UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS QUIXADÁ

BACHARELADO EM SISTEMAS DE INFORMAÇÃO

AMANDA OLIVEIRA DE SOUSA

PREMIER: UM PROCESSO DE ELICITAÇÃO E MODELAGEM DA VARIABILIDADE DE REQUISITOS PARA LINHAS DE PRODUTO DE SOFTWARE DINÂMICAS

QUIXADÁ – CEARÁ

2016

AMANDA OLIVEIRA DE SOUSA

PREMIER: UM PROCESSO DE ELICITAÇÃO E MODELAGEM DA VARIABILIDADE DE REQUISITOS PARA LINHAS DE PRODUTO DE SOFTWARE DINÂMICAS

Monografia apresentada no curso de Sistemas de Informação da Universidade Federal do Ceará, como requisito parcial à obtenção do título de bacharel em Sistemas de Informação. Área de concentração: Computação.

Orientadora: Prof^{a}. Dra. Carla Ilane Moreira Bezerra

QUIXADÁ – CEARÁ

2016

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

S696p Sousa, Amanda Oliveira de.
PREMIeR: Um Processo de Elicitação e Modelagem da Variabilidade de Requisitos para Linhas de Produto de Software Dinâmicas / Amanda Oliveira de Sousa. – 2016.
100 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Sistemas de Informação, Quixadá, 2016.
Orientação: Profa. Dra. Carla Ilane Moreira Bezerra.

1. Engenharia de Linha de Produto de Software. 2. Análise de Requisitos. 3. LPS - Gerenciamento da Variabilidade. 4. Software - Reutilização. I. Título.

CDD 005

AMANDA OLIVEIRA DE SOUSA

PREMIER: UM PROCESSO DE ELICITAÇÃO E MODELAGEM DA VARIABILIDADE DE REQUISITOS PARA LINHAS DE PRODUTO DE SOFTWARE DINÂMICAS

Monografia apresentada no curso de Sistemas de Informação da Universidade Federal do Ceará, como requisito parcial à obtenção do título de bacharel em Sistemas de Informação. Área de concentração: Computação.

Aprovada em:

BANCA EXAMINADORA

Prof^{a}. Dra. Carla Ilane Moreira Bezerra (Orientadora)
Campus Quixadá
Universidade Federal do Ceará – UFC

Prof. Dr. Marcos Antonio de Oliveira
Campus Quixadá
Universidade Federal do Ceará - UFC

Prof. Msc. Camilo Camilo Almendra
Campus Quixadá
Universidade Federal do Ceará - UFC

Prof. Dr. João Bosco Ferreira Filho
Departamento de Computação
Universidade Federal do Ceará - UFC

A Deus.

Aos meus pais, irmãos e amigos.

AGRADECIMENTOS

A Prof[a]. Dra. Carla Ilane Moreira Bezerra, pela excelente orientação e pela paciência.

Aos meus pais, pela educação, pelo amor e dedicação e que apesar das dificuldades sempre confiaram na minha capacidade, mesmo que eu as vezes não o fizesse.

Aos meus irmãos, pelo carinho e pelas ligações divertidas ao telefone que aliviaram meus dias escuros.

Aos professores participantes da banca examinadora, Prof. Msc. Camilo Almendra, Prof. Dr. João Bosco Ferreira Filho e Prof. Dr. Marcos Oliveira pelo tempo, pelas valiosas colaborações e sugestões.

Aos meus tios Josenias e Saliete que sempre torceram pelo meu sucesso e me apoiaram nos momentos de necessidade.

Ao meu eterno amigo Flávio pela força e parceria durante toda a graduação.

Aos meus colegas do grupo PET-SI por fazerem meus dias mais alegres durante essa jornada, em especial Danrley e Daniel que me presentearam com sua imensa amizade e carinho.

Ao tutor Davi Romero e o co-tutor Lucas Ismaily pelos valiosos aconselhamentos no tempo que fiz parte do grupo PET-SI.

Aos meus amigos Anderson e Luan, pelo companheirismo e motivação na realização deste trabalho.

“A persistência é o menor caminho do êxito”.

(Charles Chaplin)

RESUMO

Linhas de Produto de Software Dinâmicas (LPSD) constituem uma extensão de Linhas de Produto de Software (LPS), que por sua vez representam um conjunto de sistemas que compartilham um conjunto de ativos comuns e satisfazem as necessidades de um determinado segmento de mercado ou missão. As LPSDs têm sido utilizadas como estratégia para produzir sistemas complexos, que requerem adaptações em tempo de execução, tais como sistemas sensíveis ao contexto. Para o desenvolvimento de uma LPSD, os engenheiros de domínio devem se atentar a identificação e modelagem de aspectos dinâmicos e ao conceito de variabilidade. Entretanto, ainda há lacunas relacionadas a técnicas e abordagem que promovam elicitação e modelagem de requisitos para LPSDs. Nesse contexto, este trabalho tem o objetivo de apresentar um processo de elicitação, modelagem e gerenciamento da variabilidade para LPSDs. O processo proposto no presente trabalho utiliza técnicas de requisitos tradicionais aliado a técnicas direcionadas para representação em LPSDs dos requisitos, *features*, contextos e adaptações. Para a definição do processo foram realizados estudos não sistemáticos da literatura, a fim de identificar técnicas, abordagens e ferramentas utilizadas na Engenharia de Requisitos de sistemas do domínio de LPSDs. A validação do processo elaborado foi obtida por meio de um estudo de observação da aplicação do processo PREMIeR e indica a possibilidade de identificar os elementos de uma LPSD por meio de sua aplicação.

Palavras-chave: Linhas de Produto de Software Dinâmicas. Elicitação de Requisitos. Gerenciamento da Variabilidade. Modelagem de Requisitos.

ABSTRACT

Dynamic Software Product Lines (LPSD) are an extension of Software Product Lines (LPS), which in turn represent a set of systems that share a set of common assets and satisfy the needs of a given market segment or mission . LPSDs have been used as a strategy to produce complex systems, which require run-time adaptations, such as context-sensitive systems. For the development of a LPSD, domain engineers must pay attention to the identification and modeling of dynamic aspects and to the concept of variability. However, there are still gaps related to techniques and approaches that promote requirements elicitation and modeling for LPSDs. In this context, this paper aims to present a process of elicitation, modeling and variability management for LPSDs. The process proposed in the present work uses techniques of traditional requirements coupled with techniques directed to representation in LPSDs of the requirements, contexts and adaptations. For the definition of the process, non - systematic studies of the literature were carried out in order to identify techniques, approaches and tools used in Requirements Engineering of LPSD domain systems. The validation of the elaborated process was obtained by means of an observation study of the application of the PREMIeR process and indicates the possibility of identifying the elements of a LPSD through its application.

Keywords: Dynamic Software Product Lines. Requirements Elicitation. Varaibility Management. Requirements Modeling.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

BPMN	*Business Process Management*
LPS	Linha de Produto de Software
LPSD	Linhas de Produto de Software Dinâmica
UFC	Universidade Federal do Ceará
DyMMer	*Dynamic Feature Model Tool Based on Measures*

SUMÁRIO

1 INTRODUÇÃO

Devido à necessidade de se produzir sistemas de software com qualidade, baixo custo e curto prazo de entrega, indústria e academia têm buscado e desenvolvido técnicas e abordagens para o processo de desenvolvimento de software visando alcançar estes objetivos (SOMMERVILLE, 2011). Na academia e no meio organizacional uma estratégia bem conhecida é o reuso, que torna possível a reutilização de produtos de software já desenvolvidos e de processos utilizados durante o desenvolvimento de software (PRESSMAN, 2011).

Dentre as várias maneiras de implementar o reuso de software, existe a abordagem de Linhas de Produto de Software (LPS), um paradigma que se aplica em torno de um domínio contendo uma base comum que é adaptada para clientes diferentes (SOMMERVILLE, 2011). Northrop et al. (2007) definem LPS como um conjunto de sistemas de software que compartilham um conjunto de características comuns que satisfazem um segmento de mercado ou missão e que são desenvolvidos a partir de um conjunto de ativos principais de maneira predeterminada.

Em LPS, a reusabilidade é maximizada e apoia amplamente o processo de desenvolvimento. Ela não tem foco apenas no código, se estendendo desde a fase de engenharia de requisitos até a fase de testes. Dessa forma, os artefatos reusáveis formam uma estrutura para a Linha de Produto (LINDEN; SCHMID; ROMMES, 2007). A maneira como o reuso é conduzido em uma LPS gera melhoria da qualidade, aumenta a confiabilidade, reduz o esforço na manutenção do software e torna os ciclos de desenvolvimento mais curtos (POHL; BÖCKLE; LINDEN, 2005).

Contudo, as Linhas de Produto de Software tradicionais não dão suporte a alguns sistemas de maior complexidade, como softwares sensíveis ao contexto. Para apoiar esse tipo de sistema, surgiram as Linhas de Produto de Software Dinâmicas (LPSDs), uma abordagem variante das LPSs que possui a reconfiguração dinâmica e o gerenciamento de variabilidade em tempo de execução como aspectos exclusivos (CAPILLA et al., 2014). As LPSDs se atentam para o fato de o produto sofrer variações no momento da execução, processo que deve ser gerenciado para que o produto obtenha uma configuração válida e útil para o usuário (HALLSTEINSEN et al., 2008).

Linhas de Produtos de Software tradicionais e dinâmicas abrangem tanto a Engenharia de Domínio quanto a Engenharia de Aplicação em seus ciclos de vida. A Engenharia de Domínio está relacionada com o desenvolvimento dos principais ativos da linha de produto e abrange por exemplo, a análise do domínio e a análise do contexto. A Engenharia

de Aplicação é responsável pelas atividades que se referem ao desenvolvimento da aplicação em si, como a implementação do produto e a configuração do produto (NORTHROP et al., 2007).

Alguns dos principais ativos desenvolvidos pela Engenharia de Domínio estão presentes na Engenharia de Requisitos, sendo esta responsável por desenvolver e manter todos os requisitos importantes para o domínio, incluindo a identificação de variabilidades, e o gerenciamento das *features* (LINDEN; SCHMID; ROMMES, 2007). A variabilidade é o aspecto mais marcante das linhas de produto de software e consiste na habilidade de alterar ou customizar o software a partir da descoberta de um ponto de variação. Melhorar a variabilidade do sistema implica em torná-lo mais flexível para certos tipos de mudanças (GURP; BOSCH; SVAHNBERG, 2001). Dessa forma é possível perceber o impacto que a execução das atividades da Engenharia de Requisitos tem nas LPSDs.

A identificação dos elementos de uma LPSD não é trivial, uma vez que envolve elementos estáticos e elementos capazes de sofrer alterações em tempo de execução. Devido a dificuldade de prever ou estimar o comportamento destes elementos, é necessário estudar e analisar técnicas afim de obter uma identificação mais completa possível desses elementos. Nesse sentido, a elicitação possui grande importância, uma vez que todas as atividades de modelagem seguintes serão baseadas nos elementos identificados nesta atividade.

No que diz respeito a modelagem em LPSDs, o modelo de *features* é um dos ativos mais importantes e o mais utilizado para representar a variabilidade (SILVA et al., 2016). Para produzir um modelo de *features* consistente e capaz de auxiliar o engenheiro de domínio na análise e entendimento do domínio, se faz necessário o detalhamento e modelagem os elementos do domínio e todos os conceitos que impactarão na construção do modelo de *features* e suas derivações. O modelo de *features* também é usado no gerenciamento da variabilidade (CHEN; BABAR; ALI, 2009), e permite a visualização das diferentes derivações de aplicações e o comportamento de *features* quando o sistema é inserido em um contexto.

Quando se trata de LPSD, o processo de Engenharia de Requisitos deve realizar atividades adicionais que apoiem, por exemplo, as mudanças de requisitos do produto em tempo de execução e a análise da variabilidade (POHL; BÖCKLE; LINDEN, 2005). Para Linhas de Produto de Software tradicionais existem processos de Engenharia de Requisitos específicos para atender as necessidade dessa abordagem (NEIVA et al., 2010; SILVA; BORBA; CASTRO, 2010). No entanto, não foi identificado na literatura um processo de engenharia de requisitos para LPSDs (SILVA et al., 2016).

Silva et al. (2016) verificaram, por meio de uma revisão sistemática da literatura, quais são as atividades de Engenharia de Requisitos e Gerenciamento de Variabilidade utilizadas em LPSDs. Além disso, verificaram a inexistência de um processo de engenharia de requisitos para LPSD. Dessa forma, este trabalho tem como objetivo o desenvolvimento de um processo de Engenharia de Requisitos que incorpore atividades compatíveis com a abordagem de LPSDs, se concentrando apenas nas fases relacionadas as atividades de elicitação de requisitos, modelagem de requisitos e gerenciamento de variabilidade. Esse trabalho se destina a engenheiros de domínio, pesquisadores e profissionais que trabalham com LPSDs. Como contribuição, espera-se que a abordagem de LPSD torne-se mais completa no que diz respeito a Engenharia de Requisitos da Engenharia de Domínio, agrupando técnicas e atividades compatíveis na composição desse processo.

2 TRABALHOS RELACIONADOS

Esta Seção apresenta trabalhos realizados na área de Linhas de Produtos de Software e Linhas de Produto de Software Dinâmicas e que estão relacionados ao trabalho proposto, apresentando as diferenças e similaridades entre esses estudos e o presente trabalho.

Neiva et al. (2010) elaboram um processo de Engenharia de Requisitos específico para LPSs. O processo contém a identificação das atividades, modelos e técnicas de Engenharia de Requisitos que são adotadas em abordagens de LPSs e como essas abordagens lidam com a variabilidade. O processo de Engenharia de Requisitos é dividido três atividades: Modelagem do Escopo, Definição dos Requisitos e Definição dos Casos de Uso. Para cada uma dessas atividades foi estabelecido um conjunto de tarefas, artefatos de entrada e saída, e produtos de trabalho. Esse processo apenas trata a Engenharia de Requisitos no contexto de LPSs tradicionais, não se concentrando em LPSDs. A Engenharia de Requisitos de LPSDs, possui desafios devido a complexidade dos produtos gerados por esse tipo de Linha de Produto. O foco do presente trabalho está direcionado à tratar das peculiaridades das LPSDs no que diz respeito ao processo de Engenharia de Requisitos.

Silva et al. (2016) verificaram, por meio de uma revisão sistemática da literatura como são executados os processos de Engenharia de Requisitos e Gerência de Variabilidade em LPSDs. Por meio do estudo dos trabalhos selecionados, foram identificadas quais atividades de Engenharia de Requisitos e Gerência de Variabilidade são usadas em LPSDs, como os ativos são construídos, como os requisitos são documentados, como as variabilidades são representadas, quais abordagens são usadas para apoiar o processo de reconfiguração e quais abordagens são utilizadas para eliminar inconsistências no modelo de variabilidades de LPSDs. Silva et al. (2016) proveem a este trabalho um conjunto de atividades e produtos de trabalho relacionados a Engenharia de Requisitos em LPSDs relatados na literatura, facilitando a etapa inicial para a elaboração de um processo de Engenharia de Requisitos para LPSDs, em especial quanto à composição do processo. Silva et al. (2016) também identificaram uma falha relacionada à identificação e o tratamento dos requisitos não funcionais, uma lacuna que o presente trabalho se concentra em preencher juntamente com a construção de um processo de Engenharia de Requisitos.

Capilla et al. (2014) apresentam um *framework* que contém um conjunto de atividades a serem realizadas no desenvolvimento de LPSDs. No estudo realizado, são apresentados métodos para resolver problemas identificados inicialmente pelos autores, gerenciamento da variabilidade

e reconfiguração dinâmica. Para a aplicação das soluções propostas, a atividade de Análise de Domínio tradicional foi dividida em duas novas atividades: Análise de Domínio e Análise de Contexto. Os autores abordam o modo pelo qual as peculiaridades das LPSDs deve ser tratadas já na fase de Engenharia de Requisitos, relacionando-se com o presente trabalho, que se concentra em estabelecer meios para executar as tarefas da engenharia de maneira organizada e direcionada especificamente para LPSDs. No entanto, o estudo apresenta apenas soluções para o cenário dos desafios identificados e não fornecem informações sobre como deve ocorrer a realização das atividades, quais técnicas e ferramentas devem ser utilizadas, os envolvidos nas atividades e a participação dos *stakeholders*.

3 FUNDAMENTAÇÃO TEÓRICA

Nesta Seção são apresentados os principais conceitos para o desenvolvimento deste trabalho. Na primeira Seção 3.1 é apresentada a abordagem de LPSD, suas principais características, benefícios e sua importância. Na Seção 3.3, a Engenharia de Requisitos é abordada em relação a sua execução nas abordagens de LPSs e LPSDs. Na Seção 3.4, a atividade de análise de contexto é descrita juntamente com as atividades de Engenharia de Domínio em LPSDs. A última Seção 3.5 trata dos processos existentes em LPS e suas diferentes metodologias de execução.

3.1 Linhas de Produto de Software Dinâmicas

Linhas de Produtos de Software (LPS) têm sido utilizadas como estratégia pela indústria de software nos últimos anos. Por meio da LPS, empresas conseguiram alcançar uma produção de baixo custo e alta qualidade. As LPSs também permitiram por meio do reuso que as organizações fossem capazes de criar famílias de produtos para domínios específicos. No entanto, o número de sistemas que demandam adaptação em tempo de execução em diferentes contextos tem aumentado e as LPSs, não oferecem o suporte necessário por serem estáticas (CAPILLA et al., 2014).

Os sistemas de software necessitam, cada vez mais de capacidade de execução contínua sobre condições adversas, e alguns ainda precisam executar sem interrupções. Sistemas espacias e sistemas de telecomunicações são fortes referências dessa tendência. É possível ainda citar *smartphones*, que são capazes de adaptar seu comportamento geral em razão dos níveis de energia, qualidade de conexão da rede, entre outros (HINCHEY; PARK; SCHMID, 2012).

Partindo da necessidade de produzir sistemas capazes de adaptação em tempo de execução, surgiram as LPSDs. Esse paradigma emergente possui, como princípio fundamental, a variabilidade dinâmica permitindo que, no momento da execução, o software tenha várias possibilidades de configurações disponíveis para diferentes contextos (CAPILLA et al., 2014).

LPSD é uma abordagem estendida de LPS e por esse motivo, herda as principais características das LPSs. Contudo, LPSDs possuem propriedades dinâmicas, isto é, monitoram uma situação atual e controlam pontos de variação em tempo de execução (HALLSTEINSEN et al., 2008).

Segundo Hallsteinsen et al. (2008), as LPSDs possuem as seguintes propriedades:

- Variabilidade dinâmica;
- Mudanças de conexão diversas vezes durante seu ciclo de vida;
- Pontos de variação mudam durante o tempo de execução;
- Capacidade de lidar com mudanças advindas dos usuários, tal como requisitos funcionais ou de qualidade;
- Sensibilidade ao contexto;
- Prioridade na situação de ambiente ou contexto individual ao invés de "*marketing*".

Como as LPSDs surgiram devido à necessidade de se ter mecanismos voltados para sistemas que exigem mudanças em tempo de execução, é nesse tipo de domínio que se pode obter os melhores resultados a partir da aplicação dessa abordagem. Nesse sentido, Sistemas Orientados a Serviço são alvos de LPSDs, pois fornecem serviços customizáveis que dependem de fatores de qualidade e de propriedades de contexto. O domínio de Dispositivos Móveis é também um cenário adequado para a aplicação de LPSDs, pois é um dispositivo que permite alto grau de personalização, seja de serviços ou das mais variadas aplicações instaladas nele. Sistemas de casas inteligentes também se constituem em um claro exemplo de aplicação dessa abordagem porque são as propriedades de contexto que fazem com que o sistema opere dinamicamente (CAPILLA et al., 2014).

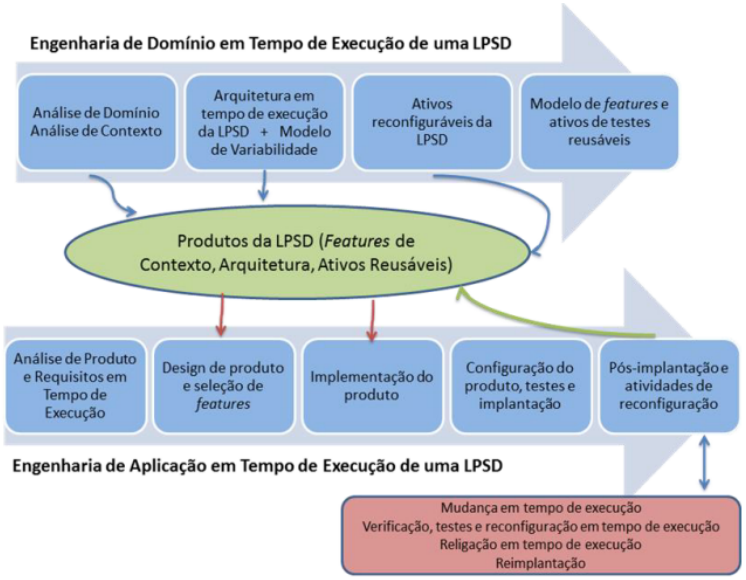
Domínios emergentes de sistemas dinâmicos mostram como a complexidade dos softwares têm crescido. As LPSDs se tornam importantes para esse cenário pois estão focadas em realizar adaptações para situações ou necessidades individuais. Levam em consideração que os ambientes onde esses sistemas estão inseridos exigem um alto grau de adaptabilidade nos dispositivos, em redes e interfaces. (HALLSTEINSEN et al., 2008).

Pelo fato de LPSDs incorporarem propriedades de tempo de execução, elas se tornam mais configuráveis e adaptáveis. Essa característica de adequação a novos ambientes pode promover uma considerável vantagem competitiva. No entanto, esses sistemas devem satisfazer fatores de qualidade, incluindo segurança, confiabilidade e desempenho (BOSCH; CAPILLA, 2012).

