Análise de Domínio: Um Método para Geração Automática do Modelo de *Features*

Marianne B. Diniz da Silva[1], Isabel D. Nunes[2]

[1] Programa de Pós-Graduação em Ciência da Computação (PROCC) Universidade Federal de Sergipe (UFS) – São Cristóvão, SE – Brasil

[2] Instituto Metrópole Digital (IMD – UFRN), Natal – RN – Brasil.

mariannedinizsi@gmail.com, bel@imd.ufrn.br

Abstract. *The Software Product Line - SPL are families of software products that share characteristics but which allows a variety of electronic devices to use them, even having no compatibility with each other. However, in these projects, one of the major problems identified is the failure of the process requirements. In SPL, the Domain Engineering is responsible for the requirements, and aims to carry out the activities to collect, organize and store past experience in the development of systems or parts of a specific domain system. So if there is a failure in one of these activities will cause a ripple effect on others, jeopardizing the construction of the final product. In this context, this article describes a method for generating the feature model automatically in Software Product Lines that minimizes the problems encountered in the propagation of errors from the Domain Engineering.*

Resumo. *A Linha de Produto de Software - LPS são famílias de produto de software que compartilham características mas que permite que uma variedade de dispositivos eletrônicos os utilizem, mesmo não possuindo compatibilidade entre si. Porém, nesses projetos, um dos maiores problemas diagnosticados é a falha nos processos de requisitos. Em LPS, a Engenharia de Domínio é a responsável pelos requisitos, e tem como objetivo realizar as atividades de coletar, organizar e armazenar experiências passadas no desenvolvimento de sistemas ou partes do sistema de um domínio específico. Assim, se houver falha em uma dessas atividades vai ocasionar um efeito cascata nos demais, comprometendo a construção do produto final. Neste contexto, este artigo descreve um método para geração do modelo de features de forma automática em Linhas de Produto de Software que minimize os problemas encontrados na propagação de erros a partir da Engenharia de Domínio.*

1. Introdução

A utilização de equipamentos com software integrados, por exemplo, *tablets*, *smartphones*, *Global Positioning System* - GPS, computadores e câmeras, vem se tornando bastante comum nos últimos anos. O crescimento desses acessos aos

equipamentos fez com que várias aplicações surgissem, fazendo com que dispositivos que antes só eram utilizados com uma única função pudessem acumular várias. No entanto, o desenvolvimento desses softwares nem sempre apresenta compatibilidade entre os diversos modelos de equipamentos (ALMEIDA; MIRANDA, 2010).

Uma solução para o problema é o desenvolvimento de várias versões da aplicação para cada tipo ou categoria de equipamentos. Porém, essa solução não é adequada no momento da manutenção, pois seria necessário ajustar cada produto com a nova funcionalidade ou modificação (CZARNECKI; EISERNECK, 2005).

As versões de softwares compartilham um conjunto de características que são funcionais ou não. As características comuns de cada versão são chamadas de similaridades. Já as características variáveis de uma versão para a outra são chamadas de variabilidades. Cada versão composta por um conjunto de características similares e variáveis caracteriza-se como uma família de sistemas (GERINER et al., 2012), caracterizando uma Linha de Produto de Software - LPS (NORTHROP; CLEMENTS, 2007).

Para o desenvolvimento de LPS destacam-se três atividades principais: Engenharia de Domínio com ênfase em análise, projeto e implementação da similaridade e variabilidade, que serão usadas para criar os produtos da linha; a Engenharia de Aplicação, na qual os produtos de softwares são criados por meio da reutilização dos artefatos criados na atividade da Engenharia de Domínio; e o Gerenciamento que supervisiona e coordena as atividades da linha (HIRAMA, 2012).

Para Dias e Araújo (2010) 40% a 60% dos problemas diagnosticados em um projeto de software são ocasionados por falhas nos processos de requisitos, visto que, na Análise de Domínio todas as suas fases são criadas manualmente e que uma depende da outra, podendo ocasionar um grande problema na família de produto. Problema esse que está relacionado às características dos produtos finais, pois se foi realizado o processo de Análise de Requisitos errado, que por sua vez foi base para o Diagrama e Especificação de Caso de Uso que também poderá conter erros, que será base para a construção da Modelagem de *Features*, ou seja, seria um erro em cascata.

O objetivo deste artigo é propor um método para minimizar erros na Modelagem de *Features*, dado que as ferramentas existentes realizam a modelagem de *features* de forma manual, podendo ocasionar inconsistências entre os artefatos de entrada e o de saída. O método proposto realiza a extração das *features* por meio de diagramas de casos de usos adaptados para a abordagem *Product Line* UML- *Based Software Engineering* – PLUS (GOMAA, 2005), e modela as *features* com base na abordagem *Feature Oriented Domain Analysis* - FODA (KANG et al., 1990).

O artigo está organizado da seguinte forma: na Seção 2, são apresentadas teorias relevantes; na Seção 3, o método é detalhado; e a Seção 4 apresenta as considerações finais.

2. Fundamentação Teórica

Para a construção do método, alguns trabalhos e referências bibliográficas foram levados em consideração, de modo que ao final da elaboração do método, este pudesse contribuir com as pesquisas na área de Linha de Produto de Software.

2.1. Linha de Produto de Software - LPS

A linha de produtos compreende um conjunto de produtos que juntos se destinam a um segmento de mercado específico ou a uma missão particular (NORTHROP; CLEMENTS, 2007).

