UNIVERSIDADE ESTADUAL DO NORTE DO PARANÁ

FACULDADES LUIZ MENEGHEL

ALESSANDRO HENRIQUE DE QUEIROZ DA SILVA

# UM ESTUDO DE CASO QUALITATIVO DE FERRAMENTAS DE GERENCIAMENTO DE VARIABILIDADES EM LINHA DE PRODUTO DE SOFTWARE

**Bandeirantes**
**2007**

**ALESSANDRO HENRIQUE DE QUEIROZ DA SILVA**

# UM ESTUDO DE CASO QUALITATIVO DE FERRAMENTAS DE GERENCIAMENTO DE VARIABILIDADES EM LINHA DE PRODUTO DE SOFTWARE

Trabalho de Conclusão de Curso submetido à Universidade Estadual Norte do Paraná campus Faculdades Luiz Meneghel, como requisito parcial para a obtenção do grau de Bacharel em Sistemas de Informação.

**Orientadora:** Profª Cristiane Yanase Hirabara Castro

**Bandeirantes**
**2007**

**ALESSANDRO HENRIQUE DE QUEIROZ DA SILVA**

# UM ESTUDO DE CASO QUALITATIVO DE FERRAMENTAS DE GERENCIAMENTO DE VARIABILIDADES EM LINHA DE PRODUTO DE SOFTWARE

Trabalho de Conclusão de Curso submetido à Universidade Estadual Norte do Paraná campus Faculdades Luiz Meneghel, como requisito parcial para a obtenção do grau de Bacharel em Sistemas de Informação.

**COMISSÃO EXAMINADORA**

_____

Profª. Cristiane Yanase Hirabara Castro
Faculdades Luiz Meneghel

_____

Profº. José Reinaldo Merlin
Faculdades Luiz Meneghel

_____

Profª. Daniela de Freitas Guilhermino Trindade
Faculdades Luiz Meneghel

Bandeirantes, __ de_____ de 2007

Dedico este trabalho

À minha mãe **Maria Matilde Queiroz**, pela pessoa que sou hoje, à meus irmãos **Fernando** e **Tamara** pela alegria que me passam todo dia e a minha namorada **Claudia** pelo amor e carinho compartilhados.

## AGRADECIMENTOS

Á Deus, o meu maior agradecimento, por iluminar-me a cada dia de minha existência, por acompanhar meus passos nesta tão árdua jornada, por ter me dado força e alento nos momentos mais difíceis de minha vida e por dar-me estrutura para concluir mais esta etapa.

Um agradecimento especial à minha Professora orientadora, Cristiane Castro (Cris), por toda paciência, incentivo, confiança e conhecimentos transmitidos, os quais, contribuíram positivamente para a realização desta pesquisa, o meu muito obrigado.

A toda minha família pelo apoio incondicional, em especial à minha mãe Maria, meu pai Luiz, meu irmãozinho Fernando, minha irmãzinha Tamara e minha vózinha Madalena, pessoas que tanto amo.

Um agradecimento especial ao meu avô Manoel e ao meu tio José, que apesar de não participarem mais fisicamente das minhas conquistas, sei que sempre torceram e continuam torcendo pelo meu sucesso lá no céu.

Um agradecimento especial também, para minha amiga nos 3 primeiros semestre de faculdade e namorada nos 5 últimos, Claudia Czepak (Claudinha), obrigado por estar sempre ao meu lado compartilhando amor, atenção, apoio e incentivo durante essa dura caminhada, amo muito você.

Á Profª Daniela de Freitas Guilhermino Trindade (Dani) e ao Profº José Reinaldo Merlin gentilmente aceitaram participar dessa banca de dissertação.

Ao grupo de professores do curso de Sistemas de Informação da FALM, que direta ou indiretamente contribuíram para o desenvolvimento dessa pesquisa.

Termino dedicando a todos os meus amigos, em especial aos da faculdade, pela torcida e compreensão com os sucessivos “nãos”, principalmente para o tão esperado “joguinho de bilhar na aula vaga”.

"Viva como se fosse morrer amanhã. Aprenda como se fosse viver para sempre."

**Mahatma Gandhi.**

## RESUMO

A abordagem de linha de produto de software (LP) surgiu da necessidade de técnicas de auxílio à construção de projetos e do desenvolvimento de softwares em um menor tempo e com uma maior qualidade. LP é a área da engenharia de software que tem por objetivo promover a geração de produtos específicos por meio da reutilização de uma infra-estrutura central, conhecida com núcleo de artefatos, para um determinado domínio. A abordagem tem como atividade o gerenciamento de variabilidades, que visa administrar as diferenças e as similaridades entre os produtos de software. O processo, considerado bastante complexo pela literatura, está envolvido em todas as etapas da abordagem de LP. Com base nessa complexidade, surge a necessidade do envolvimento de ferramentas de apoio automatizadas responsáveis no auxílio do processo de gerenciamento. Nota-se porém, a grande carência de materiais na literatura a respeito das características e funcionalidades dessas ferramentas. Diante desse contexto, o trabalho teve como objetivo a elaboração de um estudo de caso, seguindo conceitos da engenharia de software experimental, envolvendo três ferramentas automatizadas de apoio ao gerenciamento. As ferramentas foram utilizadas para a atividade real de gerenciamento de variabilidades de uma LP para Gestão de Processos Seletivos on-line (GPSOL). O estudo permitiu uma análise qualitativa, onde são comparados características funcionais entre as ferramentas envolvidas.

**Palavras-chaves:** Linha de Produto de Software. Gerenciamento de Variabilidades. Estudo de Caso.

## ABSTRACT

The approach of the software product line (LP) arose from the need for technical assistance for the construction of projects and the development of software in a shorter time and with greater quality. LP is the area of software engineering which aims to promote the generation of specific products through the reuse of a central infrastructure, known with the core devices, for a given area. The approach is to manage the activity variabilities, which seeks to manage the differences and similarities between the software products. The process, considered quite complex for literature, this involved in all stages of the approach of LP. Based on this complexity, the need arises to the involvement of tools to support automated responsible in the assistance of the management. There is however, the great shortage of materials in the literature regarding the *features* and functionality of these tools. Considering this context, the study aimed to the development of a case study, following concepts of software engineering trial, involving three tools to support automated management. The tools were used for the actual activity of managing variabilities of a LP for Management Processes Seletives on-line (GPSOL). The study led to a qualitative analysis, which compared the functional characteristics of the tools involved.

**Keywords:** Software Product Line. Managing Variabilities. Case Study.

# LISTA DE FIGURAS

# LISTA DE QUADROS

# LISTA DE ABREVIATURAS

| | |
|---|---|
| **ESE** | Engenharia de Software Experimental |
| **FPS** | Família de Produtos de Software (*Software Product Family*) |
| **FODA** | Feature-Oriented Domain Analysis |
| **GPSOL** | Gestão de Processos Seletivos *On-line* |
| **LP** | Linha de Produto de Software (*Software Product Line*) |
| **PLUS** | Product Line UML-Based Software Engineering |
| **SEI** | Instituto de Engenharia de Software (*Software Engineering Institute*) |
| **UML** | Linguagem de Modelagem Unificada (*Unified Modeling Language*) |

**SUMÁRIO**

# 1 INTRODUÇÃO

## 1.1 Contextualização

O surgimento de atividades voltadas ao desenvolvimento de *software* orientado a objetos e a componentização de *software*, surgidos na década de 80, fizeram com que a comunidade da área da engenharia de software tomasse consciência de que, para a obtenção de um produto com um alto grau de qualidade e que o mesmo fosse economicamente viável, era necessário um conjunto sistemático de processos, técnicas e ferramentas estando a reutilização entre a mais importante desse conjunto. Com a reutilização de partes específicas do processo, pode-se conseguir um software construído em menor tempo, com maior confiabilidade e um enorme melhoramento na manutenibilidade.

Várias são as abordagens e técnicas de reuso de software como: engenharia de domínio, frameworks, padrões, linguagem de padrões, desenvolvimento baseado em componentes, geradores de aplicação e finalmente linha de produto de software (LP), sendo o último uma proposta de construção sistemática e previsível de reuso baseada em uma família de produtos, e também utilizado como base para o estudo desse documento.

Uma LP, é um conjunto de sistemas com características comuns que visam satisfazer um segmento particular do mercado, e que são desenvolvidas de forma sistematizada a partir de um núcleo comum de artefatos (também conhecida como Engenharia de Domínio) (GIMENES; LAZILHA; PRICE 2002). Essa abordagem é adequada a segmentos em que existam uma demanda por produtos dotados de características comuns, mas com pontos de variação bem definidos.

Uma melhor compreensão do domínio, maior reutilização dos artefatos e principalmente uma grande diminuição no tempo para os produtos chegarem ao mercado (*time-to-market*), são umas das principais vantagens encontradas na abordagem (HEYMANS; TRIGAUX, 2003; CLEMENTS; NORTHROP, 2001).

Um fator importante a ser considerado, é que a apresentação de resultados satisfatórios só será realmente percebido em situações onde o processo

de desenvolvimento for extremamente complexo, ou casos em que a linha de produção for extremamente elevada.

Os pontos de variação são responsáveis pela diferenciação dos produtos através de suas características, aparecendo inicialmente na fase de análise de requisitos e posteriormente estendendo-se por todo o processo de desenvolvimento. Dessa forma, pode-se decidir por uma característica do produto tanto em fase de projeto quanto em nível de implementação.

As decisões do projeto baseado na característica, são tratadas como variabilidades. No entanto, percebe-se que ainda há uma grande dificuldade no seu envolvimento em LP, principalmente no processo de gerenciamento (VAN GURP; BOSCH, 2001).

Recentemente várias pesquisas e experiências industriais têm sido realizadas na área que lida com o gerenciamento de variabilidades em LP. Para esse processo deve-se contar com o auxílio de ferramentas automatizadas (VAN GURP; BOSCH, 2001; HEYMANS; TRIGAUX, 2003; SEI, 2004).

Baseado nesse contexto foi realizado um estudo de caso qualitativo, originando assim, uma análise comparativa de 3 ferramentas de apoio ao gerenciamento de variabilidades em LP, em relação as suas funcionalidades para atividade de geração do modelo de *features*. Para o gerenciamento das variações de LP e experimentação das ferramentas foram utilizadas como artefatos da LP os casos de uso do domínio GPSOL (CASTRO, 2007), dotado de pontos de variações e similaridades suficientes para a aplicação da proposta.

## 1.2 Objetivos

### 1.2.1 Objetivo Geral

A análise da utilização de 3 (três) ferramentas de gerenciamento de variabilidades, para a realização do estudo qualitativo comparando as ferramentas.

### 1.2.2 Objetivos Específicos

- uma pesquisa de conceitos relativos à LP e ao gerenciamento de variabilidades em LP;
- pesquisa e estudo de três ferramentas automatizadas de apoio ao gerenciamento de variabilidades em LP;
- adaptação do estudo de caso do domínio GPSOL, utilizado para o gerenciamento de variabilidades e análise das ferramentas;
- realização de uma análise qualitativa, em que valida-se os resultados de comparação das ferramentas, por meio de um estudo de caso qualitativo;
- seleção da ferramenta mais apropriada para o processo de gerenciamento de variabilidades;

## 1.3 Justificativa

Devido a grande complexidade envolvendo o gerenciamento de variabilidades em LP, várias literaturas enfatizam que essa etapa da atividade do desenvolvimento deve sempre contar com o auxilio de ferramentas automatizadas (VAN GURP; BOSCH, 2001; HEYMANS; TRIGAUX, 2003; SEI, 2004). Com isso, surge a necessidade de adoção de uma ferramenta que auxilie de modo satisfatório a complexidade das características do sistema envolvido.

Nota-se porém, a grande carência de materiais de estudo em relação às características e funcionalidades dessas ferramentas no processo de gerenciamento, o que poderia auxiliar de uma forma melhor planejada sua adoção para determinada LP.

Isso pode estar ligado ao fato de que, durante várias décadas, o desenvolvimento de métodos, técnicas e ferramentas foram centrados principalmente na construção de sistemas de software únicos, havendo então poucas ferramentas voltadas ao auxílio do gerenciamento e manipulação de variabilidades em LP (CIRILO; KULESZA; LUCENA, 2007).

## 1.4 Organização do Trabalho

Este documento está organizado da seguinte forma: A contextualização do projeto de pesquisa, os objetivos e a justificativa foram vistos no capítulo 1. Nos capítulos 2 e 3 é apresentada uma fundamentação teórica envolvendo conceitos relativos à LP e das ferramentas de gerenciamento. No capítulo 4 é apresentado o estudo de caso do domínio GPSOL e os etapas que compõem o processo de gerenciamento de variabilidade em LP. O capítulo 5 apresenta a avaliação do processo proposto. Conclusão e trabalhos futuros são vistos no capítulo 6.

# 2 LINHA DE PRODUTO DE SOFTWARE

A abordagem de LP é uma adaptação do que vem sendo estudado e conhecido internacionalmente como *Software Product-Lines* (SPL), e surge como uma evolução das práticas de reuso conhecidas até o momento.

Segundo Cohen (2002) a definição mais aceita pela indústria e pela área acadêmica é o de Clements e Northrop (2001) e Northrop (2002), que diz 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 ativos principais e de uma forma preestabelecida".

Em Chastek (1997), uma LP define uma Família de Produtos de Software (FPS) em um domínio que compartilha características.

Na abordagem de LP, são herdados do inglês três termos de difícil tradução: *features, commonalities* e *variabilities*. O termo *feature* é conceituado em Bosch (2000), como sendo uma construção utilizada para agrupar requisitos relacionados, em outras palavras, pode-se dizer que *features* são uma forma de se abstrair requisitos. Porém não será traduzido nesse documento por entender que o correspondente no português, característica, não traduz o que tecnicamente significa. Os termos *commonalities* e *variabilities* são traduzidos como características comuns e variabilidades, respectivamente (GIMENES; TRAVASSOS, 2002).

Atualmente existem várias abordagens envolvendo LP, e algumas por apresentarem similaridade com seu conceito, ainda são confundidas com ela. A abordagem fundamental para uma LP esta na reutilização de uma infra-estrutura central, conhecida como núcleo de artefatos, formada normalmente por arquiteturas de software reconfiguráveis e seus componentes, dotadas de características particulares e com pontos de variação bem definidos (CLEMENTS; NORTHROP, 2001; GIMENES; TRAVASSOS, 2002; SEI, 2004). Pode-se dizer ainda que algumas técnicas como *frameworks,* padrões, componentes entre outras, favorecem o desenvolvimento de uma LP (GIMENES;TRAVASSOS, 2002).

