UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS – ICEx
DEPARTAMENTO DE CIÊNCIAS DA COMPUTAÇÃO

Thalisson Vinicius Lauton da Silva

ANÁLISE COMPARATIVA DA FERRAMENTA SPLOT PARA LINHA DE PRODUTOS DE SOFTWARE

Belo Horizonte
2012 / 1º Semestre

THALISSON VINICIUS LAUTON DA SILVA

ANÁLISE COMPARATIVA DA FERRAMENTA SPLOT PARA LINHA DE PRODUTOS DE SOFTWARE

Monografia apresentada ao Curso de Especialização em Informática do Departamento de Ciências Exatas da Universidade Federal de Minas Gerais, como requisito parcial para a obtenção de título de Especialização em Ciência da Computação

Área de Concentração>: Engenharia de Software

Orientador :Prof. Dr. Eduardo Magno Lages Figueiredo

Belo Horizonte
2012 / 1º semestre

AGRADECIMENTOS

Agradeço primeiramente a Deus por todas as oportunidades que me foram dadas na vida. Agradeço a minha mãe Judith Silva Lauton pessoa humilde e trabalhadora fonte de toda minha inspiração na vida. Agradeço aos meus amigos por me proporcionarem momentos de alegria em tempos difíceis. E finalmente agradeço ao meu orientador Eduardo M. L. Figueiredo pela paciência e colaboração com estre trabalho.

RESUMO

Com a intenção de atender às novas demandas na atividade de desenvolvimento de software como a diminuição de custos, esforço e tempo de produção, novos métodos que favoreçam o reuso de artefatos de software têm sido propostos, como a linha de produtos de software. Esse método consiste basicamente na implementação de produtos com alto grau de similaridade a partir do compartilhamento de um conjunto de artefatos. Linha de produtos tem apresentado uma grande aceitação no ambiente corporativo. Este trabalho realiza um estudo de caso com a ferramenta SPLOT, que foi recentemente desenvolvida para apoiar o desenvolvimento de linha de produtos de software. O objetivo deste trabalho é fazer uma análise comparativa da ferramenta SPLOT com outras três ferramentas utilizadas no suporte às diversas etapas do desenvolvimento de uma linha de produto de software.

Palavras-chave: linha de produtos de software, modelo de característica, ferramentas, engenharia de software, reuso de software.

ABSTRACT

To fit new demand of software development activities such as cost reduction, effort and production time, new methods, such as software product lines, have been proposed to support reuse of software artifacts. A software product line consists of implementing a set of products with a high degree of similarity based on a basic set of shared artifacts. Product line has gained great acceptance in the corporate environment. This paper conducts a case study with the SPLOT tool, which has recently been developed to support the development of software product lines. The objective of this study is a comparative analysis of SPLOT with three other tools used to support several stages of software product line development.

Keywords: Software product lines, feature model, tools, software engineering, software reuse.

LISTA DE FIGURAS

LISTA DE SIGLAS

API	Application Programming Interface
BDD	Binary Decision Diagrams
CNF	Conjuctive Normal Form
EMF	Eclipse Modeling Framework
FMP	Feature Model Plugin
FODA	Feature Oriented Domain Analysis
GEF	Graphical Editing Framework
GNU	General Public License
HTML	HyperText Markup Language
ICRS	International Conference on Software Reuse
IDE	Integrated Development Environment
LPS	Linha de Produtos de Software
SEI	Software Engineering Institute
SPL	Software Product Lines
SPLOT	Software Product Lines Online Tools
XML	Extensible Markup Language
XSL	Extensible Stylesheet Language
XSLT	Extensible Stylesheet Language Transformation

SUMÁRIO

1 INTRODUÇÃO

Com os desafios enfrentados pelo desenvolvimento de software como a busca pela redução de custos, esforço, redução do *time-to-market* e pelo ganho de produtividade na produção de software, a reutilização de código é hoje uma realidade. Abordagens de reuso de software ganharam força na década de 80 através de iniciativas de componentização de software **Erro! Fonte de referência não encontrada.** e do desenvolvimento orientado a objeto. Estas iniciativas foram fundamentais para fomentar discussões sobre as vantagens e desvantagens da reutilização de software nas diversas etapas do processo de desenvolvimento. Em particular o reuso de software colabora para um aumento considerável na satisfação dos clientes após sua adoção **Erro! Fonte de referência não encontrada.**.

A crescente necessidade de um desenvolvimento cada vez mais ágil de sistemas de software cada vez maiores e mais complexos resultou em um aumento significativo nas exigências. Exigências essas aplicadas sobre os requisitos que afetam diretamente os usuários finais dos sistemas, os chamados requisitos não funcionais. Estes requisitos podem ser classificados como fatores de qualidade, por exemplo, a disponibilidade, a tolerância a erros, a usabilidade, entre outros.

A partir do surgimento do conceito de Linha de Produtos de Software (LPS) **Erro! Fonte de referência não encontrada.**, o desenvolvimento de famílias de sistemas passou a ser mais amplamente difundido. Famílias de sistemas surgem em situações nas quais o mesmo sistema é utilizado por diferentes clientes sendo necessárias apenas pequenas modificações para atender os requisitos específicos de cada cliente. Com isso o processo de desenvolvimento de software começa a ter um novo enfoque diferente do utilizado tradicionalmente, visando o favorecimento do reuso e o cumprimento de exigências específicas dos clientes. Com o conceito de LPS, o reuso que viabiliza a diminuição do esforço utilizado no desenvolvimento pode ser atingido. Outras vantagens da adoção de uma LPS são a utilização de software previamente testado e a obtenção de mais tempo para dedicar às exigências dos requisitos não funcionais.

O conceito de desenvolvimento de software em famílias consiste basicamente na criação e manutenção de uma linha de produtos de software que servirá como base para

a produção de conjuntos de sistemas relacionados a partir de artefatos comuns **Erro! Fonte de referência não encontrada. Erro! Fonte de referência não encontrada.**. Sendo assim, o foco é na produção de coleções de produtos, onde todos são pertencentes a essa mesma família que é desenvolvida a partir de um conjunto de artefatos básicos. A utilização desse modelo objetiva melhorias quantitativas em produtividade explorando as similaridades e controlando as variabilidades dos membros desses conjuntos de sistemas.

É comum nos dias atuais encontrar empresas que mantém ou desenvolvem produtos de software de um mesmo segmento ou domínio. Esse alto grau de similaridade dentre esses produtos de software possibilita a utilização do modelo de linha de produtos de software trazendo consigo melhorias consideráveis em seus processos de produção. Podemos assim dizer que a abordagem de linha de produtos de software permite que um conjunto de aplicações similares seja desenvolvido de forma mais eficiente por meio do reuso de módulos da aplicação.

A implantação de uma linha de produtos de software em uma empresa de desenvolvimento de software envolve minuciosos processos, como o de levantamento dos requisitos. Em busca da identificação de facilitadores na implantação ou possíveis problemas na mesma, uma análise de domínio é realizada nos produtos desenvolvidos pela empresa em questão a fim de modelar esse domínio e identificar características variáveis e compartilhadas entre esses produtos. Depois de realizada a análise de domínio, muitos problemas no processo de implantação da LPS poderão ser evitados.

Este trabalho apresenta uma descrição das principais propriedades das ferramentas XFeature **Erro! Fonte de referência não encontrada. Erro! Fonte de referência não encontrada.**, fmp **Erro! Fonte de referência não encontrada. Erro! Fonte de referência não encontrada.** e pure::variant **Erro! Fonte de referência não encontrada. Erro! Fonte de referência não encontrada.**. Essas descrições servirão como base na comparação realizada dessas ferramentas com a ferramenta SPLOT (*Software Product Lines Online Tools*). Essa última ferramenta é baseada em internet e oferece uma extensa variedade de serviços para usuários finais e seus pesquisadores. Este trabalho também apresenta uma análise descritiva da ferramenta SPLOT, assim como, um estudo de caso realizado com a ferramenta baseado na linha de produtos EShop **Erro! Fonte de referência não encontrada.**. Através das informações

adquiridas com a análise e com o estudo de caso foi possível efetuar a comparação entre a ferramenta SPLOT e as demais citadas no referente trabalho.

O restante do trabalho está organizado da seguinte forma. O Capítulo 2 apresenta uma fundamentação teoria sobre linha de produtos de software (Seção 2.1), modelo de características (Seção 2.2) e análise de domínio orientado a características (Seção 2.3). O Capítulo 3 apresenta uma análise das ferramentas XFeature (Seção 3.1), fmp (Seção 3.2), pure::variantes (Seção 3.3) e SPLOT (Seção 3.4). O Capítulo 4 apresenta a metodologia utilizada na execução do estudo de caso com as ferramentas (Seção 4.1), a descrição do estudo de caso realizado com a ferramenta SPLOT baseado na linha de produtos EShop **Erro! Fonte de referência não encontrada.** (Seção 4.2), uma breve descrição sobre a execução do estudo de caso com as outras ferramentas citadas no trabalho também baseado na linha de produtos EShop (Seção 4.3) e por fim a comparação entre os resultados encontrado na modelagem com as ferramentas (Seção 4.4). O Capítulo 5 apresenta uma conclusão sobre os estudos realizados, juntamente com a sugestão de trabalhos futuros (Seção 5.1).

2 FUNDAMENTAÇÃO TEÓRICA

Nesse capítulo será apresentada a informação necessária para a correta leitura e entendimento deste trabalho. Alguns dos conceitos relacionados ao contexto do modelo de LPS serão apresentados de forma concisa buscando a fácil compreensão.

2.1 Linha de Produtos de Software – LPS

O termo Linha de Produto de Software (LPS) é utilizado pela comunidade brasileira e se originou do termo internacionalmente conhecido como *Software Product Line* (SPL). Uma LPS pode ser considerada como um conjunto pré-definido de sistemas de software que compartilham de um mesmo conjunto de características que são utilizadas como base para seu desenvolvimento. Essas características compartilhadas por sistemas de um mesmo nicho de mercado são responsáveis pela satisfação das necessidades específicas desses clientes.

Em outras palavras conforme Cohen **Erro! Fonte de referência não encontrada.**, a principal definição de SPL é a de Clements e Northrop, onde eles afirmam que:

> "Uma linha de produto de software é um conjunto de sistemas que usam software intensivamente, compartilhando um conjunto de características comuns e gerenciadas, que satisfazem as necessidades de um segmento particular de mercado ou missão, e que são desenvolvidos a partir de um conjunto comum de artefatos principais e de uma forma preestabelecida"**Erro! Fonte de referência não encontrada.**.

Contudo uma Linha de Produto de Software pode ser considerada também como um *framework* que objetiva o desenvolvimento de produtos focados em atingir um mercado específico e com uma base comum de artefatos.

A implementação de um módulo para um determinado sistema é estendida aos demais sistemas desenvolvidos em um mesmo conjunto sem a necessidade de sua re-implementação para cada um dos sistemas existentes, promovendo com isso alto grau de reuso. Com essa reutilização constante há um ganho considerável na detecção precoce dos erros pelos inúmeros testes realizados com o mesmo módulo em cada reutilização. A consequência do reuso desses artefatos nos diferentes sistemas do

conjunto é que teremos módulos de qualidade superiores a cada reutilização, além de uma redução de custos no desenvolvimento e do *time-to-market*.

Técnicas como a compilação condicional, programação orientada a aspectos (POA) **Erro! Fonte de referência não encontrada.**, componentização **Erro! Fonte de referência não encontrada.**, *Extension Points* do Eclipse, entre outras são utilizadas na implementação de LPS. Através destas técnicas é possível obter ganhos no processo de desenvolvimento até o lançamento do produto no mercado.

O modelo de linhas de produto de software propõe algumas vantagens que foram mapeadas por Cohen **Erro! Fonte de referência não encontrada.**. Tais vantagens podem ser caracterizadas como benefícios tangíveis, ou seja, benefícios possíveis de serem medidos em termos métricos, e benefícios intangíveis, benefícios esses que não podem ser medidos em termos métricos, mas são relatados por partes envolvidas. Cohen **Erro! Fonte de referência não encontrada.** explora com mais detalhes a classificação realizada por ele.