A aplicação de LPSDs demanda mudanças e/ou adaptações nas atividades do desenvolvimento de software. Dessa forma, toda a arquitetura do software em tempo de execução deve incluir todos os mecanismos necessários para lidar com as mudanças dinâmicas da LPSD, sejam ativos, produtos, ativação ou desativação de *features* (CAPILLA et al., 2014). O ciclo de vida de uma LPSD estende o modelo tradicional de LPS com algumas modificações,

como mostra a Figura 1. Neste trabalho será explorada a etapa de Engenharia de Domínio do desenvolvimento de LPSDs e se concentrará principalmente na fase de Análise de Contexto que será apresentada na Seção 3.4.

Figura 1 – Ciclo de vida de atividades de uma LPSD.



Fonte: Capilla et al. (2014).(com adaptações)

A Figura 1 mostra um ciclo de vida que abrange Engenharia de Domínio e Engenharia de Aplicação para o desenvolvimento de uma LPSD. Em LPSDs a etapa de engenharia de domínio em tempo de execução inicia seu ciclo com a tarefa de análise de domínio, que vem acompanhada da análise de contexto, esta identifica não apenas os requisitos e os membros da família de produtos mas também as propriedades de contexto necessárias à LPSD. A arquitetura desenvolvida para LPSDs deve se concentrar nas variações em tempo de execução, portanto deve se preocupar com o desenvolvimento de ativos reconfiguráveis e não reconfiguráveis, além de implementar módulos específicos para suportar reconfigurações em momento de execução e vinculações dinâmicas para automatizar o processo de tomada de decisões que também ocorre durante a execução do sistema. Os artefatos do repositório de uma LPSD devem ser testados e o modelo de *features* deve ser validado. Durante a fase de engenharia de aplicação em tempo de execução, requisitos dinâmicos e não dinâmicos levam à construção dos membros da família de produtos da LPSD. O modelo de *features* determina os produtos da LPSD que irão exibir as características reconfiguráveis requeridas. Por fim os produtos são testados, configurados e implantados, se uma nova reconfiguração é requerida em tempo de execução, o ciclo de vida suporta as atividades de reconfiguração e *post-deployment* para lidar com essas mudanças

(CAPILLA et al., 2014).

A adoção de uma abordagem de uma família ou linha de produto oferece vários benefícios para o desenvolvimento industrial de software. Essa abordagem busca uma produção de baixo custo por meio do compartilhamento de um referencial comum dos conceitos de arquitetura e dos produtos, e permite que algumas características dos produtos se diferenciem para atender aos diferentes requisitos do mercado. Esse processo maximiza o reuso de similaridades entre os produtos e minimiza o custo nas variações dos produtos (FANTECHI et al., 2004). Dessa forma, o tempo para o lançamento do produto no mercado é reduzido, trazendo vantagens competitivas para a organização. Os clientes também recebem benefícios, uma vez que a produção tem um baixo custo, o cliente tem a possibilidade de adquirir o produto por um menor preço.

3.2 Engenharia de Requisitos

Um requisito de sistema é uma descrição de algo que o sistema deve fazer, serviços que ele pode oferecer e restrições relacionadas ao seu funcionamento. Essas descrições são estabelecidas, de acordo com as necessidades do cliente para um sistema com um objetivo específico. A Engenharia de Requisitos é o processo responsável por identificar, analisar, documentar e verificar esses requisitos (SOMMERVILLE, 2011).

O processo de Engenharia de Requisitos permite examinar o contexto do desenvolvimento de software; as necessidades do projeto; as prioridades em relação a sequência das atividades realizadas e o fluxo de informações; funções e comportamentos que impactarão o projeto de software (PRESSMAN, 2011). Este processo pode incluir quatro atividades de alto nível: estudo de viabilidade, elicitação e análise de requisitos, especificação e validação. A Engenharia de Requisitos é um processo que pode ser apresentado com suas atividades em sequência, no entanto na prática, acaba sendo um processo iterativo em que as atividades são intercaladas (SOMMERVILLE, 2011).

O presente trabalho baseia-se neste conceito de Engenharia de Requisitos, no entanto não apresenta um processo completo, e sim um processo que se concentra na elicitação e modelagem dos requisitos e *features* para LPSDs.

3.3 Engenharia de Requisitos em LPSs e LPSDs

A Engenharia de Requisitos é responsável pela identificação das funcionalidades e restrições que um sistema deve possuir para atender as necessidades dos usuários. Em se tratando de LPS identificar e entender os requisitos se torna uma tarefa complexa pois os requisitos são afetados pelos variados contextos de utilização dos sistemas. O processo se torna mais difícil devido à fabricação de uma maior quantidade de produtos e a atenção que deve ser dirigida às variabilidades e similaridades (BATISTA; SILVA, 2015).

Em LPSs, a definição e gerenciamento das variabilidades ocorre antes da execução do software. Em LPSDs, o processo de Engenharia de Requisitos requer ainda mais esforço, pois é preciso estruturar as variabilidades no momento da execução. É possível estabelecer um intervalo de potenciais variabilidades em um modelo, um produto de trabalho que será útil para guiar as tarefas que correspondem a construção do sistema (HINCHEY; PARK; SCHMID, 2012). No processo de Engenharia de Requisitos de LPSDs são realizadas atividades e técnicas que são utilizadas também em processos tradicionais de desenvolvimento de software mas, levando em consideração as peculiaridades dessa abordagem, existem tarefas, métodos e produtos de trabalho mais específicos relacionados a este processo (SILVA et al., 2016).

No domínio de sistemas em que as LPSDs atuam, existem os requisitos sensíveis aos contextos em que o sistema deve operar. A volatilidade dos contextos faz com eles não sejam perfeitamente entendidos, tornando a construção dos sistemas de software um desafio. Esses sistemas devem ser hábeis para adaptar-se a novos contextos de ambientes dinamicamente, mas a incerteza sobre a demanda desses contextos faz com que seja difícil formular, validar e gerenciar esses requisitos (SAWYER et al., 2010).

Para superar os desafios impostos pelas abordagens de linhas de produto, devem existir práticas maduras de desenvolvimento e gerenciamento de software que sejam capazes de cobrir toda a complexidade arquitetural. Engenharia e Gerenciamento dos Requisitos são as tarefas centrais dessas abordagens e devem ser capazes de lidar com a variedade na comunidade dos *stakeholders*, com o modelo de domínio e com o constante fluxo dos requisitos (BIRK; HELLER, 2007).

Birk e Heller (2007) identificaram uma lista de desafios relacionados à documentação dos requisitos em LPSs:

- Definir um meta modelo para a documentação dos requisitos;

- Definir pontos de vista específicos de partes interessadas sobre base de requisitos;
- Gerar documentos de visões específicas da base de requisitos;
- Garantir a qualidade da documentação;
- Mapear requisitos para produtos, lançamentos e variantes;
- Modelar diferentes tipos de status de requisitos.

O trabalho de Silva (2016) apresenta, em seus resultados, um conjunto de atividades, produtos de trabalho, abordagens, ferramentas e papéis que estão relacionados com a Engenharia de Requisitos quando se trata de LPDSs, conforme ilustrado na Figura 2.

Figura 2 – Síntese de resultados.

<table>
<tr><th colspan="4">Engenharia de Domínio</th></tr>
<tr><th></th><th colspan="2">Análise de Domínio</th><th rowspan="2">Projeto de Domínio</th></tr>
<tr><th>Resultados</th><th>Análise de Domínio</th><th>Análise de Contexto</th></tr>
<tr><td rowspan="4">Atividades</td><td>Concepção</td><td>Identificar Propriedades físicas do sistema</td><td rowspan="2">Definir regras operacionais</td></tr>
<tr><td>Elicitação</td><td>Identificar features de contexto</td></tr>
<tr><td>Especificação</td><td>Modelar features de contexto</td><td rowspan="2">Definir conexões múltiplas</td></tr>
<tr><td>Validação</td><td>Definir regras operacionais</td></tr>
<tr><td>Abordagens utilizadas</td><td>Modelo de Features, Diagrama de Classes, Casos de Uso, Diagramas de Sequência, Schemas, Goals</td><td colspan="2">Modelo de Aspecto, Actor Model, MVRP, OWL, OCL, Modelo de features.</td></tr>
<tr><td>Produtos de Trabalho</td><td>Diagrama de Classes, Casos de Uso, Diagrama de Sequência</td><td colspan="2">Regras de Adaptação, ECA, MAPE-K, MAPE, Regras de Transformação, Modelos de Aspectos, Mapeamento de Contexto, PrtNets, Satisfação de Restrição</td></tr>
<tr><td>Ferramentas</td><td colspan="3">BPMN, Fama, MOSkit4SPL, Atlas Model, Familiar, FeatureIDE, eMoflon, Odyssey, DOPPLER, VariaMos, GNU Prolog, Clafer.</td></tr>
<tr><td>Responsáveis</td><td colspan="3">Analista, Projetista, Arquiteto, Desenvolvedor</td></tr>
</table>

Fonte: Silva (2016)

A partir da observação da Figura 2 percebe-se, que existe um conjunto de atividades utilizadas para apoiar o processo inicial de modelagem de LPSD. Essas atividades se concentram, também no tratamento da variabilidade, considerando as propriedades que uma LPSD possui.

Em relação à execução das atividades, o trabalho de Silva (2016) constatou que, as tarefas ligadas a fase de elicitação possuem menos relevância pelos trabalhos analisados. A maioria dos trabalhos, não realiza especificação dos requisitos de domínio em um documento preciso e também não apresentam abordagens utilizadas para converter os requisitos em *features* relacionadas ao modelo de *features*.

Os trabalhos que especificam os requisitos precisamente utilizam, abordagens geralmente adotadas pelo desenvolvimento de software tradicional. Alguns trabalhos representam os requisitos de domínio por meio de modelos de *features* com extensões. O principal uso das atividades identificadas por Silva (2016), está relacionado ao suporte nas fases de especificação e modelagem da variabilidade do domínio. Entretanto, os trabalhos não

apresentam ferramentas para apoiar a concepção, elicitação e documentação dos requisitos.

Nas Subseções seguintes, alguns resultados encontrados por esta revisão sistemática da literatura serão discutidos, pois constituem a principal fonte de dados para o desenvolvimento do trabalho proposto.

3.3.1 Elicitação

A elicitação tradicionalmente é um processo em que são identificados os requisitos iniciais do sistema através de observação de sistemas existentes, discussão com *stakeholders*, entre outras etapas. É um processo que pode envolver sistemas de protótipos que ajudam a entender o sistema que deverá ser especificado (SOMMERVILLE, 2011).

Em LPSs, a elicitação é um subprocesso da Engenharia de Domínio, que tem a função de identificar os requisitos de domínio, as comunalidades e variações de uma família de produtos. Isso pode ocorrer através de sessões de *brainstorming*, entrevistas, questionários e outras técnicas também usadas no processo de elicitação conhecido no desenvolvimento tradicional de software. Ao realizar a atividade de elicitação também são identificados conflitos que devem ser resolvidos através de acordos que satisfaçam os *stakeholders* (NEIVA et al., 2010).

Na metodologia de linhas de produto, especialistas de domínio e engenheiros de linhas de produto, devem trabalhar juntos para elicitar e modelar as comunalidades e as variabilidades, em um processo que é interativo e consome bastante tempo. Os especialistas e engenheiros contribuem com conhecimento sobre o domínio de aplicação, e sobre os processos e produtos da engenharia de linhas de produto respectivamente (FANTECHI et al., 2004). Nas LPSs a atividade de análise domínio é desenvolvida para entender o cenário real e capturar os requisitos da linha de produto. Nas LPSDs, essa atividade se estende para a análise de contextos que identifica os requisitos dinâmicos e como eles reagem em determinadas situações (CAPILLA et al., 2014).

Características como sensibilidade ao contexto, estão presentes quando se trata de LPSDs. Consequentemente, elicitações de propriedades de contexto devem ser realizadas para que os sistemas possam explorar aspectos de um determinado ambiente e reagir a ele (CAPILLA et al., 2014). Para dar suporte às adaptações de contexto, nessa abordagem devem ser estudadas as interações entre usuários em diferentes ambientes e contextos, os possíveis contextos, como esses contextos podem afetar as interações do usuário, e quais são os requisitos específicos para cada contexto (HONG; CHIU; SHEN, 2005). Em LPSDs, a elicitação deve identificar *features*

de contexto que estarão presentes também no modelo de variabilidades e comunalidades, além dos requisitos e das *features* comuns.

Ao introduzir o conceito de variabilidade no desenvolvimento de software, a primeira fase que explorará esse conceito é a fase de desenvolvimento dos requisitos, que abrange a elicitação, análise, verificação e validação dos requisitos. A elicitação é a tarefa mais complexa devido à interação humana, e é a uma das atividades que mais pode colocar todo o projeto em risco se for negligenciada. Seu produto de trabalho de saída é uma versão inicial dos requisitos.

A principal diferença entre a elicitação de uma abordagem tradicional e a elicitação de uma abordagem de família de produtos, é que a elicitação utilizada em linhas de produto recebe, um conjunto de ativos reusáveis como entrada para a execução das suas subtarefas. Além dessa diferença, a elicitação em LPSDs tem a função de identificar as novas *features* do produto, levando em consideração requisitos de referência (DURÁN; BENAVIDES; BERMEJO, 2004).

A atividade de elicitação em uma linha de produto apresenta algumas dificuldades (FANTECHI et al., 2004):

- Captura de todos os requisitos comuns para todos os membros da linha de produto;
- Identificação de todos os requisitos válidos apenas para parte dos membros;
- Especialização e instanciação de requisitos genéricos em requisitos de aplicação para um único produto.

No levantamento de trabalhos realizado por Silva (2016), não foram identificadas ferramentas que apoiassem o processo de elicitação e, nenhum dos papéis identificados foi associado a elas, apesar da execução de algumas atividades de elicitação citadas em alguns trabalhos.

3.3.1.1 Casos de Uso de Linhas de Produto de Software Sensíveis ao Contexto

O *template* de Casos de Uso de Linhas de Produto de Software Sensíveis ao Contexto (do inglês *Context Aware Software Product Line Use Case - CAPLUC*) é um modelo de descrição textual que busca lidar com a informação de contextos como uma nova entrada que impacta no comportamento da aplicação em qualquer momento durante a execução (SANTOS; ANDRADE; NETO, 2013).

O *template* CAPLUC possibilita a definição de funcionalidades, de variabilidade e da influência dos contextos na linha de produto. Alguns elementos importantes são detalhados a seguir:

- **Categoria de Reúso:** Permite identificar se o caso de uso é mandatório, opcional ou alternativo.
- **Restrição de Contexto:** Especifica o contexto em que o caso de uso é aplicável.
- **Passos:** Representam os passos executados pelos atores do cenário em estilo tabular, incluindo uma representação específica para passos que possuem restrições de contexto por meio do símbolo "*".
- **Sumário de Variações:** Exibe variações que afetam o caso de uso.

3.3.2 Especificação

A especificação é o detalhamento de todos os aspectos que serão usados para construir um determinado sistema. A especificação pode ser apresentada através de documentos, modelos gráficos, cenários de uso, protótipos ou a combinação desses. Esta etapa deve ser executada de maneira apropriada para que os requisitos sejam apresentados de maneira consistente (PRESSMAN, 2011).

A fase de especificação é o momento em que as informações obtidas na atividade de elicitação são extraídas e descritas em um documento que contém um conjunto de requisitos. Neste documento estão presentes os requisitos de usuário, que são informações mais abstratas direcionadas ao cliente e usuário final, e os requisitos de sistema, que fornecem um conteúdo mais detalhado sobre as funcionalidades que o sistema deve prover (SOMMERVILLE, 2011).

Para LPSs, a atividade de especificação abrange algumas tarefas únicas, já que leva em consideração a variabilidade, que deve ser analisada e definida. A definição da variabilidade envolve a modelagem e a documentação em uma notação adequada da mesma (POHL; BÖCKLE; LINDEN, 2005). Nesse cenário, os documentos podem ser textuais, heterogêneos, com diferentes níveis de abstração e com diferentes formatos de estruturas (ALVES et al., 2008).

Gurp, Bosch e Svahnberg (2001) acreditam que o documento de especificação deve conter apenas informações sobre o ambiente, para que o artefato de especificação não seja influenciado pela implementação. Cada *feature* presente neste documento é a tradução de um requisito, nesse sentido, requisitos independentes de implementação devem ser mapeados para a implementação das *features* de contexto.

A especificação dos requisitos é um pré-requisito para a construção do diagrama de *features*, e ele pode ser construído no início do processo de Engenharia de Requisitos. A construção do diagrama de *features* pode ajudar os desenvolvedores na identificação de quais

partes do sistema devem suportar a variabilidade, além de auxiliar na gestão da variabilidade (GURP; BOSCH; SVAHNBERG, 2001).

De acordo com os resultados de Silva et al. (2016) e Silva (2016), a atividade de especificação em LPSDs apesar de fazer uso de abordagens direcionadas para identificar a variabilidade dinâmica do domínio, também faz uso de abordagens utilizadas em desenvolvimento tradicional de software, por exemplo, diagramas UML. Ainda nos trabalhos de Silva et al. (2016) e Silva (2016), também foram identificados papéis para a execução das atividades de Engenharia de Requisitos. Estes papéis por sua vez, encontram-se mais relacionados com a fase de especificação e modelagem de domínio de LPSDs. Por exemplo, projetistas e arquitetos normalmente são responsáveis por especificar a arquitetura da linha de produto.

Em relação às abordagens utilizadas para documentar os requisitos, o trabalho de Silva (2016) detectou o uso tanto de abordagens convencionais, quanto de abordagens voltadas para documentar a variabilidade. As abordagens identificadas para a documentação da variabilidade de requisitos foram: Modelo de *features*, Diagrama de Classes Casos de uso, Diagrama de Sequência, *Schemas*, e *Goals*.

3.3.2.1 Metamodelo de Características da Notação Odyssey-FEX

A notação Odyssey-FEX é utilizada para a representação da variabilidade em modelos de *features* e sua semântica é formalizada por um metamodelo. O trabalho de Oliveira et al. (2005) oferece uma descrição detalhada das classes que compõem o metamodelo.

Para um melhor entendimento do metamodelo, os autores dividiram as classes em três (3) pacotes: 1) ***Principal,*** que representa a parte de definição da taxonomia das *features*, 2) ***Relacionamento,*** que especifica propriedades dos relacionamentos existentes no modelo de *features*, e 3) ***Regras de Composição,*** que especifica as regras de dependência e exclusividade entre as *features* do modelo.

Neste trabalho, o metamodelo é utilizado na fase de especificação para classificar *features*, definir seus relacionamentos e restrições.

3.3.2.2 Classificação de Contextos

As aplicações sensíveis a contextos, assim como todas as aplicações devem satisfazer os objetivos dos usuários. Dessa forma, se faz importante entender quais são as expectativas

dos usuário a cerca da aplicação. Em aplicações que requerem uma adaptação em tempo de execução, ocorrem mudanças de comportamento advindas de determinados contextos que podem impactar a expectativa e a experiência do usuário.

Sabendo do impacto das adaptações de sistemas sensíveis ao contexto, nota-se a importância de estudar situações ou contextos em que o produto de software pode ser inserido, e quais mudanças de comportamento os contextos trazem para a aplicação. Nesse sentido, para um melhor entendimento dos contextos de uma aplicação, Hong, Chiu e Shen (2005) agruparam os contextos em três (3) categorias, que são: i) Contextos Computacionais, ii) Contextos de Usuário e, iii) Contextos Físicos de Ambiente, as categorias de contexto são detalhadas a seguir.

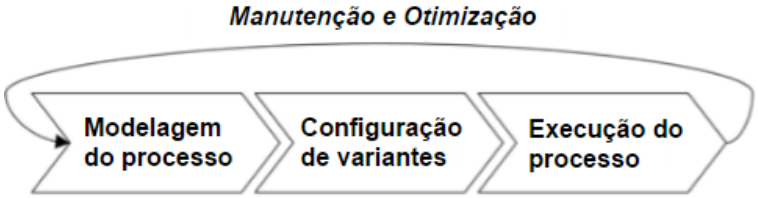
- **Contextos Computacionais** se referem a informações de ambiente computacional, como, por exemplo, energia, memória etc.
- **Contextos de Usuário** se referem a preferências do usuário, sejam elas predefinidas ou aprendidas.
- **Contextos Físicos de Ambiente** se referem a informações que são fornecidas por um ambiente do mundo real.

3.3.2.3 Provop

Provop é uma abordagem capaz de lidar com conceitos avançados para configuração e gerenciamento de variantes de processos de negócio. Essa abordagem provê soluções flexíveis para o gerenciamento das variantes dos processos de negócio ao longo do seu ciclo de vida (REICHERT; HALLERBACH; BAUER, 2015).

Os autores conduziram estudos em diferentes domínios para identificar requisitos chave para o gerenciamento das variantes dos processos. Foram identificados requisitos relacionados a aspectos como, contextos de processos e configurações dirigidas a contexto. O ciclo de vida padrão dessa abordagem lida com os requisitos em todas as fases, que são três (3), o projeto e modelagem do processo, a criação de uma variante particular do processo e a implantação dessa variante em um ambiente em tempo de execução.

Figura 3 – Ciclo de vida do processo



Fonte: Reichert, Hallerbach e Bauer (2015).(com adaptações)

A abordagem Provop é capaz de derivar, processos por meio de operações de adição, remoção e modificação de elementos do processo a partir de um ponto de ajuste. Essa abordagem foi adotada para modelar processos de negócios relacionados a casos de uso de sistemas. Uma vez que ela permite operações dinâmicas entre os elementos que constituem o processo, é possível utilizá-la para definir quais processos serão executadas para executar uma tarefa em um determinado contexto.

3.3.3 Modelo de Features

Os modelos de *features* foram apresentados pela comunidade de projetos de software, como uma forma de abstrair de um dado projeto arquitetural de alto nível os requisitos e representá-los. Nesse sentido, podem ser usados para expressar os propósitos do sistema, assim como também podem ser usado para documentar a variabilidade (POHL; BÖCKLE; LINDEN, 2005).

O modelo de *features* é uma abordagem usada pela maioria dos trabalhos relacionados ao gerenciamento da variabilidade em LPSDs, para representar a variabilidade devido a sua flexibilidade. Este modelo permite mudanças em suas próprias propriedades com o objetivo de atender as demandas dos usuários. Ao utilizar esse modelo, não existe um padrão para a representação da variabilidade, isto é, a forma de organização de seus elementos pode variar (SILVA et al., 2016).