Na Figura 1 é mostrado como uma LP permite a geração de um conjunto

de aplicações similares e pertencentes a um mesmo domínio (App1, App2, ..., Appn), desenvolvidas a partir de uma arquitetura genérica de LP conformada por artefatos de software (por exemplo: Diagrama de Classes, Casos de Uso, Linguagem de Padrões e modelos próprios da LP, como o modelo de *features*) e de um conjunto de componentes que enriquecem a arquitetura.

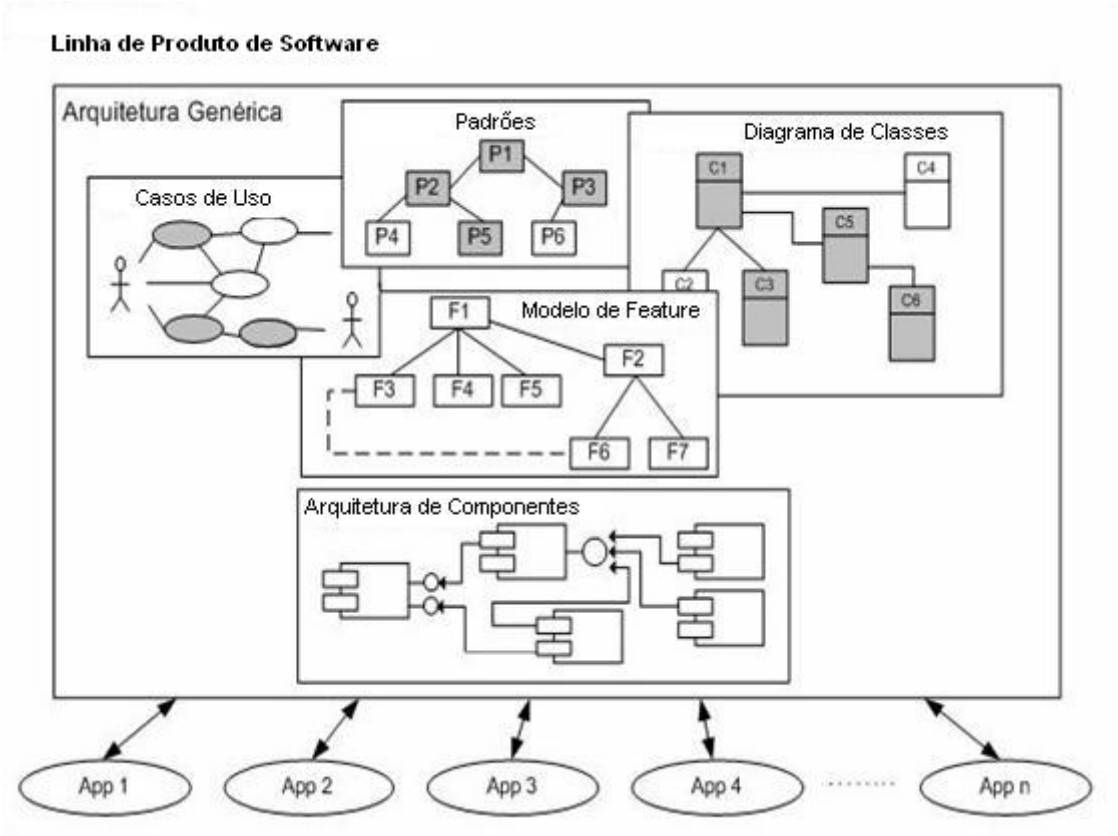



**Figura 1**: Representação de uma LP
Fonte: ARAGON (2004)

Devido aos benefícios conseguidos seguindo LP, é constante a satisfação e a implantação da abordagem em empresas que visam melhores lucros e satisfação do cliente. A Nokia por exemplo, aumentou a produção de modelos diferentes de telefones celulares de 4 modelos/ano para 25-30 modelos/ano.

Para Clements e Northrop (2001), Northrop (2002), Cohen (2002) e SEI (2004), a adoção da abordagem de LP, leva em consideração alguns requisitos essenciais para sua efetiva implantação como: a estrutura da organização, a cultura da empresa em relação à política e os costumes, a natureza dos produtos, o que o negócio visa alcançar, a disposição e a maturidade dos artefatos disponíveis, e a distribuição da equipe de trabalho.

Por muitas vezes, o gerenciamento dessa abordagem requer um longo

período de resposta aos benefícios buscados, lembrando que para alcançar resultados realmente satisfatórios na utilização de LP, sua aplicação deve-se dar em casos onde exista uma grande complexidade do sistema, ou casos onde a produção seja extremamente alta.

Na subseção seguinte serão apresentadas as atividades básicas para o desenvolvimento de uma LP.

## 2.1 Atividades Essenciais de uma LP

O desenvolvimento de uma LP envolve basicamente três atividades distintas (CLEMENTS; NORTHROP, 2001; SEI, 2004):

- **desenvolvimento do núcleo de artefatos –** também conhecida como Engenharia de Domínio;
- **desenvolvimento do produto –** também conhecida como Engenharia da Aplicação;
- **gerenciamento de linha de produto.**

Essas três atividades estão intrinsecamente relacionadas de tal forma que a alteração em uma delas implicaria na análise do impacto nas demais (GIMENES; TRAVASSOS, 2002). Contudo a ordem das duas primeiras atividades pode ser permutada, analisando-se o fato de que durante a fase de desenvolvimento do produto, novos artefatos podem surgir ou evoluir, ou do núcleo de artefatos pode ser extraído de produtos já existentes. Na Figura 2, pode ser visto a iteração das atividades.

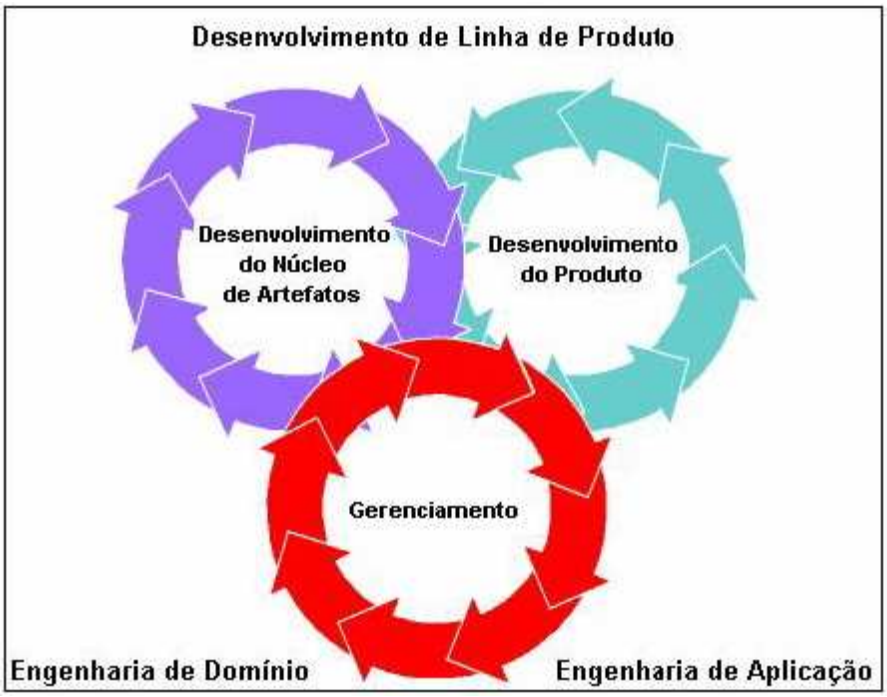



**Figura 2**: Atividades essenciais da linha de produto de software
Fonte: SEI (2004).

Nos itens seguintes serão apresentadas as atividades de uma forma mais detalhada.

### 2.1.1 Desenvolvimento do Núcleo de Artefatos

Esta atividade tem por objetivo estabelecer uma infra-estrutura central que será reutilizada pelos produtos membros gerados por uma LP.

Os artefatos de entrada para essa atividade são (CLEMENTS; NORTHROP, 2001):

- **restrições do produto:** São normas a serem seguidas para a definição de características (variações e semelhanças) entre os produtos que constituirão a LP;
- **restrições de produção:** Baseia-se nos componentes que serão utilizados, e as normas dessa utilização;
- **estratégia de produção:** Estratégia geral para a produção dos artefatos;
- **repositório dos artefatos pré-existentes:** é a catalogação de todos os componentes utilizados nessa fase, para uma futura reutilização.

Os artefatos de saída para essa fase da atividade são:

- **contexto da LP:** descrição dos produtos que constituirão a LP, ou o que esta será capaz de produzir (suas semelhanças, variações e a característica funcional de cada produto);
- **núcleo de artefatos da LP:** base para produção de produtos em uma LP. Envolve a especificação dos requisitos, os modelos de domínio, modelos de desempenho real-time, artefatos gerenciais como cronograma e orçamento, entre outros. Dentre todos os artefatos que fazem parte deste núcleo, a arquitetura é considerada a mais importante, pois define a estrutura dos produtos a serem desenvolvidos;
- **plano de produção de produtos:** descrição das decisões a serem tomadas para instanciação dos produtos específicos a partir do núcleo de artefatos.

Na Figura 3 é ilustrado o processo de entrada e saída da atividade de desenvolvimento do núcleo de artefatos.

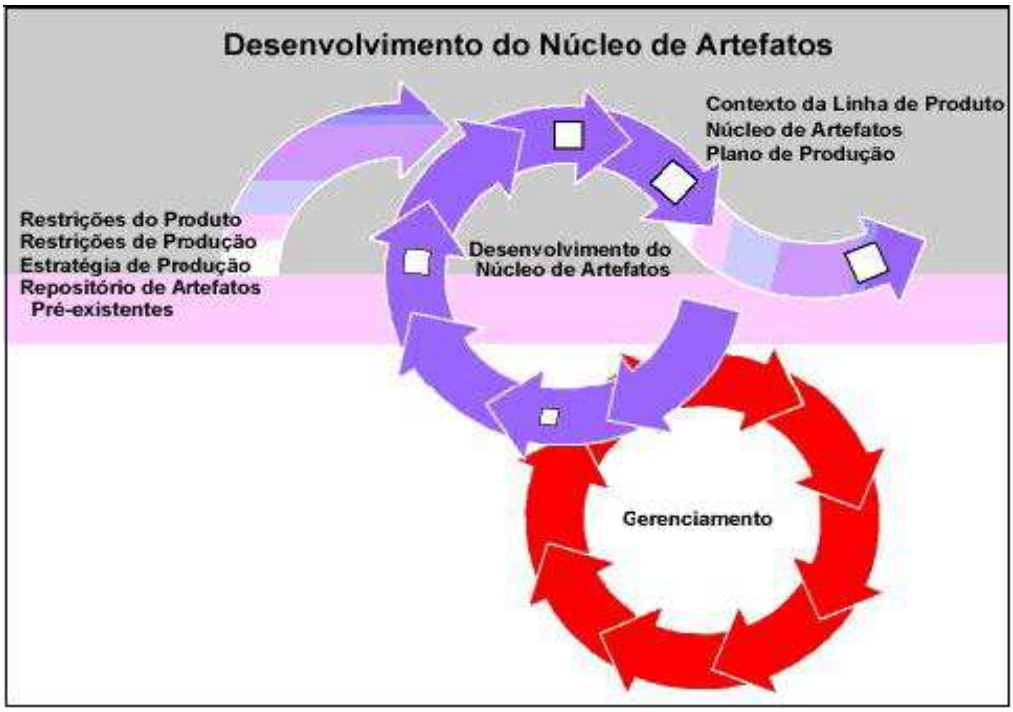



**Figura 3:** Desenvolvimento do núcleo de artefatos
Fonte: SEI (2004)

### 2.1.2 Desenvolvimento do Produto

É nessa atividade de desenvolvimento em que são gerados os produtos da LP. As saídas do desenvolvimento do núcleo de artefatos serão as entradas dessa fase, considerando que, por muitas vezes é identificado requisitos que não haviam sido vistos na fase anterior, portanto pode-se atualizar o núcleo de artefatos.

Na Figura 4 é mostrado a atividade de desenvolvimento do produto.

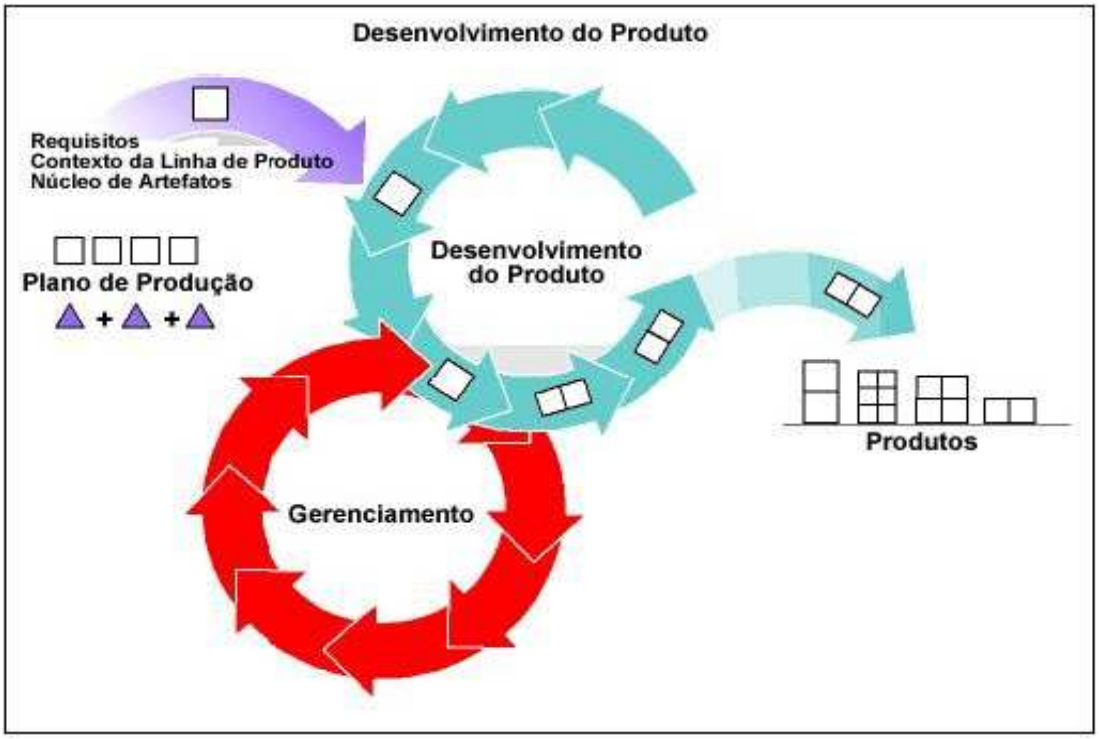



**Figura 4:** Desenvolvimento do Produto
Fonte: SEI (2004)

### 2.1.3 Gerenciamento da Linha de Produto

Essa atividade é responsável pelo bom andamento técnico das atividades de desenvolvimento de uma LP. Deve garantir também a extrema iteração entre as fases de desenvolvimento do núcleo de artefatos e do desenvolvimento de produto, gerenciando as semelhanças e variabilidades, visando assim a evolução da LP.

