UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS DE QUIXADÁ
BACHARELADO EM ENGENHARIA DE SOFTWARE

LÉUSON MÁRIO PEDRO DA SILVA

ENGENHARIA DE REQUISITOS E GERENCIAMENTO DE VARIABILIDADE NA ENGENHARIA DE DOMÍNIO DE LINHAS DE PRODUTO DE SOFTWARE DINÂMICAS: UMA REVISÃO SISTEMÁTICA DA LITERATURA

QUIXADÁ

2016

LÉUSON MÁRIO PEDRO DA SILVA

ENGENHARIA DE REQUISITOS E GERENCIAMENTO DE VARIABILIDADE NA ENGENHARIA DE DOMÍNIO DE LINHAS DE PRODUTO DE SOFTWARE DINÂMICAS: UMA REVISÃO SISTEMÁTICA DA LITERATURA

Trabalho de Conclusão de Curso submetido à Coordenação do Curso Bacharelado em Engenharia de Software da Universidade Federal do Ceará como requisito parcial à obtenção do título de Bacharel.

Área de concentração: Computação

Orientador: Profª. Msc. Carla Ilane Moreira Bezerra

QUIXADÁ

2016

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca do Campus de Quixadá

S578e Silva, Léuson Mário Pedro da
Engenharia de requisitos e gerenciamento de viabilidade na engenharia de domínio de linhas de produto de software dinâmicas: uma revisão sistemática da literatura/ Léuson Mário Pedro da Silva. – 2016.
110 f. : il. color., enc.; 30 cm.

Monografia (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Bacharelado em Engenharia de Software, Quixadá, 2016.
Orientação: Profª. Me. Carla Ilane Moreira Bezerra
Área de concentração: Computação

1. Linha de Produto de Software Dinâmica (LPSD) 2. Engenharia de requisitos 3. Engenharia de domínio 4. Gerenciamento de variabilidade I. Título.

CDD 005.1

LÉUSON MÁRIO PEDRO DA SILVA

ENGENHARIA DE REQUISITOS E GERENCIAMENTO DE VARIABILIDADE NA ENGENHARIA DE DOMÍNIO DE LINHAS DE PRODUTO DE SOFTWARE DINÂMICAS: UMA REVISÃO SISTEMÁTICA DA LITERATURA

Trabalho de Conclusão de Curso submetido à Coordenação do Curso em Bacharelado em Engenharia de Software da Universidade Federal do Ceará como requisito parcial à obtenção do título de Bacharel.

Área de concentração: Computação

Aprovado em: ___/___/_____.

BANCA EXAMINADORA

_____
Profª. Msc. Carla Ilane Moreira Bezerra (Orientadora)
Universidade Federal do Ceará (UFC)

_____
Prof. Msc. Camilo Camilo Almendra
Universidade Federal do Ceará (UFC)

_____
Prof. Dr. Márcio Espíndola Freire Maia
Universidade Federal do Ceará (UFC)

_____
Profª. Drª. Tânia Saraiva de Melo Pinheiro
Universidade Federal do Ceará (UFC)

A Deus.

Aos meus pais, irmãos e amigos.

AGRADECIMENTOS

Agradeço primeiramente a Deus, por me permitir sonhar e tornar todos estes sonhos realidade.

À minha família, que apesar de todas as dificuldades e desafios, que os foram impostos, sempre se mostraram confiantes e crentes na minha vontade de vencer e seguir meus sonhos.

À Profª. Msc. Carla Ilane Moreira Bezerra, por sua dedicação e excelente orientação. O resultado positivo deste trabalho foi fruto de muito trabalho, de confiança de sua parte para comigo e principalmente de sua sinceridade em parabenizar pelos acertos e em exigir qualidade e comprometimento quando erros foram cometidos.

Aos professores participantes da banca examinadora, Prof. Camilo Almendra, Prof. Márcio Maia e Profª. Tânia Saraiva, pelo tempo, pelas valiosas colaborações e sugestões.

Aos professores do Campus da UFC Quixadá, pela excelência no ensino e comprometimento para com os alunos.

Aos meus colegas e amigos, que mesmo presentes ou distantes, motivaram-me na execução do trabalho e alegraram-se com mais esta conquista em minha vida.

“A tarefa não é tanto ver aquilo que ninguém viu, mas pensar o que ninguém ainda pensou sobre aquilo que todo mundo vê.”

Arthur Schopenhauer

RESUMO

A necessidade por desenvolver software com um alto nível de qualidade em um curto período de tempo de desenvolvimento e que permita o reuso de diferentes artefatos do produto em outros projetos têm motivado academia e indústria a identificarem abordagens que se atentem para estes fins. Neste contexto, Linhas de Produto de Software (LPSs) têm se mostrado como uma abordagem viável para promover o reuso sistemático de software. Entretanto, há uma crescente necessidade por produtos de software dinâmicos e LPSs convencionais não são capazes de atender a esta particularidade, considerando que o processo de *deployment* de um produto em LPSs é feito de maneira estática e ainda na fase de projeto. Assim surge o conceito de Linhas de Produto de Software Dinâmicas (LPSDs), como uma abordagem capaz de produzir produtos de software capazes de mudar seus respectivos comportamentos em tempo de execução devido a mudanças no ambiente de uso do produto ou para atender às novas necessidades do usuário. LPSDs possuem características exclusivas quando comparadas com LPSs tradicionais, como o gerenciamento de variabilidade e o processo de reconfiguração do produto, em que ambas características precisam ser tratadas não somente durante a fase de projeto mas também em tempo de execução do produto. Estes aspectos do gerenciamento de variabilidade, identificação e representação das características variáveis da linha, e a engenharia de requisitos, especificação do domínio que a LPSD suportará, são atividades que devem ser realizadas de acordo com as particularidades que uma LPSD possui. Baseado nesta questão, este trabalho busca apresentar as atividades e metodologias que são comumente utilizadas pela engenharia de requisitos e gerenciamento de variabilidade em LPSDs. Este processo ocorreu por meio da realização de uma revisão sistemática da literatura em que foi identificado um conjunto de trabalho de 581 trabalhos, que após sucessivas filtragens foi reduzido para 37 trabalhos finais, que foram utilizados para a extração de informações. Como resultado foi identificado um conjunto de atividades, metodologias, papéis envolvidos e ferramentas utilizadas para apoiar estas atividades da engenharia de domínio de LPSDs.

Palavras-chave: Linhas de Produto de Software Dinâmicas. Engenharia de Domínio. Engenharia de Requisitos. Gerenciamento de Variabilidade. Revisão Sistemática da Literatura.

ABSTRACT

The need for developing software with a high level of quality in a short development time and that enabling the reuse of different artifacts of a product in other projects has been motivated academy and industry to identify approaches that aims to it. In this context, Software Product Lines (SPL) have been shown a viable approach to promote the systematic reuse of software. However there is a growing need for dynamic software products and conventional SPLs are not able to attend to this particularity considering that the deployment process of an SPL product is made in a static way and still in the project phase. Thus arises the concept of Dynamic Software Product Lines (DSPL) as an approach capable to produce software products able of changing their own behavior at runtime due to changes in product use environment or to attend the new user needs. DSPLs have exclusive characteristics when compared to conventional SPLs, like the variability management and the product reconfiguration process, in which both characteristics need to be treated not only during the project phase but also at runtime of the product. These variability management aspects, identification and representation of the variable characteristics of the product line, and the requirements engineering, domain specification that the DSPL will support, are activities that must be executed according to DSPLs particularities. Based on this question, this study presents the activities and methodologies commonly used by the requirements engineering and variability management in DSPLs. This was achieved by a systematic literature review which identified a set of 581 papers that after successive filters decreased to 37 final papers, that were used to extract the information. As a result a set of activities, approaches, involved roles and tools was identified to support these activities of the DSPLs domain engineering.

Keywords: Dynamic Software Product Lines. Requirements. Domain Engineering. Requirements Engineering. Variability Management. Systematic Literature Review.

ZUSAMMENFASSUNG

Die Notwendigkeit, Software in hoher Qualität und kurzer Zeit zu entwickeln, die die Wiederverwendung verschiedener Artefakte eines Produktes in anderen Projekten erlaubt, hat Wissenschaftler und die Industrie motiviert, Ansätze in dieser Richtung zu identifizieren. In diesem Zusammenhang haben Software-Produktlinien (SPL) die Fähigkeit gezeigt, die systematische Wiederverwendung von Software zu fördern. Allerdings gibt es einen wachsenden Bedarf an dynamischen Software-Produkten, da herkömmliche SPLs nicht in der Lage sind, diese Besonderheit zu berücksichtigen, da das Deployment eines SPL Produktes in statischer Art und Weise während der Projektphase vorgenommen wird. Aus dieser Anforderung heraus entstand das Konzept der dynamischen Software-Produktlinien (DSPL) als ein Ansatz, Softwareprodukte zu entwickeln, die fähig sind, ihr Verhalten während der Laufzeit zu verändern, abhängig von Änderungen der Produktnutzung oder der Anforderungen der Nutzer. DSPLs haben besondere Eigenschaften im Vergleich zu konventionellen SPLs, wie das Variability-Management und den Produkt-Rekonfigurationsprozess. Beide Eigenschaften werden sowohl während der Projektphase als auch während der Laufzeit des Produktes berücksichtigt. Das Variability-Management, die Identifizierung und Repräsentation der variablen Eigenschaften der Linie, das Requirements-Engineering und die Domänen-Spezifikation, die die DSPL unterstützen wird, sind Aktivitäten, die gemäß den Besonderheiten von DSPLs umgesetzt werden müssen. Basierend auf dieser Fragestellung stellt diese Studie die Aktivitäten und Methoden vor, die häufig beim Requirements-Engineering und dem Variability-Management von DSPLs verwendet werden. Dieser Prozess besteht in einer systematischen Literaturrecherche, in der 581 Paper identifiziert wurde, deren Anzahl sich nach aufeinander folgenden Auswahlvorgängen auf 37 Arbeiten reduziert hat, die für die Extraktion der Informationen verwendet wurden. Als Ergebnis wurde eine Reihe von Aktivitäten, Ansätzen, Rollen und Tools identifiziert, die die Aktivitäten des DSPLs Domain-Engineering unterstützen.

Keywords: Dynamische Software-Produktelinie. Anforderungen. Domain-Engineering. Requirements-Engineering. Variability-Management. Systematische Literaturübersicht.

LISTA DE FIGURAS

LISTA DE GRÁFICOS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

DBC	Desenvolvimento Baseado em Componentes
DRS	Especificação de Requisitos do Domínio
FODA	Análise de Domínio orientada à *Feature*
LPS	Linha de Produto de Software
LPSD	Linhas de Produto de Software Dinâmicas
OR	Ou Exclusivo
PLUSS	*Product Line Use case modeling for Systems and Software engineering*
PRS	Especificação de Requisitos do Produto
RF	Requisitos Funcionais
RNF	Requisitos Não Funcionais
RUP-SE	Processo Racional Unificado para Engenharia de Sistemas
SOA	Arquitetura Orientada a Serviços
SPLE	Engenharia de Linhas de Produto de Software
UFC	Universidade Federal do Ceará
UML	*Unified Model Language*
XOR	Ou Inclusivo

LISTA DE SÍMBOLOS

%	Porcentagem
©	Copyright
®	Marca Registrada

SUMÁRIO

1 INTRODUÇÃO

A necessidade por desenvolver produtos de software providos de um alto nível de qualidade associados a um baixo custo de desenvolvimento e de tempo para a entrega do produto, têm motivado empresas e academia a desenvolverem abordagens que implementem esta ideia (SOMMERVILLE, 2011). Uma abordagem difundida na academia e nas empresas de desenvolvimento de software é a prática de reusabilidade, como um meio de reutilizar processos utilizados no desenvolvimento de software e/ou mesmo produtos de software já desenvolvidos (PRESSMAN, 2011).

Dentro da área de reuso, existe um conjunto de abordagens que buscam atender às características que o paradigma de reuso defende. Dentre elas, o conceito de Linhas de Produto de Software (LPS) têm emergido como um meio para o reuso, em que o reuso é planejado, implementado, e gerenciado (NORTHROP, 2002). Northrop et al. (2007) definem uma LPS como um grupo de produtos que compartilham um conjunto de funcionalidades comuns e gerenciadas, em que este grupo de produtos é capaz de satisfazer às necessidades específicas de um mercado específico ou missão, e são desenvolvidos de um mesmo conjunto de principais ativos em um prescrito caminho.

Pohl, Böckle e Van Der Linden (2005) verificaram que o uso de LPSs produz um aumento na qualidade dos produtos de software e uma redução no tempo de desenvolvimento de novos produtos. Estas características podem ser notadas no processo de desenvolvimento dos produtos da linha, em que as unidades necessárias para a construção dos produtos estão sempre prontas para uso, e consequentemente, estão sempre sendo integradas a novas aplicações. Isto implica que já se realizou testes nestas unidades reutilizadas, garantindo que as unidades em uso realizam suas funções de maneira correta e válida.

Hallsteinsen et al. (2008) separam os componentes (unidades reusáveis) de uma LPS em três (3) categorias: a primeira agrupa os ativos comuns a todos os produtos de uma LPS; a segunda, pelos ativos presentes em alguns produtos da LPS; e a última por ativos individuais de um produto, atendendo aos requisitos particulares do produto. Hallsteinsen et al. (2008) concluem que um princípio fundamental de uma LPS trata-se do gerenciamento de variabilidade, de como funcionalidades podem estar distribuídas entre diferentes produtos de uma mesma linha de produto.

Entretanto, LPSs convencionais suportam apenas o desenvolvimento de produtos estáticos (HYNCHEY et al., 2012). Produtos advindos de LPSs não são capazes de adaptar seus comportamentos às mudanças nas necessidades do usuário em tempo de execução

(BENCOMO; HALLSTEINSEN; SANTANA DE ALMEIDA, 2012). Tal característica contrapõe-se com o trabalho de Bosch, Capilla e Hilliard (2015), que concluem que a evolução de sistemas complexos tendem a focar com o tempo em aspectos dinâmicos e em configurações e reconfigurações realizadas após o primeiro *deployment*, junção das diferentes partes que formam um produto.

Assim, desenvolver este tipo de software que atende à variabilidade de funcionalidades em tempo de execução do produto torna-se uma atividade bastante complexa. Exemplos desta abordagem são implementados por algumas empresas e grupos de pesquisa, como é o caso de: (i) sistemas *cyber* físicos, caracterizados pela por uma nova geração de sistemas que possuem capacidades físicas e computacionais integradas podendo interagir com seres humanos de diferentes formas (BAHETI; GILL, 2011); e (ii) sistemas sensíveis ao contexto, que são caracterizados pela habilidade de adaptarem-se às circunstâncias das mudanças e responder de acordo com o contexto em uso (KJELDSKOV; SKOV, 2004).

Um exemplo de como estes sistemas funcionam pode ser visto por meio da ativação/desativação de funcionalidades impulsionadas pelo meio onde estas aplicações estão executando; e/ou mesmo pela vontade do usuário de alterar as configurações do produto final. Quando isso ocorre há a exigência de realizar uma verificação de dependência entre as funcionalidades do produto em tempo de execução, para garantir que o produto continua executando suas funções de maneira válida e correta de acordo com as especificações definidas para o produto (CAPILLA et al., 2014).

Deste modo, para desenvolver produtos de software autogerenciáveis provenientes de uma LPS, há a necessidade por uma LPS dinâmica, que produza software capaz de adaptar-se às oscilações nas necessidades do usuário e envolver restrições de recurso (HALLSTEINSEN et al., 2008). A esta variante de LPS chama-se Linha de Produto de Software Dinâmica (LPSD).

Uma LPSD, assim como uma LPS, atua em dois diferentes contextos de uso: a (i) engenharia de domínio, responsável por definir e desenvolver os artefatos comuns e variáveis da linha de produto e (ii) engenharia de aplicação, responsável por selecionar e reusar dentre os artefatos comuns e variáveis da linha de produto, aqueles artefatos que atendem às necessidades específicas do produto (NORTHROP et al., 2007).

Considerando que uma linha de produto deve atender às particularidades de um específico domínio, é necessário capturar de maneira clara e precisa as necessidades do domínio. Pohl, Böckle e Van Der Linden (2005) afirmam que a Engenharia de Requisitos pode ser aplicada tanto para desenvolver produtos de software individuais, quanto para

desenvolver linhas de produto. No ciclo de vida da engenharia de domínio de LPSDs, a engenharia de requisitos é realizada por meio de duas atividades distintas: análise de domínio e de contexto (CAPILLA et al., 2014). A análise de domínio define e especifica as necessidades e particularidades do domínio que a LPSD irá atender, enquanto a análise de contexto captura os contextos que devem ser suportados pela LPSD, atentando-se também às informações referentes a reconhecimento de mudança de contexto e reconfigurações do produto (CAPILLA et al., 2014).

Processos tradicionais de LPSs apresentam a análise de domínio como atividade constituinte do ciclo de vida da engenharia de domínio. Entretanto, estes processos não podem ser aplicados para o desenvolvimento de LPSDs, uma vez que não apresentam atividades de análise de contexto, conjunto de atividades responsáveis pela especificação da dinamicidade de LPSDs. Processos de LPSs tratam o gerenciamento de variabilidade da linha de produtos de maneira estática, que diferentemente dos processos de LPSDS, que necessitam planejar como reconfigurações nos produtos de LPSDs serão feitos por meio das informações advindas das mudanças de contexto.

Fundamentado nestas características, dinamicidade e gerenciamento de variabilidade em LPSDs, este trabalho têm por objetivo geral identificar atividades, papéis, abordagens e ferramentas para apoiar a engenharia de requisitos e gerenciamento de variabilidade da engenharia de domínio de Linhas de Produto de Software Dinâmicas (LPSDs), por meio da realização de uma revisão sistemática. Com a execução desta revisão sistemática espera-se alcançar dois objetivos específicos, que são: (i) identificar e avaliar estudos que executam as atividades da engenharia de requisitos e gerenciamento da variabilidade da engenharia de domínio em LPSDs; e, (ii) identificar e avaliar as atividades que compõem as atividades da engenharia de requisitos e gerenciamento de variabilidade tanto para a Análise de Domínio quanto para a Análise de Contexto. Como público-alvo deste trabalho identifica-se engenheiros de software, pesquisadores acadêmicos e quaisquer profissionais que trabalham diretamente com a engenharia de requisitos e o gerenciamento da variabilidade da engenharia de domínio de LPSDs.

Como resultado da execução revisão sistemática foi identificado atividades, papéis, abordagens e ferramentas utilizadas para a especificação e modelagem do domínio, como também áreas da engenharia de requisitos e do gerenciamento de variabilidade, que apresentam deficiências quando realizadas para a engenharia de domínio de LPSDs, e áreas de pesquisa ainda pouco exploradas.

2 TRABALHOS RELACIONADOS

Para o embasamento deste trabalho foi realizada uma busca bibliográfica com o intuito de identificar estudos que estivessem relacionados à área de atuação deste trabalho. A seguir, estes trabalhos são apresentados, e também são identificados pontos que ajudaram na execução deste trabalho.

Alves et al. (2010) apresentam uma revisão sistemática para a engenharia de requisitos de linhas de produto de software tradicionais (LPSs). Semelhante ao objetivo deste trabalho, os autores buscam identificar as melhores práticas utilizadas, que facilitam e/ou apoiam as atividades da engenharia de requisitos para LPSs. A revisão sistemática foi feita a partir do estabelecimento de três (3) questões de pesquisa, seguido da identificação e análise dos trabalhos identificados. Entretanto aspectos exclusivos de Linhas de Produto de Software Dinâmicas (LPSDs), como gerenciamento de variabilidade em tempo de execução e reconfiguração dinâmica, não são tratados pelos autores.

Guedes et al. (2015) apresentam um estudo direcionado à área de LPSDs cujo objetivo assemelha-se com a proposta apresentada neste trabalho. Guedes et al. (2015) buscam apresentar quais as abordagens utilizadas para realizar o gerenciamento de variabilidade em LPSDs. O processo para a identificação das abordagens foi feito seguindo uma metodologia tradicional de um mapeamento sistemático, através da definição de questões de pesquisa e critérios para avaliação dos trabalhos. Entretanto os resultados encontrados compreendem apenas as abordagens que são utilizadas para realizar o gerenciamento de variabilidade; não há orientações de como tais abordagens devem ser utilizadas nem como tais abordagens relacionam-se com a engenharia de requisitos em LPSDs.

Capilla et al. (2014) apresentam um *framework* que reúne um conjunto de atividades a serem realizadas no contexto de LPSDs. Semelhante à proposta deste trabalho, os autores abordam o modo pelo qual as particularidades de uma LPSD devem ser tratadas já durante a fase de requisitos. No estudo são apresentadas as soluções para os desafios que os autores identificaram inicialmente, gerenciamento de variabilidade e processo de reconfiguração dinâmica, através da divisão da análise de domínio tradicional em duas novas atividades: análise de domínio e análise de contexto. Entretanto o estudo apenas apresenta essas soluções no contexto dos desafios identificados, não há orientações de como estas atividades devem ser realizadas, quais técnicas/ferramentas a serem utilizadas, os envolvidos em cada atividade e a definição e participação de *stakeholders*.

3 FUNDAMENTAÇÃO TEÓRICA

Nesta seção serão apresentados os conceitos abordados para o desenvolvimento deste trabalho. Nas primeiras Seções (4.1 e 4.2) são introduzidos os conceitos referentes à Linhas de Produto de Software Dinâmicas (LPSDs) e Gerenciamento de Variabilidade, o que compreende seus respectivos componentes, seu escopo de atuação e suas particularidades quando comparadas com Linhas de Produto de Software tradicionais (LPSs). Na Seção 4.3, os conceitos referentes à Engenharia de Domínio são abordados; na Subseção 4.3.1 é apresentado os conceitos da Análise de Domínio, abordando as atividades de Engenharia de Requisitos no contexto de linhas de produto, e na Subseção 4.3.2 é abordada a Análise de Contexto, apresentando sua importância para LPSDs e como ela diferencia-se da análise de domínio. Na última Seção (4.4) é apresentado um conjunto de trabalhos que apresentam suas respectivas abordagens para o tratamento das atividades da Engenharia de Domínio em linhas de produto.

3.1 Linha de Produto Software Dinâmica

Linha de Produto de Software (LPS) trata-se de uma abordagem para promover o Reuso de Software através da reutilização de artefatos entre todos os produtos da linha, para atender as necessidades de um determinado domínio. Estes produtos possuem similaridades e variabilidades entre si, resultado das necessidades e características que cada produto deve adotar.

Em alguns domínios emergentes, robótica, software de assistência médica, sistemas de software estão se tornando cada vez mais complexos (HALLSTEINSEN et al., 2008). Esses domínios necessitam de sistemas capazes de alterarem suas próprias configurações em tempo de execução, adaptando-se às mudanças ocorrentes nas necessidades do usuário. Entretanto, LPSs tradicionais não foram planejadas para alterar o comportamento de seus produtos de modo dinâmico, pois a adaptação de seus produtos é concebida durante o tempo de projeto dos produtos (BENCOMO; HALLSTEINSEN; SANTANA DE ALMEIDA, 2012).

Para desenvolver estes tipos de sistemas dinâmicos foram adotadas algumas abordagens no processo de desenvolvimento, com o intuito de atender a estas características de adaptabilidade em tempo de execução, mas sem perder os benefícios que uma LPS oferece (HALLSTEINSEN et al., 2008). Assim, surgiu o conceito de Linhas de Produto de Software Dinâmicas (LPSD).

Bencomo, Hallsteinsen e Santana de Almeida (2012) definem LPSD como uma abordagem que estende o conceito convencional de LPS por permitir geração de variantes de software em tempo de execução. Trata-se de uma abordagem que permite desenvolver produtos de software capazes de adaptarem-se às mudanças nas necessidades dos usuários, mudanças de requisitos e restrições impostas pelo ambiente em que estes produtos funcionam. Trata-se de um único sistema que é capaz de adaptar seu próprio comportamento em tempo de execução (PASCUAL; PINTO; FUENTES, 2015).

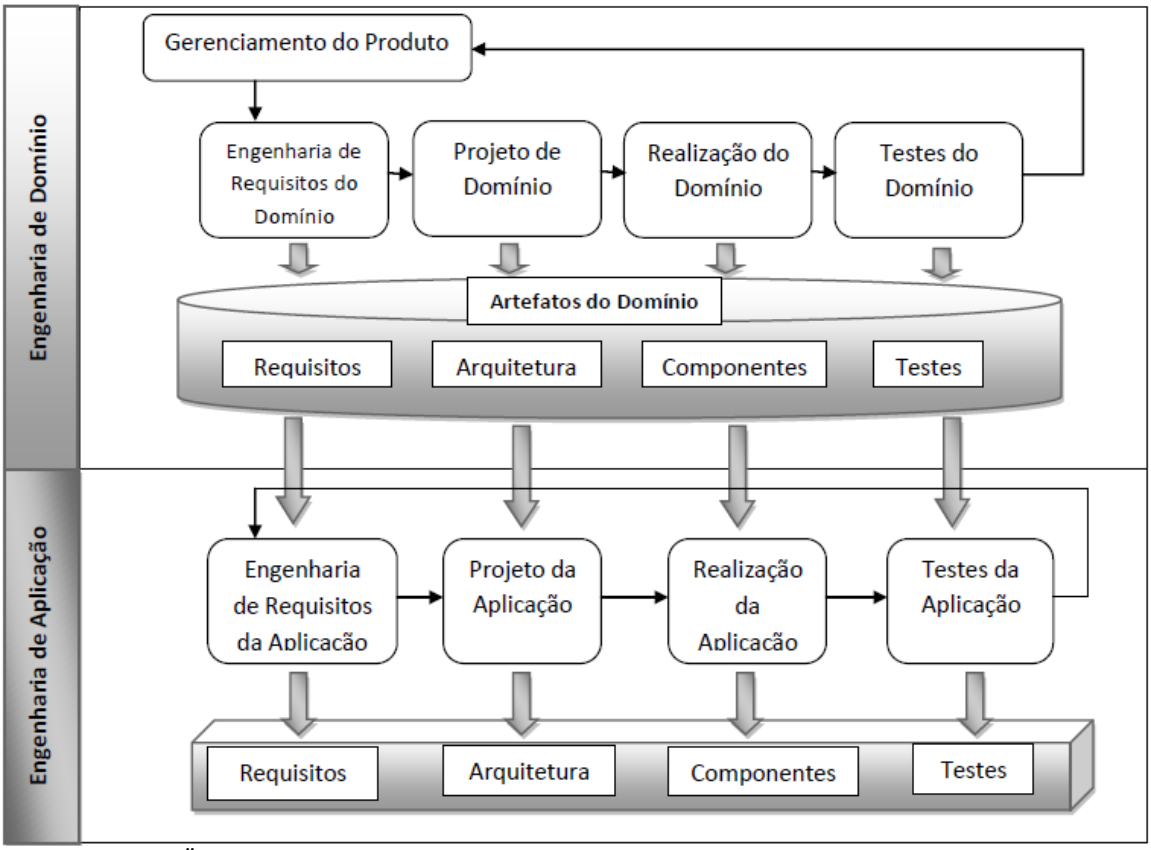
Pohl, Böckle e Van der Linden (2005) apresentam um paradigma para a Engenharia de Linhas de Produto de Software (SPLE, do inglês *Software Product Line Engineering*). Este paradigma divide a SPLE em dois processos separados: Engenharia de Domínio e Engenharia de Aplicação (**Figura 1**).

Czarnecki e Eisenecker (2000) definem a Engenharia de Domínio como o processo responsável por coletar, organizar e armazenar experiências passadas na construção de sistemas, ou partes do sistema, em um particular domínio na forma de ativos reusáveis, artefatos de desenvolvimento, provendo uma adequada estrutura para reusar estes ativos na construção de novos produtos. É neste processo em que a Especificação dos Requisitos do Domínio é feita (DRS, do inglês *Domain Requirements Specification*). Este processo é constituído por cinco (5) subprocessos: Gerenciamento do Produto, Engenharia de Requisitos do Domínio, Projeto do Domínio, Realização do Domínio e Teste do Domínio.

Dentro da representação da Engenharia de Domínio (**Figura 1**) é possível verificar os artefatos do domínio: requisitos, arquitetura, componentes e testes. Estes artefatos são entendidos como ativos (do inglês, *assets)* e compreendem a um conjunto de artefatos que podem ser usados para a construção de futuros produtos da Linha (NORTHROP et al., 2007) (POHL; BÖCKLE; VAN DER LINDEN, 2005). Estes ativos representam os produtos de trabalho da atividade de Engenharia de Aplicação.

Pohl, Böckle e Van der Linden (2005) atribuem à Engenharia de Aplicação a responsabilidade de construir as aplicações da linha de produto, por meio do reuso dos artefatos do domínio, e por explorar a variabilidade da linha. É neste processo em que a Especificação dos Requisitos do Produto são identificados (PRS, do inglês *Product Requirements Specification*). Este processo é dividido em quatro (4) subprocessos: Engenharia de Requisitos da Aplicação, Projeto da Aplicação, Realização da Aplicação e Testes da Aplicação. Como saída deste processo, têm-se os Artefatos de Aplicação, que representam o desenvolvimento de artefatos de específicas aplicações da linha.

