Identification and Relationship Between Notation and Tool for Feature Models with Graphic Representation

Gustavo Vale
Department of Computer Science
Federal University of Lavras
Lavras, Brazil
gustavo.a.vale@sistemas.ufla.br

Ramon Abílio
Department of Computer Science
Federal University of Lavras
Lavras, Brazil
ramon@posgrad.ufla.br

Juliana Pereira
Department of Computer Science
Federal University of Minas Gerais
Lavras, Brazil
juliana.pereira@dcc.ufmg.br

Eduardo Figueiredo
Department of Computer Science
Federal University of Minas Gerais
Lavras, Brazil
figueiredo@dcc.ufmg.br

Paulo Afonso Junior
Department of Computer Science
Federal University of Lavras
Lavras, Brazil
pauloa.junior@dcc.ufla.br

Heitor Costa
Department of Computer Science
Federal University of Lavras
Lavras, Brazil
heitor@dcc.ufla.br

***Abstract* — Feature models are composed of features and their relationships. They are used to represent the common features and variabilities of a Software Product Line (SPL). Many proposals for notations and tools to feature models can be found in the literature. Thus, the choosing the notation and the tool that best fits the needs of the software engineer is a hard task. This paper shows a study on which tools allow to represent some notations. Fifteen notations were identified and compared, which are related to five tools. The eight attributes of notations related to tools are: graphical structure, type of features (mandatory, optional, and external), type of decomposition (OR, XOR, and AND) and cross restrictions. In the analysis, notations, attributes, and tools were related. As main contribution, we presented a relationship between tools and notations of feature models that can assist in the choice of one notation/tool for modeling a domain.**

Keywords — Software product line; feature models; notations; tools;

I. INTRODUÇÃO

A comercialização de produtos que se diferenciam por variações em suas características é comum, por exemplo, computadores de mesmo modelo diferenciados pela velocidade do processador ou pela quantidade de memória. Na produção de produtos, há características comuns aos produtos e características que variam de um produto para outro. Os principais objetivos da estratégia de produção em massa são o aumento da variedade e da qualidade de produtos comercializados e a agilidade da produção. Na produção de sistemas de software, os objetivos são similares, o que impulsiona a estratégia de Linhas de Produtos de Software (LPS) [10].

Uma LPS é uma estratégia para projeto e implementação de sistemas de software cujo objetivo é promover o reúso sistemático e em larga escala de componentes desses sistemas [10]. Esse reúso é obtido pela forma com que a arquitetura da LPS é projetada [21]: características comuns a um domínio de aplicação formam o núcleo e características específicas. Essas características podem ser definidas como (conjunto de) módulos de um sistema com funções coerentes, bem definidas, independentes e combináveis [7]. Uma vez que um sistema é decomposto em características combináveis, diferentes versões desse sistema podem ser obtidas com a inclusão/exclusão de características. A inserção de uma característica em uma LPS não deve irradiar alterações em módulos não diretamente dependentes dessa característica. Caso aconteça, a estabilidade da arquitetura pode estar comprometida. A utilização da tecnologia de orientação a características pode permitir o reúso e a estabilidade de uma LPS [31].

A diferença entre seus produtos, conhecida como variabilidade, é um ponto chave a ser gerenciado em uma LPS e, por ser complexo, esse gerenciamento tem a atenção da academia [38]. Modelo de Características, utilizando a notação FODA (*Feature-Oriented Domain Analysis*), foi introduzido em 1990 para representar a variabilidade de uma LPS [29]. Algumas extensões dessa notação foram criadas para melhorar ou acrescentar semântica e sintaxe à proposta inicial, por exemplo, a criação da notação GPFT (*Generative Programming Feature Tree*); nela, foi acrescido o relacionamento OR em relação a notação FODA. Esses modelos são compostos por representação da organização e do relacionamento das características, podendo ser textuais ou gráficos. A representação textual pode ser de utilização mais prática para engenheiros e desenvolvedores, por facilitar a automação de tarefas [22][7], mas, de forma geral, essa representação requer conhecimento técnico para ser utilizada. A representação gráfica é mais utilizada [7] e pode-se reconhecer a variabilidade da LPS de forma simples e intuitiva. Procurando atender prós e contras das notações, ferramentas para construir modelos de características possibilitam o intercâmbio entre as representações, exportando a representação gráfica em representação textual.

A existência de uma variedade de notações e de ferramentas para Modelos de Características e o fato delas estarem dispersas motivaram a realização deste trabalho, cujo objetivo é

978-1-5090-3339-3 / 16 / $ 31,00 © 2016 IEEE

identificar e caracterizar notações e identificar ferramentas na literatura, para verificar quais ferramentas permitem representar notações e constituir um catálogo com essas informações. O intuito desse catálogo para auxiliar engenheiros de software e pesquisadores a escolher a notação mais adequada para atender suas necessidades de modelagem.

O restante do artigo está organizado da seguinte forma. A diferença entre Modelo de Características e Diagrama de Características é apresentada na Seção II. A pesquisa e os procedimentos realizados para identificar e selecionar 15 notações para Modelos de Características são apresentados na Seção III. Descrição dos elementos que compõem as notações para Modelo de Características, de forma genérica, é tratada na Seção IV. As notações encontradas são descritas na Seção V. Uma discussão sobre as notações é vista na Seção VI. Um estudo analítico das notações representadas por ferramentas de suporte à configuração e gerência de variabilidade em LPS é apresentado na Seção VII. A apresentação de trabalhos relacionados está resumida na Seção VIII. Conclusões, limitações, contribuições e sugestões de trabalhos futuros estão na Seção IX.

II. Modelo versus Diagramas de Características

Na primeira notação proposta, denominada FODA [26], Modelos de Características são compostos por Diagrama de Características, regras de composição (tratadas neste trabalho como restrições transversais) e "questões e decisões". Diagramas de Características são representados como combinações das características identificadas sob um domínio. Uma regra de composição define semânticas de relacionamentos existentes entre características não representadas em um Diagrama de Características [26]. Questões e decisões envolvem, por exemplo, medidas para coletar o esforço esperado para aplicar a análise de domínio ou as tomadas de decisão necessárias para especificar ou implementar um novo sistema [26]. Algumas definições foram apresentadas posteriormente:

- Modelo de Características consiste em um Diagrama de Características e alguma informação adicional associada (e.g., análise relacional, restrições e regras de dependência). A organização hierárquica das características é apresentada utilizando notação gráfica de árvore em um Diagrama de Características [50];
- Modelo de Características corresponde a combinações válidas de características em um domínio de aplicação. Esse modelo é representado em uma estrutura de árvore em que os nós e arcos representam as características e os relacionamentos entre elas, respectivamente [40];
- Modelo de Características é composto por um Diagrama de Características e uma fórmula proposicional sobre o conjunto de características. Não são todas as combinações de características válidas em um Modelo de Características, algumas regras devem ser seguidas definindo um conjunto de configurações de características válidas [1];
- Características relacionais, *stakeholders*, restrições transversais, pontos de vinculação e propriedades são documentados em um Modelo de Características. O elemento chave é o Diagrama de Características que corresponde a uma notação gráfica para descrever dependências entre características variáveis [15].