O enfoque de linha de produtos é identificar os requisitos comuns que os produtos similares possam ter, para diminuir o tempo na sua construção. Esta ideia não é recente. É uma referência clara às linhas de montagem fordistas do século XX, que trouxeram um novo modo de fabricação dos produtos, a produção em massa (BOTELHO, 2008).

A LPS consiste, então, de partes comuns e variabilidades existentes em uma família de sistemas (BURGARELI, 2009). As características comuns são aquelas partilhadas por todas as aplicações de LPS, e a variabilidade destaca a capacidade de um sistema ser mudado ou customizado, para atender à necessidade do cliente. Nas características comuns, apresentam-se todas as funcionalidades comuns para todos os membros da linha de produto, e na variabilidade, apresentam-se as funcionalidades que são disponíveis para alguns, mas não por todos os membros da linha de produtos de software (SEI, 2015).

2.2. Atividades da LPS

O *Software Engineering Institute* - SEI, por meio da iniciativa *Product Line Practice* - PLP, estabeleceu as atividades fundamentais para uma abordagem de LPS. Essas atividades são o Desenvolvimento do Núcleo de Artefatos, conhecida como Engenharia de Domínio, o Desenvolvimento do Produto, conhecido como Engenharia de Aplicação, e o Gerenciamento de Linha de Produto (NORTHROP; CLEMENTS, 2007).

2.2.1 Engenharia de Domínio

Engenharia de Domínio constitui um conjunto de componentes ou artefatos de softwares e modelos de domínio que possui aplicação em sistemas existentes e futuros, dentro do domínio de aplicação (HIRAMA, 2012). O objetivo da atividade de engenharia de domínio é realizar as tarefas de coletar, organizar e armazenar experiências passadas no desenvolvimento de sistemas ou partes do sistema de um domínio específico na forma de *core assets* e providenciar meios apropriados para reusá-los ao construir novos sistemas (CZARNECKI; EISENECKER, 2005).

2.2.2 Engenharia de Aplicação

A Engenharia de Aplicação constitui em receber os requisitos do usuário e desenvolver novos sistemas de software a partir do reuso dos componentes ou artefatos e modelos de domínio disponibilizado pela Engenharia de Domínio (HIRAMA, 2012).

A Engenharia de Aplicação procura a variabilidade da linha de produto e garante sua correta instanciação das aplicações finais de acordo com as necessidades específicas (REIS, 2014).

2.2.3 Gerenciamento

Essa atividade tem responsabilidade de supervisionar e coordenar a construção da linha de produtos de software e consistem de (NORTHROP; CLEMENTS, 2007):

Gerenciamento Organizacional: origina uma estratégia por meio da criação de um plano de produção que descreve as necessidades da organização e a estratégia para atender às necessidades. Responsável por prover recursos de maneira eficiente para as unidades organizacionais envolvidas na manutenção da linha de produto de software.

Gerenciamento Técnico: inspeciona as atividades relacionadas à Engenharia de Domínio e de Aplicação, garantindo que os artefatos dos *core assets* e os produtos desenvolvidos estejam de acordo com as atividades requeridas. Além disso, define o método de produção e fornece meios para o gerenciamento de projeto do plano de produção.

2.3 Abordagens da Linha de Produto de Software

Algumas abordagens de LPS são existentes na literatura, como *Syntheses/ Reuse Driven Software Process* –RSP (CAMPBEL et al., 1991), *Family Oriented Abstraction Specification and Translation* – FAST (LAI; WEISS, 1999), *Product Line* UML *Based Software Engineering* – PLUS (GOMAA, 2005) e *Feature Oriented Domain Analysis* - FODA (KANG et al., 1990).

Para a construção do método foram consideradas abordagens Baseadas em *Features,* as Baseadas em Família foram descartadas como Synthesis/RSP e FAST, pois a Análise de Domínio trabalha com a Abordagem Baseada em *Features*.

As abordagens Baseada em *Features* escolhidas são PLUS e FODA pois possuem apoio a Análise de Domínio. A PLUS se destaca pois na Modelagem de Caso de Uso explicita os pontos de variação e variantes, permitindo um complemento para a modelagem de *features* (GOMAA, 2005). Já a FODA se destaca por ser responsável para a extração dos requisitos dos softwares para cada produto a ser desenvolvido, por ser uma das pioneiras na Modelagem de *Features* (KANG et al., 1990) e também por ser uma das notações mais utilizadas para a modelagem nas ferramentas existentes.

2.4 Modelo de *Features*

O Modelo de F*eatures* possui como objetivo identificar os aspectos similares e diferentes nos produtos da LPS, esses aspectos são chamados variabilidade (KANG et al., 1990). A identificação e modelagem dessas variabilidades consistem em definir o meio de se construir produtos da mesma família. O resultado dessa modelagem é uma compacta representação dos possíveis produtos de um LPS. O Modelo de *Features* é uma árvore, na qual a raiz representa um conceito e as folhas as suas *features.* Quanto mais detalhado for o modelo, mais produtos são esperados (CZARNECKI; EISERNECK, 2005).

2.5 Análise de Domínio

A Análise de Domínio é uma das fases da Engenharia de Domínio, que analisa as informações disponíveis do projeto de software (documentação, modelagem e códigos), com o objetivo de identificar as *features* que poderão ser reutilizadas e com o intuito de

conseguir conhecimentos do domínio e concretizar uma análise mais especifica dos dados que serão importantes para reutilização no domínio de aplicação a ser construído, gerando como artefato de entrada para a Engenharia de Aplicação o Modelo de *Features*.