Figura 1 - Framework para Engenharia de Linhas de Produto de Software



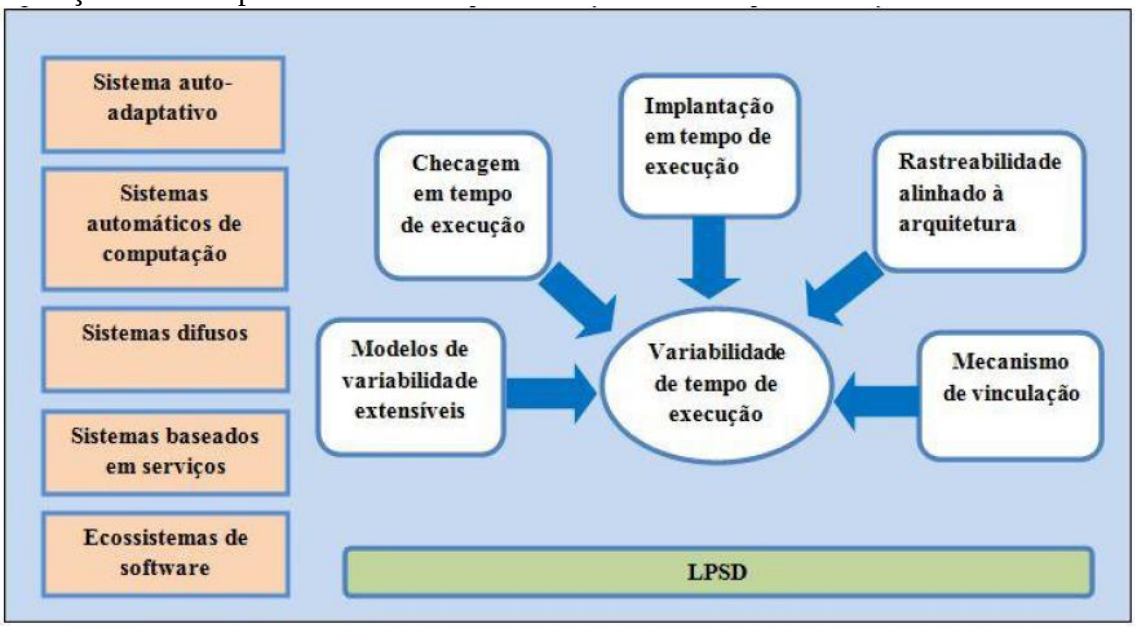
Fonte: POHL; BÖCKLE; VAN DER LINDEN; 2005. (com adaptações)

Capilla e Borsch (2011) apresentam, na Figura 2, um conjunto de diferentes tipos de sistemas que são caracterizados como sistemas dinâmicos, por alterarem seu próprio comportamento em tempo de execução. São eles: sistemas autoadaptativos, sistemas automáticos de computação, sistemas difusos, sistemas baseados em serviços e ecossistema de software. Os próximos parágrafos desta seção apresentam brevemente as características principais que caracterizam estes tipos de sistemas.

Salehie e Tahvildari (2009) definem que autoadaptabilidade representa a capacidade geral que um sistema de software possui de ser embutido para monitorar sua execução e reagir, adaptar seu comportamento, concordantemente. Um sistema de software autoadaptativo é um sistema que é capaz de mudar seu próprio comportamento e estrutura em resposta à sua percepção de ambiente, mudanças de contextos e mudanças de seus requisitos (DE LEMOS et al., 2013) (MARCÍA-ESCRIVÁ et al., 2013). Sistemas autoadaptativos possuem as mesmas características que sistemas automáticos, comentadas no próximo parágrafo; através destes aspectos é que estes tipos de sistemas tornam-se capazes de responder às mudanças em tempo de execução (HUEBSCHER; MCCANN, 2008). Estas

características acabam sendo redundantes em outros tipos de sistemas, como será nas próximas descrições de tipos de sistemas.

Figura 2 - Mecanismos de variabilidade em tempo de execução necessários para a transição de LPSs para LPSDs



Fonte: CAPILLA; BORSCH, 2011 (com adaptações).

Horn (2001) introduziu o conceito de computação automática a partir dos conceitos da biologia. Analogamente ao sistema nervoso, que trabalha continuamente para manter seu estado de equilíbrio psicológico, a tecnologia da informação precisa de computação automática para ser dinamicamente adaptada às mudanças de ambiente (ALAYA et al., 2012). Sistemas autonômicos são caracterizados por possuírem as seguintes características (NHANE; SONG, 2000):

- autoconfiguração, capacidade de um sistema de autoconfigurar-se quando mudanças ocorrem no ambiente de uso;
- autorresolução de problemas, capacidade que o sistema tem de identificar, diagnosticar e reagir às alterações que causam mau funcionamento do sistema;
- autoproteção, capacidade de antecipar, detectar, identificar e proteger a si mesmo de ataques de quaisquer fontes; e
- auto-otimização, relacionado à monitoração e ajustes de seus próprios recursos.

Sistemas pervasivos, ou difusos, são os responsáveis pelo uso da computação em todo lugar e a todo momento; este paradigma introduziu o conceito de utilização da computação em todos os lugares e a todo tempo (dispositivos móveis, *smartphones*, *tablets*) e

que são constantemente usados nas rotinas diárias de seres humanos e tornam-se invisíveis aos sentidos humanos (JAROUCHEH; LIU; SMITH, 2010b).

Sistemas baseados em serviços são construídos através de composições dinâmicas de serviços acoplados oferecidos por independentes provedores e encontram-se operando em ambientes caracterizados por contínuas mudanças de requisitos, estados dos serviços componentes e sistemas de uso de perfis (CALINESCU et al., 2011).

Para Mens et al. (2014) ecossistemas de software consistem de um relativo núcleo de sistema de software que provê a funcionalidade básica e que é desenvolvido por um time núcleo de desenvolvedores cercado por uma larga coleção de contribuições providas por outros desenvolvedores ou mesmo usuários finais; representam um conjunto de construções técnicas que dispõem de fronteiras bem definidas, tais como sistemas distribuídos (BARBOSA; ALVES, 2011). Hanssen (2012) acredita que todo ecossistema de software é fundamentado perante dois (2) conceitos principais: (i) uma rede de organizações e (ii) um comum interesse na tecnologia central do software. Para Bosch (2009) ecossistemas de software são relacionados à LPSs devido à implicação de reuso de software; nesse sentido, há alguns trabalhos que utilizam de tais características de ecossistemas aliado aos conceitos de LPSs. Como por exemplo, o trabalho de Urli et al. (2014) em que apresentam um estudo de caso de como organizar um ecossistema de software utilizando os conceitos de LPSs.

Ainda na Figura 2, é possível identificar cinco (5) mecanismos de variabilidade característicos de uma LPSD; estes mecanismos serão abordados detalhadamente na próxima seção de Gerenciamento de Variabilidade (Seção 4.2).

LPSDs diferencia-se de LPSs tradicionais no modo como cada uma trata o gerenciamento de variabilidade. Esta característica representa apenas uma das diferenças/particularidades que uma LPSD possui. Capilla et al. (2014) apresentam um trabalho em que exploram as particularidades que uma LPSD possui, e os desafios a que o desenvolvimento de LPSDs têm de atentar-se. Hinchey, Sooyong e Schmid (2012) acreditam que para uma melhor diferenciação entre uma LPS e uma LPSD, é necessário apontar como uma LPSD diferencia-se do conceito de linha de produto. A seção 3.2 aborda o papel e como é executado o Gerenciamento de Variabilidade em LPSDs.

3.2 Gerenciamento de Variabilidade

Para entender como o Gerenciamento de Variabilidade deve ser executado em Linhas de Produto de Software Dinâmicas (LPSD) é necessário esclarecer alguns conceitos

referentes à Variabilidade. Van Gurp, Bosch e Svahnberg (2001) definem variabilidade como a possibilidade de configuração, ou mesmo, a habilidade que um sistema ou artefato de software possui de ser alterado, customizado, ou configurado para um contexto em particular.

Características de variabilidade são definidas a partir de meios chamados Pontos de Variação. Pontos de Variação são satisfeitos a partir da vinculação de uma ou mais variantes a um ponto de variação (VON DER MAßEN; LICHTER, 2002). Uma variante é entendida como uma alternativa de configuração para um produto, uma maneira de implementar uma determinada característica no produto (HEYMANS; TRIGAUX, 2003). Deste modo, se faz necessário controlar como a distribuição de variabilidade será feita ao longo do ciclo de vida de uma linha de produto; é neste ponto de atuação que o Gerenciamento de Variabilidade atenta-se.

Alguns trabalhos buscam explorar essas características, sendo que para alguns autores estas características são consideradas desafios a que LPSDs precisam satisfazer (CAPILLA et al., 2014) (BENCOMO; HALLSTEINSEN; SANTANA DE ALMEIDA, 2012) (CAPILLA; BOSCH, 2011).

Capilla e Borsch (2011) apresentam quais mecanismos de variabilidade em tempo de execução são necessários para que uma LPS tradicional torne-se uma LPSD; estes mecanismos serão abordados de maneira detalhada a seguir, e encontram-se representados na Figura 2, apresentada anteriormente.

Hinchey, Sooyong e Schmid (2012) observaram que o *binding time* de Linhas de Produto de Software (LPS), mecanismo de vinculação necessário para conectar as diferentes partes do produto, ocorre normalmente antes da execução de fato do produto ainda na fase de projeto. Logo, o gerenciamento de variabilidade em uma LPS ocorre de maneira estática, através do mapeamento dos requisitos do novo produto com os ativos disponíveis, seleção dos ativos que satisfazem aos requisitos, e por fim, a composição do produto através da junção destes ativos (POHL; BÖCKLE; VAN DER LINDEN; 2005).

Bencomo, Hallsteinsen e Santana de Almeida (2012) afirmam que produtos advindos de uma LPSD podem ser reconfigurados em tempo de execução, após a derivação inicial do produto; o gerenciamento ocorre em tempo de execução. Hinchey, Sooyong e Schmid (2012) definem que o gerenciamento de variabilidade em LPSD corresponde às diferentes adaptações que o produto de uma Linha pode ter durante o seu funcionamento; como o produto de uma LPSD pode adaptar-se às mudanças de comportamento ocorrentes em seu ambiente de execução, resultando na ativação/desativação de funcionalidades do produto.

Para Capilla e Bosch (2011), a capacidade de representar a variabilidade, modificando pontos de variação em existentes e novas unidades de software durante a execução e reconfiguração automática do sistema representa um desafio para LPSDs. Capilla et al. (2014) consideram este aspecto como uma das mais simples formas de gerenciamento de variações em tempo de execução, em que o usuário ou o sistema, de maneira automatizada, pode ativar/desativar *features* do produto dinamicamente. Estas características de ativação/desativação de funcionalidades devem ser documentadas já durante o processo de Requisitos, garantindo que se têm conhecimento sobre quais requisitos podem sofrer mudanças em tempo de execução do produto.

Capilla e Bosch (2011) também se atentam para a capacidade que LPSDs devem ter de automatizar a variabilidade em tempo de execução e checar os modelos de *features* reconfigurados para manter a consistência e estabilidade do sistema. Capilla et al. (2014) afirmam que quando uma dada mudança ocorre no produto, adição/desativação de *features*, é necessário garantir que o sistema encontra-se consistente, e que a mudança introduzida não introduziu dependências e/ou restrições no produto, que possam desencadear a ocorrência de erros na nova configuração. Estas dependências devem ser documentadas durante o processo de requisitos através de dependências entre as *features* associadas aos requisitos do produto.

Laurenroth e Pohl (2008) acreditam que esta verificação de inconsistências deve acontecer durante a engenharia de domínio devido à necessidade de identificar inconsistências na Especificação de Requisitos do Domínio (DRS, do inglês *Domain Requirements Specification*), antes do desenvolvimento dos artefatos reusáveis. Laurenroth e Pohl (2008) definem um *framework* para a verificação de inconsistências nos requisitos do domínio; este *framework* busca relacionar os requisitos com *features* variáveis e posteriormente, identificar e eliminar as inconsistências identificadas.

Capilla e Bosch (2011) também apresentam como desafio o processo de automação do *deploy* do produto e reconexão dos produtos reconfigurados em tempo de execução com o mínimo de interrupções, implantação em tempo de execução. Capilla et al. (2014) acreditam que a reconfiguração em tempo de execução do produto deve ser entendida como a capacidade do sistema de garantir a reconfiguração quando uma dada mudança ocorre.

Pascual, Pinto e Fuentes (2015) verificaram em seu trabalho que o processo de *deploy* de um produto pode ser feito de duas maneiras principais: em tempo de projeto (i) e em tempo de execução (ii). Em (i) a montagem de todas as partes necessárias para todas as possíveis adaptações do produto ocorrem durante a montagem do produto para a sua entrega.

Já em (ii) a conexão das partes necessárias para as possíveis adaptações do produto ocorrem a medida que surgem as necessidades de adicionar variabilidade ao produto (MORIN et al., 2009) (ORTIZ et al., 2012).

Bencomo, Hallsteinsen e Santana de Almeida (2012) também atentam-se a este processo de reconfiguração dos produtos e apresentam como pesquisadores têm tratado esta característica através da combinação de ideias de linhas de produto com mecanismos de variabilidade em tempo de execução; como o Desenvolvimento Baseado em Componentes (DBC) e Arquiteturas Orientadas a Serviços (SOA) (ISTOAN et al., 2009). Bencomo, Hallsteinsen e Santana de Almeida (2012) afirmam que as abordagens de desenvolvimento citadas acima suportam o processo de reconfiguração de seus produtos.

Segundo Alves et al. (2010) o Modelo de *Features* representa um dos meios mais utilizados para tratar da variabilidade em linhas de produto. Lisboa et al. (2014) constataram que em domínios simples, onde há poucas restrições entre variabilidade, o modelo de *features* ajuda o engenheiro de domínio a ter uma visão sobre o domínio, obtendo assim um melhor entendimento e manutenibilidade do modelo. Na Subseção 3.2.1 este modelo é apresentado, com base em sua organização e nas características utilizadas para representação da variabilidade.

3.2.1 *Modelo de Features*

O Modelo de *Features* foi apresentado pela primeira vez em 1990 como uma modelagem para a representação de *features*, proposta como parte da Análise de Domínio Orientada à *Feature* (FODA) (KANG et al., 1990). Kang et al. (1990) definem uma *feature* como uma característica do sistema visível ao usuário final. Entretanto, segundo os estudos de Berger et al. (2015), varia muito a noção do que uma *feature* realmente é; os autores ainda concluem que uma *feature* precisa representar um distinto e bem entendido aspecto presente no sistema.

Capilla e Bosch (2011) definem a modelagem de *features* como uma abordagem comum para projetar produtos que compartilham de um mesmo conjunto de características ou estão no mesmo segmento de mercado.

Com o decorrer do tempo, alguns trabalhos foram desenvolvidos e novas características foram adicionadas ao Modelo de *Features*, como a inclusão do conceito de cardinalidade no modelo (CZARNECKI et al., 2002). No Quadro **1**, é apresentado o conjunto de tipos válidos que uma *feature* pode assumir, e adicionalmente é mostrada a cardinalidade para cada respectivo tipo.

Quadro 1 - Tipos de *Features*

Tipo	Descrição	Cardinalidade
Obrigatória	Representa o conjunto de *features* que devem estar presentes em todos os produtos da Linha.	<1,1>
Opcional	Representa o conjunto de *features* que podem ou não fazer parte da configuração de um produto.	<0, N>
Ou Exclusivo (XOR)	Representa uma única *feature* que fará parte da configuração de um produto, advinda de um conjunto de *features.*	<1,1>
Ou Inclusivo (OR)	Representa um subconjunto de *features* que farão parte da configuração de um produto, advindas de um conjunto de *features.*	<1,N>
Alternativa	Representa o conjunto de *features* que podem fazer parte da configuração de um produto, dependendo diretamente de um número mínimo e máximo de funcionalidades.	<M, N>

Fonte: KANG et al., 1990. CZARNECKI et al., 2002. (com adaptações)

O Modelo de *Features* suporta a definição de restrições entre *features*: restrição de inclusão e restrição de exclusão (KANG et al., 1990). As restrições de inclusão definem a obrigatoriedade de inclusão de uma ou de um conjunto de *features*, quando uma determinada *feature* for selecionada para fazer parte da configuração de um produto. Restrições de exclusão compreendem a exclusão de uma *feature* ou o conjunto delas, quando uma determinada *feature* for incluída na configuração do produto.

Restrições representam uma importante fonte de informações para o tratamento das particularidades de LPSD. Por exemplo, restrições de inclusão e de exclusão fornecem as informações necessárias para o processo de ativação/desativação de funcionalidades. Deste modo, quando uma dada *feature* for incluída ou excluída da configuração de um produto, restrições de inclusão e exclusão definem o conjunto de *features* a ser adicionado ou excluído da futura nova configuração do produto (POHL; BÖCKLE; VAN DER LINDEN, 2005).

Outra importante característica de restrições é que elas representam uma importante fonte de informação para o processo de reconfiguração do produto em tempo de execução. Por exemplo, uma vez que uma *feature* X exclui um conjunto de *features* Y (y1, y2 e y3) da configuração atual do produto, é necessário verificar se há alguma *feature* da futura nova configuração do produto que depende de uma ou mais *features* do conjunto Y, que deve ser excluído. Em se tratando de uma restrição de inclusão, em que uma *feature* X inclui uma *feature* Y na futura nova configuração do produto, é necessário verificar se as *features* dependentes da *feature* Y também encontram-se na futura nova configuração do produto (CAPILLA et al., 2014).

3.3 Engenharia de Domínio

Unidade constituinte de uma Linha de produto de Software Dinâmica (LPSD), a Engenharia de Domínio compreende o conjunto de atividades utilizadas para a definição de requisitos e exploração de potenciais arquiteturas de software, que podem ser aplicadas por todas as fases de desenvolvimento de software e em diferentes níveis de abstração para desenvolver sistemas, componentes, e ambientes (WHITE; EDWARDS, 1996). Bjorner (2007) acredita que a proposta da engenharia de domínio é construir, desenvolver e pesquisar por descrições do domínio, devendo buscar uma visão clara e precisa sobre as necessidades do domínio em foco.

Bencomo, Hallsteinsen e Santana de Almeida (2012) definem a engenharia de domínio no contexto de linha de produto, como a atividade responsável por produzir a infraestrutura da linha de produto. Esta infraestrutura é formada por: (i) uma arquitetura comum e seus pontos de variação; (ii) um conjunto de partes reusáveis que são adicionados à arquitetura; e (iii) um modelo de representação da variabilidade, que identifica as variabilidades e similaridades nos requisitos no domínio, e como os requisitos de um produto específico são mapeados para as *features* presentes no modelo de variabilidade.

Bosch (2001) explica que, apesar de a Engenharia de Domínio não ter contato direto com os clientes dos futuros produtos da linha, ela precisa ter um bom entendimento sobre as necessidades destes futuros clientes e dos requisitos que estes produtos devem possuir. Deste modo, o conjunto de requisitos de todos os produtos da linha, identificados pela engenharia de aplicação, devem corresponder aos requisitos do domínio, identificados na engenharia de domínio. Hallsteinsen et al. (2008) apresentam um modelo de ciclo de vida de uma Linha de produto de Software (LPS), em que unem a engenharia de domínio e a engenharia de aplicação através de específicos artefatos reusáveis da linha de produto. Esta abordagem relaciona-se diretamente com o *framework* proposto por (NORTHROP et al., 2007).

Harsu (2002) agrupa as atividades da Engenharia de Domínio em um conjunto de três (3) atividades: Análise, Projeto e Implementação do Domínio; Hallsteinsen et al. (2008) acrescentam a atividade de Testes a este conjunto. A etapa de análise é a responsável por determinar o domínio que a linha irá atender, e posteriormente, identificar as necessidades do domínio; a Análise de Domínio será tratada de maneira detalhada na Subseção 3.3.1.

Harsu (2002) define a atividade de Projeto como o processo de transformação do modelo de domínio para um projeto de arquitetura adaptável. É realizado o planejamento da

arquitetura de referência base da linha de produto a ser utilizada, considerando que esta arquitetura estará presente em todos os futuros produtos da Linha.

Para entender o papel que a arquitetura desempenha em LPSDs, é necessário entender o papel que uma arquitetura possui em sistemas convencionais. McGovern et al., 2004 apresentam dois papéis principais que uma arquitetura possui: definição dos componentes do sistema, suas interfaces e interações; e restrição em aspectos ligados a projeto e implementação.

No que diz respeito a LPSs, a arquitetura é responsável por prover um *framework* comum para um conjunto de arquiteturas de produtos individuais (HINCHEY; SOOYONG; SCHMID, 2012). Para a construção de um novo produto de uma linha é reusado uma mesma arquitetura de referência, e a partir desta arquitetura são realizadas alterações para atender aos requisitos particulares de cada produto.

Entretanto, quando se trata de LPSDs algumas particularidades precisam ser consideradas. Hinchey, Sooyong e Schmid (2012) afirmam que em LPSDs há uma única arquitetura de sistema, a qual provê a base para todas as possíveis adaptações do produto. A arquitetura da Linha deve suportar as possíveis mudanças (reconfigurações e/ou vinculações dinâmicas do produto) que podem ocorrer em tempo de execução.

Como terceira atividade da Engenharia de Domínio encontra-se a atividade de implementação, responsável pelo desenvolvimento dos ativos reusáveis, componentes e infraestrutura de reuso (NORTHROP et al., 2007). Por fim, a atividade de Testes propõe-se a validar os ativos desenvolvidos; ressaltando que testes adicionais nestes ativos são feitos quando eles são usados para fazer parte da configuração de algum produto da Linha.

As próximas Subseções (3.2.1 e 3.2.2) apresentam as duas (2) atividades principais responsáveis pela especificação do domínio.

3.3.1 *Análise de Domínio*

A análise de domínio é utilizada para aumentar o grau de exigências referentes a aplicação na fase de elicitação, a fim de identificar e antecipar mudanças, possibilitando a determinação de similaridades e variações nos produtos da linha de produto, dando apoio à criação de arquiteturas robustas (NORTHROP et al., 2007). É a atividade responsável por atender aos seguintes objetivos: (i) definição do domínio a ser trabalhado, (ii) identificação e análise dos requisitos; e (iii) identificação de funcionalidades variáveis do domínio (HARSU, 2002). Estes objetivos relacionam-se diretamente com as atividades da Engenharia de

Requisitos. Nesta Subseção será apresentado como estes objetivos são alcançados por meio de algumas atividades da Engenharia de Requisitos.

Standish Group International (2013) constatou que a atividade de elicitação, seleção e implementação dos requisitos representam as atividades mais difíceis no desenvolvimento de aplicações. Assim, torna-se clara a necessidade de, não somente executar este conjunto de atividades em projetos de LPSDs, como também garantir que sua execução seja realizada de maneira precisa e concisa.

Em LPSDs, a primeira atividade da Engenharia de Requisitos a ser feita é a atividade de concepção. Ela é responsável por identificar o domínio pelo qual a LPSD atuará (NORTHROP et al., 2007); com o domínio definido, as demais atividades tornam-se mais fáceis de serem realizadas. Mellado, Fernández-Medina e Piattini (2008) realizam a identificação dos *stakeholders* em seu subprocesso de gerenciamento de escopo; entretanto neste mesmo subprocesso já é realizado atividades de identificação de *features*, atividade que é geralmente realizada durante a Elicitação de Requisitos.

Achterkamp e Vos (2008) avaliam a importância dos *stakeholders* na execução de um projeto, constatando que os fatores que causam a falha de projetos não estão relacionados ao resultado ou a ineficiência das práticas do projeto de gerenciamento, mas das inapropriadas interações entre *stakeholders*. Esta comunicação pode ser mais complexa quando se trata de LPSD, considerando que uma LPSD busca atender às necessidades de um domínio específico, e este domínio pode compreender um domínio de negócio local ou mesmo um domínio que compreende um país, ou mesmo um continente.

Damian (2007) acredita que, em projetos distribuídos, grupos de *stakeholders* precisam especificar e definir requisitos com base em aspectos culturais, fusos-horários, e fronteiras organizacionais; é necessária a utilização de práticas de identificação e comunicação com *stakeholders* que atendam a estas particularidades. Neiva et al. (2010) indicam como exemplos de *stakeholders* para LPSs, especialistas do domínio, gerentes, analistas e usuários.

Alves et al. (2010) verificaram que as atividades de Planejamento e Elicitação, e Modelagem e Análise correspondem às atividades mais utilizadas na Engenharia de Linhas de Produto de Software (SPLE, do inglês *Software Product Line Engineering*). A ênfase na atividade de Modelagem e Análise ocorre pela necessidade de tratamento da variabilidade, característica núcleo da SPLE.

Sobre a Elicitação de Requisitos, Pohl (2010) a define como a atividade núcleo da Engenharia de Requisitos para determinar as fontes relevantes de requisitos, e identificar e

desenvolver requisitos a partir destas fontes. É responsável por identificar e entender quais são as necessidades do domínio, requisitos do produto, e a partir destas necessidades estabelecer uma solução para o problema. Pohl, Böckle e Van Der Linden (2005) acreditam que é comum o uso de um conjunto de sistemas no domínio que a linha de produto irá atender; estes sistemas podem ser utilizados como fontes de características variáveis e comuns nos futuros produtos da linha. Estas informações são agrupadas em uma Matriz de Requisitos de Aplicação; o Quadro **2** apresenta um exemplo desta matriz.

Quadro 2 - Estrutura de uma Matriz de Requisitos de Aplicação

Requisitos da Aplicação	**Aplicação 01**	**Aplicação 02**	**Aplicação 03**	**Aplicação 04**
R1	obrigatório	obrigatório	Obrigatório	obrigatório
R2	-	-	Obrigatório	obrigatório
R3	-	obrigatório	-	-
...	...	...	...	...

Fonte: POHL; BÖCKLE- VAN DER LINDEN, 2005. (com adaptações)

Neiva et al. (2010) acreditam que a técnica a ser adotada para realizar a Elicitação deve estar de acordo com o tipo das fontes de requisitos, e que a identificação destas fontes são essenciais para o sucesso dos resultados da execução destas técnicas. Os autores recomendam em seu processo o uso de entrevistas ou de sessões de *brainstorming* com os respectivos *stakeholders* que detém as informações necessárias. Quando a elicitação é realizada através de uma sessão de *brainstorming,* é necessário elicitar o maior número de informações possíveis. Estas sessões podem ser executadas sob a presença de analistas e especialistas do domínio, além de outros *stakeholders* que possuam particulares interesses na linha de produto.

Em relação à execução desta fase em LPSD é importante salientar algumas características. Uma vez que LPSD apresenta variabilidade de funcionalidades se faz necessário usar uma técnica adequada, para que esta variabilidade possa ser claramente entendida. Mellado, Fernández-Medina e Piattini (2008) executam o processo de elicitação através de análise de casos e das dependências conhecidas entre eles. Depois é realizada a identificação de requisitos comuns e variáveis da LPS e suas respectivas dependências ocorrem por meio de uma prévia análise de riscos realizada. Esta análise de riscos consiste em analisar os risco de implementar ou não uma dada variabilidade de um requisito, com base em suas dependências.

Uma vez que as informações sobre os requisitos estão disponíveis inicia-se a fase de Elaboração, destinada a refinar o modelo de requisitos, determinando os aspectos de uso do sistema, e consequentemente, definindo o comportamento do sistema perante seu uso em situações reais (PRESSMAN, 2011).

Alves et al. (2010) realizam uma revisão literária sobre o processo de Engenharia de Requisitos para LPSs e constatam que as três principais abordagens usadas para o tratamento de Requisitos, são:

- Uso da Linguagem Textual, corresponde à própria escrita de requisitos em linguagem natural;
- Uso de *Features*, uso do Modelo de *Features*, abordado na Subseção 3.2.1 anteriormente;
- Uso de Casos de Uso, prática amplamente utilizada no desenvolvimento convencional de software.

Neiva et al. (2010) representam os requisitos e suas variabilidades por meio da Visão de Casos de Uso; requisitos funcionais são definidos em termos de casos de uso e atores. Em relação ao nível de detalhamento e completude da elaboração a ser feita, isso dependerá da necessidade do contexto da LPS. Mellado, Fernández-Medina e Piattini (2008) também definem a utilização de casos de uso em seu processo. É utilizada a técnica de casos de uso e *template*s parametrizados, que são utilizados para rastrear a variabilidade do modelo. A Figura 3 apresenta um pequeno exemplo da técnica de aspectos do XML (do inglês, *Extensible Markup Language*). XML é estruturado por meio de *tags*, que permite a liberdade de uso para o usuário (W3C, 2015). Baseado nesta capacidade, Mellado, Fernández-Medina e Piattini (2008) definiram suas *tags* e a partir delas a variabilidade foi modelada; na Figura 3 é possível verificar a definição do requisito SR1, *tag* SecReq_id, e seu respectivo conjunto de valores variáveis, FIA_UID.2.1, FIA_UAU.1.1 e A.11.1.1, definidos por meio da *tag* std_name.