Portanto, o Diagrama de Características é uma representação gráfica e o Modelo de Características pode ser representado de forma textual ou gráfica. Modelos de Características gráficos são compostos de um Diagrama de Características, mas deve-se ter alguma informação adicional (e.g., restrições transversais). A adição de informações torna o modelo mais conciso e próximo da realidade. Contudo, a consistência e a validade desse modelo podem ser verificadas utilizando ferramentas automáticas. A busca por Modelos de Características com representação gráfica, mesmo que implicitamente, deve apresentar Diagramas de Características, mas a busca por Diagramas de Características não necessariamente traz como retorno Modelos de Características.

III. Seleção dos Modelos de Características

A revisão de literatura foi iniciada com 50 artigos relacionados aos temas "Programação Orientada a Características" e "Linha de Produtos de Software", resultado de uma pesquisa *ad-hoc*. Outros artigos foram acrescentados a esse montante por meio da leitura das referências bibliográficas desses artigos (*snowballing*). Ao final, foram identificados, lidos e analisados 197 artigos. Para ampliar a quantidade de artigos, foram utilizadas as fases (Planejamento, Execução e Análise) da técnica revisão sistemática da literatura [30]. Essa técnica consiste em identificar, avaliar e interpretar estudos ou pesquisas relevantes para uma questão de pesquisa. A questão de pesquisa é:

```
Quais são as ferramentas existentes na
literatura que permitem representar
notações de Modelos de Características em
Linhas de Produtos de Software?
```

Com base nessa questão de pesquisa, foi definida a seguinte *string* de busca, sendo utilizada em dois repositórios de trabalhos científicos:

```
("method" OR "notation") AND ("feature
   diagram" OR "feature model") AND
   ("feature oriented" OR "feature-
oriented") AND ("software product line"
 OR "SPL" OR "software product family")
```

Os repositórios foram IEEE (http://ieeexplore.ieee.org) e Scopus (http://www.scopus.com) por causa do fácil acesso e da credibilidade de seus repositórios de artigos/textos científicos. A pesquisa foi realizada nesses repositórios utilizando a *string* de busca no título, no resumo e nas palavras-chave, obtendo 227 resultados, sendo 5 (2,2%) da IEEE e 222 (97,8%) da Scopus. Desse total, foram desconsiderados 4 resultados (1,76%) por estarem repetidos nos dois repositórios e 1 resultado (0,44%)

por não ser artigo. Esses 222 artigos foram agrupados com os 197 artigos encontrados com a revisão da literatura, totalizando 419 artigos. Foram excluídos 25 artigos (5,97%) repetidos o que gerou um montante final de 394 artigos. Essa diferença entre os dois montantes (222 e 197), no qual, apenas 25 artigos repetiram, deve-se a amplitude da pesquisa *ad-hoc*. Quatro pesquisadores realizaram a leitura do título e do resumo desses artigos e dois pesquisadores avaliaram o resultado final. Após a leitura, foram selecionados 53 artigos (13,45%) com indícios de abordagem sobre os temas "notações" e/ou "Modelos de Características", os quais foram lidos na íntegra. Desses artigos, foram selecionados 28 (52,83%) que efetivamente tinham os temas. Para a seleção das notações, foram definidos quatro critérios, os quais foram considerados quando foi feita a leitura completa dos artigos:

- **Seleção de notações gráficas para Modelos de Características**. Esse critério é justificado por ter sido decidida a criação de um catálogo apenas com notações que apresentam modelos com representação gráfica;
- **Proposta mais antiga foi mantida, para notações semelhantes**. Esse critério foi determinado por acreditar que a notação mais antiga apresenta maior contribuição;
- **A fonte primária das notações deve estar acessível**. Esse critério justifica-se, pois, com a descrição total da notação, pode-se realizar análise consistente e correta. Com isso, quando uma notação foi encontrada, mas não se conseguiu acesso a fonte primária, ela não foi considerada;
- **Foram selecionadas notações cujo domínio (problema/conceito) é representado em um único modelo**. Esse deve-se ao fato das críticas apresentadas por alguns autores, no qual, é dito que modelos de características com representação gráfica complexos ocupam muito espaço [34] e o acréscimo de um modelo complementar aguçaria mais ainda essa crítica.

Dos 28 artigos, foram excluídos 13 (46,43%) respeitando os critérios definidos. Ao atender o primeiro critério, foram excluídos 4 artigos (14,29%). Em um desses, foi apresentada a notação FOOM (*Feature-based Object Oriented Modeling*) [2] que consiste em um *template* utilizado para descrever a transformação arquitetural com foco na identificação de características dirigidas ao usuário de uma LPS e não um Modelo de Características. Em outro [41], foi abordada a arquitetura de LPS. No terceiro [32], foi tratada a realização de testes no *design* de uma LPS. No último [42], a notação tratada é apropriada para dar suporte em tempo de execução para a variabilidade ser reconfigurada não sendo voltada para Modelos de Características. Ao atender o segundo critério, foram excluídos 4 artigos (14,29%). Em um desses [44], foi apresentada uma notação que se diferencia em dois tipos de características (*Optional Alternative*, *Optional Or*) da notação GPFT (*Generative Programming Feature Tree*). Esses tipos são considerados como pontos variantes decorrentes das decomposições XOR e OR, com características opcionais. Isso aconteceu em outro artigo [15], em que foram acrescidos três tipos de características. Essas características substituíam os relacionamentos `AND`, `XOR` e `OR` da notação GPFT. No terceiro [34], a notação proposta é semelhante a notação de um outro artigo [13]. No último [4], apesar de apresentar uma inovação às restrições transversais utilizando conceitos, não existe a proposta de uma notação de fato, apenas a adição de um conceito em um modelo existente.

Ao atender o terceiro critério, foi excluído 1 artigo [28] (3,57%) no qual é abordado o *Feature-Oriented Product Line Engineering* (FOPLE). Ao atender o quarto critério, foram excluídos 4 artigos (14,29%), sendo um deles [8] uma notação na qual são necessários um modelo de família e um modelo de restrição. Embora essa proposta expresse as características e os relacionamentos diretos (decomposições), necessita-se de outro modelo para expressar as restrições transversais. Ressalta-se que a notação presente nesse artigo possui pontos distintos das demais, como as *features macros* e o *group element*. Em outro [48], é apresentada a notação FODAcom para modelar sistemas de software. O motivo da sua exclusão foi pela utilização de três modelos distintos (modelo de contexto, de casos de uso e de características) serem utilizados na análise de requisitos e no domínio da aplicação. No terceiro [19], é apresentada a notação UbiFEX, evolução da notação Odyssey-FEX, que possui características de contexto (*Context Features*). No último [3], são utilizados Diagramas de Caso de Uso, Modelo de Processos de Negócio e Modelo de Características.

IV. Elementos de Modelagem dos Modelos de Características

A caracterização do Modelo de Características está dividida nos tipos de estrutura gráfica que ele pode ser representado, nos tipos de características e nos relacionamentos. Os relacionamentos são divididos em decomposições e em restrições transversais. Os Modelos de Características podem ter a estrutura de Árvores ou de Grafos Acíclicos Direcionados (*Directed Acyclic Graph* - DAG). Os tipos de características e os tipos de relacionamentos são apresentados na Table I e na Table II, respectivamente. A classificação foi realizada a partir do artigo no qual a notação foi proposta, quando determinada informação não estava presente ou explicita, a classificação de trabalhos relacionados foi utilizada, por exemplo, a classificação do tipo de estrutura gráfica da notação VFD obtida em [25], um trabalho relacionado.