3.3.3.1 DyMMer

Tendo em vista a importância do modelo de *features* tanto para LPSs quanto pra LPSDs, para a documentação da variabilidade e da similaridade de um domínio em particular (KANG et al., 1990), esta Seção apresenta uma ferramenta capaz de auxiliar na visualização, edição e criação de modelos de *features* e na análise de métricas do modelo.

A ferramenta DyMMer - Ferramenta de Modelos de *features* de LPSDs baseada em Métricas (do inglês - *DSPL feature Models tool based on Measures*) foi desenvolvida com o objetivo de extrair medidas de diferentes modelos, que são descritos no formato proposto pelo repositório de modelos de *features* S.P.L.O.T (MENDONCA; BRANCO; COWAN, 2009), (BEZERRA et al., 2016).

A DyMMer permite visualizar e analisar um modelo específico importado. É possível visualizar para cada modelo de *features*, as *features* ativadas e desativadas de acordo com suas restrições. Ainda relacionado a visualização é possível selecionar métricas individuais, coletar os resultados dessas métricas e visualizar na tela. Além de visualizar as métricas, é possível também exportar para um arquivo no formato Microsoft Office Excel.

O recurso de edição de modelos fornecido pela ferramenta permite que o usuário possa adicionar contextos em modelos sem contexto, adicionar ou remover *features*, adicionar ou remover restrições, etc. Quando um contexto é adicionado, o Engenheiro de Domínio deve ativar ou desativar *features* e incluir as devidas restrições. Modelos de *features* de LPSD devem possuir um contexto ou mais.

A utilização da DyMMer neste trabalho se dá principalmente pelo fato de a versão atual permitir a construção de modelos de *feature* do início e a aplicação de contextos que impacta na mudança de comportamento desses modelos. A versão atual pode ser encontrada em um repositório online [1].

3.3.4 Verificação de Modelos de feature

A atividade de verificação de requisitos é responsável por garantir que os requisitos documentados estejam definidos da forma correta (POHL; BÖCKLE; LINDEN, 2005). Neste trabalho a atividade de verificação se concentra em assegurar que as *features* sejam documentadas de maneira correta. Para garantir a corretude e a consistência do modelo de *features* foi adotada uma técnica de inspeção de modelos de *features* em forma de *checklist*.

3.3.4.1 FM Check

O *FM Check* - Verificação de Modelo de *Features*(do inglês - *Features Model Check*) é uma técnica de inspeção baseada em *checklist* que busca a detecção de defeitos em modelos de *features* por meio de inspeções individuais de itens e de casos discrepantes relacionados

[1] https://github.com/DyMMerProject/DyMMerV2

à consistência, clareza, consistência, corretude e completude de um modelo de *features* em comparação a descrição textual correspondente do domínio (MELLO et al., 2014) . Essa técnica também aborda a identificação de casos discreprantes, que podem ser definidos como uma situação genérica encontrada em artefatos que pode configurar uma discordância, ou seja, onde um defeito pode ser detectado após passar por uma inspeção (MELLO; PEREIRA; TRAVASSOS, 2010). Os casos discrepantes devem ser identificados e categorizados quanto a omissão, fato incorreto, inconsistência, ambiguidade e informação estranha.

O *FM Check* tem o objetivo de apoiar inspeções individuais para detectar defeitos em modelos de *features* quando estes são baseados na notação FODA, incluindo também itens particulares relacionados a notação Odyssey-FEX. Para a aplicação dessa técnica, o inspetor não precisa possuir conhecimento sobre o domínio, desde que seja estabelecido um documento como, por exemplo, de especificação de requisitos, para ser usado como base para comparação durante a aplicação da técnica (MELLO et al., 2014).

A organização da técnica de inspeção considera 48 casos discrepantes, que foram divididos em sete grupos, e estes por sua vez foram analisados e reagrupados para permitir a construção do *checklist*. Baseando-se nisso, os itens do *FM Check* foram separados em três grupos de verificação: i) verificação individual por *feature*; ii) verificação de relacionamentos entre *features*; e iii) verificação de regras de composição. Cada item de verificação pode obter três possíveis respostas: "Sim", "Não"e "N.A"(não se aplica).

3.3.5 Gerenciamento da Variabilidade

Linhas de Produto têm entre suas principais propriedades a flexibilidade, que é o aspecto que permite que haja a derivação de produtos diferentes de uma mesma família por meio de customizações em massa. Essa flexibilidade em LPSs é também chamada de variabilidade, um aspecto essencial dos artefatos do domínio, que é a base para a customização dos softwares (POHL; BÖCKLE; LINDEN, 2005).

A variabilidade pode ser definida como a habilidade que um software possui de alterar-se ou customizar-se a partir da descoberta de um ponto de variação (GURP; BOSCH; SVAHNBERG, 2001). Ela está relacionada a todas as atividades que se preocupam com a identificação e documentação de variações, sendo definida na etapa de Engenharia de Domínio e explorada na Engenharia de Aplicação através da vinculação de variantes apropriadas (POHL; BÖCKLE; LINDEN, 2005).

A variabilidade pode e deve ser gerenciada em estágios diferentes do ciclo de vida de uma LPS. Segundo Pohl, Böckle e Linden (2005) esse gerenciamento incorpora os seguintes conceitos:

- Apoio a atividades que se concentram em definir variabilidades;
- Gestão de artefatos variáveis;
- Apoio as atividades que se concentram em resolver as variabilidades;
- Coletar, armazenar e gerenciar o histórico de informações necessárias para realizar essas tarefas.

Gerir a variabilidade envolve manutenção adaptativa, corretiva e de melhorias, além da contínua adição de novas variantes para substituir as que já não são mais usadas. Além disso, *features* variantes podem ser completamente removidas, assim como, novos produtos são inseridos e outros são excluídos na linha de produto em decorrência a mudanças nos requisitos. O gerenciamento de variabilidades também envolve a distribuição de novas variantes que já foram implantadas e modelos de cobrança que conseguem lucrar com novas variantes (SVAHNBERG; GURP; BOSCH, 2005).

As LPSDs possuem a variabilidade em tempo de execução como uma de suas características básicas. O papel que ela tem no lançamento e na evolução dos produtos se mais torna relevante quando os sistemas mudam dinamicamente ao longo do tempo. Entretanto, implementar esse mecanismo ainda é uma dificuldade devido à necessidade de se gerenciar uma grande variedade de cenários em tempo real, decorrentes das propriedades de contexto (CAPILLA et al., 2014).

Em relação à variabilidade desse domínio, os sistemas podem evoluir de três maneiras (BOSCH; CAPILLA, 2012):

- pontos de variação podem se tornar cada vez mais dinâmicos, permitindo alterações durante o funcionamento do sistema;
- Um conjunto de variantes de um ponto de variação pode ser estendido depois da implantação do sistema, ou quando o sistema está em funcionamento;
- Sistemas podem selecionar cada vez mais variantes para alcançar métricas como parte da operação, muitas vezes definas como auto-comportamento.

Nos produtos providos por LPSDs, é certo que haverá mudanças em tempo de execução que podem impactar na estrutura no modelo de *features* já implantado e na estrutura do sistema de *features* que suporta uma configuração atual. Nesse sentido, se faz necessário o

gerenciamento das operações de variabilidades antes que o sistema troque novamente para o modo de operação normal (CAPILLA et al., 2014).

Como parte do Gerenciamento da Variabilidade, a revisão sistemática de Silva (2016) identificou um conjunto de sete (7) abordagens, sendo o modelo de *features* a mais utilizada, também empregada em abordagens de LPSs tradicionais. Além disso, foi possível detectar variações do modelo de *features*, como o modelo de *features* estendido e o modelo de *features* de contexto.

3.3.5.1 Matriz de Rastreabilidade de Requisitos

A matriz de rastreabilidade de requisitos de uma aplicação é uma forma de representar uma aproximação a cerca da comunalidade e também da variabilidade, para um dado conjunto de requisitos de uma aplicação. Essa matriz detalha a definição de *features* comuns e variáveis em um alto nível de abstração (POHL; BÖCKLE; LINDEN, 2005).

Quadro 1 – Estrutura de uma matriz de requisitos de aplicação para quatro aplicações

Requisitos da Aplicação	**App. 1**	**App. 2**	**App. 3**	**App. 4**
R1	obrigatório	obrigatório	obrigatório	obrigatório
R2	-	-	obrigatório	obrigatório
R3	-	obrigatório	-	-
...	...	...	...	...

Fonte: (POHL; BÖCKLE; LINDEN, 2005)(com adaptaçãoes).

Na matriz acima, apresentada por Pohl, Böckle e Linden (2005), o requisito 'R1' é obrigatório para todas as aplicações e, portanto é propenso a ser definido como um requisito comum na linha de produto. O requisito 'R2' não está presente em 'App. 1' nem em 'App. 2'. Portanto, não pode ser definido como um requisito comum para a linha de produto. O mesmo vale para o 'R3'.

A matriz de Pohl, Böckle e Linden (2005) é útil para a análise da comunalidade e da variabilidade da linha de produto, como também permite a gerência dos requisitos por meio da rastreabilidade entre as aplicações.

3.4 Análise de Contexto

Engenharia de Domínio é um processo de LPS que estabelece uma plataforma reusável onde as similaridades e variabilidades de uma família de produtos são percebidas e

definidas para apoiar a customização em massa. (POHL; BÖCKLE; LINDEN, 2005). A mesma engloba a análise domínio e a análise contexto como atividades do seu ciclo de vida. A análise de domínio especifica as características e necessidades do cenário de domínio em que a LPSD irá atuar, enquanto a análise de contexto identifica as mudanças de contextos que a LPSD deve ser capaz de suportar (CAPILLA et al., 2014).

Um contexto é caracterizado por variações imprevisíveis e previsíveis. As variações previsíveis são aquelas que os projetistas são capazes de identificar, isto é, podem ser detectadas em tempo de projeto. Já as variações imprevisíveis são aquelas que aparecem apenas em tempo de execução (INVERARDI; MORI, 2011).

Em se tratando de LPSDs, a Engenharia de Requisitos incorpora a tarefa de análise de contexto, que deve distinguir as propriedades de contexto que são usadas para explorar características de ambientes em tempo de execução. O escopo do domínio identifica o escopo de um sistema da família de produto, enquanto a análise de contexto estende essa atividade para capturar informações sobre reconfiguração, mudanças de contexto do produto e sobre *features* de contexto que são relevantes para a LPSD (CAPILLA et al., 2014).

Por meio dos trabalhos de Silva et al. (2016) e Silva (2016), foi possível identificar que a fase de análise de contexto tem foco direcionado para a identificação e modelagem de contexto. Ela engloba, em suas atividades, a identificação de *features* de contexto, identificação de propriedades físicas do sistema, modelagem de *features* de contexto e definição de regras operacionais.

3.5 Processos de Engenharia de Requisitos em LPSs

Como foi citado na Seção 4.3, a Engenharia de Requisitos é uma atividade importante da Engenharia de Domínio e suas tarefas estão distribuídas entre as atividades de análise de domínio e análise de contexto. Nesta Seção são apresentados alguns processos e abordagens da Engenharia de Requisitos desenvolvidos para LPSs. Estes processos apresentam informações sobre o modo de realização de suas atividades, motivados por diferentes métodos.

3.5.1 G2SPL: Um Processo de Engenharia de Requisitos Orientada a Objetivos para Linhas de Produtos de Software

Silva, Borba e Castro (2010) acreditam que abordagens orientadas a objetivos podem ser usadas para descobrir os requisitos comuns e variáveis de uma LPS ao mesmo tempo que é essa metodologia é capaz de reduzir os custos associados a configuração de um determinado produto da linha de produto. Esse processo de Engenharia de Requisitos é movido pelos objetivos dos *stakeholder*s e ataca a dificuldade que existe em estabelecer um relacionamento entre as *features* de um produto de software e os objetivos e necessidades dos *stakeholders*. Para a construção do processo são definidas atividades e heurísticas que são modeladas com o auxilio da ferramenta BPMN.

No trabalho é apresentada, uma visão geral do *framework* i*, que é uma abordagem orientada a objetivos, desenvolvida para analisar e modelar processos que envolvem vários indivíduos sob uma óptica estratégia e intencional. Em seguida é apresentada uma extensão do modelo i* para inclusão da cardinalidade.

Os autores definiram atividades e heurísticas para alcançar o objetivo do trabalho, que é permitir a identificação de *features* nos modelos i* e a configuração de produtos específicos dentro de uma LPS. As atividades e heurísticas formam um processo que é ilustrado por meio de sua aplicação a um caso de uso de uma LPS de mídias móveis. O processo é composto por sete (7) atividades, possuindo duas atividades opcionais. Elas estão relacionadas a construção de modelos de Razão Estratégica (do inglês - *Strategic Rationale model* - SR), construção de modelos de *features* e configuração de produtos. Essas atividades estão associadas aos papéis de engenheiro de domínio e engenheiro de configuração.

3.5.2 Um processo de Engenharia de Requisitos para Linhas de Produtos de Software

Neiva et al. (2010) elaboraram um processo de Engenharia de Requisitos iterativo incremental, permitindo melhorias e refinamentos nos requisitos durante os diferentes estágios do ciclo de vida do processo. O processo foi dividido em estágios maiores: (i) Definição do escopo; (ii) Definição dos requisitos; e (iii) Definição dos cados de uso, cada um contento um conjunto de tarefas executadas de maneira iterativa. Para cada tarefa são apresentados artefatos de saída, artefatos de entrada e papeis envolvidos. O processo utiliza o modelo de *features* para representar uma abstração dos requisitos além das variabilidades e similaridades. As variabilidades também

são representadas através da visão de casos de uso.

Antes da elaboração do processo de Engenharia de Requisitos em si, foi realizada uma revisão sistemática com o objetivo de identificar atividades e técnicas utilizadas em abordagens de LPSs e como as abordagens lidam com a variabilidade.

Em (i) são abordadas as atividades de elicitação, modelagem de *features* e verificação. Sobre a atividade de elicitação, que se repete em (ii) e (iii), é apresentado o uso das técnicas de *brainstorming* e entrevistas para a captura dos requisitos de domínio e variações da LPS. Também são apresentadas fontes de informação para identificação dos requisitos, caso os *stakeholders* não estejam disponíveis para participar no processo de identificação dos requisitos.

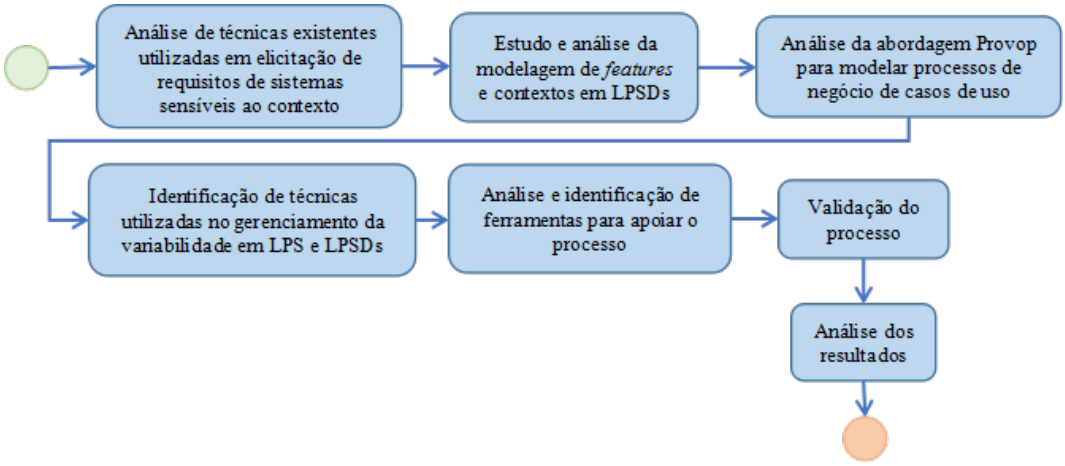
Modelos de *features* são utilizados com o objetivo de modelar a definir o escopo da LPS, constituindo a primeira tarefa de interpretação e organização dos requisitos. Ao executar essa tarefa, são capturadas e representadas as comunalidades e variabilidades no modelo de *features*, que ajudará a identificar que partes do sistema deverá suportar a variabilidade.

Em (ii) é abordada a atividade de descrição dos requisitos, que tem o objetivo de especificar os requisitos em dimensões funcionais e não funcionais, descrevendo também as propriedades da LPS. Por fim, em (iii) é apresentada a atividade de definição de casos de uso, que se concentra em representar os requisitos e suas variabilidades na visão de casos de uso.

4 PROCEDIMENTOS METODOLÓGICOS

Esta Seção tem o objetivo de apresentar a sequência de passos realizados para a execução deste trabalho. A Figura 4 representa graficamente o curso das atividades desenvolvidas para a realização deste trabalho, desde a análise de técnicas de elicitação existentes para sistemas sensíveis ao contexto, até a análise dos resultados obtidos com a aplicação do processo.

Figura 4 – Procedimentos realizados para execução deste trabalho.



Fonte: Elaborada pelo Autora.

Inicialmente foi realizado um estudo e análise sobre técnicas existentes de elicitação no domínio de sistemas sensíveis ao contexto. Em seguida foi realizado um estudo sobre a modelagem de contextos e *features* de LPSDs, afim de identificar os elementos que devem ser documentados e modelados. Em decorrência do resultado da atividade anterior a abordagem Provop foi analisada em relação a sua utilidade neste trabalho. Depois, foi realizada uma pesquisa para a identificação de técnicas e abordagens mais utilizadas no gerenciamento da variabilidade em LPSs e LPSDs. A última atividade de estudo sobre a composição do processo foi a análise de ferramentas para realizar as atividades do processo e representar o fluxo das atividades. Com o processo já estabelecido, foi realizada a validação e, por fim seus resultados foram analisados. As atividades de elicitação, modelagem e gerenciamento da variabilidade foram escolhidas com base principalmente nos resultados obtidos em Silva et al. (2016).

4.1 Análise de técnicas existentes utilizadas em elicitação de requisitos de sistemas sensíveis ao contexto

A primeira etapa para a criação do processo se concentrou em definir que técnica seria adotada para elicitar os requisitos da LPSD, considerando que a abordagem de LPSD proposta neste trabalho é proativa. Como a abordagem proativa trata da criação da linha de produto do início, esperava-se encontrar uma técnica capaz de elicitar requisitos eficientemente a partir do conhecimento do engenheiro de domínio em relação ao domínio da LPSD e das necessidades dos *stakeholders*. Com isso em mente, a técnica que melhor se encaixou foi a técnica de elicitação por meio da narração de cenários. O uso de cenários em processos de elicitação pode ser encontrado nos trabalhos Hong, Chiu e Shen (2005), Batista e Silva (2015), Alawairdhi e Aleisa (2011) onde se mostrou eficiente para levantar informações sobre sistemas sensíveis ao contexto.

4.2 Estudo e análise da modelagem de *features* e contextos em LPSDs

Como a proposta deste trabalho trata do domínio de sistemas sensíveis ao contexto, é necessário analisar técnicas existentes para tratar dos elementos estáticos e variáveis deste domínio. Nesta etapa do trabalho foi realizada uma revisão não sistemática da literatura em busca de técnicas e/ou abordagens existentes capazes de lidar com a modelagem de *features* e contextos em LPSD. No estudo realizado por Silva et al. (2016), contsta que o modelo de *features* é a abordagem mais usada para documentar *features* e suas variabilidades, e portanto, essa abordagem foi adotada neste trabalho.

Em relação aos contextos, também por meio de uma revisão não sistemática da literatura foram identificadas formas de definir propriedades dos contextos capazes de ajudar na construção do modelo de *features* e auxiliar no entendimento mais completo do domínio, como por exemplo, a classificação de Hong, Chiu e Shen (2005).

4.3 Análise da abordagem Provop para modelar processos de negócio de casos de uso

A abordagem Provop, foi analisada e escolhida por dar suporte a variações de contexto em um único modelo. A mesma é baseada na notação BPMN e foi vista como útil para fornecer ao engenheiro de domínio uma visão mais abrangente das interações dos usuários com

o sistema em meio a cenários de adaptações de contexto.

4.4 Identificação de técnicas utilizadas no gerenciamento da variabilidade em LPSs e LPSDs

Para identificar as técnicas de gerenciamento da variabilidade, foi realizado um estudo não sistemático na literatura. O objetivo desse estudo foi encontrar técnicas direcionadas tanto para LPSs quanto para LPSDs e passíveis de adaptação caso necessário. Por meio desse estudo foi possível identificar três atividades para compor a fase de gerenciamento da variabilidade, que foram: matriz de rastreabilidade de requisitos e *features*; modelo de *features* e; varificação do modelo de *features*.

4.5 Análise e identificação de ferramentas para apoiar o processo

A Notação para Modelagem de Processos de Negócio (BPMN, do inglês *Business Process Modeling Notation*), promove uma representação gráfica de modelagem de processos que facilita o entendimento dos usuários, e é suportada por diferentes ferramentas de software para descrever fluxos de processos (OMG, 2016). Uma delas é a ferramenta Bizagi, que será utilizada para realizar essa etapa do trabalho. A notação BPMN facilita o entendimento de desenvolvedores responsáveis por implementar a tecnologia que irá utilizar o processo, pessoas que monitoram o processo e outros usuários (WHITE, 2004). Neste trabalho, a notação BPMN também foi utilizada para a modelagem dos processos de casos de uso, como é realizado na abordagem Provop.

Para a construção do modelo de *features*, foi escolhida a ferramenta DyMMer [1], pois esta permite a criação e a edição de modelos de *features* de LPSs e LPSDs.

4.6 Validação do processo

A validação do processo foi realizada por meio de um estudo de observação, no qual os participantes realizam tarefas enquanto são observados por um experimentador. Este estudo tem o objetivo de recolher informações sobre como uma determinada tarefa é executada. Dessa forma, é possível coletar informações sobre como o processo é utilizado. A finalidade do estudo de observação foi analisar a sequência de atividades definidas no processo, visando caracterizar

[1] https://github.com/DyMMerProject/DyMMerV2

sua aplicação em relação a elicitação, modelagem e gerenciamento da variabilidade de requisitos e *features* de uma LPSDs.