Após a Elaboração, é realizado a Especificação dos Requisitos. Nos processos referentes à LPSs é importante utilizar de uma abordagem que possibilite não somente a formalização dos requisitos, como também tratar da variabilidade de funcionalidades que um produto pode possuir. Nos processos de Neiva et al. (2010) e Mellado, Fernández-Medina e Piattini (2008) ambos geram Documentos de Requisitos da Linha a ser desenvolvida; nestes documentos são agrupados os Requisitos e Casos de Uso identificados em atividades prévias.

A variabilidade de funcionalidades é documentada em ambos os documentos através da elaboração do Modelo de *Features*.

Figura 3 - Tratamento da Variabilidade usando XML

```
<SecurityRequirement>
<SecReq_id= "SR1" name="Ensure User Authenticity" />
<SecReqSummary= "The security functions of [VP_SPL_app] shall
identify and authenticate an [VP_type_user] by using [Variant] before an
[VP_type_user] can bind to the shell of [VP_SPL_app]" />
<Security_use_case_template href= "....." />
<stds>
<std_name= "Common Criteria" id= "FIA_UID.2.1" name= "user
identification" description= "..." />
<std_name= "Common Criteria" id= "FIA_UAU.1.1" name= "user
authentication" description= "..." />
<std_name= "ISO27001" id= "A.11.1.1" name= "access control policy"
description= "..." />
<\stds>
<threats> <threat_id="A1T1" name="masquerading user id"/>
<threat_id="A1T2" name="unauthorised access"/> </threats>
<\SecurityRequirement>
```

Fonte: MELLADO; FERNÁNDEZ-MEDINA; PIATTINI, 2008.

A fase de Validação compreende o conjunto de práticas utilizadas para identificar e corrigir todos os erros (inconsistências, omissões, informações incorretas ou incompletas) que possam ter sido cometidos durante a execução do projeto (SOMMERVILLE, 2011). O objeto de estudo da etapa de Validação são os documentos produzidos durante as atividades da Engenharia de Requisitos até o presente momento da execução do projeto.

Mellado, Fernández-Medina e Piattini (2008) utilizam em seu processo inspeções para garantir a validade dos documentos produzidos ao longo do projeto. Neiva et al. (2010) realizam o processo de validação em seu processo através da realização de revisões nas Especificações dos Requisitos de Domínio. Esta revisão deve ser realizada por algum entendedor do domínio ou outros potenciais *stakeholders*. Neiva et al. (2010) recomendam que ao menos um *stakeholder*, que não participou do processo de produção da documentação, deve participar deste processo. As eventuais inconsistências devem ser reportadas pelo próprio revisor, em que deve descrever todos os itens que foram revisados e os respectivos problemas encontrados.

Os requisitos de um sistema precisam ser tratados como organismos vivos que mudam com a evolução do tempo (STANDISH GROUP INTERNATIONAL, 2013). Uma técnica utilizada para controlar o processo de gerência de requisitos em projetos de software, sendo comumente utilizada foi proposta em 1998 no trabalho de Jarke (1998). Assim, foi apresentada a técnica de Rastreabilidade como a habilidade de acompanhar a vida de um

requisito em ambas as direções do processo de software e durante todo o seu ciclo de vida. Jarke (1998) define a existência de quatro tipos de rastreabilidade:

- rastreabilidade a partir dos requisitos: responsável por vincular cada um dos requisitos aos componentes do sistema;
- rastreabilidade de volta aos requisitos: que verifica a aderência do sistema desenvolvido a seus requisitos;
- rastreabilidade em direção a novos requisitos: atuando no acompanhamento das mudanças nas necessidades dos usuários e nas condições técnicas assumidas, avaliando seu impacto na relevância dos requisitos definidos;
- rastreabilidade de volta a novos requisitos: associando os requisitos às estruturas que contribuíram para originá-los, essenciais na validação de requisitos.

Em relação à LPSs, o processo de gerência de requisitos deve atentar-se a tratar não somente de alterações nos requisitos mas também das *features* que podem surgir ao longo da execução do projeto. Neiva et al. (2010) definem um processo de gerência para o tratamento de evolução dos *features* de uma LPS; o processo é executado quando novas *features* são identificadas, e precisam ser aceitas, para que possam ser incluídas no escopo da linha. Uma solicitação de mudança é submetida ao Gerenciamento de Evolução; se a nova *feature* é aceita para fazer parte do escopo, é realizada então a atualização do Modelo de *Features*.

3.3.2 *Análise de Contexto*

Abowd et al. (1999) definem contexto como qualquer informação que pode ser usada para caracterizar a situação de uma entidade; uma entidade por sua vez, é definida pelos autores como uma pessoa, lugar ou objeto, que é considerado relevante para a interação entre um usuário e uma aplicação. Ali, Dalpiaz e Giorgini (2009) definem contexto como um estado parcial do mundo que é relevante para satisfazer os objetivos de um determinado *stakeholder*. Entretanto, é necessário estabelecer atividades que sejam capazes de identificar estas informações; é neste âmbito que a atividade de análise de contexto atenta-se, sendo responsável por identificar e modelar as propriedades de contexto do sistema e as possíveis transições entre seus modos de operação (CAPILLA; ORTIZ; HINCHEY, 2014).

Segundo Lapouchnian e Mylopoulos (2009), as características variáveis de um sistema definem os contextos em que o software deverá operar a fim de atender às expectativas dos stakeholders, impactando nos requisitos do sistema. Capilla, Ortiz e Hinchey

(2014) definem que as *features* de contexto devem ser claramente identificadas, modeladas e autogerenciadas para sistemas autoadaptativos e sensíveis ao contexto, permitindo assim, discutir sobre as potencias mudanças de contexto e executá-las de acordo com um conjunto estratégico ou planos de reconfiguração.

Capilla, Ortiz e Hinchey (2014) identificam quatro (4) propriedades que modelos de variabilidade em tempo de execução devem apresentar para proporcionar um gerenciamento eficiente de sistemas sensíveis ao contexto. Algumas destas propriedades já foram abordadas neste trabalho na Subseção 4.2 (capacidade de mudança do comportamento do sistema e múltiplos tempos de conexões em tempo de execução). As demais são referentes a apresentar um apropriado suporte para evolução, tal como adicionar novas capacidades em uma maneira controlada em tempo de execução; e apresentar a maneira mais rápida e que represente a melhor opção dentre as alternativas existentes, para responder a uma reação nas condições de contexto variáveis, mudanças.

Capilla et al. (2014) definem em seu trabalho, que a análise de contexto em LPSDs deve ser responsável por identificar as *features* de contexto que representam as características variáveis do domínio de LPSDs. Estas *features* de contexto são identificadas com o propósito de especificar quais *features* estão relacionadas com as mudanças de contexto de LPSDs. Deste modo, quando um produto de uma LPSD realizar uma transição de contexto, mudança de contexto A para um novo contexto de uso B, sabe-se quais *features* devem ser adicionadas à nova configuração do produto, novo contexto de uso do produto.

Capilla, Ortiz e Hinchey (2014) apresentam duas (2) abordagens que podem ser utilizadas para realizar a modelagem de *features* de contexto: (i) modelos de *features* relacionados e (ii) modelo de *features* únicos.

Em (i) o modelo de *features* é elaborado com a inclusão de um ramo adicional no modelo, que fica responsável por modelar as *features* de contexto separadamente das *features* convencionais, *features* que comuns do domínio. Fernandez e Werner (2008) recomendam o uso de um modelo de *features* de contexto separado, considerando que a modelagem de *features* de contexto juntamente com outros tipos de *features* pode comprometer o entendimento do modelo. Marinho et al. (2012) utilizam desta abordagem para o desenvolvimento de sua LPS, em que utilizam a notação UbiFEX (FERNANDES; WERNER; MURTA, 2008) para representar as informações de contexto da LPS. Mizouni et al. (2014) também fizeram o uso de um modelo adicional para representação das *features* de contexto do seu trabalho.

Já em (ii) o modelo de *features* a ser elaborado reúne tanto as *features* de contexto quanto as *features* convencionais. Capilla, Ortiz e Hinchey (2014) ainda apresentam uma comparação entre as duas abordagens apresentadas anteriormente; segundo os autores, a primeira abordagem é indicada quando contextos severos são envolvidos, enquanto a segunda simplifica o modelo de *features* e reduz o número de dependências entre as *features* do modelo.

Segundo Capilla, Ortiz e Hinchey (2014), a análise de contexto em LPSDs deve atender a cinco (5) questões principais. Estas questões encontram-se reunidas no Quadro 3.

Quadro 3 - Tarefas da Análise de Contexto de uma LPSD

Tarefas	Descrição
Identificar propriedades físicas do sistema	O projetista é responsável por identificar as propriedades pertinentes do sistema para mudanças relativas ao meio ou com as quais o sistema interage.
Modelar *features* de contexto	As propriedades físicas identificadas anteriormente são modeladas como *features* de contexto. O projetista deve prover uma taxonomia onde estas *features* de contexto podem ser classificadas ou agrupadas de acordo com as diferentes funcionalidades do sistema.
Identificar *features* modificadoras	Adicionalmente às *features* de contexto que podem ser ativadas/desativadas, o projetista tem de identificar àquelas *features* que podem impactar a variabilidade estrutural quando o sistema precisa modificar sua funcionalidade.
Definir regras operacionais	Baseado nas restrições de inclusão e exclusão definidas convencionalmente devem ser definidas as dependências operacionais que são responsáveis por guiar o processo de ativação ou desativação de *features* de contexto em tempo de execução.
Definir modos múltiplos de conexão	O projetista tem de definir os respectivos tempos de conexão para cada *feature* de contexto ou grupo associado, especificando a transição entre diferentes tempos de conexão ou entre modos operacionais do sistema.

Fonte: CAPILLA; ORTIZ; HINCHEY, 2014. (com adaptações)

3.4 Processos de LPSs

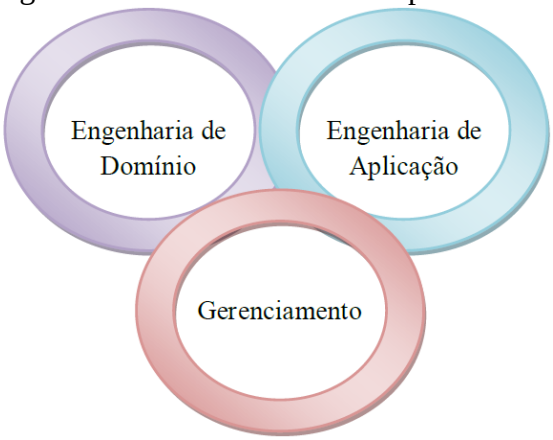
Nesta Seção são apresentados alguns processos e abordagens da Engenharia de Domínio desenvolvidos para LPSs e LPSDs. Estes processos apresentam informações a respeito de como as atividades destes processos foram realizadas, os produtos de trabalho necessários e papéis envolvidos.

3.4.1 *Framework de Linhas de Produto de Software*

Northrop et al. (2007) definem um framework para a Engenharia de Linhas de Produto de Software (SPLE, do inglês *Software Product Line Engineering*). O paradigma utilizado divide a SPLE em três (3) processos principais: Desenvolvimento dos Ativos núcleos, Desenvolvimento do Produto e Gerenciamento.

A Figura 4 apresenta a disposição de como estes três (3) processos encontram-se sobrepostos. Os três (3) processos encontram-se ligados entre si, como uma maneira de dizer, que a execução de um processo complementa a execução dos outros processos envolvidos.

Figura 4 - Processo de Linha de produto



Fonte: NORTHROP, et al., 2007 (com adaptações).

O processo de Desenvolvimento dos Ativos núcleos assemelha-se à Engenharia de Domínio; e o de Desenvolvimento do Produto à Engenharia de Aplicação. Estes conceitos podem ser revisto na Seção 3.3.

Um aspecto importante apresentado no *framework* são os documentos que são gerados como saída do processo de Desenvolvimento dos Ativos núcleos. Estes documentos apresentam um método para que se possa realizar um acompanhamento das atividades desenvolvidas e do progresso do projeto da linha. O Quadro 4 reúne estes documentos, e suas respectivas descrições.

Sobre o processo de gerenciamento, Northrop et al.(2007) o definem como o processo responsável por organizar todo o processo organizacional e técnico exigido durante a vida da linha de produto. Em relação ao Gerenciamento Organizacional, esse processo assume um papel semelhante à Gerência de Projetos no desenvolvimento tradicional de software, em que define a estrutura organizacional da linha e garante que as atividades tenham acesso aos recursos necessários para suas realizações. O Gerenciamento Técnico, especificamente em relação à Análise de Domínio, é responsável por supervisionar as atividades de construção

dos ativos, garantindo que as atividades necessárias não estão somente sendo realizadas, como também são realizadas em conformidade com o que foi planejado.

Quadro 4 - Documentos de saída de desenvolvimento de núcleos ativos

Saída	Descrição
Escopo da Linha de produto	Trata-se de uma descrição dos produtos que irão constituir a linha de produto. Estas descrições podem ser estabelecidas através das similaridades e diferenças entre produtos. Uma maneira eficiente de realizar este processo é através de um mapeamento do modelo de variabilidade utilizado para representar a linha. Por exemplo, o modelo de *features* apresenta de maneira rápida e simples as similaridades e variabilidades que uma linha de produto pode apresentar.
Ativos núcleos base	Trata-se de todos os ativos núcleos desenvolvidos. Em relação à Análise de Domínio, as especificações dos requisitos e os modelos de representação do domínio são de fundamental importância.
Plano de Produção	Trata-se de um planejamento a ser feito para cada ativo núcleo desenvolvido. Este planejamento deve conter as dependências e inclusões a serem utilizadas quando um determinado ativo for fazer parte de um produto. As restrições estabelecidas nos modelos de variabilidade representam uma grande fonte de informação para estabelecer essas dependências requeridas.

Fonte: NORTHROP et al., 2007. (com adaptações)

3.4.2 *PLUSS*

Motivados pela pobre assistência para o tratamento da modelagem de variabilidade que casos de uso tradicionais oferecem (VON DER MAßEN; LICHTER, 2002), Eriksson, Börstler e Borg (2005) definiram a abordagem PLUSS (do inglês, *Product Line Use case modeling for Systems and Software engineering*) que busca gerenciar o comportamento variável em modelos de casos de uso.

A maior contribuição do trabalho, segundo Eriksson, Börstler e Borg (2005) é em relação ao gerenciamento de variabilidade durante a etapa de projeto da linha de produto. As três principais contribuições identificadas são:

- Uma abordagem melhorada para gerenciar comportamento variável em casos de uso;

- Meio para rastrear o comportamento de caso de uso variável para o projeto do sistema; e
- Meio para gerar modelos de casos de uso de um modelo de família comum.

A abordagem encontra-se estruturada em dois processos principais: Modelagem de casos de uso e Descrição de cenários de casos de uso. Antes de apresentar a abordagem definida é importante explanar sobre alguns conceitos que foram abordados no trabalho e utilizados para o tratamento de variabilidade proposto. Fey, Fajita e Boros (2002) acreditam que o Modelo de *Features* não apresenta mecanismos para tratar de relações do tipo *at-least-one-out-of-many* (pelo menos um de muitos). Motivados por essa necessidade, Eriksson, Börstler e Borg (2005) determinaram dois novos tipos de *features*: adaptador único e múltiplo. Estes novos tipos associados aos tipos tradicionais possibilitam estabelecer novas relações, como *at-least-one-out-of-many,* pelo-menos-um-de-muitos, e *exactly-one-of-many,* exatamente-um-de-muitos.

O uso destas novas relações no modelo de *features* permite ter um bom entendimento sobre como o funcionamento da variabilidade da linha pode ajudar nas especificações dos requisitos. Por exemplo, caso um ponto de variação do modelo de *features* determine uma relação do tipo pelo-menos-um-de-muitos, sabe-se que a funcionalidade relacionada ao ponto de variação pode conter *features* variáveis (ERIKSSON; BÖRSTLER; BORG, 2005). No momento de especificar este requisito, Eriksson, Börstler e Borg (2005) definem que se deve levar em consideração dois aspectos: (i) especificação do requisito quando há apenas uma *feature* selecionada e (ii) especificação do requisito quando há mais de uma *feature* selecionada. Sobre as relações exatamente-um-de-muitos, os autores concluíram que a especificação do requisito deve atentar-se a apenas uma forma de especificação do requisito; o comportamento das variáveis formas dos requisitos deve ser o mesmo.

Para a modelagem de cenários de casos de uso foi feita uma adaptação da notação *Black Box Flow of Events,* técnica definida pelo *Rational Unified Process for Systems Engineering* (RUP-SE). Eriksson, Börstler e Borg (2005) acreditam que a notação resultante da adaptação apresenta vantagens em relação à descrição de casos de uso tradicionais, considerando que ela força os analistas a sempre pensarem sobre interfaces de partes separadas para descrever as ações do sistema e do usuário. Em relação a realizações de casos de uso, foi utilizada uma abordagem, com base na técnica *White Box Flow of Events* (RUP-SE), que permite estabelecer uma descrição em linguagem natural dos casos de uso.

Para a modelagem de variantes em casos de uso, Eriksson, Börstler e Borg (2005) identificaram quatro (4) tipos de variantes que podem existir em produtos de linhas de

produto. Os tipos de variantes identificados e suas respectivas formas de representação no Modelo de *Features* são apresentadas no **Quadro 5**.

Quadro 5 - Tipos de variabilidade e seu tratamento em PLUSS

Tipo de Variante	**Representação no Modelo de *Features***
Variação de casos de uso entre produtos da Linha.	Relacionar um ou mais casos de uso com uma *feature* ou algum outro tipo no modelo.
Variação do conjunto de cenários de casos de uso presentes em um caso de uso.	Relacionar um cenário ou mais com uma *feature* ou algum outro tipo no modelo.
Variação do conjunto de passos incluídos em cada cenário de caso de uso.	Relacionar passos de cenários com *features* ou algum outro tipo no modelo.
Variação de aspectos transversais que podem afetar vários casos de usos em vários níveis.	Aspectos são modelados como parâmetros de casos de uso.

Fonte: ERIKSSON; BÖRSTLER; BORG, 2005. (com adaptações)

Para descrever variantes em especificações de casos de uso, foi definido que cada caso de uso estabelecido apresente seus respectivos fluxos de execução, abordando todas as possibilidades de passos de execução (ERIKSSON; BÖRSTLER; BORG, 2005). Por exemplo, uma vez que um caso de uso encontra-se vinculado à uma *feature* obrigatória, que representa um passo obrigatório no fluxo do caso de uso, é necessário especificar o comportamento desta *feature*. Outro exemplo é quando um caso de uso encontra-se relacionado com um conjunto de *features* do tipo Ou Inclusivo (OR), que representa um conjunto de passos opcionais no caso de uso; com isso, é necessário definir como cada *feature* do conjunto do tipo Ou Inclusivo irá realizar o passo de execução do caso de uso. Com a definição dos passos de execução do conjunto de *features* do tipo OR, o gerenciamento de variabilidade torna-se possível de ser realizado devido a especificação dos passos de todas as variantes do ponto de variação (*features)*.

3.4.3 *RiDE: The RiSE process for domain engineering*

Santana de Almeida (2007) apresenta um processo completo direcionado à Engenharia de Domínio de uma LPS, RIDE, em português Processo do RiSE (Grupo de Reuso em Engenharia de Software) para Engenharia de Domínio. Entretanto, nesta Subseção será abordado apenas o conjunto de atividades referentes à Análise de Domínio. O processo

encontra estruturado em três (3) atividades principais: (i) Planejar, (ii) Modelar e (iii) Validar Domínio.

Em (i) o processo apresenta um conjunto de atividades com o intuito de estabelecer um conhecimento sólido do domínio que a linha destina-se a tratar. A primeira atividade apresentada é opcional e responsável por verificar a viabilidade de se construir uma infraestrutura de reuso para o domínio em foco. Esta atividade assemelha-se ao Estudo de Viabilidade realizado na fase de Concepção da Engenharia de Requisitos.

O restante das atividades deste processo atenta-se na identificação, e posteriormente, exploração das possíveis *features* da linha. Este processo ocorre através da exploração do domínio, que inclui a identificação de *stakeholders* e dos seus respectivos objetivos associados ao domínio, e do levantamento de informações referentes ao domínio (documentação, aplicações em uso). As *features* identificadas compõem o grupo de *features* candidatas a fazerem parte linha. Quando se têm este grupo formado, ocorre a execução das atividades responsáveis por estabelecer os objetivos associados a cada *feature* e os benefícios de adicionarem-nas à linha. Estas atividades representam uma boa maneira para verificar a real necessidade de implementação de cada *feature* na linha.

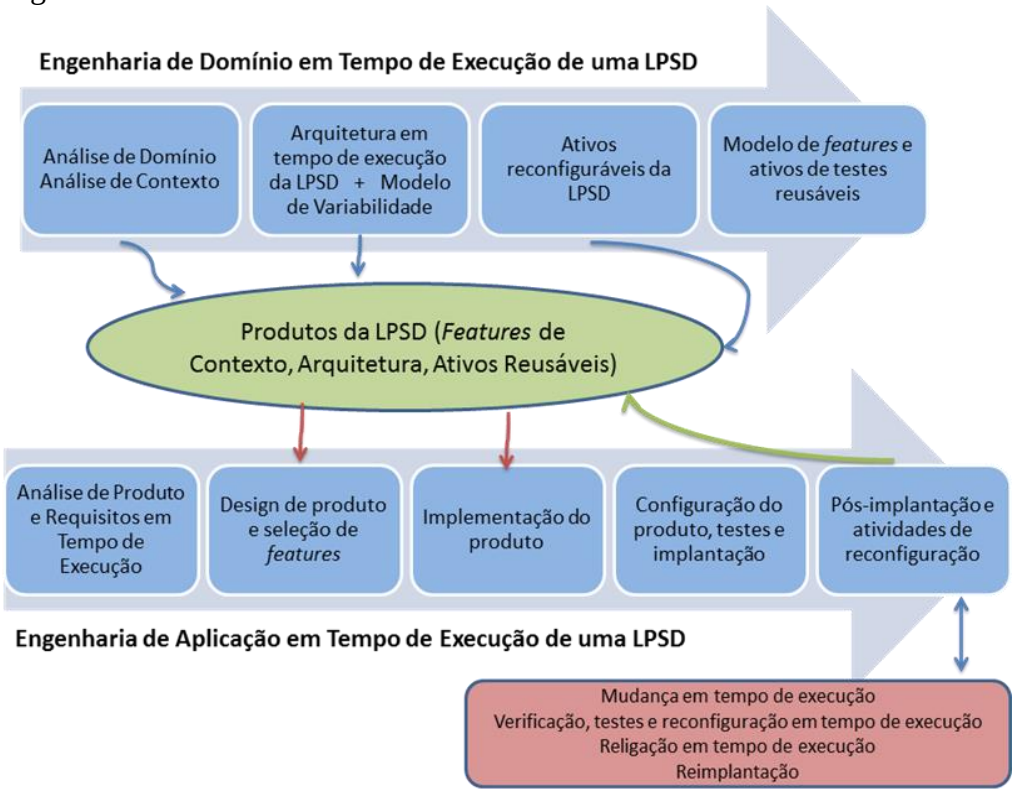
Em (ii), com o conjunto de *features* já estabelecido, é realizado a modelagem do domínio, através da construção do Modelo de *Features*. Uma importante consideração apresentada é que a presença de *features* opcionais nas camadas mais baixas do modelo indicam diferentes possíveis maneiras de se ter reuso; o contrário indica dificuldades nas opções de reuso. Esta observação representa uma boa prática para a construção do modelo de *features*.

Finalmente em (iii) é apresentada a atividade para validação do domínio. A validação ocorre em todos os aspectos do processo, o que representa um comprometimento em garantir que os requisitos identificados correspondem de fato às necessidades do domínio. Em relação às *features*, elas são avaliadas através de análises em suas definições, verificando a motivação para uma *feature* fazer parte da linha, e quais os *stakeholders* interessados em sua implementação. O Modelo de Domínio, que reúne as necessidades do domínio, é avaliado através de um mapeamento entre as necessidades dos *stakeholders* com a respectiva representação da necessidade no modelo de domínio estabelecido. A última atividade de validação corresponde à avaliação da documentação do domínio, em que se avalia a completude da documentação associada ao domínio.

3.4.4 *Framework para caracterização de LPSDs*

Capilla et al. (2014) apresentam um processo de ciclo de desenvolvimento de uma LPSD adaptada do processo proposto Van der Linder, Schmid e Rommes (2007). O processo reúne um conjunto de atividades responsáveis por atender às características que uma LPSD possui, que são: gerenciamento e suporte em tempo de execução, conexão dinâmica e múltipla, e, autoadaptação e sensibilidade ao contexto para comportamento autonômico. O processo definido encontra-se representado na Figura 5.

Figura 5 - Ciclo de vida de atividades de uma LPSD



Fonte: CAPILLA et al., 2014. (com adaptações).

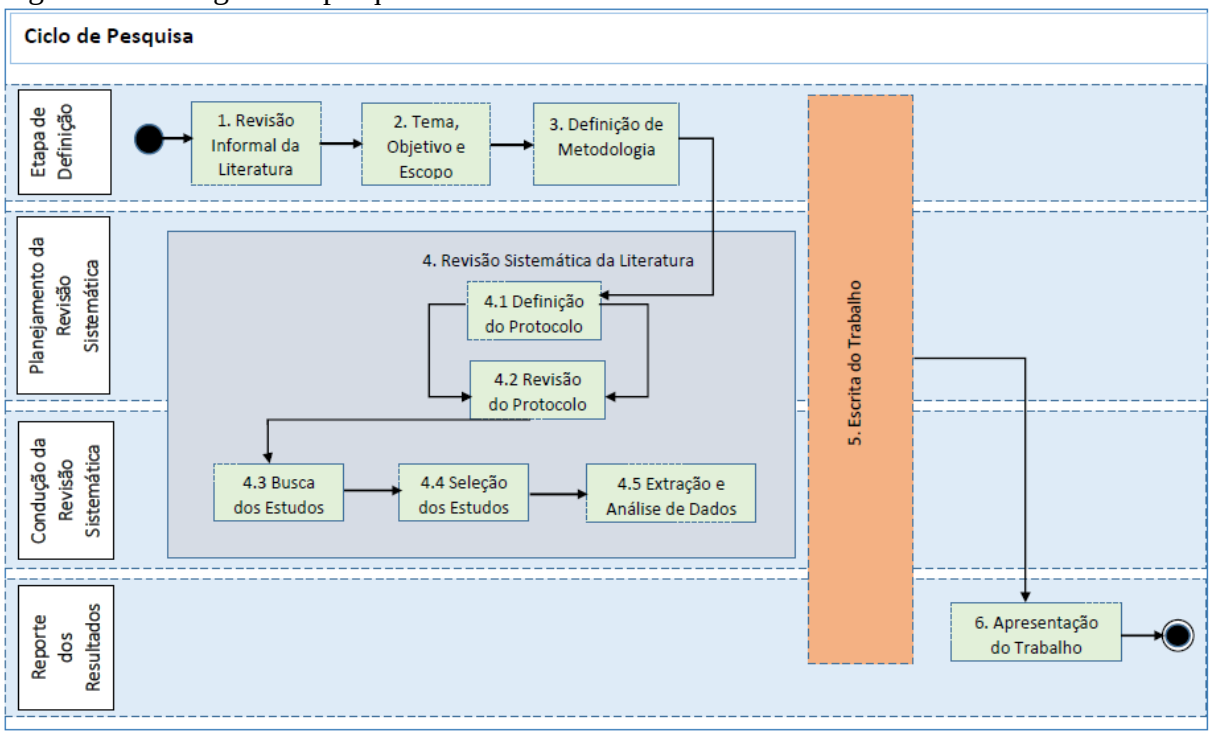
Em relação à Análise de Domínio, Capilla et al. (2014) definem a divisão desta atividade em duas: (i) Análise de Domínio e (ii) Análise de Contexto. Em (i) se continua a identificar os requisitos e as respectivas *features* da linha, mas adicionalmente se deve modelar as propriedades de contexto, que exploram as características do ambiente em tempo de execução. É através desta modelagem de propriedades que são identificados os pontos de variabilidade que podem sofrer alterações durante o tempo de execução do produto. Em relação a (ii), esta atividade é responsável por identificar as *features* de contexto que são relevantes para os produtos de LPSDs; *features* para um determinado contexto de uso são identificadas. Assim, quando o produto mudar seu comportamento para um novo contexto de uso, se sabe que *features* devem ser adicionadas à nova configuração do produto.

4 PROCEDIMENTOS METODOLÓGICOS

O objetivo desta Seção é apresentar o ciclo geral da pesquisa estabelecido para a execução deste trabalho. A Figura 6 apresenta as etapas e as atividades relacionadas, definidas a partir do planejamento até o reporte dos resultados do trabalho.

Cada etapa de pesquisa encontra-se representada na Figura 6 por meio de raias tracejadas; cada uma das etapas definidas corresponde a uma subseção nesta seção. Adicionalmente a cada etapa, há um conjunto de atividades que orientam a execução, e posteriormente, o alcance dos objetivos estabelecidos para cada etapa da pesquisa. Cada atividade é representada na Figura 6 por meio de um retângulo pontilhado. O processo de escrita do trabalho é representado por um retângulo tracejado, que foi realizado durante toda a execução do trabalho, ocorrendo em modo paralelo às demais etapas. A ordem de execução das atividades é indicada pelo fluxo das setas de cada atividade.

