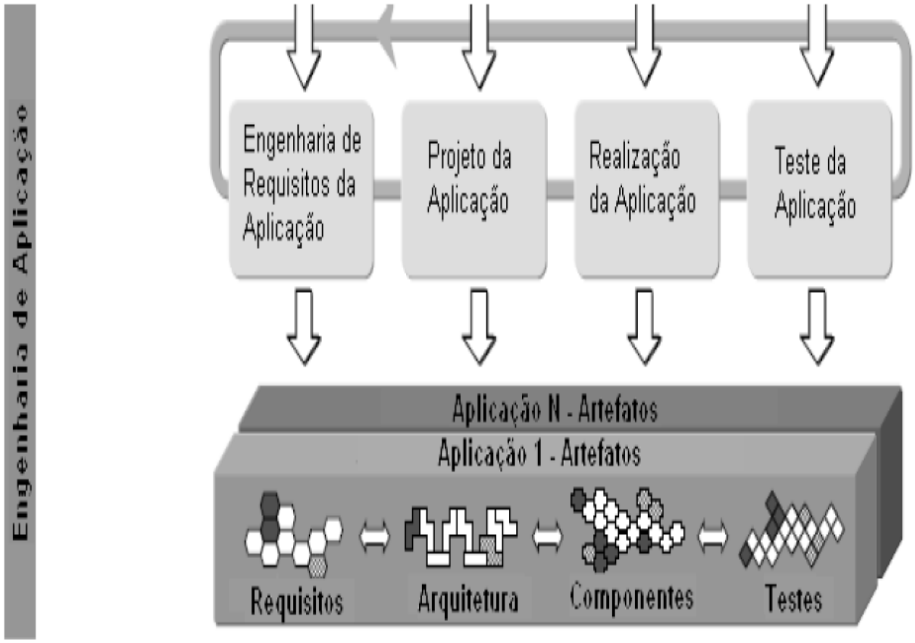
• Projeto de domínio: cria um plano de referência da linha de produto de software, e apresenta um plano de produção.

• Realização do domínio: implementa os ativos principais.

• Teste de domino: verifica e aplica os ativos principais por meio de testes.

Engenharia de aplicação - A Engenharia de Aplicação, ou desenvolvimento com reuso, é responsável pelo desenvolvimento do produto, utilizando o núcleo de artefatos previamente definidos na engenharia de domínio. Esta fase é composta por quatro processos: requisitos de aplicação; projeto de aplicação; realização da aplicação e teste aplicação.

Figura 9. - Atividades que Compõem a Engenharia de Aplicação.



Fonte: BURGARELI (2009).

De acordo com Burgareli (2009), a descrição de cada atividade da Engenharia de aplicação é apresentada a seguir:

• Engenharia de requisitos da aplicação: corresponde ás atividades para a definição da arquitetura do produto com base na arquitetura elaborada durante o projeto de domínio.

• Projeto da aplicação: corresponde a atividades para produzir uma arquitetura de produção utilizando a arquitetura de referencia desenvolvida no projeto de domínio.

• Realização da aplicação: cria a aplicação, utilizando a arquitetura desenvolvida no projeto de aplicação e os artefatos criados na Engenharia de Domínio.

• Teste da aplicação: valida e verifica a aplicação de acordo com sua especificação.

Gerenciamento da linha de produtos - A atividade de gerenciamento tem a responsabilidade de coordenar e supervisionar o desenvolvimento da LPS, e pode ser dividido em duas categorias:

Gerenciamento técnico - coordena as atividades de desenvolvimento relacionado à Engenharia de Domínio e de Aplicação para garantir que as equipes de desenvolvimento sigam os processos definidos para a linha de produto e coletem dados suficientes para acompanhar o progresso desta (COSTA 2008).

Gerenciamento organizacional – determina uma estratégia de produção, por meio da criação de um plano de produção que descreve as necessidades da organização e a estratégia para atender estas necessidades (BURGARELI 2009).

2.5.2 Variabilidade na linha de produto de software

Uma das definições para Variabilidade em linha de produto de software é a capacidade de um sistema ser alterado ou customizado. Melhorar a variabilidade em um sistema de software significa tornar mais fácil realizar certos tipos de mudanças (VAN GRUP, BOSCH SVAHNBERG, 2001 apud BURGARELI).

O gerenciamento de variabilidade é uma atividade que tem como objetivo identificar, projetar, implementar e rastrear a flexibilidade na linha de produto de software (VOELTER, GROHER, 2007 apud BURGARELI).

O gerenciamento de variabilidade está relacionado a todas as atividades de desenvolvimento de LPS e deve conter, pelo menos, as seguintes atividades.

• Variabilidade: Uma característica que varia de uma aplicação para outra em uma LPS.

• Pontos de Variação: Local específico onde uma variabilidade ocorre em um artefato da linha de produto de software.

• Variantes: As diferentes possibilidades para instanciar uma determinada variabilidade.

2.5.3 Principais metodologias para linha de produto de software

O fato dos Métodos FAST (*Family Oriented Abstraction, Specification and Translaction*), PLUS (*Product Line UMLBased Software Engineering*) e FODA (*Feature Oriented Domain Analysis*), serem os métodos mais utilizados na literatura para o desenvolvimento de LPS, para este estudo estes serão os métodos abordados.

Método FAST *(Family Oriented Abstraction, Specification and Translaction*), para Matinlassi (2004), *apud* Burgareli (2009), o método FAST é uma alternativa ao processo clássico de desenvolvimento de software, e tem como objetivo redução das múltiplas tarefas, redução do custo de produção em qualquer contexto que possua múltiplas versões de produtos que compartilham conjuntos de atributos comuns tais como: comportamentos comuns, interfaces comuns e códigos comuns.

Método PLUS *(Product Line UMLBased Software Engineering),* o método PLUS foi desenvolvido baseado em UML. Neste método é feita distinção entre caso de uso e features, casos de uso são orientados ao Desenvolvimento, pois determinam os requisitos funcionais da LPS Features são orientadas ao reuso, pois organiza os resultados de uma análise de semelhanças e variabilidades em LPS (PACIOS, *et al*, 2006 p. 19).