A fim de demonstrar as inúmeras vantagens propostas pelo modelo, um estudo de caso foi realizado por Cohen **Erro! Fonte de referência não encontrada.** contemplando todas as fases do processo de implantação do modelo. Ao final do estudo de caso são apresentados os resultados da implantação como a queda de 65% para 20% da participação do software no custo do sistema, redução do número de pessoas trabalhando no desenvolvimento de um mesmo projeto, redução do *time-to-market* de anos para meses e aumento na qualidade dos sistemas desenvolvidos, assim como, aumento da satisfação do cliente.

Podemos citar o *Windows Vista* como um exemplo de LPS, partindo do princípio de que esse sistema computacional possui várias versões mantendo uma mesma base arquitetural. Nessa mesma base arquitetural são efetuadas adições de componentes ou a remoção dos mesmos, gerando assim as diversas versões existentes através das alterações sofridas. Sendo assim, em uma linha de produto há sempre uma base arquitetural genérica, ou seja, comum a todos os produtos de uma mesma linha.

2.2 Modelo de Característica (*Feature Model*)

Uma característica (*feature*) é qualquer aspecto proeminente e distintivo ou característica que é visível a várias partes interessadas e é usada para capturar funcionalidades comuns ou variáveis entre sistemas de uma mesma família de produtos. Existe também uma segunda definição proposta por Kang e colegas **Erro! Fonte de referência não encontrada.**:

> "As características são atributos de sistema que afetam diretamente o usuário final" **Erro! Fonte de referência não encontrada.**.

O conceito de características surgiu originalmente através do método Análise de Domínio Orientado a Características (FODA) **Erro! Fonte de referência não encontrada.**. É através do modelo de característica que é possível representar toda e qualquer similaridade entre os recursos compartilhados pelas famílias do sistema, assim como, suas dependências e suas diferenças de maneira simples e hierárquica **Erro! Fonte de referência não encontrada.**.

Através do modelo de característica podemos visualizar as principais características comuns e variáveis de uma LPS. A característica pode ser definida como uma propriedade chave de um determinado produto, relevante para os *steakholders*, mas visível para os usuários. A seleção das características é o que define cada um dos produtos da linha. Essa seleção pode tomar por ser realizada com base nos serviços a serem oferecidos, das tecnologias empregadas, do tipo de rede utilizado, do nível de segurança requerido, do tipo de interface, das técnicas de implementação, do ambiente operacional onde será implantado o sistema, dentre outros.

No contexto da LPS, um modelo de característica representa a própria linha de produto. Um modelo de característica é composto basicamente por três elementos principais, são eles **Erro! Fonte de referência não encontrada.**:

- Diagrama de Característica: Possibilita a visualização hierárquica das características padrão de uma família de sistemas em um mesmo domínio e seus relacionamentos. Essas características podem ser classificadas como obrigatórias (tem de estar presente em todos os membros da LPS), opcionais (pode ou não estar presente em um membro da LPS) ou alternativas (conjunto de características das quais se escolhe uma ou mais através de indicação), além de poder existir relacionamentos de especializações entre as características.

- Regras de Composição: É responsável pela validação das combinações de características, ou seja, garante a correta semântica entre as combinações realizadas.
- Análise Racional: São recomendações indicando a melhor situação para que uma determinada característica seja selecionada ou não.

Através da utilização de uma ferramenta específica e da aplicação do conceito de características, para que seja criado um modelo de características é necessário que as características relevantes de um sistema de software sejam documentadas nesse modelo.

2.3 Análise de Domínio Orientado a Características - FODA

O método FODA é considerado um dos métodos de análise de domínio mais maduro, documentado e conhecido por sua descrição detalhada dentre os métodos de mesma função. Além disso, o método tem contribuído com a análise orientada a modelo por permitir o uso de visões complementares de um domínio a fim de transmitir informações mais completas sobre ele **Erro! Fonte de referência não encontrada.**. Basicamente, a análise de domínio visa obter informações de forma investigativa sobre os sistemas de software que compartilham características em comum. Com base nas informações adquiridas através dessa análise, é delimitado o escopo da linha de produtos e suas características são representadas.

O método FODA foi desenvolvido no *Software Engineering Institute* (SEI) em 1990 e visa uma investigação sistêmica do domínio da aplicação. O objetivo do SEI é identificar as importantes variáveis presentes no domínio e representá-las graficamente. Por ter sido o primeiro método proposto para esse fim, o método FODA hoje é a base para as demais variações que também fazem uso da estrutura baseada em árvore.

O modelo FODA é utilizado fundamentalmente no processo de modelagem das características. Após a devida identificação e classificação das características, suas relações e dependências serão representadas através desse modelo, tornando mais evidente e facilmente interpretáveis tais informações.

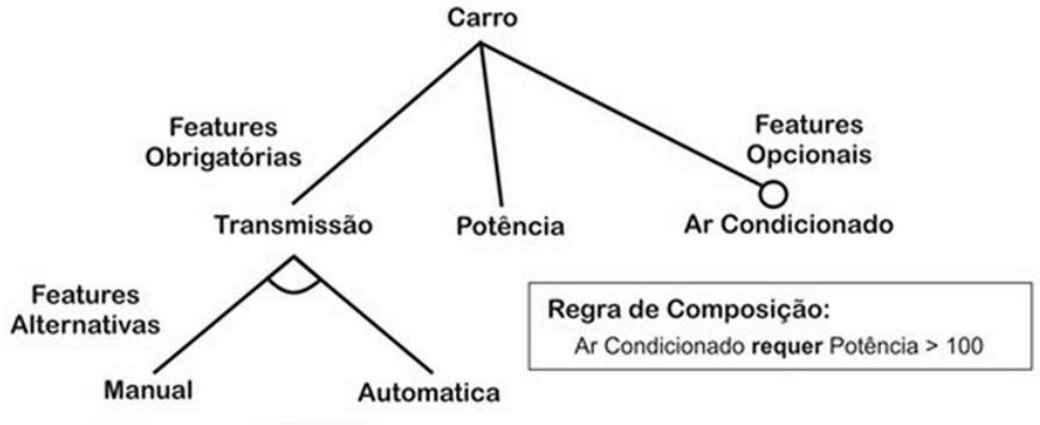

Figura 1. Modelo de Características para LPS de Carro no Modelo FODA [26]

No exemplo representado na Figura 1 podemos observar que no modelo FODA as características são representadas por textos e interligadas por arestas que representam suas variabilidades. Demais dependências existentes no modelo como agregações e composições são representadas através de quadros simples.

Na representação do exemplo da Figura 1 através do modelo FODA, podemos observar que o nó raiz (Carro), representa a indicação do domínio referente ao modelo representado. A aresta mais a esquerda representa que a transmissão é uma característica obrigatória. No caso dessa característica em específico o cliente tem a obrigatoriedade de escolher entre as opções de transmissão (manual ou automática). A aresta ligada à característica potência, também representa a obrigatoriedade da mesma. No caso da aresta ligada à característica ar condicionado, representada com um circulo vazio na extremidade, representa que a mesma é opcional. Contudo para a escolha da característica opcional ar condicionado, antes é necessário satisfazer a regra de composição (potência > 100) existente no ambiente **Erro! Fonte de referência não encontrada.**.

3 FERRAMENTAS PARA LPS

Neste capítulo será apresentado um levantamento de algumas ferramentas utilizadas para LPS. O objetivo é identificar suas particularidades e tornar mais fácil a escolha da ferramenta ideal através da comparação entre as mesmas levando em consideração os objetivos específicos para adoção de uma LPS.

3.1 XFeature

O *XFeature*[1] é um *plugin* para o Eclipse desenvolvido originalmente para aplicações espaciais, mas que é utilizado em vários outros contextos já que a ferramenta é desprovida de restrições de direitos de cópia. O *XFeature* surgiu com o objetivo de demonstrar o conceito de uma ferramenta que tivesse a capacidade de fazer com que as etapas de modelagem e configuração de artefatos reusáveis passassem a ser processos automatizados. Por outro lado, é objetivo também da ferramenta permitir a customização do meta-modelo e do conjunto de características semelhantes da linha de produtos trabalhada.

Essa ferramenta teve sua primeira versão lançada em 2005 se tornando então uma ótima opção no suporte à modelagem do domínio. Atualmente, o site do projeto indica que o mesmo encontra-se na versão 2.1.2 sendo atualizada periodicamente, ela foi desenvolvida pela *P&P Software*, empresa que se originou no *Institute of Automatic Control Laboratory ETH (Swiss Federal Institute of Technology).*

A ferramenta é o*pen source* licenciado sob a *General Public Licence* (GNU) e mantém seu código fonte disponível a fim de receber contribuições de seus usuários. Tais atualizações podem vir a ser incorporadas em seu projeto oficial. Não existem registros do uso comercial da ferramenta, o que a configura como uma ferramenta acadêmica, mas que também é utilizada de forma corporativa.

[1] http://www.pnp-software.com/XFeature/

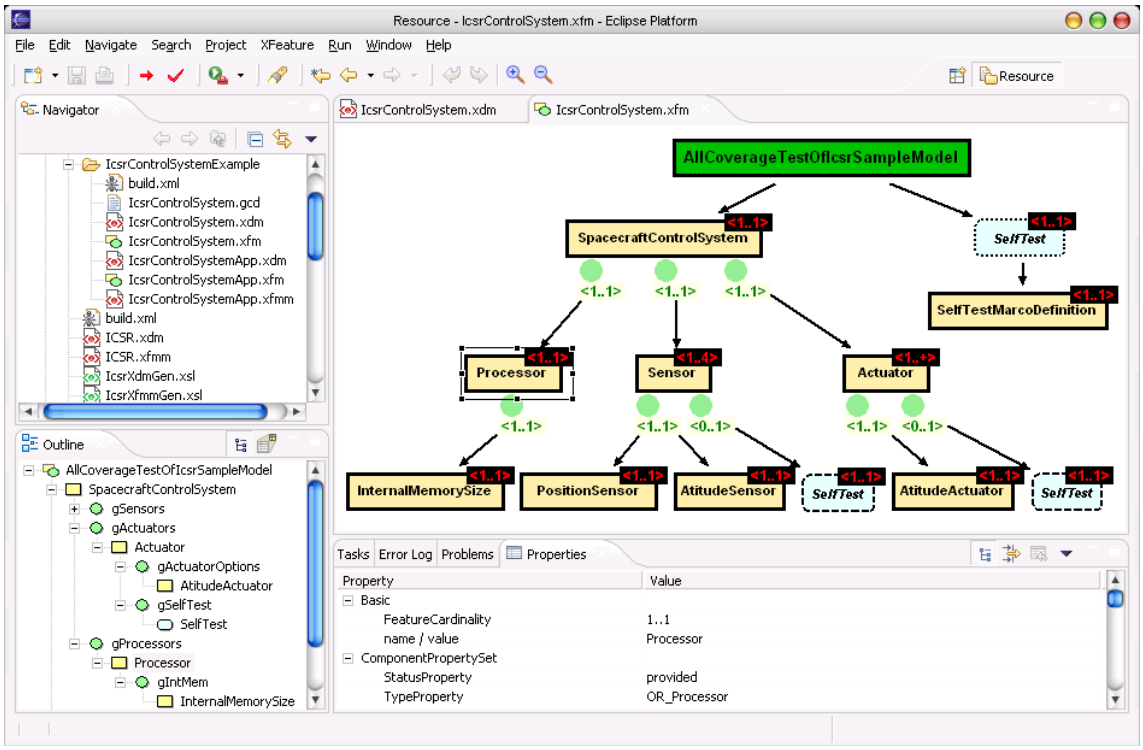

Figura 2. Interface do *XFeature* [6]

Podemos observar na Figura 2 algumas funcionalidades da ferramenta *XFeature*, como o editor de propriedades das características na parte inferior da imagem, a características raiz no topo da árvore, características opcionais que se encontram pontilhadas e as características obrigatórias em retângulos amarelos, todas agrupadas na parte central da imagem.

A implementação da ferramenta usou como base a versão 3.3 do Eclipse, tendo seu editor também baseado no GEF (*Graphical Editing Framework*) que oferece apenas o formato em árvore para visualizar e editar os seus modelos como pode ser observado na Figura 2. A ferramenta faz o uso do formato XML para armazenamento do modelo e de transformações XSL. A escolha da configuração do novo modelo a ser gerado no *XFeature* é o primeiro passo no processo. A ferramenta disponibiliza quatro opções de configuração são elas: FD, FMP, *SimplifiedICSR* ou ICSR. Após esta etapa, é criado o meta-modelo do domínio chamado pela ferramenta de *famliy model*. Finalmente, após a validação do meta-modelo pela configuração utilizada, dois outros meta-modelos serão gerados pelo usuário: (i) o meta-modelo de aplicação utilizado como configuração dos

modelos que contêm as instâncias da LPS e (ii) o meta-modelo de *display* que define os elementos visuais a serem apresentados no editor da ferramenta.