Figura 6 - Ciclo geral da pesquisa



Fonte: Elaborado pelo autor.

4.1 Definição da Pesquisa

Esta etapa foi responsável por definir o escopo da pesquisa a ser considerado no trabalho, bem como a metodologia adotada. Nas subseções seguintes, estas etapas são apresentadas de maneira detalhada.

4.1.1 Revisão Informal da Literatura

A fim de obter um conhecimento maior sobre as abordagens que são utilizadas para realizar a engenharia de requisitos e o gerenciamento de variabilidade em LPSDs, foi realizado um estudo exploratório com o objetivo de identificar trabalhos que apresentassem abordagens e desafios acerca destes temas. Por meio do estudo destes trabalhos foi possível entender quais as necessidades e os desafios aos quais se deve estar atento para a realização das atividades da engenharia de requisitos e do gerenciamento de variabilidade em LPSDs. Após este estudo deu-se início ao planejamento da revisão sistemática.

4.1.2 Tema, Objetivo e Escopo

A fase de Engenharia de Requisitos para o desenvolvimento de qualquer produto de software representa um importante meio para identificar as necessidades do usuário, e a partir deste entendimento, prosseguir com o desenvolvimento do projeto. Esta importância aumenta consideravelmente quando trata-se de linhas de produto, em que os requisitos identificados na engenharia de domínio são transformados em produtos, e estes produtos são utilizados ao longo de toda a vida útil da linha.

A análise de domínio deve compreender as atividades responsáveis por capturar as necessidades do domínio, através da identificação e exploração das fontes de requisitos do domínio. Em seguida, é necessário desenvolver os requisitos, identificar e solucionar requisitos em conflitos, e por fim, documentar os requisitos para que possam ser usados nas atividades posteriores.

Adicionalmente à análise de domínio, executa-se a análise de contexto, que trata-se de uma atividade responsável por identificar os contextos de uso, que a linha deve suportar, e os componentes que estes contextos necessitam para realizar seu funcionamento de maneira correta. Em relação ao processo de requisitos, esta atividade deve identificar, especificar e agrupar as *features* de contexto, reunindo essas informações de contextos de uso do produto no modelo de variabilidade da linha.

Em relação ao domínio de LPSD, as atividades da engenharia de requisitos precisam atentar-se para aspectos particulares de uma LPSD, como em relação à elicitação de requisitos, em que há a necessidade de agrupar os requisitos através do estabelecimento de um modelo de variabilidade, por exemplo o modelo de *features*. Em relação à especificação de requisitos, esta atividade deve ser responsável por estabelecer como será o processo para especificar a variabilidade da linha de produto através do uso de cenários ou casos de uso dos

requisitos; e em relação ao gerenciamento da variabilidade, de como gerenciar a variabilidade da linha entre os produtos resultantes.

Diante disto, este trabalho busca identificar este conjunto de atividades da engenharia de requisitos e do gerenciamento de variabilidade com o foco direcionado à uma LPSD.

4.1.3 Definição da Metodologia

Como este trabalho busca identificar, avaliar e comparar técnicas comumente utilizadas na análise de domínio e de contexto da engenharia de domínio de LPSDs, optou-se pela realização de uma revisão sistemática da literatura. Esta metodologia permite interpretar todas as evidências disponíveis relacionadas para uma questão de pesquisa específica (KITCHENHAM, 2007).

A abordagem utilizada como referência para realizar a revisão sistemática deste trabalho segue as orientações propostas por Kitchenham et al. (2009) e Kitchenham (2007). As etapas e atividades relacionadas a cada etapa adotadas neste trabalho também seguem às orientações definidas por Kitchenham (2007). Adicionalmente segue-se o processo de execução realizado no trabalho de Lins Barreiros (2015).

Como ferramenta de apoio à execução da revisão sistemática foi utilizado a ferramenta StArt®[1], *State of the Art through Systematic Reviews*. A ferramenta StArt® apoia a realização da revisão sistemática desde a elaboração do protocolo de revisão até a consolidação dos resultados. Entretanto, como trata-se de uma aplicação *desktop*, a ferramenta não permite o trabalho em paralelo entre os pesquisadores envolvidos na execução do trabalho, em algumas etapas da pesquisa. Para solucionar esta limitação, foi estabelecido *templates*, permitindo que diferentes pesquisadores realizassem uma mesma etapa da pesquisa simultaneamente. Os *templates* produzidos encontram-se na Seção de Apêndices deste trabalho.

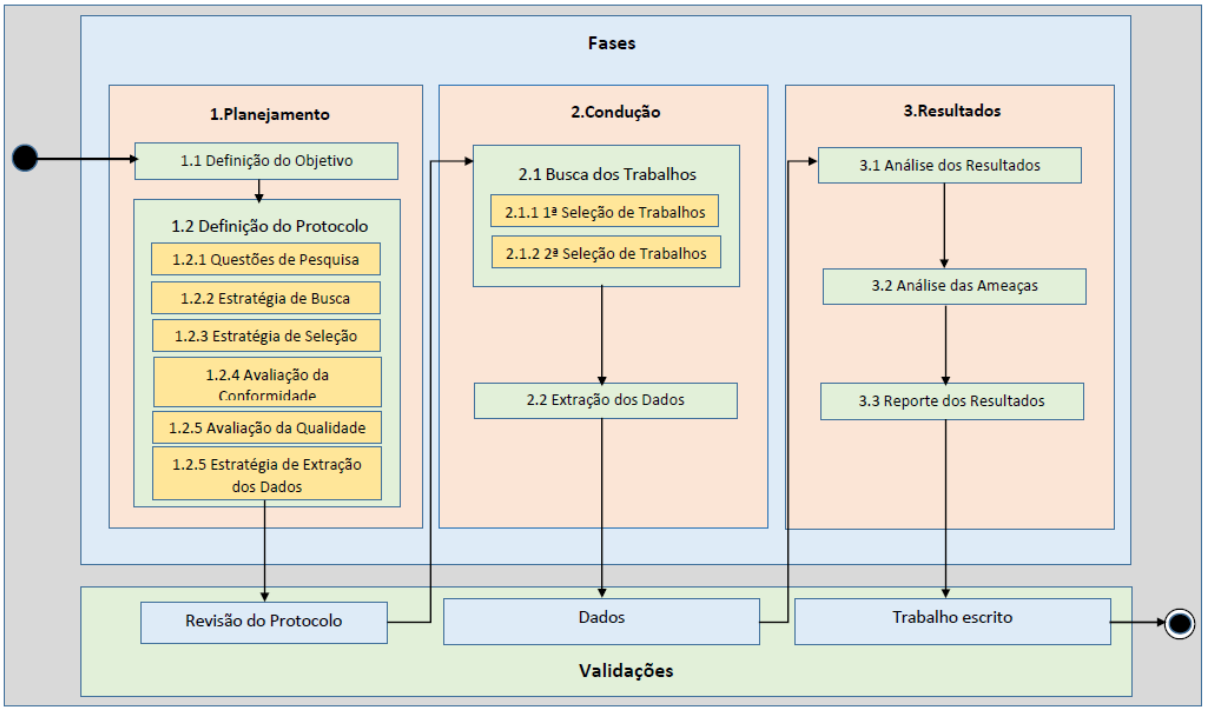
4.2 Planejamento da Revisão Sistemática

Esta seção apresenta o conjunto de atividades planejadas para todo o processo relacionado à execução da revisão sistemática. A Figura 7 apresenta o conjunto de atividades estabelecidas para esta etapa de planejamento. A execução deste conjunto de atividades associado a esta etapa é responsável pelo estabelecimento do protocolo de revisão. O objetivo

[1] http://lapes.dc.ufscar.br/tools/start_tool

inerente a cada uma destas atividades do protocolo, bem como orientações de como elas foram realizadas, são apresentados nas próximas subseções. O documento completo do protocolo de revisão encontra-se na Seção de Apêndices deste trabalho (Apêndice C).

Figura 7 - Metodologia para realização da revisão sistemática



Fonte: Elaborado pelo autor.

4.2.1 *Definição de Objetivo*

O primeiro passo realizado foi definir o objetivo que esperava-se alcançar com a execução da revisão. O objetivo estabelecido para esta revisão sistemática concentra-se em identificar as atividades e abordagens da engenharia de requisitos e como ocorre o gerenciamento de variabilidade na engenharia de domínio de LPSDs.

4.2.2 *Definição do Protocolo*

Para guiar o processo de execução da revisão sistemática foi definido o Protocolo de Revisão, que é responsável por apresentar uma visão clara e concisa dos procedimentos a serem executados durante a realização da revisão. A seguir, será abordado como cada uma das atividades relacionadas com o estabelecimento do protocolo foram definidas.

4.2.2.1 <u>Questões de Pesquisa</u>

Questões de Pesquisa representam um guia pelo qual o processo de revisão sistemática se baseará para alcançar o objetivo associada à revisão. Ao final da revisão espera-

se que as questões de pesquisa definidas a priori possam ser fortemente atendidas. Primeiramente foi estabelecida uma questão de pesquisa principal e a partir desta questão de pesquisa foi identificado questões de pesquisa secundárias. A seguir é apresentada a questão de pesquisa principal.

RQ 1: Como a engenharia de requisitos e o gerenciamento de variabilidade são executados na engenharia de domínio de LPSDs?

O objetivo desta questão de pesquisa principal é identificar como a engenharia de domínio de LPSDs é apoiada pelas atividades da engenharia de requisitos e de como o gerenciamento de variabilidade é realizado com base nas particularidades que uma LPSD possui. As atividades da engenharia de requisitos irão compor o conjunto de atividades de duas etapas principais, Análise de Domínio e Análise de Contexto, e serão classificadas com base neste mesmo princípio.

A seguir, são apresentadas as questões de pesquisa secundárias estabelecidas para este trabalho.

RQ 1.1: Quais atividades da engenharia de requisitos e do gerenciamento de variabilidade são usadas em LPSDs?

Identificar o conjunto de atividades da engenharia de requisitos e do gerenciamento de variabilidade que são executadas na engenharia de domínio de LPSDs. Adicionalmente busca-se identificar para cada atividade identificada:

- Produtos de trabalhos utilizados;
- Responsáveis e envolvidos na execução da atividade;
- Dependências entre atividades, quando existentes;
- Ferramentas utilizadas, quando disponível.

RQ 1.2: Quais abordagens são usadas para documentar os requisitos em LPSDs?

Identificar o conjunto de abordagens utilizadas para documentas os requisitos do domínio da LPSD. Dada às particularidades de LPSDs, busca-se verificar se as abordagens utilizadas diferem das abordagens comumente utilizadas em LPSs tradicionais e/ou dos requisitos de sistemas tradicionais de software.

RQ 1.3: Quais produtos de trabalhos da engenharia de requisitos e do gerenciamento de variabilidade são produzidos em LPSDs?

Identificar o conjunto de produtos de trabalho que são gerados no processo de especificação do domínio da LPSD. Estes produtos podem estar relacionados com a documentação dos requisitos do domínio e da representação da variabilidade.

RQ 1.4: Quais abordagens são usadas para representar a variabilidade de LPSDs?

Determinar quais são as abordagens utilizadas para representar as características variáveis e comuns do domínio da LPSD. Como LPSDs possuem além de características convencionais de linhas de produto, *features* convencionais, é necessário que estas características particulares, *features* de contextos, sejam tratadas por estas abordagens.

RQ 1.5: Quais abordagens são usadas para apoiar o processo de reconfiguração em LPSDs?

Identificar como a especificação do domínio apoia as atividades do processo de identificação de mudança de contexto de uso, que resulta em uma reconfiguração do produto. As abordagens podem expressar como ocorre este processo de identificação de mudanças de contexto.

RQ 1.6: Quais abordagens são usadas para eliminar possíveis inconsistências no modelo de variabilidade de LPSDs?

Estas abordagens podem ser executadas tanto durante a especificação do domínio, fase estática do projeto da LPSD, quanto durante a execução do produto, fase dinâmica, quando disponível.

Questões de Estrutura

Questões de Estrutura representam os critérios utilizados para definir o escopo pelo qual as questões de pesquisas são inseridas. Kitchenham (2007) aponta alguns pontos de visão a serem considerados como questões de estrutura. Montagud, Abrahão e Insfran (2012) utilizam cinco (5) destas estruturas em seu trabalho; o Quadro 6 reúne os tipos de pontos de visão adotados pelos autores.

Quadro 6 - Tipos de questões de estrutura

Ponto de Visão	Descrição
População	Uma área ou objeto que é utilizado como meio de estudo para realização de um trabalho.
Intervenção	Metodologia ou procedimento de software que direciona-se para uma determinada questão.
Comparação	Metodologia utilizada para comparar a intervenção em uso.
Resultados	Fatores clínicos e econômicos que serão usados para comparar as intervenções.
Contexto	O meio em que o trabalho foi realizado, acadêmico, industrial.

Fonte: MONTAGUD; ABRAHÃO; INSFRAN, 2012. (com adaptações)

O Quadro 7 apresenta os pontos de visão estabelecidos para esta revisão sistemática.

Quadro 7 - Questões de estrutura estabelecidas

Ponto de Visão	Descrição
População	Trabalhos para a engenharia de requisitos e gerenciamento de variabilidade da engenharia de domínio de LPSDs.
Intervenção	Atividades e abordagens no contexto de LPSDs.
Comparação	Utilização de critérios de qualidade para identificar as melhores práticas para um determinado fim.
Resultados	Identificar um conjunto de atividades, providas de técnicas consistentes e apoio ferramental, quando disponível, utilizadas para a Engenharia de Domínio em LPSDs.
Contexto	Pesquisa acadêmica.

Fonte: Elaborado pelo autor.

4.2.2.2 Estratégia de Busca

Para realizar a busca automática pelos trabalhos foi estabelecida uma *string* de busca. Essa *string* de busca foi estabelecida por meio da extração de palavras-chave das

questões de pesquisa. Por uma decisão estratégica, os trabalhos a serem considerados como válidos nesta revisão sistemática deveriam ser escritos na língua inglesa. Por conta disso, as palavras-chave extraídas das questões de pesquisa foram traduzidas para inglês. As palavras-chave extraídas foram: *requirements, requirements engineering, elicitation, analysis, specification, verification, management, variability, feature model, decision model, variability model.*

Todos estes termos foram combinados com os termos referentes ao objetivo central deste trabalho, *Domain Analysis, Context Analysis, Domain Engineering, Dynamic Software Product Lines*, através do operador de busca AND. Para combinar todos estes termos em uma única *string* foi utilizado o operador de busca OR, resultando na *string* de busca apresentada no Quadro 8.

Quadro 8 - String de busca automática

(Software Product Line Engineering **<OR>** Domain Engineering **<OR>** Domain Analysis **<OR>** Context Analysis)
[AND]
(Requirements **<OR>** Requirements Engineering **<OR>** Elicitation **<OR>** Analysis **<OR>** Specification **<OR>** Verification **<OR>** Management)
[AND]
(*Feature* Model **<OR>** Variability Model **<OR>** Decision Model **<OR>** Domain Model)
[AND]
(Software Product Line **<OR>** Product Family)
[AND]
(Autonomic **<OR>** Pervasive **<OR>** Ecosystems **<OR>** Dynamic **<OR>** Context-aware* **<OR>** Adaptive)

Fonte: Elaborado pelo autor.

Fontes de Dados

Esta parte do processo de planejamento da revisão foi responsável pela definição dos tipos de trabalhos a serem considerados como válidos. Foi definido como tipos de trabalhos válidos, o conjunto de trabalhos provenientes de conferências (artigos) e periódicos (revistas acadêmicas), que se dedicavam a tratar de assuntos referentes à área de LPSDs.

Em relação às fontes dos trabalhos, durante a realização de testes com a *string* de busca, foi verificado que ocorria o retorno de interessantes trabalhos de diferentes fontes, e que sempre eram retornados trabalhos de importantes conferências e *journals*, trabalhos

conhecidos previamente pelos pesquisadores, S06 e S13; informações sobre estes trabalhos encontram-se disponíveis na Seção de Apêndices deste trabalho, Apêndice A. Assim, não foi definido um conjunto de fontes de estudos a serem considerados nesta revisão sistemática. Em vez disso, foi estabelecido que todas as fontes, dos respectivos trabalhos retornados pela busca, seriam consideradas válidas desde que os artigos de controle (S06 e S13) estivessem no conjunto de trabalhos retornados pela busca.

Grande parte de trabalhos publicados encontram-se disponíveis em bibliotecas digitais. As bibliotecas digitais usadas neste trabalho foram: Scopus[2], EI Compendex/Inspec[3] e Web of Science[4].

4.2.2.3 Estratégia de Seleção dos Estudos

Critérios de Inclusão e de Exclusão

A definição de critérios de inclusão representa uma importante forma de auxiliar o processo de avaliação dos trabalhos encontrados, dado que estes critérios são utilizados para estabelecer quais trabalhos farão parte do escopo da revisão sistemática ou não. O Quadro **9** apresenta os critérios de inclusão definidos para este trabalho.

Quadro 9 - Lista dos critérios de inclusão estabelecidos

Critérios de Inclusão
Trabalhos sobre atividades da engenharia de requisitos para engenharia de domínio em LPSDs.
Trabalhos sobre abordagens para atividades da engenharia de requisitos em LPSDs.
Trabalhos sobre abordagens para tratamento da variabilidade de requisitos.
Trabalhos sobre abordagens para tratamento de inconsistências na configurações dos produtos da LPSD.

Fonte: Elaborado pelo autor.

Assim como os critérios de inclusão definem quais trabalhos farão parte do escopo da revisão sistemática, os critérios de exclusão são os responsáveis por determinar quais trabalhos não farão parte deste escopo. Os critérios de exclusão estabelecidos para este trabalho são apresentados no Quadro **10**.

[2] www.scopus.com.br
[3] http://www.engineeringvillage.com/
[4] http://login.webofknowledge.com/

Quadro 10 - Lista dos critérios de exclusão de trabalhos

Critérios de Exclusão
Trabalhos direcionados a outras atividades de LPSDs diferentes das atividades da engenharia de requisitos e/ou gerenciamento de variabilidade.
Trabalhos sobre características de LPSs.
Processo de Requisitos para LPSs ou outras áreas.
Trabalhos fora do escopo de linhas de produto.
Trabalhos duplicados.
Trabalhos não escritos em inglês.
Trabalho fora do formato de artigos válidos.

Fonte: Elaborado pelo autor.

Para auxiliar o processo de análise dos trabalhos, a partir dos critérios de inclusão e exclusão definidos, foi utilizado o suporte que a ferramenta StArt® apresenta. Entretanto, devido a necessidade dos pesquisadores trabalharem em paralelo durante as análises dos trabalhos, foi estabelecido o uso de um *template*. Este documento reúne detalhes da análise feita, guardando informações de quando a análise foi realizada, quem foram os responsáveis por sua execução e quais foram os critérios utilizados para determinar a inclusão/exclusão de um trabalho. O *template* utilizado encontra-se na Seção de Apêndices deste trabalho (Apêndice D).

4.2.2.4 Avaliação da Conformidade

O processo estabelecido para avaliação da conformidade entre os revisores sobre a classificação dos trabalhos foi baseado nos métodos da Estatística de Kappa, que apresenta apresenta uma medida de concordância permitindo verificar a confiabilidade que um dado objeto apresenta quando este objeto é avaliado por mais de uma (1) pessoa. Esta medida pode variar entre 1 e -1, em que o valor 1 representa total concordância, 0 representa conformidade por acaso, e -1 representa concordância inferior ao acaso, ponto de discórdia entre os envolvidos (VIERA; GARRET, 2005).

O Quadro 11 apresenta de maneira resumida os possíveis níveis de concordância que um objeto pode obter. Para facilitar o cálculo de Kappa foi definido um *template*; este *template* encontra-se na Seção de Apêndices deste trabalho (Apêndice G).

Quadro 11 - Valores de referência - Estatística de Kappa

Estatística de Kappa	Nível de Concordância
< 0	Inferior à chance por acaso
0.0	Pobre concordância
0.01 – 0.20	Leve concordância
0.21 – 0.40	Razoável concordância
0.41 – 0.60	Moderada concordância
0.61 – 0.80	Substancial concordância
0.81 – 0.99	Quase perfeita concordância

Fonte: (VIERA; GARRET, 2005) com adaptações.

4.2.2.5 Avaliação da Qualidade

A Avaliação da Qualidade busca avaliar o conteúdo dos trabalhos selecionados com base em critérios de qualidade. A abordagem utilizada foi baseada no trabalho de Kitchenham, Mendes e Travassos (2006), que realizam esta avaliação por meio da definição de seis (6) questões de alto nível e, posteriormente, a definição de uma função de valoração, que permite atribuir um valor numérico, de maneira individual a cada questão, com base nas variações de atendimento às questões pelos trabalhos. Assim, a partir da soma dos valores de cada questão, foi possível avaliar a qualidade do conteúdo de cada trabalho perante a variação de muito pobre, soma igual a zero (0), e excelente, soma igual a seis (6).

Seguindo a abordagem descrita anteriormente foram estabelecidas seis (6) questões para este trabalho, que são apresentadas no Quadro **12**.

Quadro 12 - Lista dos critérios de qualidade estabelecidos

Critérios de Qualidade
As atividades utilizadas para especificação do domínio encontram-se definidas?
As atividades utilizadas apresentam descrições e/ou orientações de como devem ser realizadas?
Há a definição de produtos de trabalho de entrada e saída para as atividades?
Há a definição de papéis responsáveis pela execução e/ou participação das atividades?
As abordagens utilizadas pelas atividades encontram-se definidas?
As abordagens utilizadas apresentam descrições e/ou orientações de como devem ser realizadas?

Fonte: Elaborado pelo autor.

Foi definida uma função de valoração baseado na completude das questões estabelecidas pelos trabalhos. Levando em consideração a existência de trabalhos que

pudessem apresentar certo nível de detalhamento para algumas atividades do processo de requisitos e não para outras, foi adotado o modelo de avaliação utilizado pelo Guia de Avaliação do MPS.BR, que trata-se de um modelo que permite avaliar de maneira objetiva os processos de software e de serviço de uma organização (SOFTEX, 2013). O processo de valoração varia entre 1, para trabalhos que satisfazem à questão avaliada, e 0, para trabalhos que não satisfazem. A função de valoração associada a este processo encontra-se representada no Quadro 13.

Quadro 13 - Função de valoração dos critérios de qualidade

Completude do Critério	Valoração associada
Totalmente satisfeito	1
Largamente satisfeito	0,75
Parcialmente satisfeito	0,5
Fracamente satisfeito	0,25
Não satisfeito	0

Fonte: Elaborado pelo autor.

Para facilitar a execução dessa avaliação, os critérios identificados foram organizados em um *template*. Esse *template* encontra-se na Seção de Apêndices deste trabalho (Apêndice F).

4.2.2.6 Extração de Dados

Para a extração de dados dos trabalhos foi utilizada a abordagem adotada por Montagud, Abrahão e Insfran (2012); os autores dividem as questões de pesquisa em vários critérios. Estes critérios podem ser classificados por meio de atributos de qualidade e medidas. Para auxiliar o processo de extração de dados foi estabelecido um *template*; este *template* encontra-se na Seção de Apêndices deste trabalho (Apêndice H). O Quadro **14** apresenta os critérios identificados a partir das questões de pesquisa.

A definição destes critérios ajudou a ter uma rápida e concisa visão sobre quais aspectos da engenharia de requisitos e do gerenciamento de variabilidade um determinado trabalho atendia, e posteriormente, ajudou durante a fase de reportar os resultados da revisão sistemática.

Quadro 14 - Guia de valores para extração de dados

Questão de Pesquisa	Questão de Análise	Critério associado
RQ01.1	Atividades da Análise de Domínio	Concepção, Elicitação, Elaboração, Negociação, Especificação, Validação, Gerenciamento.
	Atividades da Análise de Contexto	Identificar propriedades físicas do sistema, Modelar *features* de contexto, Identificar *features* modificadores, Definir regras operacionais, Definir modos múltiplos de conexão (CAPILLA; ORTIZ: HINCHEY, 2014).
	Papéis	Engenheiro de Domínio, Especialista do Domínio, Gerente de Projeto, Analista, Cliente, Projetista, outros.
RQ01.2	Abordagens para Especificação	Especificação de Requisitos, Especificação de Variabilidade de *Features*.
RQ01.3	Produtos de Trabalho	Documento de Requisitos, Documento de Especificação de Requisitos, Modelo de *Features*, Modelo de *Features* de contexto.
RQ01.4	Representação de Variabilidade	Modelo de *Features*, Modelo de Decisão, outros.
RQ01.6	Tratamento de inconsistências	Tratamento em Modo estático, Tratamento em Modo dinâmico.

Fonte: Elaborado pelo autor.

4.2.2.7 **Revisão do Protocolo**

Com a definição do protocolo de revisão foi realizado sucessivas revisões deste documento; este processo de revisão foi realizado por um especialista em LPSDs. O intuito destas revisões era garantir que o documento estabelecido encontrava-se de acordo com as necessidades e exigências estabelecidas para o objetivo principal deste trabalho. A maior parte das mudanças realizadas foram relacionadas à *string* de busca e à definição dos componentes dos *templates* adotados.

4.3 Condução da Revisão Sistemática

O processo de condução da revisão sistemática consistiu em realizar as atividades de seleção e análise dos trabalhos resultantes das buscas, manual e automatizas nas bibliotecas digitais estabelecidas. Este processo encontra-se detalhado na Seção 5 deste trabalho.

4.4 Reporte da Revisão Sistemática

A última etapa é responsável por apresentar os resultados da revisão sistemática. Com o conjunto final de trabalhos foi feita uma análise a fim de mapear quais trabalhos, atividades e abordagens, satisfazem a quais questões de pesquisa, obtendo assim o relatório final da revisão.

Esta etapa também foi responsável pela escrita deste trabalho. Este processo de escrita, diferentemente das outras etapas que ocorreram de maneira sequencial, foi realizado ao longo do processo pelo qual as atividades planejadas foram executadas, em um processo iterativo e incremental.

5 RESULTADOS

Nesta Seção serão apresentados os resultados deste trabalho, de acordo com os procedimentos da revisão sistemática que foram executados, conforme o planejamento da revisão apresentado na Seção 4.

5.1 Conduzir Revisão Sistemática

Conduzir a Revisão Sistemática agrupa as atividades responsáveis por selecionar os trabalhos da pesquisa, classificá-los, por meio dos critérios de inclusão e exclusão, e extrair as informações relevantes.

5.1.1 *Processo de Busca de Trabalhos*

Com o planejamento da execução da revisão sistemática estabelecido, foi realizada a identificação dos primeiros trabalhos. O processo de identificação foi executado por meio de uma busca automática, realizada no dia 14 de agosto de 2015 nas três (3) bibliotecas digitais selecionadas, utilizando a *string* de busca definida. Adicionalmente foi realizada uma busca manual nas edições do ano de 2015 das conferências SPLC e VaMoS. Este processo foi feito por meio de uma busca manual nos *proceedings* das respectivas conferências do ano em questão.

Após o processo de busca dos trabalhos, todos os resultados foram adicionados à ferramenta de apoio à execução da revisão. Os resultados advindos da busca automática das bibliotecas Scopus e EI Compendex/Inspec foram adicionados por meio da importação de arquivos XML, gerados pelas próprias bibliotecas digitais. Já os resultados advindos da biblioteca Web of Science e da busca manual foram adicionados à ferramenta de maneira manual.

O Quadro **15** apresenta os resultados encontrados em cada biblioteca digital e, adicionalmente, os resultados advindos da busca manual. Ao final, o conjunto inicial de trabalhos constituiu um grupo de quinhentos e oitenta e um (581) trabalhos.

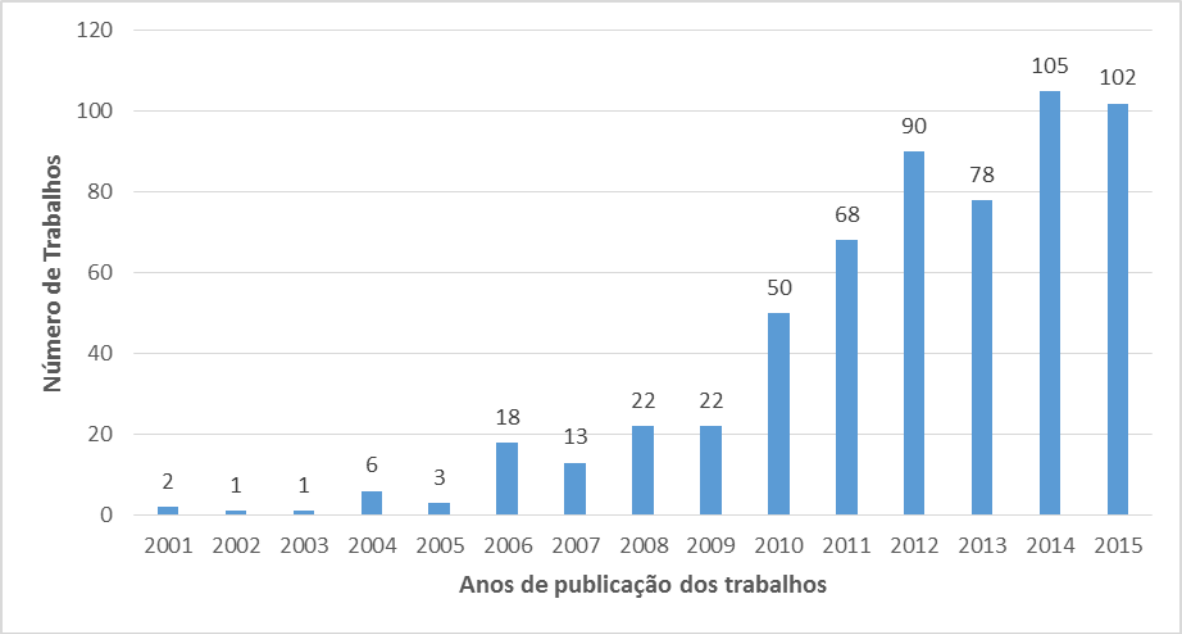
O histograma ilustrado no Gráfico 1 apresenta os trabalhos identificados agrupados pelos seus respectivos anos de publicação. Os trabalhos identificados são datados a partir de 2001 até 2015. Analisando o histograma é possível verificar que trata-se de um diagrama de tendência assimétrica do tipo negativo, que significa que a maioria dos trabalhos identificados concentram-se entre os anos de 2010 e 2014 constatando que os estudos nesta área vem ganhando notoriedade ao longo dos últimos cinco (5) anos.