Como entradas dessa fase têm-se o Relatório de Análise de Viabilidade da Linha de Produto (visa fazer o levantamento das estimativas de custos para a projeção da família de aplicação, com base no retorno do investimento), Documento de Requisitos (responsável por fazer o levantamento de funcionalidades para servir de base a família.) e Especificações de Caso de Uso (faz a descrição de uma sequência de ações que será realizado pelo sistema para fornecer uma resposta para um ator). Dessas entradas têm-se como saída o Modelo de *Features* (é a modelagem dos requisitos comuns, variáveis e adaptados para a especificação de uma família de domínio de um produto), Glossário do Domínio (é a descrição dos termos e conceitos essências ao domínio especificado) e Documento de Arquitetura de Domínio (define a arquitetura para a família de produto).

Para a especificação de Caso de Uso, faz-se necessário ter feito toda a análise de viabilidade e o documento de requisitos, pois é com o conteúdo desses documentos que a Especificação é criada. Com a Especificação de Caso de Uso, é realizado a modelagem de *features* e com a modelagem pronta são construídos o Glossário de Domínio e o Documento de Arquitetura de Domínio (DIAS; ARAÚJO, 2010).

2.6 Trabalhos relacionados

O trabalho de Brito (2013), implementa novas funcionalidades na ferramenta Heaphaestus. Essa ferramenta possuí funcionalidades para atender o processo de derivação de produtos, como também a análise dos modelos de *features*. Uma outra contribuição desse trabalho é que ele faz uma análise extensiva da escalabilidade dos modelos de *features* com diferentes graus de complexidade. A limitação dessa ferramenta é que todo o processo de criação do modelo é de forma manual e não possui nenhuma ligação com a especificação dos casos de usos, podendo ocasionar um modelo de *features* divergente com o que foi especificado nas documentações anteriormente.

O trabalho de Oliveira (2011), identifica um conjunto de funcionalidades e características que devem compor uma ferramenta para suporte ao Gerenciamento da variabilidade na Engenharia de Requisitos em LPS, além de apresentar uma avaliação das principais soluções. Ao fim foi identificado as funcionalidades mais relevantes das ferramentas, descrição de novas funcionalidades, aprimoramentos e critério de usabilidade para melhorar ou desenvolver uma nova ferramenta. A limitação é que nenhuma das ferramentas citadas e analisadas no trabalho apresenta ligação entre as etapas da Engenharia de Domínio, no qual essas etapas são auto complementares, se feito de forma isolada pode ocasionar divergências entre suas saídas.

3. Método Proposto

Conforme Arruda (2012), a Engenharia de Requisitos é base para o desenvolvimento de software, visto que utiliza algumas etapas dela na Análise de Domínio e que ela pode ser considerada como um pré-requisito para obtenção de sucesso nos produtos a serem desenvolvidos. Se um projeto começar com a modelagem de forma errada, resultará em um desenvolvimento todo comprometido.

Para minimizar esse problema na Modelagem de *Features* este trabalho propõe a geração do Modelo de *Features* de forma automática a partir da Especificação de Caso de Uso, salientando que essa modelagem não é tão simples e que muitos fatores definidos errados podem influenciar nas decisões para realizar as suas atividades (KIM et al., 2006), (GOMAA; SHIN, 2007). Visto também que se a Especificação de Caso de Uso e a Modelagem de *Features* forem feitos de forma manual os erros e divergências entre um e outro será maior, já que a atividade humana está propícia a erros (SOUZA; SILVA, 2013), (IANZEN et al., 2012), (SCHACH, 2009).

Diante as duas abordagens escolhidas, determinou-se que, para a entrada (Diagrama e Especificações de Caso de Uso) do método proposto, serão utilizados os requisitos da abordagem PLUS que faz a adaptação do modelo de UML para a especificação de Caso de Uso de Linha de Produto de Software, utilizando estereótipos de identificação para atividades variáveis e comuns.

A saída do método proposto será o Modelo de *Features* que é um dos princípios da abordagem FODA, conforme visto na seção 2, e terá como base nos estereótipos que ela define para cada *feature* existente na família de domínio que são: obrigatória, opcional, alternativa: exclusiva e/ou inclusiva. O processo geral do método proposto é ilustrado na Figura 1.

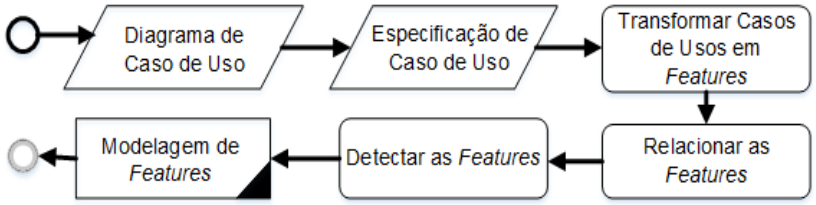


Figura 1. Processo Geral do Método

Para entendimento do método será descrito o modelo computacional para as suas etapas, que são: Entrada, Extração e Saída, descrevendo o que é armazenado de informações em cada processo executado e como é realizado, como mostra a Figura 2.