O *XFeature* é considerado uma ferramenta extremamente flexível por permitir um maior controle do usuário na escolha dentre as configurações suportadas, não fazendo com que o mesmo se sinta preso a uma configuração pré-definida. Após todo o processo e a obtenção dos novos modelos de instâncias da linha de produto, eles podem ser utilizados como entradas para outras ferramentas após sua validação sobre seu meta-modelo definido, a utilização desses modelos como entradas se restringe a ferramentas que possuem o *Configuration knowledge* para gerar os produtos finais.

3.2 Feature Modeling Plugin (fmp)

O *Feature Model Plugin* (fmp) encontra-se atualmente completo em sua versão 0.6.6, não sendo mais atualizado. Existe ainda uma versão 0.7.0 da ferramenta em desenvolvimento disponível no seu site oficial[2]. É uma ferramenta que suporta a modelagem de domínio e foi desenvolvido pelo *Generative Software Development Lab*, da Universidade de Waterloo como um *plugin* do Eclipse. Essa ferramenta utiliza como base o *Eclipse Modeling Framework*, porém ela não permite que as características e os artefatos das linhas de produtos sofram um mapeamento mínimo necessário para a geração dos produtos.

A ferramenta é *open source* e mantém seu código fonte disponível em seu site a fim de receber contribuições de seus usuários. Tais contribuições podem vir a ser incorporadas em seu projeto oficial. Ela é uma ferramenta com características puramente acadêmica sendo utilizada no auxílio da produção dos planos de implantação de uma LPS. Não foram encontrados registros de uso comercial da mesma.

[2] http://gsd.uwaterloo.ca/fmp

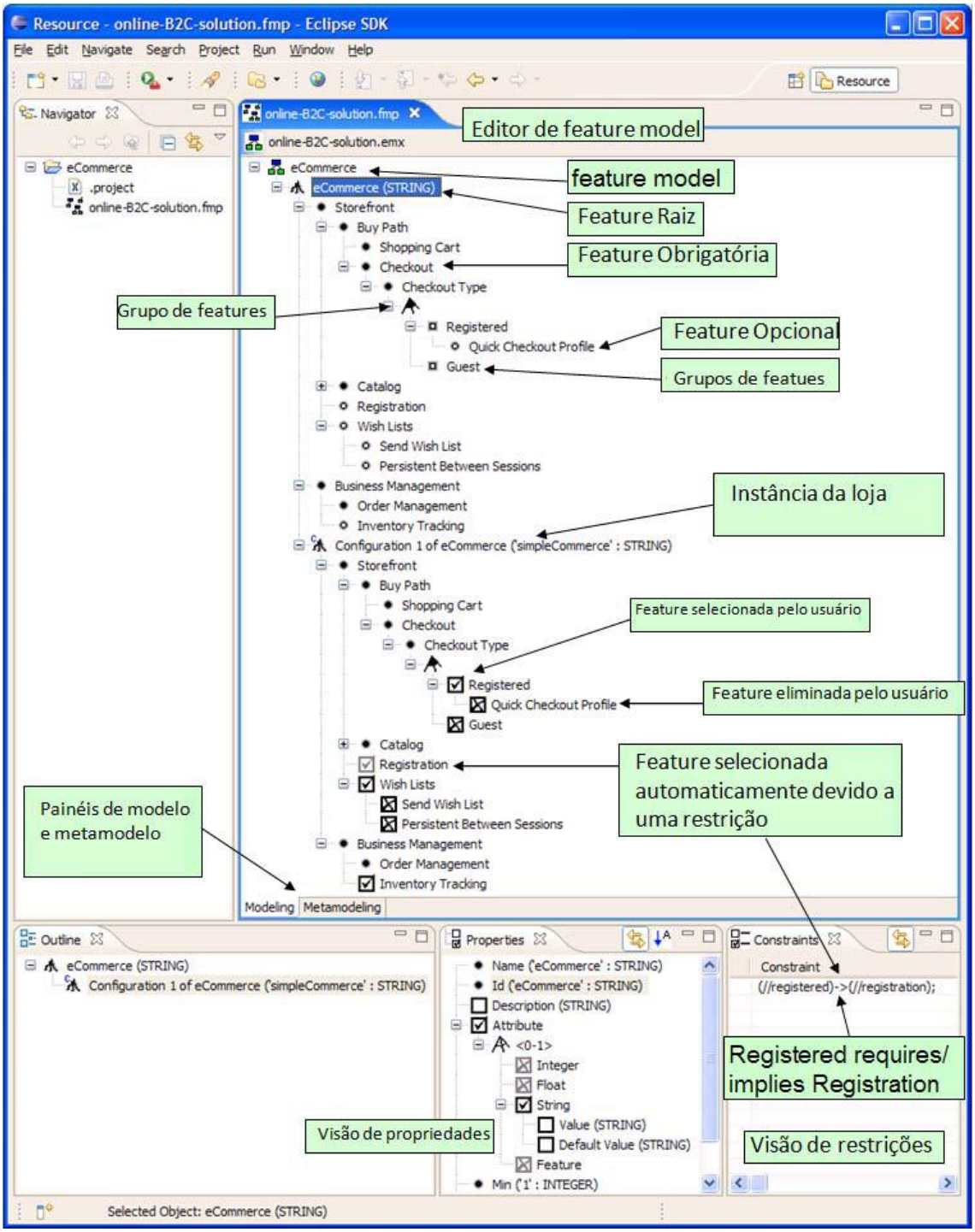

Figura 3. Interface fmp [6]

O fmp permite, dentre outras coisas, a edição dos atributos do meta-modelo através da aba de propriedades. Ela permite também uma configuração de restrições (*constraints*) sobre o modelo de características que podem ser geradas sobre sua

característica não permitindo certas ações do usuário. Somente o formato em estilo "árvore de diretórios" é disponibilizado para a visualização e edição do modelo de características manipulado. A Figura 3, mostra os recursos disponibilizados pela ferramenta como a seleção de característica pelo usuário ou automática, as características obrigatórias, opcionais, grupos de características, instâncias, a edição de propriedades da característica, dentre outras.

O fmp facilita a modelagem de suas características pelo usuário ao permitir sua modelagem baseada em cardinalidade. Através desse estilo de modelagem o usuário tem total controle sobre a definição das cardinalidades de característica e de grupo, de atributos de característica, de referências e anotações tornando o modelo um modelo customizado. A ferramenta utiliza o formato XML (*Extensible Markup Language*) para registrar os modelos criados pelo modelo de características permitindo assim que geradores que fazem uso do formato XSLT (*Extensible Stylesheet Language Transformation*) tenham a capacidade de carregar esses modelos.

A integração do fmp é facilitada por ter sido desenvolvida como um *plugin* bem definido, baseado na versão do Eclipse 3.2 e faz uso do EMF (*Eclipse Modeling Framework*). Assim ela possui recursos de geração de código na definição de seu meta-modelo do modelo de características. É uma ferramenta de notação simples, mas pouco intuitiva. Ela é flexível a ponto de permitir a importação de um modelo por outro, mas pode confundir o usuário nas definições de instâncias do modelo de características com as definições do próprio modelo de características encontrado no mesmo modelo.

3.3 pure::variants

O pure::variants **Erro! Fonte de referência não encontrada.** é uma ferramenta utilizada no suporte à modelagem, implementação, implantação e ainda no desenvolvimento durante as atividades de análise. Ela foi desenvolvida pela *pure-systems GmbH*, empresa criada pelo Instituto *Otto-von-Guericke-Universität Magdeburg* e pelo *Fraunhofer Instituts Rechnerarchitektur und Softwaretechnik* como um *plugin* para o Eclipse na versão 3.3.

O pure::variants teve seu início como uma ferramenta com características puramente acadêmicas e encontra-se em desenvolvimento até hoje. O principal objetivo

é o apoio no desenvolvimento de software embarcado. Entretanto, nos dias de hoje podemos nos deparar com sua utilização em vários outros ambientes como, por exemplo, a indústria automotiva. A ferramenta é comercial.

O pure::variants permite efetuar o mapeamento entre características e artefatos da linha de produtos. Com esse mapeamento é efetuada a criação dos produtos, isso porque a ferramenta trabalha com funcionalidades como a *configuration knowledge.* Esse mapeamento permitido através da ferramenta é a elaboração de um conjunto de regras que tem o objetivo de definir os artefatos constituintes de cada um dos produtos criados. Sendo assim, esse mapeamento acontece entre o modelo de características e os artefatos da LPS. O *pure::variants* funciona através de uma arquitetura cliente-servidor, tendo como cliente o *plugin* do IDE Eclipse utilizado pelo usuário e como servidor um aplicativo C++ rodando em *background.*

Por se tratar de uma ferramenta comercial, o interessado teria como custo adicional a aquisição da licença em uma de suas versões *developer, professional* ou *enterprise*, que da direito a instalação da ferramenta em uma máquina. A quantidade de licenças a serem adquiridas também é um fator muito influente no preço final a ser investido. Encontra-se disponível na internet uma versão gratuita com algumas restrições em sua utilização que pode ser utilizadas para testes da ferramenta sendo proibido seu uso para fins comerciais.

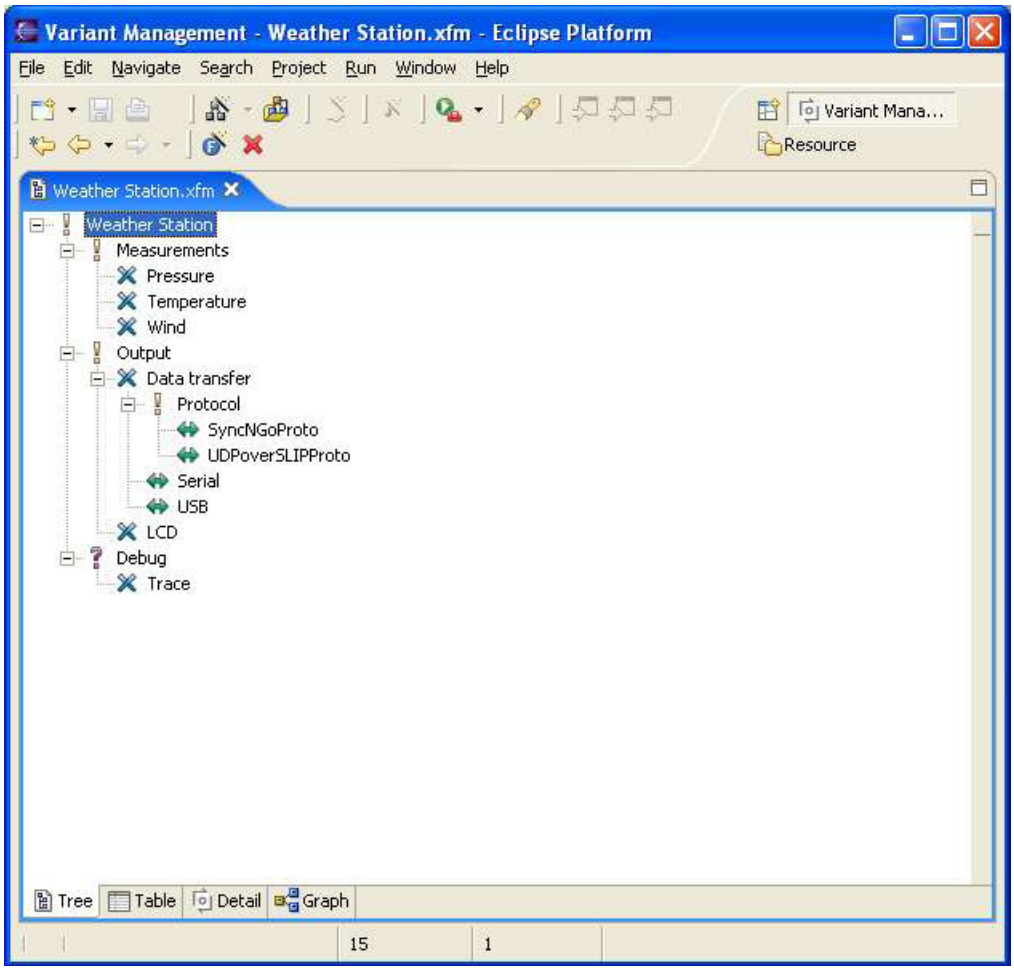

Figura 4. Modo Árvore da Interface de Edição do Feature Model [6]

Através da tela da ferramenta apresentada na Figura 4, podemos visualizar a interface de edição do modelo de característica em modo árvore. Nesta interface, encontramos as chamadas características obrigatórias (**!**), características alternativas *'or'* (**X**) e características opcionais (**?**). Sua arquitetura é definida pela notação FODA **Erro! Fonte de referência não encontrada.**, sendo também o pure::variants dividido em 3 modelos distintos: o modelo de características, modelo de variantes e o modelo de família. No primeiro modelo são definidas as características usadas posteriormente na validação do modelo de variantes. O modelo de variantes representa a instância da linha de produto. Por último, o modelo de família pode ser usado juntamente com o modelo de características para gerar os produtos da linha após a validação dos modelos.