Quadro 15 - Número de trabalhos encontrados por biblioteca

Fonte	Número de trabalhos encontrados
Scopus	496
EI Compendex/Inspec	7
Web of Science	7
Busca Manual	71

Fonte: Elaborado pelo autor.

Gráfico 1 - Distribuição dos trabalhos selecionados por ano pela busca realizada.



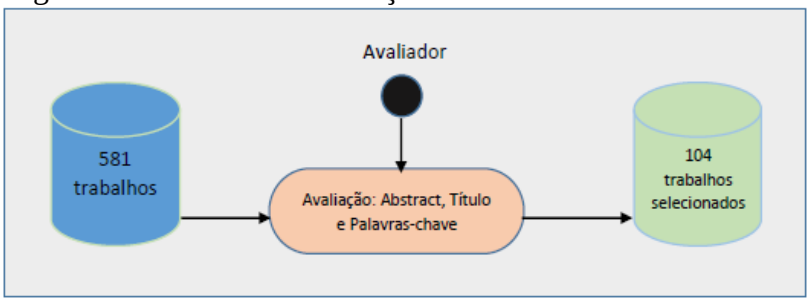
Fonte: Elaborado pelo autor.

5.1.1.1 Primeira Seleção de Estudos

Com a busca realizada deu-se início ao processo de seleção e análise dos trabalhos. A primeira seleção realizada foi responsável por eliminar os trabalhos que apesar de terem sido selecionados pela busca eletrônica não apresentavam informações importantes para satisfazer às questões de pesquisa desta revisão. Isto ocorre devido ao fato de que alguns trabalhos podem conter algumas das palavras-chave da *string* de busca estabelecida, e mesmo assim, não corresponderem de maneira satisfatória com a área em estudo associada.

O processo para realizar esta primeira seleção consistiu na leitura dos *abstracts*, palavras-chave e título de cada trabalho, realizado por um (1) pesquisador. Cada trabalho foi avaliado com base nos critérios de inclusão e exclusão, estabelecidos durante o planejamento da revisão, e posteriormente classificados como aceitos ou não para a próxima fase de análise. A Figura 8 apresenta o processo pelo qual a seleção dos trabalhos foi realizada.

Figura 8 - Processo da 1ª seleção de trabalhos



Fonte: Elaborado pelo autor.

Foram eliminados nesta primeira seleção apenas os trabalhos claramente fora do escopo de atuação desta pesquisa, 464 trabalhos. Dentre os critérios de exclusão mais utilizados para a exclusão de um trabalho, o mais utilizado foi o critério relacionado a trabalhos não direcionados à área de atuação da engenharia de requisitos e gerenciamento de variabilidade em LPSDs. Adicionalmente, os trabalhos duplicados foram eliminados de maneira automatizada pela própria ferramenta de apoio à execução da revisão, 13 trabalhos. Ao final deste processo foram extraídos cento e quatro (104) trabalhos. O Quadro 16 apresenta as informações gerais do estado da pesquisa após a primeira seleção.

Quadro 16 - Estado da pesquisa após a 1ª seleção

Categoria	Quantidade
Trabalhos selecionados	104
Trabalhos rejeitados	464
Trabalhos duplicados	13

Fonte: Elaborado pelo autor.

O Quadro **17**, apresenta o número de trabalhos selecionados por cada biblioteca de busca após a primeira seleção.

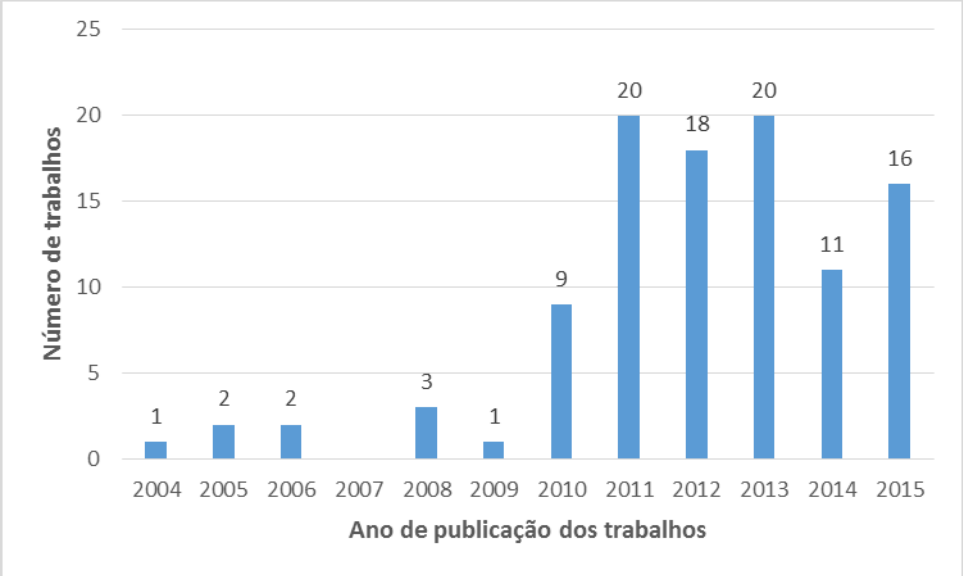
O histograma de frequência ilustrado no Gráfico **2**, apresenta os trabalhos selecionados agrupados pelos respectivos anos de publicação. A partir do histograma é possível verificar que os trabalhos selecionados concentraram-se entre os anos de 2010 e 2015.

Quadro 17 - Trabalhos selecionados na 1ª Seleção agrupados por biblioteca

Fonte	Número de trabalhos encontrados
Scopus	91
EI Compendex/Inspec	4
Web of Science	0
Busca Manual	9

Fonte: Elaborado pelo autor.

Gráfico 2 - Distribuição dos trabalhos da 1ª seleção classificados por ano



Fonte: Elaborado pelo autor.

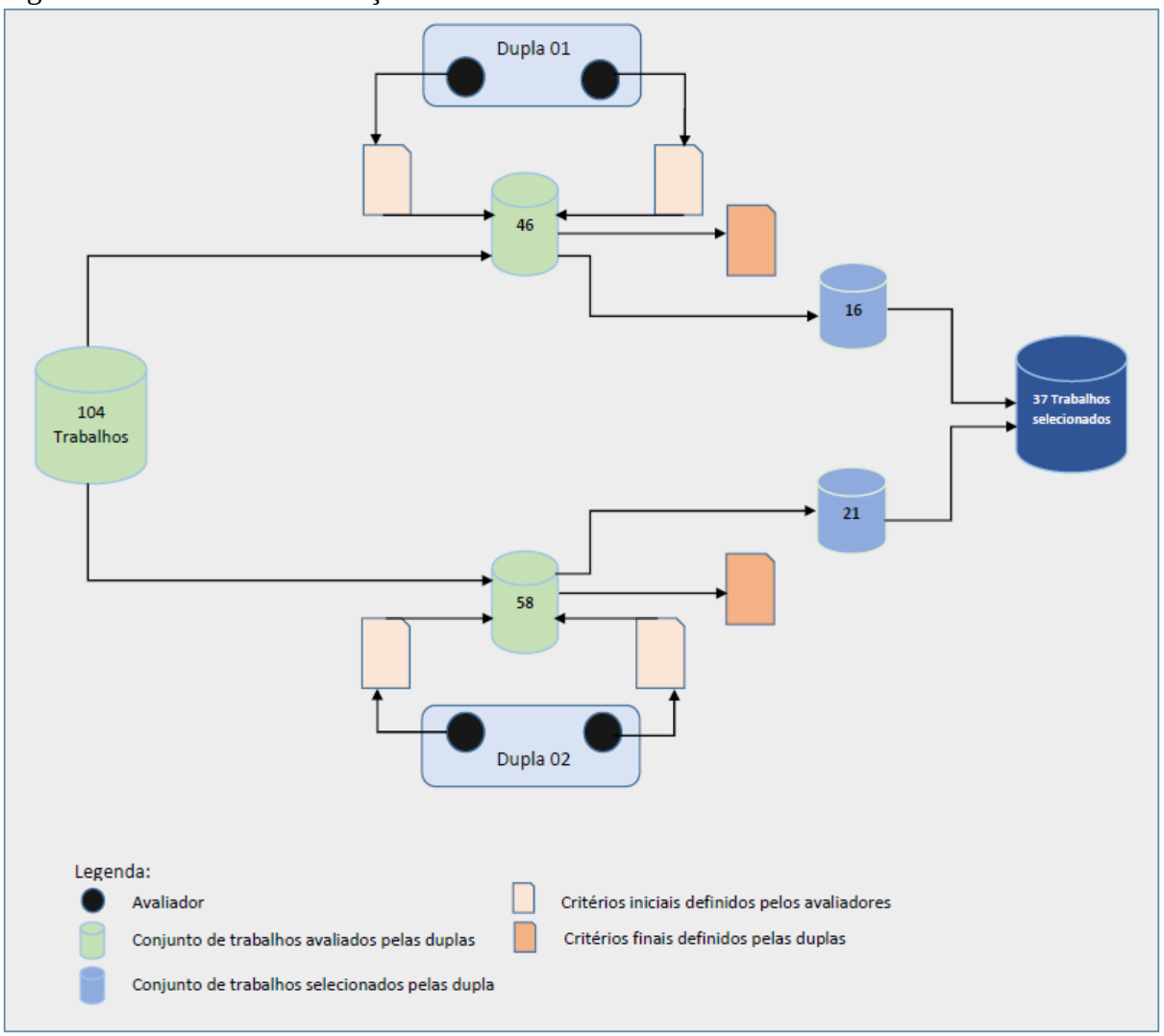
5.1.1.2 Segunda Seleção de Trabalhos

A segunda seleção de trabalhos ocorreu por meio de uma leitura dos trabalhos selecionados durante a primeira seleção por três (3) pesquisadores. O acesso dos leitores aos trabalhos deu-se por meio do acesso autorizado aos periódicos e conferências dos respectivos trabalhos selecionados, perante a regulamentação da Universidade Federal do Ceará, UFC. Entretanto alguns trabalhos não se encontravam disponíveis pelo acesso disponibilizado pela UFC. Para isso, foram enviados e-mails aos autores dos trabalhos solicitando acesso aos trabalhos indisponíveis. Quando estes e-mails não foram respondidos foram estabelecidas parcerias com pesquisadores de outras universidades e solicitado a estes o acesso aos trabalhos. A Figura 9 ilustra como este segundo processo de seleção de trabalhos ocorreu.

O conjunto de cento e quatro (104) trabalhos foi separado em dois conjuntos de trabalhos distintos, quarenta e seis (46) e cinquenta e oito (58) trabalhos em cada subconjunto,

respectivamente. Em seguida, cada subconjunto de trabalhos foi atribuído a uma dupla de leitores, formada a partir dos três (3) pesquisadores responsáveis por esta fase. Cada leitor era responsável pela leitura de cada trabalho, e atribuir um resultado preliminar de classificação ao trabalho, com base nos critérios de avaliação atribuídos pelo leitor após a leitura.

Figura 9 - Processo da 2ª Seleção de trabalhos



Fonte: Elaborado pelo autor.

O processo de leitura dos trabalhos foi diferente para cada uma das duas duplas de leitores. Os leitores da primeira dupla realizaram a leitura dos trabalhos por meio da técnica de leitura em pares; cada leitor teve a responsabilidade de ler de maneira individual cada trabalho. Em relação à segunda dupla, o primeiro leitor realizou a leitura dos trabalhos, e ao final, o segundo leitor realizou uma leitura superficial dos trabalhos com base nos critérios atribuídos pelo primeiro leitor. Quando o segundo leitor discordava da classificação

preliminar atribuída a um trabalho, ele atribuía uma nova classificação ao trabalho com base em seus próprios critérios de classificação.

Ao final da leitura de todos os trabalhos, cada dupla de leitores foi responsável por verificar a conformidade dos resultados preliminares atribuídos aos trabalhos. Quando inconformidades ocorriam, era de responsabilidade da dupla de leitores discutir sobre os critérios definidos por cada leitor e sobre a corretude pela qual estes critérios foram atribuídos aos trabalhos, com o objetivo de chegar a um consenso entre eles. Com a resolução de todas as inconformidades chegou-se a um conjunto de critérios finais para cada trabalho, e posteriormente, sua classificação final.

Com o objetivo de verificar a conformidade pela qual este processo de classificação dos trabalhos foi feita pelos leitores, foi feita uma análise de conformidade, seguindo as orientações do planejamento da revisão. Os resultados desta análise são apresentados no Quadro **18**.

Quadro 18 - Avaliação da conformidade da 2ª seleção de trabalhos

Avaliações	Nível de Conformidade	Avaliação
Dupla 01	0,741401274	Substancial concordância
Dupla 02	0,609450338	Substancial concordância
Média das Duplas	0,675425806	Substancial concordância

Fonte: Elaborado pelo autor.

A partir da leitura do Quadro **18** é possível verificar que o processo de classificação dos trabalhos apresentou uma conformidade contínua entre as duas duplas de leitores, e posteriormente, na avaliação final desta segunda classificação. Estes dados revelam uma conformidade substancial entre os envolvidos na seleção.

Ao final deste processo foram selecionados trinta e sete (37) trabalhos. O Quadro **19** apresenta as informações gerais do estado da pesquisa após a segunda seleção realizada, número de trabalhos selecionados, rejeitados e duplicados por cada dupla de leitores, respectivamente.

Quadro 19 - Estado da pesquisa após a 2ª seleção

Dupla	Trabalhos Selecionados	Trabalhos Rejeitados	Trabalhos Duplicados
Dupla 01	16	30	1
Dupla 02	21	37	0
Total	37	67	1

Fonte: Elaborado pelo autor.

O Quadro **20** apresenta o número de trabalhos selecionados por cada biblioteca de busca após a segunda seleção.

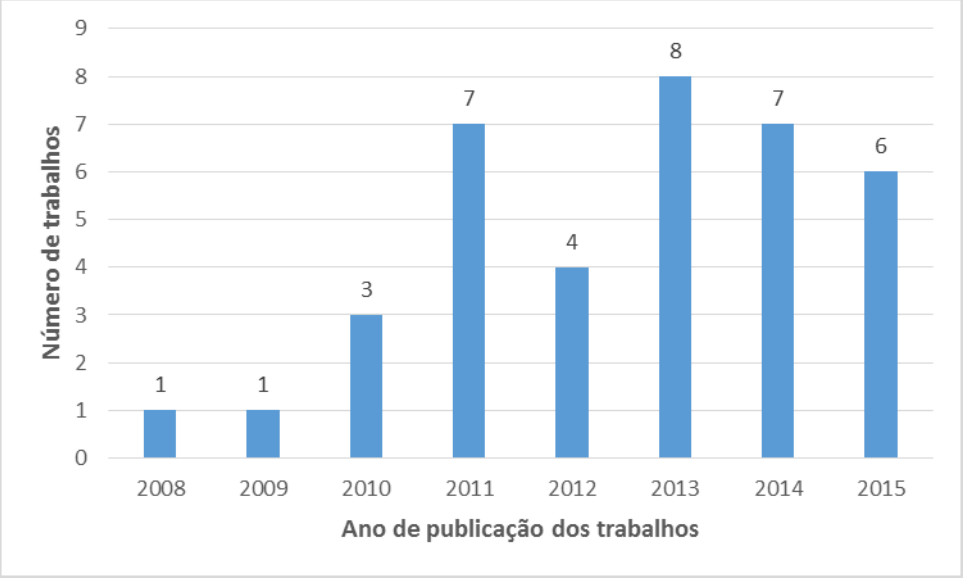
Quadro 20 - Trabalhos selecionados na 2ª Seleção agrupados por biblioteca

Fonte	Número de trabalhos encontrados
Scopus	31
EI Compendex/Inspec	2
Web of Science	0
Busca Manual	4

Fonte: Elaborado pelo autor.

O histograma de frequência absoluta, Gráfico 3, apresenta a quantidade de trabalhos selecionados agrupados pelos respectivos anos de publicação. Há uma concentração predominante de trabalhos entre os anos de 2011 e 2015.

Gráfico 3 - Distribuição dos trabalhos da 2ª seleção classificados por ano



Fonte: Elaborado pelo autor.

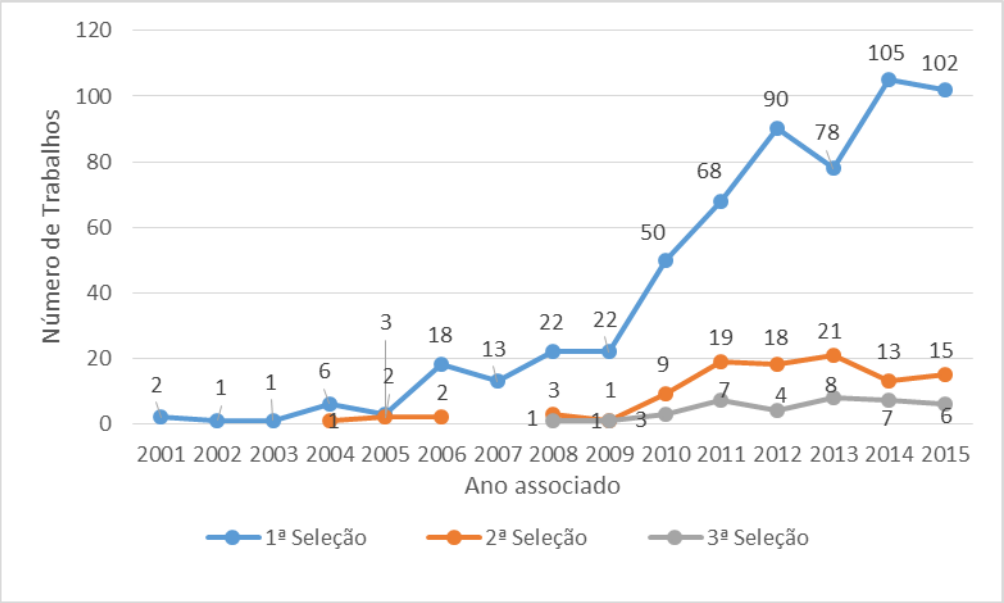
5.1.2 *Extração de informações*

Com os resultados da segunda seleção dos trabalhos iniciou-se o processo de extração das informações de cada trabalho. Este processo foi realizado a partir da leitura completa de cada trabalho e foi realizado por um (1) pesquisador. Adicionalmente foi realizado a avaliação da qualidade de cada um dos trabalhos.

5.2 Visão Geral

Antes da análise de cada questão de pesquisa, esta Seção busca apresentar uma análise geral dos dados obtidos pela revisão sistemática realizada. O Gráfico **4** apresenta um mapeamento do número de trabalhos selecionados ao longo dos anos associados aos três processos de seleção de trabalhos.

Gráfico 4 - Trabalhos selecionados ao longo dos anos



Fonte: Elaborado pelo autor.

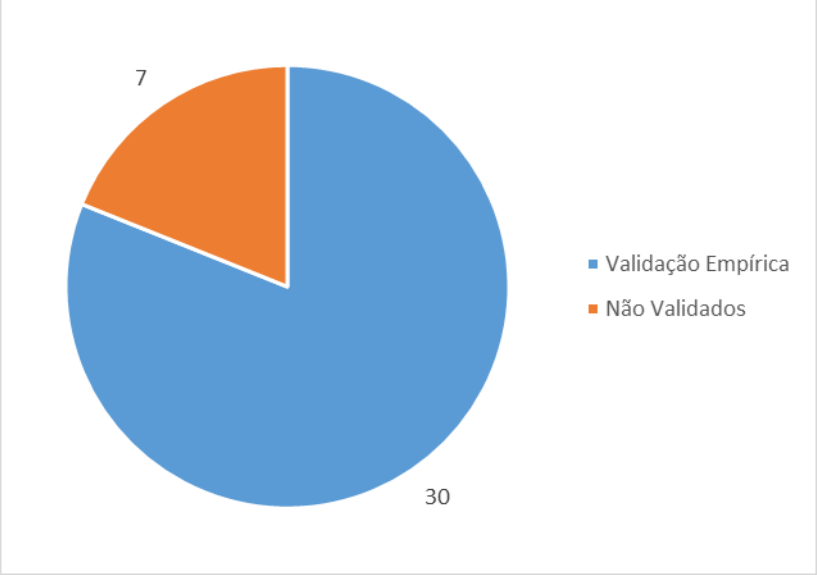
De acordo com o Gráfico **4** é possível ver que há um crescimento constante no volume de produção de trabalhos na área de LPSDs. Em relação ao terceiro (3º) processo de seleção, nota-se que o trabalho mais velho selecionado é datado do ano de 2008, S33, revelando que trata-se de uma área de pesquisa nova. A partir do ano de 2011 é possível ver também que há uma regularidade maior do número de trabalhos selecionados por ano, variando entre quatro (4) e sete (7) trabalhos.

Em relação às fontes dos trabalhos verifica-se um amplo espaço amostral dado o número de trabalhos selecionados; não há a supremacia de uma determinada fonte em relação

as demais. Entretanto, as duas (2) fontes mais representativas foram: SPLC, *Software Product Line Conference*, e Science of Computer Programming; a lista completa das fontes dos trabalhos encontra-se na Seção de Apêndices (Apêndice B).

Informações sobre os procedimentos de validação utilizados pelos trabalhos podem ser vistos no Gráfico 5. Trinta (30) trabalhos realizam uma validação empírica, enquanto o restante, sete (7) trabalhos, não realizam nenhuma forma de validação até o momento da publicação de seus trabalhos. Este alto índice de trabalhos, que realizam validações, representam um compromisso em garantir que os resultados advindos destes trabalhos passaram por aplicações práticas reais de uso.

Gráfico 5 - Procedimentos de validação adotados pelos trabalhos



Fonte: Elaborado pelo autor.

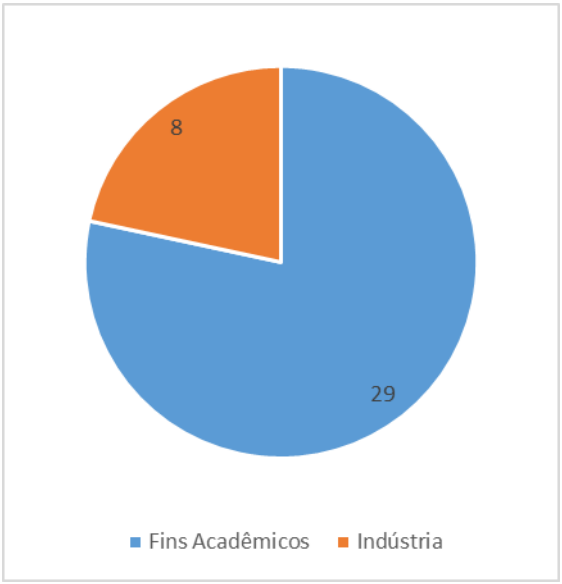
Informações sobre os modos de validação adotados podem ser vistos no Gráfico 6. Dentre os trabalhos que realizaram a validação empírica, vinte e sete (27) fizeram suas validações por meio de casos de estudos; os outros três (3) trabalhos restantes realizaram a validação por meio de experimento controlado, prototipação e implementação do produto final, respectivamente.

Gráfico 6 - Formas de procedimentos de validação adotados

Fonte: Elaborado pelo autor.

Sobre o contexto de uso dos trabalhos, Gráfico 7, foi constatado que a maioria, vinte e nove (29) trabalhos, foram realizados para fins acadêmicos. Enquanto o restante, oito (8) trabalhos, foram destinados à indústria. Isto reforça a ideia de que as pesquisas na área acadêmica vêm crescendo, mas em contrapartida, já nota-se trabalhos que buscam promover o uso de LPSDs na indústria.

Gráfico 7 - Contextos de uso dos trabalhos



Fonte: Elaborado pelo autor.

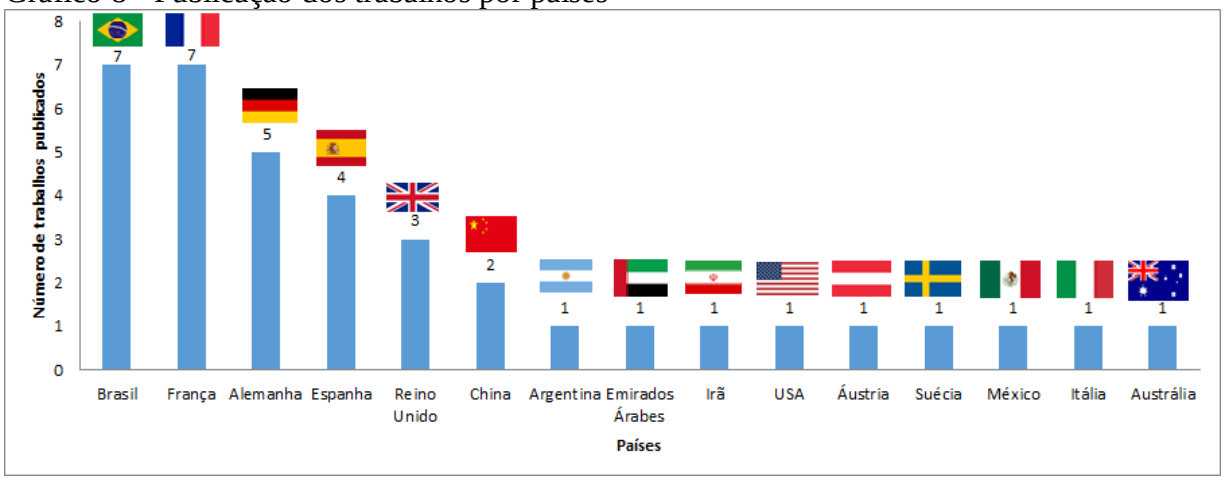
Outro resultado interessante está relacionado com o nível de qualidade dos trabalhos avaliados. Após a execução da avaliação da qualidade de cada trabalho foi possível

verificar que somente um trabalho, S06, atingiu a nota máxima na avaliação, 6. A maioria dos outros trabalhos, oitenta e seis por cento (86%), apresentam pontuações entre três (3) e cinco (5). A pontuação atribuída a cada um dos trabalhos pode ser vista na Seção de Apêndices deste trabalho, Apêndice A.

Foi possível verificar também o estado das pesquisas realizadas em LPSDs classificados por países; o Gráfico 8 apresenta estas informações. Ao todos, dos quinze (15) países responsáveis pelas publicações de trabalhos, Brasil, França e Alemanha foram os três (3) países que mais se destacaram.

Na França os esforços de pesquisas concentram-se em três (3) universidades, Université Lille Nord de France, Université de Nice Sophia Antipolis e Université de Paris 1 Panthéon-Sorbone. No Brasil há iniciativas de trabalhos por parte de algumas instituições, Universidade Federal do Rio de Janeiro (UFRJ), Universidade Federal de Campina Grande (UFCG), Universidade Estadual de Feira de Santana (UEFS) e Universidade Estadual de Campinas (Unicamp), mas a maior parte dos trabalhos, três (3) trabalhos, advêm da Universidade Federal do Ceará (UFC).

Gráfico 8 - Publicação dos trabalhos por países



Fonte: Elaborado pelo autor.

Na terceira posição aparece a Alemanha, que exclusivamente como o Brasil apresenta uma universidade, Universidade Técnica de Darmstadt, com três (3) trabalhos publicados. As demais instituições foram: Universidade Técnica de Dresden e Universidade de Magdeburg.

