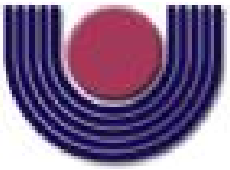

Unioeste - Universidade Estadual do Oeste do Paraná
CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
Colegiado de Ciência da Computação
Curso de Bacharelado em Ciência da Computação

Ferramentas de Gerenciamento de Variabilidade no Contexto da Migração de Sistemas Únicos em Linha de Produtos de *Software*: Replicando uma Revisão Sistemática

Leonardo R. Nardelli

CASCAVEL
2019

Leonardo R. Nardelli

Ferramentas de Gerenciamento de Variabilidade no Contexto da Migração de Sistemas Únicos em Linha de Produtos de *Software*: Replicando uma Revisão Sistemática

Monografia apresentada como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação, do Centro de Ciências Exatas e Tecnológicas da Universidade Estadual do Oeste do Paraná - Campus de Cascavel

Orientador: Ivonei Freitas da Silva

CASCAVEL
2019

Leonardo R. Nardelli

Ferramentas de Gerenciamento de Variabilidade no Contexto da Migração de Sistemas Únicos em Linha de Produtos de *Software*: Replicando uma Revisão Sistemática

Monografia apresentada como requisito parcial para obtenção do Título de Bacharel em Ciência da Computação, pela Universidade Estadual do Oeste do Paraná, Campus de Cascavel, aprovada pela Comissão formada pelos professores:

Ivonei Freitas da Silva (Orientador)
Colegiado de Ciência da Computação,
UNIOESTE

Victor Francisco Araya
Colegiado de Ciência da Computação,
UNIOESTE

Sidgley Camargo de Andrade
Colegiado de Tecnologia em Sistemas para
Internet, UTFPR

Cascavel, 21 de dezembro de 2019

Lista de Figuras

2.1 Interação entre três *features* (APEL et al., 2013). 7
2.2 Notação para *features* obrigatórias *features* (APEL et al., 2013). 8
2.3 Notação para a escolha exclusiva, operador *XOR* (APEL et al., 2013). 8
2.4 Notação para a escolha de uma ou mais *features* filhas, operador *OR* (APEL et al., 2013). 9
2.5 Processo de engenharia de linha de produtos de *software* (APEL et al., 2013). . 10
2.6 Abordagens baseadas em anotação e composição para implementação de uma linha de produtos (APEL et al., 2013). 12
2.7 Modelo de adoção extrativo de *SPL* (KRUEGER, 2002) 14

4.1 Porcentagem da presença de cada característica nas ferramentas 28
4.2 Porcentagem da presença de cada funcionalidade nas ferramentas 33

Lista de Tabelas

4.1 Número de estudos por ano 21
4.2 Ferramentas e suas características principais 27
4.3 Funcionalidades das ferramentas 31

Lista de Abreviaturas e Siglas

SPL	Software Product Line
SLR	Systematic Literature Review

Sumário

Lista de Figuras **iv**

Lista de Tabelas **v**

Lista de Abreviaturas e Siglas **vi**

Sumário **vii**

Resumo **ix**

1 Introdução **1**

1.1 Contexto 1

1.2 Objetivo 3

1.3 Contribuições 3

1.4 Estrutura do Trabalho 3

2 Introdução à Linha de Produto de *Software* **4**

2.1 Promessas das Linhas de Produtos de *Software* 5

2.2 *Feature* 6

2.3 Abordagem Orientada a *Features* 6

2.4 Interações Entre *Features* 6

2.5 Diagrama de *Features* 7

2.6 Processo Para o Desenvolvimento de Linha de Produtos de *Software* 9

2.7 Abordagens para Implementação de Linha de Produtos de *Software* 11

2.7.1 Abordagem Baseada em Anotações 11

2.7.2 Abordagem Baseada em Composição 12

2.8 Sistema Legado e Sistema Único 13

2.9 Migração de Sistemas Únicos para Linha de Produtos de *Software* 13

3 Planejamento da Revisão Sistemática **16**

3.1 Necessidade da Pesquisa 16

3.2 Definição das Questões de Pesquisa 16

3.3 Definindo a *String* de Busca 17

3.4 Seleção das Fontes de Busca 18

3.5 Definição dos Estudos 18

3.6 Extração de Dados e Avaliação de Qualidade dos Estudos 19

4 Resultados e Análises **21**

4.1 Coleta dos Estudos 21

4.2 Principais Características das Ferramentas 26

4.3 Funcionalidades das Ferramentas 28

4.4 Aplicação das Ferramentas para a Migração 32

4.5 Ameaças à Validade da Revisão 33

5 Conclusão e Trabalhos Futuros **35**

5.1 Trabalhos Futuros 36

Referências Bibliográficas **37**

Resumo

O gerenciamento da linha de produtos de *software* (*SPL*) é uma atividade fundamental para a engenharia de linhas de produtos de *software*. A ideia por trás do gerenciamento de *SPL* é focar em *features* compartilhadas para suportar a reutilização e adaptação de código. Ganhos são esperados em termos de tempo de lançamento no mercado, consistência entre produtos, redução de custos, melhor flexibilidade e gerenciamento de requisitos. Neste contexto, existem opções de ferramentas de gerenciamento de variabilidade disponíveis para *SPL*, porém, pouco se é discutido sobre o suporte que estas ferramentas oferecem no processo de migração de sistemas únicos para linha de produtos de *software*. Este trabalho realiza uma revisão sistemática da literatura (*SLR*) de ferramentas de gerenciamento de *SPL* para avaliar suas aplicações no processo de migração de sistemas únicos em produtos pertencentes a uma *SPL*. Por meio dos resultados obtidos, a documentação limitada, a complexidade e a indisponibilidade das ferramentas são alguns dos motivos que podem acabar desencorajando a adoção e o amplo uso dessas ferramentas nas organizações e na academia. Além disso, ainda existem lacunas no suporte completo ao processo de gerenciamento em todas as ferramentas investigadas, principalmente em funcionalidades de planejamento que auxiliariam no processo de migração.

Palavras-chave: Ferramentas, Gerenciamento, Linhas de Produto de Software, Migração, Revisão Sistemática, Sistemas Únicos.

Capítulo 1

Introdução

1.1 Contexto

Ao desenvolver *software*, independentemente da metodologia utilizada, existem vários passos a serem dados; alguns destes podem não ser tão triviais quanto outros. A dificuldade pode ser facilmente escalada de acordo com o tamanho do projeto: projetos simples não oferecem tanta dificuldade quanto projetos mais complexos e extensos.

A complexidade tende a crescer exponencialmente quando, ao invés de apenas um *software*, tratamos de uma família de produtos, como acontece em uma linha de produtos de *Software* (KANG; SUGUMARAN; PARK, 2009), onde existem muitas variantes a partir de um único conceito de aplicação.

Uma linha de produtos é uma família organizada para tirar proveito de seus aspectos comuns, além de prover variações para cada produto em particular (WEISS, 2009). Tratando-se de *software*, possuímos as linhas de produtos de *software* (*SPL*). Para além dos processos comumente adotados ao se desenvolver um *software*, em *SPL* existem processos análogos, não levando em conta os produtos e suas particularidades, mas sim o domínio de atuação de todos eles.

A engenharia de uma *SPL* pode apresentar muitas barreiras, algumas das principais são (BASTOS et al., 2011): (i) o custo inicial associado na maioria das vezes é elevado; (ii) o tempo necessário para o desenvolvimento é maior que no desenvolvimento de *software* tradicional; (iii) a ausência de profissionais peritos e o alto custo para treinamento; (iv) a falta de ferramentas que auxiliem no gerenciamento da grande quantidade de informações, relacionamentos e dependências entre as *features*.

Um cenário comum na indústria de *software* é o desenvolvimento de sistemas de maneira única, sem considerar um domínio de aplicação. Neste cenário, normalmente desenvolve-se um único sistema, e com o passar do tempo, novas versões do sistema são construídas. Este é um cenário onde não se adota uma engenharia de linha de produtos de *software* por falta de conhecimento da técnica ou pelas barreiras mencionadas anteriormente.

Contudo, com o passar do tempo, mesmo as empresas no cenário de desenvolvimento de sistema único começam a criar versões ou mesmo novos produtos a partir do primeiro sistema desenvolvido. Embora não tenham sido organizados como uma *SPL*, esses produtos tem muitas similaridade entre si, o que promove a necessidade de um gerenciamento dessas semelhanças e, principalmente, das variabilidades existentes entres os produtos.

Essas empresas ao entrarem em contato com as técnicas para linhas de produtos de *software*, identificam a necessidade de migrar seus sistemas para *SPL*. Porém, as adversidades impostas para essa conversão muitas vezes impossibilitam tal migração. Para tais casos, ferramentas que auxiliem em cada etapa do processo de uma *SPL* é essencial tanto na construção de uma nova linha de produtos, a partir de requisitos, quanto na reengenharia de um sistema único para uma *SPL* (BASTOS et al., 2011).

Neste cenário, surge uma questão para as organizações que desejam migrar seus produtos desenvolvidos individualmente para uma estrutura de linha de produtos de *software*: Quais são as ferramentas disponíveis (industrialmente e academicamente) para apoiar a adoção de um processo de *SPL*? Um estudo "*A Systematic Literature Review of Software Product Line Management Tools*" (PEREIRA; CONSTANTINO; FIGUEIREDO, 2014), o qual foi executado em 2014 detectou lacunas, como a falta de suporte industrial durante a configuração das ferramentas. Contudo, este trabalho não focou na migração de sistemas únicos para *SPL*.

Este trabalho de conclusão de curso atualiza essa revisão sistemática, uma vez que as ferramentas mencionadas naquele estudo podem ter sido atualizadas como também novas ferramentas podem ter surgido. Além disso, será feita uma investigação dentro do grupo de estudos finais selecionados, para averiguar quais suportam, de alguma forma, algum tipo de migração, refatoração, remodelagem ou até mesmo adaptação de um sistema único para uma *SPL*.

1.2 Objetivo

O objetivo deste trabalho é re-executar a revisão sistemática de literatura realizada em "*A Systematic Literature Review of Software Product Line Management Tools*" (PEREIRA; CONSTANTINO; FIGUEIREDO, 2014) afim de atualizar a análise sobre as ferramentas para gerenciamento de *SPL*, para então, averiguar através das funcionalidades destas ferramentas, se elas suportam, mesmo que indiretamente, um processo de migração de sistema único para *SPL*. Os resultados foram extraídos da busca de evidências em periódicos e anais de congressos desde 2000.

1.3 Contribuições

Os resultados deste trabalho visam contribuir com informação relevante para: (i) Dar suporte na escolha apropriada de ferramentas de *SPL* que auxiliem no processo de migração de um ou mais sistemas únicos em uma *SPL*; (ii) Apontar atributos e requisitos relevantes para aqueles com interesse no desenvolvimento de novas ferramentas, tanto no contexto de migração como de gerenciamento de *SPL* em geral; (iii) Auxiliar com a melhoria e expansão de ferramentas de *SPL* já existentes.

1.4 Estrutura do Trabalho

Este trabalho está dividido nos seguintes capítulos:

- **Capítulo 2:** Introdução à linha de produto de *software*;
- **Capítulo 3:** Planejamento da revisão sistemática;
- **Capítulo 4:** Resultados e análises;
- **Capítulo 5:** Conclusão e trabalhos futuros.