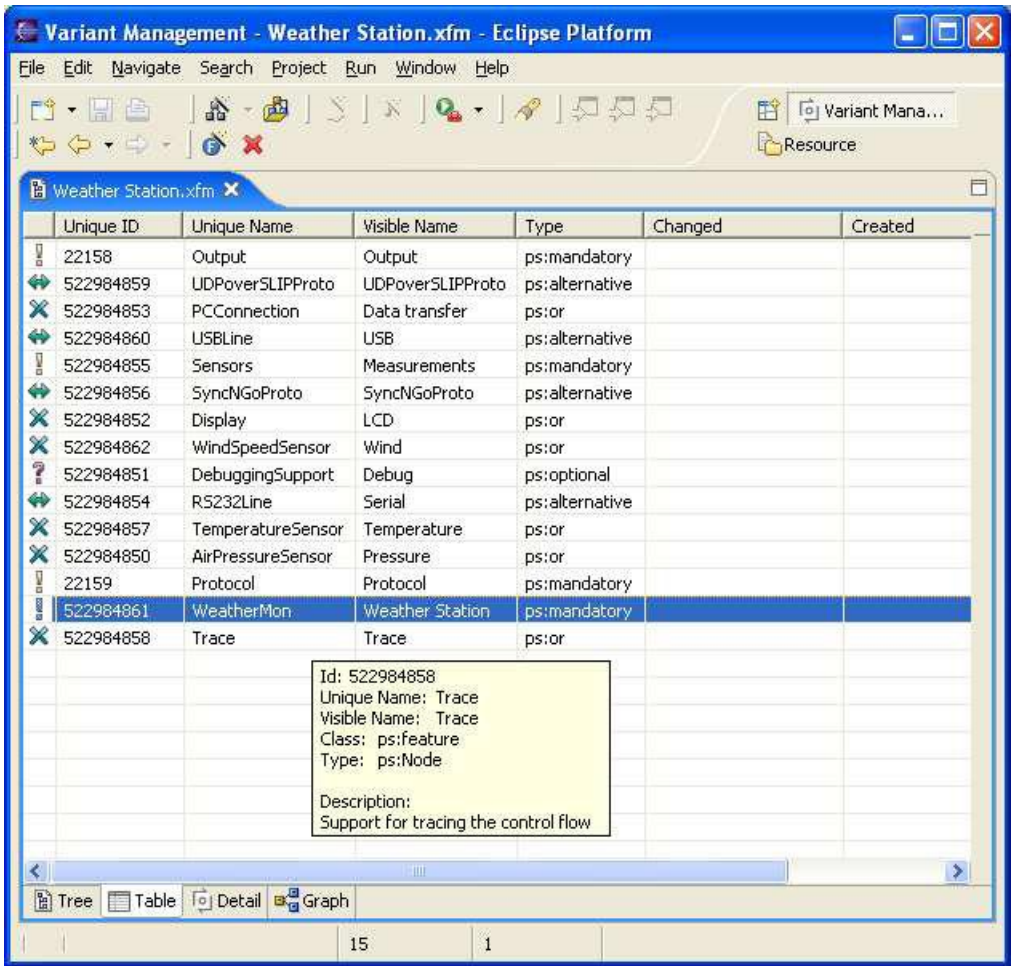

Figura 5. Modo Tabela da Interface de Edição do Modelo de Característica [6]

Na Figura 5 é possível visualizar a mesma interface de edição mostrada anteriormente na opção de visualização do modo tabela. Essa opção nos permite visualizar maiores informações da característica manipulada, tais como seu ID, nome único, nome visível e seu tipo por exemplo.

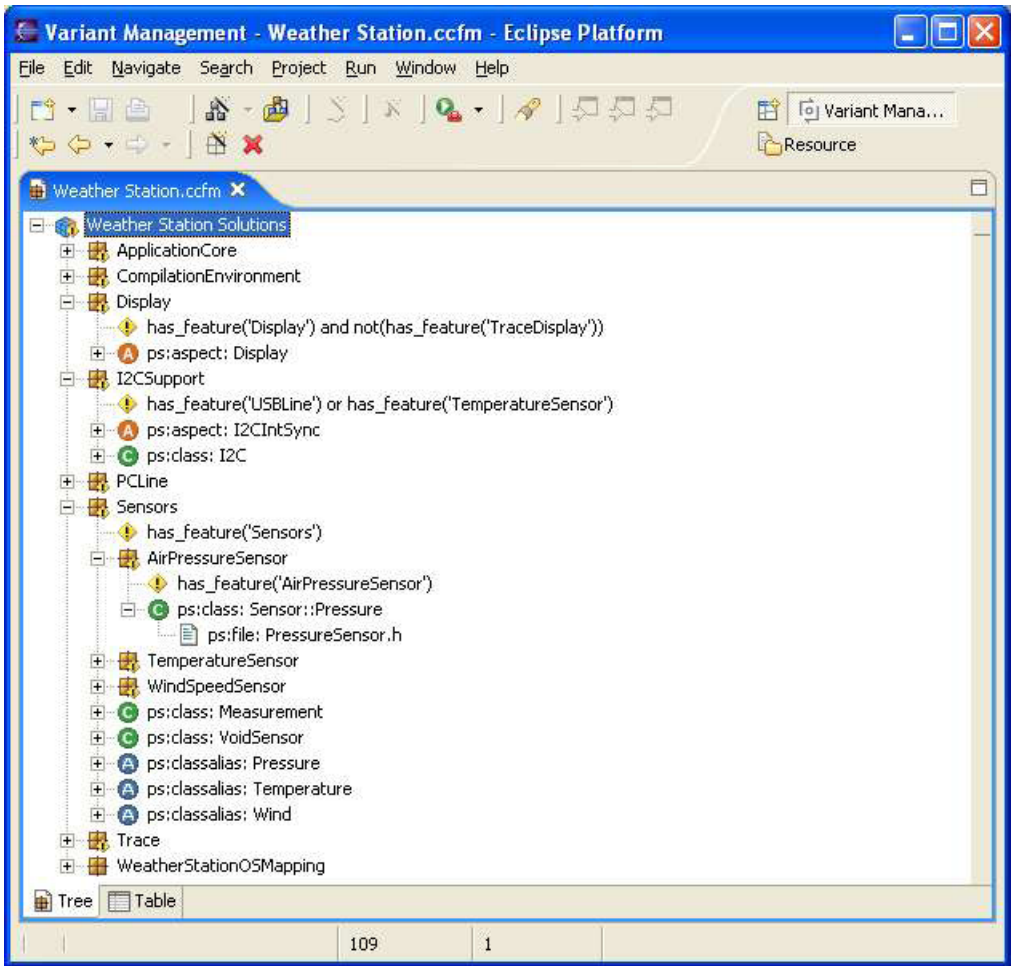

Figura 6. Interface de Edição do Family Model [6]

Na Figura 6 é possível visualizar os componentes (representados pelas caixas na cor marron), as restrições (representadas pelas placas amarelas contendo uma exclamação), classes (representadas por um circulo verde com um ‘C’), os aspectos (representados por círculos de cor laranja com um ‘A’) e os arquivos (representado por uma folha de papel dobrada na ponta) que pertencem à tela de edição do modelo de família.

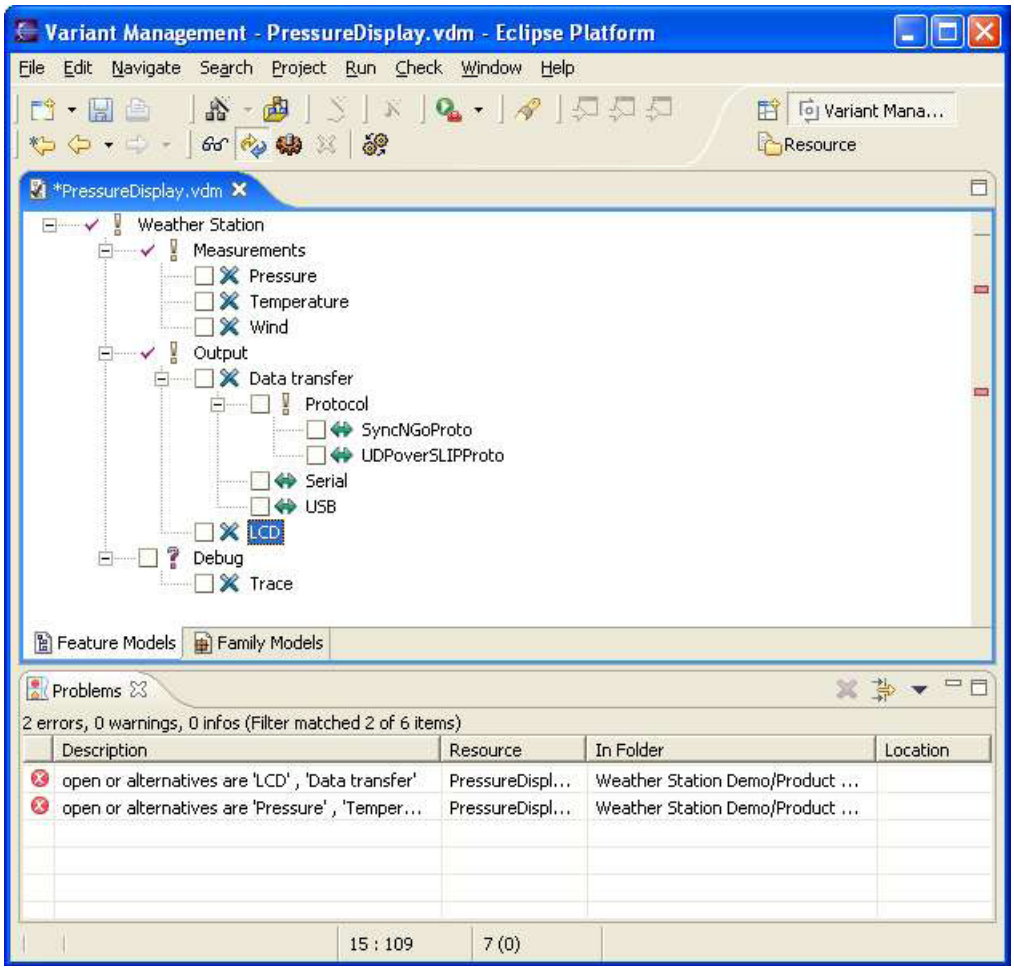

Figura 7. Interface de Edição de um Variant Model [6]

No *pure::variantes* a tela apresentada na Figura 7 é utilizada para se fazer a seleção de características em uma instância do modelo de características. Essas características são selecionadas através da marcação do *checkbox* relacionado às mesmas. Apesar da dificuldade apresentada por usuários iniciantes ao utilizar a ferramenta, ela tem como pontos positivos uma maior variabilidade de editores e *wizards*, melhor organização de arquivos, tornando-a uma ferramenta mais versátil e de maneira geral mais organizada e completa. Além disso, ela oferece uma linguagem proprietária alternativa e muito intuitiva, o "pvscl", voltada para usuários comuns.