TABLE I. Tipos de Características

Característica	Descrição
Conceito ou Raiz	Refere-se ao sistema por completo. O conceito/raiz deve existir em Modelos de Características [38].
Mandatória	Trata o conceito de obrigatoriedade no domínio como todo [45]. Corresponde ao núcleo das funções principais do domínio da aplicação em nível do problema e constitui a infraestrutura de domínio [21].
Opcional	Quando habilitada, adiciona valor ao núcleo das características do produto [45]. Corresponde à identificação de algumas funções que podem ser desnecessárias. Uma característica que não aparece nos sistemas não se torna necessariamente opcional [21].
Externa	Oferecida pela plataforma-alvo do sistema. Embora não seja diretamente parte do sistema, ela é importante, pois o sistema a utiliza e depende dela [45].
Abstrata	Utilizada para estruturar o Modelo de Características. No entanto, não tem impacto no nível de implementação. Representa decisões de domínio, mas não afeta a geração de uma variante do sistema [46].

TABLE I. TIPOS DE CARACTERÍSTICAS (CONT.)

Característica	Descrição
Oculta	Faz parte do Modelo de Características, mas não apresenta alteração no nível de implementação. Percebe-se que a característica não pode pertencer ao modelo e, ao invés de excluí-la, ela torna-se oculta. Quando uma característica que possui características-filhas é oculta, as descendentes tornam-se ocultas [33][37].
Variante/ Alternativa	Corresponde a formas alternativas de configurar uma característica mandatória ou opcional [45]. Não possui representação explícita na maioria das notações. Ela não é uma lista das formas alternativas de fazer algo como encontrado nos sistemas em análise, mas uma seleção ou uma revisão delas [21].
De Domínio	Intimamente ligada à essência do domínio da aplicação. Representa funções/conceitos do modelo e corresponde a casos de uso e componentes estruturais concretos. Pode ser considerada como conceito ou característica raiz [35].
Opcional Falsa	Corresponde a uma característica opcional, mas necessita ser obrigatória para respeitar as regras do modelo. Ela é semelhante às características mortas, pois ela não deveria existir em Modelos de Características com configurações legais e bem formados [37].
Morta	Quando uma característica não está presente na configuração válida do Modelo de Características, ela é considerada uma característica morta [5]. Essa característica aparece por causa de erros de modelagem e não deve estar presente em Modelos de Características bem formados [14], podendo ser detectada a partir de expressões lógicas [37].
Clonada	Corresponde a característica que possui cardinalidade maior que 1 ([0..2] ou [1..*], por exemplo). O último caso é especial, em que cada "seleção" de característica é representada uma instanciação. Assim, características-filha de característica clonada são configuradas separadamente para cada instanciação ou seleção. Pode representar um subdomínio dentro de um domínio [46].
De Ambiente Operacional	Representa atributos de um ambiente que uma aplicação do domínio pode utilizar e operar. Por exemplo, tipo de terminal, sistemas operacionais e bibliotecas [35].
De Entidade	Corresponde aos atores do modelo. Entidade do mundo real que atua sobre o domínio. Por exemplo, pode expor a necessidade de uma interface com o usuário [35].
Extrafuncional/ Parâmetro	Oferece informação extra, podendo ser o relacionamento entre um ou mais atributos de uma característica. Informações que agregam valor a característica podem proporcionar ideia de preço ou tempo. Por exemplo, para uma característica mandatória "preço", a característica extrafuncional indica qual é o preço daquela característica [5].
De Tecnologia de Domínio	Representa detalhes de implementação de baixo nível, específicos para o contexto de um domínio. Por exemplo, métodos de navegação em um domínio de aviões [35].
De Técnicas de Implementação	Representa detalhes de implementação de baixo nível, contudo de cunho mais genérico que a característica de tecnologia de domínio. Por exemplo, técnicas de sincronização [35].

TABLE II. TIPOS DE RELACIONAMENTOS

	Identificação	Descrição
Decomposições	AND	Subcaracterísticas são selecionadas [4].
	OR	Uma ou mais subcaracterísticas podem ser selecionadas [4].
	XOR ou Alternativa	Apenas uma subcaracterística pode ser selecionada [4].
	OPT	Subcaracterísticas podem ser selecionadas em um conjunto de subcaracterísticas [3].
	MUX	Nenhuma ou uma subcaracterística pode ser selecionada [3].
	Composição	Uma característica é composta por outras. Denota relação na qual uma característica é parte de outra [35].
	Agregação	Uma característica representa o todo e as partes [35].
	Herança	Há generalização/especialização das características, indicando que as características mais especializadas (características-filha) herdam as peculiaridades de características mais generalizadas (características-mãe) [35].
	Associação	Relacionamento simples entre duas características. Denota algum tipo de ligação entre seus membros. Pode ser nomeada, indicando um tipo específico de ligação [35].
	Alternativo	Relacionamento entre um ponto de variação e suas variantes. Denota a pertinência de uma variante a um determinado ponto de variação. Semelhante à decomposição `XOR` [35].
	Implementado por	Relacionamento entre características de Domínio e características Tecnológicas ou entre características Tecnológicas que se encontram em camadas diferentes. Indica que uma determinada tecnologia é usada para implementar a característica relacionada [35].
	Ligação de Comunicação	Relacionamento entre características de Entidade e de Domínio. Cumpre o mesmo papel do relacionamento de associação entre atores e casos de uso na UML [35].

TABLE I. TIPOS DE RELACIONAMENTOS (CONT.)

	Identificação	Descrição
Restrições Transversais	De Inclusão (*Requires*)	Se a característica `A` *requires* a característica `B` significa que se A é incluída no sistema implica que `B` deve ser incluída nesse sistema [40].
	De Exclusão (*Excludes*)	Se a característica `A` *excludes* a característica `B` significa que ambas características não podem ser parte de um mesmo sistema [40].
	Complexas ou *non-grammar constraints*	Combinação dos operadores lógicos `AND`, `OR` e `NOT`. Por exemplo, a característica A *requires* a característica `B OR` a característica `C OR` característica D [7][4].
	Mathematical	Trata uma restrição transversal entre características extrafuncionais definidas para verificar seus valores, seguir as regras matemáticas dadas e tornar o modelo mais completo e consistente [43].
	Hint	Capturar a informação necessária para dar suporte ao cliente enquanto ele está no processo de decisão. Essa restrição pode apoiar os processos de configurações do cliente e apontar características úteis e adicionais [43].

Alguns desses elementos podem fazer parte de algumas notações e não estar presentes em outras. Por exemplo, as características Externas possuem representação apenas em duas notações ([45] e [9]). Outro exemplo é as características Morta e Opcional Falsa identificadas quando o Modelo de Características passa por uma validação. Além disso, existem decomposições representadas pelas notações utilizando cardinalidade [13]. A cardinalidade foi introduzida nos Modelos de Características para evitar que sistemas não coerentes fossem criados [23] e pode aumentar a expressividade dos modelos por permitir a criação de decomposições resultantes dos valores mínimo e máximo da cardinalidade [13].