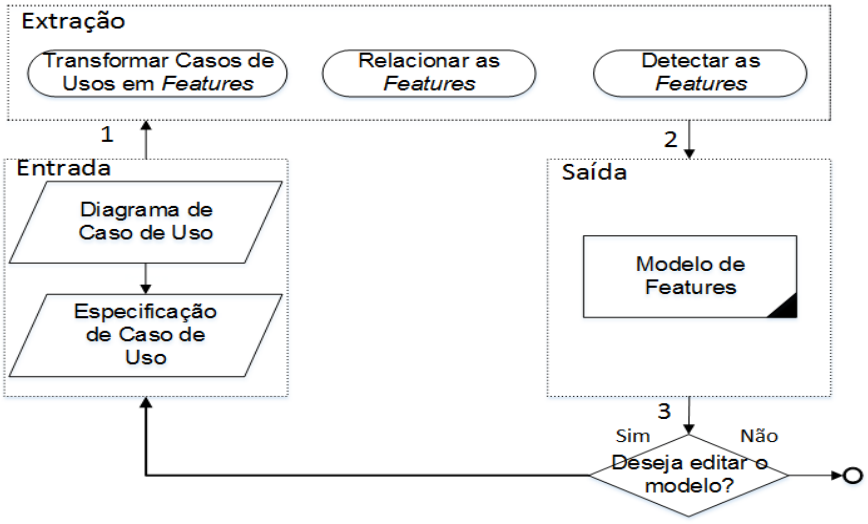


Figura 2. Modelo Completo

3.1 Entrada – Diagrama e Especificação de Caso de Uso

Nesta seção, são apresentadas as duas entradas necessárias para o método proposto, que são elas o Diagrama e a Especificação de Caso de Uso, conforme processo ilustrado na Figura 2.

3.1.1 Diagrama de Caso de Uso

Para o diagrama será realizado a Modelagem, conforme a abordagem PLUS, que, para (GOMAA, 2005), os tipos de casos utilizados pela abordagem são:

Comum: são utilizados em todos os produtos da família do domínio; Opcional: são utilizados em alguns produtos da família do domínio; Alternativo: são utilizados em diferentes versões requeridas por diferentes produtos da família do domínio.

No diagrama serão usados os estereótipos para diferenciar os diferentes tipos de casos de uso, são eles <<comum>>, <<opcional>> e <<alternativo>>.

Tendo em vista essas regras para a Modelagem do Diagrama de Casos de Uso, a Figura 3 é ilustrado como será a entrada do método proposto com um Diagrama de Caso de Uso para uma família de produto de um aplicativo de Câmera do Android.

Neste exemplo para o caso do projeto de um sistema para uma família de aplicativos de câmera, o Diagrama de Caso de Uso identifica casos de usos comum, opcionais e alternativos. O caso de uso Capturar Fotografia, Configurar Câmera e Visualizar Mídia são comum, pois em todos os aplicativos de câmera devem possuir essas características. Os casos de usos Definir Localização e Criar Vídeo são opcionais, apenas alguns aplicativos possuem essas funcionalidades.

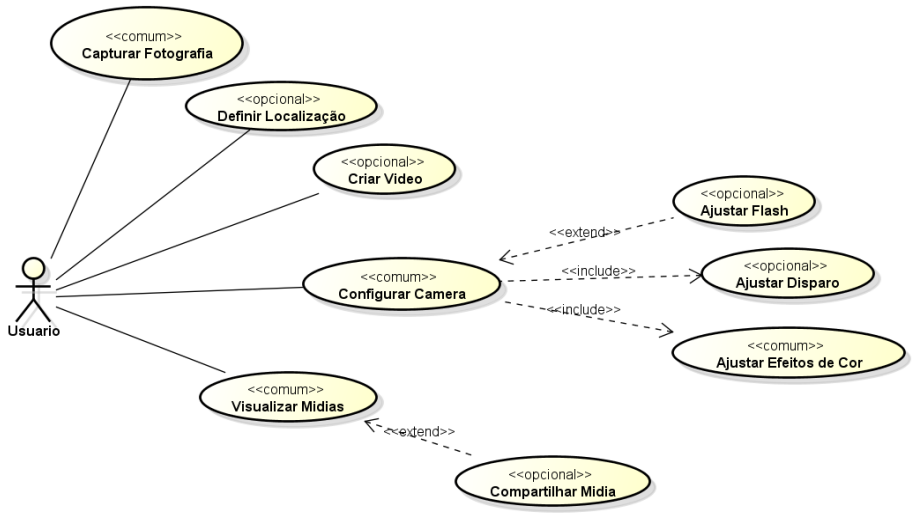


Figura 3. Exemplo de Diagrama de Caso de Uso para Entrada do método

3.1.2 Especificação de Caso de Uso

Com a modelagem do Diagrama de Caso de Uso concluída faz-se necessária a Especificação de cada Caso de Uso no ponto de vista de linha de produto de software, no qual será detalhado a especificação dos seguintes pontos:

Nome de caso de uso: faz a identificação do caso de uso descrito, que já será uma informação *default,* por meio da modelagem do diagrama;

Categoria: identifica a sua relação com a classificação (comum, opcional ou alternativo), que também será uma informação *default*, por meio da modelagem do diagrama;

Dependência: identifica a que caso de uso ele depende, que será uma opção de escolha para o engenheiro a partir dos casos modelados;

Resumo: descreve o que o caso de uso faz, que será um campo de texto;

Ator: lista o(s) ator(es) que participa(m) do caso de uso descrito; é uma informação *default* da modelagem de Diagrama;

Pré-condição: descreve a condição em que o sistema deve estar antes da execução do caso descrito, que é um campo de texto;

Descrição da Sequência: define a sequência de passos para execução do caso de uso descrito, que é um campo com opções de *input* dependendo da sequência de cada caso; para cada sequência terá a opção de marcar se há ponto de variação ou não;

Ponto de Variação lista as variações encontradas na Descrição da Sequência, no qual já irá aparecer como *default* por meio da descrição da sequência, sendo necessário informar apenas as variantes para cada ponto de variação e se é inclusiva ou exclusiva.