Em SEI (2004), essa fase de desenvolvimento pode ser dividida em 2 níveis de gerência:

- **gerenciamento técnico:** monitora as fases de desenvolvimento do núcleo de artefatos e de desenvolvimento do produto, fazendo com que os grupos envolvidos coletem dados suficientes para o núcleo de artefato, seguindo a linha de processo de uma LP;
- **gerenciamento organizacional:** cuida para que as unidades organizacionais envolvidas recebam recursos corretos e em quantidade suficientes.

## 2.2 O Conceito de Variabilidade em LP

Segundo Weiss e Chi Tau (1999) variabilidade pode ser definido como a forma que os membros de uma família de produtos podem se diferenciar entre si.

Para a geração de produtos específicos na abordagem de uma LP é necessário representar as variações que podem ocorrer em cada artefato que faz parte desta. Esses artefatos podem se diferenciar em comportamento, atributos de qualidade, plataforma, configuração física, fatores de escala, entre muitos outros (CLEMENTS; NORTHROP, 2001).

Às fases do projeto que ainda não foram resolvidas é dado o nome de ponto de variação, e as alternativas desse ponto é chamado de variantes. Portanto a resolução de um ponto de variação se da pela escolha de uma ou mais variantes ou de um conjunto de variantes (HEYMANS; TRIGAUX, 2003).

Segundo Gimenes e Travassos (2002), variações são diferenças entre produtos reveladas em qualquer fase de desenvolvimento de uma LP a começar pela fase de captura de requisitos, e distribuídas entre os artefatos da LP, sejam eles a arquitetura, os componentes, as interfaces entre componentes ou as conexões entre componentes. Assim, a variabilidade surge do adiamento de certas decisões fundamentais, pois decisões tomadas apenas em fases iniciais do projeto dizem respeito à abordagem tradicional da geração de um único produto, e não da abordagem de LP. Diante desse contexto, pode-se dizer que quanto maior o número de decisões adiadas maior será o número de variabilidades de um produto de software encontradas (HALMANS; POHL, 2003).

Em orientação a objetos, variabilidades podem ser representadas em nível de projeto, por meio de especializações de determinadas classes (GIMENES;

TRAVASSOS, 2002). Por exemplo, suponha uma classe pagamento que de acordo com o produto desejado possa ser boleto, cartão ou depósito. Isto pode ser representado por meio da especialização da classe pagamento.

Um importante conceito para a identificação e representação de variabilidades é o de *feature* (VAN GURP; BOSCH, 2001), que segundo Kang (1990), teve seu inicio na engenharia de domínio. *Feature* é considerado como uma característica do produto relevante e visível para usuários e clientes, e importante para a descrição e distinção de membros de uma FPS (GRISS, 2000).

As *features* podem ser classificadas como sendo (GRISS; FAVARO; D´ALESSANDRO, 1998; VAN GURP; BOSCH, 2001):

- **obrigatórias:** estão sempre presentes e identificam um produto;
- **opcionais:** podem ou não estar presentes em um produto;
- **variáveis:** possuem um conjunto de *features* relacionadas em que zero ou mais dessas *features* podem ser selecionadas para estar presentes em um produto;
- **externas:** são as *features* oferecidas pela plataforma-alvo do sistema.

Além dessas Czarnecki e Eisenecker (2000), Czarnecki, Helsen e Eisenecker (2005) e Oliveira Junior (2005), classificam ainda os grupos de 2 ou mais *features* como:

- **alternativas inclusivas:** indicam que zero ou mais *features* podem fazer parte de uma LP (na notação da Linguagem de Modelagem Unificada (UML) são representadas pela relação OR da lógica booleana);
- **alternativas exclusivas:** indicam que somente uma *feature* do conjunto pode fazer parte de uma LP (na notação da UML são representadas pela relação XOR da lógica boolena).

Seguindo a notação UML (OLIVEIRA JUNIOR, 2005), a relação entre as *features* pode ser de dois tipos:

- ***requires:*** indica que para que uma *feature* possa fazer parte de um produto da LP, uma ou mais *features* devem estar presentes no mesmo produto;
- ***mutex:*** indica que para que uma *feature* possa fazer parte de um produto da LP, uma ou mais *features* não devem estar presentes no mesmo produto.

Várias são as formas para representar as *features* em uma FPS, mas todas tomam como modelo a forma estruturada de árvores, em que os nós correspondem as *features* e seus atributos (GIMENES; TRAVASSOS, 2002).

Na Figura 5 é apresentado um exemplo de modelo de *features* baseado em diagramas de classes da UML proposto em Oliveira Junior (2005), em que as *features* são representadas por classe e que além destas são utilizados três tipos de relações da UML:

- **composição:** indica que uma *feature* é formada por outras *features;*
- **generalização/especialização:** é usada para representar exclusivamente as *features* alternativas exclusivas;
- **dependência:** é utilizada para indicar a relação *requires* e *mutex* entre as *features.*

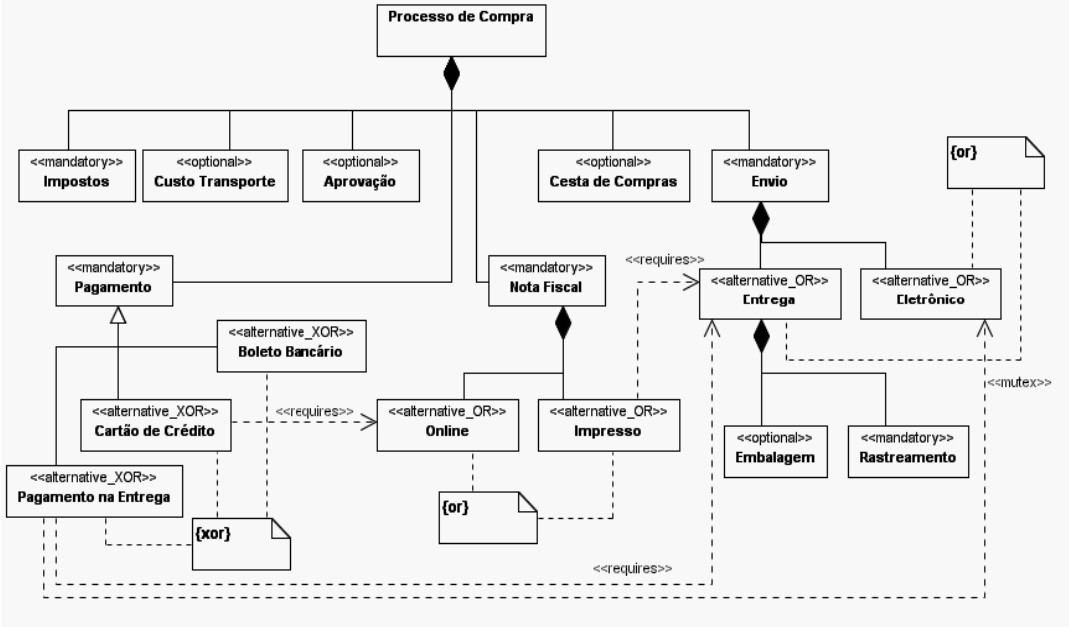



**Figura 5:** Exemplo de modelo de *features* de um processo de compra via internet
Fonte: Oliveira Junior (2005)

Apesar do modelo de *features* apresentar relação com o modelo de casos de uso, existem algumas diferenças importantes entre esses modelos como (GRISS; FAVARO; D´ALESSANDRO, 1998):

- o modelo de casos de uso é baseado para o usuário enquanto o modelo de *features* é baseado para o reutilizador;
- o modelo de casos de uso descreve a funcionalidade do sistema para o usuário, enquanto o modelo de *features* organiza o resultado da análise dos

aspectos comuns e variáveis, preparando uma base para a reutilização;

- a notação utilizada para representar *features* é diferente da utilizada para representar casos de uso;
- enquanto o modelo de *features* inclui apenas as características que o analista do domínio considera importante o modelo de casos de uso deve cobrir todos os requisitos de um sistema do domínio;
- nem todas as *features* aparecem no modelo de casos de uso, pois algumas aparecem em outras fases do projeto.

A seção a seguir apresenta conceitos relativos ao gerenciamento de variabilidades, e um exemplo de elaboração de um modelo de processo de gerenciamento de variabilidades.

## 2.3 Gerenciamento de Variabilidades em LP

O gerenciamento de variabilidades esta relacionado a todas as atividades que compõe o processo de desenvolvimento de uma LP (FRITCH; LEHN; STROHM, 2002; BECKER, 2003). Devido a isso Van Gurp e Bosch (2001), definiram um conjunto de atividades relacionadas a essa etapa tão importante da abordagem de LP:

- **identificação da variabilidades**: com base no modelo de *features* ou nas especificações dos requisitos, deve-se identificar onde as variabilidades são necessárias;
- **delimitação das variabilidades:** delimitar os pontos de variação, o que envolve atividades como definição do tempo de resolução para cada ponto de variação, decisão de como e quando as variabilidades serão adicionadas ao sistema e a escolha da representação para realizar um ponto de variação;
- **implementação das variabilidades:** envolve a escolha da técnica utilizada para implementar as variabilidades;
- **gerenciamento das variantes:** controle das variantes do ponto de variação.

Segundo Oliveira Junior (2005), a elaboração de um processo de gerenciamento de variabilidades é necessário para melhor compreensão do

gerenciamento de uma LP, aumentando assim a qualidade dos artefatos produzidos em relação ao número de produtos constantes em um domínio.

Na Figura 6 é mostrado através de um diagrama de atividades o processo de gerenciamento de variabilidades, composto por atividades e artefatos.

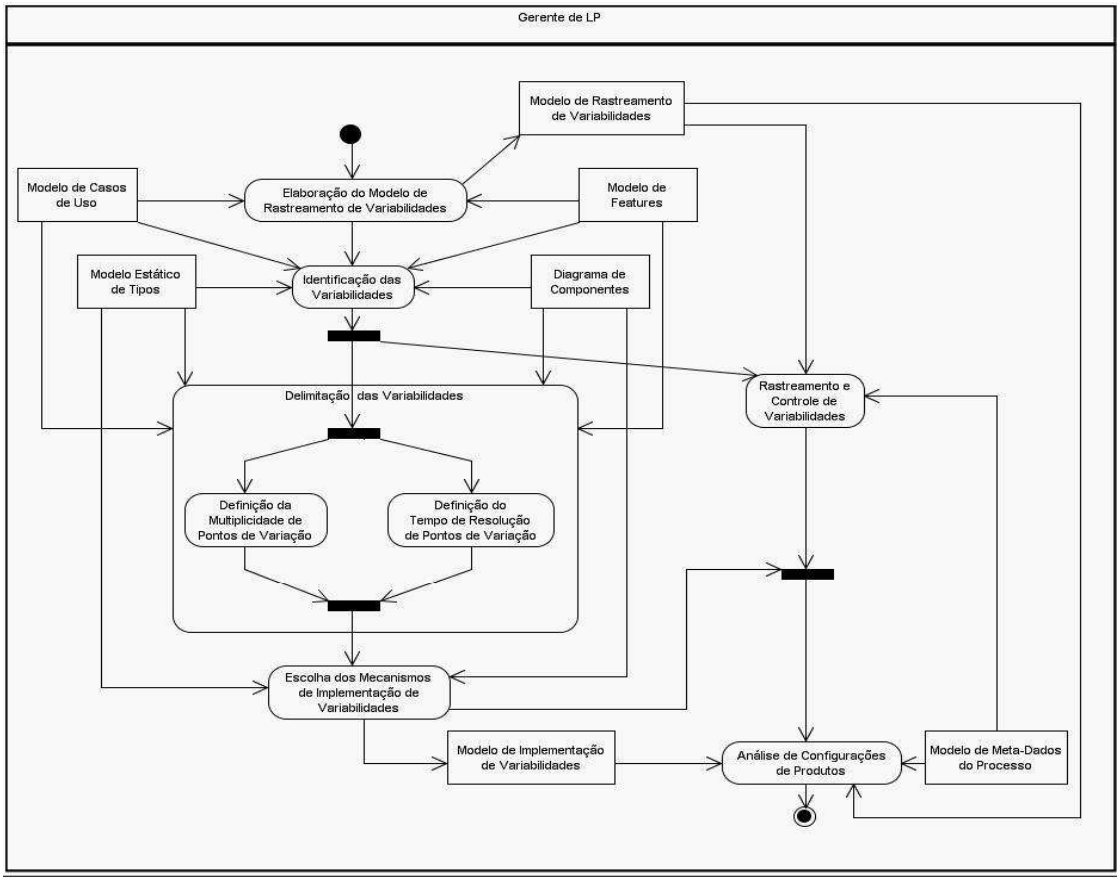



**Figura 6:** Atividades para o processo de gerenciamento de variabilidade para LP
Fonte: Oliveira Junior (2005)

O processo é realizado ao final de cada atividade do desenvolvimento de uma LP, o que o torna interativo. No diagrama as atividades são representadas pelas elipses, e os artefatos de entrada e saída pelos retângulos.

O relacionamento entre as atividades de desenvolvimento de LP, as atividades de gerenciamento de variabilidades e a de geração de produto é visto na Figura 7.