V. NOTAÇÕES DE MODELOS DE CARACTERÍSTICAS

Nesta seção, são apresentados um breve histórico de cada notação e os elementos de modelagem que podem ser representados. A primeira notação proposta foi FODA em 1990. Além disso, a notação Odyssey-FEX foi proposta por um grupo de pesquisa brasileiro e está fortemente baseado na notação UML (*Unified Modeling Language*). Uma análise comparativa dessas notações é apresentada na Table III. A ordem de apresentação dos modelos é pelo ano de publicação do trabalho no qual a notação foi proposta (Anexo). São elas:

- **FODA - *Feature-Oriented Domain Analysis* [26]**. Essa notação foi proposta em 1990, sendo a primeira a utilizar conceito de características, e visa ao atendimento de dois principais objetivos: i) desenvolver produtos de um domínio que suportam implementação de novas aplicações; e ii) estar incorporado no processo de desenvolvimento de sistemas;
- **FORM - *Feature-Oriented Reuse Method* [27]**. Essa notação foi proposta em 1998, sendo uma extensão da notação FODA, cuja proposta é cobrir o aspecto de domínio e a engenharia de aplicação, incluindo o desenvolvimento de arquiteturas reutilizáveis e componentes de código. Ela é caracterizada pela construção de um modelo de domínio. Seus principais objetivos são (i) detectar similaridades e diferenças de sistemas de software em um domínio de aplicação e (ii)

utilizar as análises de resultados para desenvolver arquitetura de domínio e componentes;

- **FeatuRSEB [21]**. Essa notação é uma combinação das notações FODA e *Reuse-Driven Software Engineering Business* (RSEB). A notação RSEB é um caso de uso dirigido a processos de reúso sistemático, sendo a variabilidade capturada por Diagramas de Casos de Uso da UML e modelos de objetos com pontos de variação. A notação FeatuRSEB foi proposta em 1998 para expressar a variabilidade das características da LPS;
- **GPFT - *Generative Programming Feature Tree* [11]**. Os Modelos de Características têm sido estudados e adaptados no contexto de *Generative Programming*, tecnologia de programação que visa à automação do processo de desenvolvimento de sistemas para LPS. Essa notação foi proposta em 2000 e é a mais encontrada nos artigos estudados, sendo utilizada para modelar famílias de sistemas quando elas são altamente customizadas/otimizadas;
- **Notação de Svahnberg [45] Posteriormente Chamado VBFD por Schobbens [38]**. Essa notação foi proposta em 2001, sendo uma extensão da notação FeatuRSEB para lidar com os tempos de ligação, que indica quando características podem ser selecionadas. Além disso, ela possui influência da notação FORM com as características representadas por retângulos. Nessa notação, pode-se sugerir princípios de *design* para a seleção de técnicas adequadas para facilitar a uso de variabilidade. As decomposições possuem operadores que fornecem a informação se elas ocorrem em tempo de compilação (*compiletime*) ou em tempo de execução (*runtime*);
- **Notação de CLAUß [9]**. Essa notação foi proposta em 2001, com a percepção da necessidade de uma notação uniforme e consistente. Ela possui a robustez e a aceitação da notação UML com adição de elementos comuns no contexto de análise de domínio expressando variabilidade. Além disso, é capaz de modelar ampla gama de sistemas, mas apresenta limitações para sistemas complexos ou conexos a outros sistemas;
- **Notação de Riebisch [36] Posteriomente Chamado EFD por Schobbens [39]**. Essa notação foi proposta em 2002, após estudos sobre as notações FODA e GPFT em que se percebeu amplo uso de multiplicidade da UML. Foi proposto um Modelo de Características com cardinalidade, tentando abstrair as melhores ideias das notações existentes;
- **Notação de Streitferdt [43]**. Em 2003, com a percepção de que a verificação da consistência em Modelos de Características não era abordada adequadamente, foi proposta uma definição formalizada para Modelagem de Características utilizando OCL (*Object Constraint Language*) e um conjunto de associações e de restrições transversais a serem utilizadas;
- **VFD - *Varied Feature Diagrams* [6]**. Essa notação foi proposta em 2004 e é baseada nas notações FODA e Riebisch (EFD). Por meio de definições e de teoremas, ela possui formalização dos seus relacionamentos entre características utilizando cardinalidade. Pelo fato da notação EFD utilizar cardinalidade apenas nas decomposições `OR` e `XOR`, foi criada uma notação em que os relacionamentos fossem baseados em cardinalidade. As restrições são representadas utilizando cardinalidade;
- **Notação de Dolog e Nejdl [17]**. Essa notação foi proposta em 2004, sendo desenvolvida a partir de mais flexibilidade dos Modelos de Características em relação aos diagramas fornecidos pela UML. Ela possui limitações na modelagem de informações baseadas em aplicações Web. Seu objetivo é apresentar um Modelo de Características que segue os padrões da UML, oferecendo liberdade na variabilidade que esse modelo proporciona;
- **PLUSS - *Product Line Use Case Modeling for Systems and Software Engineering* [18]**. Essa notação, proposta em 2005, é baseada na notação FODA e o seu objetivo é expressar uma decomposição que a notação FODA não apresenta: Adaptador Múltiplo sem usar cardinalidade, pois a cardinalidade reduz a legibilidade. Podem ser expressas decomposições Adaptador Múltiplo e Adaptador Simples (*single adaptor*), semelhantes às decomposições `OR` e `XOR`, respectivamente. O Adaptador Múltiplo é definido como "pelo menos um de muitos" e o Adaptador Simples como "exatamente um de muitos";
- **Notação de Czarnecki [13]**. A utilização de cardinalidade foi motivada para aplicações práticas [12], mas inicialmente desencorajada [11]. Os objetivos dessa notação, proposta em 2004, são apresentar a cardinalidade em Modelos de Características e propor um conceito de estágio de configuração a ser conseguido pela especialização gradual ou por configurações de vários níveis (*multi-level*). Posteriormente, a formalização da cardinalidade baseada em Modelos de Características foi proposta. As restrições transversais ou globais não aparecem na notação, uma característica pode possuir mais de uma característica-mãe, implicando em restrições transversais implicitamente;
- **Notação de Benavides [5]**. Essa notação foi proposta em 2005 e pode ser considerada extensão das notações GPFT e de Streitferdt com a introdução de características extrafuncionais, melhorando-as por permitir relações entre atributos sem a utilização de uma restrição transversal;
- **Notação de Vranic e Snirc [49]**. Essa notação foi proposta em 2006 e é baseada na notação GPFT. Para enquadrar o Modelo de Características às normas da UML, os relacionamentos representados são transformados em relacionamentos de cardinalidade;

- **Odyssey-Fex [35]**. Essa notação foi proposta em 2006 e é utilizada na verificação da consistência do Modelo de Características no desenvolvimento de sistemas críticos. Além disso, heurísticas são estabelecidas para a programação de variabilidades para o Diagrama de Classes da UML. Essa notação é não convencional, pois foge dos padrões propostos pela notação FODA e suas evoluções, seguindo uma linha de modelagem diferente, por criar relacionamentos, tipos de características e representações gráficas diferentes das outras notações apresentadas. Pode-se expressar características de Domínio, de Entidade, de Ambiente Operacional, de Tecnologia de Domínio e de Técnicas de Implementação.