Como descrito no modelo computacional, todos esses pontos que não são apresentados como *default*s são informados em um formulário para complementar a especificação dos casos de usos.

Na Tabela 1, tem-se o formato de como se deve colocar a especificação de Casos de Uso.

Tabela 1 - Modelo para especificação de Casos de Usos

Nome do Caso de Uso:	Capturar Fotografia
Categoria:	Comum
Dependência	-
Resumo:	Realiza a captura de imagens
Ator:	Usuário
Pré-Condição:	Aplicativo está aberto
Descrição da Sequência:	
1. O usuário escolhe qual câmera para realizar a captação da imagem. 2. Ponto de Variação 1 – Escolher Câmera 3. Usuário escolhe o zoom para captação da imagem 4. Ponto de Variação 2 – Escolher o zoom 5. Usuário clica no botão para capturar imagem	
Pontos de Variação:	
1. Escolher câmera (Sequência 2) O usuário realiza a escolha entre câmera frontal e traseira. 2.Escolher zoom (Sequência 4) O usuário escolhe o zoom que pode variar de 1x a 8x.	

Com todos os detalhes levantados na Especificação e Digrama de Caso de Uso, o próximo passo é a Extração de *Features* que a partir dele, chega ao objetivo do método que é a geração do Modelo de *Features*.

3.2 Extração das *Features*

Nesta subseção, são apresentadas as três etapas necessárias para a extração do Modelo de *Features*, que são: Transformar Caso de Uso em *Features*, Relacionar as *Features* e Detectar as *Features*, conforme processo ilustrado na Figura 2.

3.2.1 Transformar Caso de Uso em *Features.*

A transformação dos Casos de Usos em *Features* é realizado por meio de consulta no banco de dados de cada nome de caso, conforme informado no modelo computacional, e com os dados dessa consulta é transformado, via codificação da aplicação, os casos de usos em *features*.

3.2.2 Relacionar as *Features*

Nesse processo é feita a identificação da Dependência, conforme descrito no modelo computacional.

Por meio da consulta no banco de dados, os casos de usos transformados que não possuem nenhuma informação como Dependência, são relacionados diretamente na raiz, no qual a raiz é o nome definido para a aplicação da LPS a ser construída, os que possuem dependência é relacionada com o caso de uso informado na especificação, que também foram previamente transformados em *feature. T*odo esse processo é realizado por meio de codificação da aplicação.

3.2.3 Detectar as *Features*

A classificação das *features* detectadas nos diagramas e nas especificações de cada caso tem como base a abordagem FODA, que possui como notação os seguintes relacionamentos: obrigatória, opcional, alternativa: exclusiva e/ou inclusiva, notação bem parecida com a abordagem PLUS.

As *features* obrigatórias representam características indispensáveis da linha de produto de software. Sua identificação é através de casos de uso comuns. Nestes casos de usos, também é possível identificar as variabilidades, que podem ser mandatórias inclusivas ou exclusivas, que são *features* alternativas que possuem dependência de alguma *feature* obrigatória.

As alternativas são as que possui uma relação exclusiva na linha de produto, e as inclusivas são as características adicionais incluídas na linha de produto, identificadas através de pontos de variações na especificação de cada caso de uso.

As *features* opcionais, são as que possui uma relação opcional, ou seja, são incluídas de acordo com o produto de software; são identificadas por meio de casos de usos opcionais.

Para detectação e geração do tipo de *features* são utilizados os campos definido na Especificação de Caso de Uso como estereótipo, dependência, pré-requisito e ponto de variação armazenada no banco de dados, sendo avaliadas algumas informações para cada geração específica.

Para *features* obrigatórias e opcionais a detectação será transformando diretamente os dados de estereótipo do banco de dados.

Para *features* mandatórias inclusiva ou exclusiva, é necessário verificar a informação no banco sobre a Dependência da sua *feature* e verificar se ela é obrigatória para que ela se torne mandatória inclusiva ou exclusiva.

Para *features* alternativas, é necessário a verificação do campo ponto de variação, no qual é definido qual a variação de cada *feature*.

Neste caso, para realizar a Extração de *Features*, houve a definição do seguinte processo:

Como descrito anteriormente, a entrada é a Especificação e o Diagrama de Caso de Uso para a extração das *features* cada uma notação foi definida como uma fase que são elas: Detectar *Features* Obrigatórias, Detectar *Features* Opcionais, Detectar *Features* de Ponto de Variação e Detectar *Features* Variantes, conforme descrição a seguir:

Fase 1 - Detectar *Features* Obrigatórias
a. Listar casos de uso comuns. Caso de Uso considerado como fundamental, ou seja, é sempre requerido pelo sistema em questão. Ex.: para a LPS do aplicativo de câmera foi detectado como *features* obrigatória os seguintes casos: Capturar Fotografia, Configurar Câmera, Visualizar Mídia;

b. Gerar *features* obrigatórias, por meio dos Casos de Usos comuns encontrados. Conforme descrito anteriormente, é feita uma comparação com a variável do estereótipo informado no diagrama, e transformado diretamente para casos obrigatórios os que estão definidos como comum. É necessário listar os requisitos que devem existir para implementar as funcionalidades. Ex.: os requisitos que devem existir são: câmera, zoom.