3.4 Software Product Lines Online Tools (SPLOT)

O SPLOT é uma ferramenta acadêmica online que promove a pesquisa colaborativa em LPS. Ela foi lançada em maio de 2009 e desenvolvida pelo Marcílio Mendonça, Donald Cowan da Universidade de Waterloo no Canadá e Moises Branco do Banco do Nordeste do Brasil. Conforme o website da ferramenta[3], ela foi visitada por grupos de pesquisas de mais de 15 países diferentes desde o seu lançamento. A ferramenta é *open source* mantendo seu código fonte disponível para que possa ser customizado e/ou aperfeiçoado. Tais modificações podem ser incorporadas ao projeto principal. Por ser uma ferramenta baseada na internet, seu acesso e compartilhamento são facilitados. Além disso, não são necessárias atualizações através de *download*s. Pelo que sabemos, o SPLOT foi o primeiro sistema baseado em internet a oferecer tamanha variedade de serviços para usuários finais de LPS e seus pesquisadores.

Apesar de a ferramenta estar disponível apenas em inglês, a linguagem encontrada em seus documentos explicativos é simples, muito bem detalhada e sempre acompanhada de exemplos. A ferramenta SPLOT possui disponível para seus usuários um repositório contendo cerca de 180 modelos reais publicados anteriormente, além de modelos gerados automaticamente com até 100.000 características. A manutenção desse repositório parte de relatos dos próprios pesquisadores de LPS onde eles descrevem a dificuldade em encontrar modelos de características disponíveis publicamente. Com isso, o SPLOT procura incentivar o compartilhamento de informação entre os pesquisadores através da publicação de modelos no repositório.

[3] http://www.splot-research.org/

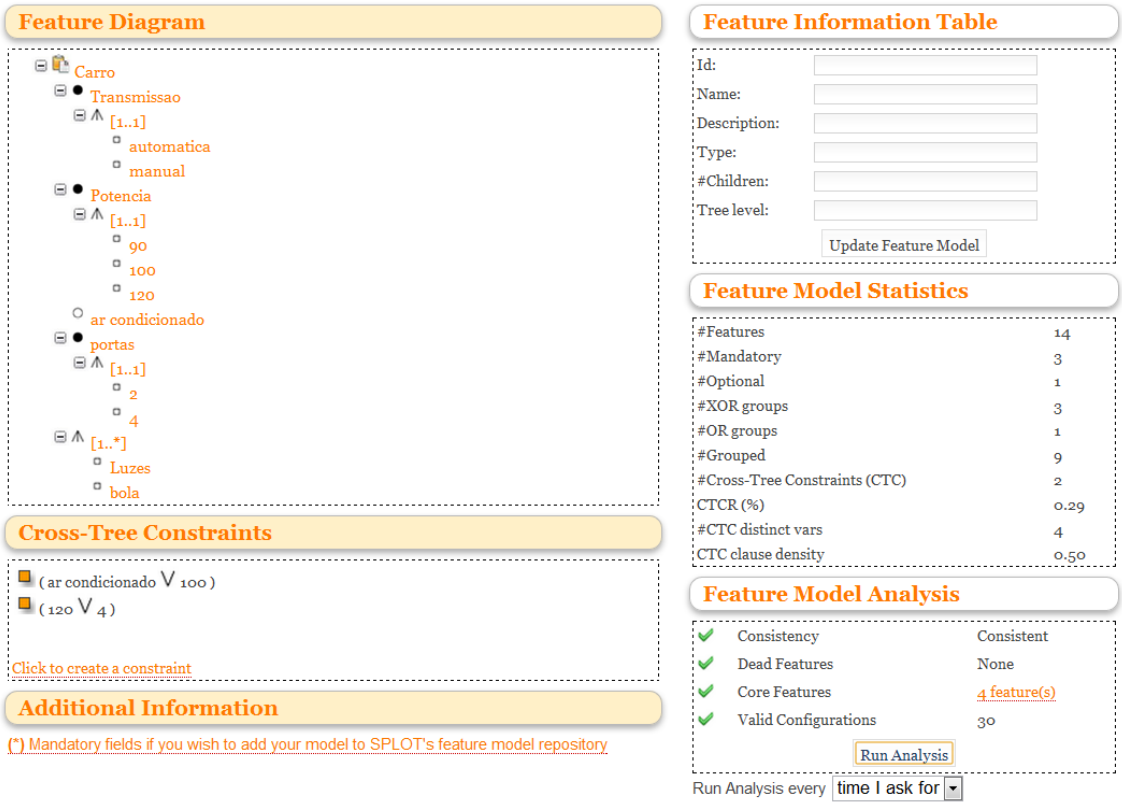

Figura 8. Interface do Editor de Modelos de Características

Na Figura 8 podemos observar o editor de modelos de características dessa ferramenta que foi desenvolvido em Java e utiliza um modelo HTML baseado em Ajax na geração das interfaces de interação com o usuário. A ferramenta faz uso de soluções na construção de seus modelos de características de técnicas baseadas em BDDs [19] e SAT **Erro! Fonte de referência não encontrada.**. A escolha da utilização dessas técnicas são resultados de pesquisas realizadas na exploração da conexão entre os modelos de características e a lógica proposicional **Erro! Fonte de referência não encontrada.**. Como é sabido, a utilização de tais técnicas não resulta em um sistema consistente, podendo gerar alguns problemas. O SPLOT utiliza também uma nova heurística em BDD **Erro! Fonte de referência não encontrada.** para reduzir o tamanho dos BDDs o máximo possível, na tentativa de minimizar esses possíveis problemas.

Automated Analysis

Click the ***"Click to Run"*** links below to run feature model analysis based on SAT solvers and Binary Decision Diagrams.

Data	**Carro Model** (view)
Statistics	
#Features	14
- Optional	1
- Mandatory	3
- Grouped	9
- Groups	4
Tree Depth	3
ECR (%)	28
#Extra constraints	2
#Distinct extra constraints variables	4
Clause Density	0.5
#CNF Clauses	28
Debugging Analyses (Click to Run the SAT solver)	
Consistency	consistent
Running Time (ms)	0
#Dead Features	0
- Running Time (ms)	1
#Common Features	4 view
- Running Time (ms)	1
Metrics (Click to run the BDD engine)	
Count Configurations	30
- Running Time (ms)	4
Variability Degree (%)	1.8311E-1
- Running Time (ms)	4
#BDD Nodes	22
#BDD Variable Order Heuristic	Pre-CL-MinSpan

Figura 9. Interface da Análise Automatizada

A ferramenta SPLOT apresenta ainda dois principais serviços: a análise automatizada e a configuração do produto. O serviço de análise automatizada oferecer suporte a propriedades de medição como o número de configurações válidas ou o grau de variabilidade entre os modelos de características. A Figura 9 apresenta um exemplo

de análise automatizada da ferramenta. Este serviço é responsável por automatizar o cálculo de métricas e estatísticas, como por exemplo, a profundidade da árvore de características ou o número de características. É ele também que executa tarefas críticas de depuração do sistema, tais como, a verificação da consistência dos modelos de características e a detecção de características mortas e características em comum. As características mortas são as características que em um dado momento por influencia das restrições e do resultado parcial das configurações se tornam inutilizáveis, ou seja, não podem ser utilizadas pelo usuário.

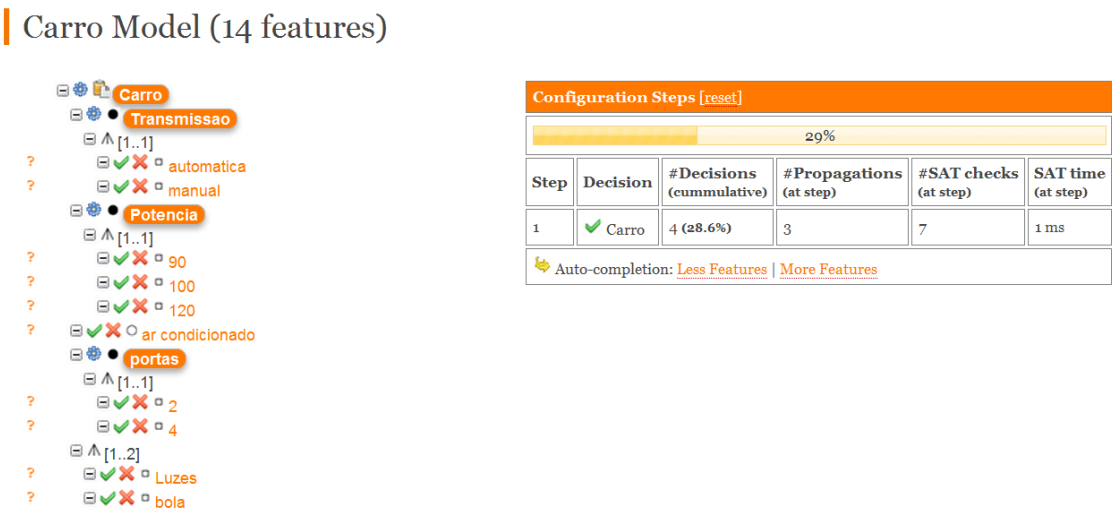

Figura 10. Interface de Configuração do Produto

O serviço de configuração do produto do SPLOT tem por base uma configuração interativa **Erro! Fonte de referência não encontrada.** onde o usuário é responsável por tomar decisões a todo o momento. Com isso, o sistema automaticamente propaga essas decisões, tornando o sistema mais coeso e consistente. A Figura 10 mostra um exemplo de produto sendo construído através da escolha das características que o compõe. Para suportar possíveis erros nas tomadas de decisões do usuário, o sistema também oferece um processo livre de retorno das configurações, beneficiando diretamente os usuários que nunca são forçados a rever suas decisões. Resumidamente, a ferramenta SPLOT permite que você edite, depure, analise, configure, compartilhe e acesse outros modelos de características instantaneamente **Erro! Fonte de referência não encontrada.**.

4 ESTUDO DE CASO

4.1 Metodologia

Este capítulo se propõe a apresentar uma avaliação da ferramenta SPLOT através de um estudo de caso baseado na literatura do EShop **Erro! Fonte de referência não encontrada.**.

Apesar de não representar uma modelagem completa o EShop, possui uma família de produtos com características suficientes para a demonstração dos recursos da ferramenta de modelo de característica SPLOT. É possível ilustrar situações de cardinalidades (senha), características opcionais (detecção de fraudes), características obrigatórias (forma de pagamento) e características com atributos (expiração).

O EShop é uma linha de produtos de lojas eletrônicas que efetua vendas on-line e compartilham características como:

- A forma de pagamento: Podendo ser através de cartão de crédito, cartão de débito e ordem de compra;
- Um sistema de detecção de fraude;
- Modo de envio do produto: Podendo ser realizado através de métodos próprios de envio personalizados ou utilizar algum método de envio existente (Correios);
- Sua política de senha composta por:
    - Expiração: Onde é definido depois de quanto tempo a senha irá expirar.
    - Caracteres necessários: Tipo de caracteres necessários na composição da senha, letras minúsculas, letras maiúsculas, números e caracteres especiais.

Através dessas características compartilhadas nessa linha de produtos é possível uma customização entre as lojas virtuais.

Algumas ações básicas foram realizadas a fim de atingir os objetivos do estudo de caso. A ferramenta SPLOT que se encontra disponível na *Web* foi acessada, sua

documentação foi estudada para facilitar o procedimento e em paralelo ao processo será realizada a análise da ferramenta.

4.2 Modelagem do EShop no SPLOT

A implementação do estudo de caso foi realizada objetivando a igualdade entre o modelo criado e o modelo apresentado no artigo que descreve o estudo de caso **Erro! Fonte de referência não encontrada.**. As características foram criadas na ordem em que são apresentadas no modelo. Diversos recursos foram utilizados como: o grupo de características, características opcionais, características mandatórias, as cardinalidades dentre outras no editor de modelos de características. Foi utilizado também o gerador de analise automatizado, assim como, a interface de configuração do produto.

O primeiro passo do estudo de caso foi realizado utilizando o editor de modelo de característica do SPLOT. A interface é de fácil manipulação com uma linguagem simples e intuitiva. Esse editor tem uma disponibilidade dos comandos de forma a facilitar sua utilização na modelagem, tornando mais simples todo o processo. A interface ainda conta com um analisador de modelo de característica usado para verificar a consistência do modelo configurado. Existe também a possibilidade de se adicionar informações sobre o modelo em questão através de um campo dedicado, o "*Additional Information"*.