TABLE III. ANÁLISE COMPARATIVAS DAS NOTAÇÕES

#	Notação	Origem	Ano	Estrutura	Características	Decomposições	Restrições Transversais
1	FODA	1ª a usar conceito de característica	1990	Árvore	Mandatória e Opcional	AND, OR	Inclusão (requires) e Exclusão (mutex)
2	FORM	Extensão de FODA	1998	DAG	Mandatória e Opcional	AND, XOR	Inclusão (requires) e Exclusão (mutex)
3	FeatuRSEB	Combinação de FODA e RSEB	1998	DAG	Mandatória e Opcional	AND, XOR, OR	Inclusão (*requires*) e Exclusão (*mutual_exclusion*)
4	GPFT	Não identificada	2000	Árvore	Mandatória e Opcional	AND, XOR, OR	Inclusão (*requires*) e Exclusão (*mutual_exclusion*)
5	Svahnberg (VBFD)	Extensão de FeatuRSEB	2001	DAG	Mandatória, Opcional e Externa	AND, XOR, OR	Inclusão (*requires*) e Exclusão (*excludes*)
6	CLAUß	Combinação de FODA, FORM, Svahnberg e UML	2001	Árvore	Mandatória, Opcional, Alternativa e Externa	OR, XOR	Inclusão e Exclusão
7	Riebisch (EFD)	Melhores ideias das notações existentes	2002	DAG	Mandatória e Opcional	AND, XOR, OR, OPT, MUX	Inclusão (*requires*) e Exclusão (*excludes*)
8	Streitferdt	Não identificada	2003	Arvore	Mandatória, Opcional e Extrafuncional	AND	Inclusão, Exclusão, *Mathematical* e *Hint*
9	VFD	Baseada em FODA e Riebisch	2004	DAG	Mandatória e Opcional	XOR, OR, OPT, MUX (utilizam cardinalidade com quantidade mínima e máxima) AND	Inclusão (*requires*) e Exclusão (*excludes*)
10	Dolog e Nejdl	Segue padrões da UML	2004	DAG	Mandatória e Opcional	AND, OR	Não possui
11	PLUSS	Baseada em FODA	2005	Arvore	Mandatória e Opcional	AND, Adaptador Múltiplo (OR), Adaptador Simples (*single adaptor*) (XOR)	Inclusão (*requires*) e Exclusão (*excludes*)
12	Czarnecki	Não identificada	2004	Árvore	Mandatória e Opcional e com cardinalidade (característica particular dessa notação: sequência de intervalos e pode ser enquadrada como característica clonada)	AND, XOR, OR, OPT, MUX (utiliza cardinalidade)	Não aparecem
13	Benavides	Extensão de GPFT e Streitferdt	2005	Árvore	Mandatória e Opcional e Extrafuncionais	AND, XOR, OR	Inclusão e Exclusão
14	Vranic e Snirc	GPFT	2006	Árvore	Mandatória e Opcional e Clonadas	XOR, OR (representadas pela cardinalidade), AND	Não possui
15	Odyssey-Fex	Não identificada	2006	Árvore	Mandatória e Opcional	Composição, Herança Agregação, Associação, Alternativo, Ligação de Comunicação, Implementado por	Inclusão e Exclusão

VI. DISCUSSÃO SOBRE AS NOTAÇÕES

Dentre as 15 notações apresentadas neste trabalho, três tipos de modelos de características podem ser identificados:

- Apenas funções presentes na implementação da LPS são mencionadas no Modelo de Características;
- Além das funções presentes no primeiro tipo, estão funções que não são parte do sistema, mas o sistema depende delas;
- Possui foco genérico e cuida do que o sistema deve fazer, da lista tecnologias, atores e da arquitetura do sistema. A diferença deste tipo está na forma como o modelo é estruturado e nos elementos que o compõe.

A maioria das notações enquadra-se no primeiro tipo (FODA, FORM, FeatuRSEB, GPFT, Riebisch, Streitferdt, VFD, Dolog, PLUSS, Czarnecki, Benavides e Vranic - total: 12). Essas notações são evoluções de propostas anteriores, por exemplo, a notação FORM é uma evolução da notação FODA. Essas evoluções ocorreram para atender uma necessidade específica, como a adição de um tipo de característica ou relacionamento. Entre as notações que se enquadram nesse tipo de modelo, a notação de Dolog é a única que altera o formato claro de hierarquização e apresenta um formato próximo ao da UML. Contudo, nessas 12 propostas de notação, as alterações entre uma notação e outra procuram aumentar a expressividade para facilitar o entendimento entre cliente e desenvolvedor.

As notações de Svahnberg e de Clauß enquadram-se no segundo tipo de modelos de características, pois são parecidas com as notações do primeiro tipo e acrescentam informações fora do âmbito de implementação, tais como, a tecnologia a ser implementada (explicitada no modelo pela característica Externa) e se a decomposição acontece em tempo de compilação ou execução (*compiletime* e *runtime*), peculiaridade da notação de Svahnberg. Essas notações foram incluídas neste tipo por causa da presença da característica Externa.

A notação Odyssey-FEX enquadra-se no terceiro tipo de Modelo de Características, pois tem enfoque ampliado, diferenciando-se das demais. Nela, as características são mais amplas, tais como, as características de tecnologia de domínio

e de técnica de implementação. Os relacionamentos seguem a mesma linha, tais como, decomposições de associação, de composição e de agregação. Por isso, ela foi classificada como um tipo de Modelo de Características distinto das demais.

Se o Modelo de Características for utilizado para representar, restritamente, funções que os produtos de uma LPS devem conter, recomenda-se uma das notações do primeiro tipo. Se o domínio a ser implementado necessitar de decisões rápidas e em tempo de compilação/execução ou deixar claro no Modelo de Características de uma tecnologia de auxílio na implementação, uma das notações do segundo tipo é recomendado. Se a estratégia de LPS e/ou o domínio for novo para equipe que desenvolverá o sistema, recomenda-se uma das notações do terceiro tipo, por ser recente e estar mais documentada.

VII. Notações vs. Ferramentas para Construção de Modelos de Características

Como auxílio para configuração e gerência de variabilidade em LPS, algumas ferramentas são utilizadas por engenheiros de software e pesquisadores (e.g., SPLOT, FeatureIDE, XFeature, fmp e pure::variants). Nessas ferramentas, o foco é desviado de um desenvolvimento convencional para uma abordagem de desenvolvimento baseada na composição de artefatos e na modelagem do domínio. Elas são utilizadas para modelar/verificar a variabilidade de um produto, utilizando representações gráficas de diagramas/modelos de características. A representação do Modelo de Características varia conforme a ferramenta utilizada. Além da variação gráfica, as várias formas de notações possuem riquezas de representação sintática importantes, que determinam a capacidade de representação semântica da realidade da LPS. Por exemplo, as restrições transversais são representadas utilizando Forma Normal Disjuntiva na ferramenta SPLOT, mas as demais ferramentas não possuem esse tipo de representação.