4.7 Análise dos resultados

Para analisar os resultados do processo foram coletados e analisados os resultados produzidos pelos participantes em cada tarefa realizada no estudo de observação. Os principais dados coletados foram o esforço e o tempo para a execução das tarefas, assim como uma contagem de resultados previstos pela experimentadora. Além disso, também foram coletados dados acerca dos *templates* e documentos utilizados para auxiliar na realização das tarefas.

A partir dos dados coletados foi verificado que o esforço dos participantes para realizar a maioria das tarefas foi intermediário. E ainda que o processo foi capaz de auxiliar no que se propõe. A desvantagem da abordagem é que algumas tarefas não triviais e de certa forma extensas não foram automatizadas.

5 PREMIER: PROCESSO DE ELICITAÇÃO E MODELAGEM DA VARIABILIDADE DE REQUISITOS EM LPSDS

Este processo é baseado na abordagem proativa de LPSD e seu escopo proposto por este trabalho está limitado a três fases: elicitação de requisitos, modelagem de requisitos e gerenciamento de variabilidade. Como primeira atividade de definição do processo tem-se o detalhamento das atividades de alto nível, onde foram definidas as atividades de: i) elicitação de requisitos; ii) especificação de requisitos; iii) construção do modelo de *features*; iv) verificação do modelo de *features* e; v) gerenciamento da variabilidade. Estas por sua vez, dividas em macro fases. O processo tem como objetivo produzir um modelo de *features* adequado para LPSDs considerando mudanças de contexto. A estrutura do processo está representada no Quadro 2.

Quadro 2 – Estrutura do Processo

Fases	**Atividades**	**Produtos de Trabalho**
Elicitação	*Elicitação de cenários*	*Documento de Cenários*
Modelagem	*Especificação de Requisitos*	*Documento de Especificação*
Gerenciamento da Variabilidade	*Rastreamento de Requisitos e Features* *Construção do Modelo de Features* *Verificação do Modelo de Features*	*Matriz de Rastreabilidade* *Modelo de Features* *Relatório de Verificação*

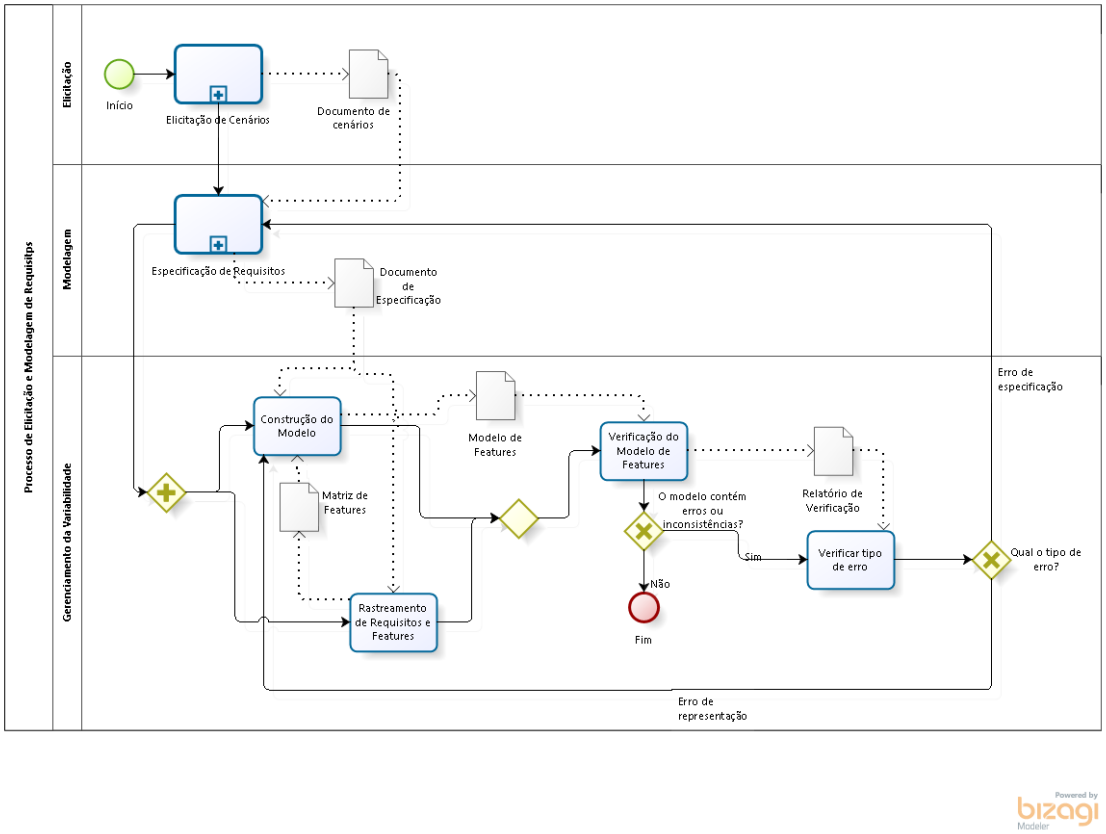
Fonte: Elaborado pela Autora.

Para modelar o processo, foi utilizada a notação BPMN em conjunto com a ferramenta Bizagi. A Notação para Modelagem de Processos de Negócio (BPMN, do inglês *Business Process Modeling Notation*), promove uma representação gráfica de modelagem de processos que facilita o entendimento dos usuários, e é suportada por diferentes ferramentas de software para descrever fluxos de processos (OMG, 2016). A notação BPMN facilita o entendimento de desenvolvedores responsáveis por implementar a tecnologia que irá utilizar o processo, pessoas que monitoram o processo e outros usuários (WHITE, 2004).

Entre os trabalhos que utilizam a notação BPMN para modelar e dar suporte a processos de software estão: Alawairdhi e Aleisa (2011), Santos et al. (2012), Chiarello (2014), Batista e Silva (2015), Yousef e Almarabeh (2015).

A Figura 5 ilustra o processo geral incluindo todas as cinco atividades propostas por este trabalho. Algumas atividades possuem tarefas que serão detalhadas nas Seções seguintes.

Figura 5 – Modelagem do Processo de Elicitação e Modelagem de Requisitos para LPSDs



Fonte: Elaborado pela Autora.

A Elicitação é a atividade inicial do processo e, é responsável por identificar todos os requisitos iniciais do sistema, os requisitos e demais informações identificadas estarão presentes em um documento de cenários, que constitui o artefato de saída dessa atividade. Em seguida tem-se a atividade de especificação, responsável por documentar as informações coletadas na atividade anterior levando em consideração o registro de peculiaridades das LPSDs, como por exemplo a classificação de contextos e *features*. O documento de saída da atividade de especificação é o artefato de entrada para a construção do modelo de *features*, uma vez que possui todas as informações necessárias, e também permite o gerenciamento de variabilidade através da criação da matriz de rastreabilidade de *features*. Por fim é realizada a atividade de verificação do modelo de *features*. Nessa fase é realizado um *checklist* sobre cada item do modelo e sobre seus relacionamentos, o relatório de erros determina se o processo chegou ao fim ou se precisa de reparos na especificação ou na representação do modelo.

Nas Seções seguintes cada atividade do processo e seus componentes serão detalhados.

5.1 Elicitação

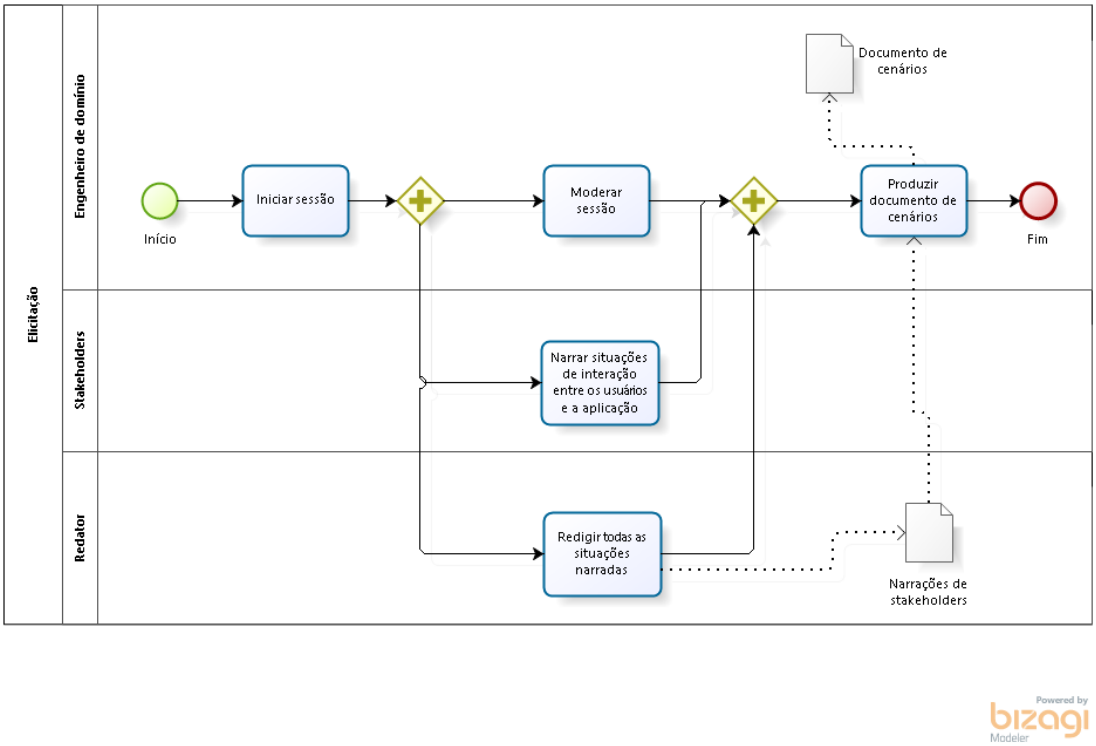
Para definir a fase de elicitação, a técnica de cenários foi estudada. Essa técnica é normalmente utilizada, no desenvolvimento tradicional de software. No entanto, levando em consideração que é necessário que haja a identificação dos contextos e de propriedades dinâmicas na construção de LPSDs, essa técnica se mostrou útil sendo combinada com outras. A aplicação de cenários em processos de elicitação pode ser encontrada nos trabalhos de Batista e Silva (2015), Hong, Chiu e Shen (2005) e Alawairdhi e Aleisa (2011).

A utilização dos cenários contribui para (ZOWGHI; COULIN, 2005):

- Entendimento do domínio;
- Identificação das fontes de requisitos;
- Análise dos *stakeholders*; e
- Elicitação dos requisitos.

Nesse sentido, essa técnica foi escolhida para contribuir na realização deste trabalho. A Figura 6 mostra o processo de elicitação modelado em BPMN.

Figura 6 – Etapa de elicitação modelada em BPMN



Fonte: Elaborado pela Autora.

O primeiro passo do processo é a elicitação de requisitos que tem o objetivo de

identificar os requisitos funcionais e não funcionais, identificar contextos, identificar *features* comuns e *features* de contexto. Para alcançar esse objetivo a técnica de Narrativa de Grupo é aplicada. Essa técnica permite a detecção de cenários e seus respectivos contextos (BATISTA; SILVA, 2015).

Os cenários permitem a identificação de requisitos e consequentemente possibilitam a captura de *features*, uma vez que uma *feature* pode ser representada como uma abstração de um requisito. Ao descobrir as *features* presentes nos cenários, são descobertos também os pontos de variação e suas variantes. Por exemplo, uma *feature* chamada "conexão"representa um ponto de variação, que pode alternar entre opções (variantes), que são tipos de conexão como *Wi-Fi*, ou *bluetooh*. Os contextos podem ser identificados, ou mapeados por meio do elemento “restrições de contexto”, ou do elemento “pré-condições”, uma vez que um contexto pode ser definido, como todo conhecimento sobre de uma entidade em um domínio, que permita particularizar uma situação, que é capaz de interferir ou ativar um comportamento, durante a interação entre o usuário e a aplicação na execução de uma tarefa (XU et al., 2012).

Nessa atividade existem três papéis envolvidos, são eles: Engenheiro de Domínio, *Stakeholders* e um Redator. O Engenheiro de domínio inicia a sessão, uma vez que como especialista do domínio do sistema a ser desenvolvido, ele deve ter conhecimento de possíveis contextos que podem ser usados para dar início a narrativa. Após a primeira atuação do Engenheiro de Domínio, ele ainda será responsável por moderar a sessão, ficando atento para que os participante cooperem sem que uns participem mais que os outros. Em paralelo os *Stakeholders* narram situações de interação entre os usuários e a aplicação, onde é possível encontrar informações sobre o comportamento do sistema, funcionalidades que ele deverá possuir e onde ele irá atuar. E por fim, os redatores registram essas histórias em um documento. Ao final da sessão os registros feitos pelo redator serão utilizados pelo Engenheiro de Domínio para a construção do documento de cenários.

O documento de cenários constitui o artefato de saída da atividade de elicitação, e contém cenários descritos em forma tabular e baseados no *template* ***CAPLUC*** (SANTOS, 2013). A construção desse documento se dá por meio da detecção de correspondências entre elementos de cenários narrados por *stakeholders* com os elementos fornecidos pelo *template* de Santos (2013). Este *template* foi adotado nesta etapa pois possui formato direcionado a sistemas sensíveis a contexto, e estes são parte do domínio de sistemas de LPSDs. Um ponto positivo em relação ao *template* ***CAPLUC*** é que ele facilita a documentação dos contextos em

que determinados casos de uso são aplicáveis.

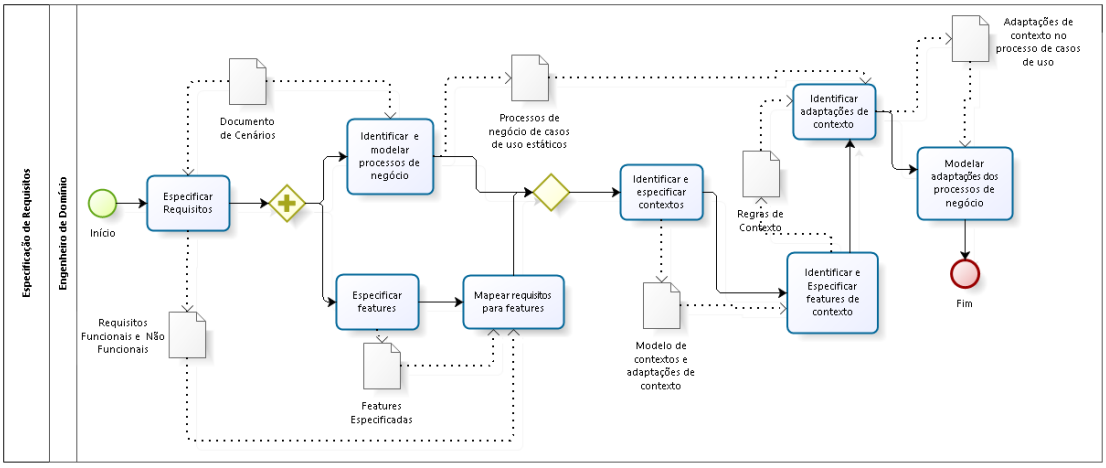
5.2 Especificação

A atividade de especificação constitui a segunda etapa do processo. É nessa atividade que os requisitos e as *features* serão organizados e estudados a fundo, a fim de garantir um melhor entendimento do domínio e de apoiar a construção do modelo de *features*.

Assim como no desenvolvimento tradicional de software, também serão documentados requisitos funcionais e não funcionais. O diferencial dessa etapa para tratar as peculiaridades das LPSDs é incluir a documentação dos contextos e suas propriedades e a modelagem dos processos de negócio dos casos de uso.

Na fase de elicitação tem-se como artefato de saída o documento de cenários. Esse documento possui cenários de caso de uso bastante detalhados, portanto no artefato de especificação os elementos contextuais e as *features* serão mais explorados que os requisitos comuns. Esse documento será útil para a identificação de processos de negócios de cada caso de uso, identificação de contextos e identificação das adaptações que o sistema de software deverá realizar. A Figura 7 ilustra as atividades da etapa de especificação.

Figura 7 – Etapa de especificação modelada em BPMN



Fonte: Elaborado pela Autora.

A etapa de especificação deste processo se preocupa principalmente com uma pré-modelagem de *features*. A tarefa inicial é a que se concentra em identificar requisitos a partir do artefato de saída da atividade de elicitação(Documento de Cenários de Casos de Uso). A partir

dele é possível identificar e modelar os processos de negócio e especificar *features*. Depois que as *features* forem especificadas, elas deverão ser associadas aos requisitos correspondentes. Em seguida os contextos devem ser identificados e especificados em um modelo de contextos. A partir desse modelo deverão ser especificadas *features* de contexto e será possível identificar pontos em que os processos de negócios sofrerão mudanças e como se adaptarão a elas. E por fim, deve-se modelar os ajustes dos processos de negócio dos casos de uso. Essa fase terá como artefato de saída um documento de especificação de requisitos e *features*.

5.2.1 Especificação de Requisitos e Features

A tarefa inicial da fase de especificação trata do detalhamento dos requisitos e das *features*. O detalhamento dos requisitos se concentra em documentar os requisitos, mapeá-los para os casos de uso e por fim, definir a prioridade desses casos de uso. O detalhamento das *features* se concentra na classificação das mesmas, sua associação com os requisitos e na definição de prioridades. No diagrama, as fases que correspondem a essa etapa são: Especificar Requisitos; Especificar *Features* e; Mapear requisitos para *features*.

O detalhamento dos requisitos ocorre como em um desenvolvimento tradicional de software e deve definir uma lista de requisitos com um identificador para cada um deles. Em seguida cada requisito deve ser associado a um ou mais casos de uso correspondentes e sua prioridade deve ser definida. Nessa tarefa também são detalhados os requisitos não funcionais requisitados e suas prioridades de acordo com a necessidade deles para o sistema.

Para estabelecer as prioridades dos requisitos e dos casos de uso, foram adotadas as denominações "essencial", "importante"e "desejável".

- **Essencial** é aquele requisito ou caso de uso sem o qual o sistema não entra em funcionamento, são aqueles que devem ser implementados impreterivelmente.
- **Importante** é aquele requisito ou caso de uso sem o qual o sistema entra em funcionamento, mas de forma insatisfatória, são aqueles que devem ser implementados mas, se não forrem o sistema pode ser implantado sem eles.
- **Desejável** é o requisito ou caso de uso que não compromete as funcionalidades básicas do sistema, ou seja, o sistema pode funcionar de forma satisfatória sem ele. São aqueles que podem ser deixados para versões posteriores, caso não haja tempo para implementá-los.

Uma *feature* pode ser definida como um conceito relevante e distintivo ou uma característica que é visível aos *stakeholders* (LEE; KANG; LEE, 2002). Neste trabalho, as

features serão classificadas com base no meta-modelo *Odyssey FEX* (OLIVEIRA et al., 2005), serão classificadas quanto a categoria, quanto a variabilidade e quanto a opcionalidade. É importante ressaltar que serão utilizados apenas fragmentos do metamodelo que se mostraram mais convenientes para este processo. A camada de técnica de implementação definida em Lee, Kang e Lee (2002), não será utilizada neste trabalho. Além disso, foi adicionada a categoria de *Feature* de Contexto.

As *features* serão classificadas dentro das camadas de Lee, Kang e Lee (2002). Quanto a categoria as *features* poderão ser *Features de Entidade*, *Features de Domínio* e *Features Tecnológicas*. Quanto a variabilidade podem ser invariantes, variantes ou pontos de variação. E ainda quanto a opcionalidade, podem ser mandatórias ou opcionais. A classificação é detalhada a seguir e adaptada de Oliveira et al. (2005).

Quanto a categoria:

- ***Feature* de Entidade**: são aquelas que representam os atores do modelo. São entidades do mundo real que atuam sobre o domínio. Se relacionam com *features* de ambiente operacional e com *features* de domínio.
- ***Feature* de Domínio**: são aquelas que estão intrinsecamente ligadas a essência do domínio. São responsáveis por representar funcionalidades e conceitos do modelo e podem ser mapeadas para casos de uso.
  - *Features* Conceituais: representam um conceito do domínio.
  - *Features* Funcionais: representam funcionalidades do domínio.
- ***Features* Tecnológicas**: é uma categoria abstrata que agrupa características que complementam a camada conceitual. Essas *features* não devem se relacionar com *features* conceituais.
  - *Features* de Ambiente Operacional: representam um ambiente que uma aplicação do domínio pode operar ou utilizar.
- ***Features* de Contexto**: são aquelas usadas para detectar e gerenciar condições de contexto (CAPILLA et al., 2014), portanto representam características que alteram o comportamento do modelo de acordo com a entrada de um determinado contexto.

Quanto a variabilidade:

- **Invariante**: são elementos fixos ou estáticos, que não são configuráveis ao domínio.
- **Variante**: é um elemento fatalmente ligado a um ponto de variação e atua como alternativa para se configurar àquele ponto de variação.

- **Ponto de Variação**: é uma *feature* que reflete a definição dos parâmetros necessários no domínio de forma abstrata. É configurável por meio de variantes.

Quanto a opcionalidade:

- **Mandatória**: *feature* que está presente em todas as derivações de mudanças de configuração do domínio.
- **Opcional**: *feature* que não está presente em todas as derivações de mudanças de configuração do domínio.

Depois de classificadas, as *features* devem ser associadas a requisitos e suas prioridades devem ser definidas. A prioridade das *features* é definida utilizando-se da mesma denominação utilizada para definir as prioridades dos requisitos e dos casos de uso. A associação de requisitos e *features* tem o objetivo de ajudar na etapa de gerenciamento da variabilidade que servirá para auxiliar o rastreamento das *features* e se dá por meio da heurística adotada por Blois et al. (2006).

5.2.2 Modelagem de Contextos

A etapa de modelagem de contextos é a responsável por produzir um modelo que contém todos os contextos estimados pelo engenheiro de domínio e os contextos narrados pela comunidade de *stakeholders*.

O modelo de contextos define a qualificação, a quantificação e a classificação de cada contexto, assim como também contribui para a especificação e documentação das *features* de contexto.