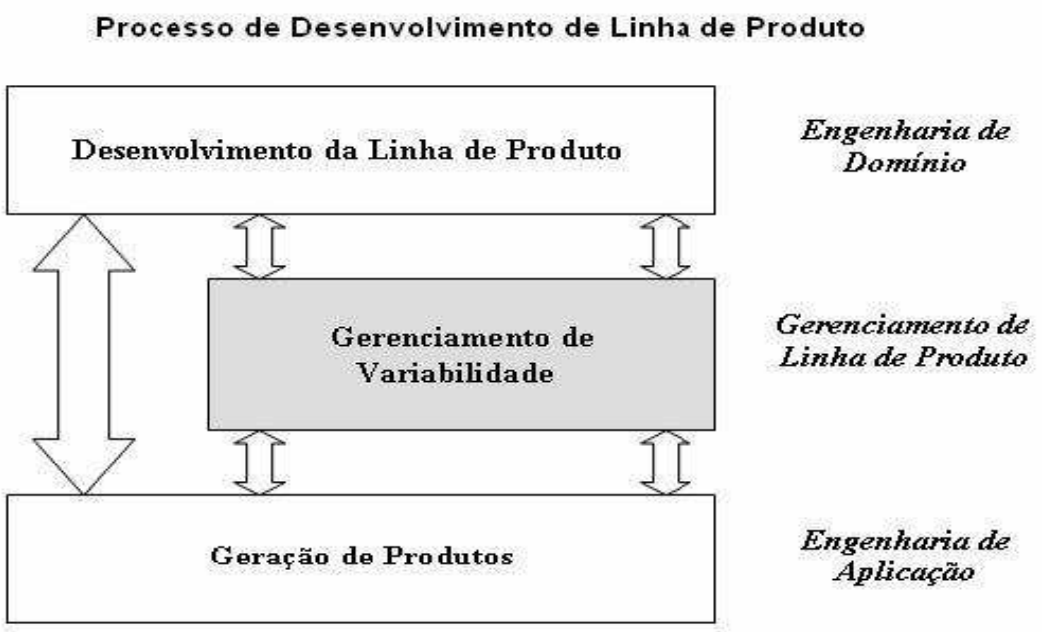



**Figura 7:** Processo de desenvolvimento de LP com gerenciamento de variabilidade
Fonte: Oliveira Junior (2005)

No capítulo seguinte são apresentados conceitos relativos às características técnicas das ferramentas de apoio ao gerenciamento de variabilidades utilizadas para a proposta desse trabalho.

# 3 FERRAMENTAS DE APOIO AO GERENCIAMENTO DE VARIABILIDADES EM LP

O gerenciamento de variabilidades é uma técnica que está envolvida em todas as etapas de desenvolvimento de uma LP. Assim surge a necessidade do uso de técnicas e principalmente de ferramentas automatizadas que auxiliem nesse processo. O gerenciamento de variabilidades deve contar com um apoio automatizado (VAN DER HOEK, 2000; SUCCI; YIP; PEDRYCZ, 2001; VAN GURP; BOSCH, 2001; BEUCHE; PAPAJEWSKI; SCHRÖDER-PREIKSCHAT, 2003; HEYMANS; TRIGAUX, 2003; HALMANS; POHL, 2003; SEI, 2004; CIRILO; KULESZA; LUCENA, 2007).

Na literatura atual são encontradas diversas ferramentas que podem auxiliar no processo de gerenciamento como: Ménage (VAN DER HOEK, 2000), pure:variants (BEUCHE et al., 2003), Holmes (SUCCI; YIP; PEDRYCZ, 2000), Xfeature, Captain Feature (BEDNASCH; ENDLER; LANG, 2002-2004), FeaturePlugin (ANTKIEWICZ; CZARNECKI, 2004), GenArch (CIRILO; KULESZA; LUCENA, 2007), Jude Community (JUDE UML MODELING TOOL, 2007), entre outras.

As ferramentas pure:variants, Xfeature, FeaturePlugin e GenArch são ferramentas utilizadas para a geração do modelo de *features,* sendo implementadas na forma de *plug-in* da plataforma do Eclipse.

Para a realização da proposta, dentre as ferramentas citadas, foram selecionadas três que objetivam o apoio à realização da mesma atividade, mas apresentam características, arquiteturas e configurações distintas entre si: o Captain Feature, FeaturePlugin e Jude community.

Nas subseções a seguir, são apresentadas as características das ferramentas utilizadas, tal como disponibilidade, ambiente gráfico e configurações.

## 3.1 Captain Feature

Captain Feature é uma ferramenta desenvolvida para a análise de domínio de famílias de sistemas de software. Trata-se de uma ferramenta voltada à modelagem de *features* em LP, sendo integrada a um configurador especializado na criação do diagrama de *features.*

Esse documento utilizou a versão Captain Feature 1.0, disponível para download em Sourceforge (2007), na forma de software livre.

A ferramenta surge como sucessora da AmiEDDI-XML[1] (SELBIG, 2002-2004; SELBIG, 2000), sendo desenvolvida por M. Lang, T. Bednasch, U. W. Eisenecker e K. Czarnecki, rendendo diagrama de *features* no estilo *Feature-Oriented Domain Analysis*[2] (FODA) (KANG, 1990), com modelagem baseada na notação de Czarnecki e Eisenecker (2000).

O modelo de *features* desenvolvido na ferramenta é usado para descrever famílias de sistemas, LP's ou domínios com características comuns, variabilidades e explicitar dependências. Os arquivos da base de dados da ferramenta são gerados com extensão XML e organizados em repositórios.

Captain Feature foi desenvolvido em Java, portanto ele se torna disponível tanto para sistemas de plataforma proprietária como Windows e sistemas de plataforma livre como o Linux, necessitando para sua execução a instalação da máquina virtual Java.

A interface gráfica de Captain Feature é mostrada na Figura 8, onde é ilustrada a divisão das suas principais áreas:

- ***SpaceExplorer*:** área onde se tem a visão estruturada em árvore dos repositórios abertos;
- ***Workbench*:** local usado para a edição e configuração do diagrama de *features*.

[1] Primeira ferramenta a suportar a notação de Czarnecki e Eisenecker (2000), para a modelagem de *features.*

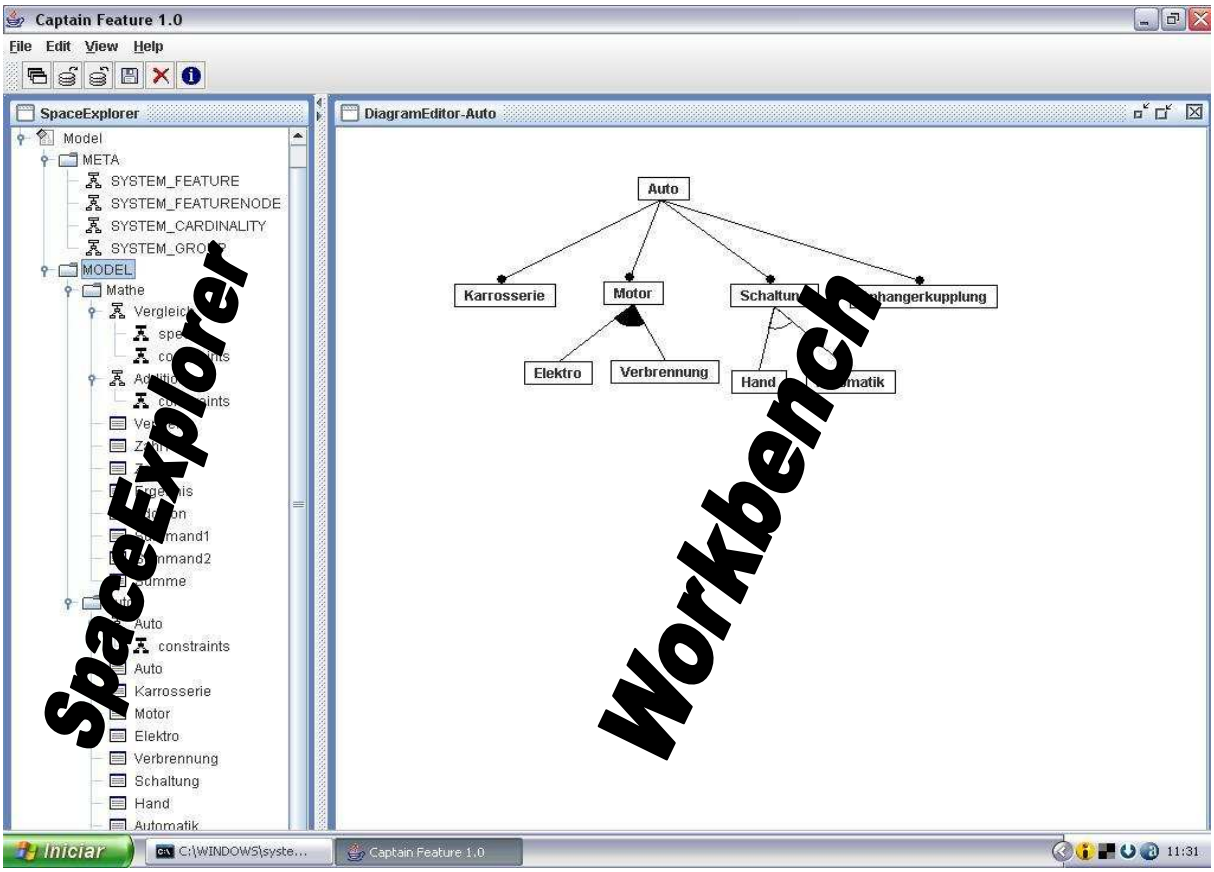



**Figura 8:** Interface gráfica da ferramenta Captain Feature 1.0

## 3.2 FeaturePlugin

Feature Modeling Plug-in (FMP) ou FeaturePlugin, é uma ferramenta que pode ser usada para a configuração e edição dos requisitos de uma LP e as dependências entre elas. Conforme citado anteriormente, é uma das várias ferramentas implementadas na forma de *plug-in* da plataforma Eclipse.

Para a realização desse trabalho foi utilizado a versão Eclipse 3.0.2 disponível em (ECLIPSE PROJECT, 2007) e o FeaturePlugin versão 0.0.6 que segundo Antkiewicz e Czarnecki (2004) pode ser avaliada em http:\\gp.uwaterloo.ca\fmp. A plataforma Eclipse e o FeaturePlugin também são disponíveis na forma de software livre, e necessitam da instalação da máquina virtual Java para sua execução em sistemas operacionais proprietário e livres. A

[2] Um dos precursores da abordagem de LP. Desenvolvido no SEI como um método para análise de domínio. Destaca-se pela introdução do modelo de *features*, amplamente utilizado nas abordagens de LP.

versão do Eclipse utilizada é necessária, pois apesar de existirem outras versões, elas podem gerar problemas de incompatibilidade na execução do *plug-in*.

Para o armazenamento dos arquivos é necessário a criação de um diretório, que por padrão é definido como *workspace*.

A ferramenta organiza as *features* em uma estrutura hierárquica na forma de árvore configurável, fazendo com que a cada novo produto seja especificado como uma configuração de uma *feature.*

FeaturePlugin pode ser usado de forma única na plataforma do Eclipse, ou em conjunto com o *plug-in* FM2RSM (Rational Software Modeler (RSM) ou Rational Software Architect (RSA)) permitindo a modelagem UML da LP.

A interface do Eclipse e a visualização do *plug-in* são vistos na Figura 9, e pode ser dividida em:

- **navegador:** área onde os processos estão dispostos de forma estruturada;
- **área de trabalho:** local onde é feito a modelagem e a visualização hierárquica do modelo de *features;*
- **área de configuração:** local usado para a edição e configuração das *features* ou do grupo de *features* do modelo.

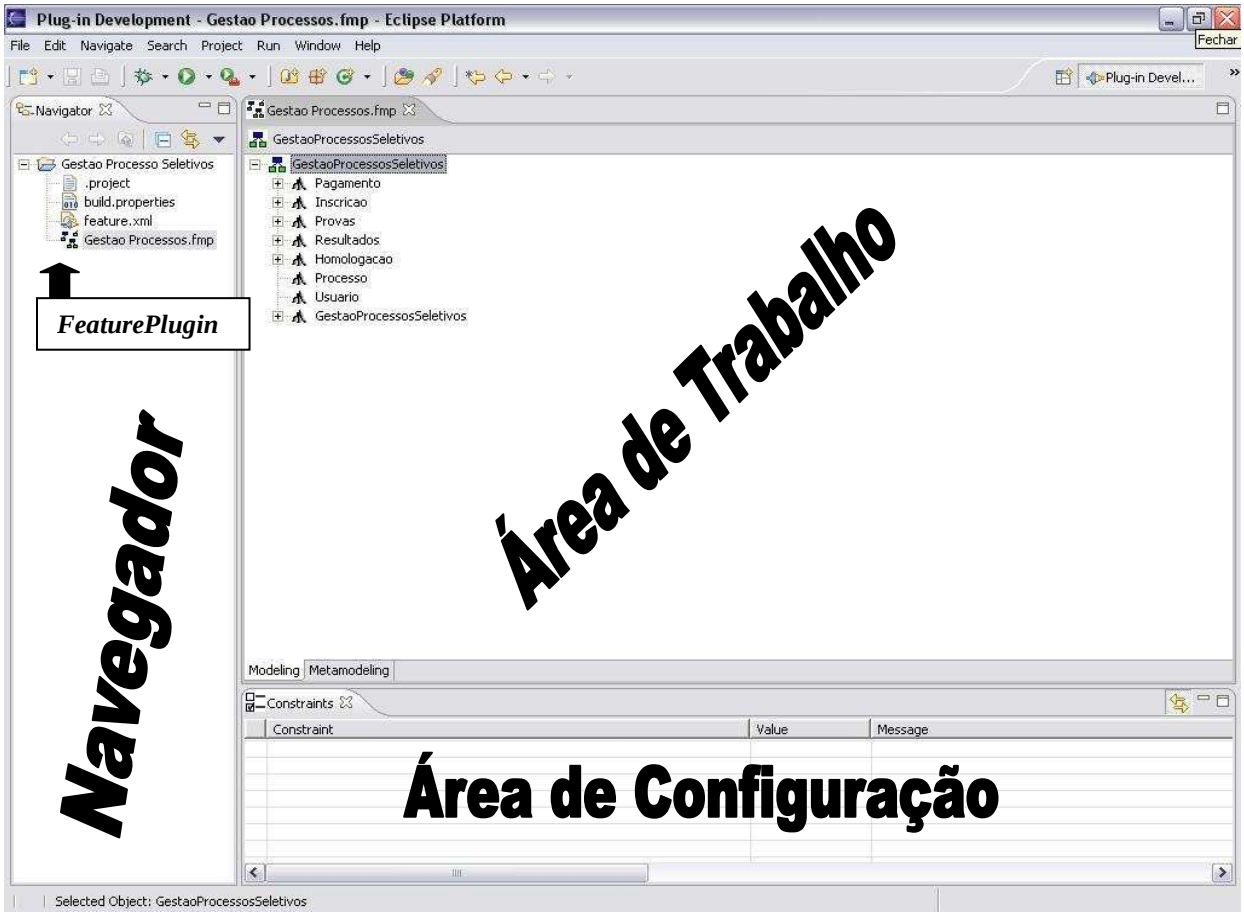



**Figura 9:** Interface gráfica da ferramenta FeaturePlugin 0.0.6

### 3.3 Jude Community

Jude Community é vista como uma ferramenta *case* para a modelagem UML 1.4, baseada na abordagem *Product Line UML-Based Software Engineering*[3] (PLUS) proposta em Gomaa e Webber (2004) e Gomaa (2005), pode ser usada para a modelagem do domínio das fases que envolvem o processo de gerenciamento de variabilidades.