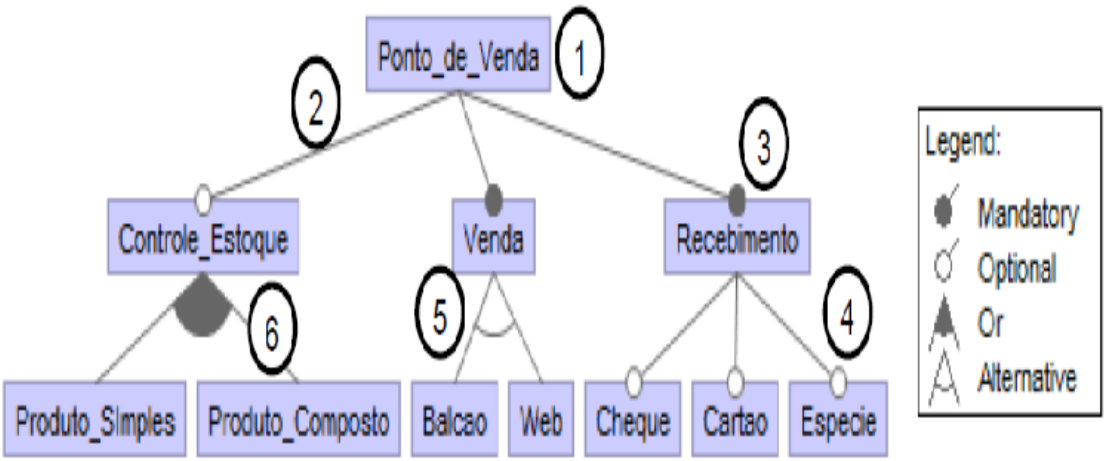
Método FODA (*Feature Oriented Domain Analysis),* também abordado em algumas bibliografias como Modelo de características é baseado em características, e consiste na análise do contexto e modelagem do domínio da aplicação, com a finalidade de obter e representar informações de software que representam características comuns que possam ser reutilizáveis para a criação de outros softwares de mesmo domínio.

Burgareli (2009), entende que na análise do contexto o propósito é definir os limites do domínio a ser modelado, na modelagem do domínio o propósito é identificar os principais atributos comuns e as variabilidades.

O modelo baseado em características pode ser representado visualmente por uma estrutura semelhante a uma árvore, os nós representam as características e as arestas representam o relacionamento entre elas. Para Filgueira (2009), cada nó na árvore ilustra uma *feature* com uma relação pai e filho (s), e as arestas indicam o tipo de representação, se a *feature* é mandatória, opcional ou alternativa. Uma *feature* mandatória é terminada por um círculo preenchido, uma *feature* opcional é

representada por uma aresta terminada por um círculo vazio. Já as *features* alternativas são representadas por arestas ligadas e conectadas por um arco, se o arco for representado de forma vazia permite escolher apenas uma alternativa, se o arco for preenchido permite escolher mais de uma alternativa.

Figura 10 - Modelo de características.



Fonte: Corrêa (2011).

2.6 INTEGRAÇÃO DAS ABORDAGENS LINHA DE PRODUTO DE SOFTWARE E DESENVOLVIMENTO DIRIGIDO A MODELO

Corrêa (2011), compreende que as combinações entre LPS e MDD vêm sendo utilizadas de diferentes maneiras, as abordagens de LPS e MDD têm como objetivo possibilitar a criação de produtos a partir de processos automatizados ou semi-automatizados. Para isso, a abordagem combina linha de produto de software, desenvolvimento dirigido a modelos, frameworks de arquitetura e orientações para a execução das tarefas.

A primeira etapa desta abordagem é o desenvolvimento da linha de produtos, que é correspondente à engenharia de domínio, é nesta etapa, que é realizada a análise da linha de produtos, serão definidos os produtos a serem desenvolvidos as características comuns, e o quê varia entre os produtos.

A segunda etapa é o desenvolvimento do produto e envolve várias atividades como: análise do problema que busca verificar se o produto está dentro do escopo da linha de produtos; especificação do produto, que tem o objetivo de definir os

requisitos do produto em função dos requisitos da linha; projeto do produto nesta atividade, as diferenças em relação aos requisitos são mapeadas para as diferenças correspondentes em relação à arquitetura; implementação do produto, que tem a finalidade da criação do produto propriamente dito; teste e liberação do produto.

Filgueira (2009, p. 20), destaca que a abordagem dirigida a modelos estabelece a utilização de modelos com o objetivo de facilitar o processo de criação de novos softwares para que estes sejam obtidos de forma automática ou semi-automática, e também para que facilite na ocorrência de mudanças. Neste contexto estudos MDD vem sendo propostos para minimizar a lacuna entre os modelos no desenvolvimento de LPS [Santos *et al*., 2006; Anquetil *et al*., 2008; Souza *et al*., 2008] apud FILGUEIRA, (2009, p. 21)

2.6.1 Engenharia de Domínio – MDD

A Engenharia de Domínio segundo COSTA (2008). inclui três atividades principais, na primeira atividade estão envolvidas a criação e o gerenciamento do núcleo base, compostas de características reutilizáveis ou variáveis, na segunda atividade estão envolvidas a especificação de um modelo de domínio, que consiste das características do domínio a nível conceitual, na terceira atividade esta envolvida a definição da transformação, que especifica como as instâncias do modelo de domínio são mapeadas em uma aplicação utilizando o núcleo base.

2.6.1.1 Engenharia de requisito do domínio – MDD

A Engenharia de requisito domínio envolve a identificação das funcionalidades necessárias aos sistemas do domínio. É por meio da análise de domínio que será identificado, os pontos que são comuns a todas as aplicações do domínio e os pontos que variam.

De acordo com Lucrédio (2009) *apud* (Kang *et* al., 1990; Kang; *Lee*; Donohoe, 2002) uma das abordagens mais comuns para esta tarefa é a modelagem de *features.* Conforme visto na seção 2 da revisão bibliográfica existem *features* opcionais, alternativas e obrigatórias.

Lucrédio (2009), compreende que a principal dificuldade, em implementar um domínio que representa as *features*, está em como mapear estas *features* para a arquitetura e o código, analisando os possíveis relacionamentos e as restrições que envolvem a mesma. É neste ponto que as técnicas do MDD podem ser utilizadas.

Ao invés de deixar que estas tarefas sejam executadas pelo desenvolvedor da aplicação, o conhecimento sobre como implementar estas restrições é encapsulado em transformações MDD.