Para correlacionar e analisar o quanto as notações discutidas neste trabalho estão interligadas e podem ser expressas pelas ferramentas de suporte a configuração e gerência de variabilidade em LPS, uma relação das ferramentas SPLOT, FeatureIDE, XFeature, fmp, pure::variants, e das 15 notações descritas neste trabalho é apresentada na Table III. Na primeira coluna estão relacionadas as 15 notações, nas demais os seis elementos escolhidos para análise sendo tipo de estrutura gráfica, três tipos de características (mandatória, opcional e externa), três tipos de decomposição (`OR`, `XOR` e `AND`) e restrições transversais. Quando, com uma ferramenta, pode ser representado um elemento igualmente é descrito na notação, o nome da ferramenta é apresentado. Quando a notação não possui um elemento, é apresentada a expressão "Não existe". Quando não se pode representar um elemento com atributos das ferramentas, é apresentado "-". Por exemplo, na correlação entre as ferramentas analisadas e a notação FORM, as ferramentas XFeature e FeatureIDE apresentam da mesma forma o tipo de estrutura gráfica, mas não representam a característica mandatória e as características externas e a decomposição `OR`.

As ferramentas FeatureIDE e pure::variants permitem representar mais e menos atributos de notações (39 e 10, respectivamente). A notação de Benavides é a única que teve os oito atributos representados por pelo menos uma ferramenta. As notações de Clauß e Odyssey-FEX tiveram apenas dois atributos relacionados a alguma ferramenta. A única ferramenta que permite representar características externas é XFeature. A ferramenta pure::variants foi relacionada a apenas um atributo (restrições transversais). Em apenas um único caso, uma notação é totalmente expressa por uma das ferramentas. Isso ocorre com a ferramenta XFeature na qual pode-se expressar a notação de Vranic. Apesar de ser o único caso verificado nas ferramentas analisadas, há exemplos em que as notações são quase totalmente representadas pelas ferramentas, como é o caso das notações GPFT e de Benavides em que a maioria das representações estão disponíveis na ferramenta FeatureIDE. Isso porque a característica Externa não existe nessas notações e o único atributo não relacionado (restrições transversais) pode ser representado na ferramenta diferentemente da representação dessas notações.

Assim, engenheiros de software e pesquisadores, após avaliarem qual notação melhor atende suas necessidades, estarão aptos a escolherem uma ferramenta que melhor utilize a notação selecionada, pois, realizar a modelagem sem apoio ferramental, pode demandar tempo e a consistência e a coerência do modelo terão de ser feita de forma manual.

VIII. Trabalhos Relacionados

A quantidade de estudos sobre Programação Orientada a Características segue em curva ascendente desde 1990. Em alguns estudos, são realizadas comparações entre notações encontradas na literatura. Para cada notação, pontos diferentes foram abordados para justificar a sua elaboração. A identificação dos elementos que compõem as notações encontradas em alguns estudos [47] [39] [24] [25] [29] serviram como parâmetro para o desenvolvimento deste trabalho. Nesses estudos, os elementos de modelagem foram classificados em tipos de características, decomposições, restrições transversais e tipos de estrutura do modelo.

A avaliação e a formalização da semântica das notações utilizadas para construir o Modelo de Características foram encontradas em alguns estudos. Nesses estudos, foi apresentada uma comparação semântica entre sete notações (FODA, FORM, FeatuRSEB, EFD, GPFT, PLUSS e a de Van Deursen) [47], foi encontrada outra análise semântica, considerando oito notações (FODA, FORM, FeatuRSEB, EFD, GPFT, PLUSS, VBFD e VFD) [38] e, em dois estudos [6] [39], é apresentada uma avaliação/análise semântica entre 4 (FODA, FeatuRSEB, GPFT, VFD) e 8 (FODA, FORM, FeatuRSEB, GPFT, PLUSS, EFD, VBFD, a de Czarnecki) notações, respectivamente. Com isso, percebe-se que as notações FODA, FORM, FeatuRSEB, GPFT, PLUSS e EFD foram amplamente exploradas nesses estudos.

TABLE IV. NOTAÇÕES VS. FERRAMENTAS PARA CONSTRUÇÃO DE MODELOS DE CARACTERÍSTICAS

Notação	Representação							
	Estrutura Gráfica	Característica			Decomposição			Restrições Transversais
		Mandatória	Opcional	Externa	OR	XOR	AND	
FODA	XFeature e FeatureIDE	-	SPLOT, FeatureIDE e fmp	Não existe	Não existe	FeatureIDE e fmp	FeatureIDE e fmp	pure::variants
FORM	XFeature e FeatureIDE	-	SPLOT, FeatureIDE e fmp	Não existe	Não existe	FeatureIDE e fmp	FeatureIDE e fmp	pure::variants
FeatuRSEB	XFeature e FeatureIDE	-	SPLOT, FeatureIDE e fmp	Não existe	Não existe	-	-	pure::variants
GPFT	XFeature e FeatureIDE	SPLOT, FeatureIDE e fmp	SPLOT, FeatureIDE e fmp	Não existe	FeatureIDE e fmp	FeatureIDE e fmp	FeatureIDE e fmp	pure::variants
VBFT	XFeature	-	SPLOT, FeatureIDE e fmp	XFeature	-	-	-	pure::variants
Notação de Clauß	-	-	-	XFeature	-	-	Não existe	pure::variants
EFD	XFeature e FeatureIDE	SPLOT, FeatureIDE e fmp	SPLOT, FeatureIDE e fmp	Não existe	SPLOT	SPLOT e fmp	FeatureIDE e fmp	pure::variants
Notação de Streitferdt	XFeature e FeatureIDE	SPLOT, FeatureIDE e fmp	SPLOT, FeatureIDE e fmp	XFeature	Não existe	Não existe	FeatureIDE e fmp	pure::variants
VFD	XFeature e FeatureIDE	-	-	Não existe	XFeature e SPLOT	XFeature e SPLOT	XFeature	XFeature
Notação de Dolog	-	-	-	Não existe	-	Não existe	XFeature	Não existe
PLUSS	XFeature e FeatureIDE	SPLOT, FeatureIDE e fmp	SPLOT, FeatureIDE e fmp	Não existe	-	-	FeatureIDE e fmp	pure::variants
Notação de Czarnecki	XFeature e FeatureIDE	XFeature	XFeature	Não existe	SPLOT	SPLOT e fmp	XFeature	Não existe
Notação de Benavides	XFeature e FeatureIDE	SPLOT, FeatureIDE e fmp	SPLOT, FeatureIDE e fmp	XFeature	FeatureIDE, fmp	FeatureIDEe fmp	FeatureIDE e fmp	pure::variants
Notação de Vranic	XFeature e FeatureIDE	XFeature	XFeature	Não existe	XFeature, SPLOT	XFeature, SPLOT e fmp	XFeature	Não existe
Odyssey-FEX	-	-	-	Não existe	-	fmp	Fmp	-

Em dois estudos [24] [25], são realizadas avaliações sobre a formalidade de oito notações (FODA, FORM, FeatuRSEB, GPFT, Svahnberg, EFD, VFD, PLUSS). Em outro estudo [29], é apresentada uma comparação entre os conceitos de ontologia e os elementos de modelagem de 6 notações (FODA, FORM, FeatuRSEB, EFD, GPFT e o PLUSS). Em outro estudo [16], as notações foram avaliadas quanto aos parâmetros não semânticos (e.g., legibilidade, simplicidade e adaptabilidade).

Este trabalho diferencia-se dos demais: i) pela quantidade de notações analisadas (15 notações); (ii) pelo foco na análise das notações, pois a preocupação é descrevê-las e relacioná-las com ferramentas de modelagem; iii) pelo início de elaboração de um catálogo de notações para auxiliar engenheiros de software e pesquisadores a escolher a mais apropriada; iv) pela forma como as notações foram dispostas; e v) pela análise das notações utilizando o mesmo domínio de aplicação, adicionando/removendo características e/ou relacionamentos para atender a especificidade semântica das notações.