Para a implementação da literatura EShop foram encontradas algumas limitações que não puderam ser representadas com o editor do SPLOT. Limitações por exemplo para a criação das restrições que fazem parte do modelo original. Essas limitações serão discutidas com mais detalhes na seção 4.4 onde será realizada uma comparação das ferramentas. A Figura 11 representa o resultado final da modelagem utilizando essa ferramenta, que é o mais próximo do proposto pelo EShop.

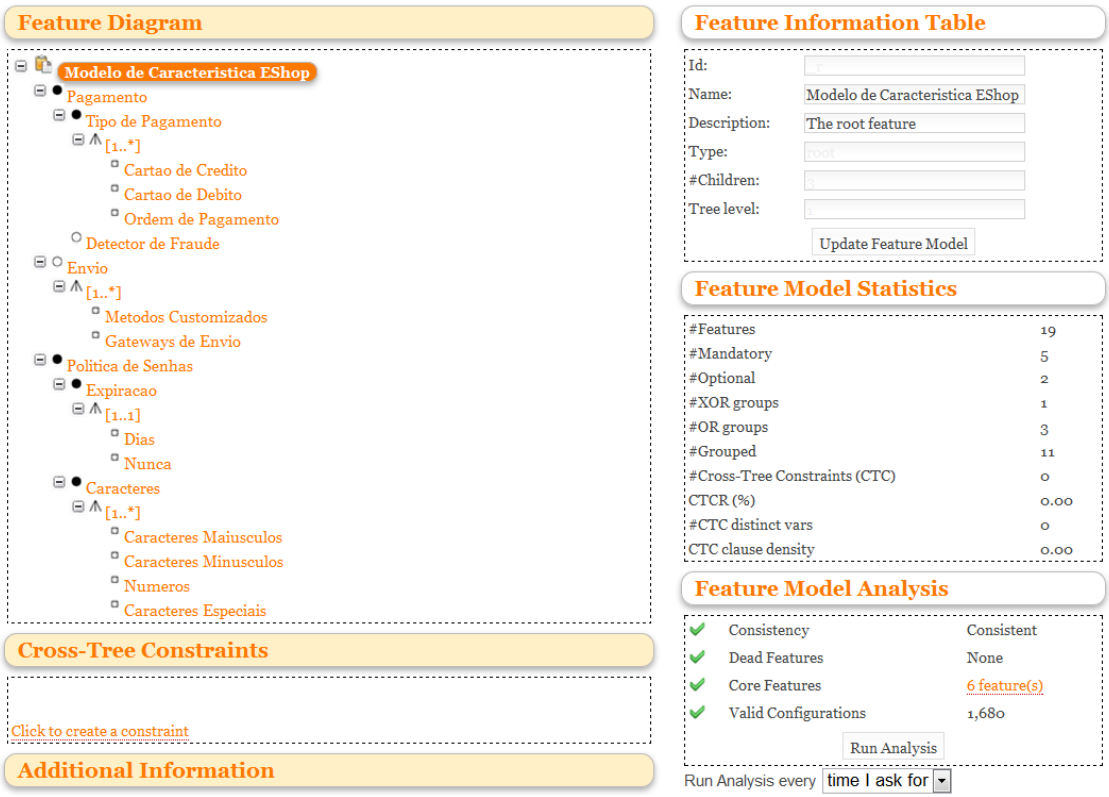

Figura 11. Modelo EShop no Editor de Modelos de Características

A interface de análise automatizada também foi utilizada no processo com o intuito de efetuar uma verificação sintática do modelo configurado e gerar também um levantamento estatístico de sua configuração final. Essa interface também muito simples de ser utilizada necessita apenas que o modelo seja indicado e carregado para que suas estatísticas sejam geradas. Com essa interface também é possível efetuar a depuração do seu modelo utilizando a biblioteca Java SAT. Ou até mesmo efetuar o levantamento das métricas através do BDD *engine*. A Figura 12 mostra as estatísticas geradas pela análise do modelo EShop.

Automated Analysis

Click the "***Click to Run***" links below to run feature model analysis based on SAT solvers and Binary Decision Diagrams.

Data	**Modelo de Caracteristica EShop (view)**
Statistics	
#Features	26
- Optional	2
- Mandatory	9
- Grouped	14
- Groups	5
Tree Depth	5
ECR (%)	11
#Extra constraints	1
#Distinct extra constraints variables	3
Clause Density	0.3
#CNF Clauses	42
Debugging Analyses (Click to Run the SAT solver)	
Consistency	consistent
Running Time (ms)	0
#Dead Features	0
- Running Time (ms)	0
#Common Features	10 view
- Running Time (ms)	2
Metrics (Click to run the BDD engine)	
Count Configurations	11,700
- Running Time (ms)	4
Variability Degree (%)	1.7434E-2
- Running Time (ms)	4
#BDD Nodes	32
#BDD Variable Order Heuristic	Pre-CL-MinSpan

Figura 12. Modelo EShop no Analisador Automatizado

A interface de configuração do produto também foi testada no processo através da configuração de um produto exemplo utilizando as características inseridas anteriormente em nosso modelo. Esse produto exemplo foi configurado sem nenhuma dificuldade, facilidade essa que é o resultado de uma interface simples e fácil de ser manipulada. A interface é bem flexível permitindo que as características sejam marcadas e desmarcadas a qualquer momento do processo de configuração. Além de oferecer opções de auto-complemento das características que facilita o processo oferecendo as opções de auto-complemento do número máximo de características (*More Features*) ou o número mínimo de características (*Less Features*) possíveis. A Figura 13 mostra o resultado parcial da configuração de um produto teste na interface do SPLOT.

Figura 13. Modelo EShop no Configurador de Produtos

4.3 Modelagem do EShop em Outras Ferramentas

A implementação do estudo de caso nas demais ferramentas citadas neste trabalho seguiu o mesmo princípio utilizado na ferramenta SPLOT a fim de obter dados reais possibilitando efetuar a comparação entre as mesmas. Para o estudo de caso também foi utilizado o documento q descreve o exemplo de estudo de caso EShop **Erro! Fonte de referência não encontrada.**. O processo foi executado da mesma forma que na ferramenta SPLOT utilizando o maior número de recursos possíveis e objetivando a maior similaridade com o modelo original proposto. Nessa seção será

apresentado uma breve descrição e o resultado do modelo de características modelado na ferramenta em questão. Demais detalhes serão apresentados na seção 4.3 onde uma comparação entre as ferramentas é apresentada.

Na ferramenta *XFeature* o modelo foi implementado utilizando como meta-modelo a configuração FMP disponível na ferramenta para definição do modelo de características. Essa configuração provê os mesmos recursos disponíveis na ferramenta *fmp* facilitando assim o processo de modelagem, onde apenas algumas diferenças na notação utilizada pôde ser observada. A Figura 14 e a Figura 15 ilustram o modelo resultante da modelagem na ferramenta *XFeature.*

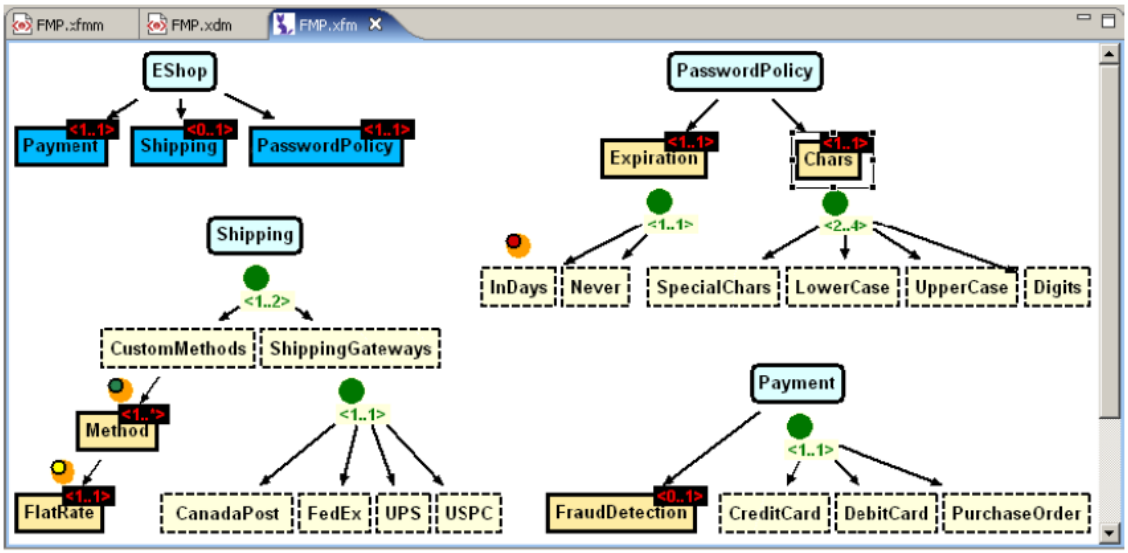

Figura 14. Modelo de Característica do EShop no XFeature [6]

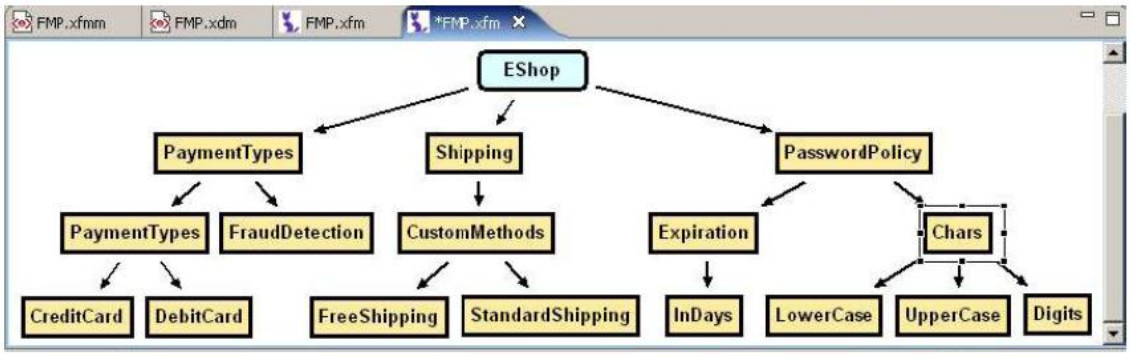

Figura 15. Instâncias do EShop no XFeature [6]

Não houve problemas na modelagem utilizando a ferramenta *fmp* pois o modelo proposto EShop foi originalmente implementado para esta ferramenta. No caso dessa modelagem em especial foram criados sub-modelos inicialmente para cada uma das características (pagamento, envio e politica de senha). Com isso o modelo da loja EShop foi criado utilizando referências para os sub-modelos das características criados anteriormente. Na Figura 16 pode ser observado o modelo resultante da implementação na ferramenta *fmp*.

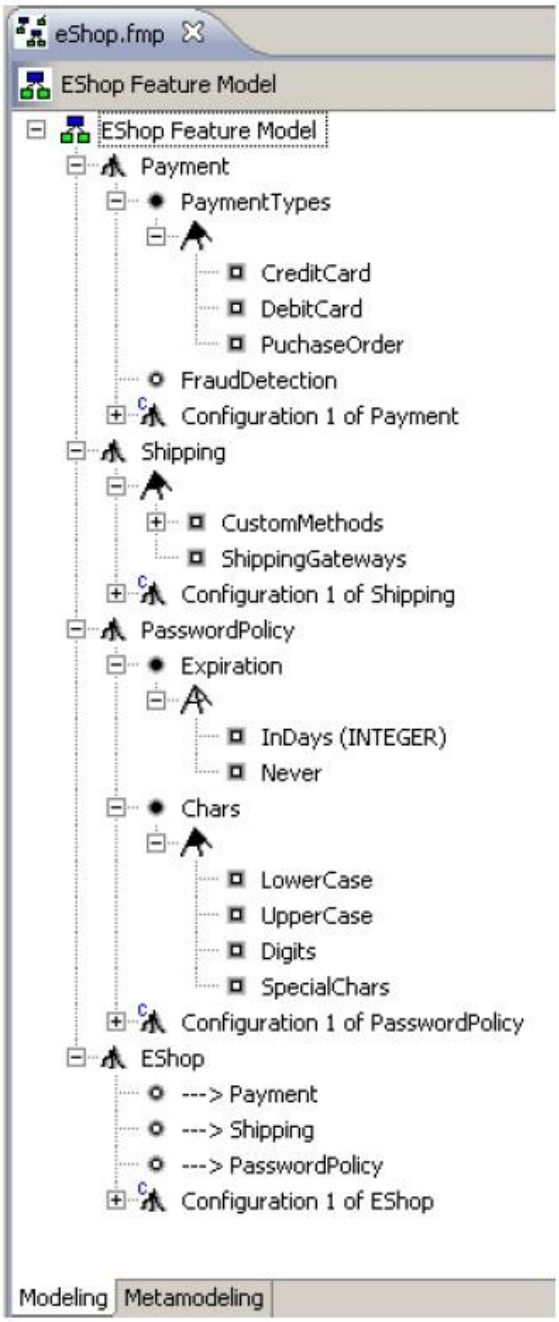

Figura 16. Modelo Implementado na Ferramenta fmp [6]

A modelagem no pure::variant não pode contemplar a representação das cardinalidades mas a ferramenta disponibiliza artifícios como o *configuration knowledge* para configuração e o *check model* para checagem do modelo. O *Family mode* da ferramenta não foi utilizado por requerer uma aplicação *on-line* o que fugiria do escopo do projeto. A Figura 17 e a Figura 18 ilustram o resultado final da modelagem nessa ferramenta.