2.6.2.2 Projeto do domínio – MDD

De acordo com Lucrédio (2009), o projeto do domínio é considerado um processo iterativo de refinamento. Inicia-se com o domínio todo, e a cada iteração é feito um refinamento que identifica novos módulos, que serão refinados na próxima iteração, e assim por diante, até que o projeto esteja concluído, em função de um entendimento satisfatório sobre o que deverá ser implementado.

O objetivo principal do projeto de domínio é definir uma arquitetura de software específica de domínio, e um conjunto de artefatos reutilizáveis. Uma arquitetura de software específica de domínio é uma combinação de componentes de software, especializada para um tipo de tarefa em particular (domínio), quando se deseja automatizar o suporte à variabilidade utilizando MDD, a tecnologia escolhida é normalmente uma linguagem específica de domínio (DSL), pois ela pode complementar o modelo de *features* com detalhes sobre variabilidades mais complexas em partes do domínio.

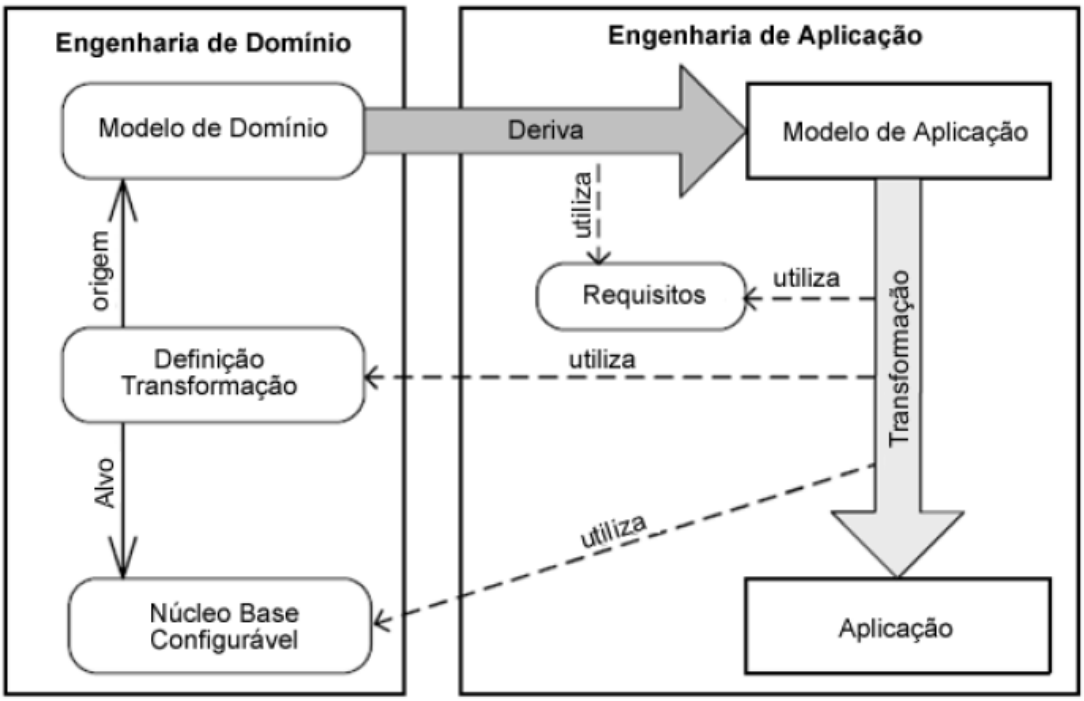
2.6.2.3 Realização do domínio – MDD

Na Realização do domínio segundo Lucrédio (2009), o objetivo é implementar o domínio, ou seja, implementar componentes, DSLs, transformações e geradores de código, seguindo o projeto definido na fase anterior.

2.6.2 Engenharia da aplicação – MDD

De acordo com COSTA (2008), na engenharia da aplicação é gerado um modelo de aplicação com base nas características alternativas do modelo de domínio. É a partir deste processo que as características alternativas são selecionadas baseando-se nos requisitos do produto, assim o modelo de aplicação utiliza a definição da transformação, o núcleo base e os requisitos do produto para a compilação da aplicação.

Figura 11 - Engenharia de linha de produto utilizando a abordagem MDA.



Fonte: (Costa apud DEELSTRA *et al*.,2003).

2.6.3 Gerenciamento da Variabilidade – MDD

Para costa *apud* (DEELSTRA *et al*., 2003) um sistema de software que é especificado de acordo com as descrições da abordagem MDA, especifica o modelo de aplicação para uma linha de produto composto por produtos que implementam as mesmas funcionalidades, sobre plataformas diferentes, essa escolha por plataformas diferentes é um ponto de variação para uma linha de produtos. Utilizando o MDD, o gerenciamento do ponto de variação da plataforma é realizado de forma automatizada através das transformações. A plataforma, entretanto, não é um ponto de variação que precisa ser gerenciado em uma linha de produtos.

Ao apresentar uma pesquisa sobre abordagens de gerenciamento de variabilidades, as principais variantes consideradas foram: sistemas configuráveis, linhas de produtos de software e plataformas de software. Desta forma, linhas de produtos de software representam, segundo a visão dos autores, 23% das iniciativas de gerenciamento de variabilidade, atrás apenas de sistemas configuráveis que representam, naquela amostra, 32%. Paludo *apud* (*VILLELA et al*., 2014),

3. ANÁLISE COMPARATIVA

A comparação consiste em verificar a utilização dos principais conceitos propostos pelas abordagens MDD e LPS, tanto a nível organizacional, quanto a nível acadêmico, e apresentar um cenário sobre reuso de software utilizando as duas abordagens.

Para análise no contexto acadêmico foram selecionadas três bibliografias que apresentaram uma melhor combinação entre as duas abordagens.

Para análise no contexto organizacional foi selecionada uma bibliografia que apresenta um estudo de caso, onde foram analisadas nove organizações, com o objetivo de traçar o cenário atual de práticas de reuso de software nas organizações desenvolvedoras de software, considerando as abordagens linha de produto de software e engenharia dirigida a modelos como foco principal para a investigação.

O resultado da analise no contexto acadêmico e o resultado da analise no contexto organizacional, é onde se apresenta o estudo comparativo.