IX. CONSIDERAÇÕES FINAIS

O desenvolvimento de sistemas de software tende a ser mais rápido e oferecer produtos com mais qualidade, principalmente a qualidade vista pelo usuário. Assim, a estratégia de desenvolvimento utilizando LPS vem ganhando espaço no mercado por causa das vantagens em relação ao desenvolvimento tradicional. Entre as vantagens, pode-se citar o reúso sistemático e em larga escala. Com essa tendência a modelagem de características torna-se meio para o desenvolvimento de LPSs e a necessidade de escolha de um modelo de características/ferramenta para modelagem faz-se necessária para desenvolvimento eficaz e eficiente de uma LPS.

Neste artigo, foram apresentados os resultados de uma revisão de literatura com uma pesquisa em duas máquinas de busca. Com esses resultados, foram identificadas e descritas 15 notações para Modelos de Características, ampliando os estudos anteriores. A ampla revisão fez com que 28 trabalhos que possuem propostas de notações no contexto de LPS fossem encontrados. Das 15 notações analisadas, foi possível identificar 16 tipos de características e 17 tipos de relacionamentos entre características. O resultado da revisão sugere que entre 2000 e 2005 houve mais interesse em propor notações. Após esse período, é suposto que as necessidades de modelagem passaram a ser melhor representadas utilizando Modelos de Características anteriormente propostos. Pôde-se observar que, com novas notações, novos tipos e representações de características foram propostas, fazendo com que itens anteriormente abstratos fizessem parte da representação.

Foram identificados três tipos de Modelos de Características, um deles com foco nas funções que o sistema deve conter (implementação) e os outros dois aumentando o foco da modelagem, deixando explícito a tecnologia a ser utilizada na implementação. Com essa classificação, a escolha de uma notação que atenda as necessidades torna-se mais direta e objetiva. Com as 15 notações descritas e as cinco ferramentas identificadas, pode-se perceber que algumas ferramentas e notações seguem padrões parecidos, como é o caso da ferramenta FeatureIDE que relaciona 39 atributos. Outras utilizam elementos diferentes, como a ferramenta pure::variants, que possui representação semelhante as notações em apenas um atributo (restrições transversais), e a ferramenta XFeature, que atendeu completamente os atributos relacionados a uma notação (de Vranic).

Dentre as contribuições, pode-se destacar a: i) identificação de um conjunto de notações que representam Modelos de Características; ii) elaboração de um catálogo inicial de notações; iii) reunião de tipos de características e de relacionamentos; iv) relação das notações utilizadas por algumas ferramentas para modelagem de características; e v)

identificação das principais referências bibliográficas sobre propostas para modelagem de características.

Como trabalhos futuros, pode-se citar: i) pesquisas que envolvam linguagens de Programação Orientada a Características para extrair LPSs, modelar e gerar produtos/serviços; ii) realizar estudo similar com representação textual; e vi) propor uma notação correlacionada com uma ferramenta para modelar características de acordo com as necessidades do projeto, modelagem e/ou mercado, absorvendo os melhores pontos das notações apresentadas.

REFERÊNCIAS

[1] Acher, M.; Heymans, P.; Collet, P.; Quinton, C.; Lahire, P.; Merle, P. Feature Model Differences. In: International Conference on Advanced Information Systems Engineering. vol. 1, 16 p., 2012.

[2] Ajila, S. A.; Tierney, P. J. The FOOM Method - Modeling Software Product Lines in Industrial Settings. In: International Conference on Software Engineering Research and Practice, 11p. 2002.

[3] Bae, J.; Kang, S. A Method to Generate a Feature Model from a Business Process Model for Business Applications. In: IEEE International Conference on Computer and Information Technology, pp. 879-874, 2007.

[4] Batory, D. S. Feature Models, Grammars, and Propositional Formulas. In: Software Product Line Conference, v. 3714, pp. 7-20, 2005.

[5] Benavides, D.; Trinidad, P.; Ruiz-Cortés, A. Automated Reasoning on Feature Models. In: International Conference on Advanced Information Systems Engineering, v. 3520, pp.491-503, 2005.

[6] Bontemps, Y.; Heymans, P.; Schobbens, P.-Y.; Trigaux, J.-C. Semantics of FODA Feature Diagrams. In: Workshop on Software Variability Management for Product Derivation, pp. 48-58, 2004.

[7] Boucher, Q.; Classen, A.; Faber, P.; Heymans, P. Introducing TVL, a Text-Based Feature Modelling Language. In: International Workshop on Variability Modelling of Software-Intensive Systems, pp. 159-162, 2010.

[8] Cechticky, V.; Pasetti, A.; Rohlik, O.; Schaufelberger, W. XML-Based Feature Modelling. Lecture Notes in Computer Science, v. 3107, pp. 101-114, 2004.

[9] Clauß, M. Modeling Variability with UML. Young Researchers Workshop on Generative and Component-Based Software e Engineering, pp. 226-230, 2001.

[10] Clements, P.; Northrop, L. M. Software Product Lines: Practices and Patterns. Addison-Wesley, 3th Edition, 563p., 2002.

[11] Czarnecki, K.; Eisenecker, U. W. Generative Programming: Methods, Tools, and Applications. Addison-Wesley, 864p., 2000.

[12] Czarnecki, K., Bednasch, T., Unger, P. Eisenecker, U. W. Generative Programming for Embedded Software: An Industrial Experience Report. In: Conference on Generative Programming and Component Engineering. v. 2487, pp. 156-172, 2002.

[13] Czarnecki, K.; Helsen, S.; Eisenecker, U. Formalizing Cardinality-Based Feature Models and Their Specialization. Software Process Improvement and Practice pp. 7-29, 2005.

[14] Czarnecki, K.; Wasowski, A. Feature Diagrams and Logics: There and Back Again. In: International Software Product Line Conference, pp. 23-34, 2007.

[15] Damasevicius, R.; Štuikys, V. Specification and Generation of Learning Object Sequences for e-Learning Using Sequence Feature Diagrams and Metaprogramming Techniques. In: International Conference on Advanced Learning Technologies, pp. 572-576, 2009.

[16] Djebbi, O.; Salinesi, C. Criteria for Comparing Requirements Variability Modeling Notations for Product Lines. In: Workshop on Comparative Evaluation in Requirements Engineering, pp. 20-35, 2006.

[17] Dolog, P.; Nejdl, W. Using UML-Based Feature Models and UML Collaboration Diagrams to Information Modelling for Web-Based Applications. In: International Conference of UML, pp. 425-439, 2004.

[18] Eriksson, M.; Borstler, J.; Borg, K. The PLUSS Approach - Domain Modeling with Features, Use Cases and Use Case Realizations. In: International Conference Software Product Lines, vol. 3714, pp. 33-44, 2005.

[19] Fernandes, P.; Werner, C. Ubifex: Modeling Context-Aware Software Product Lines. In: International Workshop on Dynamic Software Product Lines, pp. 3-8, 2008.