Capítulo 2

Introdução à Linha de Produto de *Software*

A crescente necessidade pelo desenvolvimento de softwares mais robustos e complexos requer um melhor suporte para o reuso de características que constituem um *software* (POHL; BöCKLE; LINDEN, 2005). Para resolver essa demanda, linha de produto de *software* (*SPL*) tem sido largamente adotada pela indústria de *software* (CLEMENTS; NORTHROP, 2001), (LINDEN; SCHMID; ROMMES, 2007). *SPL* é um grupo de *softwares* que compartilham um conjunto de características similares e variáveis se adaptando a necessidade particular de cada segmento do mercado (POHL; BöCKLE; LINDEN, 2005). O *software* é construído em volta desse conjunto de características que permitem em certos pontos, diferentes configurações (CLEMENTS; NORTHROP, 2001), (LINDEN; SCHMID; ROMMES, 2007). *SPL*, portanto, trás avanço significativo no processo de desenvolvimento de *software* (POHL; BöCKLE; LINDEN, 2005), (SANTOS et al., 2013).

Um Importante conceito sobre uma *SPL* é o modelo de *features*. Os modelos de *features* são usados para representar o que é comum e variável em uma *SPL*. Um *feature model* pode se referir a requisitos funcionais ou não, decisões de arquitetura, ou padrões de projeto. O real potencial das *SPL* é atingido através da arquitetura projetada que aumente o reuso de características em vários produtos.

Na prática, o desenvolvimento de uma *SPL* envolve a modelagem das características para representar diferentes pontos de vista, subsistemas e detalhes dos produtos (BEUCHE; SCHRöDER-PREIKSCHAT; PAPAJEWSKI, 2004). A partir desse motivo se cria a necessidade de uma ferramenta que dê suporte a empresas durante o gerenciamento de variabilidade da

SPL. Ferramentas de suporte proveem para as empresas um meio, pois precisam conhecer para operar, para o desenvolvimento de *SPL*, como também, o desenvolvimento e manutenção do ambiente da *SPL*. Porém, a escolha de uma ferramenta que melhor se adapte às necessidades da empresa é longe de ser trivial. Em particular, essa escolha é crítica devido ao crescente aumento do número de ferramentas de gerenciamento de *SPL* disponibilizadas nos últimos anos. Além disso, ferramentas de suporte deveriam auxiliar processo completo do desenvolvimento de *SPL*, e não apenas em algumas atividades (PEREIRA; CONSTANTINO; FIGUEIREDO, 2014).

2.1 Promessas das Linhas de Produtos de *Software*

Há inúmeras vantagens na escolha de uma *SPL* em comparação ao desenvolvimento individual de produtos de *software*, as que mais se destacam são as seguintes (APEL et al., 2013):

- **Sob medida:** Facilita uma adaptação mais precisa dos produtos para necessidades exclusivas de cada cliente.

- **Qualidade:** A indústria perante a sua produção em massa, contribui com resultado positivo na qualidade no desenvolvimento dos produtos, sendo comprovado pelas partes padronizadas que são utilizadas e testadas em múltiplos produtos. No entanto, mesmo que as combinações de todos as partes reusáveis não sejam utilizadas, aquelas que têm o uso habitual garantem produtos de *software* seguros e com estabilidade.

- **Redução de custos:** O *design* e o desenvolvimento não precisam partir do zero para atender a solução desejada. Visto que, com a combinação das partes reutilizáveis podem ser desenvolvidas inúmeras maneiras de moldar diferentes produtos. Então neste caso, o custo do produto final está baseado na escolha das partes, previamente construídas, que serão adicionadas. No entanto, neste processo não se pode esquecer que inicialmente, existe um custo agregado no desenvolvimento de módulos que sejam genéricos de forma a serem reutilizáveis em inúmeros produtos, é superior ao do desenvolvimento individual de *software*.

- **Comercialização:** Produtos que são desenvolvidos isoladamente requerem além do custo, um certo tempo significativo para seu desenvolvimento. Diferente de *software*

que fazem parte de uma linha de produtos, que já estão prontos dependendo apenas do processo da escolha e da conexão dos partes pré-fabricadas.

2.2 *Feature*

Uma *feature* representa uma característica ou um comportamento de um sistema de *software* (APEL et al., 2013). Além disso, as *features* especificam semelhanças e diferenças dos produtos entre *stakeholders* (público estratégico). Como também em todos os ciclos de vida do *software*, orientam a estrutura, o reuso e as variações de componentes. Os produtos encapsulam essas *features* que também são usadas para distinguir produtos de uma linha (APEL et al., 2013).

2.3 Abordagem Orientada a *Features*

Com Linha de produtos de *software* o desenvolvimento faz uso de partes reutilizáveis para constituir um novo produto. Baseado neste aspecto, é visto que *SPLs* promovem a indústria do desenvolvimento, com um conjunto de peças que podem gerar produtos com pré-requisitos específicos para cada cliente. Portanto, o reuso é uma estratégia importante para agilizar o processo de desenvolvimento, e *features* são a base para atingir esse nível de automação. No entanto, não basta só reusar, o conceito principal das *features* é diminuir a distância dos requisitos entre usuário e desenvolvedor, em termos de funcionalidades. Em suma, os usuários reportam os problemas para suas necessidades, o desenvolvedor interpreta, organiza, estrutura e transforma em ações aplicando no processo de linha de produção, assim como em todos os *features* envolvidos. Sumarizando tecnicamente, a abordagem orientada a *features* e aponta as *features* por todo o ciclo da linha de produção.

2.4 Interações Entre *Features*

A interatividade diz respeito ao fato de as *features* não atuarem de forma isolada ou independente e podem ocorrer de diversas formas no desenvolvimento da *SPL*. As interações podem ser planejadas e positivas, ou de forma crítica e impensada, com produtos sem similaridade, que podem gerar irregularidades e resultar em estados críticos do sistema. Portanto, na orientação a *features* é necessário que tais interações estejam explícitas no *design* e código, tanto através

da anotação de trechos de código relacionados a cada funcionalidade, quanto pela separação e modularização dos mesmos. No processo de modelagem, espera-se que *features* interajam, troquem informações e apurem o desempenho de outras *features* com o reuso das funcionalidades, para uma conclusão final cooperativa. Assim a gestão e especificação de interações planejadas em conjunto a detecção e resolução das não planejadas é um dos maiores desafios no desenvolvimento de *SPL* dirigido à *features*. A Figura 2.1 demonstra a interatividade entre *features*.

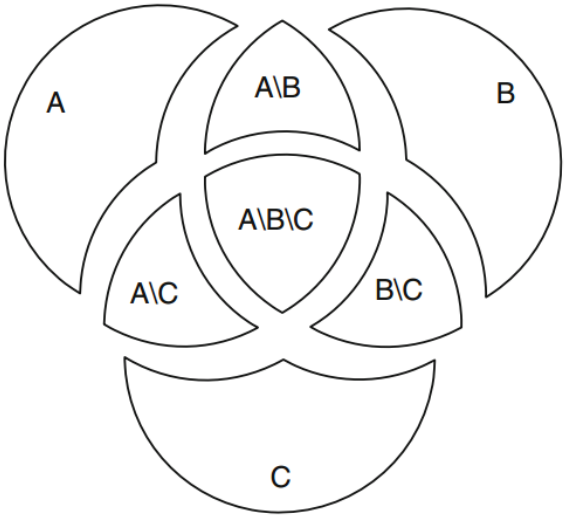

Figura 2.1: Interação entre três *features* (APEL et al., 2013).

2.5 Diagrama de *Features*

No diagrama de *features* é possível descrever as *features* e itens adicionais, é uma notação visual específica de um modelo de *features*. Ele consiste em uma árvore de nós com informações, pontos de variabilidade, prioridades e regras de dependência entre as *features*.

Um fator importante é que *features* podem estar relacionadas, e esse tipo de diagrama inclui relações mútuas. Isto é, quando uma *feature* **f** representa uma especialização vinda da *feature* **p**, que representa um conceito mais geral, neste caso, para o uso da **f** é obrigatório também o uso da *feature* **p**.

Além das *features* obrigatórias, que devem estar presentes em todos os membros da *SPL*, têm as opcionais, sua presença atribuído ao seu significado é opcional nos membros da *SPL*.

Portanto as *features* são diferenciadas com círculo presente no nó **f**, quando vazio indica uma *feature* opcional, quando preenchido indica uma obrigatoriedade (Figura 2.2).

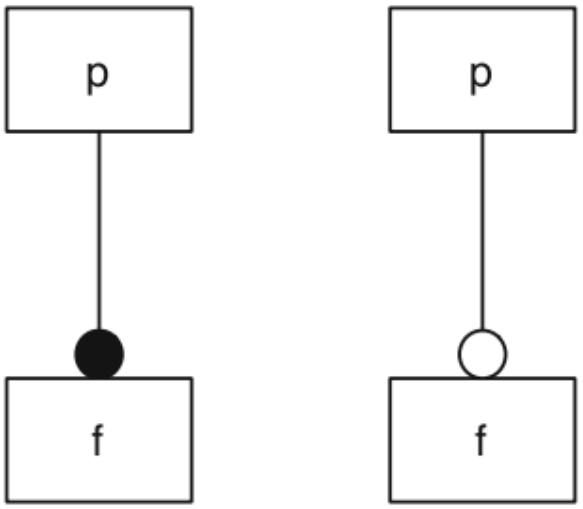

Figura 2.2: Notação para *features* obrigatórias *features* (APEL et al., 2013).

A Figura 2.3 apresenta uma *feature* pai (**p**) com um grupo de *features* filhas (**f**) ligados por um arco vazio. Essa representação indica a escolha de uma apenas uma das *features* filhas.

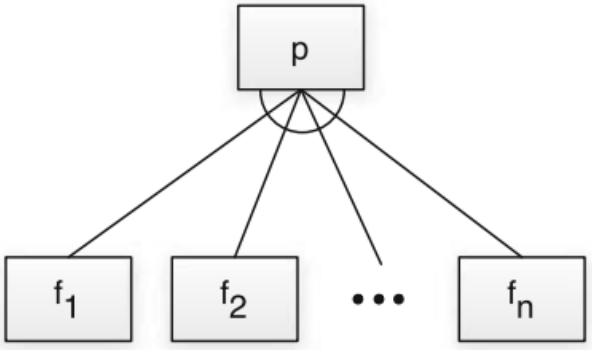

Figura 2.3: Notação para a escolha exclusiva, operador *XOR* (APEL et al., 2013).

Já a Figura 2.4 exibe um grupo de *features* filhas ligadas a uma *feature* pai por um arco preenchido. Essa convenção indica a escolha de uma ou mais filhas.

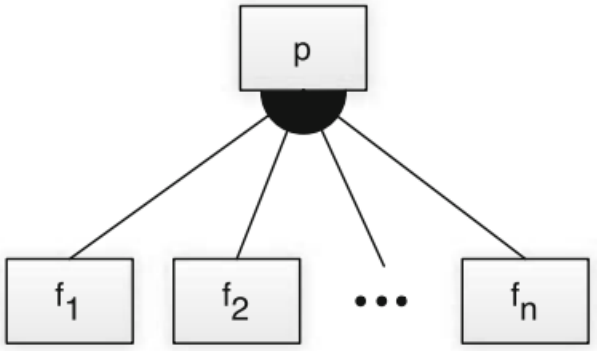

Figura 2.4: Notação para a escolha de uma ou mais *features* filhas, operador *OR* (APEL et al., 2013).