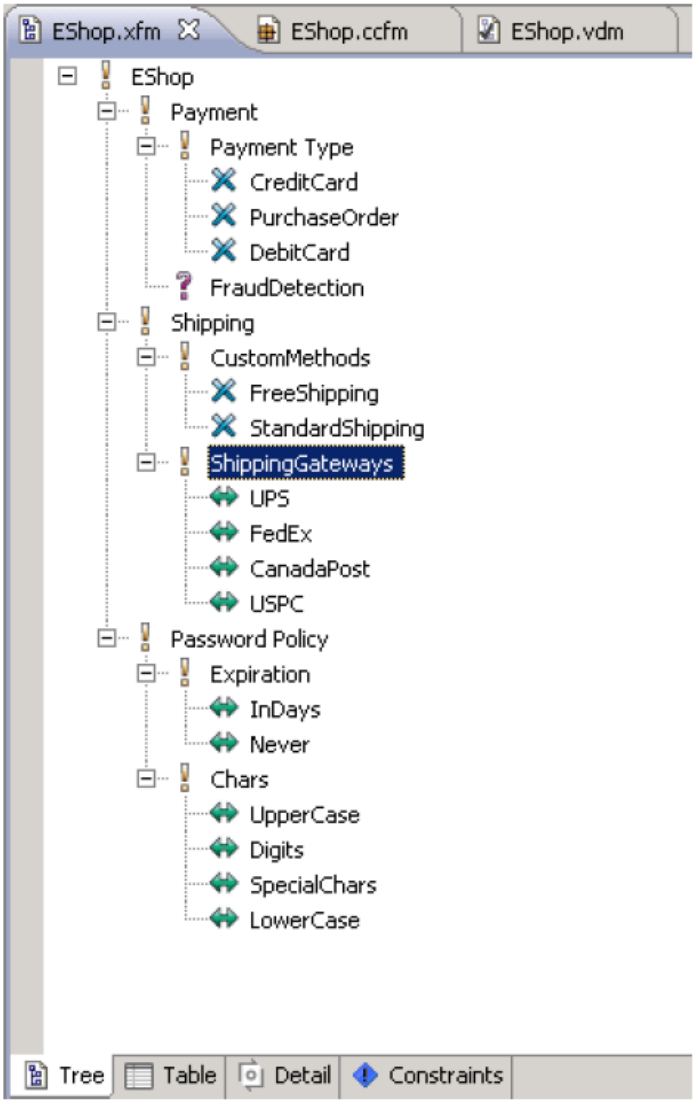

Figura 17. Modelo de Caracteristica do EShop no pure::variant [6]

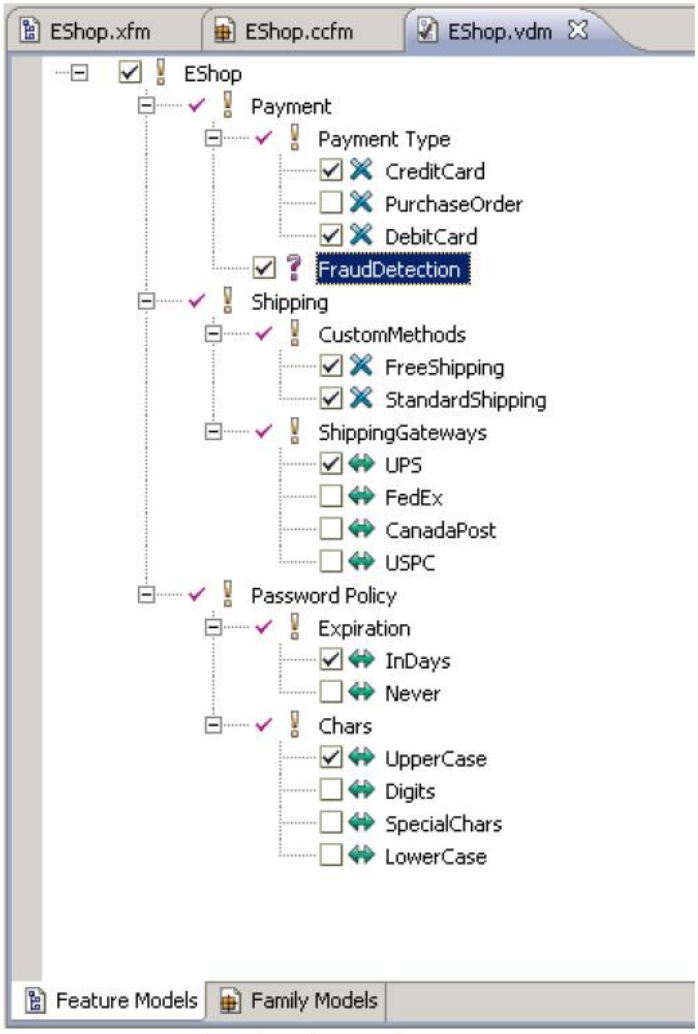

Figura 18. Instâncias do EShop no pure::variant [6]

4.4 Resultado Comparativo

O modelo utilizado como exemplo no estudo de caso EShop **Erro! Fonte de referência não encontrada.** é um modelo baseado no *fmp* ferramenta também desenvolvida pela Universidade de Waterloo. As duas ferramentas desenvolvidas por essa universidade são de fato muito semelhantes. Mas apesar das semelhanças algumas limitações foram encontradas na ferramenta SPLOT no processo de implementação do modelo proposto. Limitações essas que não tornam a ferramenta em questão pior levando em consideração que a mesma oferece serviços não encontrados nas outras ferramentas.

A ferramenta SPLOT demonstrou através do estudo de caso que o processo de criação de um modelo de características e de suas instâncias é simples e intuitivo.

Comportamento equivalente ao encontrado na ferramenta *fmp, pure::variants* e também na *XFeature.* Essa última ferramenta citada possui uma configuração que provê os recursos disponíveis no *fmp,* mas leva desvantagem por complicações no processo de criação e manipulação de suas instâncias e de seus meta-modelos obrigatórios. Todas as características do modelo EShop foram modeladas claramente na ferramenta SPLOT mas com algumas limitações em suas configurações.

No SPLOT a representação da característica atributo "Dias" como do tipo inteiro não foi possível de ser configurada, assim como, nas ferramentas *XFeature e pure::variant*. Obrigando o usuário a tratar tal configuração de outras maneiras. Comportamento distinto da ferramenta *fmp* que representa de forma simples e objetiva essa configuração onde somente números inteiros são validos para a característica em questão.

Outra limitação da ferramenta SPLOT foi à customização da cardinalidade nos grupos de características. No grupo de características "Caracteres" o modelo propõe que de duas até quatro [2-4] das opções disponíveis possam ser selecionadas. Configuração possível de ser configurada na ferramenta *fmp* e no *XFeature*. Na ferramenta SPLOT a cardinalidade que mais se aproxima da proposta pelo modelo é a configuração onde uma ou mais características [1..*] possam ser selecionadas. Comportamento esse melhor do que o encontrado na ferramenta *pure::variants* que não permite a definição das cardinalidades.

Assim como nas outras ferramentas a representação do modelo de características em forma de árvore utilizada no SPLOT facilita a visualização do usuário e a organização das características criadas. Outra função importante existente no SPLOT e nas demais ferramentas é a atualização automática do modelo de características sempre que uma modificação é realizada, isso permite que o usuário visualize as consequências das mudanças realizadas.

Existem duas maneiras de se exportar ou gravar os modelos de características criados utilizando a ferramenta SPLOT. Uma delas é semelhante a maneira utilizada no *fmp* e no *XFeature* que dependem fortemente do XML, no caso do SPLOT o modelo é exportado no formato SXFM que representa uma URL contendo um código XML que é exibido no próprio *browser* onde o sistema está rodando. O uso desse formato é justificado pela vantagem de ser possível criar um modelo de características a partir de

um simples editor de texto. Na Figura 19 podemos visualizar uma API (SPLOT *Feature Model Generator*) disponibilizada para download no site oficial da ferramenta. Essa API é utilizada para gerar códigos no formato SXFM utilizado na exportação de seus modelos.

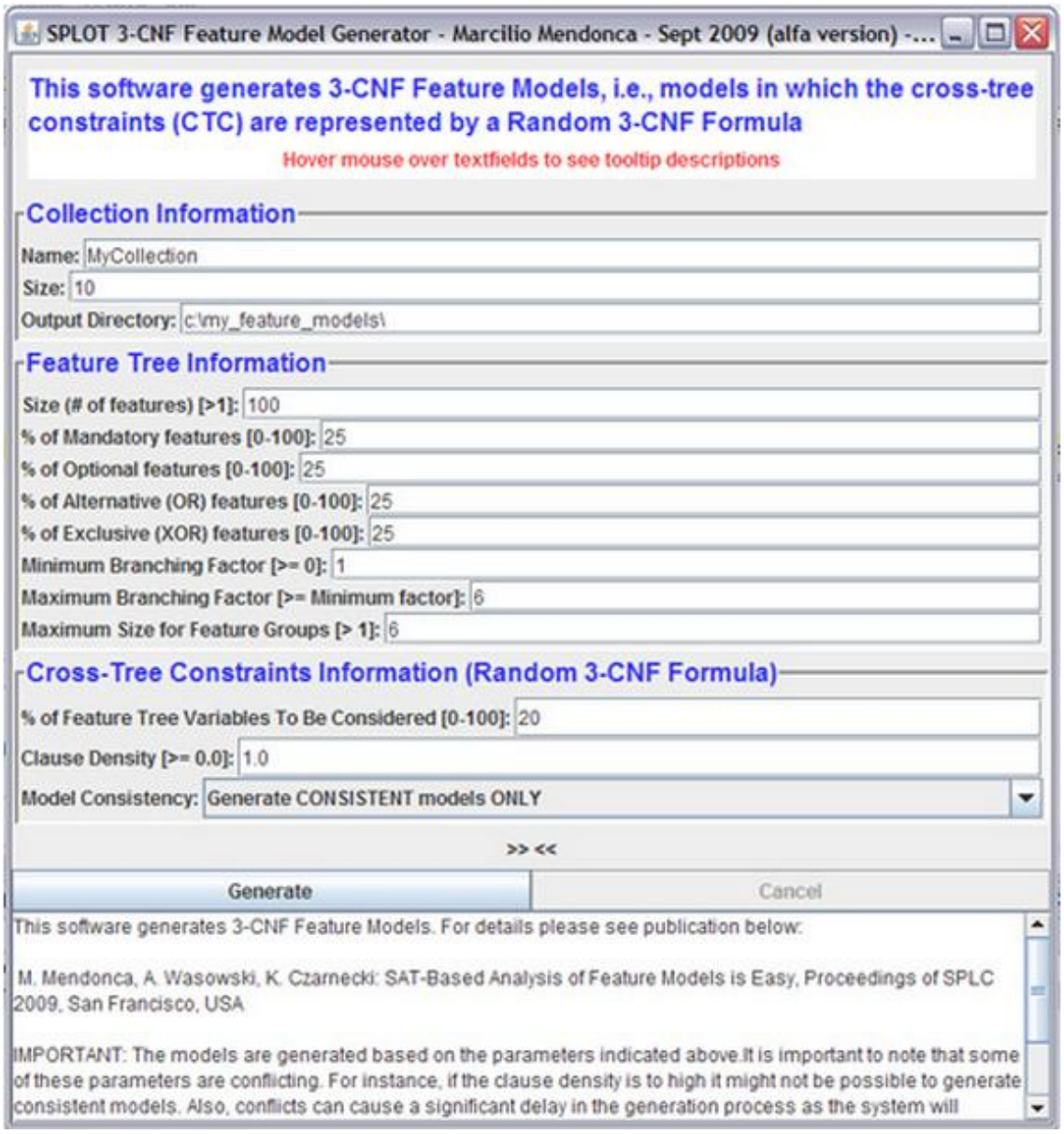

Figura 19. API SPLOT Feature Model Generator [27]

Outra opção é salvar seu modelo no repositório de modelos de características que a ferramenta oferece. Levando em consideração que alguns requisitos mínimos devem ser cumpridos como o preenchimento informações obrigatórias e o mínimo de 10 características válidas no modelo. A ferramenta também permite a verificação de erros de conversão na URL no formato SXFM antes que o modelo de características

seja carregado. Diferente do *pure::variant* que faz uso de transformações baseadas em XSLT na exportação de seus arquivos.