[20] Figeiredo, E.; Cacho, N.; Monteiro, M.; Kulesza, U.; Garcia, R.; Soares, S.; Ferrari, F.; Khan, S.; Filho, O.; Dantas, F. Evolving Software Product Lines with Aspects: An Empirical Study on Design Stability. In: International Conference on Software Engineering, pp. 261-270, 2008.

[21] Griss, M. L.; Favaro, J.; D'Alessandro, M. Integrating Feature Modeling with the RSEB. In: International Conference on Software Reuse, pp. 76-85, 1998.

[22] Gronniger, H.; Krahn, H.; Rumpe, B.; Schindler, M.; Volkel, S. Text-Based Modeling. In: International Workshop on Software Language Engineering, pp. 1-9, 2007.

[23] Heidenreich, F.; Sánchez, P.; Santos, J.; Zschaler, S.; Alférez, M.; Araújo, J.; Fuentes, L.; Ana Moreira, U. K.; Rashid, A. Relating Feature Models to Other Models of a Software Product Line A Comparative Study of Feature Mapper and VML. In: Special Issue on A Common Case Study for Aspect-Oriented Modeling, pp. 69-114, 2010.

[24] Heymans, P.; Schobbens, P.-Y.; Trigaux, J.-C.; Matulevivius, R.; Classen, A.; Bontemps, Y. Towards the Comparative Evaluation of Feature Diagram Languages. In: Software and Services Variability Management Workshop Concepts, Models and Tools, pp. 1-16, 2007.

[25] Heymans, P.; Schobbens, P.-Y.; Trigaux, J.-C.; Bontemps, Y.; Matulevivius, R.; Classen, A. Evaluating Formal Properties of Feature Diagram Languages. Language Engineering, In: IET Software, pp. 281-302, 2008.

[26] Kang, K. C.; Cohen, S. G.; Hess, J. A.; Novak, W. E.; Peterson, A. S. Feature-Oriented Domain Analysis (FODA) - Feasibility Study. SEE Technical Report CMU/SEI-90-TR-021, 1990.

[27] Kang, K. C.; Kim, S.; Lee, J.; Kim, K.; Shin, E.; Huh, M. FORM: A Feature-Oriented Reuse Method with Domain-Specific Reference Architectures. In: Annals of Software Engineering, pp.143-168,1998.

[28] Kang, K.; Lee, K.; Lee, J. FOPLE - Feature Oriented Product Line Software Engineering: Principles and Guidelines. Pohang University of Science and Technology, pp. 58-65, 2002.

[29] Kang, D.; Song, C.; Baik, K. A Method of Service Identification for Product Line. In: International Conference on Convergence and Hybrid Information Technology, pp. 1040-1045, 2008.

[30] Kitchenham, B. Guidelines for Performing Systematic Literature Review in Software Engineering. EBSE Technical Report, 2.3, Keele University, 2007.

[31] Lucrédio, D. Uma Abordagem Orientada a Modelos para Reutilização de Software. Tese de Doutorado. Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo, 287 p., 2009.

[32] Marinho, F. G.; Andrade, R. M. C.; Costa, P. A. S.; Maia, P. H. M.; Vidal, V. M. P.; Werner, C. Safe Adaptation in Context-Aware Feature Models. In: ACM International Conference, pp. 54-61, 2012.

[33] Mehta, A.; Heineman, G. T. Evolving Legacy System Features into Fine-Grained Components. In: International Conference on Software Engineering, pp. 417-427, 2002.

[34] Michel, R.; Classen, A.; Hubaux, A.; Boucher, Q. A Formal Semantics for Feature Cardinalities in Feature Diagrams. ACM International Conference Series, pp. 82-89, 2011.

[35] Oliveira, R. F. Formalization and Consistency Checking in Variabilities Modeling. Dissertação de Mestrado, COPPE, Universidade Federal do Rio de Janeiro, 133p., 2006.

[36] Riebisch, M.; Böllert, K.; Streitferdt, D.; Philippow, I. Extending Feature Diagrams with UML Multiplicities. In: Conference on Integrated and Process Technology, pp. 23-27, 2002.

[37] Ripon, S.; Azad, K.; Hossain, S. J.; Hassan, M. Modeling and Analysis of Product-Line Variants. In: International Software Product Line Conference, vol. 2, pp. 26-31, 2012.

[38] Schobbens, P. Y.; Heymans, P.; Trigaux, J. C. Feature Diagrams: A Survey and a Formal Semantics. In: IEEE International Requirements Engineering Conference, pp.139-148, 2006.

[39] Schobbens, P.-Y.; Heymans, P.; Trigaux, J.-C.; Bontemps, Y. Generic Semantic of Feature Diagrams. In: Computer Networks, vol. 51, pp. 456-479, 2007.

[40] Segura, S.; Hierons, R. M.; Benavides, D.; Ruiz-Cortés, A. Automated Metamorphic Testing on the Analyses of Feature Models. In: Information and Software Technology, v. 53 pp. 245-258, 2011.

[41] Shen, L.; Peng, X.; Zhao, W. A Comprehensive Feature-Oriented Traceability Model for Software Product Line Development. In: Software Engineering Conference, pp. 210-219, 2009.

[42] Shen, L.; Peng, X.; Liu, J.; Zhao, W. Towards Feature-Oriented Variability Reconfiguration in Dynamic Software Product Lines In: Lecture Notes in Computer Science, pp. 52-68, 2011.

[43] Streitferdt, D.; Riebisch, M.; Philippow, I. Details of Formalized Relations in Feature Models Using OCL. In: IEEE International Conference on Engineering of Computer-Based Systems, pp. 45-54, 2003.

[44] Sun, J.; Zhang, H.; Li, Y. F.; Wang, H. Formal Semantics and Verification for Feature Modeling. In: International Conference on Engineering of Complex Computer Systems, pp. 303-312, 2005.

[45] Svahnberg, M.; Gurp, J. V.; Bosch, J. On the Notion of Variability in Software Product Lines. In: Conference on Software Architecture, pp. 45-55, 2001.

[46] Thum, T.; Kastner, C.; Erdweg, S.; Siegmund, N. Abstract Features in Feature Modeling. In: International Software Product Line Conference, pp. 191-200, 2011.

[47] Trigaux, J. C.; Heymans, P.; Schobbens, P. Y.; Classen, A. Comparative Semantics of Feature Diagrams: FFD vs. vDFD. In: International Workshops on Comparative Evaluation in Requirements Engineering, pp. 36-47, 2006.

[48] Vici, A. D.; Argentieri, N.; Mansour, A.; D'Alessandro, M.; Favaro, J.; FODAcom: An Experience with Domain Analysis in the Italian Telecom Industry. In: International Conference on Software Reuse, pp. 166-175, 1998.

[49] Vranic, V.; Snirc, J. Integrating Feature Modeling into UML. NoDE, Erfurt, Germany, pp. 3-15, 2006.

[50] Wang, H. H.; Li, Y. F.; Sun, J.; Zhang, H.; Pan, J. Verifying feature models using OWL. Web Semantics: Science, Services and Agents on the World Wide Web, vol. 5, pp.117-129, 2007.

[51] Young, T. Using AspectJ to Build a Software Product Line for Mobile Devices. Tese de Mestrado, Universidade de Britânica de Columbia, 2005.

ANEXO - DIAGRAMAS DE CARACTERÍSTICAS DAS NOTAÇÕES ANALISADAS