A abordagem Provop possibilita a configuração de processos sensíveis a contextos, o que permite que apenas opções relevantes para um determinado contexto sejam aplicadas ao processo. Para documentar o conjunto de variáveis do processo, a abordagem Provop utiliza de um modelo de variáveis de contextos. O modelo de variáveis de contexto contém, o nome da variável, um intervalo de valores, que podem ser tipos, níveis ou valores binários, e o comportamento, que pode ser definido como estático ou dinâmico.

Um modelo de contexto de forma tabular, parcialmente semelhante ao que é usado na abordagem Provop será utilizado nessa fase para auxiliar na construção do modelo de *features* e na modelagem dos processos de negócio de casos de uso, no entanto o atributo "nome da variável", será substituído pelo nome do contexto, o intervalo de valores será representado pela qualificação dos contextos e será incluído no modelo o atributo de quantificação do contexto.

A tarefa de qualificação do contexto se concentra em determinar intervalos ou níveis de variações para o contexto, por exemplo, uma bateria de dispositivo móvel pode possuir três (3) níveis: alta, média ou baixa, dessa forma, cada um dos níveis determina uma qualificação diferente do contexto. A tarefa de quantificação é responsável por atribuir faixas de valores ao contexto, por exemplo, a bateria do dispositivo móvel é considerada baixa quando está entre 0% e 15%.

Para a classificação dos contextos foi adotado o modelo de Hong, Chiu e Shen (2005), que classifica os contextos em três (3) categorias: i) contextos de usuários; ii) contextos computacionais e; iii) contextos físicos de ambiente. Nessa classificação, os contextos computacionais se referem a configurações dos sistemas, por exemplo, bateria, memória e resolução de tela. Os contextos de usuário estão relacionados a preferências do usuário, celendário do usuário e informações pessoais. Os contextos físicos de ambiente podem estar ligados a tempo, localização e clima.

Quadro 3 – Exemplo - Modelo de Contexto - Bateria

Contexto	**Classificação**	**Qualificação**	**Quantificação**
Bateria	Computacional	Alta Média Baixa	>75% >15% AND <75% <= 15%

Fonte: Elaborado pela Autora.

Os contextos classificados estarão presentes no modelo de contexto, e facilitarão a construção do modelo de *features* e do modelo de processos, permitindo que o engenheiro de domínio tenha uma visão mais abrangente do domínio do sistema, e realize uma melhor adequação das variações e configurações para atender a cada contexto, pois cada tipo de contexto impacta a configuração do sistema de maneira diferente. Depois da construção do modelo de contextos, deve ser construído um modelo de adaptações de contexto, pois é este quem vai determinar o conjunto de *features* pertencentes a cada configuração de cenário. Este modelo contém o identificador do cenário, os contextos correspondentes e seus valores de qualificação para um contexto específico.

Ao final dessa atividade, com os contextos já modelados, o último passo é especificar as *features* de contexto. Esta última consiste em relacionar as *features* e os contextos descrevendo se haverá ativação e desativação de uma determinada *feature* para cada contexto identificado. Nesse sentindo, a partir dessa informação tem-se as regras de contexto.

5.2.3 Processos de Negócio de Casos de Uso

Nesta etapa, uma análise do artefato inicial deve ser realizada afim de identificar os elementos dos casos de uso que serão mapeados para a modelagem dos processos de negócio, atentando-se para a identificação das *triggers* que farão com que a lógica de negócio seja criada automaticamente.

Um caso de uso pode ser definido como uma descrição de possíveis sequências de interações entre sistemas e usuários para atingir um determinado objetivo (COCKBURN, 1999). Nesse sentido, com os casos de uso já descritos de forma tabular, é possível, por meio deles extrair processos de negócios referentes as funcionalidades da aplicação em um determinado cenário. O objetivo dessa modelagem é ilustrar os passos e a interação do usuário com o sistema, para que o engenheiro de domínio possa ter uma visão do comportamento do sistema em relação ao usuário.

Para a identificação dos elementos dos cenários, serão identificadas correspondências entre os elementos dos casos de uso descritos e os elementos que compõem o processo de negócio para realizar o objetivo da tarefa. As ações são convertidas para tarefas e as *triggers*, transformadas em eventos que impactarão o fluxo do processo. O fluxo do processo é identificado a partir do cenário principal.

Na fase de elicitação, são identificados cenários estáticos e dinâmicos. Ao distinguir os elementos dos cenários, é possível capturar requisitos que possuem variações e quais são as possibilidades de variantes que elas podem possuir. Os cenários que contêm esses requisitos são classificados como dinâmicos. A dinamicidade presente nos cenários pode se dar por meio de eventos (*triggers*) ou contextos, que são capazes de alterar o fluxo principal do processo de negócio.

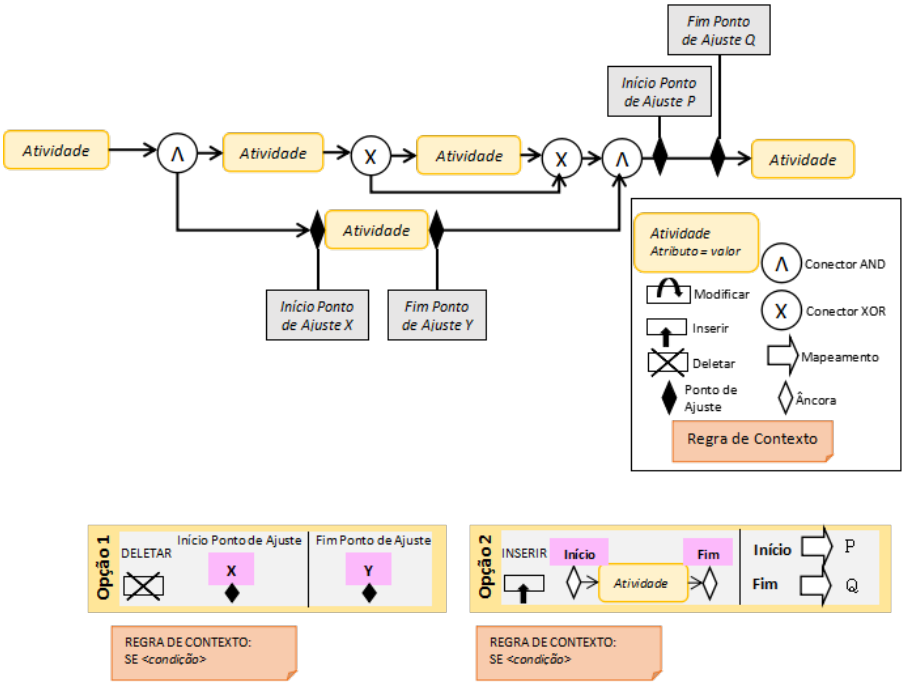
O elemento "Pré-condições", pode determinar o comportamento do sistema para uma situação em particular. Este elemento é útil para identificar os contextos dos processos, uma vez que um contexto pode ser definido, como todo conhecimento sobre de uma entidade em um domínio, que permita particularizar uma situação, que é capaz de interferir ou ativar um comportamento, durante a interação entre o usuário e a aplicação na execução de uma tarefa (XU et al., 2012). Este mesmo elemento também pode definir restrições entre as *features*. Além do elemento "pré-condições", o *template* de Santos (2013) adotado neste trabalho para a descrição dos cenários, possui um elemento chamado "Restrição de Contexto", este elemento representa

uma restrição de contexto sem a qual o caso de uso não é possível de execução, e dele também é possível extrair informações de contexto para auxiliar na modelagem das adaptações dos processos de interações dos casos de uso.

Neste ponto os contextos, processos e suas adaptações já foram identificados. Dessa forma é possível modelar os processos incluindo seus elementos dinâmicos. Para lidar com a modelagem dos processos de negócio foi adotada a abordagem Provop, pois esta possibilita o gerenciamento das variantes do processo em um único modelo. Do ponto de vista dessa abordagem, processos variantes podem ser derivados de outros processos através da adição, remoção, movimentação ou modificação de operações. A abordagem Provop, também permite configurações de processos sensíveis a contextos e utiliza um modelo que compreende um conjunto de variáveis de contexto que foi adaptado na Subseção 5.2.2.

Um ponto positivo em relação a adoção da abordagem Provop, é que ela possui uma propriedade chamada "ponto de ajuste"que se assemelha ao ponto de variação presente no modelo de *features*. O ponto de ajuste do processo permite, adequar as tarefas do processo de acordo com o contexto em que ele está inserido, levando em consideração a configuração do produto em tempo de execução. Quando um contexto determina quais variantes serão selecionadas no modelo de *features*, ele também determina que mudanças ocorrerão na realização do processo de negócio. A Figura 8 apresenta um exemplo da notação da abordagem Provop.

Figura 8 – Modelagem de Processos e Adaptações



Fonte: Adaptado de Reichert, Hallerbach e Bauer (2015).

5.3 Gerenciamento da Variabilidade

Esta Seção descreve a fase de Gerenciamento da Variabilidade e respectivas suas atividades, e está dividida em três Subseções. A Seção 5.3.1 apresenta a atividade de construção do Modelo de *Features* e o que será necessário para fazê-lo. A Seção 5.3.2 trata da aplicação de um método para o gerenciamento e rastreamento da variabilidade, dos requisitos e das *features*. E por fim, a Seção 5.3.3 apresenta tarefas direcionadas a inspeção do Modelo de *Features*.

5.3.1 Construção do Modelo de Features

A modelagem de *features* é a atividade de detectar características externamente visíveis de produtos de um determinado domínio e organizá-las em um modelo, chamado de modelo de *features* (LEE; KANG; LEE, 2002). Este modelo contém três (3) camadas, entre as camadas definidas por Lee, Kang e Lee (2002). São elas: as camadas de capacidade, ambiente operacional e de tecnologia de domínio.

O modelo será construído com o auxílio da ferramenta DyMMer (BEZERRA et al., 2016). Como citado na Seção 3.3.3.1, esta permite visualizar e analisar um modelo de *features* específico. A ferramenta possibilita editar um determinado modelo de *features* adicionando contextos, adicionando e removendo *features* e restrições, e em sua versão atual permite também a criação de modelos de *features*.

A construção do modelo de *features* é baseada nos artefatos anteriormente produzidos, eles irão determinar a representação gráfica das *features*, suas camadas, suas restrições, seus contextos e as adaptações desses contextos. Dessa forma as *features* serão naturalmente modeladas pelo Engenheiro de Domínio.

5.3.2 Rastreamento de Requisitos e Features

Watkins e Neal (1994) definem a rastreabilidade como uma conexão de relacionamento explicável entre entidades, e afirmam que a rastreabilidade de requisitos contribui significativamente para o gerenciamento de projetos. Eles ainda acrescentam: "você só pode gerenciar o que você rastreia".

Neste trabalho, como parte do gerenciamento da variabilidade será utilizada uma matriz de rastreabilidade adaptada de Pohl, Böckle e Linden (2005). A matriz de rastreabilidade de Pohl, Böckle e Linden (2005) trata de Linhas de Produto de Software convencionais, então,

para este caso ao invés de "marcar"como mandatórias as *features* que pertencem a cada aplicação derivada do modelo, cada *feature* poderá assumir dois valores diferentes. Esses valores podem ser: Mandatória e; Contexto.

5.3.3 Verificação do Modelo de features

Esta Seção se concentra nas tarefas da atividade de verificação do modelo de *features* e aborda a técnica *FM Check* (MELLO et al., 2014). Nessa atividade será realizada a tarefa de identificação de casos discrepantes encontrados dentro do modelo de *features* e uma inspeção por meio de um *checklist*.

Em relação ao modelo de *features* foram encontrados 48 casos discrepantes no trabalho de Mello et al. (2014), e estes foram definidos em sete grupos. Como primeira tarefa desta atividade os casos discrepantes identificados devem ser classificados nas categorias: omissão, fato incorreto, inconsistência, ambiguidade e informação estranha. Os grupos de casos discrepantes e o modelo para categorizá-los são apresentados no documento de verificação produzido nesta atividade.

Como mencionado na Seção 3.3.4, o *FM Check* possui um *checklist* de itens de verificação dividido em três grupos: i)verificação individual de itens; ii)verificação de relacionamento entre itens e;verificação de regras de composição. O objetivo desta atividade é assegurar a consistência e a corretude do modelo de *features* construído neste trabalho. Um modelo consistente é aquele que possui pelo menos uma configuração válida, enquanto um modelo correto é aquele que não possui *features* mortas (BEZERRA et al., 2016).

- **Verificação Individual de *Features***

  Essa verificação busca garantir que cada *feature* tem uma descrição clara, correta e objetiva. Além disso, também verifica se a *feature* realmente pertence ao domínio.

- **Verificação de Relacionamento entre *Features***

  Essa verificação guia a inspeção dos relacionamentos entre as *features*, verificando se a representação apresenta um modelo entendível e compatível com o domínio.

- **Verificação de Regras de Composição**

  Esse grupo de verificação se concentra em guiar a inspeção no que diz respeito a clareza, corretude, completude, consistência e relevância das regras de composição do modelo.

Este trabalho faz uso do *FM Check* utilizando os três grupos de verificação com a adição da verificação das *features* de contexto. O *checklist* foi adaptado de acordo com a

classificação definida na atividade de especificação de requisitos.

6 AVALIAÇÃO DO PROCESSO PREMIER

O processo PREMIeR apresentado Seção 5 se concentra em fornecer um processo capaz de apoiar a identificação de elementos candidatos a composição de uma determinada LPSD de forma proativa, dar suporte na etapa de especificação desses elementos, principalmente em relação a elementos variáveis como *features* ativadas por contextos e os próprios contextos, e ainda, auxiliar no gerenciamento da variabilidade da LPSD. Foi realizado um estudo preliminar, descrito nesta Seção para avaliar a aplicabilidade deste processo.

Esta Seção está distribuída em Objetivo, Descrição, Estudo de observação, Descrição e Procedimentos. A Subseção 6.1 apresenta o que pretende-se verificar com a aplicação do processo. A Subseção 6.2 descreve a definição do estudo realizado. E por fim, a Subseção 6.3 apresenta a descrição do estudo e os procedimentos realizados para sua execução.

6.1 Objetivo

O estudo realizado para avaliar o processo PREMIeR é do tipo estudo de observação, neste estudo o participante executa alguma tarefa enquanto é observado pelo experimentador. O objetivo deste estudo é recolher informações sobre como uma determinada tarefa é executada. Nesse sentido, é possível obter informações sobre a maneira que o processo é utilizado. Wohlin et al. (2012) propôs um modelo que permite descrever o objetivo do estudo de observação, este foi usado para descrever o objetivo do estudo realizado neste trabalho, como ilustra o Quadro 4.

Quadro 4 – Descrição do objetivo do estudo de observação.

Analisar as atividades definidas no processo PREMIeR
Com a finalidade de caracterizar sua aplicação
Em relação à identificação de requisitos, *features*, contextos, regras de contextos, adaptações de contextos e processos de casos de uso, com suas respectivas restrições, para apoiar a modelagem de *features* e a Engenharia de Requisitos de LPSDs.
Do ponto de vista do engenheiro de domínio
No contexto de alunos de graduação, analisando um cenário de um produto de software sensível ao contexto, com o uso de atividades definidas no processo PREMIeR

Fonte: Elaborado pela Autora.

6.2 Definição

O experimento foi realizado com dois alunos concludentes do curso de Engenharia de Software da Universidade Federal do Ceará - Campus Quixadá. Para seleção dos alunos participantes, estes teriam que possuir um conhecimento prévio sobre linhas de produto, e no contexto da universidade ter cursado a disciplina de Reuso de Software. Os participantes tiveram total liberdade para decidir sobre sua contribuição no experimento. O experimento foi realizado em uma única etapa, divido em três módulos e oito tarefas.

Antes de iniciar o experimento foi apresentado aos participantes o Termo de Consentimento (Apêndice C), que descreve tópicos relacionados a participação no estudo, contendo um campo de assinatura para que o participante possa confirmar a sua participação no experimento. Em seguida foi aplicado um questionário direcionado a avaliar os conhecimentos prévios dos participantes no que diz respeito as questões relacionadas a este estudo, por exemplo, Linhas de Produto de Software Dinâmicas, modelo de *features*, requisitos, e outros. Em sequência, foi apresentada uma versão geral do processo e cada participante recebeu a descrição das tarefas que deveriam ser realizadas durante o estudo (Apêndice D), juntamento com um conjunto de *templates* de artefatos relacionados as tarefas e um cenário de uso de um sistema sensível ao contexto.

O primeiro módulo possui apenas uma tarefa em que cada participante deveria identificar casos de uso a partir do dado cenário de sistema sensível ao contexto. Para realizar esta tarefa o participante teria que analisar o cenário, identificar os casos de uso e transcrevê-los para o *template* CAPLUC (SANTOS, 2013) (Anexo A).

O segundo módulo contém cinco tarefas e para a execução deste módulo foram entregues *templates* aos participantes para a documentação dos elementos identificados, estes *templates* por sua vez estão presentes no documento de especificação de requisitos (Apêndice E), que é o artefato de saída deste módulo. A tarefa inicial deste módulo é relacionada a especificação dos requisitos, e nela os participantes deveriam descrever os requisitos funcionais e não funcionais dos casos de uso identificados e associá-los a estes casos de uso, como no desenvolvimento de software tradicional. A segunda tarefa deste módulo diz respeito a especificação de *features*, nesta tarefa os participantes deveriam identificar *features* a partir dos requisitos e casos de uso identificados, e classificá-las de acordo com a descrição da classificação fornecida e associá-las a requisitos funcionais. Na terceira tarefa foi pedido aos participantes que modelassem

os casos de uso identificados sem considerar alterações de contexto na notação BPMN. A quarta tarefa é referente a modelagem de contextos, nesta tarefa foi pedido que os participantes identificassem contextos e preenchessem um modelo de contextos que contém o tipo de contexto, a qualificação e a quantificação. Ainda nesta tarefa os participantes deveriam criar dois cenários de adaptações de contexto, que no fim serviriam para gerar modelos de *features* adaptados para esse cenários especificamente. E como a última subtarefa, foi pedido também que os participantes registrassem a influência (ativação/desativação) dos contextos apresentados no cenários criados sobre as *features*. Essa influência, informa que um determinado contexto ativa ou desativa uma determinada *feature* e também pode ser lida como uma regra de contexto. Depois de realizar a modelagem dos contextos, os participantes deveriam modelar novamente os processos de casos de uso, mas dessa vez considerando as adaptações de contexto utilizando a abordagem Provop (REICHERT; HALLERBACH; BAUER, 2015);

O terceiro módulo está dividido em duas tarefas e está relacionado ao gerenciamento da variabilidade. A primeira tarefa deste módulo é preencher um *template* de matriz de rastreabilidade de requisitos e *features* (Apêndice F). Os participantes devem identificar as regras de contexto para cada cenário criado no módulo anterior e na matriz registrar as *features* de cada derivação de aplicação. Por fim, a última tarefa é construção do modelo de *features* com base nos artefatos criandos anteriormente e com o auxilio da ferramenta DyMMer (BEZERRA et al., 2016).

6.3 Estudo de observação

Esta Seção apresenta as tarefas realizadas no estudo. A Subseção 6.3.1 descreve o estudo e os dados mais importantes a serem coletados. A Subseção 6.3.2 apresenta o procedimento realizado para a execução das tarefas.

6.3.1 Descrição

Como primeira tarefa, os participantes deveriam identificar casos de uso presentes no cenário de uso dado. Eles deveriam ler o cenário fictício relacionado a um sistema de refrigerador inteligente, detectar casos de uso e transcrevê-los para o *template* CAPLUC (SANTOS, 2013). O principal objetivo dessa tarefa era observar quais casos de uso foram identificados pelos participantes com a utilização apenas do cenário de uso textual. As principais informações

coletadas nestaa tarefa foram: casos de uso identificados e; tempo decorrido para a realização da tarefa.

O segundo módulo de tarefas, relacionado a fase de modelagem do processo envolve, a especificação e modelagem de requisitos, *features*, contextos e processos de negócios de casos de uso. Nessa etapa os participantes deveriam identificar requisitos funcionais e não funcionais, *features*, contextos, processos de casos de uso e modelá-los com o auxílio dos *templates* fornecidos. O objetivo desta etapa foi observar os graus de facilidade e dificuldade em cada parte da etapa de modelagem. Os dados mais pertinentes coletados nessa etapa foram: os elementos identificados; os benefícios trazidos pelo uso dos *templates* e; o tempo para a realização das tarefas.

O terceiro módulo de tarefas, relacionado ao gerenciamento da variabilidade e envolve a rastreabilidade de requisitos e *features* e a construção do modelo de *features*. Nessas tarefas os participantes deveriam utilizar o *template* de cenários de adaptação de contextos, relacioná-lo com o *template* de regras de contextos para *features* e preencher o *template* de matriz de rastreabilidade de requisitos e *features*. Depois de preencher a matriz, os participantes deveriam construir um modelo de *features* baseado em todas as informações coletadas nas tarefas anteriores. O principal objetivo desta etapa foi identificar o grau de facilidade ou dificuldade na utilização dos *templates* e principalmente em relação ao uso da matriz de rastreabilidade e a sua utilidade para a construção do modelo de *features* no que diz respeito a adição de adaptações de cenários. Os principais dados coletados neste módulo de tarefas foram: tempo decorrido para a execução das tarefas; benefícios da utilização dos *templates* e o modelo de *features* construído pelos participantes.

6.3.2 Procedimentos

Para cada tarefa do estudo participaram dois alunos concludentes de graduação. O estudo foi realizado com a experimentadora e todos os participantes, em uma única sessão.

Como descrito na Seção 6.2, primeiramente, os participantes deveriam assinar o Formulário de Consentimento (Apêndice) e preencher o questionário para caracterização e avaliação dos conhecimentos prévios de cada participante(Apêndice C).

Em seguida foi realizada uma breve apresentação sobre o processo, as fases, os artefatos e a sequência de atividades utilizados no processo PREMIeR. A apresentação ocorreu durante 15 minutos, em média. Depois da apresentação de uma visão geral do processo, os

participantes tiveram um momento para sanar as possíveis dúvidas sobre o conteúdo apresentado pela experimentadora.

Logo depois, foram fornecidos aos participantes: a descrição das tarefas e a descrição de um cenário de uso fictício em forma textual. Para conduzir o experimento nas etapas predefinidas foi realizada a leitura de cada tarefa e foi fornecido em sequência aos participantes, um documento de cenários, um documento de especificação e um *template* referente a matriz de rastreabilidade.