Fase 2 - Detectar *Features* Opcionais
a. Listar Casos de Uso Opcionais. São considerados não fundamentais para a função principal do sistema. Esta funcionalidade, comumente é representada por uma sequência de interações isoladas da função principal. Ex.: para a LPS do aplicativo de câmera foi detectado como *features* opcionais os seguintes casos: Definir Localização, Criar Vídeo;
b. Gerar *features* opcionais, por meio do Caso de Uso de Opcionais, para as opcionais, é feito o mesmo processo do obrigatório verificando a variável do estereótipo informado no diagrama e transformando diretamente para casos opcionais os que estão definidos como opcional. É necessário listar os requisitos que realizam a funcionalidade não fundamental para a função principal do sistema. Ex.: os requisitos que devem existir são: para o caso de uso definir a localização GPS ou 3G.

Fase 3 - Detectar *Features* Pontos de Variação
a. Listar os requisitos encontrados nos casos de usos comuns identificados na fase 1, que possuem funções fundamentais e que apresentam variações. Estes requisitos são *features* de variações mandatório inclusivo ou exclusivo, verificando os dados informados na descrição da sequência como variantes na especificação de cada caso de uso. Ex.: para a LPS do aplicativo de câmera foi detectado como *features* mandatórias inclusiva: ajustar efeito de cor;

b. Listar os requisitos encontrados nos casos de usos opcionais identificados na fase 2, que possuem funções opcionais e que apresentam variações, verificando os dados

informados na descrição da sequência como variantes na especificação de cada caso de uso. Estes requisitos são *features* de variações opcional. Ex.: para a LPS do aplicativo de câmera foi detectado como *features* de variações opcional: ajustar flash, ajustar disparo, compartilhar mídia, definir localização.

Fase 4 – Detectar *Features* Variantes (alternativas)

Listar as variações para os requisitos detectados no ponto de variação na fase 3, verificando as informações do ponto de variação na especificação de cada caso de uso. Ex.: As Alternativas Inclusiva para a LPS do aplicativo de câmera são: Câmera: frontal e traseira; Zoom: 1x a 8x; Localização: GPS, Dados Móveis; *Flash*: automático e manual; Disparo: automático, 3 segundos, 5 segundos; Compartilhar: Facebook, whatsapp; Efeitos de Cor: automático, preto e branco.

Concluindo o processo de Detectação e Geração de *Features*, por meio do Diagrama e Especificação dos Casos de Usos de Linha de Produto de Software, realiza-se uma consulta no banco de dados para geração de um relatório que irá considerar todas as informações dos casos de usos e *features* salvas no banco de dados. Esse relatório (quadro) faz a relação dos casos de usos do produto de software com as *features* identificadas que é a saída do processo de extração para que o Engenheiro de Software possa fazer a análise se todas as funções da linha foram definidas nos casos e modelada, como também ajudar no processo de Modelagem de *Features*.

Para Gomaa (2005), as *features* possuem associações de relacionamentos com os casos de usos de muitos para muitos. Muitas *features* podem ser geradas por um caso de uso. Como também uma mesma *feature* pode ser extraída de mais de um caso de uso.

Na Tabela 2, é listado o exemplo de como é a descrição das *Features* geradas do Diagrama e Especificação de Caso de Uso.

Tabela 2 - *Features* geradas do Diagrama de Caso de Uso

Casos de Uso	Estereótipo dos Casos de Usos	Dependência	*Features*	Variações	Classificação das *Features*
Capturar Fotografia	Comum	Capturar Fotografia	Frontal	Câmera (mandatória)	Mandatória inclusiva
Compartilhar Mídia	Opcional	-	Compartilhar Mídias	-	Opcional
Compartilhar Mídia	Alternativo	Compartilhar Mídia	Facebook	Compartilhar (opcional)	Inclusiva
Configurar Câmera	Comum	-	Configurar Câmera	-	Obrigatória

Conforme Tabela 2, por meio da coluna casos de uso que identifica cada caso de uso do diagrama, a coluna Estereótipo dos Casos de Usos que faz representação da classificação dos casos de usos, na coluna Dependência identifica a que caso ele depende, na coluna *Features* representa a referência de cada caso de uso, na coluna Variações identifica quais as variações para a *feature* e a coluna Classificação das *Features* que faz a classificação de acordo com o método FODA para cada *feature.*

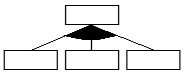
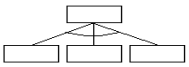
Ao concluir toda a identificação das *features* e geração da tabela, é realizado a modelagem de *features*.

3.2.4 Saída - Modelagem de Feature

Cada produto em LPS é diferenciado de acordo com as suas características, no qual são especificados por meio do modelo de *features.* Termo referenciado por Kang et al. (1990), na abordagem FODA, que busca representar relações e características no domínio da família de sistemas.