3.1 Análise no contexto acadêmico

Para a análise a nível acadêmico foram utilizadas duas teses de doutorado e um trabalho para conclusão de graduação, e procura verificar dois pontos, se os principiais conceitos propostos pela abordagem desenvolvimento dirigidos estão sendo utilizados, e se principiais conceitos propostos pela abordagem linha de produto de software estão sendo utilizados.

As bibliografias utilizadas para análise no contexto acadêmico abordam diferentes propostas sobre como lidar com desenvolvimento dirigido a modelos em conjunto com Linha de Produto de Software.

Lucrédio (2009) apresenta uma abordagem sistemática e bem definida com foco na identificação de como aplicar o MDD durante a engenharia de domínio.

Costa (2008), apresenta um estudo de caso utilizando a ferramenta AndroMDA.

A proposta de Corrêa (2011), é apresentar uma abordagem que permita a evolução da plataforma e dos produtos de uma LPS-DM.

3.1.1 Tecnologias e ferramentas utilizadas no Contexto Acadêmico

No estudo de caso COSTA (2008), utilizou a ferramenta AndroMDA, e detalha as fases necessária durante o processo de desenvolvimento da aplicação.

1. Definição do modelo de domínio, para ser transformado no modelo independente de plataforma e modelado em uma ferramenta CASE;
2. Exportação do modelo independente de plataforma no formato XMI;
3. Configuração dos *templates* que serão utilizados no projeto durante o processo de transformação;
4. Inicializar o processo de transformação do modelo conforme as configurações definidas no projeto;
5. Realizar a implementação dos métodos que não são automatizados pela transformação;
6. Realização de Testes;

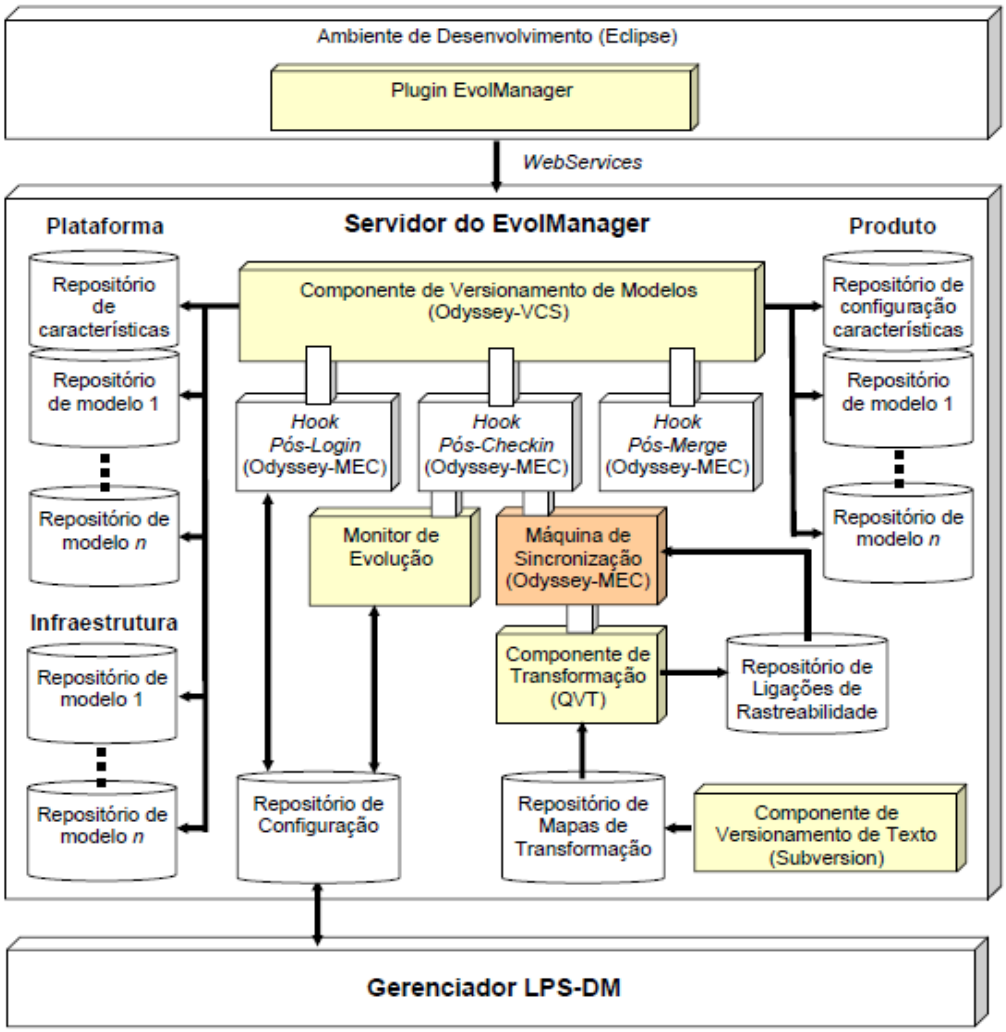
Após o processamento de transformações utilizando o AndroMDA podem haver alguns artefatos que apresentam características que não foram implementadas automaticamente durante o processo de transformação, e devem ser implementadas manualmente pelo desenvolvedor utilizando alguma IDE. Para realizar as implementações manualmente COSTA (2008), utilizou a IDE Eclipse.

O trabalho de Corrêa (2011), apresenta uma arquitetura cliente/servidor, denominado *EvolManager* de modo que todos os modelos da infraestrutura, da plataforma e dos produtos fiquem no servidor.

Na proposta de Corrêa (2011), o servidor *EvolManager* está sendo implementado para funcionar na plataforma Eclipse (ECLIPSE, 2011), uma vez que este ambiente fornece suporte à LPS e ao DDM. A comunicação entre o *EvolManager* e o Eclipse é realizada através de um *plugin*.

A figura 12 ilustra a arquitetura do *EvolManager.*

Figura 12 - Arquitetura do EvolManager



Fonte: Corrêa (2011).

Os componentes do servidor do *EvolManager* proposto por Corrêa (2011), são:

- Odyssey-VCS, para o versionamento de modelos
- Subversion, para o versionamento de especificações de transformação,
- Odyssey-MEC,responsável pelo processo de sincronização. O Odyssey-MEC utiliza um componente de transformação para executar as transformações e gerar modelos.
- Máquina de transformação QVT (*Query View Transformation*), para especificações de transformação.