Uma vez com as tarefas concluídas, os participantes deveriam, responder às questões do Formulário de Avaliação do Processo (Apêndice H), de forma a auxiliar a coleta de dados do estudo.

O tempo total para a realização da sessão foi, em média, de 3 horas e 18 minutos.

7 RESULTADOS

Esta Seção destina-se a descrever os resultados obtidos por meio do experimento descrito na Seção 6. São apresentados resultados individuais nas Seções 7.1 e 7.2. Em seguida, na Seção 7.3 é apresentada uma análise geral dos resultados referentes ao estudo aplicado ao processo PREMIeR, e por fim, na Seção 8.1 são descritos alguns pontos identificados como ameaças a validade.

O Quadro 5 apresenta um resumo da caracterização dos participantes. Nos itens da pergunta 2, os níveis de experiência variam de 1 (baixo) a 5 (excelente).

Quadro 5 – Resumo da caracterização do *Background* dos participantes

Pergunta			**P1**	**P2**
1		Formação acadêmica	Graduando	Graduando
2	2.1	LPS	3	4
	2.2	LPSD	3	4
	2.3	Modelo de features	3	4
	2.4	Modelagem de software	2	4
	2.5	Modelagem de aspectos dinâmicos	2	3
	2.6	BPMN	1	2

Fonte: Elaborado pela Autora.

Durante a realização do experimento, o tempo de realização de cada tarefa foi medido em minutos. A contabilização desses valores é apresentada no Quadro 6 e é considerado apenas o intervalo entre o início e o término de cada tarefa. A tarefa relacionada aos processos de casos de uso foi dividida em duas fases e no Quadro 6 consta a soma dos dois tempos.

Quadro 6 – Tempo decorrido nas tarefas de cada participante

	Módulo 1	**Módulo 2**				**Módulo 3**	
	Identificação de Casos de Uso	Especificação de Requisitos	Especificação de Features	Processos de Casos de Uso	Especificação de Contextos	Rastreamento de Features e Requisitos	Construção do Modelo de Features
P1	00:54:12	00:14:58	00:27:50	00:30:30	00:52:00	00:03:47	00:15:30
P2	00:40:00	00:12:52	00:26:41	00:36:33	00:30:20	00:07:30	00:15:36

Fonte: Elaborado pela Autora.

A seguir são descritos os resultados obtidos de cada participante em cada uma das tarefas realizadas no estudo de observação.

7.1 Participante 1

O participante 1 identificou os casos de uso “Notificar usuários” e “Alterar temperatura e pressurização”. O participante relatou que o esforço para realizar esta tarefa foi baixo, pois o *template* utilizado era de fácil compreensão. Apenas o primeiro caso de uso foi previsto pela experimentadora.

O participante 1 identificou o conjunto de *features* ilustrado no Quadro 7 de acordo com o *template* fornecido pelo processo. O participante relatou que o esforço para especificar tais *features* foi baixo, entretanto relatou que foi “complicado” associar *features* e requisitos.

Quadro 7 – *Features* identificadas pelo participante 1

ID	Feature	Categoria	Variabilidade	Opcionalidade
1	Tipo de conexão.Bluetooth	Feature Funcional	Variante	-
2	Tipo de conexão.Rede sem fio	Feature Funcional	Variante	-
3	Tipo de conexão	Feature Conceitual	Ponto de variação	opcional
4	Geladeira	Feature Conceitual	Invariante	mandatório
5	Notificação	Feature Funcional	Invariante	mandatório
6	Verificação de falta de produto	Feature Funcional	Invariante	mandatório
7	Detecção de quantidade de produtos	Feature Funcional	Invariante	mandatório
8	Mudança de temperatura	Feature Funcional	Invariante	mandatório
9	Mudança de pressurização	Feature Funcional	Invariante	mandatório

Fonte: Elaborado pela Autora.

O participante 1 relatou que o esforço para realizar a especificação dos contextos foi médio, e afirmou ser um pouco complicado associar os *templates* para modelar os contextos.

Quadro 8 – Contextos identificados pelo participantes 1

Contexto	Classificação	Qualificação	Quantificação
Sensor de bluetooth	Contexto computacional	true, falso	-
Sensor de Wifi	Contexto computacional	true, falso	-
Temperatura	Contexto físico	alto, médio, baixo	baixa: <5, média: >5 , alta: >10.
Produtos	Contexto de usuário	alto, médio, baixo	baixo: 7, média: >7 e <50, alto: 50
Mudança Produto	Contexto físico	true, false	-

Fonte: Elaborado pela Autora.

Sobre a modelagem dos processos de casos de uso, o participante 1 afirmou que os processos podem ser usados para explorar e comunicar as ações prováveis dos usuários quando interagem com a aplicação. O participante relatou esforço médio para a realização desta tarefa, e relatou que a abordagem adotada para modelar as adaptações de contexto no processo auxilia na identificação dos comportamentos alternativos.

Quadro 9 – Cenários de adaptação criados pelo participante 1

Cenário	Contexto	Valor de Qualificação
C1	Temperatura	baixa
	Sensor de bluetooth	TRUE
	Sensor de Wifi	TRUE
	Produtos	baixo
	Mudança produtos	FALSE
C2	Temperatura	baixa
	Sensor de bluetooth	TRUE
	Sensor de Wifi	FALSE
	Produtos	alto
	Mudança produtos	TRUE

Em relação a criação das regras para a ativação e desativação de *features* por meio de contextos, o participante 1 relatou que o esforço para o preenchimento to *template* foi médio. O Quadro 10 ilustra a regras criadas pelo participante 1.

Quadro 10 – Regras de Contexto para Ativação e Desativação de *features*

Feature	Ação	Contexto
Tipo de conexão.Bluetooth	ativado / desativado	Sensor de bluetooth true / Sensor de bluetooth false
Tipo de conexão.Rede sem fio	ativado / desativado	Sensor de Wifi true / Sensor de Wifi false
Tipo de conexão	desativado	Sensor de Wifi false and Sensor de bluetooth false
Geladeira	ativada	
Notificação	ativada	Sensor de Wifi true or Sensor de bluetooth true
Verificação de falta de produto	ativada	
Detecção de quantidade de produtos	ativada	
Mudança de temperatura	ativada	
Mudança de pressurização	ativada	

Fonte: Elaborado pela Autora.

Sobre a matriz de rastreabilidade, o participante 1 relatou esforço baixo para construí-la, e afirmou que esta pode ser facilmente preenchida com os *templates* preenchidos nas tarefas anteriores.

Quadro 11 – Matriz de rastreabilidade do participante 1

Requisitos da Aplicação	Features da Aplicação	Aplicação 1	Aplicação 2
	Tipo de conexão.Bluetooth	contexto	contexto
	Tipo de conexão.Rede sem fio	contexto	contexto
1, 2	Tipo de conexão	contexto	contexto
	Geladeira		
1, 2	Notificação	contexto	contexto
1, 2	Verificação de falta de produto	mandatória	mandatória
3, 4	Detecção de quantidade de produtos	mandatória	mandatória
3, 4	Mudança de temperatura	mandatória	mandatória
3, 4	Mudança de pressurização	mandatória	mandatória

Fonte: Elaborado pela Autora.

Em relação a modelagem das *features*, o participante 1 relatou que o processo em si e o fato de criar o modelo de *features* de forma automatizada utilizando a ferramenta DyMMer

auxiliou a modelagem e a tarefa pôde ser realizada de maneira intuitiva. O participante conseguiu realizar a tarefa com o auxílio da matriz de rastreabilidade e do *template* de cenários de adaptação preenchido anteriormente.

7.2 Participante 2

O participante 2 identificou os casos de uso "Informar produtos próximos da validade" e "Informar produtos favoritos em falta ou se a quantidade minima ou maxima foi atingida" para o cenário dado. O participante relatou que o esforço para realizar a tarefa foi baixo, pois o *template* utilizado foi intuitivo. Entretanto o mesmo também relatou a falta de experiência em relação a alguns elementos do *template* e isso pode explicar o tempo decorrido da tarefa, que foi longo. Todos os casos de uso identificados pelo participante 2 foram previstos pela experimentadora. O participante também afirmou que utilizar o caso de uso para identificar os requisitos e as *features* ajudou.

O participante 2 identificou o seguinte conjunto de *features* de acordo com o *template* fornecido pelo processo. O mesmo relatou que o esforço para a especificação das *features* foi médio.

Quadro 12 – *Features* identificadas pelo participante 2

ID	Feature	Categoria	Variabilidade	Opcionalidade
1	Enviar Notificação	Capturar informações de contexto	Invariante	Mandatório
2	Troca de Mensagem	Feature conceitual	Ponto de Variação OR	Mandatório
3	via Rede sem fio	Feature funcional	Variante	Opcional
4	via Redebluetooth	Feature funcional	Variante	Opcional
5	via tela LCD	Feature funcional	Variante	Mandatório
6	Capturar informações de contexto	Feature conceitual	Ponto de Variação OR	Mandatório
7	Produtos com quantidade mínima	Feature de contexto	Variante	Mandatório
8	Produtos com quantidade máxima	Feature de contexto	Variante	Mandatório
9	Produtos perto da data de validade	Feature de contexto	Variante	Mandatório
10	Tipo de Conexão	Feature tecnológica	Ponto de Variação XOR	Mandatório
11	via Rede sem fio	Feature de Ambiente Operacional	Variante	Opcional
12	via Redebluetooth	Feature de Ambiente Operacional	Variante	Opcional

Fonte: Elaborado pela Autora

O participante 2 afirmou que o *template* fornecido no processo ajudou a identificar e modelar contextos e suas adaptações. O participante identificou os contextos apresentados no Quadro 13. De acordo com os contextos identificados, o participante 2 criou dois cenários de adaptação de contexto, como é ilustrado no Quadro 14.

Quadro 13 – Contextos identificados pelo participante 2

Contexto	Classificação	Qualificação	Quantificação
Conectado com internet	Contexto computacional	booleano	true false
Quantidade de produtos	Contexto do usuário	mínimo máximo	750
Nível de temperatura do refrigerador	Contexto computacional	Alto Médio Baixo	>25 graus <=23 graus e >= 16 graus <16 graus
Consumo de energia	Contexto físico	Alto Médio Baixo	>30 e <=50 >=20 e <=30 <20
Conectado viabluetooth	Contexto computacional	booleano	true false

Fonte: Elaborado pela Autora.

Quadro 14 – Cenários de Adaptação criados pelo participante 2

Cenário	Contexto	Valor de Qualificação
C1	Conectado com internet	TRUE
	Quantidade de produtos	mínimo
	Nível de temperatura do refrigerador	média
	Consumo de energia	baixo
	Conectado viabluetooth	FALSE
C2	Conectado com internet	FALSE
	Quantidade de produtos	máximo
	Nível de temperatura do refrigerador	baixo
	Consumo de energia	alta
	Conectado viabluetooth	TRUE

Fonte: Elaborado pela Autora.

Em relação aos processos de casos de uso o participante 2, relatou que o esforço foi médio, pois conseguiu modelar o processo estático sem dificuldades, entretanto afirmou sentir dificuldades em modelar as adaptações de contextos dos casos de uso, e que seria válida um sessão de treinamento apenas para a abordagem utilizada para modelar esses processos. Mesmo assim o participante 2 conseguiu completar a tarefa.

O participante 2 verificou os cenários de adaptação de contexto e de acordo com os contextos definidos por ele, identificou as regras de contexto ilustradas no Quadro 15. O participante relatou que o esforço para construir as regras foi médio, e observou que poderia ser melhor representar nas regras a quantificação ao invés da qualificação.

Quadro 15 – Regras de Contexto para Ativação e Desativação de *Features*

Feature	Ação	Contexto
Enviar Notificação	Ativação	-
Troca de Mensagem	Ativação	-
via Rede sem fio	Ativação	Conectado com internet TRUE Conectado via bluetooth false
via Redebluetooth	Ativação	Conectado com internet false Conectado via bluetooth true
via tela LCD	Ativação	-
Capturar informações de contexto	Ativação	-
Produtos com quantidade mínima	Ativação	-
Produtos com quantidade máxima	Ativação	-
Produtos perto da data de validade	Ativação	-
Tipo de Conexão	Ativação	-
via Rede sem fio	Ativação	Conectado com internet TRUE Conectado viabluetooth false
via Redebluetooth	Ativação	Conectado com internet false Conectado via bluetooth true

Fonte: Elaborado pela Autora.

O participante 2 produziu a matriz de rastreabilidade ilustrada do Quadro 16 abaixo, utilizando os *templates* de especificação de *features* e os *templates* referentes aos contextos. As colunas de Aplicação 1 e Aplicação 2 correspondem a variações do modelo de *features* referentes a cada cenário de adaptação. O participante relatou que o esforço para preencher a matriz de rastreabilidade foi baixo, e que a mesma contribuiu para construção do modelo de *features*, afirmando que foi possível visualizar as possíveis diferenças na configuração do modelo.

Quadro 16 – Matriz de Rastreabilidade produzida pelo participante 2

Requisitos da Aplicação	Features da Aplicação	Aplicação 1	Aplicação 2
RF3,RF4, RF5	Enviar Notificação	mandatória	mandatória
RF3,RF4, RF5	Troca de Mensagem	mandatória	mandatória
	via Rede sem fio	ativada por contexto	desativada por contexto
	via Redebluetooth	desativada por contexto	desativada por contexto
	via tela LCD	mandatória	mandatória
RF1, RF2	Capturar informações de contexto	mandatória	mandatória
	Produtos com quantidade mínima	mandatória	mandatória
	Produtos com quantidade máxima	mandatória	mandatória
	Produtos perto da data de validade	mandatória	mandatória
	Tipo de Conexão	mandatória	mandatória
	via Rede sem fio	ativada por contexto	desativada por contexto
	via Redebluetooth	desativada por contexto	ativada por contexto

Depois do preenchimento da matriz de rastreabilidade o participante construiu um modelo de *features* com o auxílio da ferramenta DyMMer, e tendo como guia a matriz de rastreabilidade e os cenários de adaptação de contextos. O participante conseguiu cumprir a tarefa e o modelo refletiu tudo o que foi construído nos *templates* e documentos anteriores.

7.3 Análise geral

Como apresentado no Quadro 5, é possível verificar que os participantes já possuíam uma certa experiência sobre o tópicos abordados no estudo, e isso se dá pelo fato do critério de seleção dos participantes ser ter cursado a disciplina de Reuso de Software.

Em relação ao tempo utilizado nas tarefas, apresentados do Quadro 6 a tarefa mais demorada foi a tarefa inicial, e nela pode-se detectar a maior variação de tempo entre os dois participantes, que é de em média, 14 minutos. Embora os dois participantes tenham afirmado que o *template* utilizado foi intuitivo e de fácil compreensão, a atividade foi a que mais demorou, isso pode ser explicado devido a tamanho extenso do *template* utilizado.

De maneira geral os participantes afirmaram que o processo auxiliou na elicitação, modelagem e gerenciamento da variabilidade de requisitos em LPSDs, relatando a utilidade do uso dos *templates* e a facilidade no uso da ferramenta escolhida para modelagem de *features*. Além disso os participantes apontaram como vantagem, a facilidade na elicitação e gerenciamento da variabilidade, e como desvantagem o tempo que é gasto no preenchimento dos *templates* e a complexidade de utilizar a abordagem escolhida para modelar os processos de casos de uso. Como sugestões, ambos os participantes disseram que a automatização de todas as tarefas ajudaria muito, principalmente em se tratando de sistemas mais complexos.

7.4 Considerações finais sobre o experimento

Nesta Seção, foi apresentado o estudo de observação para avaliação do processo PREMIeR, que tem como objetivo fornecer um processo capaz de apoiar a identificação de elementos candidatos a composição de uma determinada LPSD de forma proativa, apoiando a modelagem desses elementos e o gerenciamento da variabilidade dos requisitos e *features*.

A partir da análise dos resultados obtidos nesses estudo, foi possível perceber que o processo foi avaliado parcialmente bem e que a execução de suas tarefas exigiu um esforço intermediário. Entretanto, esses resultados são restringidos em razão da pequena quantidade de participantes.

Fazendo uma comparação entre os resultados obtidos pelos participantes, cada participante identificou dois casos de uso, sendo que dos casos de uso identificados pelo participante 1, apenas um foi previsto pela experimentadora, e dos casos de uso identificados pelo participante 2 todos os casos de uso eram esperados pela experimentadora. Em relação ao

conjunto de *features* identificadas por cada participante, o participante 1 registrou 19% das *features* previstas, e o participante 2 registrou 38% das *features* previstas pela experimentadora. O resultado mais positivo foi sobre a identificação dos contextos, em que o participante 1 detectou 60% dos contextos esperados e o participante 2 identificou 80% dos contextos esperados, o que reafirma a eficiência na utilização de cenários de uso para a detecção de contextos.

De modo geral todos os participantes responderam que o processo auxiliou na elicitação, modelagem e gerenciamento da variabilidade para o cenário especificado no experimento. Contudo, estudos mais aprofundados devem ser realizados para confirmar esses dados, e ainda validar a etapa de verificação do processo que não foi observada no estudo.

8 CONSIDERAÇÕES FINAIS

Esta Seção descreve as considerações do finais do trabalho. Inicialmente são apresentadas as possíveis ameaças à validade dos resultados obtidos neste trabalho. Em seguida, são apresentados tópicos sobre os possíveis trabalhos futuros. Logo após são apresentadas as contribuições deste trabalho. Por fim, são descritas as conclusões sobre a execução deste trabalho.

8.1 Ameaças a validade

A seleção dos participantes do experimento foi realizada da UFC - Campus Quixadá, dentro de um grupo de alunos que cursaram a disciplina de Reuso de Software e fizeram parte de um mesmo grupo de estudos da experimentadora. Os participantes convidados para participar do estudo deveriam ter conhecimentos prévios sobre os assuntos envolvidos no estudo, dessa forma o experimento ficou limitado a contar apenas com esses participantes. Nesse sentido, os resultados podem ter sido afetados devido a relação entre os participantes e a experimentadora.

Outra ameaça é a realização do experimento com uma pequena quantidade de participantes, embora estes possuam conhecimentos sobre os tópicos envolvidos no estudo, pode impactar na extensão dos resultados obtidos.

Além disso, a utilização de um cenário em um domínio pouco conhecido pelos participantes pode ter afetado na identificação de alguns elementos.

Por fim, a atividade de verificação do processo não foi validada, portanto este estudo não é capaz de informar sobre seu uso, benefícios ou dificuldades.

8.2 Trabalhos futuros

A partir dos resultados alcançados neste trabalho, foi possível detectar pontos de melhoria e possíveis trabalhos futuros.

O processo possui atividades de certa forma extensas, e por este motivo o utilizador do processo pode perder muito tempo no preenchimento dos *templates*. Nesse sentido, um possível trabalho futuro seria a automatização das etapas do processo por meio de uma ferramenta. Ainda no sentido de reduzir o esforço dos usuários do processo, outro ponto de melhoria seria uma reorganização dos *templates*.

Neste trabalho não foi aplicada a atividade de verificação do modelo de *features*, um

trabalho futuro pode ser a validação dessa etapa utilizando um modelo de *features* de LPSD e verificar se o mesmo é capaz de garantir a completude e corretude do modelo.

8.3 Contribuições

As principais contribuições identificadas com a realização deste trabalho foram:

- Criação de *templates* para a modelagem de contextos;
- Adaptação de um *template* de rastreabilidade de requisitos de LPSs para a rastreabilidade de requisitos e *features* de LPSDs;
- Proposta de processo proativo inicial para criação de uma LPSD; e
- Utilização da abordagem Provop para auxiliar o Engenheiro de Domínio sobre a visualização e entendimento do comportamento do usuário em relação ao sistema.

8.4 Conclusões

Este trabalho propôs um processo de elicitação, modelagem e gerenciamento da variabilidade de LPSDs, denominado PREMIeR. O processo foi divido em três fases e cinco tarefas. A primeira fase tem como objetivo identificar casos de uso a partir de cenários de uso. A segunda fase se concentra em identificar e modelar requisitos, *features*, contextos e processos de casos de uso. A terceira fase se refere ao gerenciamento da variabilidade e tem foco direcionado à construção de um modelo de *features* referente a um determinado domínio, a rastreabilidade dos requisitos e *features* para diversos cenários, a análise da comunalidade e variabilidade e, a verificação do modelo de *features*.

Antes da apresentação do processo proposto por este trabalho, foi apresentado um conjunto de conceitos que fundamentam a sua execução e um grupo de outros trabalhos que se relacionam com o presente trabalho.

O processo PREMIeR foi apresentado, explorando suas fases, atividades e tarefas necessárias para a sua aplicação. Depois de definido o processo, foi apresentada a validação do mesmo, por meio da descrição do estudo de observação realizado. Sobre este estudo foram recolhidos dados que mostraram a possibilidade de identificar elementos de uma LPSD na etapa de Engenharia de Requisitos, assim como gerenciar a variabilidade desses elementos.

Um ponto positivo em relação ao processo é o uso de uma matriz que permite construir um modelo de *features* e suas adaptações em relação a vários cenários de contexto.

Além disso a modelagem de contextos permite que o Engenheiro de Domínio obtenha um maior entendimento do domínio explorando as possibilidades de configuração do modelo de *features* e também seja capaz de visualizar alterações nos casos de uso em decorrência de contextos.

Com a realização deste trabalho, espera-se que o processo PREMIeR auxilie engenheiros de domínio e pesquisadores em uma identificação mais completa dos elementos que compõem um LPSD de determinado domínio, garantindo a gerência e análise da variabilidade no ciclo de vida da LPSD.

REFERÊNCIAS

ALAWAIRDHI, M.; ALEISA, E. A scenario-based approach for requirements elicitation for software systems complying with the utilization of ubiquitous computing technologies. In: **Computer Software and Applications Conference Workshops (COMPSACW), 2011 IEEE 35th Annual**. [S.l.: s.n.], 2011. p. 341–344.