O trabalho utilizou a ferramenta Jude Community versão 5.0.2 disponível em Jude (2007), em versão gratuita. Para a instalação e execução da ferramenta é necessário que a máquina virtual Java também esteja instalada.

O gerenciamento utiliza a notação proposta em Oliveira Junior (2005), para diagrama de casos de uso e desenvolvimento do modelo de *features.*

[3] Abordagem baseada na notação UML para as várias fases que envolvem variabilidades em LP

O ambiente gráfico da ferramenta Jude Community é visto na Figura 10, sendo divido em:

- **Navegador:** local onde encontram-se disponíveis de forma hierárquica os componentes da UML (diagrama de classes, diagrama de casos de uso, diagrama de atividades, etc.);
- **Ambiente de trabalho:** área de visualização gráfica do diagrama aberto, usado para criação, edição e configuração da notação UML.

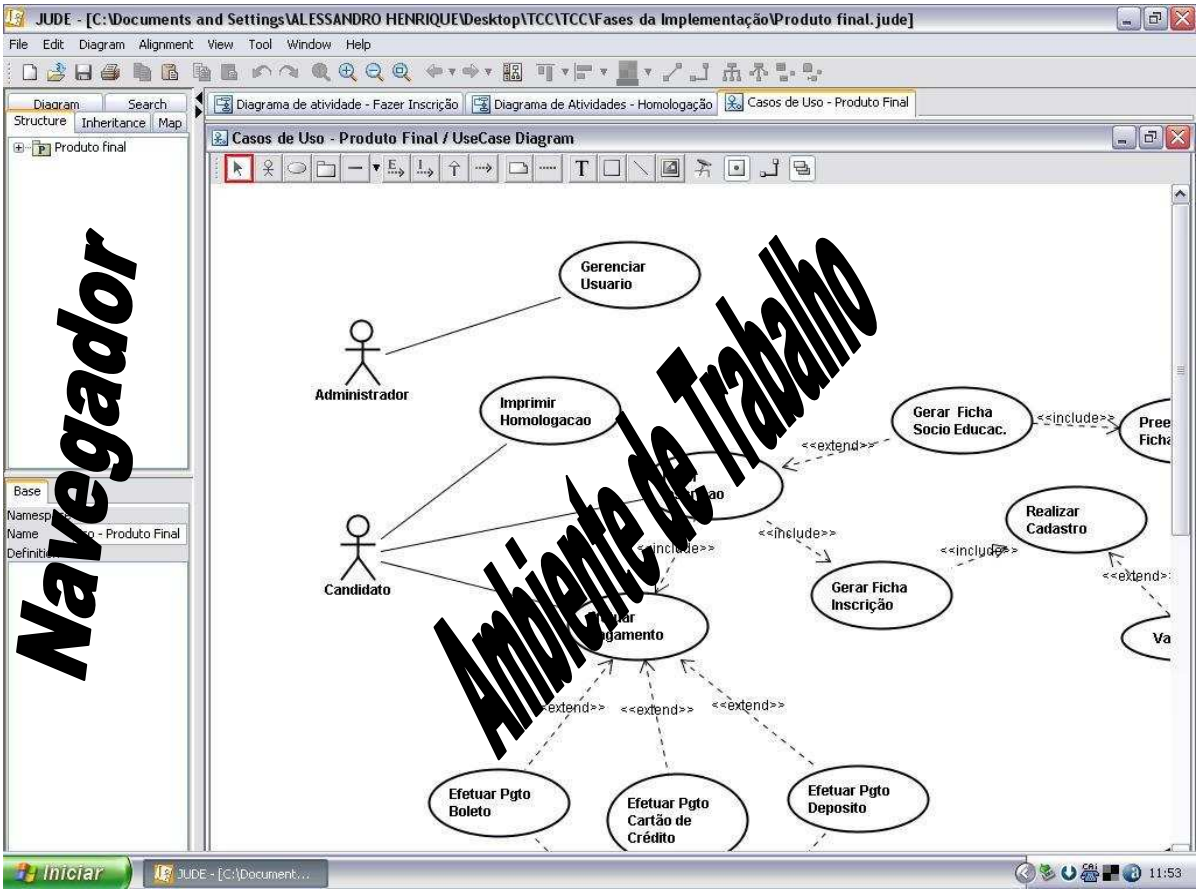



**Figura 10**: Interface gráfica da ferramenta Jude Community 5.0.2

No próximo capítulo utilizando como base o estudo de caso do domínio GPSOL, são apresentadas algumas etapas que envolvem o processo de gerenciamento de variabilidade em LP.

# 4 ESTUDO DE CASO PROPOSTO PARA A ABORDAGEM DE LP

Este capítulo toma como base o domínio GPSOL proposto em Castro (2007). Para a realização da proposta e do processo de gerenciamento, algumas modificações foram necessárias no domínio, por não objetivar, por exemplo, o envolvimento de aspectos na modelagem.

O estudo de caso envolve etapas do processo de gerenciamento de variabilidades em LP, onde através das atividades e especificações é possível chegar à fase de geração dos artefatos (diagramas de casos de uso e modelo de *features*).

A criação dos casos de uso e as etapas do gerenciamento do domínio utilizou como apoio automatizado a ferramenta de modelagem Jude Community.

## 4.1 Descrição Geral do Domínio de Estudo GPSOL

O Sistema baseado na *web* convencional visa o controle de uma gestão de processos seletivos, em que o candidato, único responsável pela realização do seu cadastro no processo, obterá de forma totalmente interativa, informações sobre editais, normas e procedimentos. O candidato poderá ainda, após a realização do pagamento pelos meios fornecidos, verificar através de sua homologação o local onde a prova será aplicada, horários e ensalamentos. O processo de inscrição e de homologação é representado nos diagramas de atividades das Figuras 11 e 12 respectivamente.

O processo deverá contar com um gerenciamento das instituições responsáveis pela formulação das provas, em relação à quantidade de fases que possa ter.

O controle de pagamento deverá ser feito por uma entidade externa (sistema bancário) e será disponibilizado para o gerente do processo uma relação em formato de arquivo texto com os possíveis candidatos que realizaram o pagamento da inscrição, para dar ínicio ao processo de homologação. O candidato

que não realizar o pagamento na data prevista, não será considerado inscrito para o processo seletivo e terá sua inscrição cancelada.

Disponibilização de resultados e de correção de prova também são requisitos que o sistema deverá administrar.

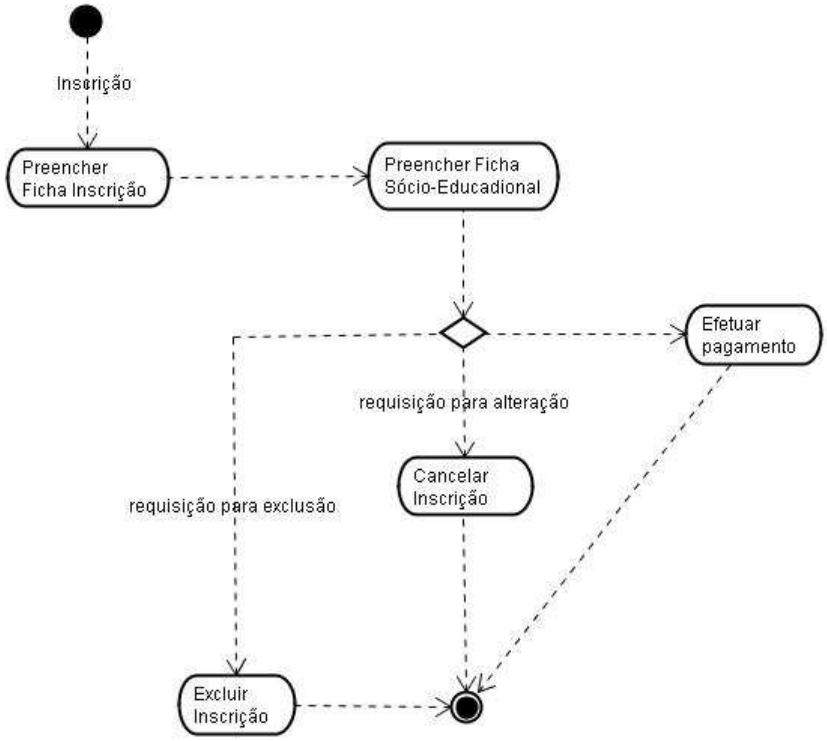



**Figura 11:** Diagrama de atividades - Processo de Inscrição

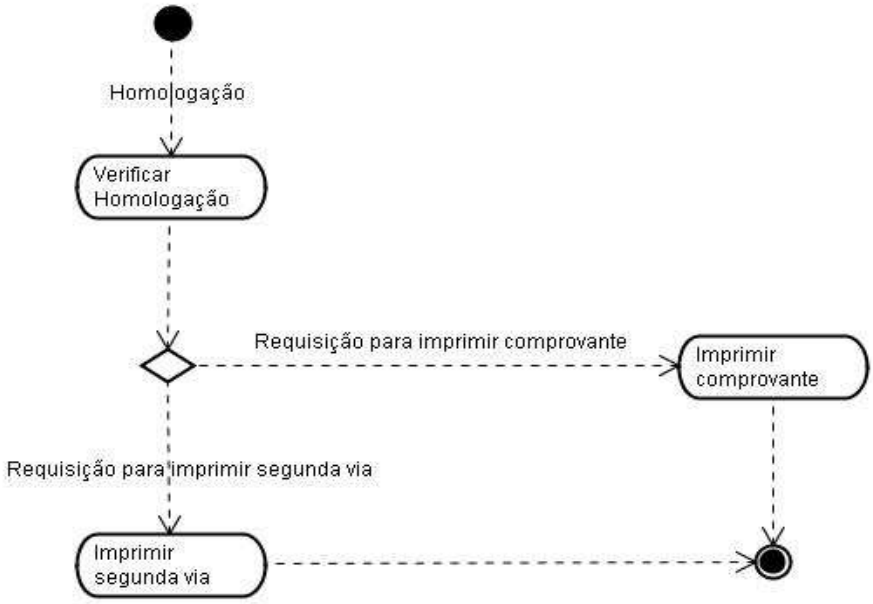



**Figura 12**: Diagrama de atividades – Processo Homologação

## 4.2 Especificação de Requisitos para o desenvolvimento de uma LP

Tendo como base a notação UML esta seção é ilustrada pelo estágio de especificação de requisitos do processo de desenvolvimento de uma LP para GPSOL.

A especificação de requisitos identificou cinco atores cujas responsabilidades são mostradas no Quadro 1.

| ATOR | PAPEL |
|---|---|
| **Administrador** | - administrar o sistema<br>- gerenciar os usuários do processo seletivo |
| **Coordenadores do Processo** | - controlar o sistema geral<br>- cadastrar os dados do processo seletivo<br>- disponibilizar resultados do processo<br>- verificar pagamento do candidato<br>- gerenciar homologação<br>- gerenciar provas<br>- gerenciar locais de prova<br>- gerenciar ensalamento |
| **Candidato** | - preencher a ficha de inscrição on-line<br>- preencher ficha sócio-educacional<br>- efetuar pagamento referente à ficha de inscrição<br>- na data prevista verificar sua homologação e imprimir o comprovante de inscrição |
| **Sistema Bancário** | - gerenciar pagamento on-line<br>- gerenciar pagamento boleto<br>- gerenciar pagamento depósito<br>- fornecer arquivo com candidatos que efetuaram o pagamento para o coordenador do processo |
| **Sistema de Correção** | - disponibilizar a correção da prova para o sistema |

**Quadro 1:** Identificação dos atores/papéis do GPSOL

O artefato gerado nesse estágio é o diagrama de casos de uso para cada um dos atores, representados na Figura 13. O diagrama de casos de uso apresentado servirá como artefato de entrada essencial para a análise das ferramentas de apoio no processo de gerenciamento.

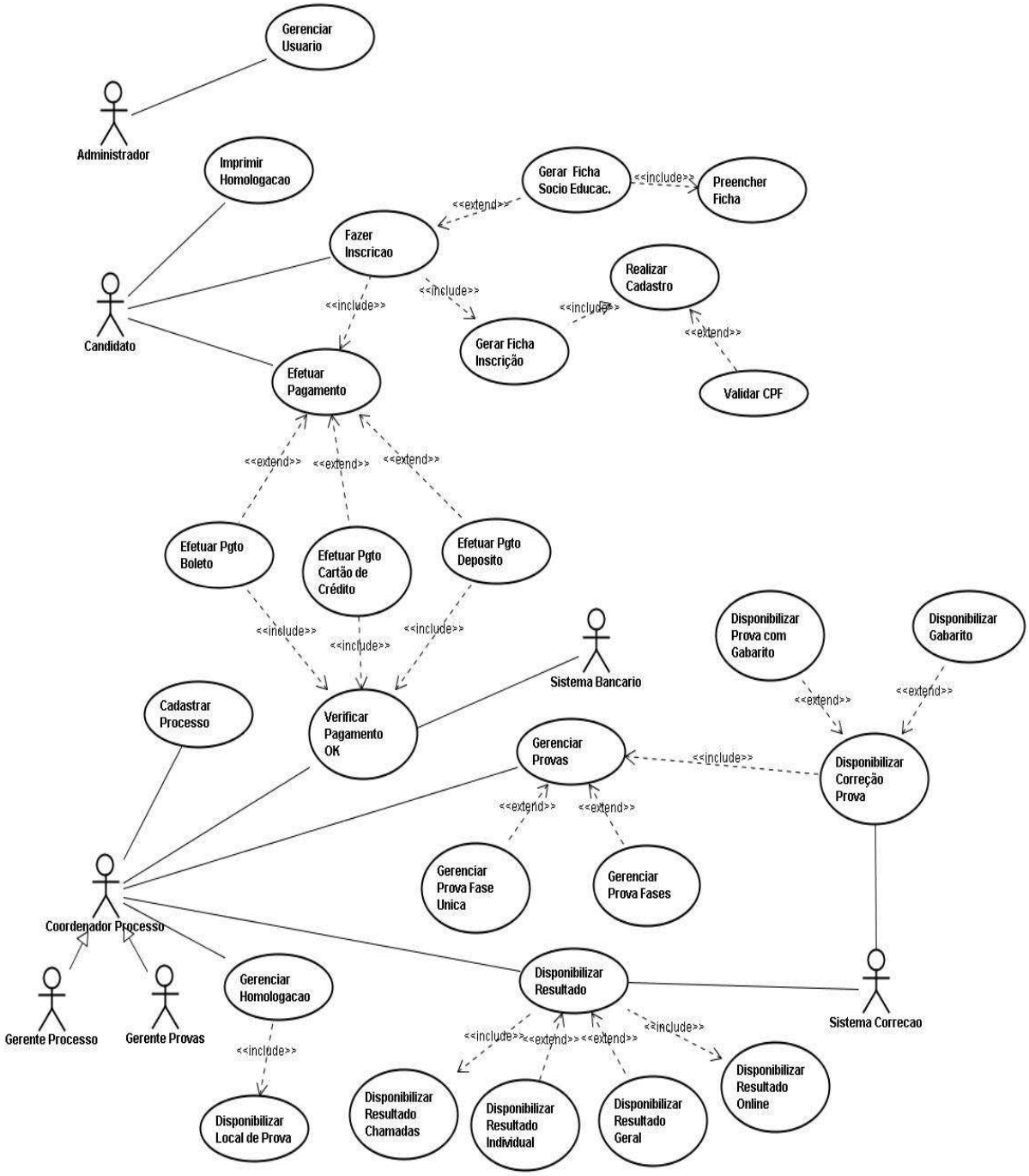



**Figura 13:** Diagrama de casos de uso para GPSOL

Após a geração do diagrama de casos de uso da Figura 16, na próxima seção são mostradas as atividades que compõem o gerenciamento de variabilidades.