Na Tabela 3 é listada a classificação das f*eatures* para a Modelagem de *Features*.

Tabela 3 - Classificação das Features para Modelagem

Classificação das *Features*	Modelo
Feature Raiz	
Feature Obrigatória	
Feature Opcional	
Feature Variante Inclusivo	
Feature Variante Exclusivo	

Conforme Tabela 3, por meio das colunas Classificação das *Features,* que identifica o tipo de *feature,* e a coluna Modelo, que faz representação gráfica de acordo com a classificação, todo o processo de detectar as *features* na representação gráfica será realizada por codificação da aplicação.

Para a Modelagem de *Features*, são utilizados retângulos para representar as *features*, e símbolos para as classificações dos estereótipos que são descritos como circunferência preenchida na cor preta para as *features* obrigatórias, circunferência sem preenchimento de cor para as *features* opcionais, cone com preenchimento de cor preta para as *features* variantes inclusivas, e cone sem preenchimento de cor para as *features* exclusivas.

Para a geração da Modelagem de *Features* são necessárias as informações das *features* com as suas dependências, variações e classificações, conforme foi definido no Tabela 2 na seção de Extração, transformando essas informações conforme classificação gráfica das *features*, estabelecida na Tabela 3.

A modelagem é feita conforme ilustração da Figura 4. Toda essa transformação será realizada por meio de codificação da aplicação.

A Figura 4 representa a LPS de um aplicativo de Câmera do Android que define como *feature* obrigatória todos os requisitos informados com a circunferência preta. Como *feature* opcional todos os requisitos informados pela circunferência branca. Como *feature* variável inclusivo todos os requisitos informados nos cones pretos.

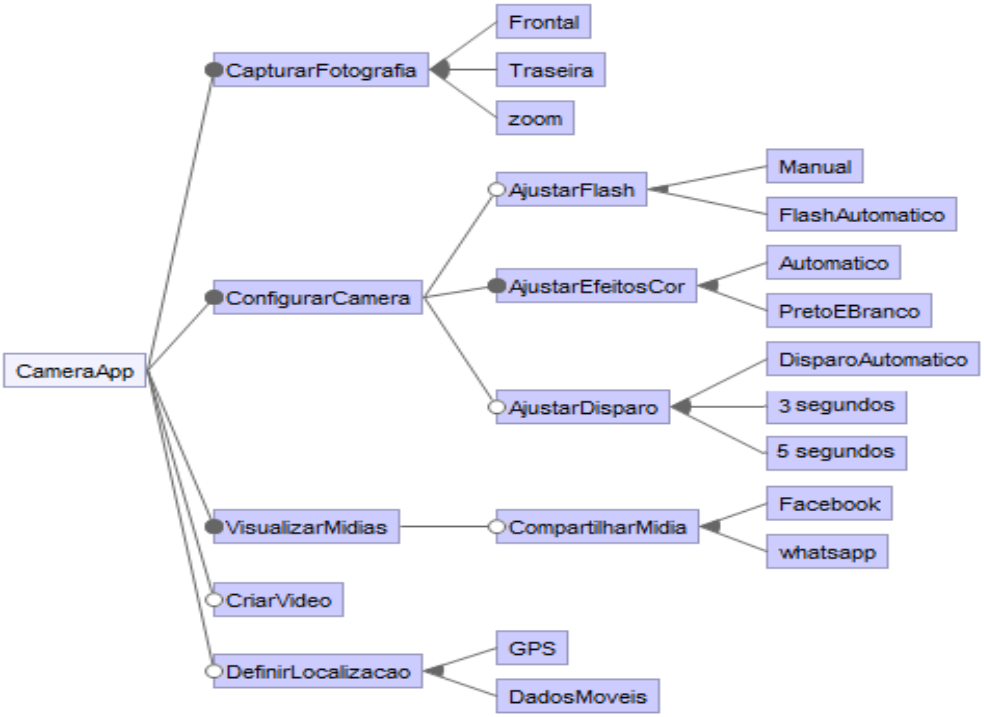


Figura 4. Exemplo do Modelo de Features

3.2.5 Edição do Modelo de *Feature*

Se o Engenheiro de Software, ao concluir todo o processo de modelagem do Diagrama e Especificação de Casos de Usos e todo o processo com o Modelo de *Features*, perceber que faltou alguma informação ou que ele definiu alguma coisa errada, então ele tem a opção de editar a *feature* específica ou até mesmo o modelo completo, no qual ele voltará para a Entrada do método para ajustar o Diagrama e/ou Especificação de Casos de Usos e todo o processo de extração do método e refeito.

4. Conclusão

A Linha de Produto de Software estabelece como meta o reuso para a criação e geração de sistemas de uma família específica. Para conseguir isso, tem-se as atividades de LPS que são elas Engenharia de Domínio, Engenharia de Aplicação e Gerenciamento, que são responsáveis pela organização e definição das partes comuns e variáveis dos sistemas, a geração do software e o funcionamento ordenado destas atividades, garantindo um processo de produção mais econômico, com qualidade e eficaz.

A qualidade e conformidade da LPS estão diretamente ligadas a Modelagem de *Features*, pois é com ela que se realiza a derivação dos produtos da linha. Portanto, a modelagem é considerada uma atividade instrutiva devido à dificuldade para determinar as várias decisões para identificar, especificar e implementar a variabilidade de cada linha.