ALVES, V.; SCHWANNINGER, C.; BARBOSA, L.; RASHID, A.; SAWYER, P.; RAYSON, P.; POHL, C.; RUMMLER, A. An exploratory study of information retrieval techniques in domain analysis. In: **Software Product Line Conference, 2008. SPLC '08. 12th International**. Limerick: [s.n.], 2008. p. 67–76.

BATISTA, C.; SILVA, C. Um processo criativo de elicitação de contextos para sistemas sensíveis ao contexto. In: **XI Brazilian Symposium on Information System**. Goiânia, Brasil: [s.n.], 2015. p. 323–330.

BEZERRA, C. I. M.; BARBOSA, J.; FREIRES, J. H.; ANDRADE, R. M. C.; MONTEIRO, J. M. S. Dymmer: A measurement-based tool to support quality evaluation of dspl feature models. In: SPRINGER. **International Conference on Software Product Lines**. [S.l.], 2016.

BIRK, A.; HELLER, G. Requirements engineering: Foundation for software quality: 13th international working conference, refsq 2007, trondheim, norway, june 11-12, 2007. proceedings. In: ____. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007. cap. Challenges for Requirements Engineering and Management in Software Product Line Development, p. 300–305. ISBN 978-3-540-73031-6.

BLOIS, A. P. T. B.; OLIVEIRA, R. F. de; MAIA, N.; WERNER, C.; BECKER, K. Variability modeling in a component-based domain engineering process. In: SPRINGER. **International Conference on Software Reuse**. [S.l.], 2006. p. 395–398.

BOSCH, J.; CAPILLA, R. Dynamic variability in software-intensive embedded system families. **Computer**, v. 45, n. 10, p. 28–35, Oct 2012. ISSN 0018-9162.

CAPILLA, R.; BOSCH, J.; TRINIDAD, P.; RUIZ-CORTÉS, A.; HINCHEY, M. An overview of dynamic software product line architectures and techniques: Observations from research and industry. **Journal of Systems and Software**, Elsevier, v. 91, p. 3–23, 2014.

CHEN, L.; BABAR, M. A.; ALI, N. Variability management in software product lines: a systematic review. In: CARNEGIE MELLON UNIVERSITY. **Proceedings of the 13th International Software Product Line Conference**. [S.l.], 2009. p. 81–90.

CHIARELLO, M. A. An approach of software requirements elicitation based on the model and notation business process (bpmn). **Lecture Notes on Software Engineering**, IACSIT Press, v. 2, n. 1, p. 65, 2014.

COCKBURN, A. Writing effective use cases. **preparation for Addison-Wesley Longman. www. infor. uva. es/˜ mlaguna/is2/materiales/BookDraft1. pdf**, 1999.

DURÁN, A.; BENAVIDES, D.; BERMEJO, J. Software product-family engineering: 5th international workshop, pfe 2003, siena, italy, november 4-6, 2003. revised papers. In: ____. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004. cap. Applying System Families Concepts to Requirements Engineering Process Definition, p. 140–151. ISBN 978-3-540-24667-1.

FANTECHI, A.; GNESI, S.; JOHN, I.; LAMI, G.; DÖRR, J. Software product-family engineering: 5th international workshop, pfe 2003, siena, italy, november 4-6, 2003. revised papers. In: ____. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004. cap. Elicitation of Use Cases for Product Lines, p. 152–167. ISBN 978-3-540-24667-1.

GURP, J. V.; BOSCH, J.; SVAHNBERG, M. On the notion of variability in software product lines. In: IEEE. **Software Architecture, 2001. Proceedings. Working IEEE/IFIP Conference on**. Amsterdam, The NETHERLANDS, 2001. p. 45–54.

HALLSTEINSEN, S.; HINCHEY, M.; PARK, S.; SCHMID, K. Dynamic software product lines. **Computer**, IEEE, v. 41, n. 4, p. 93–95, 2008.

HINCHEY, M.; PARK, S.; SCHMID, K. Building dynamic software product lines. **Computer**, IEEE, n. 10, p. 22–26, 2012.

HONG, D.; CHIU, D. K.; SHEN, V. Y. Requirements elicitation for the design of context-aware applications in a ubiquitous environment. In: ACM. **Proceedings of the 7th international conference on Electronic commerce**. Taiwan, China, 2005. p. 590–596.

INVERARDI, P.; MORI, M. Model checking requirements at run-time in adaptive systems. In: **Proceedings of the 8th Workshop on Assurances for Self-adaptive Systems**. New York, NY, USA: ACM, 2011. (ASAS '11), p. 5–9. ISBN 978-1-4503-0853-3.

KANG, K. C.; COHEN, S. G.; HESS, J. A.; NOVAK, W. E.; PETERSON, A. S. **Feature-oriented domain analysis (FODA) feasibility study**. [S.l.], 1990.

LEE, K.; KANG, K. C.; LEE, J. Concepts and guidelines of feature modeling for product line software engineering. In: SPRINGER. **International Conference on Software Reuse**. [S.l.], 2002. p. 62–77.

LINDEN, F. J. Van der; SCHMID, K.; ROMMES, E. **Software product lines in action: the best industrial practice in product line engineering**. [S.l.]: Springer Science & Business Media, 2007.

MELLO, R. M. de; NOGUEIRA, E.; SCHOTS, M.; WERNER, C. M. L.; TRAVASSOS, G. H. Verification of software product line artefacts: A checklist to support feature model inspections. **J. UCS**, v. 20, n. 5, p. 720–745, 2014.

MELLO, R. M. de; PEREIRA, W. M.; TRAVASSOS, G. H. Activity diagram inspection on requirements specification. In: IEEE. **Software Engineering (SBES), 2010 Brazilian Symposium on**. [S.l.], 2010. p. 168–177.

MENDONCA, M.; BRANCO, M.; COWAN, D. Splot: software product lines online tools. In: ACM. **Proceedings of the 24th ACM SIGPLAN conference companion on Object oriented programming systems languages and applications**. [S.l.], 2009. p. 761–762.

NEIVA, D.; ALMEIDA, F. C. D.; ALMEIDA, E. S. D.; MEIRA, S. L. A requirements engineering process for software product lines. In: IEEE. **Information Reuse and Integration (IRI), 2010 IEEE International Conference on**. Las Vegas, NV, USA, 2010. p. 266–269.

NORTHROP, L.; CLEMENTS, P.; BACHMANN, F.; BERGEY, J.; CHASTEK, G.; COHEN, S.; DONOHOE, P.; JONES, L.; KRUT, R.; LITTLE, R. et al. A framework for software product line practice, version 5.0. **SEI.–2007–http://www. sei. cmu. edu/productlines/index. html**, 2007.

OLIVEIRA, R. F. de; BLOIS, A. P.; VASCONCELOS, A.; WERNER, C. Metamodelo de características da notação odyssey-fex: Descrição de classes. 2005.

OMG. **Object Management Group. Business Process Model and Notation**. 2016. Acessado em 26 de Maio 2016. Disponível em: <http://www.bpmn.org/>.

POHL, K.; BÖCKLE, G.; LINDEN, F. J. van D. **Software product line engineering: foundations, principles and techniques**. [S.l.]: Springer Science & Business Media, 2005.

PRESSMAN, R. S. **Engenharia de software: uma abordagem profissional. 7.ed**. Porto Alegre: AMGH Ed., 2011.

REICHERT, M.; HALLERBACH, A.; BAUER, T. Lifecycle management for business process variants. In: **Handbook on Business Process Management (1), 2nd edition**. Berlin Heidelberg: Springer, 2015. p. 251–278.

SANTOS, E.; PIMENTEL, J.; CASTRO, J.; FINKELSTEIN, A. On the dynamic configuration of business process models. In: **Enterprise, Business-Process and Information Systems Modeling**. [S.l.]: Springer, 2012. p. 331–346.

SANTOS, I. d. S. **Um ambiente para geração de cenários de testes para linhas de produtos de software sensíveis ao contexto**. Dissertação (Mestrado), 2013.

SANTOS, I. de S.; ANDRADE, R. M. de C.; NETO, P. d. A. dos S. A use case textual description for context aware spl based on a controlled experiment. In: CITESEER. **CAiSE Forum**. [S.l.], 2013. p. 1–8.

SAWYER, P.; BENCOMO, N.; WHITTLE, J.; LETIER, E.; FINKELSTEIN, A. Requirements-aware systems: A research agenda for re for self-adaptive systems. In: **2010 18th IEEE International Requirements Engineering Conference**. [S.l.: s.n.], 2010. p. 95–103. ISSN 1090-705X.

SILVA, C. T.; BORBA, C.; CASTRO, J. G2spl: Um processo de engenharia de requisitos orientada a objetivos para linhas de produtos de software. In: **WER**. Cuenca, Equador: [s.n.], 2010. p. 5–16.

SILVA, L. M. P. da; BEZERRA, C. I. M.; ANDRADE, R. M. C.; MONTEIRO, J. M. S. Requirements engineering and variability management in dspls domain engineering: A systematic literature review. In: **Proceedings of the 18th International Conference on Enterprise Information Systems**. Rome, Italy: [s.n.], 2016. p. 544–551. ISBN 978-989-758-187-8.

SILVA, L. M. Pedro da. **ENGENHARIA DE REQUISITOS E GERENCIAMENTO DE VARIABILIDADE NA ENGENHARIA DE DOMÍNIO DE LINHAS DE PRODUTO DE SOFTWARE DINÂMICAS: UMA REVISÃO SISTEMÁTICA DA LITERATURA**. 112 f. Monografia (Graduação) — Universidade Federal do Ceará, Quixadá, 2016.

SOMMERVILLE, I. **Engenharia de Software. 9. ed.** São Paulo, SP: Pearson Prentice Hall, 2011.

SVAHNBERG, M.; GURP, J. van; BOSCH, J. A taxonomy of variability realization techniques: Research articles. **Softw. Pract. Exper.**, John Wiley & Sons, Inc., New York, NY, USA, v. 35, n. 8, p. 705–754, jul. 2005. ISSN 0038-0644.

WATKINS, R.; NEAL, M. Why and how of requirements tracing. **Ieee Software**, IEEE, v. 11, n. 4, p. 104–106, 1994.

WHITE, S. A. Introduction to bpmn. **IBM Cooperation**, v. 2, n. 0, p. 0, 2004.

WOHLIN, C.; RUNESON, P.; HÖST, M.; OHLSSON, M. C.; REGNELL, B.; WESSLÉN, A. **Experimentation in software engineering**. [S.l.]: Springer Science & Business Media, 2012.

XU, C.; CHEUNG, S.; MA, X.; CAO, C.; LU, J. Adam: Identifying defects in context-aware adaptation. **Journal of Systems and Software**, v. 85, n. 12, p. 2812 – 2828, 2012. ISSN 0164-1212. Self-Adaptive Systems.

YOUSEF, R.; ALMARABEH, T. An enhanced requirements elicitation framework based on business process models. **Scientific Research and Essays**, Academic Journals, v. 10, n. 7, p. 279–286, 2015.

ZOWGHI, D.; COULIN, C. Requirements elicitation: A survey of techniques, approaches, and tools. In: ____. **Engineering and Managing Software Requirements**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005. p. 19–46. ISBN 978-3-540-28244-0.

APÊNDICE A – FORMULÁRIO DE CONSENTIMENTO

1. **OBJETIVO DO ESTUDO**

   Este estudo se concentra em caracterizar o processo PREMIeR em relação ao apoio na elicitação e modelagem de requisitos e features no desenvolvimento de Linhas de Produto de Software Dinâmicas (LPSDs).

2. **IDADE**

   Eu declaro ter mais de 18 anos de idade e concordar em participar de um estudo conduzido por Amanda Oliveira de Sousa na Universidade Federal do Ceará - Campus Quixadá.

3. **PROCEDIMENTO**

   Este estudo será realizado em três etapas. Na primeira etapa é aplicado um questionário para a avaliação dos conhecimentos prévios dos participantes. Em seguida é apresentada uma visão geral do processo e o conjunto de artefatos que será utilizado neste estudo. Ainda nessa etapa, é feita uma breve apresentação das notações e estruturas utilizadas para o preenchimento dos artefatos. Na segunda etapa, os participantes deverão analisar um cenário de um sistema sensível ao contexto e preencher os artefatos presentes no processo. Essa etapa é dividida em 2 atividades. Essas atividades incluem: i) elicitação e; ii) especificação; A terceira etapa, é apresentado o processo de utilização da ferramenta que suporta o processo PREMIeR. Ainda nesta etapa, os participantes deverão construir um modelo de features com base nos artefatos construídos na etapa anterior e com o auxílio da ferramenta DyMMer. Por fim, um questionário sobre o estudo realizado é preenchido pelo participante.

4. **CONFIDENCIALIDADE**

   Toda informação coletada neste estudo é confidencial, e meu nome não será divulgado. Da mesma forma, me comprometo a não comunicar os meus resultados enquanto não terminar o estudo, bem como manter sigilo das técnicas e documentos apresentados e que fazem parte do experimento.

5. **BENEFÍCIOS E LIBERDADE DE DESISTÊNCIA**

Eu entendo que os benefícios que receberei deste estudo são limitados ao aprendizado do material que é distribuído e apresentado. Também entendo que sou livre para realizar perguntas a qualquer momento, solicitar que qualquer informação relacionada a minha pessoa não seja incluída no estudo ou comunicar minha desistência de participação. Por fim, declaro que participo de livre e espontânea vontade com o único intuito de contribuir para o avanço e desenvolvimento de técnicas e processos para a Engenharia de Software.

6. **PESQUISADOR RESPONSÁVEL**

   Amanda Oliveira de Sousa

   Universidade Federal do Ceará - UFC Campus Quixadá

7. **PROFESSORA RESPONSÁVEL**

   Profa. Dra.Carla Ilane Moreira Bezerra

   Universidade Federal do Ceará - UFC Campus Quixadá

_____, ____de_____de 201____

_____

Nome (em letra de forma)

_____

Assinatura

APÊNDICE B – MODELAGEM DE LINHAS DE PRODUTOS DE SOFTWARE DINÂMICAS - FORMULÁRIO CONHECIMENTO PRÉVIO DOS PARTICIPANTES

Formulário de Conhecimento Prévio do experimento controlado Elicitação e Modelagem de Requisitos de LPSDs para caracterização do participante. Este formulário contém algumas perguntas sobre sua experiência acadêmica e profissional.

B.1 PARTE 1 - PERFIL DO ESPECIALISTA

1. **Nome Completo:**

   _____

2. **E-mail:**

   _____

3. **Instituição:**

   _____

4. **Qual o seu nível de formação acadêmica ?**
   (a) Pós-Doutorado
   (b) Doutorado
   (c) Doutorando
   (d) Mestrado
   (e) Mestrando
   (f) Graduado
   (g) Graduando

5. **Quantos anos desde a sua formação acadêmica ?**
   (a) Menos de 1 ano
   (b) Mais de 1 ano e até 5 anos
   (c) Mais de 5 anos e menos de 10 anos
   (d) Mais de 10 anos

B.2 PARTE 2 - CONHECIMENTO TÉCNICO

Esta Seção será utilizada para compreender quão familiar você é com o domínio que será utilizado para as atividades durante o experimento. Por favor, indique o grau de experiência nesta seção, selecionando a opção que melhor se encaixa em seu perfil.

1. **Uma Linha de Produto de Software é ?**
   (a) Um conjunto de sistemas de software intensivos que compartilham um conjunto comum e gerenciado de *features*, desenvolvido a partir de um conjunto de core *assets* de uma maneira prescrita
   (b) Um conjunto de produtos construídos a partir de uma plataforma de componentes
   (c) Não estou certo sobre isso

2. **Qual o seu tempo de experiência em Linhas de Produtos de Software (LPS) ?**
   (a) Menos de 1 ano
   (b) Mais de 1 ano e até 5 anos
   (c) Mais de 5 anos e menos de 10 anos
   (d) Mais de 10 anos

3. **Você já aplicou a abordagem de LPS para desenvolver software ?**
   (a) Sim, mas somente no domínio de pesquisa
   (b) Sim, mas somente no domínio da indústria
   (c) Sim, em ambos, pesquisa e indústria
   (d) Não

4. **Qual o seu tempo de experiência em Linhas de Produtos de Software Dinâmicas (LPSD) ?**
   (a) Menos de 1 ano
   (b) Mais de 1 ano e até 5 anos
   (c) Mais de 5 anos e menos de 10 anos
   (d) Mais de 10 anos

5. **Você já usou uma abordagem de LPSD para desenvolver software ?**
   (a) Sim, mas somente no domínio de pesquisa
   (b) Sim, mas somente no domínio da indústria
   (c) Mais de 5 anos e menos de 10 anos
   (d) Sim, em ambos domínios pesquisa e indústria
   (e) Não

6. **Qual o seu tempo de experiência em modelagem de software ?**
   (a) Menos de 1 ano
   (b) Mais de 1 ano e até 5 anos
   (c) Mais de 5 anos e menos de 10 anos
   (d) Mais de 10 anos

7. **Você já usou ou analisou uma técnica de modelagem para desenvolver software ?**
   (a) Sim, mas somente no domínio de pesquisa
   (b) Sim, mas somente no domínio da indústria
   (c) Sim, em ambos domínios pesquisa e indústria
   (d) Não

8. **Qual o seu tempo de experiência em modelagem de aspectos dinâmicos ?**
   (a) Menos de 1 ano
   (b) Mais de 1 ano e até 5 anos
   (c) Mais de 5 anos e menos de 10 anos
   (d) Mais de 10 anos

9. **Você já usou ou analisou uma técnica de modelagem de aspectos dinâmicos para desenvolver software?**
   (a) Sim, mas somente no domínio de pesquisa
   (b) Sim, mas somente no domínio da indústria
   (c) Sim, em ambos domínios pesquisa e indústria
   (d) Não

10. **Qual o seu tempo de experiência em modelagem de requisitos de LPSDs ?**

    (a) Menos de 1 ano

    (b) Mais de 1 ano e até 5 anos

    (c) Mais de 5 anos e menos de 10 anos

    (d) Mais de 10 anos

11. **Você já usou ou analisou uma técnica de elicitação e modelagem de requisitos e features para desenvolver software ?**

    (a) Sim, mas somente no domínio de pesquisa

    (b) Sim, mas somente no domínio da indústria

    (c) Sim, em ambos domínios pesquisa e indústria

    (d) Não

12. **Por favor, indique o grau de sua experiência nos tópicos abaixo, seguindo a escala de 5 pontos:**

	Excelente	**Alto**	**Bom**	**Médio**	**Baixo**
LPS					
LPSD					
Modelo de features					
Modelagem de software					
Modelagem de aspectos dinâmicos					
Gerenciamento da Variabilidade					
BPMN					

APÊNDICE C – DESCRIÇÃO DO CENÁRIO DE USO

Neste momento você vai assumir o papel do Engenheiro de Domínio contratado por uma empresa de desenvolvimento de software. Essa empresa almeja por meio da abordagem proativa, construir uma linha de produtos para sistemas de refrigeradores inteligentes. Para isso você deve executar um conjunto de atividades que garantam o entendimento e a documentação desse domínio.

Maria Chiquinha e Genaro Meubem são um casal que viaja bastante a trabalho, mas quando estão em casa gostam de caprichar nas refeições. Chiquinha sempre bebe leite pela manhã e para Genaro não pode faltar ovos. Para ajudar a garantir a presença de produtos favoritos e sua validade, e economizar energia abrindo menos a geladeira, Maria Chiquinha convenceu Genaro a comprar um refrigerador inteligente. Após a aquisição, passou a receber notificações em seus dispositivos móveis por meio da rede sem fio ou por bluetooth, ou ainda na tela LCD presente no refrigerador, sobre os produtos previamente cadastrados. Na tela LCD pode ser visualizada uma imagem dos alimentos armazenados. O refrigerador monitora os produtos por meio de tags eletrônicas, e informa quais produtos estão perto de perder a validade (por meio da leitura do código de barras do produto), se algum produto favorito está em falta ou se a quantidade mínima ou máxima foi atingida. Maria definiu a quantidade mínima de produtos em 7 unidades, e a máxima em 50 unidades. Para diferentes quantidades de produtos armazenados o refrigerador altera suas configurações de resfriamento e pressurização e notifica os usuários. Além disso essas mesmas configurações podem refletir a quantidade de vezes em que a porta é aberta em um intervalo de tempo.

APÊNDICE D – DESCRIÇÃO DE TAREFAS

D.1 Instruções

Este é um estudo de observação, por isso, sempre que possível, verbalize seus pensamentos, para que o experimentador possa melhor avaliar os resultados obtidos. Pergunte e comente tudo que achar necessário.

D.2 Contextualização

Linhas de Produto de Software têm sido utilizadas para produzir famílias de softwares capazes de reconfiguração em tempo de execução. No entanto, ainda existem lacunas em seu desenvolvimento, começando na etapa de Engenharia de Requisitos, onde o conceito de variabilidade já está inserido.

O desenvolvimento de uma LPSD é um desafio, pois já na fase de Engenharia de Requisitos é necessário identificar aspectos dinâmicos que o sistema deverá possuir e é onde o conceito de variabilidade é introduzido. As atividades relacionadas a Engenharia de Requisitos de LPSDs podem exigir um esforço significativo, uma vez que é necessário tratar tanto elementos convencionais, presentes no desenvolvimento tradicional de software, quanto elementos particulares de LPSDs, como por exemplo, features, contextos e suas adaptações.