## 4.3 Etapas do Gerenciamento de Variabilidade em LP

O processo de gerenciamento de variabilidades utilizou como recursos disponíveis a ferramenta Jude Community 5.0.2, o diagrama de caso de uso GPSOL gerado na seção 4.2 e o processo de gerenciamento de variabilidades proposto em Oliveira Junior (2005).

A atividade de geração do modelo de *features* tem como entrada o diagrama de casos de uso inicial, e é dividido nas etapas abaixo:

1. **identificação das *features*:** fase onde são identificados os casos de uso relacionados entre si que representam pontos de reutilização da LP. Criar uma ou mais *features* para cada grupo de casos de uso identificado. Exemplo: o grupo de casos de uso “efetuar pagamento”, “efetuar pgto boleto”, “efetuar pgto cartão” e ”efetuar pgto depósito”, poderiam gerar a *feature* pagamento composta pelas possíveis variações de *features* “boleto”, “cartão” e ”deposito”;
2. **definição do tipo das *features:*** nessa etapa é definido o tipo de cada *feature* da fase anterior como: obrigatórias (<<mandatory>>), opcionais (<<optional>>), alternativas inclusivas (<<alternative_or>>), alternativas exclusivas (<<*alternative_xor*>>) e externas (<<external>>), podendo a relação entre elas ser do tipo *requires (*<<requires>>*)* e *mutex* (<<mutex>>*).* Os estereótipos encontrados entre parênteses indicam a notação UML utilizada na representação do modelo de *features* por meio de diagrama de classes;
3. **representação gráfica do modelo de *features:*** após as *features* estarem definidas e identificadas é possível representar graficamente as relações entre elas. Para isso utiliza-se a notação da UML permitida pela ferramenta Jude Community, em que as *features* são representadas por classes e indicadas pelas relações de composição (indica que uma *feature* é formada por outras *features*), generalização/especialização (usada exclusivamente para representar as relações entre as *features* exclusivas) e dependências (utilizada para representar as relações *requires* e *mutex* entre *features*).

A Figura 14 apresenta o **modelo de *features*** com as *features* identificadas e definidas para o domínio GPSOL.

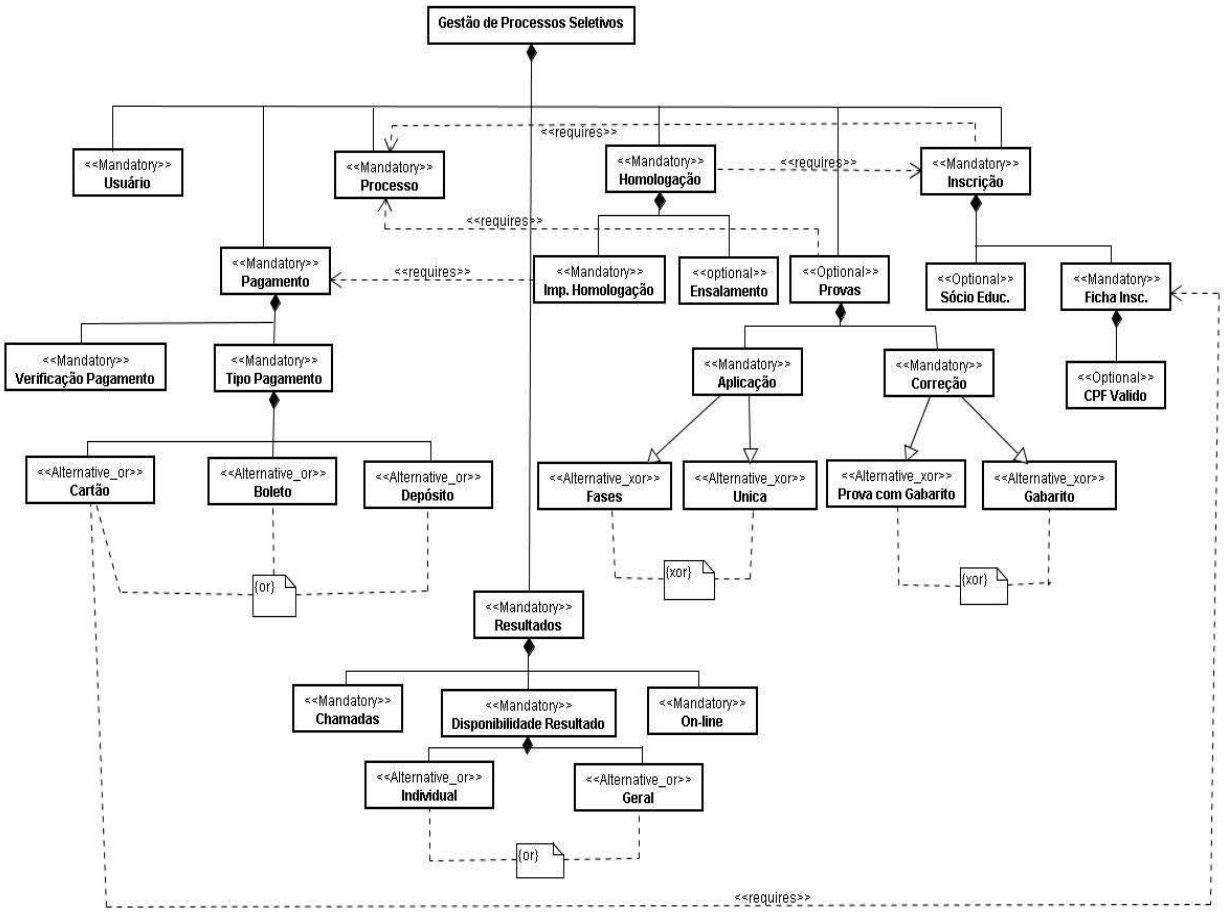



**Figura 14:** Modelo de *features* do domínio GPSOL

Ao término da atividade da elaboração do modelo de *features* é dado inicio a uma nova iteração do processo de gerenciamento de variabilidade, a atividade de **elaboração do modelo de rastreamento de variabilidades,** tendo como artefatos de entrada o modelo de casos de uso e o modelo de *features* desenvolvidos gerados na fase anterior.

Para a construção do modelo de rastreamento de variabilidades, deve-se proceder da seguinte maneira:

1. listar todas as *features* da LP;
2. listar todos os casos de uso da LP;
3. para cada *feature* listada, identificar os casos de uso que originam tal *feature*;
4. marcar com o símbolo "•" o cruzamento entre as *features* e os casos de uso relacionados. Por exemplo, o caso de uso "Fazer Inscrição" origina as *features* "Inscrição", "Sócio Educacional", "Ficha Inscrição", "CPF Válido" e

“Processo”.

Com isso as *features* enumeradas no Quadro 2 são:

| | | | |
|---|---|---|---|
| **1** | Pagamento | **15** | Única |
| **2** | Tipo Pagamento | **16** | Correção |
| **3** | Boleto | **17** | Prova com Gabarito |
| **4** | Depósito | **18** | Gabarito |
| **5** | Cartão | **19** | Inscrição |
| **6** | Verificação Pagamento | **20** | Sócio Educacional |
| **7** | Resultados | **21** | Ficha Inscrição |
| **8** | Chamada | **22** | CPF Válido |
| **9** | On-line | **23** | Homologação |
| **10** | Individual | **24** | Ensalamento |
| **11** | Geral | **25** | Processo |
| **12** | Provas | **26** | Usuário |
| **13** | Aplicação | **27** | Imprimir Homologação |
| **14** | Fases | **28** | Disponibilidade Resultado |

Os casos de uso enumerados no Quadro 2 são:

| | | | |
|---|---|---|---|
| **1** | Efetuar Pagamento | **13** | Disponibilizar Prova com |
| **2** | Efetuar Pagamento Boleto | **14** | Gerenciar Prova |
| **3** | Efetuar Pagamento Cartão | **15** | Gerenciar Prova Fase Única |
| **4** | Efetuar Pagamento Depósito | **16** | Gerenciar Prova Fases |
| **5** | Verificar Pagamento OK | **17** | Fazer Inscrição |
| **6** | Disponibilizar Resultado | **18** | Gerar Ficha Sócio-Educacional |
| **7** | Disponibilizar Resultado Individual | **19** | Gerar Ficha Inscrição |
| **8** | Disponibilizar Resultado Geral | **20** | Validar CPF |
| **9** | Disponibilizar Resultado On-line | **21** | Gerenciar Homologação |
| **10** | Disponibilizar Resultado Chamadas | **22** | Disponibilizar Local Prova |
| **11** | Disponibilizar Correção Prova | **23** | Cadastrar Processo |
| **12** | Disponibilizar Gabarito | **24** | Gerenciar Usuário |

| CASOS DE USO | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *FEATURES* | | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** | **16** | **17** | **18** | **19** | **20** | **21** | **22** | **23** | **24** |
| | **1** | ● | ● | ● | ● | ● | | | | | | | | | | | | | | | | ● | | | |
| | **2** | ● | ● | ● | ● | | | | | | | | | | | | | | | | | | | | |
| | **3** | ● | ● | | | | | | | | | | | | | | | | | | | | | | |
| | **4** | ● | | | ● | | | | | | | | | | | | | | | | | | | | |
| | **5** | ● | | ● | | | | | | | | | | | | | | | | | | | | | |
| | **6** | ● | | | | ● | | | | | | | | | | | | | | | | | | | |
| | **7** | | | | | | ● | ● | ● | ● | ● | | | | | | | | | | | | | | |
| | **8** | | | | | | ● | | | | ● | | | | | | | | | | | | | | |
| | **9** | | | | | | ● | | | ● | | | | | | | | | | | | | | | |
| | **10** | | | | | | ● | ● | | | | | | | | | | | | | | | | | |
| | **11** | | | | | | ● | | ● | | | | | | | | | | | | | | | | |
| | **12** | | | | | | | | | | | ● | ● | ● | ● | ● | ● | | | | | | | | |
| | **13** | | | | | | | | | | | | | | ● | ● | ● | | | | | | | | |
| | **14** | | | | | | | | | | | | | | ● | | ● | | | | | | | | |
| | **15** | | | | | | | | | | | | | | ● | ● | | | | | | | | | |
| | **16** | | | | | | | | | | | ● | ● | ● | | | | | | | | | | | |
| | **17** | | | | | | | | | | | ● | | ● | | | | | | | | | | | |
| | **18** | | | | | | | | | | | ● | ● | | | | | | | | | | | | |
| | **19** | | | | | | | | | | | | | | | | | ● | ● | ● | ● | | | | |
| | **20** | | | | | | | | | | | | | | | | | ● | ● | | | | | | |
| | **21** | | ● | | | | | | | | | | | | | | | ● | | ● | ● | | | | |
| | **22** | | | | | | | | | | | | | | | | | ● | | ● | ● | | | | |
| | **23** | | | | | | | | | | | | | | | | | | | | | ● | ● | | |
| | **24** | | | | | | | | | | | | | | | | | | | | | ● | ● | | |
| | **25** | | | | | | | | | | | | | | | | | ● | ● | ● | ● | | | ● | |
| | **26** | | | | | | | | | | | | | | | | | | | | | | | | ● |
| | **27** | | | | | | | | | | | | | | | | | | | | | ● | | | |
| | **28** | | | | | | ● | ● | ● | | | | | | | | | | | | | | | | |

**Quadro 2:** Modelo de rastreamento de variabilidades para GPSOL

Com o modelo de rastreamento é possível a partir de uma *feature* rastrear as variabilidades até chegar aos casos de uso e aos demais artefatos da UML, como diagramas de classes e diagramas de componentes. O artefato se torna importante também, pois é possível saber exatamente quais artefatos sofrem alterações com a adição e/ou remoção de *features* de um determinado produto.

Após a geração do modelo de rastreamento de variabilidades, dá-se inicio a etapa de **identificação e delimitação das variabilidades** no diagrama de casos de uso inicial. Para esse processo são propostas algumas orientações que podem ser seguidas:

- variantes do tipo alternativas inclusivas ou exclusivas, normalmente são identificadas através da relação com o estereótipo <<extends>>. Assim os casos de uso, que representam variantes, são identificadas com um dos estereótipos <<alternative_OR>> ou <<alternative_XOR>>, enquanto os pontos de variação são representados com o estereótipo <<variationPoint>>;
- variantes do tipo opcionais e obrigatórias são normalmente identificadas com uma simples relação de associação;
- casos entre relações de variantes, onde para que uma exista a outra deve também existir, ou para que uma exista a outra não deva estar presente do produto são representados pela relação de dependência com os estereótipos <<requires>> e <<mutex>>, respectivamente.

Utilizando-se notas da UML é possível também representar informações adicionais sobre os pontos de variação e variantes, como o nome da variação, o seu tipo e a delimitação das variabilidades, sendo que:

- **{ }:** significa que a variante é obrigatória ou opcional;
- **{or}:** significa que a variante é alternativa inclusiva;
- **{xor}:** significa que a variante é alternativa exclusiva;
- **multiplicidade do ponto de variação**: a definição da multiplicidade deve considerar o tipo de relação existente entre o ponto de variação e as suas variantes. Pontos de variação com relações do tipo opcional sempre terão sua multiplicidade definida como 0 (zero), uma vez que pode ou não ser escolhida. Pontos de variação do tipo obrigatória e exclusiva, sempre terão sua multiplicidade definida como 1 (um). Ponto de variação com relações do

tipo alternativa inclusiva podem variar sua multiplicidade definidas de 0 (zero) até o número de variante associadas ao ponto de variação;

- **tempo de resolução:** indica em que momento as variantes do conjunto associado ao ponto de variação serão escolhidas, por exemplo: tempo de projeto, implementação, compilação, execução, etc.;
- **adição de variantes:** determina se o ponto de variação permite a adição de mais variantes (*true* e *false*).