2.6 Processo Para o Desenvolvimento de Linha de Produtos de *Software*

Em Geral, a engenharia tradicional de *software* foca apenas em um produto a ser desenvolvido não levando em consideração futuros produtos semelhantes que possam vir a surgir futuramente. Já em uma perspectiva de *SPL*, existe a preocupação do desenvolvimento de produtos não idênticos mas semelhantes em inúmeros aspectos. Portanto o sucesso do desenvolvimento de uma *SPL* se baseia na boa estruturação do domínio que os produtos pertencem. Uma caracterização comum de *SPLs* é a separação entre a engenharia de domínio, engenharia de aplicação, espaço do problema e espaço de solução.

A Figura 2.5 mostra o processo de engenharia para o desenvolvimento de linha de produtos de *software* proposto por (APEL et al., 2013).

- **Engenharia de Domínio**: Nesta fase, é realizada a análise do domínio da linha de produtos, desenvolvendo *features* para reuso, preparando para que possam ser usados em múltiplos produtos e não apenas em um produto de *software* específico. Neste caso, este processo é responsável por estabelecer um conjunto de *features* reutilizáveis.

- **Engenharia de Aplicação**: este processo é responsável no desenvolvimento de um produto específico para suprir a necessidade de um cliente em particular. É o processo de desenvolvimento do *software* a partir do uso ou não da engenharia de domínio reaproveitando as *features*. A engenharia de aplicação é repetida para cada produto da linha, ela explora a variabilidade desenvolvida das *features* que concordam com as necessidades

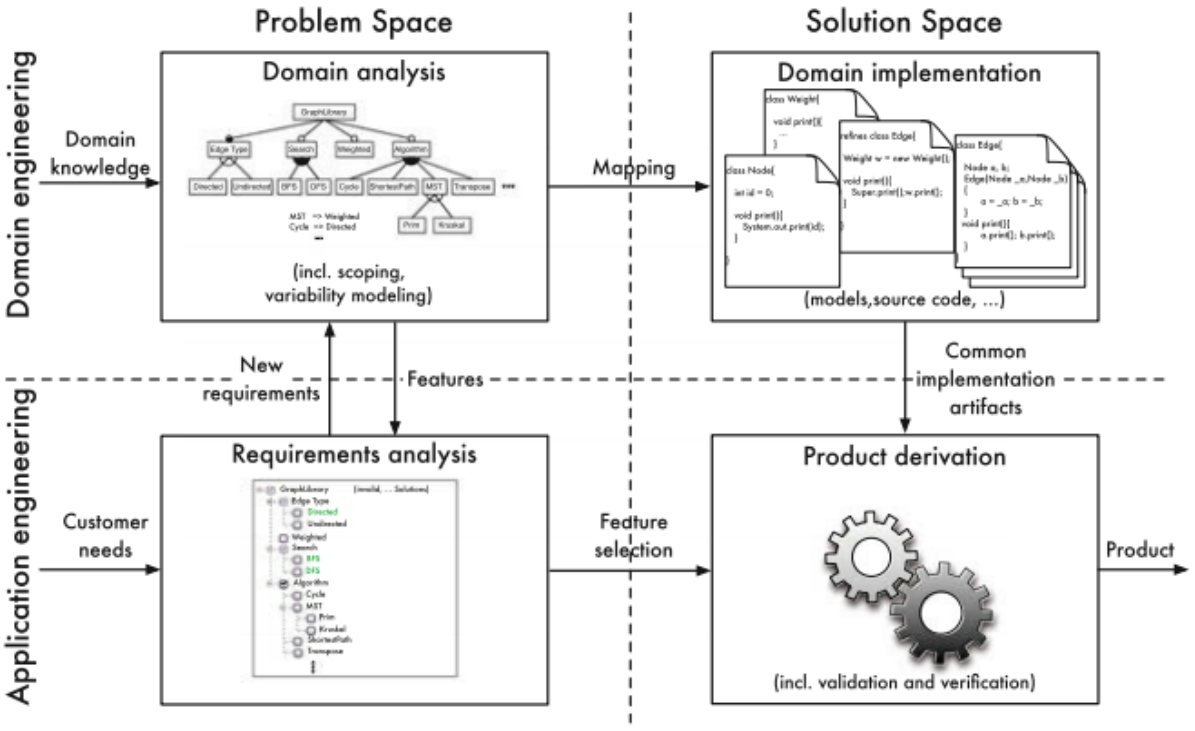

Figura 2.5: Processo de engenharia de linha de produtos de *software* (APEL et al., 2013).

específicas de um produto final.

- **Espaço do Problema**: O objetivo é assumir a perspectiva dos interessados, seus problemas e necessidades em relação ao domínio como um todo e também a produtos de forma individual.

- **Espaço da Solução**: já no caso de solução o espaço representa as perspectivas não do cliente, mas do fornecedor e desenvolvedor de *software*. O espaço de solução abrange a concepção, implementação, validação e verificação de *features* e combinações adequadas, que são muito importantes para facilitar o reuso.

Quatro grupos de tarefas são geradas na linha de desenvolvimento dos produtos a partir das diferenças entre as duas engenharias (domínio e aplicação) e os dois espaços (problema e solução):

1. **Análise de domínio:** este modelo da engenharia faz análise de domínio nas *features* de requisitos. Neste caso, define quais produtos devem ser englobados pela linha de produtos e, quais *features* são relevantes e devem ser implementadas como *features* reutilizáveis.

2. **Análise de requisitos:** a análise de requisitos é necessária para gerar uma especificação adequada para a *SPL*. Nesta sequência, verifica-se às necessidades específicas do cliente, em seguida estes requisitos são projetados para uma escolha de *features* com base nas *features* já identificadas durante a análise de domínio. Caso novos requisitos surjam, pode ocorrer um retorno à análise de domínio, que resultará em prováveis modificações no modelo de *features*.

3. **Implementação do domínio:** Nessa fase ocorre o processo de desenvolvimento de *features* para reuso com potencial às *features* identificadas na análise de domínio.

4. **Derivação de produtos:** Nessa etapa ocorre uma montagem, a partir de uma configuração de *features* reutilizáveis, gerando novos produto de *software*. Ocorre a produção com as *features* reutilizáveis, no qual foram ajustados de acordo com as conclusões da análise de requisitos. Dependendo da implementação, esse processo pode ser mais ou menos automatizado, possivelmente envolvendo várias tarefas de desenvolvimento e customização.

2.7 Abordagens para Implementação de Linha de Produtos de *Software*

De modo a facilitar o entendimento, o principal alvo da engenharia de linha de produtos orientada a *features* é a geração de produtos a partir do código variável, de acordo com modelos criados na engenharia de domínio com as *features* selecionadas pelo usuário. Na implementação da *SPL*, as abordagens podem diferir tanto na representação da variabilidade em códigos como na geração de produtos (APEL et al., 2013). Duas abordagens muito utilizadas na implementação de *SPL* são descritas abaixo:

2.7.1 Abordagem Baseada em Anotações

Na abordagem baseada em anotações, o controle da escolha das *features* é feito por trechos de códigos anotados. As anotações são realizadas em um código base que contém a implementação das funcionalidades da *SPL*, de tal forma que estes trechos implementem e demarquem adequadamente diferentes *feature*. Toda *feature* dentro do código base que não é escolhida ou gera uma combinação inválida de *features* é removida na compilação ou ignorado na execu-

ção. Esta abordagem tem a vantagem na facilidade de uso graças ao suporte nativo presente em diversos ambientes de desenvolvimento.

2.7.2 Abordagem Baseada em Composição

O desenvolvimento das *features* é realizada através de unidades compostas, em que em um cenário ideal, uma *feature* é composta por uma única unidade. Durante a derivação dos produtos, todas as unidades selecionadas e combinadas são compostas para a geração do produto final. O desafio é manter o mapeamento entre *features* e unidades de composição simples e maleável, relacionando de um para um. Como mencionado anteriormente, em um cenário ideal em que cada *feature* possui sua própria implementação sob a forma de uma unidade de composição, um gerador precisa simplesmente adicionar na composição a unidade referente a *feature* que é desejada.

As diferenças entre as duas abordagens (anotação e composição), está no fato que uma suporta variabilidade negativa e outra positiva. A abordagem baseada em anotação (negativa) os trechos de código podem ser removidos quando necessário, enquanto a abordagem baseada em composição (positiva) permitem que unidades da composição sejam adicionadas sob demanda. e A Figura 2.6 ilustra as diferenças entre as duas abordagens na implementação de uma *SPL*.

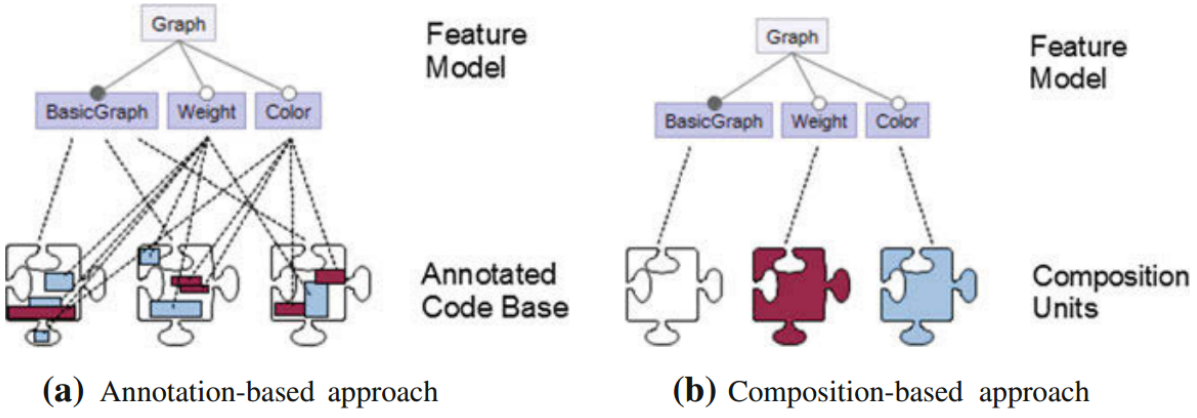

Figura 2.6: Abordagens baseadas em anotação e composição para implementação de uma linha de produtos (APEL et al., 2013).

2.8 Sistema Legado e Sistema Único

Sistemas legados podem ser definidos como ”grandes sistemas de *software* os quais não se sabe como lidar, mas são vitais para uma organização” (BENNETT, 1995) ou ”qualquer sistema de informação que resiste significativamente à modificação e à evolução” (STONEBRAKER; BRODIE, 1995). Alguns autores consideram ainda, como conceito para sistema legado: ”toda aplicação em produção”. A partir da entrada de um determinado sistema em produção, o mesmo já pode estar ultrapassado, considerando as tecnologias adotadas e prazo para implementação. Já um sistema único, no contexto deste trabalho, refere-se a um sistema que simplesmente não faz parte de uma linha de produtos.