D.3 Tarefas

1. Elicitação
    - Análise de Cenários. Essa é a primeira atividade e consiste em identificar os elementos estáticos e dinâmicos da LPSD, para isso você deve seguir as seguintes instruções:
        - Leia com atenção a descrição do cenário contido no anexo I.
        - Após a leitura, analise o texto novamente e identifique casos de uso.
        - De acordo com o cenário descrito no anexo I e com os cenários que você identificou, preencha o Template de Casos de Uso Sensíveis ao Contexto, contido no documento de cenários.
2. Especificação
    - Essa atividade consiste em modelar e especificar requisitos, features, contextos e suas

adaptações, processos de negócio de casos de uso e suas adaptações. Para realizar essa atividade realize as tarefas a seguir:

- Requisitos
  - Identifique requisitos funcionais, escreva-os e atribua um identificador para cada um.
  - Identifique requisitos não funcionais, escreva-os, atribua um identificador e uma propriedade para cada um.
  - Em seguida relacione cada requisito ao caso de uso que ele está associado.
- Modelagem de casos de uso estáticos
  - Modele processos de negócio de casos de uso estáticos utilizando a notação bpmn a partir dos casos de uso identificados.
- *Features*
  - Você deve analisar o informativo sobre a classificação das features e de acordo com a estrutura apresentada você deve classificá-las.
  - Identifique e Classifique as features.
  - Agora você deve associá-las aos requisitos funcionais.
- Contextos
  - Agora você pode consultar o documento de cenários e identificar contextos presentes no cenário descrito.
  - Com os contextos identificados, agora você pode classificá-los. Defina o valor para o tipo de contexto de acordo com a classificação ilustrada, atribua valores de qualificação, e para cada um destes atribua valores de quantificação. EX: contexto: bateria; qualificação: alta, média, baixa; quantificação: 3, 2, 1.
  - Modele as adaptações de contexto em forma tabular como descreve a estrutura. Considere o cenário principal usado no começo do processo.
  - Agora você que você sabe quais os contextos presentes no cenário, você deve definir as features de contexto. Você deve informar se em um determinado contexto uma feature x é ativada ou desativada. Veja bem, para cada definição de feature de contexto que você realiza, você tem uma regra de adaptação de contexto!
- Adaptações de processos de negócio

    * Muito Bem! Estamos quase no fim! Agora você já sabe quais requisitos, features e contextos estão presentes no cenário. Agora é o momento de verificar como cada caso de uso é executado.
    * Consulte novamente o documento de cenários e identifique o fluxo de cada caso de uso identificado. Em seguida analise a notação/estrutura da abordagem Provop, presente no anexo V.
    * Consulte um exemplo de aplicação da estrutura no anexo IV. Se você entendeu a estrutura para esta etapa, mãos à obra! Modele as adaptações de contextos.

3. Gerenciamento da Variabilidade
    - Rastramento de Requisitos e *Features*. Esta atividade dá início a fase de gerenciamento da variabilidade da LPSD. Aqui será produzida uma matriz de rastreabilidade de requisitos e features. Agora você vai identificar as possíveis derivações de aplicações da LPSD siga as instruções a seguir:
        – Analise a associação de requisitos e features, e depois analise as features de contexto e o modelo de adaptação de contextos. Essa tarefa fará você identificar quais requisitos e features pertencem a quais derivações.
        – Analise a estrutura da matriz de rastreabilidade.
        – Agora forme possíveis derivações da linha, considerando adaptações de contexto. Marque com "M" as features mandatórias para uma determinada aplicação e com "C" as features que são ATIVADAS por contexto. Atenção! Cada aplicação derivada corresponde a um cenário de adaptação de contexto.
    - Construção do Modelo de *Features*
        – Para a construção usaremos a ferramenta DyMMer. Para a realização dessa atividade você deve seguir as instruções:
            * Execute a ferramenta DyMMer.
            * Crie um novo modelo.
            * Adicione features de acordo com a necessidade do modelo.
            * Adicione os cenários de adaptações de contextos.

APÊNDICE E – ESPECIFICAÇÃO DE REQUISITOS

E.1 REQUISITOS

Os requisitos deste documento são divididos em requisitos funcionais e não funcionais, e a eles são atribuídas prioridades, neste documento definidas como “essencial”, “importante” e “desejável”. Essa mesma definição para prioridades dos requisitos será aplicada na Seção referente a especificação das *features*.

- **Essencial** é aquele requisito ou caso de uso sem o qual o sistema não entra em funcionamento, são aqueles que devem ser implementados impreterivelmente.
- **Importante** é aquele requisito ou caso de uso sem o qual o sistema entra em funcionamento, mas de forma insatisfatória, são aqueles que devem ser implementados mas, se não forrem o sistema pode ser implantado sem eles.
- **Desejável** é o requisito ou caso de uso que não compromete as funcionalidades básicas do sistema, ou seja, o sistema pode funcionar de forma satisfatória sem ele. São aqueles que podem ser deixados para versões posteriores, caso não haja tempo para implementá-los.

E.1.1 REQUISITOS FUNCIONAIS

Identificador	**Requisito Funcional**
RFX	Descrição do requisito

E.1.2 REQUISITOS NÃO-FUNCIONAIS

Identificador	**Requisito Não-Funcional**	**Prioridade**
RFX	Descrição do requisito	Nível de prioridade

E.1.3 ASSOCIAÇÃO DE REQUISITOS E CASOS DE USO

Requisitos	**Caso de Uso**		**Prioridade**
Requisito(s) associado(s).	Descrição do caso de uso	[Identificador do Caso de Uso]	Nível de prioridade

E.2 *features*

Identificador	**RF**	***feature***
FX	RFX (se houver)	Descrição da *feature*

E.2.1 CLASSIFICAÇÃO DE feature

Quanto a Categoria: Indica o tipo da *feature* de acordo com o Odyssey-FEX.

- ***feature* de Entidade:** são aquelas que representam os atores do modelo. São entidades do mundo real que atuam sobre o domínio. Se relacionam com *features* de ambiente operacional e com *features* de domínio.
- ***feature* de Domínio:** são aquelas que estão intrinsecamente ligadas a essência do domínio. São responsáveis por representar funcionalidades e conceitos do modelo e podem ser mapeadas para casos de uso
  - ***features* Conceituais:** representam um conceito do domínio
  - ***features* Funcionais:** representam funcionalidades do domínio.
- ***features* Tecnológicas:** é uma categoria abstrata que agrupa características que complementam a camada conceitual. Essas *features* não devem se relacionar com *features* conceituais.
  - ***features* de Ambiente Operacional:** representam um ambiente que uma aplicação do domínio pode operar ou utilizar.
- ***features* de Contexto:** são aquelas usadas para detectar e gerenciar condições de contexto (CAPILLA et al., 2014), portanto representam características que alteram o comportamento do modelo de acordo com a entrada de um determinado contexto.

Quanto a Variabilidade: Indica o a possibilidade de variação para cada *feature*.

- **Invariante:** são elementos fixos ou estáticos, que não são configuráveis ao domínio.
- **Variante:** é um elemento fatalmente ligado a um ponto de variação e atua como alternativa para se configurar àquele ponto de variação.
- **Ponto de Variação:** é uma *feature* que reflete a definição dos parâmetros necessários no domínio de forma abstrata. É configurável por meio de variantes.

Quanto a Opcionalidade

- **Mandatória:** *feature* que está presente em todas as derivações de mudanças de configuração do domínio.
- **Opcional:** *feature* que não está presente em todas as derivações de mudanças de configuração do domínio.

ID *feature*	Categoria	Variabilidade	Opcionalidade
Identificador da *feature*	Nome da categoria	Tipo de variabilidade	Tipo de opcionalidade

E.3 CONTEXTOS

O modelo de contextos usa da classificação de Hong, Chiu e Shen(2005) para categorizar os tipos de contextos identificados no cenários. Dessa forma, um contexto pode ser classificado em três tipos:

- **Contexto computacional -** é aquele que se refere configurações dos sistemas, por exemplo, bateria, memória, resolução de tela.
- **Contexto físico -** de ambiente está relacionado, por exemplo, com localização, clima, tempo.
- **Contexto de usuário -** é aquele que está relacionado com preferências do usuário, agendas, informações pessoais.

A estrutura no modelo é descrita como segue:

- Modelo de Contextos

Contexto	Classificação	Qualificação	Quantificação
Nome do contexto	Tipo de Contexto	Qualificação do contexto [intervalo de valores, booleano]	Valoração para cada item de qualificação

Para auxiliar na identificação de aplicações derivadas para cada cenário a estrutura abaixo deve ser replicada.

- Modelo de Adaptações de Contexto

Cenário	Contexto	Valor de Qualificação
Identificador do Cenário	Nome do contexto	"[intervalo de valores booleano"

E.3.1 REGRAS DE CONTEXTO PARA ATIVAÇÃO E DESATIVAÇÃO DE features

feature	**Ação**	**Contexto**
Nome da *feature*	Ativação/Desativação	Nome do contexto associado

APÊNDICE F – MATRIZ DE RASTREABILIDADE DE REQUISITOS E *FEATURES*

F.1 NOTAÇÃO/ESTRUTURA

Requisitos da Aplicação	Fatures da Aplicação	Aplicação 1	Aplicação 2	Aplicação 3

APÊNDICE G – VERIFICAÇÃO

G.1 CASOS DISCREPANTES EM MODELOS DE *FEATURES*

Um caso discrepante é uma situação genérica que pode ser encontrada em artefatos de software e configura uma discrepância, onde poderia ser detectado um defeito depois de realizada uma inspeção.

G.1.1 TIPOS DE CASOS DISCREPANTES

Identificador	Nome do Caso Discrepante	Descrição
CDA	Descrição da *feature*	Verifica a clareza da descrição da *feature* e se esta deve ser considerada no escopo do domínio de acordo com a descrição textual do domínio.
CDB	Categoria da *feature*	Verifica a propriedade de classificação das *features*.
CDC	*features* Opcionais/Mandatórias	Verifica se a propriedade de opcionalidade de cada *feature* foi aplicada corretamente.
CDD	Variabilidade	Verifica a classificação de uma *feature* em relação a cada ponto de variação e suas *features* alternativas.
CDE	Relacionamentos de Dependência e Mutualidade Exclusiva	Verifica a seleção de *features* conjuntas(dependência) ou a inviabilidade de uma ou mais *features* selecionada sem um conjunto (exclusão mútua) para o desenvolvimento de produtos.
CDF	Regras de Raciocínio e Composição	Verifica a descrição textual lógica e as regras de composição do modelo.
CDG	Visão geral do modelo	Verificação do modelo como um todo. As *features* são avaliadas quanto a sua pertinência no escopo do domínio e a sua adequação para o nível de abstração usado para entender o modelo e futuras aplicações na implementação do domínio.

G.1.2 TIPOS DE DEFEITOS EM CASOS DISCREPANTES

Tipo de defeito	Descrição
Omissão	Algumas informações sobre o domínio não são propriamente incluídas no modelo de *features*.
Fato incorreto	Alguma informação ou comportamento do modelo de *features* contradiz sua especificação de domínio.
Inconsistência	Algum elemento do modelo de *features* não está consistente com outro elemento do mesmo modelo de *feature*.
Ambiguidade	Alguma informação do modelo de *feature* não está clara, permitindo múltiplas interpretações para o domínio especificado.
Informação estranha	Alguma informação no modelo de *features* está fora do escopo do domínio.

G.1.3 CATEGORIZAÇÃO DE CASOS DISCREPANTES

Este quadro deve ser preenchido apenas depois do *checklist* de verificação de itens.

Caso Discrepante	**Categorias para classificação de casos discrepantes (assinale com um X)**				
	Omissão	**Fato Incorreto**	**Inconsistência**	**Ambiguidade**	**Informação estranha**
CDA					
CDB					
CDC					
CDD					
CDE					
CDF					
CDG					

G.2 *CHECKLIST* DE VERIFICAÇÃO

Respostas possíveis para os itens de verificação:

- Sim
- Não
- N.A (Não se Aplica)

#	**Verificação individual dos itens (cada *feature*)**	**Resposta**	**Observação**
1	Todas as *features* estão claras e corretamente descritas?		
2	A opcionalidade de cada *feature* (opcional/mandatória) é descrita em conformidade com a especificação do domínio?		
3	É possível identificar a categoria da*feature*pela sua descrição no domínio?		
4	As*features*que representam aspectos conceituais do domínio, são devidamente classificadas como *features* Conceituais?		
5	As *features* que representam aspectos funcionais do domínio, são devidamente classificadas como *features* Funcionais?		
6	As *features* que representam uma entidade real (ator) do domínio, são devidamente classificadas como *features* de Entidade?		
7	As *features* que representam atributos do ambiente relacionadas ao domínio, são devidamente classificadas como *features* de Ambiente Operacional?		
8	As *features* que representam alguma tecnologia usada para modelar ou implementar o domínio estão devidamente classificadas como *features* de Tecnologia de Domínio?		
10	As *features* que não possuem relação concreta com o domínio, mas auxiliam seu entendimento, são representadas como *features* Organizacionais?		
11	Existe alguma *feature* no modelo que, mesmo estando correta está fora do escopo do domínio?		
12	Existem *features* diferentes no modelo que representam o mesmo conceito do domínio?		
13	Existe algum conceito do domínio que foi omitido do modelo?		
14	As *features* ativadas por meio de contextos, seja de ambiente físico, computacional ou de usuário, são devidamente classificadas como *features* de Contexto?		
15	As *features* de contexto estão de acordo com as definições do domínio?		

#	Verificação de Relacionamentos entre *features*	Resposta	Observação
1	As variabilidades do domínio são adequadamente representadas como grupos alternativos (pontos de variação e suas variantes)?		
2	Os pontos de variação são claramente descritos, refletindo o significado das suas variantes?		
3	Existem duas ou mais *features* possuindo um relacionamento no modelo que não está definido no domínio?		
4	Existe algum relacionamento descrito no domínio que não está presente no modelo?		
5	É estabelecida uma hierarquia entre cada *feature* compatível com o domínio?		
6	Existe alguma *feature* no modelo que foi classificada incorretamente como uma generalização de uma outra *feature*?		
7	Existe alguma relação de dependência ou mutualidade exclusiva entre *features* de domínio no modelo que estão consistentes com a realidade desse domínio?		
8	Existe alguma relação de dependência ou mutualidade exclusiva entre *features* que não está aplicada para descrever o domínio?		
9	Existe alguma relação de dependência ou mutualidade exclusiva entre *features* que não está representada no modelo?		
10	Existe alguma *feature* no modelo que contradiz outras *features*?		
11	A *feature* raiz auxilia o entendimento do significado do domínio?		
12	De uma perspectiva geral, é possível entender o domínio a partir das *features* representadas no modelo?		
13	O modelo descreve o domínio em um nível de detalhes apropriado para o entendimento da perspectiva pretendida?		
14	O modelo tem *features* suficientes para guiar a implementação do domínio?		

#	Verificação de Regras de Composição	Resposta	Observação
1	Todas as regras de composição são descritas de forma clara e objetiva e compatíveis com as definições do domínio?		
2	Existe alguma regra de composição que contradiz outra no mesmo modelo?		
3	Existe alguma regra de composição que apesar de estar correta, não é aplicável a este domínio?		
4	Todas as regras de composição do domínio são adequadamente apresentadas no modelo?		
5	O modelo apresenta regras de composição suficientes para guiar sua implementação?		

APÊNDICE H – FORMULÁRIO DE AVALIAÇÃO DO PROCESSO PREMIER

Este formulário de *background* do experimento controlado Modelagem de DSPL para caracterização do participante. Este formulário contém algumas perguntas sobre sua experiência acadêmica e profissional.

1. **Você concorda que a utilização do processo PREMIeR auxiliou na identificação e especificação de requisitos e *features*? Justifique a resposta.**
   (a) Concordo totalmente
   (b) Concordo parcialmente
   (c) Indeciso
   (d) Discordo em parte
   (e) Discordo totalmente

   _____

   _____

   _____

2. **Você concorda que a utilização do processo PREMIeR auxiliou na modelagem de *features*? Justifique a resposta.**
   (a) Concordo totalmente
   (b) Concordo parcialmente
   (c) Indeciso
   (d) Discordo em parte
   (e) Discordo totalmente

   _____

   _____

   _____

3. **Você concorda que o processo PREMIeR ajudou na identificação e modelagem de processo de negócio de casos de uso ? Justifique a resposta.**
   (a) Concordo totalmente

(b) Concordo parcialmente

(c) Indeciso

(d) Discordo em parte

(e) Discordo totalmente

4. **Você concorda que os processos de casos de uso auxiliam o engenheiro de domínio a ter uma visão mais abrangente do domínio no que diz respeito a interação do usuário com o sistema? Justifique a resposta.**

   (a) Concordo totalmente

   (b) Concordo parcialmente

   (c) Indeciso

   (d) Discordo em parte

   (e) Discordo totalmente

5. **Você concorda que o processo PREMIeR ajudou na identificação e modelagem de contextos e suas adaptações? Justifique a resposta.**

   (a) Concordo totalmente

   (b) Concordo parcialmente

   (c) Indeciso

   (d) Discordo em parte

   (e) Discordo totalmente

6. **Você concorda que a atividade de verificação do processo PREMIeR contribuiu para a garantia da corretude e completude do modelo de *features*? Justifique a resposta.**
   (a) Concordo totalmente
   (b) Concordo parcialmente
   (c) Indeciso
   (d) Discordo em parte
   (e) Discordo totalmente

7. **Você concorda que a matriz de rastreabilidade utilizada contribuiu para a construção do modelo de *features* e análise da variabilidade? Justifique a resposta.**
   (a) Concordo totalmente
   (b) Concordo parcialmente
   (c) Indeciso
   (d) Discordo em parte
   (e) Discordo totalmente

8. **Você concorda que o processo PREMIeR contribuiu para a elicitação, modelagem e gerenciamento da variabilidade de LPSD? Justifique a resposta.**
   (a) Concordo totalmente
   (b) Concordo parcialmente
   (c) Indeciso
   (d) Discordo em parte

(e) Discordo totalmente

_____

_____

_____

9. **Na sua percepção, qual foi o esforço para o preenchimento do artefato de documento de cenários? Justifique a resposta.**

   (a) Alto

   (b) Médio

   (c) Baixo

   _____

   _____

   _____

10. **Na sua percepção, qual foi o esforço para o preenchimento do documento de especificação? Justifique a resposta.**

    (a) Alto

    (b) Médio

    (c) Baixo

    _____

    _____

    _____

11. **No documento de especificação existem seções referentes a requisitos, features, contextos e processos de negócio de casos de uso. Assinale o grau de esforço para cada um dos itens abaixo presentes no documento de especificação.**

	Excelente	**Alto**	**Bom**	**Médio**	**Baixo**
Especificação de Requisitos					
Especificação de Features					
Especificação de Contextos					
Especificação de Cenários de Adaptação de Contextos					
Especificação de Features de Contexto					
Modelagem de Processos de Negócio					

12. **Na sua percepção, qual foi o esforço para construir a matriz de rastrabilidade? Justifique a resposta.**

    (a) Alto

    (b) Médio

    (c) Baixo

    _____

    _____

    _____

13. **Na sua percepção, qual foi o esforço para realizar a verificação do modelo de features? Justifique a resposta.**

    (a) Alto

    (b) Médio

    (c) Baixo

    _____

    _____

    _____

14. **Quais são as vantagens e desvantagens do processo PREMIeR ?**

    _____

    _____

    _____

15. **Quais sugestões você teria para melhorar o processo PREMIeR ?**

    _____

    _____

    _____

ANEXO A – TEMPLATE CAPLUC

<table>
<tr><th>Elemento</th><th colspan="3">Descrição</th></tr>
<tr><td>Nome</td><td colspan="3">Nome do caso de uso.
O nome deve refletir o objetivo do caso de uso</td></tr>
<tr><td>Caso de Uso Estendido</td><td colspan="3">Nome do caso de uso cujo
comportamento é estendido
por este caso de uso</td></tr>
<tr><td>Ponto de Extensão</td><td colspan="3">Local do caso de uso estendido
onde este caso de uso atua</td></tr>
<tr><td>Categoria de reuso</td><td colspan="3">Especificar o quanto o caso de uso é obrigatório,
opcional ou alternativo
{ colocar o nome dos casos de uso alternativos aqui}</td></tr>
<tr><td>Restrições de contexto</td><td colspan="3">Descrição da condição de contexto que precisa ser
verdadeira para que este caso de uso seja passível de execução</td></tr>
<tr><td>Resumo</td><td colspan="3">Resume o objetivo do caso de uso</td></tr>
<tr><td>Atores</td><td colspan="3">Descreve os atores do caso de uso, tanto os primários
(que iniciam o caso de uso) quanto os secundários
(que podem participar do caso de uso)</td></tr>
<tr><td>Pré-condição</td><td colspan="3">Especifica uma ou mais condições que devem ser
verdadeiras no início do caso de uso</td></tr>
<tr><td>Pós-condição</td><td colspan="3">Especifica a condição que é sempre verdadeira ao final
caso de uso se a sequência principal for seguida</td></tr>
<tr><td colspan="2">Passo</td><td>Usuário</td><td>Sistema</td></tr>
<tr><td>[Nome do ponto de variação]
(Alt) [Nome da Variante]</td><td>Número do passo</td><td>Ação do usuário</td><td>Resposta do sistema</td></tr>
<tr><td colspan="4">Fluxos alternativos</td></tr>
<tr><td>Restrição para a execução
do fluxo alternativo.
No caso do produto, aqui
também podem ser colocadas
restrições de contexto para
influenciar o comportamento da aplicação.</td><td colspan="3">Descrição dos passos alternativos</td></tr>
<tr><td colspan="4">Sumário das Variações Alternativas</td></tr>
<tr><td rowspan="3">[Nome do ponto
de variação alternativo]:
Questão descritiva</td><td colspan="2">[Nome da Variante Alternativa 1]
(Restrição de Contexto: contexto
que deve ser verdadeiro para a
variante ser passível de escolha)</td><td>Identificação do passo na forma "Passo X"</td></tr>
<tr><td colspan="2">[Nome da Variante Alternativa 2]
(Restrição de Contexto: contexto
que deve ser verdadeiro para a
variante ser passível de escolha)</td><td>Identificação do passo na forma "Passo X"</td></tr>
<tr><td colspan="2">[Nome da Variante Alternativa N]
(Restrição de Contexto: contexto
que deve ser verdadeiro para a
variante ser passível de escolha)</td><td>Identificação do passo na forma "Passo X"</td></tr>
<tr><td colspan="4">Sumário das Variações Opcionais</td></tr>
<tr><td rowspan="2">[Nome do ponto
de variação opcional]:
Questão descritiva</td><td colspan="2">[COM Nome da Variante Opcional]
(Restrição de Contexto: contexto que
deve ser verdadeiro para a variante
ser passível de escolha)</td><td>Identificação do passo na forma "Passo X"</td></tr>
<tr><td colspan="2">[SEM Nome da Variante Opcional]
(Restrição de Contexto: contexto que
deve ser verdadeiro para a variante
ser passível de escolha)</td><td>Identificação do passo na forma "Passo X"</td></tr>
</table>