Um exemplo da seqüência acima é representada respectivamente na nota da UML da Figura 15 abaixo.


**Figura 15:** Exemplo de nota da UML com variabilidade identificada e delimitada

A Figura 16 apresenta a evolução do modelo de casos de uso da Figura 13 de um processo de GPSOL, com a representação gráfica de identificação de variabilidades, multiplicidade e tempo de resolução dos pontos de variação.

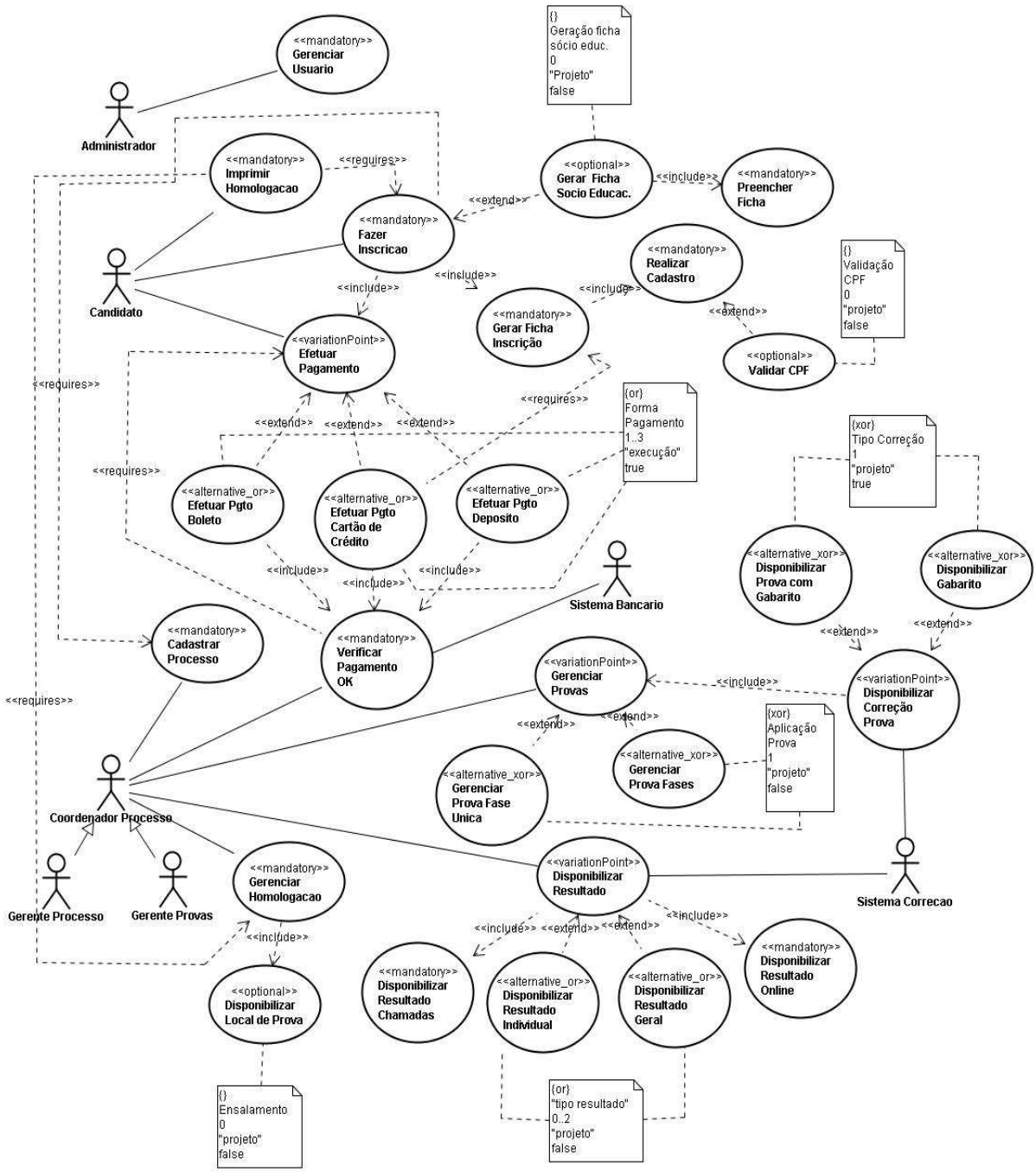



**Figura 16:** Diagrama de casos de uso para GPSOL com as variabilidades identificadas e delimitadas

A **escolha dos mecanismos de variabilidades**, **rastreamento e controle de variabilidades** e **análise de configuração de produtos específicos** são as atividades seguintes do processo de gerenciamento de variabilidades, porém não serão analisadas nesse trabalho por não fazerem uso específico das ferramentas de gerenciamento envolvidas.

# 5 AVALIAÇÃO DO PROCESSO

Este capítulo apresenta o artefato de saída do processo de gerenciamento de variabilidades, o modelo de *features* da LP sendo modelado nas ferramentas escolhidas.

Este capítulo apresenta também uma avaliação dessas ferramentas de auxílio ao gerenciamento de variabilidades em LP. O estudo de caso realizado para a avaliação das ferramentas, foi baseado no conceito de Engenharia de Software Experimental (ESE), utilizando como método o estudo de caso qualitativo.

As seções seguintes apresentam o estudo de caso realizado, precedido pelos conceitos de ESE na avaliação de métodos e ferramentas.

## 5.1 Metodologia Para Avaliação do Trabalho

Na engenharia de software o método experimental se caracteriza por possibilitar a verificação de teorias, explorar fatores críticos e permitir que novos fatores possam ser formulados e corrigidos (TRAVASSOS; GUROV; AMARAL, 2002).

Segundo Conradi et al. (2001), a literatura oferece ainda algumas outras definições dos objetivos da experimentação em engenharia de software listados em:

- A experimentação pode ajudar a construir uma base de conhecimento confiável, reduzindo assim incertezas sobre quais teorias, ferramentas e metodologias são adequadas;
- A tecnologia vem se modificando rapidamente, e as mudanças sempre trazem ou eliminam as suposições. Os pesquisadores devem então aplicar experimentos para explorar as conseqüências dessas mudanças;
- A experimentação pode acelerar o processo eliminando abordagens inúteis e suposições errôneas.

Novos métodos, técnicas, linguagens e ferramentas não deveriam ser

apenas sugeridos e publicados sem experimentação (TRAVASSOS; GUROV; AMARAL, 2002).

Segundo Kitchenham (1996), existem nove métodos experimentais agrupados em três categorias: métodos quantitativos, qualitativos e híbridos. Uma breve definição sobre os métodos experimentais quantitativos e híbridos segundo Kitchenham (1996), é dada em:

- **métodos quantitativos:** o estudo quantitativo é conduzido através de um experimento controlado, permitindo que os dados do estudo sejam comparados e analisados estatisticamente. Esse tipo de método é organizado em estudo de caso quantitativo, *survey* quantitativo e experimento formal quantitativo;
- **métodos híbridos:** O estudo híbrido caracteriza-se por possuir elementos quantitativos e qualitativos, sendo divididos em análise qualitativa de efeitos e *Benchmark*.

Informações mais detalhadas sobre esses dois métodos são encontradas em Kitchenham (1996), Kitchenham, Pickard, Pfleeger (1995), Basili, Selby, Hutchend (1986), Pfleeger (1995), Travassos, Gurov, Amaral (2002) e Amaral (2003).

Neste trabalho foi realizado o estudo de caso qualitativo como método experimental, portanto é apresentado um melhor entendimento sobre o método na seção a seguir.

### 5.1.1 Métodos Qualitativos

O estudo qualitativo esta relacionado à pesquisa sobre métodos/ferramentas quando os resultados são apresentados em termos naturais (TRAVASSOS; GUROV; AMARAL, 2002). O método é aplicado fazendo uma análise sobre as características identificadas através dos requisitos dos usuários para um determinado método ou ferramenta (KITCHENHAM, 1996). Essa análise pode ser realizada por uma única pessoa que identifica os requisitos, mapeia-os para as características, então avalia os métodos ou ferramentas.

Segundo Kitchenham (1996), os métodos qualitativos podem ser classificados em:

- **estudo de caso qualitativo:** análise baseada em características de um método ou ferramenta após este ter sido usado na prática ou em projetos reais;
- ***survey* qualitativo:** análise baseada em características de um método/ferramenta, realizada por pessoas que possuem experiência ou estudam o método/ferramenta. Deve ser organizado na forma de questionários sobre as características do método ou ferramenta;
- **experimento formal qualitativo:** análise baseada em características de um método/ferramenta realizada por um grupo de potenciais usuários. Deve ser organizado na forma de um experimento formal, escolhendo potenciais usuários de forma aleatória;
- **seleção qualitativa *(screenin*g):** análise baseada em características de um método/ferramenta, realizada por um único indivíduo que não só determina as características a serem avaliadas e suas escalas, mas também realiza a avaliação.

Em Audy (2002), é descrita as principais diferenças encontradas entre abordagens quantitativas e qualitativos, observado no Quadro 3 abaixo.

| OBJETIVOS/MÉTODOS | Quantitativo | Qualitativo |
|---|---|---|
| **Papel da pesquisa qualitativa** | Preparatória | Interpretação |
| **Relação entre pesquisador e objeto** | Distante | Próxima |
| **Posição do pesquisador com o objeto** | De fora | De dentro |
| **Relação entre a teoria/conceitos e a pesquisa** | Confirmação | Emergente |
| **Estratégia de pesquisa** | Estruturada | Não estruturada |
| **Escopo das descobertas** | Nomotética | Ideográfica |
| **Imagem da realidade social** | Estática | Processual |
| **Natureza dos dados** | Confiável, rigoroso | Rica, profunda |

**Quadro 3:** Diferenças entre métodos quantitativos e métodos qualitativos
Fonte: Audy (2002)

## 5.2 O Estudo de Caso – O Uso e a Comparação das Ferramentas

A avaliação do processo proposto nesse trabalho foi realizada na forma de um estudo de caso qualitativo.

A escolha do estudo de caso toma como base o Quadro 4 apresentado, além de outros fatores como:

- estudo de caso pode ser realizado por uma única pessoa;

- o estudo de caso não precisa envolver a aplicação de questionários, reduzindo o trâmite de documentos;
- o estudo de caso não requer a existência de uma *baseline* para comparar os dados obtidos.

| FATOR/MÉTODO | *Survey* | Estudo de Caso | Experimento |
|---|---|---|---|
| **O controle da Execução** | Nenhum | Nenhum | Tem |
| **O controle da medição** | Nenhum | Tem | Tem |
| **O controle da investigação** | Baixo | Médio | Alto |
| **Facilidade da repetição** | Baixo | Médio | Alto |
| **Custo** | Baixo | Médio | Alto |

**Quadro 4:** Comparação entre métodos experimentais mais utilizados
Fonte: Travassos, Gurov e Amaral (2002)

Nos itens seguintes, é visto a análise das ferramentas automatizadas envolvendo a atividade de modelagem das *features*, oriundas da etapa do processo de gerenciamento de variabilidades em LP.

### 5.2.1 A modelagem das *features* na ferramenta Jude Community

A atividade envolvendo a geração do modelo de *features* na ferramenta Jude Community foi realizada no processo de gerenciamento de variabilidades da seção 4.3. A Figura 14 apresenta o modelo de *features* para o processo GPSOL.

A ferramenta foi usada de uma forma totalmente intuitiva, pois sua interface baseada em modelagem gráfica de fácil compreensão facilitou muito no auxílio às técnicas de gerenciamento aplicadas no capítulo 4.3 utilizando a notação UML.

Além da geração do modelo de casos de uso, artefato característico da notação UML ligado à ferramenta, criou-se também de forma bastante adaptativa o modelo de *features,* artefato característico da abordagem de LP.

Com a ferramenta é possível um gerenciamento inicial ligado diretamente aos artefatos da UML como, casos de uso, diagramas de classe e diagramas de componentes.

A apresentação da evolução do modelo de caso uso apresentado na

Figura 16, seguindo a notação UML, deu-se por meio da utilização dos componentes dispostos na barra de ferramentas do ambiente de trabalho, mostrado na Figura 17.


**Figura 17**: Barra de ferramentas do diagrama de casos de uso - Jude Community

Os componentes envolvidos nessa atividade foram: Ator (*actor*), casos de uso (*use case*), relação de dependência (*dependency*) - para o uso das relações *mutex* e *requires* -, nota (*note*) - para informações de identificação e multiplicidade de variabilidades e relação entre *features* alternativas e exclusivas -, extensão (*extend*) e de inclusão (*include*).

Para a geração do modelo de *features* utilizou-se a atividade de diagrama de classes (Class Diagram), representada no navegador na guia *Estructure.* A barra de ferramentas dessa atividade apresenta algumas modificações em relação a barra de ferramentas da geração do diagrama de casos de uso. Tais diferenças são vistas na Figura 18.

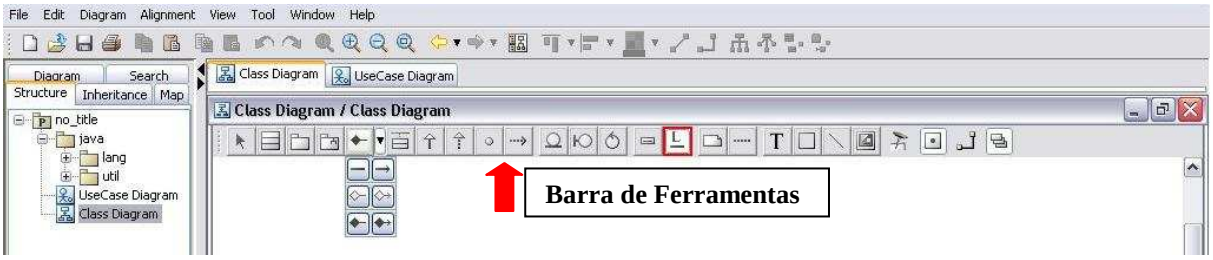



**Figura 18:** Barra de ferramentas do diagrama de classes - Jude Community

Os componentes envolvidos nessa atividade foram: Classe (*class*), composição, relação de dependência (*dependency*) - para o uso das relações *mutex* e *requires* -, nota (*note*) - para informações de identificação e multiplicidade de variabilidades e relação entre *features* alternativas e exclusivas -, linha (*line*) - para dar acabamento na união de *features*.

Além dos componentes utilizados nas atividades, também foram atribuídas propriedades aos casos de uso com o uso de estereótipos indicando pontos de variação, (<<variationPoint>>) e dos tipos de *features* (<<alternative_or>>, <<exclusive_or>>, <<optional>> e <<mandatory>>).