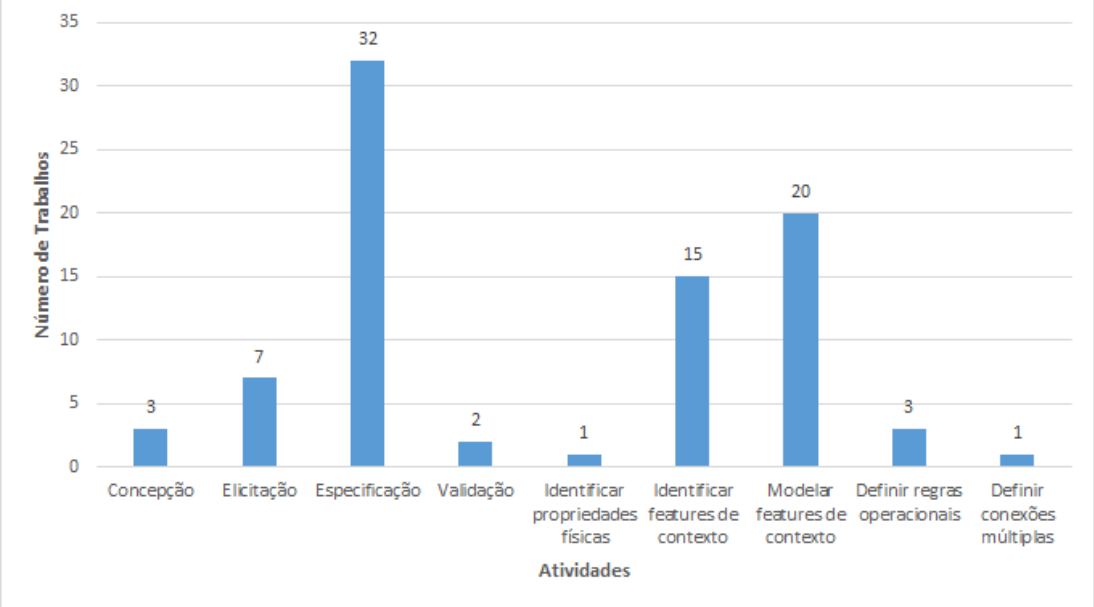
5.3 Engenharia de Requisitos e Gerenciamento de Variabilidade (RQ1)

Esta Seção busca apresentar os resultados advindos da extração de informações dos trabalhos para cada uma das questões de pesquisa secundárias estabelecidas para este trabalho.

5.3.1 *RQ1.1: Quais atividades da engenharia de requisitos e do gerenciamento de variabilidade são usadas em LPSDs?*

Para responder a esta questão de pesquisa foi analisado como o domínio da LPSD foi especificado. Para tanto, o foco da análise realizada foi em identificar as atividades envolvidas neste processo. Alguns trabalhos, que por não tratarem prioritariamente o processo de especificação do domínio, foi dada grande atenção durante a análise às seções que tratavam da especificação do domínio. Entretanto, a maioria dos detalhes de como o domínio era especificado estava mais claro nas seções de exemplos de uso e casos de estudo, do que na seção de especificação das atividades.

Gráfico 9 - Atividades realizadas na engenharia de domínio de LPSDs



Fonte: Elaborado pelo autor.

O Gráfico 9 apresenta as atividades identificadas, e adicionalmente, o número de trabalhos que apresentaram tais atividades. Adicionalmente, o Quadro 21 apresenta as atividades e as fases da engenharia de domínio a que estavam relacionadas. Ainda de acordo com o Gráfico 9 é possível verificar que as quatro (4) primeiras atividades representam as atividades que são convencionalmente realizadas em projetos de software; que são: concepção, elicitação, especificação e validação. Estas atividades representam a fase de

análise de domínio, com grande concentração das atividades de especificação e modelagem de LPSDs. As demais atividades tratam especificamente dos aspectos dinâmicos do domínio das LPSDs, análise de contexto; neste conjunto de atividades o foco é direcionado na identificação e modelagem de features de contexto.

Quadro 21- Atividades da engenharia de domínio relacionadas com suas fases de execução

Fase	**Atividades**
Análise de Domínio	Concepção, Elicitação, Especificação, Validação.
Análise de Contexto	Identificar propriedades físicas do sistema, Identificar *features* de contexto, Modelar *features* de contexto, Definir regras operacionais.
Projeto de Domínio	Definir regras operacionais, Definir conexões múltiplas.

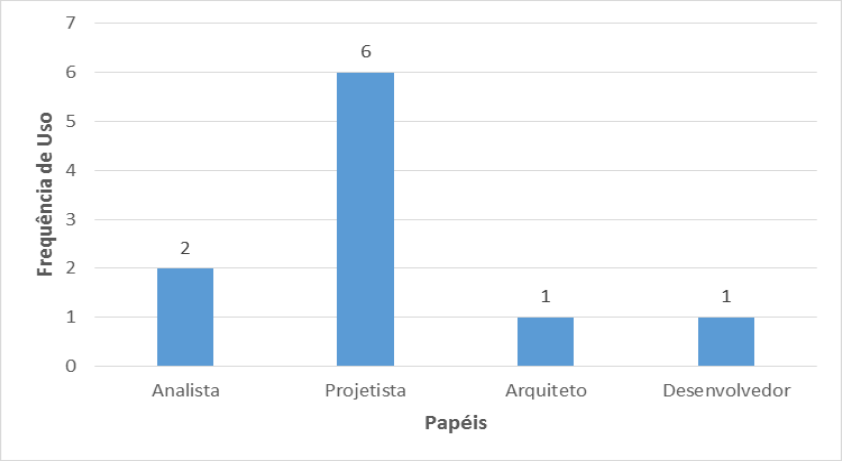
Fonte: Elaborado pelo autor.

Em relação ao modo de execução das atividades identificadas, verificou-se mudanças em como estas atividades eram realizadas, devido à capacidade de adaptação dinâmica e ao contexto que LPSDs possuem. As atividades relacionadas à modelagem do domínio foram as primeiras a apresentarem estas mudanças. Estas atividades modelam a variabilidade do domínio, por meio de um conjunto de abordagens, que comparadas com as abordagens utilizadas em LPSs tradicionais, representação da variabilidade, representam um aumento no número de abordagens disponíveis para uso.

Entretanto, estas mudanças no modo de execução não são válidas para todas as atividades. Por exemplo, as atividades de especificação e documentação de requisitos, que apesar de fazerem uso de abordagens direcionadas em identificar a variabilidade dinâmica do domínio, usam também abordagens advindas do desenvolvimento tradicional de software, diagramas UML.

Outro ponto avaliado nesta questão de pesquisa está relacionado aos papéis envolvidos na execução das atividades. Apesar de todos os trabalhos apresentarem as atividades realizadas, apenas nove (9) trabalhos apresentaram os papéis envolvidos. O Gráfico 10 apresenta os papéis identificados e sua frequência de uso pelos trabalhos.

Gráfico 10 - Papéis envolvidos nas atividades realizadas na engenharia de domínio de LPSDs



Fonte: Elaborado pelo autor.

Os papéis identificados estão mais envolvidos com as fases de especificação e modelagem do domínio de LPSDs. Por exemplo, o projetista e o arquiteto são papéis comumente vinculados às atividades de modelagem e especificação da arquitetura de uma linha de produto, S05 e S09. O Quadro **22** apresenta os papéis apresentados pelas atividades e os respectivos trabalhos relacionados.

Quadro 22 - Papéis apresentados pelas atividades da engenharia de domínio

Papel	Trabalhos
Analista	S06, S07.
Projetista	S05, S06, S15, S16, S26, S35.
Arquiteto	S09.
Desenvolvedor	S31.

Fonte: Elaborado pelo autor.

A seguir é apresentado as responsabilidades atribuídas a cada um dos papéis identificados.

- Analista: o analista é responsável pela definição e manutenção dos modelos utilizados para a especificação do domínio, S06 e S07. Adicionalmente, em S07, é atribuído a este papel a responsabilidade de definir as regras responsáveis pela capacidade de adaptação da LPSD;
- Projetista: o papel do projetista assemelha-se com a área de atuação do analista no que diz respeito a atividade de definição dos modelos de variabilidade, S35. Ele também se encontra envolvido com as atividades responsáveis pelas regras de adaptação

da linha, S05. Apesar da existência do papel do arquiteto, em alguns trabalhos foram atribuídos ao projetista a responsabilidade de modificar a arquitetura, quando necessário, S15;

- Arquiteto: responsável por especificar os atributos de qualidade relacionados com as *features* identificadas, S09. Compreende também o modo pelo qual os requisitos não funcionais são especificados e como devem ser satisfeitos, S09; e
- Desenvolvedor: apenas um trabalho apresenta o papel de desenvolvedor, S31. Apesar da definição deste novo papel, suas funções compreendem as responsabilidades dos outros três (3) papéis descritos anteriormente.

Apesar da execução de alguns trabalhos das atividades de elicitação de requisitos, tais atividades não foram atribuídas a nenhum dos papéis identificados nos trabalhos. De maneira geral notou-se que não há a definição clara das responsabilidades dos papéis, considerando a atribuição de uma mesma responsabilidade a dois (2) papéis diferentes.

Ainda nesta questão de pesquisa foi analisado como as atividades eram suportadas por apoio ferramental. Grande parte dos trabalhos, sessenta e quatro por cento (64%), não usaram ou não mencionaram o uso de ferramentas para a execução de suas atividades. Os demais trabalhos utilizaram de ferramentas para a atividade de modelagem do domínio e verificação da consistência e corretude dos modelos de variabilidade adotados. O Quadro 23 apresenta as ferramentas identificadas relacionadas a cada um de seus grupos de atuação.

As ferramentas identificadas foram agrupadas em três (3) grupos principais. Estes grupos foram:

- Representação de composição de serviço: usado para representar a composição das diferentes partes de um produto por meio de serviços. Representa uma boa maneira para conhecer e se pensar sobre o problema, a necessidade de uma solução e como esta solução deve ser realizada;
- Modelagem da variabilidade: reúne as ferramentas que apoiam as atividades de modelagem e manutenção dos modelos de variabilidade adotados; e
- Verificação de modelos: agrupa as ferramentas que realizam verificações de consistência e corretude dos modelos adotados. Vale ressaltar que estas ferramentas são usadas para avaliar os modelos de maneira estática, não há como avaliar o comportamento destes modelos em tempo de execução do produto. A subseção 5.3.6 discute de maneira detalhada sobre a avaliação da corretude dos modelos de variabilidade.

Quadro 23 - Ferramentas de suporte às atividades da engenharia de domínio de LPSDs

Atividade	Ferramentas e Trabalhos
Representação de composição de serviço	BPMN (S07).
Modelagem da variabilidade	Fama (S02), MOSkitt4SPL (S07), Atlas Model (S07), Familiar (S26), BPMN (S10), *Feature*IDE (S26), eMoflon (S8), Odyssey (S28), DOPLER (S33), VariaMos (S36, S37).
Verificação de modelos	GNU Prolog (S07), Clafer (S10), Familiar (S26), VariaMos (S36, S37).

Fonte: Elaborado pelo autor.

5.3.2 *RQ1.2: Quais abordagens são usadas para documentar os requisitos em LPSDs?*

Para responder a esta questão de pesquisa foi analisado mais cuidadosamente como as atividades de elicitação e especificação foram realizadas. Apesar da importância de se documentar os requisitos de um projeto de software, apenas seis (6) trabalhos atentaram-se a esta necessidade. O Quadro 24 apresenta as abordagens identificadas associadas aos respectivos trabalhos que as utilizam.

Quadro 24 - Abordagens identificadas para documentação da variabilidade de requisitos

Abordagem	Trabalhos
Modelo de *Features*	S01, S05
Diagrama de Classes	S13
Casos de Uso	S13
Diagrama de Sequência	S15
Schemas	S25
Goals	S35

Fonte: Elaborado pelo autor.

É possível constatar que algumas abordagens utilizadas, casos de uso, diagramas de classes e de sequência, correspondem às mesmas abordagens utilizadas no desenvolvimento de software tradicional. Apesar de que estas abordagens não terem sido desenvolvidas para suportar o conceito de variabilidade dinâmica, elas foram utilizadas para tal fim; os demais trabalhos apresentarem abordagens direcionadas ao tratamento de variabilidade.

Apesar do uso do modelo de *features* para documentação dos requisitos, não foi apresentado pelo trabalho associado como ocorreu o processo de conversão dos requisitos,

especificação em texto, para *features*. Por fim, as duas (2) últimas abordagens foram usadas por trabalhos isolados; um dos focos destes trabalhos era justamente usar e avaliar os resultados do uso de tais abordagens no contexto de LPSDs. Estas abordagens são comentadas a seguir.

A abordagem utilizando *Schemas*, Metodologia Gaia, (ZAMBONELLI; JENNINGS; WOOLDRIDGE, 2003), para a especificação de requisitos ocorreu devido à metodologia oferecer uma compreensiva análise sobre os requisitos. O processo de especificação ocorre por meio de três (3) *schemas* específicos: *Role Schema*, *Role Variation Points Schema* e *Variation Point Schema*. Sua estrutura de modelagem de variabilidade assemelha-se com o modelo de *features* tradicional. *Role Schema* é usado para representar entidades que não tinham pontos de variação; *Role Variation Points Schema* definem um *role* e os pontos de variação que um *role* podem assumir durante sua existência; e por fim, o *Variation Point Schema* é responsável por capturar as capacidades requeridas de um comportamento funcional de um *Role Variation Point*.

Outra abordagem usada foi *Goals*. Para os autores de S35, o uso de *goals* permite representar em alto nível os objetivos que um sistema tem de atingir de acordo com as condições de contexto da execução.

5.3.3 *RQ1.3: Quais produtos de trabalhos da engenharia de requisitos e do gerenciamento de variabilidade são produzidos em LPSDs?*

Para identificar os produtos de trabalho que são gerados foram analisados os produtos de trabalho que são comumente usados e gerados pelas atividades identificadas anteriormente. O Quadro 25 apresenta os resultados para esta questão de pesquisa.

O modelo de *features* bem como o modelo de *features* de contexto representam os produtos de trabalho utilizados com maior frequência, treze cada um (13). O segundo produto de uso mais utilizado é o modelo de *features* estendido; cinco (5) trabalhos estendem o modelo de *features* com o intuito de apoiar as suas próprias necessidades. Documentação relacionada à especificação de requisitos é usada para documentar os requisitos, utilizando abordagens comumente utilizadas no desenvolvimento de software tradicional; detalhes sobre estas abordagens estão disponíveis no Quadro 24.

O modelo de aspectos é usado para representar a variabilidade, em que sua abordagem de tratamento assemelha-se com a abordagem do modelo de *features*. *Weaving models* representa uma maneira que utiliza de dois (2) modelos: modelo de variabilidade e modelo de composição. O modelo de variabilidade descreve as variações dinâmicas do

processo de composição da LPSD, enquanto o modelo de composição é usado para representar as diferentes partes necessárias que são envolvidas na LPSD para suportar cada informação de contexto. Quando uma mudança acontece no modelo de variabilidade, o modelo de composição precisa com isso ser atualizado; o *weaving model* é o responsável por esse mapeamento entre estes dois (2) modelos.

Quadro 25 - Produtos de trabalhos identificados na engenharia de domínio de LPSDs

Produtos de Trabalho	**Trabalhos**
Regras de Adaptação	S24
Modelo de Aspectos	S23, S31.
Máquina de Estados	S08
Modelo de Composição	S07
Modelo de Composição Base	S07
Weaving Model	S07, S10.
Modelo de *Features*	S01, S02, S03, S04, S07, S12, S13, S20, S21, S26, S27, S28, S32, S37.
Modelo de *Features* estendido	S05, S08, S10, S16, S18, S34
Modelo de *Features* de Contexto	S01, S06, S07, S10, S11, S12, S20, S22, S28, S29, S30, S32, S34, S37
Modelo de *Features* de Adaptação	S01
Especificação de Requisitos	S13, S15, S25, S35

Fonte: Elabordado pelo autor.

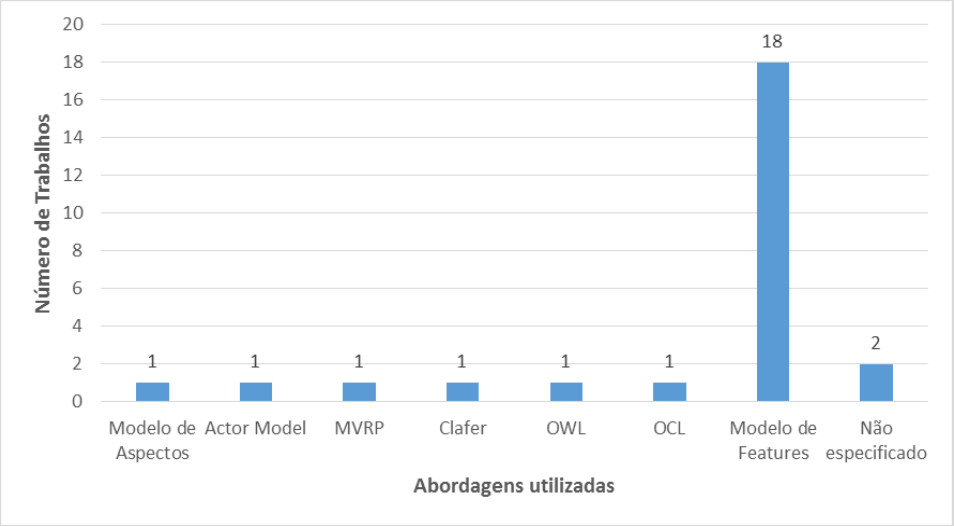
Uma extensão do modelo de composição é o modelo de composição base, que é usado para estender o modelo de composição com semânticas para variabilidade. As regras de adaptação focam em representar as regras que a LPSD usa para adaptar seu próprio comportamento quando uma mudança ocorre no ambiente de execução. Máquinas de estado são usadas para representar os estados que a LPSD poderia apresentar durante seu uso.

5.3.4 *RQ1.4: Quais abordagens são usadas para representar a variabilidade de LPSDs?*

Nesta questão de pesquisa foi feita uma análise cuidadosa na atividade de especificação. Isso se deu pelo fato de que durante a atividade de especificação é quando ocorre a escolha da abordagem a ser utilizada para representação da variabilidade, que ocorre na atividade de modelagem. O Gráfico 11 apresenta o conjunto de abordagens utilizadas bem como a frequência com que foram utilizadas; o Quadro 26, adicionalmente, apresenta as abordagens utilizadas associadas aos seus respectivos trabalhos.

De acordo com o Gráfico 11, o modelo de *features* é a abordagem mais utilizada para a representação da variabilidade de LPSDs; quase metade dos trabalhos, quarenta e nove por cento (49%), usaram-no. Diferentemente de LPSs convencionais, que utilizam fortemente outras abordagens para representação da variabilidade, como modelos de decisão, não foi constatado uma segunda opção de abordagem fortemente usada por LPSDs. As demais abordagens usadas pelos trabalhos especificam o uso de alguma abordagem isolada em relação às outras abordagens adotadas.

Gráfico 11 - Abordagens para tratamento da variabilidade em LPSDs



Fonte: Elaborado pelo autor.

Quadro 26 - Abordagens para representação da variabilidade em LPSDs

Abordagem	Trabalhos
Modelo de Aspecto	S31
Actor Model	S17
MVRP	S16
OWL	S12
OCL	S04
Modelo de *Features*	S01, S02, S03, S07, S10, S11, S13, S14, S15, S19, S21, S23, S26, S27, S28, S29, S34, S37.

Fonte: Elaborado pelo autor.

Além do modelo de *features* foram utilizadas outras abordagens; como o uso de modelos de aspectos que trata uma LPSD como um conjunto de aspectos que juntas compõem as configurações que a LPSD poderá apresentar. Outra abordagem utilizada foi *Actor Models* que se trata de um modelo para computação distribuída e concorrente. Neste modelo, um

actor trata-se de uma unidade que comunica-se com outros atores e é capaz de tomar decisões. A abordagem de MVRP (do inglês, *Model for Variable Refinement Processes*) atenta-se em estender o modelo de *features* por meio de modularizações nas *features*, que é feito por meio de sucessivas transformações na estrutura do modelo de variabilidade.

Outra maneira utilizada corresponde ao uso de OCL (do inglês, *Object Constraint Language*) que se trata de uma linguagem específica para especificar variabilidade livre de contexto.

5.3.5 *RQ1.5: Quais abordagens são usadas para apoiar o processo de reconfiguração em LPSDs?*

Esta questão de pesquisa é relacionada ao suporte pelo qual a adaptabilidade ocorre em tempo de execução. Durante a análise desta questão de pesquisa foi verificado que esse processo de especificar as abordagens está mais relacionado a uma atividade da fase de projeto de domínio. O Quadro 27 reúne os resultados desta questão de pesquisa.

Algumas das abordagens adotadas permitem não somente a adaptação do software às mudanças de contexto, mas também à adaptação às condições de uso do produto e às regras de adaptação (ECA, MAPE, MAPE-K). Essas abordagens foram consideradas como válidas neste estudo, por terem sido utilizadas para identificarem uma mudança no ambiente de uso de LPSDs.

Quadro 27 - Abordagens auxiliadoras na identificação de reconfiguração de LPSDs

Abordagens	Trabalhos
Regras de Adaptação	S07, S24
ECA	S01, S02, S22, S27, S31, S32
MAPE-K	S15, S36
MAPE	S01, S35
Regras de Transformação	S26
Modelos de Aspectos	S23
Mapeamento de Contexto	S10
PrtNets	S30
Satisfação de Restrição	S21

Fonte: Elaborado pelo autor.

É possível verificar que a abordagem mais utilizada foi a ECA (do inglês, *Event-Condition-Action*); as demais abordagens utilizadas apresentam uma distribuição uniforme

entre os trabalhos, com uma variação de um (1) a dois (2) trabalhos utilizando a mesma abordagem. ECA é baseada em regras que são criadas de a partir de diferentes fontes, podendo ser identificadas com a atividade responsável por identificar regras operacionais; detalhes sobre esta atividade são apresentados na Subseção 5.3.1.

*Feature*Ace é um framework para permitir adaptação em tempo de execução de produtos; este framework provê como um de seus produtos uma linguagem para especificação de regras de adaptação, que com base nessas regras ocorrerá o processo de reconfiguração dinâmica. O processo de especificação as regras de adaptação por meio de MAPE (do inglês, *Monitor, Analyse, Plan and Execute*) e MAPE-K (do inglês, *Monitor, Analyse, Plan, Execute and Knowledge*) seguem uma mesma abordagem. Para ambas é necessário identificar como a informação de contexto estaria acessível, especificando as regras que poderiam ser responsáveis pela identificação de uma mudança de contexto.

Regras de transformações foram utilizadas por meio de uma inter-relação com o modelo de variabilidade adotado em cada trabalho, definição das regras no próprio modelo. Apesar de modelos de aspectos serem utilizados para representar a variabilidade, tal abordagem também é válida para esta questão de pesquisa devido à definição de regras de adaptação. Mapeamento de contexto apoia o processo de reconfiguração por meio de um mapeamento responsável por descrever entre *features*, variantes e modelo de contexto. PrT nets (do inglês, *Precate/Transition*) é uma abordagem para situações de contexto e comportamentos; em S30, os autores do trabalho optaram por usar a Petri Nets devido à expressividade que esta abordagem possui.

5.3.6 *RQ1.6: Quais abordagens são usadas para eliminar possíveis inconsistências no modelo de variabilidade de LPSDs?*

Para responder a esta questão de pesquisa foi necessário uma análise em dois (2) momentos principais: (i) especificação do domínio e (ii) casos de aplicação. Assim foi possível verificar como cada trabalho tratava a consistência e corretude dos modelos de variabilidade adotados tanto durante a etapa de projeto quanto na execução do produto final. O Quadro 28 apresenta os resultados desta questão de pesquisa.

Quadro 28 - Abordagens de verificação de inconsistências em LPSDs

Abordagens	Trabalhos
Verificação dinâmica	S02, S05, S06, S08, S15.
Verificação estática	S07, S08, S10, S20, S22, S26, S28, S37.

Fonte: Elaborado pelo autor.

Embora a checagem de consistência represente um importante passo para garantir e validar os requisitos da LPSD, apenas doze (12) trabalhos mencionam a realização de alguma forma de checagem. Oito (8) trabalhos executam checagens para garantir apenas que o modelo de variabilidade adotado não apresentava inconsistências, checagens estáticas. Para realizar estas checagens são utilizadas ferramentas que podem ser revistas no Quadro 23.

Em relação aos trabalhos que realizam checagens em tempo de execução de seus produtos, apenas trabalhos específicos (S08 e S15) que se atentam a esta questão apresentam as abordagens adotadas. O processo de checagem utiliza regras de adaptação, definidas previamente, e em seguida, a aplicação de algoritmos específicos, que são responsáveis por avaliar se uma nova configuração teria alguma inconsistência ou não. Os outros trabalhos apenas citam que eles fazem esta checagem, mas não apresentam detalhes sobre como seus processos são executados. Permitir o tratamento da consistência e corretude do modelo de variabilidade em tempo de execução representa uma oportunidade de pesquisa.

5.4 Síntese dos Resultados

Esta seção apresenta uma síntese objetiva dos dados coletados para responder às questões de pesquisa abordadas anteriormente na Seção 5.3; o Quadro 29 apresenta a síntese realizada.

O Quadro 29 organiza os resultados por meio de uma classificação inicial das fases da Engenharia de Domínio, Análise de Domínio e Projeto de Domínio, seguido da especificação das atividades de cada fase. Foi considerada a fase, e posteriormente, a atividade de projeto de domínio devido à execução de duas atividades, definir regras operacionais e definir conexões múltiplas, identificadas para especificação do domínio mas que eram realizadas em alguns trabalhos durante o projeto de domínio.

Entretanto, como a execução da atividade, definir regras operacionais, também ocorre durante a análise de domínio, esta atividade encontra-se em ambas fases. Por conta disso, os resultados das abordagens utilizadas, produtos de trabalho e ferramentas, para as atividades de análise de contexto e projeto de domínio são mesclados entre si. As ferramentas identificadas e os papéis relacionados foram dispostos para todas as atividades devido a utilização das ferramentas e envolvimento dos papéis em atividades diferentes das fases da engenharia de domínio.

Quadro 29 - Síntese dos 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: Elaborado pelo autor.

5.5 Discussão dos Resultados

A engenharia de domínio compreende a fase de projeto de uma Linha de Produto de Software Dinâmica (LPSD) responsável por explorar o domínio que a LPSD irá suportar. Os resultados deste trabalho mostram que as pesquisas em LPSDs têm produzido um conjunto de produtos de trabalhos, ferramentas e abordagens para apoiar as atividades necessárias responsáveis pela engenharia de requisitos e gerenciamento de variabilidade de LPSDs. Sobre os papéis utilizados na execução destas atividades, constatou-se que na maioria dos trabalhos, 72%, vinte e sete (27) trabalhos, não há a apresentação de um conjunto de papéis envolvidos nas atividades; e, mesmo quando os trabalhos apresentam os papéis, não há um consenso sobre as responsabilidades que cada papel deve assumir. A ausência desta informação dificulta o entendimento sobre quais papéis são necessários para executar cada atividade deste processo.

De acordo com os resultados desta revisão sistemática, uma descoberta positiva foi a identificação de um conjunto de atividades utilizadas para apoiar o processo inicial de modelagem de LPSDs. Estas atividades são responsáveis não somente pela modelagem dos requisitos como também pelo tratamento da variabilidade, considerando as particularidades que uma LPSD possui.

Em relação à execução das atividades, constatou-se que as atividades relacionadas à elicitação de requisitos são executadas com menor relevância e importância pelos trabalhos analisados. A maioria dos trabalhos analisados, 83%, trinta e um (31) trabalhos, não especificam os requisitos do domínio em um documento formal; os trabalhos apenas organizam seus requisitos no modelo de variabilidade adotado. Entretanto, não foi apresentado as abordagens utilizadas para realizar a conversão das especificações formais dos requisitos, ou mesmo os próprios requisitos sem especificação, em uma feature relacionada ao modelo de *features*.

Os trabalhos que especificam os requisitos de maneira formal usam abordagens convencionalmente utilizadas no desenvolvimento de software tradicional, como a utilização de diagramas UML. Alguns trabalhos utilizam o modelo de *features* com extensões para representar os requisitos do domínio. Ferramentas são usadas para apoiar a execução do conjunto de atividades identificadas. O uso principal destas atividades está relacionado ao suporte na especificação e modelagem da variabilidade do domínio. Por outro lado, os trabalhos não apresentam ferramentas que são focadas em apoiar a concepção, elicitação e documentação dos requisitos.

Outra descoberta é que não foi constatado nos trabalhos, atividades responsáveis pelo gerenciamento de requisitos ao longo do processo de especificação do domínio. Apesar de LPSDs destinarem-se em atender às necessidades específicas de um domínio, é importante permitir que os requisitos deste domínio sejam passíveis de mudança, considerando que esta necessidade represente a real necessidade do projeto de uma LPSD.

Sobre o tratamento e identificação de Requisitos Não-Funcionais (RNF), constatou-se que estas atividades ocorrem majoritariamente na fase de projeto de domínio, fase posterior à análise de domínio. Apenas um (1) trabalho, S15, trata desta questão com a mesma relevância e importância que os requisitos funcionais (RF) recebem durante a atividade de análise do domínio. O tratamento dos RNFs está diretamente relacionado com a fase de projeto do domínio, fase da engenharia de domínio onde a arquitetura da LPSD é estabelecida. Apesar de não ocorrer o tratamento dos RNFs durante a análise de domínio, é importante garantir que o processo de modelagem da arquitetura receba da atividade de análise de domínio, as informações do RNFs, ao invés desta identificação ser feita durante a execução de alguma atividade da fase de projeto de domínio.

S09 apresenta interesse em avaliar os atributos de qualidade de LPSDs. Entretanto, o objeto de estudo destes trabalhos limita-se apenas ao modelo de *features*.

Avaliar atributos de qualidade em toda a engenharia de domínio de LPSDs representa uma oportunidade de pesquisa a ser desenvolvida.