3.1.2 Análise da Utilização dos Principais Conceitos MDD no Contexto Acadêmico.

Os principais conceitos que envolvem a abordagem de desenvolvimento dirigido a modelos de acordo com a revisão de literatura são;

- Transformação de modelos
- Modelo Independente de Computação
- Modelo independente de Plataforma
- Modelo Específico de Plataforma

A presença do caractere ✓ referente á linha coluna indica que o conceito está sendo utilizado na bibliografia.

A presença do caractere ± referente á linha coluna indica que o conceito está sendo utilizado parcialmente na bibliografia.

A presença do caractere **x** referente á linha coluna indica que o conceito não está sendo utilizado na bibliografia.

O quadro um apresenta a utilização dos principais conceitos que envolvem a abordagem MDD no contexto acadêmico.

Quadro 1 - Desenvolvimento Dirigido a Modelos no Contexto Acadêmico.

Autores / Conceitos	Lucrédio (2009)	Corrêa (2011)	Costa (2008)
Transformação de modelos	✓	✓	✓
Modelo Independente de Computação	x	x	±
Modelo independente de Plataforma	±	✓	✓
Modelo Específico de Plataforma	±	✓	x

Na tese de Lucrédio (2009), o foco do estudo é a combinação de engenharia de domínio e Linha de produto de software, sendo que a fase de desenvolvimento de aplicações não é abordada, portanto os conceitos Modelo independente de

Plataforma e Modelo Específico de Plataforma foram considerados como parcialmente aplicados.

Costa (2008) descreve em seu trabalho que o modelo de domínio elaborado não contém todos os pontos de variabilidade possíveis, assim o conceito modelo independente de computação foi considerado como parcialmente aplicado. Costa (2008) descreve ainda que a ferramenta AndroMDA utilizada ainda não permite a transformação de modelo para modelo, assim a partir do modelo independente de plataforma será gerado o código fonte.

O conceito Transformação de Modelos foi utilizado em todas as bibliografias. O conceito Modelo Independente de Computação, foi utilizado parcialmente em uma das três bibliografias, o conceito Modelo Independente de Plataforma esteve sendo utilizado de forma mais considerável em duas das três bibliografias, o conceito Modelo Específico de Plataforma esteve sendo utilizado de forma mais considerável em uma das três bibliografias.

3.1.3 Análise da Utilização dos Principais Conceitos LPS no Contexto Acadêmico.

Os principais conceitos que envolvem a abordagem de linha de produto de software de acordo com a revisão de literatura são as três atividades essências da linha de produto de software.

- Engenharia de Domínio
- Engenharia de Aplicação
- Gerenciamento, onde está inserido o conceito de Gerenciamento da variabilidade.

O quadro dois apresenta a utilização dos principais conceitos que envolvem a abordagem LPS no contexto acadêmico.

Quadro 2 - Linha de Produto de Software no Contexto Acadêmico.

Conceitos \ Autores	Lucrédio (2009)	Corrêa (2011)	Costa (2008)
Engenharia de Domínio	✓	✓	✓
Engenharia de Aplicação	x	✓	✓
Gerenciamento de variabilidade	✓	✓	✓

O conceito de Engenharia de Domínio, assim como o conceito de Gerenciamento de variabilidade esteve sendo utilizados em todas as bibliografias. O conceito de Engenharia de Aplicação mostrou utilização mais consideráveis em duas das três bibliografias analisadas, é importante destacar que a bibliografia que não apresentou a utilização do conceito de Engenharia de Aplicação tem o estudo voltado em como aplicar o MDD durante a engenharia de domínio.

3.2 Análise no contexto organizacional

Para a análise a nível organizacional foi utilizada uma tese de doutorado.

Em Paludo (2016), é apresentado um estudo de caso, analisando nove organizações, com o objetivo de traçar o cenário atual na adoção de práticas de reuso, considerando como foco principal para a investigação as abordagens linha de produto de software e engenharia dirigida a modelos. A análise aqui proposta procura verificar dois pontos, se o os principais conceitos de desenvolvimento dirigidos estão sendo utilizados nas organizações analisadas, e se os principais conceitos de linha de produto de software estão sendo utilizados nas organizações analisadas.

Perfil das Organizações Analisadas

Organização A

Capital Privado; apresenta um quadro de 10 a 49 colaboradores em relação a atividades de desenvolvimento e manutenção de produtos de software.

Organização B

Capital Público; Mais de 500 colaboradores em relação a atividades de desenvolvimento e manutenção de produtos de software.

Organização C

Capital Público; apresenta um quadro de 100 a 499 colaboradores em relação a atividades de desenvolvimento e manutenção de produtos de software.

Organização D

Capital Privado; apresenta um quadro de 100 a 499 colaboradores em relação a atividades de desenvolvimento e manutenção de produtos de software.

Organização E

Capital Privado; apresenta um quadro de 10 a 49 colaboradores em relação a atividades de desenvolvimento e manutenção de produtos de software.

Organização F

Capital Privado; apresenta um quadro de 100 a 499 colaboradores em relação a atividades de desenvolvimento e manutenção de produtos de software.

Organização G

Capital Privado; apresenta um quadro de 100 a 499 colaboradores em relação a atividades de desenvolvimento e manutenção de produtos de software.

Organização H

Capital Privado; apresenta um quadro de 10 a 49 colaboradores em relação a atividades de desenvolvimento e manutenção de produtos de software.

Organização I

Capital Privado; apresenta um quadro de 100 a 499 colaboradores em relação a atividades de desenvolvimento e manutenção de produtos de software.

Inicialmente á análise procura verificar se as organizações apresentam iniciativas da utilização das abordagens linha de produto de software e desenvolvimento dirigido a modelos, para tanto duas questões utilizadas por Paludo (2016), se faz necessário, e são apresentadas a seguir:

- Existem iniciativas de reuso de software valendo-se de abordagens dirigidas a modelos na organização.
- Existem possíveis formas de implementação de linha de produto de software na organização.

O quadro três mostra o resultado das organizações apresentarem iniciativas de utilizarem tanto a abordagem MDD quanto a abordagem LPS.