Com isso, o objetivo deste artigo foi de propor um método para Modelagem de *Feature* de forma automática na fase de Análise de Domínio, com o intuito de

minimizar erros na modelagem de *features*, dado que as ferramentas realizam a modelagem de forma manual, podendo ocasionar inconsistências entre os artefatos de entrada e o de saída, já que o ser humano é propício a erros. Para isso, foi necessário compreender como ocorre a Análise de Domínio e como as abordagens para LPS podem ajudar nessa análise de domínio.

Para realizar a pesquisa houve algumas dificuldades como a não padronização dos termos relacionados à LPS para cada abordagem, o que levou a necessidade de se estabelecer uma transformação dos termos de uma abordagem para outra, e se torna um padrão para utilização neste trabalho.

A diversidade de abordagens para LPS fez com que se realizasse uma análise para selecionar as mais apropriadas ao objetivo do trabalho, na qual foram selecionadas duas. Foi necessário estuda-las com mais ênfase, para conseguir maiores benefícios nas características de cada uma para aplicar no método proposto.

O diferencial do método proposto em relação as outras ferramentas que realizam o processo de Modelagem de *Features,* é que ele realiza a modelagem automática por meio da Especificação e Modelagem dos Casos de Usos, conduzindo a uma maior confiabilidade no modelo produzido.

É válido salientar que o método proposto de Modelagem de *Features* não pretende solucionar os problemas apresentados por outras ferramentas ou técnicas, mas proporcionar uma maior confiabilidade na produção das linhas de produtos, como uma alternativa para o desenvolvimento da Análise de Domínio.

5. Referências

Almeida, L. T.; Miranda, J. M. (2010) Código Limpo e seu Mapeamento para Métricas de Código Fonte. Universidade de São Paulo. Disponível em: http://www.ime.usp.br/~cef/mac499-10/monografias/lucianna-joao/arquivos/monografia.pdf, Maio.

Arruda, N. S. (2012) Engenharia de Requisitos-como Previnir e Reduzir Riscos. Universidade do Rio de Janeiro. Rio de Janeiro.

Botelho, A. (2008) Do Fordismo à Produção Flexível: O espaço da indústria num contexto de mudanças das estratégias de acumulação do capital. São Paulo: Editora Annablume.

Burgareli, L. A. (2009) Gerenciamento de Variabilidade de Linha de Produtos de Software com utilização de Objetos adaptáveis e reflexão. Universidade de São Paulo. São Paulo.

Campbell, G. J. O. et al. (1991) Synthesis Guidebook. Hemdon. Technical Report SPC-91122-MC. Virginia, USA.

Czarnecki, K.; Eisenecker, U.W. (2005) Generative programming methodos, techniques, and applications. Cana Addison Wesley. Disponível em: http://gsd.uwaterloo.ca/sites/default/files/2005-czarnecki-model-driven-software-product-lines.pdf, Abril.

Dias, N. K. B.; Araujo, M. A. P. (2010) Boas Práticas na Engenharia de Requisitos. Universidade do Rio de Janeiro. Rio De Janeiro.

Geriner, P. T. et al. (2012) Software Engineering Economics and Declining Budgets. São Paulo: Editora Springer.

Gomaa, H. (2005) Designing software product lines with UML from use cases to pattern-based software architectures. Universidad de Boston. Boston.

Hirama, K. (2012) Engenhria de Software, Qualidade e Produtividade com Tecnologia. São Paulo: Elsevier Editora.

Ianzen A. et al. (2012) Definição de Escopo em Linhas de Produto de Software: uma abordagem semi-automática por meio de anotação linguística. Universidade de Curitiba. Curitiba.

Jacobson, I. et al. (1999) The Unified Software Development Process. Universidade de Boston. Boston.

Kang, K. C. et al. (1990) Feature-Oriented Domain Analysis (FODA) Feasibility Study. Pittsburgh, Pennsylvania.

Kim, Y. et al. (2006) Managing variability for software productline. In Software Engineering Research, Management And Aplicattions Coference – Sera, 4., 2006, Seatle, WA. Proceedings. Washington: IEEE Computer Society.

Lai, C. T. R.; Weiss, D. M. (1999) Software Product-Line Engineering: A Family-Based Software Development Process. Addison-Wesley, 1999.

Reis, J. N. (2014) Análise comparativa de tecnologias de programação em sistemas de software derivados de linhas de produtos de software. Universidade Mineira. Minas Gerais.

Northrop, L. M.; Clements, P.C. (2007) A framework for software product line practice. Version 5.0. Pittsbur. Software Engineering Institute Disponível em: http://www.sei.cmu.edu/productlines/tools/framework/, Abril.

Oliveira, D. R. F. Um Estudo sobre Gerenciamento de Variabilidade de Requisitos em Linha de Produto de Software. Universidade do Pernambuco. Caruaru, 2011.

Schach, S. R. (2009) Os Paradigmas Clássico e Orientado a Objetos. Engenharia de Software – 7 ed. São Paulo: Editora Elsiever.

SEI - SOFTWARE ENGINEERING INSTITUTE. (2015) Capability Maturity Model for Software (SW-CMM). Disponível em: <http://www.sei.cmu.edu/cmm, Maio.

Souza, F. R. R.; Silva, P. C. (2013) Software Production Lines: an Organazation Model for Software Factories for the Reuse of the Humancomputer Interface. São Paulo: Contesci.