2.9 Migração de Sistemas Únicos para Linha de Produtos de *Software*

A adoção de uma *SPL* não é uma tarefa simples e pode acontecer de diversas formas e as vezes pode exigir uma reengenharia dos produtos já existentes. Assim sendo, o desenvolvimento não inicia do zero, mas sim a partir de uma migração para uma abordagem de linha de produtos. A adoção da *SPL* pode ser realizada tipicamente de três formas distintas (KRUEGER, 2002):

- **Proativa:** O desenvolvimento da *SPL* é completo, não existindo produtos que devem ser migrados para se adequarem a ela.
- **Extrativa:** O processo é inicialmente feito a partir da extração de características comuns e variáveis de um conjunto de produtos existentes, alterando-os a modo de formarem uma linha de produtos com base nas características extraídas.
- **Reativa:** O início é realizado de maneira incremental, com poucos produtos de fácil manipulação e cresce de acordo com a demanda expandindo o escopo com novos produtos da linha.

A abordagem proativa é custosa e requer um alto investimento na sua execução e sua adoção requer maior tempo e conhecimento. Sendo assim, uma grande parte das organizações não optam por esta abordagem. Em contrapartida, uma abordagem muito utilizada é a extrativa, consistindo em um processo de migração dos produtos já existentes em uma *SPL*.

A abordagem extrativa, ilustrada na Figura 2.7, é apropriada quando um grupo existente de sistemas únicos podem ser reutilizados. É mais apropriado quando o grupo de sistemas possui uma quantidade significativa de pontos em comum e também diferenças consistentes entre eles. Não é necessário executar a extração de todos os sistemas pré-existentes ao mesmo tempo. Por exemplo, um subconjunto dos sistemas de maior uso pode ser extraído inicialmente e, em seguida, o restante extraído incrementalmente, conforme necessário.

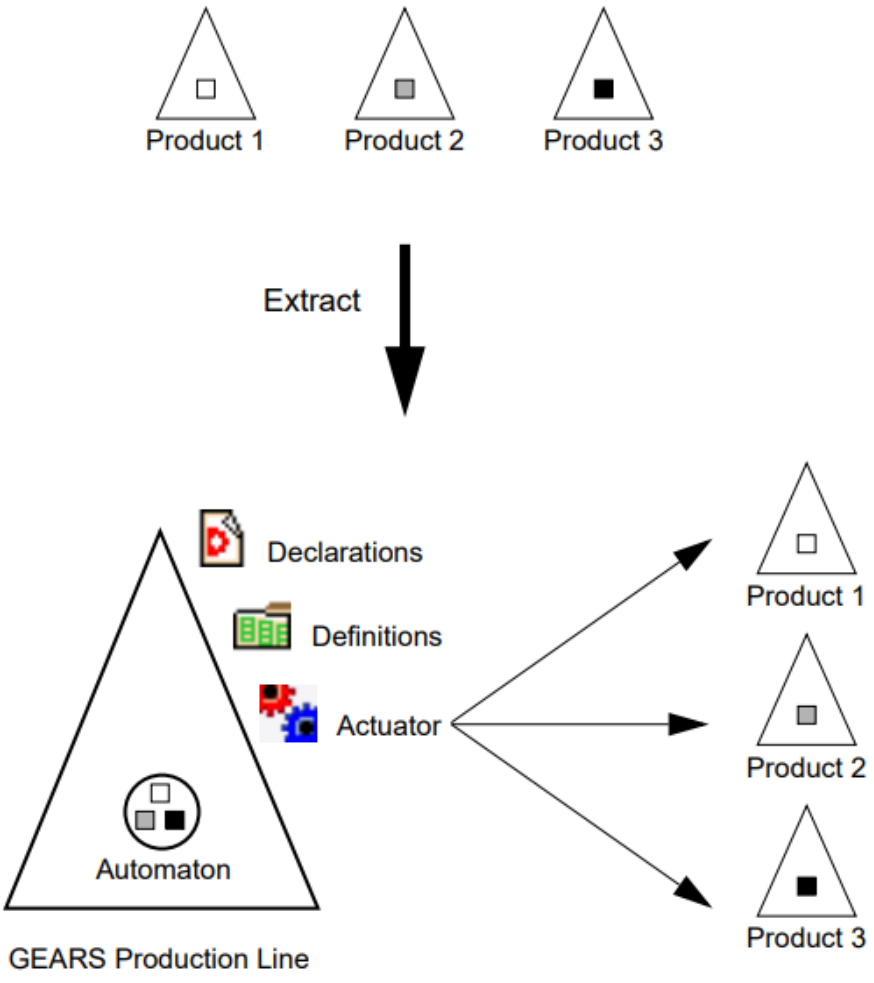

Figura 2.7: Modelo de adoção extrativo de *SPL* (KRUEGER, 2002)

As etapas em alto nível para a adoção da abordagem extrativa são as seguintes:

1. Identificar pontos em comum e variação nos sistemas existentes;
2. Fatorar em uma única linha de produtos;

3. Criar uma cópia de um dos sistemas únicos;

4. Criar declarações de *features* que modelam o escopo de variação entre os existentes sistemas;

5. Encapsular os pontos de variação na *SPL*;

6. Programar a lógica da *SPL* para mapear os valores dos parâmetros de declaração para seleções de variantes;

7. Criar as definições de produto para as versões de produto desejadas, selecionando valores para cada um dos parâmetros de definição das *features*

Considerando que o objetivo final deste trabalho é a avaliar a aplicação de ferramentas de *SPL* para a migração de sistema(s) único(s) em uma *SPL*, o critério utilizado para essa avaliação foi a abrangência de funcionalidades das ferramentas relatadas nesse trabalho, funcionalidades as quais mais fazem sentindo em se ter um ou mais sistemas únicos como auxilio para o desenvolvimento da *SPL* através da abordagem extrativa. As Funcionalidades de uma ferramenta para *SPL* como também as que tem aplicação direta na migração serão descritas no capitulo 4 deste trabalho.

Capítulo 3

Planejamento da Revisão Sistemática

Este trabalho foi realizado de acordo com o guia para revisão sistemática proposto por Kitchenham e colaboradores (KITCHENHAM et al., 2009). As técnicas foram realizadas com o intuito de identificar e classificar ferramentas de gerenciamento de *SPL*. O guia (KITCHENHAM et al., 2009) é estruturado seguindo um processo de três etapas: planejamento, condução e geração de relatórios.

3.1 Necessidade da Pesquisa

Uma revisão sistemática surge do crescente número de ferramentas para o gerenciamento de linha de produtos de *software* que se tornaram disponíveis, como por exemplo, FeatureIDE (FEATUREIDE, 2005), pure::variants (PURE::VARIANTS, 2003) e Gears (GEARS, 2001). Neste contexto, a escolha de uma ferramenta que melhor se adapte às necessidades dos pesquisadores e profissionais em um contexto específico de *SPL* está longe de ser trivial. Portanto, esta reexecução de uma Revisão sistemática da literatura *SLR* tem como objetivo fornecer uma visão geral das ferramentas disponíveis na literatura para o gerenciamento de *SPL*, a fim de descobrir se elas suportam o processo de migração de um sistema único para *SPL*.

3.2 Definição das Questões de Pesquisa

A principal questão de pesquisa deste trabalho é: Como as ferramentas disponíveis suportam o processo de gerenciamento de variabilidade? Para responder a essa questão, um conjunto de sub-questões foram derivadas da pergunta principal para identificar e analisar os trabalhos relevantes da literatura. Vale ressaltar que as questões foram extraídas do estudo realizado

em "*A Systematic Literature Review of Software Product Line Management Tools*" (PEREIRA; CONSTANTINO; FIGUEIREDO, 2014). As questões são:

- **Questão 1 (Q1):** Quantas ferramentas de gerenciamento de *SPL* foram citadas na literatura desde 2000?
- **Questão 2 (Q2):** Quais são as principais características das ferramentas?
- **Questão 3 (Q3):** Quais são as principais funcionalidades das ferramentas?

Para responder a primeira questão, foram identificadas ferramentas que são citadas na literatura desde 2000, pelo fato de a visibilidade fornecida por linha de produto de *software* após este ano ter produzido maior concentração de pesquisas na área (LISBOA et al., 2010). Com relação à segunda questão, foram vistas as características de cada ferramenta, quando e em que ambiente foram desenvolvidas. Através dessas informações, é possível definir sua aplicação em relação a *SPL*. Finalmente, no que diz respeito à terceira questão, como a ferramenta suporta cada etapa do processo de desenvolvimento, desde a concepção da *SPL*, ao seu desenvolvimento e manutenção. Em particular, quais tópicos, contribuições e novidades para *SPL* eles constituem. Assim, é possível mapear se as ferramentas estão suportando o processo de migração para *SPL* e se o processo não é totalmente suportado, ou seja, se existem lacunas nas ferramentas existentes ou se há necessidade de desenvolvimento de funcionalidades que não estão disponíveis nas ferramentas existentes.

3.3 Definindo a *String* de Busca

Com base nas questões, as palavras-chave foram selecionadas para originar uma *string* de busca para ser utilizada na pesquisa por estudos em fontes que serão posteriormente definidas. A *string* de busca utilizada foi construída utilizando a estratégia proposta por Chen e Babar (CHEN; BABAR, 2007). Assim como as questões, a *string* de busca também foi retirada do mesmo estudo (PEREIRA; CONSTANTINO; FIGUEIREDO, 2014) com o intuito de, inicialmente, para a primeira etapa deste trabalho, atualizar o trabalho realizado por esses autores. Com base nessa estratégia, este trabalho utiliza a seguinte *string* de busca:

("tool") AND ("product line" OR "product family" OR "system family") AND ("management" OR "modeling" OR "configuration")

O termo "*tool*" garante que a pesquisa seja conduzida para encontrar ferramentas. Além disso, os termos "*product line*", "*product family*" ou "*system family*" restringem a pesquisa por ferramentas de *SPL*. Por fim, os termos "*management*", "*modeling*" ou "*configuration*" se referem às principais funções para o desenvolvimento de ferramentas de *SPL*.

3.4 Seleção das Fontes de Busca

Os estudos foram identificados aplicando a *string* de busca em três bases de dados científicas:

- Biblioteca Digital ACM
- IEEE Xplore
- ScienceDirect

Essas bibliotecas foram escolhidas porque são algumas das mais relevantes na literatura de engenharia de *software* (TRAVASSOS; BIOLCHINI, 2007). A pesquisa será realizada utilizando a sintaxe específica de cada banco de dados, considerando apenas o título, o resumo e palavras-chave.

3.5 Definição dos Estudos

A base para a seleção dos estudos são os critérios de inclusão e exclusão (KITCHENHAM et al., 2009). Os seguintes critérios de inclusão (INC) foram usados para incluir estudos relevantes para responder as questões.

- **Critério de inclusão 1 (INC1):** As publicações devem ser do tipo '*journal*' ou '*conference*' e apenas os estudos escritos em inglês foram considerados.
- **Critério de inclusão 2 (INC2):** Foram incluídos apenas estudos que apresentam ferramentas para apoiar o processo de gerenciamento de *SPLs* (incluindo uma ou mais fases do ciclo de desenvolvimento e manutenção). Portanto, o título, o resumo e palavras-chave devem mencionar explicitamente que o foco do trabalho contribui com ferramentas que dão suporte o gerenciamento de variabilidade de *SPL*.