Quadro 3 – iniciativas de reuso de software utilizando as abordagens MDD e LPS no contexto organizacional

Organização	Existem iniciativas de reuso de software valendo-se de abordagens dirigidas a modelos	Existem possíveis formas de implementação de linhas de produto de software.
A	Fracamente	Fracamente
B	Largamente	Integralmente
C	Fracamente	Largamente
D	Integralmente	Fracamente
E	Fracamente	Parcialmente
F	Fracamente	Parcialmente
G	Parcialmente	Largamente
H	Fracamente	Fracamente
I	Fracamente	Parcialmente

Fonte: Adaptado Paludo (2016).

É possível verificar que dentre as nove organizações utilizadas para análise em Paludo (2016), apenas duas apresentaram condições favoráveis para reuso de software abordando Linha de Produto de Software e Desenvolvimento Dirigido a modelos.

Para a sequência das analise serão consideradas apenas as organizações B e G, pois estas foram as duas organizações que apresentaram condições favoráveis para a sequência das análises, que assim como na analise a nível acadêmico também procura verificar a utilização dos principais conceitos das abordagens de Linha de Produto de Software e Desenvolvimento Dirigido a modelos.

Organização B

- **Engenharia do domínio e engenharia da aplicação.**

Os conceitos de engenharia do domínio e engenharia da aplicação são aplicados em algumas situações.

- **Gerenciamento da variabilidade**.

É possível evidenciar que há várias formas de tratamento das variabilidades em vários artefatos.

- **Transformações de modelos**

A organização possui algumas iniciativas dirigidas a modelos institucionalizadas, em especial considerando a geração automática de tabelas a partir dos modelos de dados, porções de código e esqueletos de programas a partir de modelos.

- **Modelo Independente de Computação**

Não ha a evidencias da utilização do conceito Modelo Independente de Computação

- **Modelos independentes de plataforma**

Utilizam geração de código, tabelas e funcionalidades a partir de modelos, entretanto sem tanta ênfase para os modelos independentes de plataforma.

- **Modelo específico de plataforma**

Utilizam geração de código, tabelas e funcionalidades a partir modelos específicos de plataforma.

Organização G

- **Engenharia do domínio e engenharia da aplicação.**

A organização apresenta iniciativas que consideram as engenharia do domínio e a engenharia da aplicação, mas sem criar a linha de produto de software de maneira mais formal.

- **Gerenciamento da variabilidade**.

É possível evidenciar que há várias formas de tratamento das variabilidades, porem nem todos os ativos possuem gerenciamento de variabilidade.

- **Transformações de modelos**

A organização não faz uso de transformação de modelos até o ponto de geração de código, entretanto apresenta geração de templates e esqueletos a partir de ferramentas de modelagem.

- **Modelo Independente de Computação**

Não houve evidencias da utilização do conceito Modelo Independente de Computação

- **Modelos independentes de plataforma**

Não houve evidencias da utilização do conceito Modelo Independente de Plataforma

- **Modelo específico de plataforma**

Não houve evidencias da utilização do conceito Modelo específico de plataforma.

3.2.1 Tecnologias e Ferramentas Utilizadas no Contexto Organizacional.

Observa-se que a organização B dispõe de ferramentas que permitem o reuso, tanto ferramentas desenvolvidas internamente, quanto ferramentas a partir da modalidade de software livre. É possível identificar a utilização da ferramenta *Enterprise Architect*, ferramentas e *plugins* para o ambiente Eclipse, também é possível verificar o uso da ferramenta AndroMDA, está sendo utilizada para um único projeto.

3.2.2 Análise da Utilização dos Principais Conceitos LPS no Contexto Organizacional.

O quadro quatro apresenta o resultado da analise da utilização dos principais conceitos que envolvem a abordagem LPS no contexto organizacional.

Quadro 4 - Linha de Produto de Software no Contexto Organizacional.

Conceitos	Organização B	Organização G
Engenharia de Domínio	✓	✓
Engenharia de Aplicação	✓	✓
Gerenciamento de variabilidade	✓	✓

A organização B aplica em algumas situações os conceitos de engenharia de domínio, e engenharia de aplicação. A organização G utiliza os conceitos de engenharia de domínio, e engenharia de aplicação, mas sem criar uma linha de produto de software de maneira formal.

A organização B mostrou utilizar o conceito de gerenciamento de variabilidade, assim como a organização G também mostrou utilizar o conceito de

gerenciamento de variabilidade, contudo na organização G nem todos os ativos apresentam gerenciamento de variabilidade.

3.2.3 Análise da Utilização dos Principais Conceitos MDD no Contexto Organizacional.

O quadro cinco apresenta o resultado da analise da utilização dos principais conceitos que envolvem a abordagem MDD no contexto organizacional.

Quadro 5 - Desenvolvimento Dirigido a Modelos no Contexto Organizacional.

Conceitos	Organização B	Organização G
Transformação de modelos	✓	X
Modelo Independente de Computação	x	X
Modelo independente de Plataforma	✓	X
Modelo Específico de Plataforma	✓	X

A organização B mostrou utilizar o conceito de transformação de modelos, já a Organização G não apresentou evidências suficiente para utilização dos conceitos de transformação de modelos.

Em ambas as organizações não houve utilização sobre os conceitos Modelo Independente de Computação.

O conceito transformação de modelos independentes de plataforma para os modelos específicos de plataforma apresentou ser utilizado pela organização B, entretanto sem tanta ênfase. A organização G não apresentou evidências da utilização do conceito Modelo Independente de Plataforma.

A organização B em alguns casos utiliza o conceito Modelo específico de Plataforma para geração de código, tabelas e funcionalidades, já a organização G não apresentou utilização do conceito Modelo Específico de Plataforma.

3.3 Análise Comparativa Contexto Acadêmico e Contexto Organizacional

O quadro seis apresenta a comparação da utilização dos principais conceitos que envolvem a abordagem MDD no contexto acadêmico e no contexto organizacional.

Quadro 6 - Análise comparativa Desenvolvimento Dirigido a Modelos

Desenvolvimento Dirigido a Modelos	Contexto acadêmico	Contexto organizacional
Transformação de modelos	✓	✓
Modelo Independente de Computação	±	X
Modelo independente de Plataforma	✓	✓
Modelo Específico de Plataforma	✓	✓

Conforme apresentado no quadro1 duas das três bibliografias analisadas no contexto acadêmico não aplica a utilização do conceito Modelo Independente de Computação, apenas Costa (2008) utiliza parcialmente o conceito de modelo independente de computação tendo em vista que o modelo de domínio elaborado não contém todos os pontos de variabilidade possíveis.