Em relação ao gerenciamento de variabilidade, comparando os resultados deste trabalho com o de Guedes et al. (2015), foi possível identificar novos meios para representar a variabilidade, além de descobertas similares às descobertas identificadas por Guedes et al. (2015). Em relação ao tratamento de variabilidade, Guedes et al. (2015) apresentam o uso de cinco (5) abordagens para representar a variabilidade, enquanto este trabalho apresenta sete (7) abordagens. Estas novas abordagens usadas identificadas foram modelos de aspectos, *actor model*, MVRP, Clafer e OWL. Algumas abordagens foram identificadas pelos dois trabalhos, como o uso do modelo de *features*, *Goals* e OCL.

Dentre as abordagens identificadas para representação da variabilidade, verificou-se que o modelo de *features* representa a abordagem mais utilizada, 49% dos trabalhos, para representar a variabilidade em LPSDs; abordagem usada para o mesmo fim em Linhas de Produto de Software (LPS) tradicionais. Guedes et al. (2015) também apresentam esta mesma conclusão; entretanto, os autores não especificam o modo como o modelo de *features* é utilizado, baseado em suas extensões e em seu modo de organização. Neste trabalho, concluiu-se não apenas o uso genuíno do modelo de *features* mas também as variações, pelas quais este modelo foi usado; como o uso do modelo de *features* estendido e modelo de *features* de contexto.

Entretanto, apesar da alta frequência de uso do modelo de *features* não foi observado um padrão no modo de representar a variabilidade. Por exemplo, cada trabalho identifica suas *features* de contexto mas a maneira de organizar estas informações varia entre os trabalhos. Alguns trabalhos incluem suas *features* de contexto juntas com as *features* convencionais no modelo de *features*, enquanto outros criam um novo modelo independente para representar estas informações.

Em relação ao processo de reconfiguração, Guedes et al. (2015) apresentam as fontes pelas quais podem implicar em uma reconfiguração; contudo, não apresentam as abordagens que são utilizadas para identificar a necessidade de uma mudança na configuração do produto. Neste trabalho, o foco foi exatamente em identificar estas abordagens, que atentam-se em definir quais informações são necessárias e úteis para identificar que uma mudança ocorreu, e que uma reconfiguração, posteriormente, precisa ser realizada. As abordagens identificadas podem ser vistas no Quadro 27.

Um desafio relacionado à reconfiguração de produtos de LPSDs está relacionado em atualizar o estado do modelo de *features* em tempo de execução após a ocorrência de uma

reconfiguração. Essa reconfiguração altera a configuração do produto da LPSD, e, portanto, o estado do modelo de *features* associado ao produto. Assim, há a necessidade de pensar em como garantir a consistência do modelo, considerando as novas restrições que foram impostas durante a especificação do domínio. Para auxiliar este processo foram identificadas ferramentas que são usadas para checar a estrutura do modelo de variabilidade e sua consistência. Entretanto, foram identificadas ferramentas que suportam apenas a verificação da consistência ainda em tempo de projeto da LPSD; verificações em tempo de execução são realizadas por algoritmos específicos.

Outra questão relacionada à reconfiguração mas envolvendo os RNFs é a respeito dos trabalhos que realizam o processo de reconfiguração, em tempo de execução, considerarem apenas a verificação das regras e restrições impostas pelos RFs, não considerando os RNFs neste processo. Uma oportunidade de pesquisa seria como modelar e tratar RNFs no processo de reconfiguração dos produtos de uma LPSD.

6 CONSIDERAÇÕES FINAIS

Esta revisão sistemática da literatura buscou determinar como a engenharia de requisitos e o gerenciamento de variabilidade apoiam a engenharia de domínio de Linhas de Produto de Software Dinâmicas (LPSDs). Para responder a esta questão de pesquisa, estudos relevantes e atuais foram identificados seguindo uma abordagem formal de execução para a realização da revisão sistemática. Após a avaliação inicial de 581 trabalhos identificados pelas bibliotecas digitais, foram selecionados 104 trabalhos; estes trabalhos foram mais uma vez avaliados, resultando na seleção de 37 trabalhos, os quais foram utilizados para a extração das informações relevantes para este trabalho.

Devido grande parte dos trabalhos não serem destinados exclusivamente à fase de análise do domínio de LPSDs, as informações a respeito deste processo de especificação do domínio limitou-se muitas vezes em explorar as seções dos trabalhos que tratavam desta especificação e, quando disponíveis, dos procedimentos de validação destes trabalhos, casos de estudo, principalmente. Por conta disso, apenas um trabalho obteve a nota máxima de avaliação de qualidade de uma escala de 0 a 6. Entretanto, a grande maioria dos trabalhos restantes apresentaram notas entre 3 e 5, o que representa um bom nível de qualidade geral dos trabalhos analisados.

Os resultados deste trabalho mostram que, tipicamente, as atividades da análise de domínio estão concentradas na especificação e modelagem de LPSDs. Diagramas UML podem ser utilizados para representar os requisitos do domínio, bem como o modelo de *features*, que por sua vez pode também ser utilizado para representação da variabilidade. As ferramentas utilizadas apoiam principalmente as atividades de modelagem e verificação da consistência dos modelos de variabilidade estabelecidos.

6.1 Ameaças à Validade

Esta seção discute as ameaças que podem ter afetado os resultados da revisão sistemática realizada. Foi verificado que as ameaças identificadas estão relacionadas à validade de construção. Validade de construção preocupa-se em verificar se os passos realizados refletem a causa e se a saída reflete o efeito (WHOLIN et al., 2012).

A primeira possível ameaça é a respeito da consistência da *string* de busca utilizada, uma vez que há a necessidade de que a *string* de busca reflita o objetivo central do estudo; caso contrário, seriam retornados, pelas fontes de busca, trabalhos fora da área de foco do trabalho. Para garantir a corretude e eficiência da *string* utilizada, foram realizadas

sucessivas buscas nas bibliotecas digitais. Como resultado destas buscas foi verificado que importantes trabalhos, trabalhos conhecidos pelos pesquisadores previamente dentro da área de estudo deste trabalho (S06 e S13), eram sempre retornados.

A segunda possível ameaça pode ter sido sobre a efetividade da qualidade do processo de busca da revisão sistemática. Esta ameaça foi identificada devido as bibliotecas digitais não retornarem trabalhos das edições do ano de 2015 das conferências SPLC e VaMoS, importantes fontes de trabalhos em LPSDs. Para solucionar este problema foi realizada uma busca manual nos *proceedings* das duas (2) conferências citadas.

A última ameaça é sobre a efetividade da condução da revisão sistemática devido à qualificação dos pesquisadores envolvidos. Dois (2) dos três (3) pesquisadores envolvidos não eram especialistas na área de LPSDs. Para resolver esta ameaça foi adotado duas (2) medidas. A primeira medida foi a participação de um especialista na área de LPSDs, atuando ativamente nas fases de planejamento e execução da revisão sistemática. A segunda medida foi a realização de uma análise de conformidade entre os pesquisados durante o segundo processo de seleção de trabalhos. Vale ressaltar que mesmo com o uso desta análise de conformidade, o especialista em LPSDs participou das discussões sobre as divergências de classificação de trabalhos entre os pesquisadores, com o intuito de garantir consistência e corretude das decisões a serem tomadas.

6.2 Trabalhos Futuros

Com a realização deste trabalho foi possível identificar algumas oportunidades de pesquisas ou mesmo direcionamentos de novos trabalhos na área de engenharia de domínio de LPSDs. Estas oportunidades são listadas a seguir.

- Estabelecer um processo formal de requisitos responsável por compreender toda a engenharia de requisitos e gerenciamento de variabilidade de LPSDs, dado que os processos de LPSs convencionais não suportam às particularidades que uma LPSD possui;
- Determinar abordagens para apoiar as lacunas em algumas atividades analisadas, como o processo de concepção e elicitação de requisitos em LPSDs.
- Abordagem para tratamento de Requisitos Não Funcionais (RNF) em LPSDs, dado que apesar do tratamento de Requisitos Funcionais (RF) durante a análise de domínio, os RNFs não recebem o mesmo tratamento, ou mesmo são tratados, até que se inicie a fase de projeto de domínio, fase posterior à análise de domínio;

- Mecanismo de verificação do modelo de variabilidade em tempo de execução, dada à necessidade de garantir a consistência e corretude da LPSD quando uma reconfiguração ocorre;
- Mecanismo para atualização automatizada do estado do modelo de variabilidade em tempo de execução quando uma reconfiguração ocorre, considerando a necessidade de se ter uma representação fiel do estado do modelo de variabilidade, e consequentemente, uma representação do estado da configuração em uso em um dado momento; e
- Garantia da Qualidade em LPSDs utilizando produtos de trabalhos além do modelo de *features*, devido ao fato dos trabalhos realizam esta atividade de garantia utilizando apenas o modelo de *features* como objeto de estudo.

6.3 Contribuições

Esta seção apresenta as principais contribuições identificadas com a realização deste trabalho.

- Conjunto de atividades, apoiadas por papéis, abordagens e ferramentas, para a execução da engenharia de requisitos e gerenciamento de variabilidade da engenharia de domínio de Linhas de Produto de Software Dinâmicas (LPSDs). Este conjunto de atividades pode servir como um guia de orientações ou catálogo de atividades, que são comumente utilizadas para a especificação do domínio de uma LPSD, sendo útil tanto para academia quanto para a indústria;
- Deficiências identificadas na realização de atividades. Estas deficiências representam as dificuldades encontradas no processo de especificação do domínio de LPSDs enfrentadas pelos responsáveis por este processo de especificação de projetos de LPSDs.
- Processo consistente para realização de revisão sistemática. O processo de realização da revisão sistemática, adotado neste trabalho, utilizou de orientações da literatura e trabalhos relacionados. Este processo pode ser utilizado para a execução de revisões sistemáticas, sendo útil como um guia de orientações, apoiado por técnicas em todo o processo de execução, indicação de apoio ferramental e templates manuais, quando necessário.

REFERÊNCIAS

ABBAS, N.; ANDERSON, J. Harnessing variability in product-lines of self-adaptive software systems. *In:* INTERNATIONAL CONFERENCE ON SOFTWARE PRODUCT LINE, 19., 2015, Nashville. **Anais…** Nashville: ACM, 2015. p. 191-200.

ABOWD, G. D.; DEY, A. K.; BROWN, P. J.; DAVIES, N.; SMITH, M.; STEGGLES, P. Towards a Better Understanding of Context and Context Awareness. *In:* GELLERSEN, H.W. **Handhled and Ubiquitous Computing.** Karlsruhe: Springer Berlin Heidelberg, 1999. p. 304-307.

ACHER, M.; COLLET, P.; LAHIRE, P.; MOISAN, S.; RIGAULT, J. P. Modeling variability from requirements to runtime. *In:* INTERNATIONAL CONFERENCE ON ENGINEERING OF COMPLEX COMPUTER SYSTEMS, 16., 2011, Las Vegas. **Anais…** Las Vegas: IEEE, 2011, p. 77-86.

ACHER, M.; COLLET, P.; FLEUREY, F.; LAHIRE, P. MOISAN, S.; RIGAULT, J. P. Modeling context and dynamic context with *feature* models. *In:* INTERNATIONAL WORKSHOP MODELS, 4., 2009, Denver. **Anais…** Denver: ACM/IEEE, 2009. p. 10.

ACHTERKAMP, M. C.; VOS, J. F. J. Investigating the use of the stakeholder notion in project management literature, a meta-analysis. **International Journal of Project Management**, v. 26, n. 7, p. 749-757, 2008.

ALAYA, M. B.; MATOUSSI, S.; MONTEIL, T.; DRIRA, K. Autonomic computing system for self-management of Machine-to-Machine networks. *In:* INTERNATIONAL CONFERENCE ON AUTONOMIC COMPUTING, 9., 2012, San Jose. **Anais...** New York: ACM, 2012, p. 25-30.

ALFÉREZ, G. H.; PELECHANO, V.; MAZO, R.; SALINESI, C.; DIAZ, D. Dynamic adaptation of services compositions with variability models. **Journal of Systems and Software**, v. 91, p. 24-47. 2014.

ALI, R.; DALPIAZ, F.; GIORGINI, P. A Goal Modeling Framework for Self-Contextualizable Software. **Enterprise, Business-Process and Information Systems Modeling Journal,** v. 29, p. 326-338, 2009.

ALVES, V.; NIU, N.; ALVES, A.; VALENÇA, G. Requirements engineering for software product lines: A systematic literature review. **Information and Software Technology**, v. 52, n. 8, p. 806-820, 2010.

BAHETI, R.; GILL, H. Cyber-physical systems. **The impact of control technology,** v. 12, p. 161-166. 2011.

BARBOSA, O; ALVES, C. A Systematic Mapping Study on Software Ecosystems. *In:* WORKSHOP ON SOFTWARE ECOSYSTEMS, 3., 2011, Brussels. **Anais…** Brussels: CEUR, 2011, p. 15-26.

BENCOMO, N.; HALLSTEINSEN, S.; SANTANA DE ALMEIDA, E.S. A View of the Dynamic Software Product Line Landscape. **Computer,** v. 45, n. 10, p. 36-41, 2012.

BERGER, T.; LETTNER, D.; RUBIN, J.; GRÜNBACHER, P.; SILVA, A.; BECKER, A; CHECHIK, M.; CZARNECKI, K. What is a *feature*? A qualitative study of *features* in industrial software product lines. *In:* SOFTWARE PRODUCT LINE CONFERENCE, 19., 2015, Nashville. **Anais...** Nashville: ACM, 2015. p. 16-25.

BJØRNER, Dines. Domain theory: practice and theories a discussion of possible research topics. *In:* THEORETICAL ASPECTS OF COMPUTING, 4., 2007. **Anais...** Macau: Springer Berlin Heidelberg, 2007. p. 1-17.

BÜRDEK; J.; LITY, S.; LOCHAU, M.; BERENS, M.; GOLTZ, U.; SCHÜRR, A. Staged configuration of dynamic software product line with complex binding time constraints. *In:* INTERNATIONAL WORKSHOP OF VARIABILITY MODELLING OF SOFTWARE-INTENSIVE SYSTEMS, 8., 2014, Nice. **Anais…** Nice: ACM, 2014, p. 16.

BOSCH, J. Software product lines: organizational alternatives. *In:* INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, 23., 2001, Toronto. **Anais...** Toronto: IEEE Computer Society, 2001. p. 91-100.

BOSCH, J. From Software Product Line to Software Ecosystem. *In:* INTERNATIONAL CONFERENCE ON SOFTWARE PRODUCT LINE, 13., 2009, San Francisco. **Anais…** San Francisco: ACM, 2009, p. 111-119.

BOSCH, J; CAPILLA, R; HILLIARD, R. Trends in systems and software variability. **IEEE Software**, n. 3, p. 44-51, 2015.

CALINESCU, R.; GRUNSKE, L.; KWIATKOWSKA, M.; MIRANDOLA, R.; TAMBURRELLI, G. Dynamic QoS Management and Optimization in Service-Based Systems. **Transactions on Software Engineering**, v. 37, n. 3, 2001, p. 387-409.

CAPILLA, R.; HINCHEY, M.; DÍAZ, F. J. Collaborative Context *Features* for Critical Systems. *In:* INTERNATIONAL WORKSHOP OF VARIABILITY MODELLING OF SOFTWARE-INTENSIVE SYSTEMS, 9., 2015, Hildesheim. **Anais…** Hildesheim: ACM, 2015. p. 43.

CAPILLA, R.; BOSCH, J.; TRINIDAD, P.; RUIZ-CORTÉZ, A.; HINCHEY, M. An overview of Dynamic Software Product Line architectures and techniques: Observations from research and industry. **Journal of Systems and Software**, v. 91, n.1, p. 3-23, 2014.

CAPILLA, R.; ORTIZ, O.; HINCHEY, M. Context Variability for Context-Aware Systems. **Computer**, v. 47, n. 2, p. 85-87, 2014.

CAPILLA, R.; BOSCH, J. The promise and challenge of runtime variability. **Computer**, v. 44, n. 12, p. 93-95, 2011.

CETINA, C.; GINER, P.; FONS, J.; PELECHANO, V. Prototyping Dynamic Software Product Lines to evaluate run-time reconfigurations. **Science of Computer Programming**, v. 78, n. 12, p. 2399-2413, 2013.

CZARNECKI, K.; BEDNASCH, T.; UNGER, P.; EISENECKER, U. Generative programming for embedded software: An industrial experience report. *In:* GENERATIVE PROGRAMMING AND COMPONENT ENGINEERING, 2002, Pittsburgh. **Anais...** Pittsburgh: Springer Berlin Heidelberg, 2002. p. 156-172.

CZARNECKI, Krzysztof, EISENECKER, Ulrich. **Generative Programming:** Methods, Tools, and Applications. 1. ed. Nova York: Addison-Wesley Professional, 2000. 864p.

DAMIAN, D. Stakeholders in global requirements engineering: Lessons learned from practice. **IEEE Software**, v. 24, n. 2, p. 21-27. 2007.

DE LEMOS, R. et al. Software Engineering for Self-Adaptive Systems: A Second Research Roadmap. In: DE LEMOS, R.; GIESE, H.; MÜLLER, H. A.; SHAW, M. **Software Engineering for Self-Adaptive Systems II.** Dagstuhl Castle: Springer Berlin Heidelberg, 2013. p. 1-32.

DEHLINGER, J.; LUTZ, R. R. Gaia-PL: a product line engineering approach for efficiently designing multiagent systems. **ACM Transactions on Software Engineering and Methodology**, v. 20, n. 4, p. 17, 2011.

DERMEVAL, D. TENÓRIO, T.; BITTENCOURT, I. I.; SILVA, A.; ISOTANI, S.; RIBEIRO, M. Ontology-based *feature* modeling: An empirical study in changing scenarios. **Expert Systems with Applications**, v. 42, n. 11, p. 4950-4964. 2015

DINKELAKER, T.; MITSCHKE, R.; FETZER, K.; MEZINI, M. A dynamic software product line approach using aspect models at runtime. *In:* WORKSHOP ON COMPOSITION AND VARIABILITY, 1., 2010, Rennes. **Anais…** Rennes: ACM, 2010.

DUBSLAFF, C.; BAIER, C.; KLÜPPELHOLZ, S. Probabilistic Model Checking for *Feature*-Oriented-Systems. *In:*_____. (1 Ed.). *Transaction on Aspect-Oriented Software Development XII.* Springer Berlin Heidelberg, 2015. p. 180-220.

ERIKSSON, M; BÖRSTLER, J; BORG, K. The PLUSS Approach-Domain Modeling with *Features*, Use Cases and Use Case Realizations. *In:* SOFTWARE PRODUCT LINE CONFERENCE, 9., 2005, Rennes. **Anais...** Rennes: Springer Berlin Heidelberg, 2005. p. 33-44.

FERNANDES, O.; WERNER, C.; TEIXEIRA, E. An Approach for *Feature* Modeling of Context-Aware Software Product Line. **J. UCS**, v. 17, n. 5, p. 807-829, 2011.

FERNANDES, P.; WERNER, C.; MURTA, L. G. P. *Feature* Modeling for Context-Aware Software Product Lines. *In:* SOFTWARE ENGINEERING AND KNOWLEDGE ENGINEERING, 20., 2008, San Francisco Bay. **Anais...** San Francisco Bay: Knowledge Systems Institute Graduate School, 2008. p. 3–8.

FEY, D.; FAJTA, R.; BOROS, A. *Feature* modeling: A meta-model to enhance usability and usefulness. *In:* SOFTWARE PRODUCT LINE CONFERENCE, 2., 2002, San Diego. **Anais...** San Diego: Springer Berlin Heidelberg, 2002. p. 198-216.

GUEDES, G.; SILVA, C.; SOARES, M,; CASTRO, J. Variability Management in Dynamic Software Product Lines: A systematic mapping. *In:* BRAZILIAN SYSMPOSIUM ON COMPONENTS, ARCHITECTURES AND REUSE SOFTWARE, 9., 2015, Belo Horizonte. **Anais…** Belo Horizonte: IEEE, 2015. p. 90-99.

GHEZZI, C.; SHARIFLOO, A. M. Dealing with Non-Functional Requirements for Adaptive Systems via Dynamic Software Product Lines. *In:*_____. *Software Engineering for Self-Adaptive Systems II*. Springer Berlin Heidelberg, 2013. p. 191-213.

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

HANSSEN, G. A longitudional case study of an emerging software ecosystem: Implications for practice and theory. **The Journal of Systems and Software**, v. 85, n. 7, p. 1455-1466, 2012.

HARSU, Maarit. **A Survey on Domain Engineering,** Tampere, 2002. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.197.7897&rep=rep1&type=pdf> Acesso em: 15 mai. 2015.

HEYMANS, P; TRIGAUX, J.C. **Software Product Lines: State of the art,** Namur, 2003. Disponível em: <http://www.inf.ufpr.br/silvia/topicos/artigostrab10/artigo1-S1e2.pdf>. Acesso em: 19 mai. 2015.

HINCHEY, M., SOOYONG P., SCHMID, K. Building Dynamic Software Product Lines. **Computer**, n. 10, p. 22-26, out. 2012.

HORN, P. Autonomic Computing: The IBM Perspective on the State Of Information Technology. **IBM.** 2001.

HUEBSCHER, M., MCCANN,J. A. A survey of autonomic computing __ degrees, models, and applications. **ACM Computing Surveys,** v. 40, n. 7, 2008.

ISTOAN, P.; NAIN, G.; PERROUIN, G.; JEZEQUEL, J. M. Dynamic software product lines for service-based systems. *In:* INTERNATIONAL CONFERENCE ON COMPUTER AND INFORMATION TECHNOLOGY, 9., 2009, Dhaka. **Anais...** Danka: IEEE Computer Society, 2009. p. 193-198.

JARKE, M. Requirements tracing. **Communications of the ACM**, v. 41, n. 12, p. 32-36. 1998.

JAROUCHEH, Z.; LIU, X.; SMITH, S. Mapping *features* to context information: Supporting context variability for context-aware pervasive applications. *In:* INTERNATIONAL CONFERENCE ON WEB INTELLINGENCE AND INTELLIGENT AGENT TECHNOLOGY, 2010a, Toronto. **Anais…** Toronto: IEEE, 2010. p. 611-614.

JAROUCHEH, Z.; LIU, X.; SMITH, S. CANDEL: product line based dynamic context management for pervasive applications. *In:* INTERNATIONAL CONFERENCE ON COMPLEX, INTELLIGENT AND SOFTWARE INTENSIVE SYSTEMS, 2010b, Krakow. **Anais…** Krakow: IEEE, 2010. p. 209-2016.

JEAN-BAPTISTE, L.; MARIA-TERESA, S.; JEAN-MARIE, G.; ANTOINE, B. Modeling dynamic adaptations using augumented *feature* models. *In:* SYMPOSIUM ON APPLIED COMPUTING, 28., 2013, Coimbra. **Anais…** Coimbra: ACM, 2013. p. 1734-1741.

KANG, K. C.; COHEN, S. G.; HESS, J. A.; NOVAK, W. E.; PETERSON, A. S. *Feature*-Oriented Domain Analysis (FODA) Feasibility Study, Carnegie-Mellon Univ. Pittsburgh Pa Software Engineering Inst., 1990. Disponível em: <http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=ADA235785> Acesso em: 15 mai. 2015.

KITCHENHAM, B.; BRERETON, O. P.; BUDGEN, D.; TURNER, M.; BAILEY, J.; LINKMAN, S. Systematic literature reviews in software engineering – A systematic literature review. **Information and Software Technology**, v. 51, n. 1, p. 7-15. 2009.

KITCHENHAM, B. Guidelines for performing systematic literature reviews in software engineering. **EBSE Technical Report,** v. 2.3, p. 1-57, 2007.

KITCHENHAM, B; MENDES, E; TRAVASSOS, G. H. A Systematic Review of Cross- vs. Within Company Cost Estimation Studies. *In:* INTERNATIONAL CONFERENCE ON EVALUATION AND ASSESSMENT IN SOFTWARE ENGINEERING, 10., 2006, United King. **Anais…** United King: British Computer Society, 2006.

KJELDSKOV, J.; SKOV, M. B. Supporting work activities in healthcare by mobile electronic patient records. *In:* ASIA PACIFIC CONFERENCE HUMAN INTERACTION, 6., 2004, New Zealand. **Anais…** New Zealand: Springer Berlin Heidelberg, 2004. p. 191-200.

KRAMER, D; SAUER, C; ROTH-BERGHOFER, T. Towards explanation generation using *feature* models in software product lines. *In:* KNOWLEDGE ENGINEERING AND SOFTWARE ENGINEERING, 9., 2013, Koblenz. **Anais…** Koblenz: WÜRZBURG UNIVERSITY, 2013. p. 13-23.

KUMARA, I.; HAN, J.; COLMAN, A.; NGUYEN, T.; KAPURUGE, M. Sharing with a Difference: realizing service-based SaaS applications with runtime sharing and variation in dynamic software product lines. *In:* INTERNATIONAL CONFERENCE ON SERVICES COMPUTING, 10., 2013, Santa Clara Marriot. **Anais…** Santa Clara Marriot: IEEE, 2013. p. 567-574.

LAPOUCHNIAN, A.; MYLOPOULOS, J. Modeling domain variability in requirements engineering with context. *In:* INTERNATIONAL CONFERENCE ON CONCEPTUAL MODELING, 28., 2009, Gramado. **Anais…** Gramado: Springer Berlin Heidelberg, 2009. p. 115-130.

LAUENROTH, K.; POHL, K. Dynamic consistency checking of domain requirements in product line engineering. *In:* INTERNATIONAL REQUIREMENTS ENGINEERING, 16., 2008, Catalonia. **Anais…** Catalonia: IEEE, 2008. p. 193-202.

LINS BARREIROS, H. S. **Análise da Completude dos Relatórios de Experimentos em Elasticidade na Computação em Nuvem**. 2015. 278 f. Tese (Mestrado em Ciência da

Computação) – Centro de Informática, Universidade Federal de Pernambuco, Pernambuco, 2015.

LISBOA, L. B.; LI, J. J.; MORREALE, P.; HEER, D.; WEISS, D. M. An evaluation to compare software product line decision model and *feature* model. In: EVALUATION OF NOVEL APPROACHES TO SOFTWARE ENGINEERING, 2014, Lisbon. **Anais…** Lisbon: IEEE, 2014. p. 1-8.

MARCÍAS-ESCRIVÁ, F. D. et al. Self-adaptive systems: A survey of current approaches, research challenges and application. **Expert Systems with Applications,** v. 40, 2013, p. 7267-7279.

MARINHO, F. G.; ANDRADE, R. M. C.; WERNER, C.; VIANA, W.; MAIA, M. E. F.; ROCHA, L. S.; TEIXEIRA, E.; FERREIRA FILHO, J. B.; DANTAS, L. L.; LIMA, F.; AGUIAR, S. MobiLine: A Nested Software Product Line for the domain of mobile and context-aware applications. **Science of Computer Programming**, v. 78, n. 12, p. 2381-2398, 2013.

MARINHO, F. G.; MAIA, P. H.; ANDRADE, R.; VIDAL, V. M.; COSTA, P. A.; WERNER, C. Safe adaptation in context-aware *feature* models. *In:* INTERNATIONAL WORKSHOP ON *FEATURE*-ORIENTED SOFTWARE DEVELOPMENT, 4., 2012, Dresden. **Anais…** Dresden: ACM, 2012. p. 54-61.

MARINHO, F. G.; ANDRADE, R.; WERNER, C. A Verification Mechanism of *Feature* Models for Mobile and Context-Aware Software Product Lines. *In:* BRAZILIAN SYSPOSIUM ON SOFWTARE COMPONENTS ARCHITECTURES AND REUSE, 2011, São Paulo. **Anais…** São Paulo: IEEE, 2011. p. 1-10.

MAZO, R.; MUÑOZ-FERNÁNDEZ, J. C.; RINCÓN, L.; SALINESI, C.; TAMURA, G. Variamos: an extensible tool for engineering (dynamic) product lines. *In:* INTERNATIONAL CONFERENCE ON SOFTWARE PRODUCT LINE, 19., 2015, Nashville. **Anais…** Nashville: ACM, 2015. p. 374-379.

MCGOVERN, James. **A Practical Guide to Enterprise Architecture.** 1 ed. Pearson Prentice-Hall, 2004. 306p.

MELLADO, D.; FERNÁNDEZ-MEDINA, E.; PIATTINI, M. Towards security requirements management for software product lines: A security domain requirements engineering process. **Computer Standards & Interfaces**, v. 30, n. 6, p. 361-371, 2008.

MENS, T.; CLAES, M.; GROSJEAN, P.; SEREBRENIK, A. Studying Evolving Software Ecosystems based on Ecological Models. In: MENS, T.; CLAES, M.; GROSJEAN, P.; SEREBRENIK, A. **Evolving Software Systems.** Springer Berlin Heidelberg, 2013. p. 297-326.

MIZOUNI, R.; MATAR, M. A.; AL MAHMOUD, Z.; ALZAHMI, S.; SALAH, A. A framework for context-aware self-adaptive mobile applications SPL. **Expert Systems with applications**, v. 41, n. 16, p. 7549-7564. 2014

MONTAGUD, S; ABRAHÃO, S; INSFRAN,E. A systematic review of quality attributes and measures for software product lines. **Software Quality Journal,** v. 20, n. 3-4, p. 425-486, 2012.