Os critérios de exclusão (EXC) seguintes foram usados para excluir estudos que não são considerados relevantes para responder às questões da pesquisa.

- **Critério de exclusão 1 (EXC1):** Relatórios técnicos apresentando lições aprendidas, dissertações, estudos que descrevem eventos.
- **Critério de exclusão 2 (EXC2):** Documentos duplicados. Se um trabalho for publicado em mais de um local, por exemplo, se um trabalho de conferência for estendido em uma versão de periódico, somente a última instância deve ser considerada.
- **Critério de exclusão 3 (EXC3):** Estudos que não focam em ferramentas de gerenciamento de *SPLs*. Abordagens, métodos, e outras técnicas para *SPL* por si só não devem ser consideradas.
- **Critério de exclusão 4 (EXC4):** Ferramentas que foram descontinuadas.
- **Critério de exclusão 5 (EXC5):** Ferramentas sem executável e/ou sem uma documentação descrevendo suas funcionalidades. Além disso, as ferramentas com documentação escrita que não possuem uma descrição sobre suas funcionalidades, também foram excluídas por não ser possível entender como a ferramenta funciona.

3.6 Extração de Dados e Avaliação de Qualidade dos Estudos

Como no estudo de Petersen (PETERSEN et al., 2008), os revisores primeiro leem resumos e procuram por palavras-chave e conceitos que refletem a contribuição do artigo. Quando apenas o resumo é insuficiente para dizer se o artigo em questão deve ser incluso ou não, pode-se optar por avaliar as seções de introdução ou conclusão do artigo em busca de evidências, de que o artigo deve ser incluído ou excluído da revisão. No conjunto de resultados encontrados, é aplicado os critérios de inclusão e exclusão para filtrar os documentos relevantes. Uma vez que a lista de estudos é composta, os dados das ferramentas citadas pelos documentos são extraídos. A fase de extração de dados visa resumir os dados dos artigos selecionados para uma análise mais aprofundada. Todas as documentações de ferramentas disponíveis, serviram como fontes de dados, como tutoriais, relatórios técnicos, dissertações, *websites*, entre outros.

Os dados extraídos de cada ferramenta selecionada são:

- Nome da ferramenta;
- Ano de lançamento;
- Características principais da ferramenta;
- Principais funcionalidades de cada ferramenta.

O mecanismo de pesquisa do Google será utilizado para coletar os dados das ferramentas.

Capítulo 4

Resultados e Análises

4.1 Coleta dos Estudos

A Tabela 4.1 sumariza o números de estudos encontrados na *SLR*. Perceba que, para esta análise, as etapas de inclusão e exclusão de estudos, 76 foram incluídos e 156 excluídos. Por fim, 57 estudos foram incluídos da *ACM Digital Library* (de um total de 163 coletados), 10 da *IEEE Xplore* (de 25 coletados) e 9 da *ScienceDirect* (de 44 estudos coletados).

	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019
Coletado	1	4	1	3	8	4	7	13	14	10	16	22	30	22	41	43	41	42	32	33
Incluído	0	0	1	1	3	1	4	4	7	3	5	7	6	10	11	12	13	12	13	15

Tabela 4.1: Número de estudos por ano

Após a etapa de inclusão de estudos, 61 ferramentas foram extraídas destes estudos. Após esta extração, outra pesquisa foi realizada com o auxilio do *Google* para encontrar informações particulares de cada ferramenta. Essa busca foi orquestrada com o intuito de encontrar uma possível documentação escrita, executável, *websites*, tutoriais, guias instalação, exemplos de uso, artigos e dissertações sobre a ferramenta em questão. Durante esta extração de dados das ferramentas, 49 ferramentas foram excluídas pelos seguintes critérios:

- A ferramenta em questão já fazia parte do estudo original de (PEREIRA; CONSTANTINO; FIGUEIREDO, 2014), logo, já está incluída neste trabalho;
- A ferramenta não foi encontrada em nenhum repositório externo ao trabalho em que a mesma é citada e utilizada, sendo considerada não disponível;

- Locais de acesso a documentação, repositório, *websites* descritos nos trabalhos, não estão mais disponíveis;
- A ferramenta em questão não auxiliava no gerenciamento de variabilidade, ou em alguma etapa chave dentro do contexto de linha de produtos de *software*;
- A ferramenta foi descontinuada, ou o projeto abandonado (EXC4);
- Não foi possível evidenciar como a ferramenta funciona pela pouca ou inexistente documentação sobre ela (EXC5);

O conjunto final de 52 ferramentas incluídas neste trabalho, como também os estudos de que foram extraídas, estão listados abaixo:

1. **MERLIN**: *"Analysing meta-model product lines"* (GUERRA; CHECHIK; SALAY, 2018)
2. **PUMConf**: *"Change impact analysis for evolving configuration decisions in product line use case models"* (HAJRI et al., 2019)
3. **ProductLinRE**: *"ProductlinRE: online management tool for requirements engineering of software product lines"* (GHOFRANI; FEHLHABER, 2018a)
4. **Yakindu Project**: *"Hierarchical featured state machines"* (FRAGAL; SIMAO; MOUSAVI, 2019)
5. **DarwinSPL**: *"DarwinSPL: an integrated tool suite for modeling evolving context-aware software product lines"* (NIEKE; ENGEL; SEIDL, 2017)
6. **SuperMod**: *"SuperMod: Tool support for collaborative filtered model-driven software product line engineering"* (SCHWäGER; WESTFECHTEL, 2016)
7. **SiPL**: *"SiPL - A Delta-Based Modeling Framework for Software Product Line Engineering"* (PIETSCH et al., 2015)
8. **ArchStudio**: *"Maintaining Architecture-Implementation Conformance to Support Architecture Centrality: From Single System to Product Line Development"* (ZHENG; CU; TAYLOR, 2018)

9. **RiPLE-HC**: *"RiPLE-HC: visual support for features scattering and interactions"* (SANTOS; MACHADO; ALMEIDA, 2016)

10. **VariantSync**: *"Synchronizing software variants with variantsync"* (PFOFE et al., 2016)

11. **EASy-Producer**: *"EASy-Producer: from product lines to variability-rich software ecosystems"* (SCHMID; EICHELBERGER, 2015)

12. **FMIT**: *"Towards a Semiautomatic Tool to Support the Integration of Feature Models"* (BISCHOFF et al., 2019)

13. **LISA toolkit**: *"Supporting Variability Management in Architecture Design and Implementation"* (GROHER; WEINREICH, 2013)

14. **ISMT4SPL**: *"An Integrated Software Management Tool for Adopting Software Product Lines"* (PARK, 2012)

15. **Sysiphus-IVMM**: *"Issue-Based Variability Management Information and Software Technology"* (THURIMELLA; BRUEGGE, 2012)

16. **VariaMos**: *"Variamos: A Tool for Product Line Driven Systems Engineering with a Constraint Based Approach"* (MAZO, 2012)

17. **VULCAN**: *"VULCAN: Architecture-Model-Based Workbench for Product Line Engineering"* (LEE, 2012)

18. **Pacogen**: *"Automatic Generation of Pairwise Test Configurations from feature models"* (HERVIEU, 2011)

19. **Invar**: *"Integrating Heterogeneous Variability Modeling Approaches with Invar"* (DHUNGANA, 2013)

20. **FMT**: *"A Software Product Line Approach for Ecommerce Systems"* (LAGUNA; HERNáNDEZ, 2010)

21. **Hephaestus**: *"Hephaestus: A Tool for Managing SPL Variabilities"* (BONIFáCIO, 2009)

22. **SPLOT**: *"S.P.L.O.T.: Software Product Lines Online Tools"* (MENDONçA, 2009)

23. **S2T2**: *"A Design of a Configurable Feature Model Configurator"* (BOTTERWECK, 2009)

24. **FeatureMapper**: *"FeatureMapper: Mapping Features to Models"* (HEIDENREICH, 2008)

25. **MoSPL**: *"Software Configuration Management for Product Derivation in Software Product Families"* (THAO, 2008)

26. **VISIT-FC**: *"Interactive Visualisation to Support Product Configuration in Software Product Lines"* (CAWLEY, 2008)

27. **DecisionKing**: *"Decisionking: A Flexible and Extensible Tool for Integrated Variability Modeling"* (DHUNGANA, 2007)

28. **DOPLER**: *"The Dopler Meta-Tool for Decision-Oriented Variability Modeling: A Multiple Case Study"* (DHUNGANA, 2011)

29. **FaMa**: *"Fama: Tooling a Framework for the Automated Analysis of Feature Models"* (BENAVIDES, 2007)

30. **GenArch**: *"A product Derivation Tool based on Model-Driven Techniques and Annotations"* (CIRILO, 2008)

31. **COVAMOF-VS**: *"The COVAMOF Derivation Process"* (SINNEMA, 2006)

32. **REMAP-tool**: *"Requirements Management for Product Lines: Extending Professional Tools"* (SCHMID, 2006)

33. **YaM**: *"Yam: A Framework for Rapid Software Development"* (JAIN; BIESIADECKI, 2006)

34. **AORA**: *"Aspect-Oriented Analysis for Software Product Lines Requirements Engineering"* (VARELA, 2011)

35. **DOORS Extension**: *"Modelling Requirements Variability across Product Lines"* (BUHNE, 2005)

36. **FeatureIDE**: *"FeatureIDE: An Extensible Framework for Feature-Oriented Software Development"* (THüM, 2014)

37. **Kumbang**: *"Kumbang tools"* (MYLLäRNIEMI, 2007)

38. **PLUSS toolkit**: *"The Pluss Toolkit: Extending Telelogic Doors and IBM-Rational Rose to Support Product Line Use Case Modeling"* (ERIKSSON, 2005)

39. **VARMOD**: *"Varmod-Prime Tool"* (VARMOD-PRIME, 2005)

40. **XFeature**: *"XFeature Modeling Tool"* (XFEATURE, 2005)

41. **DREAM**: *"Dream: Domain Requirement Asset Manager in Product Lines"* (PARK, 2004)

42. **ASADAL**: *"Asadal: A Tool System for Co-Development of Software and Test Environment based on Product Line Engineering"* (KIM, 2006)

43. **Pure::Variants**: *"Modeling and Building Software Product Lines with Eclipse"* (SPINCZYK; BEUCHE, 2004)

44. **Captain Feature**: *"Captain Feature Tool"* (CAPTAIN-FEATURE, 2002)

45. **DECIMAL**: *"Decimal and PLFaultCAT: From Product-Line Requirements to Product-Line Member Software Fault Trees"* (DEHLINGER, 2007)

46. **Odyssey**: *"Odyssey: A Reuse Environment based on Domain Models"* (BRAGA, 1999)

47. **GEARS**: *"Biglever Software Gears and the 3-tiered SPL Methodology"* (KRUEGER, 2007)

48. **WeCoTin**: *"Using a Configurator for Modelling and Configuring Software Product Lines based on Feature Models"* (ASIKAINEN, 2004)

49. **Holmes**: *"Holmes: An Intelligent System to Support Software Product Line Development"* (SUCCI, 2001)

50. **DARE**: *"Dare-cots: A Domain Analysis Support tool"* (FRAKES, 1997)

51. **Metadoc FM**: *"Metadoc Feature Modeler: A Plug-in for IBM Rational Doors"* (THURIMELLA; JANZEN, 2011)

52. **001**: *"Integrating 001 Tool Support in the Feature-Oriented Domain Analysis methodology"* (KRUT, 1993)

4.2 Principais Características das Ferramentas

Após a coleta de informações sobre as ferramentas incluídas, foram identificadas suas características principais. A Tabela 4.2 mostra quais características cada ferramenta implementa. Foram consideradas as seguintes características: interface gráfica (GUI), protótipo (PRT), online (ONL), *plugin* (PLG), gratuito para uso (FRE), *open-source* (OPS), guia de usuário (USG), exemplos de soluções disponíveis (EXS) e onde a ferramenta foi desenvolvida (DEV). Foi usado "n/a"para informações não disponíveis.