Ao analisar a utilização dos principais conceitos que envolvem a abordagem MDD é possível identificar que não ocorre a utilização do conceito Modelo Independente de Computação no contexto organizacional, e no contexto acadêmico e aplicado de forma parcial. Os conceitos de Transformação de modelos e Modelo Específico de Plataforma são utilizados tanto no contexto acadêmico quanto no contexto organizacional. O conceito Modelo independente de Plataforma também mostrou ser utilizado em ambos os contextos. Entretanto com pouca ênfase no contexto organizacional.

O quadro sete apresenta a comparação da utilização dos principais conceitos que envolvem a abordagem LPS no contexto acadêmico e no contexto organizacional.

Quadro 7 - Análise Comparativa Linha de Produto de Software.

Linha de Produto de Software	Contexto acadêmico	Contexto organizacional
Engenharia de Domínio	✓	✓
Engenharia de Aplicação	✓	✓
Gerenciamento de variabilidade	✓	✓

Ao analisar a utilização dos principais conceitos que envolvem a abordagem LPS é possível identificar que, ambos os contextos procuram valer-se dos benefícios desta abordagem.

Tanto no contexto acadêmico quanto no contexto organizacional é possível verificar a utilização dos principais conceitos que envolvem a abordagem de LPS. Embora o contexto organizacional não tenha apresentado evidências em adotar uma linha de produto de software de maneira mais formal, os conceitos Engenharia de Domínio, Engenharia de Aplicação e Gerenciamento de variabilidade apresentaram estar sendo utilizados para instanciar novos produtos a partir de produtos já existentes.

4. CONSIDERAÇÕES FINAIS

Observa-se que as publicações que envolvem a integração das abordagens linha de produto de software e desenvolvimento dirigido a modelo, são recentes, principalmente a partir de 2007, e é notável a preocupação na área de engenharia de reuso, visto que as combinações corretas entre as duas abordagens pode melhorar o processo de reutilização de software.

O estudo apresenta um cenário no qual adotar a reutilização de software, utilizando as abordagens LPS e MDD, em ambos os contextos procuram se beneficiar de tecnologias e ferramentas bastante parecidas, embora o contexto organizacional tenha apresentado condições favoráveis para utilização das abordagens LPS e MDD em apenas uma das nove organizações analisadas, esta mostrou semelhanças na utilização de tecnologias e ferramentas utilizadas no contexto acadêmico.

Em linhas gerais é possível concluir com base na utilização dos principais conceitos propostos pelas abordagens LPS e MDD, que o reuso de software em nível de modelo independente de computação ocorre de forma parcial no contexto acadêmico, e no contexto organizacional o conceito não se mostrou aplicado. O reuso de software em nível de modelo independente de plataforma mostrou evidências ser aplicado em ambos os contexto, no entanto apresenta ser utilizado com pouca ênfase no contexto organizacional, Já o reuso de software em nível modelo especifico de plataforma apresenta ser aplicado tanto no contexto organizacional quanto no contexto acadêmico.

4.1 Trabalhos Futuros

A integração das abordagens linha de produto de software e desenvolvimento dirigido a modelo oferece uma importante contribuição no processo de desenvolvimento de software. Dentro deste contexto, um possível trabalho futuro a ser considerado é propor um estudo somente com organizações que abordam desenvolvimento dirigido a modelo e linha de produto de software e relatar a forma como a prática da integração das abordagens é utilizada.

Ainda diante da dificuldade da pesquisa em encontrar bibliografias brasileiras que aplica a integração das abordagens desenvolvimento dirigido a modelo e linha de produto de software seria possível propor uma expansão incluindo bibliografias estrangeiras, e explorar com mais profundidade a prática de reuso.

REFERÊNCIAS

AGUIAR, Guilherme: Uso de MDA em um Framework para Seleção de Práticas Ágeis. Universidade Federal de Santa Catarina. Florianópolis, SC, 2012.

BELIX, José Eduardo. Um Estudo Sobre MDA: Suporte fornecido pela UML e Reuso de Soluções Pré-Definidas. 2006. Dissertação (Mestrado em Engenharia Elétrica) Departamento de Engenharia da Computação e Sistemas Digitais, Escola Politécnica da Universidade de São Paulo, São Paulo, 2006.

BÉZIVIN, J. On the Unification Power of Models. Software and Systems Modeling, vol. 4,no. 2, pp.171-188, 2005.

BOOCH, G., RUMBAUGH, J., JACOBSON, I., 2005, UML - Guia do Usuário, 2 ed., Editora Campus.

BRAMBILLA, M.; CABOT, J.; WIMMER, M. Model-driven software engineering in practice. [S.l.]: Morgan & Claypool Publishers, 2012.

Burgareli, Luciana. Akemi. Gerenciamento de Variabilidade de Linha de Produtos de Software com utilização de Objetos adaptáveis e reflexão. Universidade de São Paulo. São Paulo, 2009.

CALIARI, G. Transformações e mapeamentos da MDA e sua implementação em três ferramentas. USP. 2007.

CLEMENTS, P.; NORTHROP, L. Software Product Lines: Practices and Patterns . [S.l.]: Addison-Wesley, 2002.

COSTA, Cláudio, COELHO, Diogo, OLIVEIRA, Fábio, SILVA, Marcos, VILAÇA, Susana. Model Driven Development. Disponível em: <http://paginas.fe.up.pt/~aaguiar/es/artigos%20finais/ es_final_5.pdf> Acesso em 09 outubro 2016.

COSTA. Eduardo Barbosa da. Arquitetura Orientado por Modelos aplicada a Linha de Produto de Software. Universidade Federal de Juiz de Fora. Juiz de Fora, MG, 2008.

CORRÊA, Chessamn Kennedy Faria. Evolmanager: uma Abordagem para o Controle de Evolução. Consistente de Linhas de Produtos de Software Dirigidas Por Modelos. Universidade Federal do Rio de Janeiro Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa de Engenharia Programa de Engenharia de Sistemas e Computação. RIO DE JANEIRO, 2011.