Houve também dificuldades na representação das restrições do modelo original EShop no SPLOT. A construção de restrições no SPLOT faz uso da Forma Normal Conjuntiva (*Conjunctive Normal Form - CNF*), o que dificulta e limita a utilização da mesma por não conhecedores de tal formatação. Diferente do *pure::variant* que faz uso de uma linguagem alternativa própria a "pvscl" mais simples de ser interpretada e trabalhada. Diferente também do *fmp* que antigamente trabalhava com o formato XPath 2.0 que foi substituído por não apresentar uma boa legibilidade. O novo formato ainda se encontra em fase de desenvolvimento, mas sua versão disponível apresenta uma linguagem mais simplificada. Já a ferramenta *XFeature* também apresenta um formato mais trabalhoso onde um modelo de restrições deve ser criado e compilado (*global constraints compiler*) e com isso será gerado um conjunto de arquivos XSL que permitem a verificação das restrições.

O analisador automatizado é um recurso disponível na ferramenta SPLOT de grande ajuda para análise e diagnóstico do modelo configurado. Esse serviço calcula as estatísticas referentes ao modelo em questão. Com resultado desse cálculo são apresentados valores como: a contagem geral das características do modelo, o numero de características agrupadas por categoria e a profundidade da arvore (Tree Depth). Apresenta também através do BDD *engine* o número de configurações válidas a partir desse modelo e calcula o grau de variabilidade do modelo possibilitando com isso uma configuração mais interativa. Esse serviço disponibiliza também um solucionador SAT utilizado para efetuar a depuração do modelo verificando a consistência do modelo de característica além de apontar as características comuns e mortas. Recursos esses citados que não foram encontrados nas demais ferramentas apresentadas.

O SPLOT possui um serviço de configuração do produto que apresenta semelhanças com o *configuration knowledge* da ferramenta *pure::variant.* Nessa interface o usuário encontra todos os componentes que compões os artefatos da linha de produto em questão. Com isso a configuração de um produto pode ser realizada e acompanha devidamente através da tabela "etapas da configuração" (*Configuration Steps)* para sua devida validação. A ferramenta é bem flexível permitindo que o usuário após verificar as consequências de uma determinada alteração na configuração desfaça

ou refaça a mesma a qualquer momento do processo. Nessa tabela podemos visualizar as decisões tomadas no processo de configuração, assim como, outras informações das características assinaladas e do processo como um todo. A interface ainda oferece duas opções de auto-completar (*Auto-Completion*) onde o mínimo ou o máximo de características necessárias para a configuração de um produto serão assinalados. Esse recurso pode ser utilizado a qualquer momento no processo.

5 CONCLUSÃO

O uso do desenvolvimento baseado em Linhas de Produtos de Software vem apresentando resultados expressivos nas indústrias. Com a obtenção de softwares de maior qualidade, menor custo e com menos custo e esforço de desenvolvimento. Sendo assim, uma opção interessante para empresas que pretendem manter seu crescimento e também a competitividade no mercado.

A acessibilidade e a capacidade de manipular informações a partir de qualquer lugar são providas atualmente graças à evolução constante dos sistemas computacionais ao longo do tempo. Paralela a essa evolução desses sistemas computacionais baseados em software há também a necessidade cada vez maior de um processo de desenvolvimento cada vez mais rápidos e eficazes. O modelo de Linhas de Produtos de Software (LPS) desvia o foco de um desenvolvimento convencional para uma abordagem de desenvolvimento baseada na composição de artefatos e na modelagem do domínio, atendendo assim um determinado segmento específico do mercado.

Através da análise comparativa dos resultados do estudo de caso realizado com a modelagem do EShop **Erro! Fonte de referência não encontrada.** na ferramenta SPLOT e nas ferramentas de mesmo propósito *XFeature, fmp e pure::variant*. Foi possível concluir que a ferramenta de configuração de sistemas de linha de produtos de software SPLOT apresentada por mim neste presente trabalho demonstrou ser capaz de atingir os principais objetivos vigentes em uma modelagem de LPS apesar as limitações citadas.

Podemos levar em consideração os aspectos positivos apresentados pela ferramenta SPLOT como sua indiscutível alta disponibilidade e facilidade de acesso por ser uma ferramenta baseada na *web* e apresentar uma linguagem simples e objetiva. A utilização de duas poderosas técnicas SAT *solvers* e BDD's *engine* utilizadas para automatizar a resolução no suporte da modelagem das características e tornar os serviços de configuração mais interativos. E até mesmo a criação e manutenção de um repositório de modelos de características incentivando os pesquisadores e estudiosos da área a publicarem seus modelos difundindo assim a utilização das técnicas.

Podemos concluir também que as limitações apresentadas na ferramenta são em sua grande maioria de aspecto informativo não representando real dano na

representação do modelo de características em questão. Algumas delas podendo ser facilmente evitadas ou contornadas através de outros recursos ou até mesmo de uma análise de domínio bem realizada. É sabido que para uma melhor representação de um modelo de características a ferramenta depende principalmente de uma boa definição das características utilizadas no processo de modelagem. Por esse motivo que uma análise de domínio é considerada como o primeiro passo para estabelecer os requisitos necessários que devem ser suportados pela ferramenta que irá representar o modelo. Levando isso em consideração podemos dizer que a ferramenta SPLOT suporta a maioria dos modelos de características com uma modelagem fácil e rápida, e a utilização de recursos não encontrados nas demais.

Espero que com esse trabalho eu possa contribuir com informações que ofereçam benefício para a comunidade de Engenharia de Software no ambiente da Linha de Produtos de Software e suas ferramentas.

5.1 Trabalhos Futuros

O estudo das ferramentas voltadas para o modelagem de linha de produto de software é um assunto que permite uma vasta exploração. Deste modo alguns pontos passíveis de continuidade deste trabalho puderam ser identificados.

- Aplicação da mesma proposta de análise comparativa utilizando estudos de casos mais complexos ou até mesmo projetos reais, na tentativa de validar de forma mais abrangente as regras de formatação do modelo de características.

- Um estudo poderia ser realizado na tentativa de identificar a melhor maneira de se representar os intervalos fixos das cardinalidades na ferramenta SPLOT, contornando assim sua limitação no uso desse recurso.

REFERÊNCIAS

[1] SOMMERVILLE, I. *Engenharia de Software*. 8th. ed. Pearson, 2007.

[2] COHEN, S.:*Product line state of the practice report*. Technical Report CMU/SEI-2002-TN-017. Software Engineering Institute. 2002.

[3] ETH ZURICH. *P&P Softwares, 2008.* Disponível em: <http://www.pnp-software.com/XFeature/Home.html.>

[4] GENERATIVE SOFTWARE DEVELOPMENT LAB. Disponível em: <http://gsd.uwaterloo.ca/fmp.>

[5] PURE-SYSTEMS GMBH. 2011. Disponível em:<http://www.pure-systems.com/Home.142.0.html>

[6] LIMA JUNIOR, R. *Comparação entre ferramentas para linha de produtos de Software.* Trabalho de Conclusão de Curso, Escola Politécnica de Pernambuco .2008. Disponível em: < http://tcc.dsc.upe.br/20081/RogeriomonografiaFinal.pdf>

[7] CLEMENTS, P.; BROWNSWORD, L. *A Case Study in Successful Product Line Development.* Technical Report CMU/SEI-96-TR-016, Software Engineering Institute. 1996 . http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA315802

[8] COHEN, S.: *Predicting when Product Line Investment Pays*. Technical Report CMU/SEI-2003-TN-017,. Software Engineering Institute (2003). Disponível em: http://www.sei.cmu.edu/pub/documents/03.reports/pdf/03tn017.pdf. Acesso em: 10/01/12.

[9] DURSCKIL, R.; SPINOLAL, M.; BURNETT, R.; REINEHR, S. Linhas de Produto de Software: riscos e vantagens de sua implantação. Simpósio Internacional de Melhoria de Processos de Software, 6. São Paulo. 2004. Disponível em: http://www.simpros.com.br/Apresentacoes_PDF/Artigos/Art_14_Simpros2004.pdf

[10] CLEMENTS, P. E NORTHROP, L.: *Software Product Lines: Practices and Patterns.* Reading, Addison-Wesley Professional, 2002.

[11] BATORY, D.: *Feature models, grammars, and propositional formulas.* Software Product Lines Conference - SPLC (2005).

[12] MENDONCA, M.; WASOWSKI, A.; CZARNECKI, K.; COWAN, D.: *Efficient compilation techniques for large scale feature models*. Conference on Generative Programming and Component Engineering - GPCE (2008).

[13] HADZIC, T.; SUBBARAYAN S.; JENSEN, R.; ANDERSEN, H.; MØLLER, J.; HULGAARD H.: *Fast backtrack-free product configuration using a precompiled solution space representation*. PETO Conference (2004).

[14] MENDONCA, M.; BRANCO, M.; COWAN, D.: *S.P.L.O.T. – Software Product Lines Online Tools*. Conference on Object Oriented Programming Systems Languages and Applications – OOPSLA (2009).

[15] MENDONÇA, M.: *Efficient Reasoning Techniques for Large Scale Feature Models*, Post-Doctoral Researcher, School of Computer Science University of Waterloo (2009).

[16] KANG, K.; COHEN, S.; HESS, J.; NOVAK, W.; PETERSON, A.: *Feature-Oriented Domain Analysis (FODA) Feasibility Study.* Technical Report CMU/SEI-90-TR-21. Software Engineering Institute (1990).

[17] LOBO, A.; RUBIRA, C.: *Um Estudo para Implantação de Linha de Produto de Software Baseada em Componentes.* Technical Report – IC-09-17 (2009).

[18] PROJETO YANA; disponível em: https://sites.google.com/site/projetoyanaufpb/home, 15/01/2012.

[19] D. L. BERRE, A. PARRAIN, O. ROUSSEL, AND L. SAIS. SAT4J: *A satisfiability library for Java, 2005.*

[20] J. WHALEY. The JavaBDD BDD library, 2003–2007. Disponível em: http://javabdd.sourceforge.net/.

[21] CZARNECKI, K.; EISENECKER, U. Generative Programming: Methods, Tools, and Applications. Addison-Wesley Professional, 2000.

[22] POHL, K.; BOCKLE, G.; LINDEN, F.J.V.D. Software Product Line Engineering: Foundations, Principles and Techniques. Springer, 2005.

[23] C. SZYPERSKI. Component Software: Beyond Object-Oriented Programming, Addison-Wesley, 1998.

[24] G. KICZALES, J. LAMPING, A. MENDHEKAR, C. MEADA, C. LOPES, J. LOINGTIER, J. IRWIN, *Aspect-Oriented Programming*. Proceedings of the 11th European Conference on Object-Oriented Programming (ECOOP), pp. 220–242, 1997.

[25] ANTIEWICZ, M. e CZARNECKI, K.. FeaturePlugin: Feature Modeling Plug-in for Eclipse. Conference on Object Oriented Programming Systems Languages and Applications – OOPSLA (2004).

[26] Disponível em: http://www.sciencedirect.com/science/article/pii/S147403460600067X, 16/01/2012.

[27] Disponível em: http://www.splot-research.org/, 10/01/2012.