Nesta análise identificamos que a maioria das ferramentas não são protótipos (17%) nem online (17%). Além disso, foi constatado que a maioria das ferramentas são *plugins* (60%). *Plugins* fornecem a extensão de ferramentas já estabelecidas e conhecidas. Além disso, quase metade das ferramentas são gratuitas (43%) e apenas 18 são projetos *open-source* (34%) declarados (muitas ferramentas, apesar de aparentemente se comportarem como projetos *open-source* e gratuitos para o uso, não divulgam essa informação diretamente). Guia do usuário (USG) e soluções de exemplo (EXS) não estão disponíveis para a maioria das ferramentas analisadas. A pouca documentação disponível e a falta de exemplos de apoio são alguns dos motivos que podem desencorajar a adoção e o amplo uso dessas ferramentas, pois exigiria que os usuários tenham que adivinhar o comportamento da ferramenta (PEREIRA; CONSTANTINO; FIGUEIREDO, 2014). Por fim, foram extraídas informações sobre onde as ferramentas foram desenvolvidas, isto é, se foram desenvolvidas em ambiente acadêmico (A), industrial (I) ou ambos (B). Foi constatado que a maioria das ferramentas foram desenvolvidas no ambiente acadêmico.

	Ano	GUI	PRT	ONL	PLG	FRE	OPS	USG	EXS	DEV
MERLIN	2019	x	-	-	x	n/a	n/a	n/a	x	A
PUMConf	2019	x	-	-	x	n/a	n/a	x	n/a	A
ProductLinRE	2018	x	-	x	-	x	x	n/a	n/a	A
Yakindu Project	2017	x	-	-	x	x	x	x	x	I
DarwinSPL	2016	x	-	-	x	n/a	n/a	n/a	x	A
SuperMod	2015	x	-	x	x	n/a	n/a	x	x	A
SiPL	2015	x	-	-	x	n/a	n/a	x	x	A
ArchStudio	2015	x	-	-	x	x	x	x	n/a	A
RiPLE-HC	2015	x	-	-	x	n/a	n/a	n/a	n/a	A
VariantSync	2014	x	-	-	x	n/a	x	n/a	n/a	A
EASy-Producer	2014	x	-	-	x	n/a	n/a	x	x	A
FMIT	2013	x	x	-	x	n/a	n/a	n/a	n/a	A
LISA toolkit	2013	x	-	-	x	n/a	n/a	n/a	n/a	B
ISMT4SPL	2012	x	-	-	-	n/a	n/a	n/a	n/a	A
Sysiphus-IVMM	2012	x	-	-	x	x	x	n/a	n/a	A
VariaMos	2012	x	-	-	x	x	n/a	x	n/a	A
VULCAN	2012	x	-	-	x	x	x	n/a	n/a	A
Pacogen	2011	-	-	-	-	x	x	n/a	n/a	A
Invar	2010	x	x	x	-	x	n/a	n/a	x	A
FMT	2009	x	-	-	x	x	n/a	x	n/a	A
Hephaestus	2009	x	-	-	-	x	x	n/a	n/a	A
SPLOT	2009	x	-	x	-	x	x	n/a	x	A
S2T2	2009	x	-	-	x	x	n/a	n/a	x	B
FeatureMapper	2008	x	-	-	x	x	x	x	n/a	A
MoSPL	2008	x	x	-	-	n/a	n/a	n/a	n/a	A
VISIT-FC	2008	x	x	-	-	n/a	n/a	n/a	n/a	A
DecisionKing	2007	x	-	-	x	n/a	n/a	n/a	n/a	B
DOPLER	2007	x	-	-	x	n/a	n/a	n/a	x	B
FaMa	2007	x	-	-	x	x	x	x	x	A
GenArch	2007	x	-	-	x	x	x	n/a	n/a	A
COVAMOF-VS	2006	x	-	x	x	-	-	n/a	n/a	A
REMAP-tool	2006	x	x	-	x	-	-	n/a	x	B
YaM	2006	x	-	x	-	n/a	n/a	n/a	n/a	B
AORA	2005	x	-	-	-	x	x	n/a	x	A
DOORS Extension	2005	x	x	x	-	-	-	n/a	n/a	A
FeatureIDE	2005	x	-	-	x	x	x	x	x	A
Kumbang	2005	x	-	-	x	x	x	x	n/a	B
PLUSS toolkit	2005	x	-	-	-	-	-	n/a	n/a	B
VARMOD	2005	x	-	-	x	n/a	n/a	x	x	A
XFeature	2005	x	-	-	x	x	x	x	x	B
DREAM	2004	x	-	-	-	n/a	n/a	n/a	n/a	A
ASADAL	2003	x	-	-	-	x	n/a	n/a	n/a	A
Pure::Variants	2003	x	-	-	x	-	-	x	x	I
Captain Feature	2002	x	-	-	-	x	x	x	n/a	A
DECIMAL	2002	x	-	-	-	n/a	n/a	n/a	n/a	A
Odyssey	2002	x	-	-	-	x	-	x	x	A
GEARS	2001	x	-	-	x	-	-	n/a	n/a	I
WeCoTin	2000	x	x	x	-	-	-	n/a	n/a	I
Holmes	1999	x	x	-	-	n/a	n/a	n/a	n/a	A
DARE	1998	x	x	-	-	-	-	n/a	n/a	I
Metadoc FM	1998	x	-	x	x	-	-	x	x	I
001	1993	x	-	-	-	-	-	n/a	n/a	A

Tabela 4.2: Ferramentas e suas características principais

Característica	Porcentagem (%)
Interface gráfica (GUI)	**98%**
Protótipo (PRT)	17%
Online (ONL)	17%
Plugin (PLG)	**60%**
Gratuito para uso (FRE)	43%
Open-source (OPS)	34%
Guia de usuário (USG)	36%
Exemplos de soluções (EXS)	36%
Onde foi desenvolvida (DEV)	**A - I - B** 70% - 11% - 19%

Figura 4.1: Porcentagem da presença de cada característica nas ferramentas

4.3 Funcionalidades das Ferramentas

As próximas análises estendem uma classificação de funcionalidades descritas por (LISBOA et al., 2010) e expandida por (PEREIRA; CONSTANTINO; FIGUEIREDO, 2014), incluindo a geração de código fonte e o suporte à integração. As funcionalidades e sua explicação são:

1. **Documentação de pré-análise**: armazena e recupera as informações para ajudar na identificação de quais características deve fazer parte do domínio.

2. ***Matrix* do domínio**: representa o relacionamento entre as *features* e os aplicativos incluídos no domínio.

3. **Definição de escopo**: identifica os recursos que devem fazer parte da infraestrutura de reutilização do domínio.

4. **Representação do domínio**: representa o escopo definido. Este a representação pode ser através de tabelas, modelos, árvores e outros. Este modelo tenta formalizar as variações do domínio.

5. **Variabilidade**: representa as variabilidades que uma *feature* pode ter.

6. ***Features* obrigatórias**: representam as *features* que sempre estarão nos produtos.

7. **Regras de composição**: cria restrições no domínio para representar e relacionar as *features*. Essas restrições podem ser exclusão mútua, dependência, entre outros.

8. **Tipos de relacionamento**: fornece diferentes tipos de relacionamentos entre as *features*. Eles podem ser composição, generalização/especificação e implementação.

9. **Atributos da *feature***: permite a inclusão de informações específicas para cada *feature*. Também pode representar um ponto de variação, se o número de variantes for muito grande, fornecendo um modelo de *feature* mais conciso.

10. **Geração de código fonte**: Responsável por gerar código fonte baseado no modelo.

11. **Documentação sobre o domínio**: fornece documentação sobre o domínio, consistindo em: descrição, contexto, dependências entre sistemas no domínio, entre outros.

12. **Gerenciamento de requisitos**: fornece suporte para a inclusão de requisitos e/ou casos de uso.

13. **Relacionamento**: relaciona as *features* existentes de um domínio aos requisitos (funcionais ou não funcionais) e/ou casos de uso definidos.

14. **Relatórios**: gera relatórios sobre as informações disponíveis no domínio. Os relatórios podem representar, por exemplo, o número de combinações possíveis, a frequência em que os recursos aparecem no produto.

15. **Checagem de consistência**: verifica se o domínio gerado segue as regras de composição criadas.

16. **Derivação de produto**: identifica as *features* que pertencem a um produto de acordo com as *features* definidos para o domínio.

17. **Importar e exportar**: fornece a função de importar/exportar de/para outros aplicativos.

Apenas as funcionalidades de 1 a 10 foram consideradas neste trabalho devido sua maior importância para a realização deste estudo. O motivo da seleção priorizada destas funcionalidades será descrita na seção 4.4. Um ponto a se destacar sobre as ferramentas mostradas na tabela 4.3 é que quando uma ferramenta depende obrigatoriamente de outra, foi assumido que essa ferramenta herda suas funcionalidades básicas.