CORRÊA, C.K.F., OLIVEIRA, T.C., WERNER, C.M.L., 2011, "An Analysis of Change Operations to Achieve Consistency in Model-Driven Software Product Lines". In: International Workshop on Model-driven Approaches in Software Product Line Engineering, Munich, Germany.

DEMIR, A. Comparison of model-driven architecture and software factories in the context of model-driven development. Proc. - Joint Meeting of the 4th Workshop on Model-Based Dev. of Computer-Based Systems and the 3rd Int. Workshop on Model-Based Methodologies for Pervasive and Embedded Software, MBD/MOMPES 2006, p. 75–83, 2006.

D'SOUZA, D.; WILLS, A.Objects, Components and Frameworks with UML: The Catalysis Approach. [S.l.]: Addison-Wesley, 1999. (Object Technology Series).

FERREIRA, João Paulo Augusto de Oliveira. XO2 – Um Gerador de Código MDA Baseado em Mapeamento de Modelos. 2005. Dissertação (Mestrado em Ciência da Computação) Centro de Informática da Universidade Federal de Pernambuco, Universidade Federal de Pernambuco, Pernambuco, 2005.

Filgueira, Geam Carlos de Araújo. CrossMDA-SPL: uma abordagem para gerência de variabilidades dirigida por modelos e aspectos / Geam Carlos de Araújo Filgueira. – Natal, RN, 2009. 187 f.: il.

GIL, A.C. Métodos e técnicas de pesquisa social. São Paulo: Atlas, 1999.

KLEPPE, A.; WARMER, J.; BAST, W. (2002) "MDA Explained: The Model Driven Architecture: Practice and Promisse". Addison-Wesley, 2002. ISBN: 4-8443-1869-1
KRUEGER, C. Software reuse. ACM Computing Surveys, v. 24, n. 02, p. 131–183, 1992.

Linden, F. van der; Schmid, K. & Rommes, E., 2007. Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering, Springer.

LUCRÉDIO, Daniel. Uma Abordagem Orientada a Modelos para Reutilização de Software Uma Abordagem Orientada a Modelos para Reutilização de Software. 19–232 p. Tese (Dissertação) — Universidade de São Paulo, 2009.

MAIA, Natanael E. N. Odyssey-MDA: Uma Abordagem Para a Transformação de Modelos. 2006. Dissertação (Mestrado em Ciência da Computação) Coordenação dos Programas de Pós-Graduação, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2006.

MEDEIROS, Ana. Luiza. Uma Estratégia Dirigida a Modelos e Baseada em Linguaem de Descrição Arquitetural para Linhhas de Produtos de Software. Disponível em: <http://repositorio.ufrn.br:8080/jspui/bitstream/123456789/17950/1/AnaLFM_TESE.pdf> Acesso em 26 de setembro de 2016.

MOREIRA, H.; CALEFFE, L. G. Metodologia da pesquisa para o professor pesquisador. Rio de Janeiro: DP&A, 2006.

OLIVEIRA, Jéssica Bassani de. Uma Abordagem Baseada em Engenharia Dirigida por Modelos para Suportar o Teste de Sistemas de Software Na Plataforma de

Computação em Nuvem. Dissertação (Mestrado) Programa de Pós-Graduação em Engenharia da Eletricidade. São Luiz, 2012.

OMG (2016). Mda - the architecture of choice for a changing world. Disponível em http://www.omg.org/mda/. Acesso em 01 de novembro de 2016.

Pacios, S. F.; Masiero, P. C.; Braga, R. T. V., 2006. Guidelines for Using Aspects to Evolve Product Lines. In: III Workshop Brasileiro de Desenvolvimento de Software Orientado a Aspectos, p.111-120.

PALUDO, Marco Antonio. Reúso de Software com Ênfase em Abordagens Dirigidas a modelos e Sistemas de Alta Variabilidade: Estudos de Caso no Brasil Tese (Dissertação) — Pontifícia Universidade Católica do Paraná, Curitiba, 2016.

PARVIAINEN, P. et al. Model-Driven Development Processes and practices. [S.l.]: VTT Technical Research Centre of Finland, 2009. ISBN 9789513871758.

Práticas e Experiências no Ensino de Engenharia Dirigida por Modelos. Disponível em: http://www.seer.unirio.br/index.php/isys/article/view/5441/5408. Disponível em: Acesso em 30 de setembro de 2016.

Pohl, K.; Bockle, G.; Linden, F.J.V.D. (2005) Software Product Line Engineering: Foundations, Principles and Techniques. Springer.

Poulin, J, 1997. Measuring Software Reuse — Principles, Practices, and Economic Models. Addison-Wesley.

REZENDE Izabella Cristine Oliveira. A Engenharia de Requisitos e o Desenvolvimento Dirigido a Modelos: Uma Revisão Sistemática. Disponível em: https://www.researchgate.net/publication/300556223_A_ENGENHARIA_DE_REQUISITOS_E_O_DESENVOLVIMENTO_DIRIGIDO_A_MODELOS_UMA_REVISAO_SISTEMATICA. Acesso em 27outubro. 2016

RINE, D.; SONNEMANN, R. Investment in reusable software a study on software reuse investment success factors. The Journal of Systems and Software, v. 41, p. 17–32, 1998.

SEI - Software Engineering Institute (2008). A framework for software product line practice, version 5.0. Pittsburgh. Disponível em: <http://www.sei.cmu.edu/productlines/framework.html >. Acesso em 22 outubro. 2016

TEPPOLA, S.; PARVIAINEN, P.; TAKALO, J. Challenges in Deployment of Model Driven Development. 2009 Fourth International Conference on Software Engineering Advances , Ieee, p. 15–20, set. 2009.. Disponível em <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5298434>. Acesso em 12 de outubro de 2016.

VARA, J. M.; MARCOS, E. A framework for model-driven development of information systems: Technical decisions and lessons learned. Journal of Systemsand Software,

Elsevier Inc., v. 85, n. 10, p. 2368–2384, out. 2012. ISSN 01641212. Disponível em: <http://linkinghub.elsevier.com/retrieve/pii/S0164121212001367>. Acesso em 14 de outubro de 2016.

Zambiasi, Marcelo NovaStudio: Geração de Testes a partir de uma Modelagem UML/ Marcelo Zambiasi – Porto Alegre, 2010 79 f.: il.