MORIN, B.; BARAIS, O.; JEZEQUEL, J. M.; FLEURY, F.; SOLBERG, A. Models@Run.time to Support Dynamic Adaptation. **Computer,** v. 42, n. 10, p. 44-51, 2009.

MUÑOZ-FERNÁNDEZ, J. C.; TAMURA, G.; RAICU, I.; MAZO, R.; SALINESI, C. REFAS: a PLE approach for simulation of self-adaptive systems requirements. *In:* INTERNATIONAL CONFERENCE ON SOFTWARE PRODUCT LINE, 19., 2015, Nashville. **Anais…** Nashville: ACM, 2015. p. 121-125.

MURGUZUR, A.; DE CARLOS, X.; TRUJILLO, S.; SAGARDUI, G. Context-aware staged configuration process variants @Runtime. *In:* INTERNATIONAL CONFERENCE ON ADVANCED INFORMATION SYSTEMS ENGINNERING, 26., 2014, Thessaloniki. **Anais…** Thessaloniki: Springer, 2014. p. 241-255.

NASCIMENTO, A.; RUBIRA, C. M. F.; CASTOR, F. ArCMAPE: A Software Product Line Infrastructure to Support Fault-Tolerant Composite Services. *In:* HIGH-ASSURANCE SYSTEMS ENGINEERING, 15., 2014, Miami. **Anais…** Miami: IEEE, 2014. p. 41-48.

NEIVA, D.; DE ALMEIDA, F. C.; DE ALMEIDA, E. S.; MEIRA, S. L. A requirements engineering process for software product lines. *In:* INFORMATION REUSE AND INTEGRATION, 2010, Las Vegas. **Anais…** Las Vegas: IEEE, 2010. p. 266-269.

NHANE, A. L. O., SONG, M.A.J. Self-optimization in Autonomic Computing Systems based on the Methodology of Bees Swarm Intelligence. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING RESEARCH AND PRACTICE, 14., 2014, Las Vegas. **Anais...** Athens: 2014, p. 1-7.

NORTHROP, L.; CLEMENTS, P.; BACHMANN, F.; BERGEY, J.; CHASTEK, G.; COHEN, S.; DONOHOE, P.; JONES, L.; KRUT, R.; LITTLE, R. **A Framework for Software Product Line Practice, Version 5.0**. SEI, 2007. Disponível em: <http://www.sei.cmu.edu/productlines/frame_report/index.html>. Acesso em: 16 mar. 2015.

NORTHROP, L. M. SEI's software product line tenets. **Software IEEE**, v. 19, n. 4, p. 32-40, 2002.

ORTIZ, O.; GARCIA, A. B.; CAPILLA, R.; BOSCH, J.; HINCHEY, M. Runtime Variability for dynamic reconfiguration in wireless sensor network product lines. *In:* SOFTWARE PRODUCT LINE CONFERENCE, 16., 2012, Salvador. **Anais…** Salvador: ACM, 2012. p. 143-150.

PARRA, C.; ROMERO, D.; MOSSER, S.; ROUVOY, R.; DUCHIEN, L.; SEINTURIER, L. Using constraint-based optimization and variability to support continuous self-adaptation. *In:* SYMPOSIUM ON APPLIED COMPUTING, 27., 2012, Riva. **Anais…** Riva: ACM, 2012. p. 486-491.

PARRA, C.; BLANC, X.; CLEVE, A.; DUCHIEN, L. Unifying design and runtime software adaptation using aspect models. **Science of Computer Programming**, v. 76, n. 12, p. 1247-1260, 2011.

PASCUAL, G. G.; PINTO, M.; FUENTES, L. Self-adaptation of mobile systems driven by the Common Variability Language. **Future Generation Computer Systems,** v. 47, 127-144, 2015.

POHL, Klaus. **Requirements engineering:** fundamentals, principles, and techniques. Springer Publishing Company, 2010. 814p.

POHL, Klaus; BÖCKLE, Günter; VAN DER LINDEN, Frank J. **Software Product Line Engineering:** Foundations, Principles and Techniques. Berlin, Alemanha: Springer Berlin Heidelberg, 2005. 467p.

PRESSMAN, Roger S. **Engenharia de software:** uma abordagem profissional. 7.ed. Porto Alegre: AMGH Ed., 2011. 780p.

ROSENMÜLLER, M.; SIEGMUND, N.; PUKALL, M.; APEL, S. Tailoring dynamic software product lines. *In:* ACM SIGPLAN, 2011. ACM: 2011. p. 3-12.

SABOURI, H; KHOSRAVI, R. Modeling and Verification of Reconfigurable Actor Families. **J. UCS**, v. 19, n. 2, p. 207-232, 2013.

SALEHIE, M.; TAHVILDARI, L. Self-adaptive software: Landscape and research challenges. **ACM Transactions on Autonomous and Adaptive Systems,** v. 4, n. 14, 2009.

SALLER, K; LOCHAU, M; REIMUND, I. Context-aware DSPLs: model-based runtime adaptation for resource-constrained systems. *In:* SOFTWARE PRODUCT LINES CONFERENCE, 17., 2013, Tokyo. **Anais:** Tokyo: ACM, 2013. p. 106-113.

SÁNCHEZ, L. E.; DIAZ-PACE, J. A.; ZUNINO, A.; MOISAN, S.; RIGAULT, J. P. An Approach for Managing Quality Attributes at Runtime using *Feature* Models. *In:* BRAZILIAN SYSPOSIUM ON SOFWTARE COMPONENTS ARCHITECTURES AND REUSE, 8., 2014, Maceió. **Anais…** Maceió: IEEE, 2014. p. 11-20.

SANTANA DE ALMEIDA, E. **RiDE-The RiSE Process for Domain Engineering**. 2007. 278 f. Tese (Doutorado em Ciência da Computação) – Centro de Informática, Universidade Federal de Pernambuco, Pernambuco, 2007.

SARINHO, V. T.; APOLINARIO, A. L.; DE ALMEIDA, E. S. OOFM-A *feature* modeling approach to implement MPLs and DSPLs. *In:* INTERNATIONAL CONFERENCE ON INFORMATION REUSE AND INTEGRATION, 13., 2012, Las Vegas. **Anais…** Las Vegas: IEEE, 2014. p. 740-742.

SHEN, L.; PENG,X.; ZHAO, W. Software Product Line Engineering for developing self-adaptive systems: Towards the domain requirements. *In:* COMPUTER SOFTWARE AND APPLICATIONS CONFERENCE, 36., 2012, Izmir. **Anais…** Izmir: ACM, 2012. p. 289-296.

SHEN, L.; PENG, X.; LIU, J.; ZHAO, W. Towards *feature*-oriented variability reconfiguration in dynamic software product lines. *In:* INTERNATIONAL CONFERENCE ON SOFTWARE REUSE, 12., 2011, Pohang. **Anais…** Pohang: Springer, 2011. p. 52-68.

SOFTEX. **MPS.BR - Melhoria de Processo do Software Brasileiro - Guia de Avaliação**. 2013. Disponível em: <http://www.softex.br/wp-content/uploads/2013/07/MPS.BR_Guia_de-Avaliacao_2013.pdf >. Acesso em: 14 jul. 2015.

SOMMERVILLE, Ian. **Engenharia de Software.** 9. ed. São Paulo, SP: Pearson Prentice Hall, 2011. 529p.

STANDISH GROUP INTERNATIONAL, INC. **CHAOS Manifesto 2013**. Disponível em: < https://www.versionone.com/assets/img/files/CHAOSManifesto2013.pdf>. Acesso em: 17 mar. 2015.

URLI, S.; BLAY-FORNARINO, M.; COLLET, P.; MOSSER, S.; RIVELLI, M. Managing a Software Ecosystem Using a Multiple Software Product Line: a Case Study on Digital Signage Systems. *In:* EUROMICRO CONFERENCE ON SOFTWARE ENGINEERING AND ADVANCED APPLICATIONS, 40., 2014, Verona. **Anais...** Verona: IEEE, 2014. p. 344-351.

VAN DER LINDEN, Frank J.; SCHMID, Klaus; ROMMES, Eelco. **Software Product Line in Action:** The Best Industrial Practice in Product Line Engineering. Secaunus, NJ, USA: Springer New York, 2007. 333p.

VAN GURP, J.; BOSCH, J.; SVAHNBERG, M. On the notion of variability in software product lines. *In:* SOFTWARE ARCHITECTURE, 2001, Amsterdam. **Anais…** Amsterdam: IEEE, 2001. p. 45-54.

VIERA, A. J.; GARRET, J. M. Understating interobserver agreement: the kappa static. **Family Medicine**, v. 54, n. 8, p. 360-363, 2005

VON DER MAßEN, T; LICHTER, H. Modeling Variability by UML Use Case Diagrams. *In:* INTERNATIONAL WORKSHOP ON REQUIREMENTS ENGINEERING FOR PRODUCT LINES, 2002, Essen. **Anais…** Essen**:** Avaya, 2002. p. 19-25.

W3C. XML Introduction. Disponível em: < http://www.w3schools.com/xml/xml_whatis.asp> Acesso em: 22 dez. 2015.

WHITE, S.; EDWARDS, M.. Domain engineering: The challenge, status, and trends. *In:* ENGINEERING OF COMPUTER-BASED SYSTEMS, 1996, Friedrichshafen. **Anais...** Friedrichshafen: IEEE, 1996. p. 96-101.

WHOLIN, C.; RUNESON, P.; HÖST, M.; OHLSSON, M. C.; REGNELL, B.; WESSLÉN, A. **Experimentation in Software Engineering.** Springer, 2012. 263p.

WOLFINGER, S.; REITER, S.; DHUNGANA, D.; GRÜNBACHER, P.; PRÄHOFER, H. Supporting runtime system adaptation through product line engineering and plug-in techniques. *In:* INTERNATIONAL CONFERENCE ON COMPOSITION-BASED SOFTWARE SYSTEMS, 7., 2008, Madrid. **Anais…** Madrid: IEEE, 2008. p. 21-30.

ZAMBONELLI, F; JENNINGS, N. R.; WOOLDRIDGE, M. Developing multiagents systems: The Gaya methodology. **TOSEM**, v. 12, n. 3, p. 317-370, 2003.

APÊNDICES

APÊNDICE A - Estudos Selecionados da Revisão Sistemática

Código	Referência do Trabalho	Origem	Avaliação da Qualidade
S01	(SHEN; PENG; ZHAO, 2012)	EI Compendex/Inspec	4,25
S02	(NASCIMENTO; RUBIRA; CASTOR, 2014)	EI Compendex/Inspec	3,25
S03	(DERMEVAL et al, 2015)	Scopus	3,75
S04	(DUBSLAFF; BAIER; KLÜPPELHOLZ, 2015)	Scopus	3,5
S05	(MIZOUNI et al., 2014)	Scopus	5,25
S06	(CAPILLA et al., 2014)	Scopus	4,5
S07	(ALFÉREZ et al., 2014)	Scopus	6
S08	(BÜRDEK et al., 2014)	Scopus	4,25
S09	(SÁNCHEZ et al., 2014)	Scopus	3,75
S10	(MURGUZUR et al., 2014)	Scopus	4,75
S11	(SALLER; LOCHAU; REIMUND, 2013)	Scopus	3,75
S12	(CETINA et al., 2013)	Scopus	4,25
S13	(MARINHO et al., 2013)	Scopus	4,25
S14	(KUMARA et al., 2013)	Scopus	3,25
S15	(GHEZZI; SHARIFLOO, 2013)	Scopus	5,25
S16	(JEAN-BAPTISTE et al., 2013)	Scopus	4,25
S17	(SABOURI; KHOSRAVI, 2013)	Scopus	3,25
S18	(KRAMER; SAUER; ROTH-BERGHOFER, 2013)	Scopus	4,25
S19	(SARINHO; APOLINARIO; DE ALMEIDA, 2012)	Scopus	2,25
S20	(MARINHO et al., 2012)	Scopus	4,25
S21	(PARRA et al., 2012)	Scopus	3
S22	(MARINHO; ANDRADE;WERNER, 2011)	Scopus	4,25

S23	(PARRA et al., 2011)	Scopus	3,75
S24	(ROSENMÜLLER et al., 2011)	Scopus	3,25
S25	(DEHLINGER; LUTZ, 2011)	Scopus	4
S26	(ACHER et al, 2011)	Scopus	4,5
S27	(SHEN et al., 2011)	Scopus	3,75
S28	(FERNANDES; WERNER; TEIXEIRA, 2011)	Scopus	4
S29	(JAROUCHEH; LIU; SMITH, 2010a)	Scopus	3,75
S30	(JAROUCHEH; LIU; SMITH, 2010b)	Scopus	3
S31	(DINKELAKER et al., 2010)	Scopus	4,5
S32	(ACHER et al., 2009)	Scopus	3,25
S33	(WOLFINGER et al., 2008)	Scopus	3
S34	(CAPILLA; HINCHEY; DÍAZ, 2015)	Busca Manual	3,5
S35	(MUÑOZ-FERNÁNDEZ et al., 2015)	Busca Manual	4,25
S36	(ABBAS; ANDERSON, 2015)	Busca Manual	2,25
S37	(MAZO et al., 2015)	Busca Manual	3,5

APÊNDICE B - Lista de Fontes identificadas

Fonte	SIGLA	Número de Estudos
ACM Transactions on Software Engineering and Methodology	-	1
International Conference on Aspect-Oriented Software Development	AOSD	1
Conference on Advanced Information Systems Engineering	CAiSE	1
International Conference on Complex, Intelligent and Software Intensive Systems	CISIS	1
Computer Software and Applications Conference	COMPSAC	1
Expert Systems with Applications	-	2
International Workshop on *Feature*-Oriented Software Development	FOSD	1
International Conference on Generative Programming and Component Engineering	GPCE	1
International Symposium on High-Assurance Systems Engineering	HASE	1
International Conference on Composition-Based Software Systems	ICCBSS	1
International Conference on Engineering of Complex Computer Systems	ICECCS	1
International Conference on Software Reuse	ICSR	1
International Seminar on Software Engineering for Self-Adaptive Systems II	-	1
International Conference on Information Reuse and Integration	IRI	1
Journal of Systems and Software	JSS	2
Journal of Universal Computer Science	JUCS	2
Knowledge Engineering and Software Engineering	KESE	1

International Conference on Model Driven Engineering Languages and Systems	MODELS	1
Symposium on Applied Computing	SAC	2
Brazilian Symposium on Software Components, Architectures and Reuse	SBCARS	2
International Conference on Services Computing	SCC	1
Science of Computer Programming	-	3
International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Systems	SNPD	1
Software Product Line Conference	SPLC	4
Workshop on Variability Modelling of Software-intensive Systems	VAMOS	2
International Conference on Web Intelligence	WI	1

APÊNDICE C - Protocolo de Revisão

Documento de Protocolo de Revisão Sistemática da Engenharia de Requisitos e Gerenciamento de Variabilidade na Engenharia de Domínio em Linhas de Produto de Software Dinâmicas

Introdução

Linhas de Produto de Software (LPS) representam um importante meio para promover o reuso de software; esta abordagem têm emergido como uma maneira diferenciada para promover o reuso, em que o reuso de software é planejado, implementado e gerenciado (NORTHROP, 2002). Uma LPS é entendida como a produção de um conjunto de produtos a partir de um mesmo conjunto de ativos base, propondo-se a atender às necessidades de um domínio específico (NORTHROP et al., 2007).

Uma vertente de LPS são as Linhas de Produto de Software Dinâmicas (LPSD). Em LPSDs, os produtos resultantes da linha sofrem alterações em seus modos de execução, influenciados pelo ambiente em que estes produtos executam, ou mesmo pelo usuário, com o intuito de continuar a atender às necessidades do domínio (HALLSTEINSEN, 2008).

As atividades do processo de criação de LPSs são apoiados por processos. Alguns desses processos podem ser vistos por meio dos trabalhos Neiva et al. (2010) e Mellado, Fernández-Medina e Piattini (2008). Entretanto, tais processos não podem ser utilizados para o processo de especificação do domínio que a LPSD atenderá dado que as particularidades, variabilidade dinâmica e capacidade de reconfiguração, não são suportadas por LPSs convencionais (HYNCHEY et al., 2012).

Esta revisão sistemática busca identificar na academia e indústria, trabalhos que apresentam técnicas, abordagens e/ou ferramentas que podem ser utilizadas para apoiar a engenharia de requisitos e o gerenciamento de variabilidade na engenharia de domínio em LPSDs.

Pesquisadores Envolvidos

A primeira etapa da seleção dos primeiros estudos, e o processo de extração das informações dos trabalhos serão feitas exclusivamente pelo pesquisador Léuson Mário Pedro da Silva. A segunda seleção de trabalhos será feita com o apoio de dois (2) pesquisadores, totalizando três (3) pesquisadores envolvidos.

Todo o processo de planejamento, execução e reporte dos resultados da revisão sistemática será feito sob a supervisão da orientadora Professora Carla Ilane Moreira Bezerra.

Questões de Pesquisa

Foi definido uma questão de pesquisa principal e a partir desta questão de pesquisa foi estabelecida questões de pesquisa secundárias. Todas as questões de pesquisa definidas são apresentadas a seguir:

1. RQ1: Como a engenharia de requisitos e o gerenciamento de variabilidade são executados na engenharia de domínio de LPSDs?

1.1. RQ1.1: Quais atividades da engenharia de requisitos e do gerenciamento de variabilidade são usadas em LPSDs?
1.2. RQ1.2: Quais abordagens são usadas para documentar os requisitos em LPSDs?
1.3. RQ1.3: Quais produtos de trabalhos da engenharia de requisitos e do gerenciamento de variabilidade são produzidos em LPSDs?
1.4. RQ1.4: Quais abordagens são usadas para representar a variabilidade de LPSDs?
1.5. RQ1.5: Quais abordagens são usadas para apoiar o processo de reconfiguração em LPSDs?
1.6. RQ1.6: Quais abordagens são usadas para eliminar possíveis inconsistências no modelo de variabilidade de LPSDs?

Processo de Busca
O processo de busca a ser utilizado trata-se de executar uma busca automática em específicas bibliotecas digitais. As bibliotecas digitais a serem utilizadas são: *Scopus*[5], *EI Compendex/Inspec*[6] e *Web of Science*[7]. Adicionalmente, foi estabelecido como formato de trabalhos válidos artigos advindos de conferências e *journals* (Revistas Acadêmicas).

Fontes de Busca
Com o intuito de garantir uma maior cobertura de trabalhos na área de estudo desta revisão sistemática, foi decidido que qualquer fonte de trabalho seria considerada válida, desde que um conjunto de trabalhos de controle também fosse retornado pelos resultados das buscas nas bibliotecas digitais.

Critérios de Inclusão
Considerando a necessidade classificar os trabalhos da revisão sistemática durante a fase de execução, foram definidos critérios responsáveis pela inclusão ou exclusão de um trabalho no escopo do trabalho.

Trabalhos nos seguintes tópicos serão incluídos:

1. Trabalhos sobre atividades da engenharia de requisitos para engenharia de domínio em LPSDs;
2. Trabalhos sobre abordagens para atividades da engenharia de requisitos em LPSDs;
3. Trabalhos sobre abordagens para tratamento da variabilidade de requisitos;
4. Trabalhos sobre abordagens para tratamento de inconsistências na configurações dos produtos da LPSD.

Critérios de Exclusão
Os seguintes tipos de trabalhos serão excluídos:

1. Trabalhos direcionados a outras atividades de LPSDs diferentes das atividades da engenharia de requisitos e/ou gerenciamento de variabilidade;

[5] www.scopus.com.br
[6] http://www.engineeringvillage.com/
[7] http://login.webofknowledge.com/

2. Trabalhos sobre características de LPSs;
3. Processo de Requisitos para LPSs ou outras áreas;
4. Trabalhos fora do escopo de linhas de produto;
5. Trabalhos duplicados;
6. Trabalhos não escritos em inglês;
7. Trabalho fora do formato de artigos.

Processo primário para Seleção de Estudos

O processo de seleção será realizado da seguinte forma:

1. Realização da busca nas bibliotecas digitais definidas utilizando a *string* de busca definida.
2. Análise do título, abstract e palavras-chave de todos os trabalhos resultantes da buca inicial com base nos critérios de inclusão e exclusão. Ao final, trabalhos irrelevantes serão eliminados.
3. Leitura completa dos trabalhos por meio da leitura em pares e análise da conformidade entre os pesquisadores envolvidos;
4. Processo de extração de informações dos trabalhos e avaliação da qualidade.

Avaliação da Conformidade

Com o objetivo de avaliar a conformidade pela qual os trabalhos serão classificados pelos pesquisadores envolvidos, será feita uma avaliação sobre este processo. A abordagem utilizada seguirá as orientações das Estatísticas de Kappa (VIERA; GARRET, 2005).

Avaliação da Qualidade

Para Avaliação da Qualidade da revisão realizada, foi estabelecido os seguintes critérios de qualidade:

1. As atividades utilizadas para especificação do domínio encontram-se definidas?
2. As atividades utilizadas apresentam descrições e/ou orientações de como devem ser realizadas?
3. Há a definição de produtos de trabalho de entrada e saída para as atividades?
4. Há a definição de papéis responsáveis pela execução e/ou participação das atividades?
5. As abordagens utilizadas pelas atividades encontram-se definidas?
6. As abordagens utilizadas apresentam descrições e/ou orientações de como devem ser realizadas?

Cada critério será avaliado através da associação de um valor numérico ao nível de completude do critério no trabalho. A representação de como essa valoração será realizada encontra-se representada no quadro a seguir.

Completude do Critério	Valoração associada
Totalmente satisfeito	1
Largamente satisfeito	0,75
Parcialmente satisfeito	0,5
Fracamente satisfeito	0,25
Não satisfeito	0

Coleta de Dados

A seguir é apresentado as informações a serem coletadas de cada trabalho avaliado.

- Título do Trabalho;
- Autores;
- Ano de publicação; caso o trabalho tenha sido publicado em diferentes fontes, todas os anos de publicação devem ser guardados, e em posterior análise, se necessário, deve-se usar a versão mais nova;
- Fonte do trabalho;
- Questão de pesquisa;
- Valor de avaliação de qualidade associada ao trabalho.

Análise de Dados

O processo para avaliação dos trabalhos selecionados será feito através da utilização de um formulário de extração de dados. Este formulário é formado por características advindas das Questões de Pesquisa definidas no início deste documento. O intuito do uso deste formulário é verificar como os trabalhos selecionados respondem a estas questões, e consequentemente, como eles ajudam na identificação de atividades do processo de requisitos em LPSD.

Disseminação

Os resultados encontrados com a execução desta revisão sistemática será reportado à Comunidade de Engenharia de Software e Pesquisadores na área, através da publicação deste trabalho.

APÊNDICE D – Formulário de Coleta de Dados

FORMULÁRIO PARA COLETA DE DADOS						
ENGENHARIA DE REQUISITOS E GERENCIAMENTO DE VARIABILIDADE NA ENGENHERIA DE DOMÍNIO DE LINHAS DE PRODUTO DE SOFTWARE DINÂMICAS						
Avaliador						
Título do Trabalho	Data da Avaliação	Resultado preliminar	Critérios preliminares	Revisor	Resultado Final	Critérios Finais

APÊNDICE E – Critérios de Qualidade avaliados

CRITÉRIOS PARA AVALIAÇÃO DOS CRITÉRIOS DE QUALIDADE											
ENGENHARIA DE REQUISITOS E GERENCIAMENTO DE VARIABILIDADE NA ENGENHERIA DE DOMÍNIO DE LINHAS DE PRODUTO DE SOFTWARE DINÂMICAS											
ID	Critério de Qualidade	Pontuação									
		Faixa de Avaliação válida									
QC1	As atividades do processo encontram-se definidas?	1	Totalmente	0,75	Largamente	0,5	Parcialmente	0,25	Fracamente	0	Não
QC2	As atividades do processo apresentam descrições e/ou orientações de como devem ser realizadas?	1	Totalmente	0,75	Largamente	0,5	Parcialmente	0,25	Fracamente	0	Não
QC3	Há a definição de produtos de trabalho de entrada e/ou de saída para as atividades?	1	Totalmente	0,75	Largamente	0,5	Parcialmente	0,25	Fracamente	0	Não
QC4	Há a definição de papéis responsáveis pela execução e/ou participação das atividades?	1	Totalmente	0,75	Largamente	0,5	Parcialmente	0,25	Fracamente	0	Não
QC5	As abordagens do processo encontram-se definidas?	1	Totalmente	0,75	Largamente	0,5	Parcialmente	0,25	Fracamente	0	Não
QC6	As abordagens do processo apresentam descrições e/ou orientações de como devem ser realizadas?	1	Totalmente	0,75	Largamente	0,5	Parcialmente	0,25	Fracamente	0	Não

APÊNDICE F – Formulário de Avaliação dos Critérios de Qualidade

FORMULÁRIO PARA AVALIAÇÃO DOS CRITÉRIOS DE QUALIDADE							
ENGENHARIA DE REQUISITOS E GERENCIAMENTO DE VARIABILIDADE NA ENGENHERIA DE DOMÍNIO DE LINHAS DE PRODUTO DE SOFTWARE DINÂMICAS							
Avaliador							
Título do Trabalho	**Critérios de Qualidade**						**Score**
	As atividades do processo encontram-se definidas?	As atividades do processo apresentam descrições e/ou orientações de como devem ser realizadas?	Há a definição de produtos de trabalho de entrada e/ou de saída para as atividades?	Há a definição de papéis responsáveis pela execução e/ou participação das atividades?	As abordagens do processo encontram-se definidas?	As abordagens do processo apresentam descrições e/ou orientações de como devem ser realizadas?	
							0
							0
							0
							0
							0
							0
							0

APÊNDICE G – Formulário de Avaliação da Conformidade

CÁLCULO - COEFICIENTE DE KAPPA				
ENGENHARIA DE REQUISITOS E GERENCIAMENTO DE VARIABILIDADE NA ENGENHERIA DE DOMÍNIO DE LINHAS DE PRODUTO DE SOFTWARE DINÂMICAS				
		Avaliador 1		
		Aceito	Não Aceitos	Total
Avaliador 2	Aceitos	0	0	0
	Não Aceitos	0	0	0
	Total	0	0	0
Valor Po				
Valor Pe				
Valor Kappa				
Avaliação Final				

APÊNDICE H – Formulário de Extração de Dados

FORMULÁRIO PARA A EXTRAÇÃO DE DADOS ENGENHARIA DE REQUISITOS E GERENCIAMENTO DE VARIABILIDADE NA ENGENHERIA DE DOMÍNIO DE LINHAS DE PRODUTO DE SOFTWARE DINÂMICAS				
Questão de Pesquisa Relacionada*		**Categoria de Avaliação**	**Critério Avaliado**	**Trabalho 01**
RQ01	RQ01.1	Atividades da Análise de Domínio	Concepção	☐
			Elicitação	☐
			Elaboração	☐
			Negociação	☐
			Especificação	☐
			Validação	☐
			Gerenciamento	☐
			Outras Atividades	☐
		Atividades da Análise de Contexto	Identificar propriedades físicas do sistema	☐
			Identificar *features* de Contexto	☐
			Modelar *features* de Contexto	☐
			Definir regras operacionais	☐
			Definir modos múltiplos de conexão	☐
			Outras Atividades	☐
		Papéis	Engenheiro de Domínio	☐
			Especialista do Domínio	☐
			Gerente de Projeto	☐
			Analista	☐
			Cliente	☐
			Projetista	☐
			Outros	
	RQ01.2	Abordagens para Especificação	Abordagem para especificação dos requisitos	
	RQ01.3	Produtos de Trabalho	Documento de Requisitos	☐
			Atividade Relacionada	
			Responsável	
			Documento de Especificação de Requisitos	☐
			Atividade Relacionada	
			Responsável	
			Modelo de *Features*	☐
			Atividade Relacionada	
			Responsável	
			Modelo de *Features* de Contexto	☐
			Atividade Relacionada	
			Responsável	
			Outros Documentos	
			Atividade Relacionada	
			Responsável	
	RQ01.4	Representação de Variabilidade	Modelo de Features	☐
			Modelo de Decisão	☐
			Outro	
		Disponibilidade das Partes	Estaticamente	☐
			Dinamicamente	☐
		Regras para composição	Abordagem para composição	
	RQ01.5	Representação da adaptabilidade	Abordagem utilizada	
	RQ01.6	Tratamento de Inconsistência	Tratamento de Modo Estátito	☐
			Tratamento de Modo Dinâmico	☐
		Abordagem Utilizada	Modelo de *Features*	☐
			Máquina de Estados	☐
			Outro	
Procedimento de Validação		Tipo de Validação	Validação Teórica	☐
			Validação Empírica	☐
			Não validado	☐
		Abordagem Utilizada	Casos de Estudo	☐
			Surveys	☐
			Experimentos controlados	☐
			Outras formas	
Suporte Ferramental			Uso de Ferramenta Livre	☐
			Uso de Ferramenta Paga	☐
			Não faz uso de Ferramenta	☐
			Nome(s) da(s) Ferramenta(s)	
Uso atual			Acadêmico	☐
			Industrial	☐