	1	2	3	4	5	6	7	8	9	10
SuperMod	-	-	-	x	x	x	x	x	-	x
SiPL	-	-	-	x	x	x	x	x	-	-
Yakindu Project	-	-	-	x	x	-	x	-	x	x
FMIT	-	-	-	x	x	x	x	x	-	x
MERLIN	-	-	-	x	x	x	x	x	-	x
ProductLinRE	x	x	x	-	-	-	-	-	-	-
PUMConf	-	-	-	x	x	x	x	-	-	-
ArchStudio	-	-	-	x	-	-	-	-	-	-
DarwinSPL	-	-	-	x	x	x	x	x	x	-
RiPLE-HC	-	-	-	x	x	x	x	x	-	x
VariantSync	-	-	-	x	x	x	x	x	-	x
EASy-Producer	-	-	-	-	-	-	-	-	-	-
LISA toolkit	x	-	x	x	x	x	x	x	x	x
ISMT4SPL	x	-	x	x	x	x	-	x	-	x
Sysiphus-IVMM	x	x	x	x	x	x	x	-	-	-
VariaMos	-	-	-	x	x	x	x	-	-	-
VULCAN	-	-	-	x	x	x	x	x	x	x
Pacogen	-	-	-	-	-	-	-	-	-	-
Invar	-	-	-	-	-	-	-	-	-	-
FMT	-	-	-	x	x	x	x	-	x	-
Hephaestus	-	-	-	-	-	-	-	-	-	-
SPLOT	-	-	-	x	x	x	x	-	x	-
S2T2	-	-	-	x	x	x	x	-	-	-
FeatureMapper	-	-	-	x	x	x	x	-	x	-
MoSPL	-	-	-	x	x	x	x	-	-	-
VISIT-FC	-	-	-	x	x	x	x	-	-	-
DecisionKing	x	-	x	x	x	x	x	-	x	-
DOPLER	x	-	x	x	x	x	x	-	x	-
FaMa	-	-	-	x	x	x	x	x	x	-
GenArch	-	-	-	x	x	x	x	-	x	-
COVAMOF-VS	-	-	-	x	x	x	x	-	-	-
REMAP-tool	x	-	x	x	x	x	x	-	-	-
YaM	-	-	x	-	-	-	-	-	-	-
AORA	x	x	x	x	x	x	x	-	-	-
DOORS Extension	-	-	-	x	x	x	x	-	-	-
FeatureIDE	-	-	-	x	x	x	x	x	-	x
Kumbang	-	-	-	x	x	x	x	-	x	-
PLUSS toolkit	-	-	-	x	x	x	x	-	-	-
VARMOD	-	-	-	x	x	x	x	-	x	-
XFeature	-	-	-	x	x	x	x	x	x	x
DREAM	-	x	x	x	x	x	-	x	-	-
ASADAL	-	-	-	x	x	x	x	x	-	x
Pure::Variants	-	-	-	x	x	x	x	x	x	x
Captain Feature	-	-	-	x	x	x	x	-	-	-
DECIMAL	-	-	-	x	x	x	x	-	-	-
Odyssey	-	-	-	x	x	x	-	-	-	-
GEARS	-	-	-	x	x	x	x	x	-	x
WeCoTin	-	-	-	x	x	x	x	-	x	-
Holmes	x	x	x	x	x	x	x	-	-	-
DARE	x	x	x	x	x	x	-	-	-	-
Metadoc FM	-	-	-	x	x	x	x	-	-	-
001	-	-	-	x	x	x	x	x	-	x

Tabela 4.3: Funcionalidades das ferramentas

4.4 Aplicação das Ferramentas para a Migração

Para se avaliar a aplicação de ferramentas que dão suporte ao desenvolvimento e manutenção de *SPLs*, em um cenário de migração originária de um ou mais sistemas únicos, que supostamente fazem parte de um mesmo grupo ou família, para um contexto de linha de produtos de *software*, um conjunto de funcionalidades foram selecionadas, funcionalidades estas que dão suporte a migração, em outras palavras, que a existência de um sistema único, ou mais, já em produção, auxilia de alguma forma o desenvolvimento da linha de produtos. Essas funcionalidades são:

- **Funcionalidades de planejamento**: documentação de pré-análise, *matrix* do domínio e definição de escopo;
- **Funcionalidades de modelagem**: representação do domínio, variabilidade, *features* obrigatórias, regras de composição, tipos de relacionamento, atributos da *feature* e geração de código fonte.

O conjunto de funcionalidades de planejamento, por sua característica extrativa de *features* através de um sistema único (GHOFRANI; FEHLHABER, 2018b) e o conjunto de funcionalidades de modelagem, por formalizarem a arquitetura completa SPL e seus produtos derivados.

Na Tabela 4.2 mostra que a maioria das ferramentas disponíveis suportam as quatro primeiras funcionalidades de modelagem, porém, grande parte das ferramentas analisadas não suportam o grupo de funcionalidades de planejamento. No planejamento, as funcionalidades identificadas estão disponíveis em apenas 12 ferramentas; e apenas 5 ferramentas implementam todas as funcionalidades neste grupo. Isso parece indicar que, apesar do crescente interesse e importância, o planejamento ainda não é o foco principal da comunidade de pesquisa de linha de produtos. Ademais, há evidência da falta de ferramentas para suportar todas as funcionalidades para o desenvolvimento de uma *SPL*.

Por fim, conclui-se que esta falta de ferramentas que dão suporte total ao gerenciamento de *SPL* e a lacuna nas ferramentas analisadas no quesito de apoio às funcionalidades de planejamento tem impacto negativo direto no critério de migração de sistema(s) Únicos para *SPL*.

Planejamento			Modelagem						
1	2	3	4	5	6	7	8	9	10
19%	11%	12%	88%	87%	85%	79%	34%	32%	28%

Figura 4.2: Porcentagem da presença de cada funcionalidade nas ferramentas

4.5 Ameaças à Validade da Revisão

Uma questão importante ao realizar uma *SLR* é a validade dos resultados. A pergunta é: o estudo foi projetado e realizado de maneira sólida e controlada? Esta seção apresenta as diferentes ameaças à validade relacionadas à *SLR*. Apresentamos como as ameaças foram tratadas para minimizar riscos à validade da *SLR* em relação aos quatro grupos de ameaças comuns à validade: validade interna, validade externa, validade de construção e validade de conclusão (WOHLIN et al., 2012).

- **Validade externa.** A validade externa diz respeito à capacidade de generalizar os resultados para outros ambientes, como para à indústria (WOHLIN et al., 2012). A maior validade externa para este trabalho foi durante a identificação dos estudos primários. A busca pelas ferramentas foi realizada em três bibliotecas digitais científicas relevantes, a fim de capturar o máximo possível de ferramentas disponíveis e evitar todo tipo de viés. No entanto, a qualidade das *engines* de pesquisa pode ter influenciado a completude dos estudos primários identificados. Isso significa que a pesquisa pode ter perdido os estudos cujos autores usaram termos diferentes para especificar a ferramenta de gerenciamento de *SPL*, ou não terem usado as palavras-chave que foram utilizadas para pesquisar no título, resumo e palavras-chave de seus artigos.

- **Validade interna.** A validade interna diz respeito à questão de saber se o efeito é causado por variáveis independentes (por exemplo, revisores) ou por outros fatores (WOHLIN et al., 2012). Nesse sentido, uma limitação deste estudo diz respeito à confiabilidade. Se o estudo for replicado por outro conjunto de pesquisadores, é possível que alguns estudos removidos nesta revisão possam ser incluídos e outros estudos possam ser excluídos. No

entanto, em geral, acredita-se que a validade interna da *SLR* é alta, dada a utilização de um procedimento sistemático.

- **Validade de construção.** A validade de construção reflete se o trabalho realizado realmente representa o que o pesquisador tinha em mente de acordo com os perguntas de pesquisa (WOHLIN et al., 2012). Uma ameaça à validade de construção pode ser um julgamento tendencioso. Uma possível ameaça nessa revisão é excluir alguma ferramenta relevante. Para minimizar essa ameaça, os processos de inclusão e exclusão foram novamente visitados em determinados estudos aparentemente relevantes, para que, além de seu resumo, título e palavras-chave, suas introduções e conclusões também fossem checadas. Portanto, acreditamos que não omitida nenhuma ferramenta relevante.

- **Validade da conclusão.** Ameaças à validade das conclusões estão relacionadas com as questões que afetam a capacidade de se tirar as conclusões corretas sobre o trabalho (WOHLIN et al., 2012). Uma potencial ameaça à validade da conclusão é a confiabilidade da extração de informações sobre as ferramentas, pois nem todas as informações eram facilmente encontradas para se responder as questões de pesquisa e alguns dados precisaram ser interpretados. Portanto, para garantir a validade, várias fontes de dados foram analisadas, ou seja, documentos, sites, relatórios técnicos, manuais, entre outros.

Capítulo 5

Conclusão e Trabalhos Futuros

Grandes sistemas de *software* requerem um enorme esforço para seu desenvolvimento e gerenciamento. O problema é agravado quando esses vários sistemas não fazem parte de uma linha de produtos de software, e seu desenvolvimento e manutenção é feita de forma isolada. Para enfrentar esse problema, várias ferramentas foram propostas para representar e gerenciar as variações de um sistema. Portanto, este trabalho tem como objetivo identificar e avaliar as ferramentas existentes na literatura para apoiar o gerenciamento de *SPL*, diagnosticando se tais ferramentas oferecem suporte para a migração de sistema(s) único(s) em produtos de uma *SPL*.

Para realizar o trabalho, uma revisão sistemática da literatura de ferramentas de gerenciamento de linha de produtos de *software* foi realizada, seguindo o guia de revisão sistemática proposta por (KITCHENHAM et al., 2009). Esta *SLR* fornece uma visão geral das ferramentas, onde é possível ver quais características e funcionalidades presentes em cada ferramenta. Ao final da revisão, as ferramentas coletadas foram novamente avaliadas para se concluir se além do seu proposito de gerenciamento de *SPL*, estas são aptadas a auxiliar no processo de migração de um ou mais sistemas únicos para o contexto de *SPL*. A *SLR* identificou 52 ferramentas existentes na literatura em conjunto ao estudo realizado por em "*A Systematic Literature Review of Software Product Line Management Tools*" (PEREIRA; CONSTANTINO; FIGUEIREDO, 2014).

Resultados mostram que as ferramentas possuem documentação limitada. A complexidade e/ou a indisponibilidade das ferramentas são alguns dos motivos que podem acabar desencorajando a adoção e o amplo uso dessas ferramentas nas organizações e na academia. Além disso, ainda existem lacunas de funcionalidades no suporte completo ao processo de gerenciamento em todas as ferramentas investigadas. Quanto ao processo de migração, o cenário ainda se

agrava, pois a maior parte das ferramentas não suporta funcionalidades de planejamento para extração dos requisitos e/ou *features* vindas de sistemas únicos.

5.1 Trabalhos Futuros

Como possíveis trabalhos futuros, uma opção é o desenvolvimento de uma ferramenta que contemple todas as funcionalidades que auxiliam em uma migração para *SPL*, ou até mesmo a adição de tais funcionalidades em uma das ferramentas já existentes citadas neste trabalho, através do desenvolvimento de um *plugin* para a ferramenta ou contribuição em um projeto *open-source*.

Referências Bibliográficas

APEL, S. et al. *Feature-Oriented Software Product Lines: Concepts and Implementation*. 1. ed. [S.l.]: Springer-Verlag Berlin Heidelberg, 2013.

ASIKAINEN, T. Using a configurator for modelling and configuring software product lines based on feature models. *Workshop on Software Variability Management for Product Derivation, Software Product Line Conference (SPLC)*, p. 24–35, 2004.

BASTOS, J. F. et al. Adopting software product lines: A systematic mapping study. *15th Annual Conference on Evaluation and Assessment in Software Engineering (EASE)*, Durham, UK, 2011.

BENAVIDES, D. Fama: Tooling a framework for the automated analysis of feature models. *1st International Workshop on Variability Modelling of Software Intensive Systems (VaMoS)*, p. 129–134, 2007.

BENNETT, K. H. Legacy systems: coping with success. IEEE Software, p. 19–23, 1995.

BEUCHE, D.; SCHRöDER-PREIKSCHAT, W.; PAPAJEWSKI, H. Variability management with feature models. *Journal Science of Computer Programming*, 2004.

BISCHOFF, V. et al. Towards a semiautomatic tool to support the integration of feature models. *15th Brazilian Symposium on Information Systems (SBSI)*, 2019.

BONIFáCIO, R. Hephaestus: A tool for managing spl variabilities. *Brazilian Symposium on Components, Architectures and Reuse Software (SBCARS)*, p. 26–34, 2009.

BOTTERWECK, G. A design of a configurable feature model configurator. *3rd International Workshop on Variability Modelling of Software Intensive Systems (VaMoS)*, p. 165–168, 2009.

BRAGA, R. Odyssey: A reuse environment based on domain models. *IEEE Symposium on Application-Specific Systems and Software Engineering and Technology (ASSET)*, p. 50–57, 1999.

BUHNE, S. Modelling requirements variability across product lines. *3th International Conference on Requirements Engineering (RE)*, p. 41–50, 2005.

CAPTAIN-FEATURE. *Captain Feature Tool*. 2002. Consultado na INTERNET: http://sourceforge.net/projects/captainfeature, 2019.

CAWLEY, C. Interactive visualisation to support product configuration in software product lines. *2nd International Workshop on Variability Modeling of Software- Intensive Systems (VaMoS)*, p. 7–16, 2008.

CHEN, L.; BABAR, M. A. A systematic review of evaluation of variability management approaches in software product lines. *Journal Information and Software Technology*, 2007.

CIRILO, E. A product derivation tool based on model-driven techniques and annotations. *Journal of Universal Computer Science*, p. 1344–1367, 2008.

CLEMENTS, P.; NORTHROP, L. *Software Product Lines: Practices and Patterns*. [S.l.]: Addison-Wesley, 2001.

DEHLINGER, J. Decimal and plfaultcat: From product-line requirements to product-line member software fault trees. *29th International Conference on Software Engineering (ICSE)*, p. 49–50, 2007.

DHUNGANA, D. Decisionking: A flexible and extensible tool for integrated variability modeling. *1st International Workshop on Variability Modelling of Softwareintensive Systems (VaMoS)*, p. 119–128, 2007.

DHUNGANA, D. The dopler meta-tool for decision-oriented variability modeling: A multiple case study. *Journal Automated Software Engineering*, p. 77–114, 2011.

DHUNGANA, D. Integrating heterogeneous variability modeling approaches with invar. *7th International Workshop on Variability Modelling of Software-intensive Systems (VaMoS)*, 2013.

ERIKSSON, M. The pluss toolkit: Extending telelogic doors and ibm-rational rose to support product line use case modeling. *20th International Conference on Automated Software Engineering (ASE)*, p. 300–304, 2005.

FEATUREIDE. 2005. Consultado na INTERNET: https://featureide.github.io/, 2019.

FRAGAL, V. H.; SIMAO, A.; MOUSAVI, M. R. Hierarchical featured state machines. *Science of Computer Programming*, p. 67–88, 2019.

FRAKES, W. Dare-cots: A domain analysis support tool. *International Conference of the Chilean Computer Science Society*, p. 73–77, 1997.

GEARS. 2001. Consultado na INTERNET: https://biglever.com/solution/gears/, 2019.

GHOFRANI, J.; FEHLHABER, A. L. Productlinre: online management tool for requirements engineering of software product lines. *22nd International Systems and Software Product Line Conference*, p. 17–22, 2018.

GHOFRANI, J.; FEHLHABER, A. L. Productlinre: online management tool for requirements engineering of software product lines. *SPLC18 Proceedings of the 22nd International Systems and Software Product Line Conference*, 2018.

GROHER, I.; WEINREICH, R. Supporting variability management in architecture design and implementation. *46th Hawaii International Conference on System Sciences (HICSS)*, p. 4995–5004, 2013.

GUERRA, J. d. L. E.; CHECHIK, M.; SALAY, R. Analysing meta-model product lines. *SLE 2018 Proceedings of the 11th ACM SIGPLAN International Conference on Software Language Engineering*, p. 160–173, 2018.

HAJRI, I. et al. Change impact analysis for evolving configuration decisions in product line use case models. *10th International Workshop on Software and Systems Traceability*, p. 11–11, 2019.

HEIDENREICH, F. Featuremapper: Mapping features to models. *International Conference on Software Engineering (ICSE)*, p. 943–944, 2008.

HERVIEU, A. Automatic generation of pairwise test configurations from feature models. *22nd International Symposium on Software Reliability Engineering (ISSRE)*, p. 120–129, 2011.

JAIN, A.; BIESIADECKI, J. Yam: A framework for rapid software development. *2nd IEEE International Conference on Space Mission Challenges for Information Technology (SMC-IT)*, p. 182–194, 2006.

KANG, K.; SUGUMARAN, V.; PARK, S. *Applied Software Product Line Engineering*. Boston, MA: Auerbach Publications, 2009.

KIM, K. Asadal: A tool system for co-development of software and test environment based on product line engineering. *28th International Conference on Software Engineering (ICSE)*, p. 783–786, 2006.

KITCHENHAM, B. et al. Systematic literature reviews in software engineering: A systematic literature review. *Journal Information and Software Technology*, 2009.

KRUEGER, C. Biglever software gears and the 3-tiered spl methodology. *22nd Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA)*, p. 844–845, 2007.

KRUEGER, C. W. Revised papers from the 4th international workshop on software product-family engineering. In: ____. London, UK: Springer-Verlag, 2002. cap. : Easing the Transition to Software Mass Customization, p. 282–293.

KRUT, J. R. W. Integrating 001 tool support in the feature-oriented domain analysis methodology. *Technical Report, Software Engineering Institute (SEI)*, 1993.

LAGUNA, M.; HERNáNDEZ, C. A software product line approach for ecommerce systems. *7th International Conference on e-Business Engineering (ICEBE)*, p. 230–235, 2010.

LEE, H. Vulcan: Architecture-model-based workbench for product line engineering. *16th International Software Product Line Conference (SPLC)*, p. 260–264, 2012.

LINDEN, F. J. van der; SCHMID, K.; ROMMES, E. *Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering*. 1. ed. [S.l.]: Springer-Verlag Berlin Heidelberg, 2007.

LISBOA, L. B. et al. A systematic review of domain analysis tools. *Journal Information and Software Technology*, 2010.

MAZO, R. Variamos: A tool for product line driven systems engineering with a constraint based approach. *24th International Conference on Advanced Information Systems Engineering (CAiSE)*, p. 1–8, 2012.

MENDONçA, M. S.p.l.o.t.: Software product lines online tools. *24th Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA)*, p. 761–762, 2009.

MYLLäRNIEMI, V. Kumbang tools. *11th Software Product Line Conference (SPLC)*, p. 135–136, 2007.

NIEKE, M.; ENGEL, G.; SEIDL, C. Darwinspl: an integrated tool suite for modeling evolving context-aware software product lines. *11th International Workshop on Variability Modelling of Software-intensive Systems*, p. 92–99, 2017.

PARK, J. Dream: Domain requirement asset manager in product lines. *International Symposium on Future Software Technology (ISFST)*, 2004.

PARK, K. An integrated software management tool for adopting software product lines. *11th International Conference on Computation and Information Science (ICIS)*, p. 553–558, 2012.

PEREIRA, J. A.; CONSTANTINO, K.; FIGUEIREDO, E. A systematic literature review of software product line management tools. 2014.

PETERSEN, K. et al. Systematic mapping studies in software engineering. *12th International Conference on Evaluation and Assessment in Software Engineering. (EASE)*, p. 68–77, 2008.

PFOFE, T. et al. Synchronizing software variants with variantsync. *20th International Systems and Software Product Line Conference (SPLC)*, p. 329–332, 2016.

PIETSCH, C. et al. Sipl - a delta-based modeling framework for software product line engineering. *30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2015.

POHL, K.; BöCKLE, G.; LINDEN, F. van der. *Software Product Line Engineering: Foundations, Principles and Techniques*. 1. ed. [S.l.]: Springer-Verlag Berlin Heidelberg, 2005.

PURE::VARIANTS. 2003. Consultado na INTERNET: https://www.pure-systems.com/products/pure-variants-9.html, 2019.

SANTOS, A. R. et al. Test-based spl extraction: An exploratory study. *28th ACM Symposium on Applied Computing (SAC), Software Engineering Track*, p. 1031–1036, 2013.

SANTOS, A. R.; MACHADO, I. do C.; ALMEIDA, E. S. de. Riple-hc: visual support for features scattering and interactions. *20th International Systems and Software Product Line Conference (SPLC)*, p. 320–323, 2016.

SCHMID, K. Requirements management for product lines: Extending professional tools. *10th International Software Product Line Conference (SPLC)*, p. 113–122, 2006.

SCHMID, K.; EICHELBERGER, H. Easy-producer: from product lines to variability-rich software ecosystems. *19th International Conference on Software Product Line*, p. 390–391, 2015.

SCHWäGER, F.; WESTFECHTEL, B. Supermod: Tool support for collaborative filtered model-driven software product line engineering. *31st IEEE/ACM International Conference on Automated Software Engineering (ASE)*, p. 822–827, 2016.

SINNEMA, M. The covamof derivation process. *9th International Conference on Reuse of Off-the-Shelf Components (ICSR)*, p. 101–114, 2006.

SPINCZYK, O.; BEUCHE, D. Modeling and building software product lines with eclipse. *19th conference on Object-oriented programming systems, languages, and applications (OOPSLA)*, p. 18–19, 2004.

STONEBRAKER, M.; BRODIE, M. L. Migrating legacy systems: Gateways, interfaces the incremental approach. In: ____. 1. ed. [S.l.]: Morgan Kaufmann Pub, 1995.

SUCCI, G. Holmes: An intelligent system to support software product line development. *23rd International Conference on Software Engineering (ICSE)*, p. 829–830, 2001.

THAO, C. Software configuration management for product derivation in software product families. *15th International Conference and Workshop on the Engineering of Computer Based Systems (ECBS)*, p. 265–274, 2008.

THURIMELLA, A. K.; BRUEGGE, B. Issue-based variability management information and software technology. *Journal Information and Soft. Technology*, p. 933–950, 2012.

THURIMELLA, A. K.; JANZEN, D. Metadoc feature modeler: A plug-in for ibm rational doors. *International Software Product Line Conference (SPLC)*, p. 313–322, 2011.

THüM, T. Featureide: An extensible framework for feature-oriented software development. *Journal Science of Computer Programming*, p. 70–85, 2014.

TRAVASSOS, G.; BIOLCHINI, J. Systematic review applied to software engineering. *Brazilian Symposium on Software Engineering (SBES), Tutorials*, p. 436, 2007.

VARELA, P. Aspect-oriented analysis for software product lines requirements engineering. *Proceedings of the 2011 ACM Symposium on Applied Computing*, 2011.

VARMOD-PRIME. *Varmod-Prime Tool*. 2005. Consultado na INTERNET: https://paluno.uni-due.de/en/varmod-prime, 2019.

WEISS, D. M. Architecture of product lines. *IEEE International Conference on Software Maintenance*, Edmonton, AB, Canada, 2009.

WOHLIN, C. et al. *Experimentation in Software Engineering: An Introduction*. [S.l.]: Kluwer Academic Publishers, 2012.

XFEATURE. *XFeature Modeling Tool*. 2005. Consultado na INTERNET: https://www.pnp-software.com/XFeature/, 2019.

ZHENG, Y.; CU, C.; TAYLOR, R. Maintaining architecture-implementation conformance to support architecture centrality: From single system to product line development. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 2018.