A ferramenta permite também a exportação das modelagens para extensões do tipo Java, HTML e tipo figura (JPEG) e importação de arquivos de extensão Java.

Apesar das vantagens no uso da ferramenta para o auxílio do gerenciamento de variabilidades, notou-se uma dificuldade no relacionamento entre grupos de *features,* em que a utilização do componente *line,* é usada para "mascarar" a ligação entre elas.

### 5.2.2 A modelagem das *features* na ferramenta FeaturePlugin

O auxílio do gerenciamento de variabilidades em relação à ferramenta FeaturePlugin, está relacionada à criação, configuração e edição do modelo de *features* organizado hierarquicamente em uma estrutura de árvore. A ferramenta é baseada na cardinalidade de *features.*

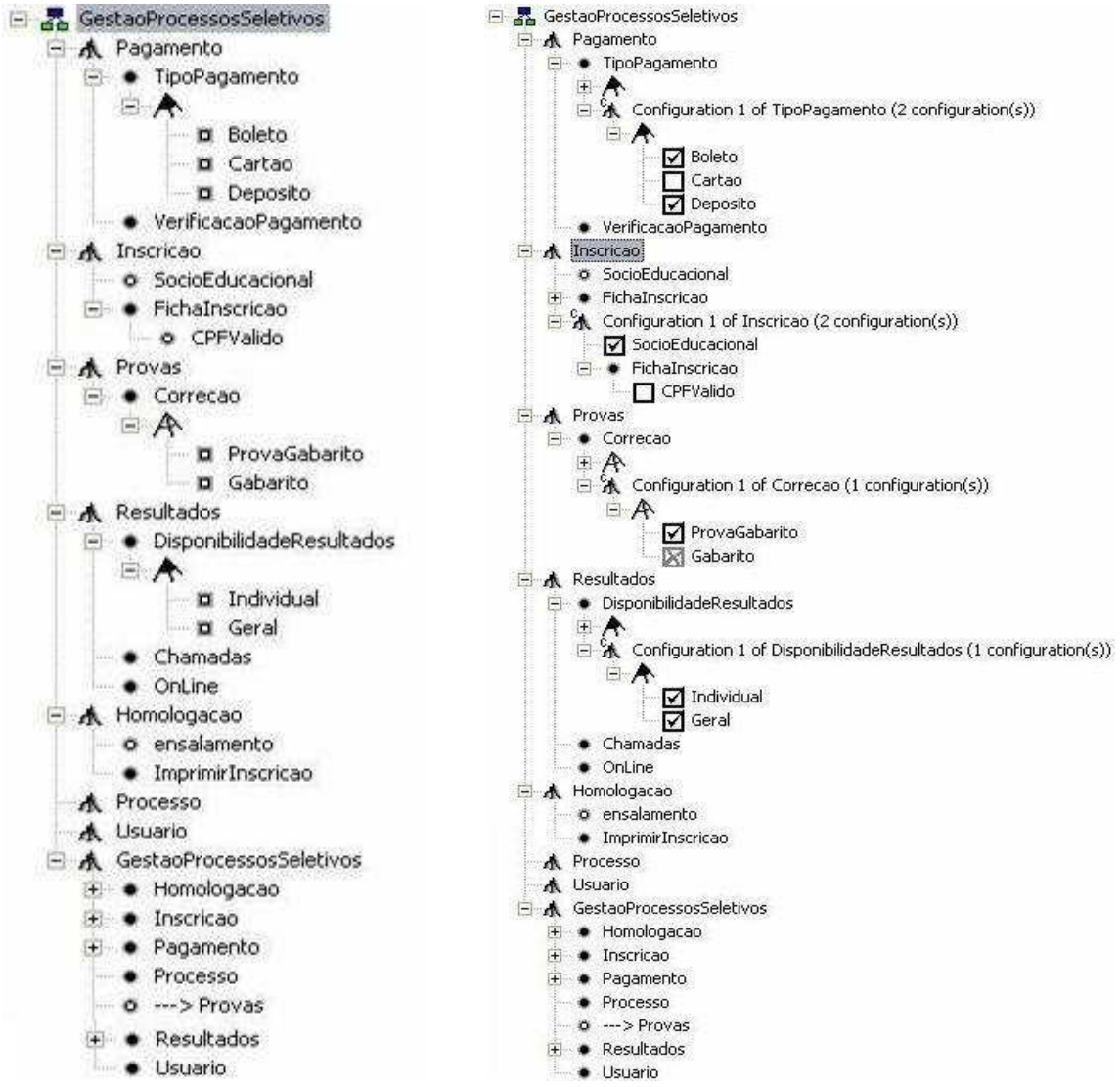
O processo tomou como recursos disponíveis a ferramenta FeaturePlugin, a plataforma Eclipse para a execução do *Plug-in*, o diagrama de casos de uso GPSOL (gerado na seção 4.2 e representado na Figura 13) e a notação de Antkiewicz e Czarnecki (2004).

A Figura 19(a) apresenta o modelo de *features*, visto como estrutura hierárquica no *plug-in.* Na estrutura são identificados 8 diagramas de *features,* tendo a *feature* raiz indicada pelo símbolo de raiz . A raiz da *feature* "Pagamento" é composta por duas subfeatures: "TipoPagamento" e "VerificacaoPagamento". O símbolo indicado na *feature* "TipoPagamento", significa uma cardinalidade [1..1], ou seja, a *feature* obrigatoriamente deve pertencer a LP. Já a *feture* opcional com cardinalidade [0..1], é indicada pelo símbolo encontrada na *feature* "*V*erificacaoPagamento". As *features* "Cartao", "Deposito" e "Boleto" encontrados na hierarquia de "TipoPagamento" indicam que são membros de um grupo de *fetures.* Esse grupo de *features* é indicado pelo símbolo , significando que o grupo possui cardinalidade (1-k), onde k é o tamanho do grupo, ou seja, o grupo possuí relação do tipo alternativas inclusivas (0 ou mais *features* podem estar presentes na LP). A relação com cardinalidade (1-1) é representada pelo símbolo , indicando a relação do tipo alternativa exclusivas (somente uma *feature* do conjunto pode fazer

parte da LP), pode ser vista no grupo de *features* “ProvaGabarito” e “prova” membros da *feature “C*orreção”.

Ao final a raiz “GestãoProcessosSeletivos” é referenciada (*references*), por outro diagrama, para que as raízes de *features* também possam ser classificadas quanto a seu tipo: obrigatórias ou opcionais.

A Figura 19(b) apresenta a configuração aplicada ao diagrama de *features* gerado na Figura 19(a). A marcação em caixas do tipo *checkbox,* encontradas nos grupos de *features,* é utilizada para selecionar o tipo de produto a ser gerado na LP, em que é gerado a configuração especifica de um sistema concreto. Por exemplo, o produto deverá ter como configuração para a *feature* “TipoPagamento” somente as opções “Boleto” e “Deposito”.




**(a)** Modelo de *features* GPSOL **(b)** Configuração do modelo de *features*

**Figura 19:** Modelo de edição de *features* e o modelo de configuração das *features* para GPSOL

Na Figura 20 é vista a caixa de propriedades das *features,* também definidas por marcação de *checkbox* para a definição de atributos das *features.*

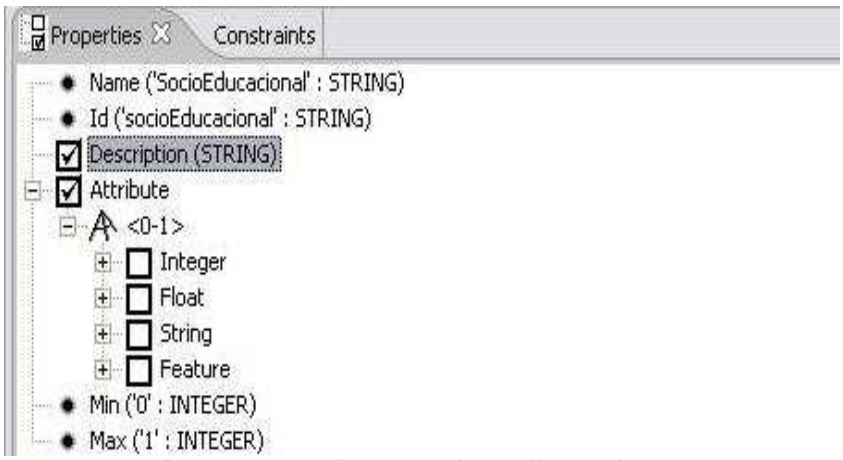



**Figura 20:** Configuração dos atributos das *features*

A ferramenta conta ainda com exportações XML das *features* ou do grupo de *features,* de uma forma totalmente automática.

Na figura 21 é visto um exemplo de exportação da *feature* “pagamento” e do subgrupo de *feature* “cartao”, “deposito” e “boleto” para arquivo XML.

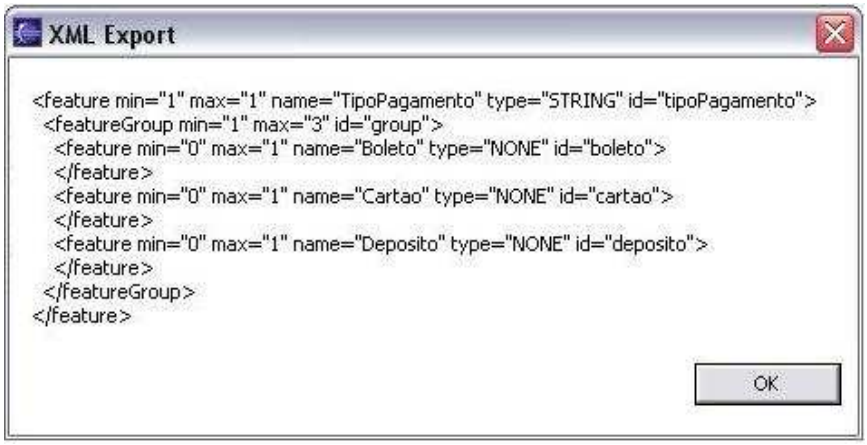



**Figura 21:** Modelo de exportação de *feature* no formato XML

A modelagem de *features* deu-se de forma muito intuitiva. Porém as configurações iniciais da ferramenta deram-se de forma um pouco mais complexa, sendo por muitas vezes necessário um conhecimento maior sobre a plataforma Eclipse onde o *plug-in* é instalado.

Para a criação e definição dos tipos de *features*, é necessário clicar com o botão direito do *mouse n*as *features* que se deseja alterar, excluir ou configurar, abrindo-se uma caixa com várias opções de configurações relacionadas à *feature*.

Um ponto fraco revelado inicialmente na ferramenta é a não geração de

um arquivo com extensão de figura (por exemplo, JPEG) para o modelo de *features* criado.

### 5.2.3 A modelagem das *features* na ferramenta Captain Feature

A modelagem do modelo de *features* da ferramenta Captain Feature é baseada na representação gráfica de árvore, podendo ser gerado a partir dessa representação a forma de árvore estruturada hierarquicamente.

Para a atividade de criação do modelo de *features,* o processo tomou como recursos disponíveis o diagrama de casos de uso GPSOL (gerado na seção 4.2 e representado na Figura 13), e a notação de Czarnecki e Eisenecker (2000).

A Figura 22 apresenta o modelo de *features* seguindo a notação clássica da ferramenta, e a Figura 23 apresenta o mesmo modelo seguindo a notação moderna baseada na visualização das cardinalidades das *features* e do grupo de *features*. Os elementos do diagrama são compostos pelas *features* opcionais representados pelo símbolo  e pela cardinalidade [0..1], indicando que a *feature* pode ou não estar presente na LP. O símbolo  indica a *feature* do tipo obrigatória representada pela cardinalidade [1..1], indicando que a *feature* deve estar obrigatoriamente presente na LP. O grupo de *feature* alternativo inclusivo é representado pelo símbolo  com cardinalidade [1-k], onde k indica o tamanho do grupo. O símbolo  com cardinalidade [1-1] indica a representação dos grupos de *features* do tipo alternativo exclusivo.

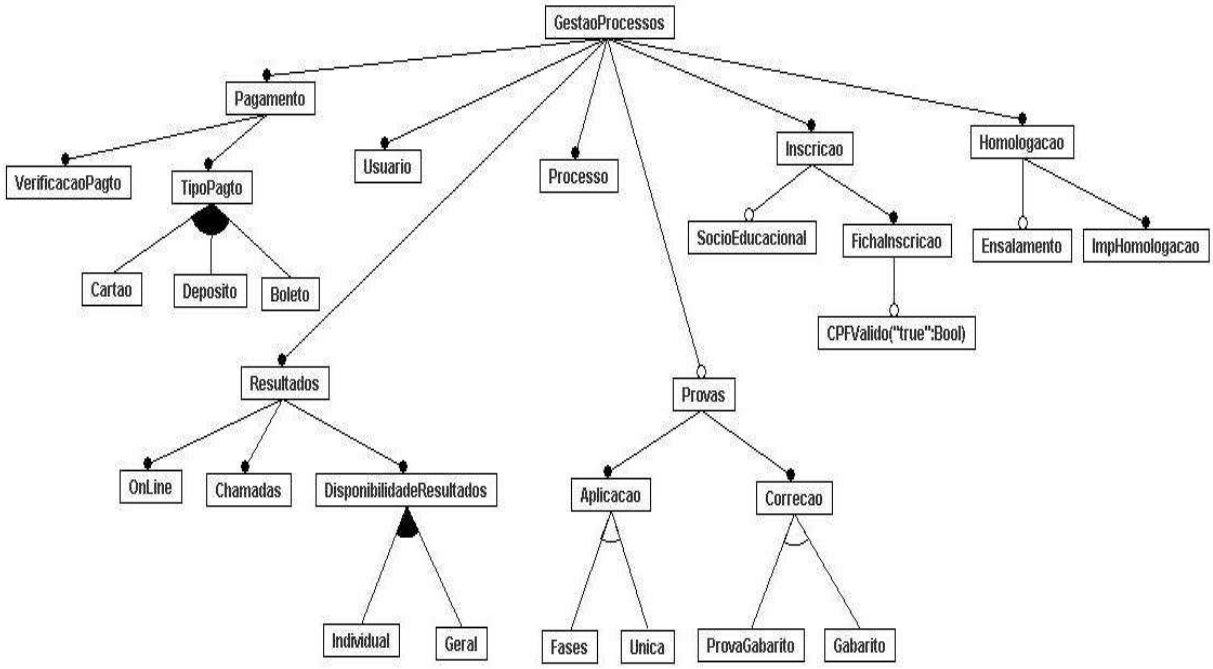



**Figura 22:** Modelo de *features* baseada na notação clássica da ferramenta

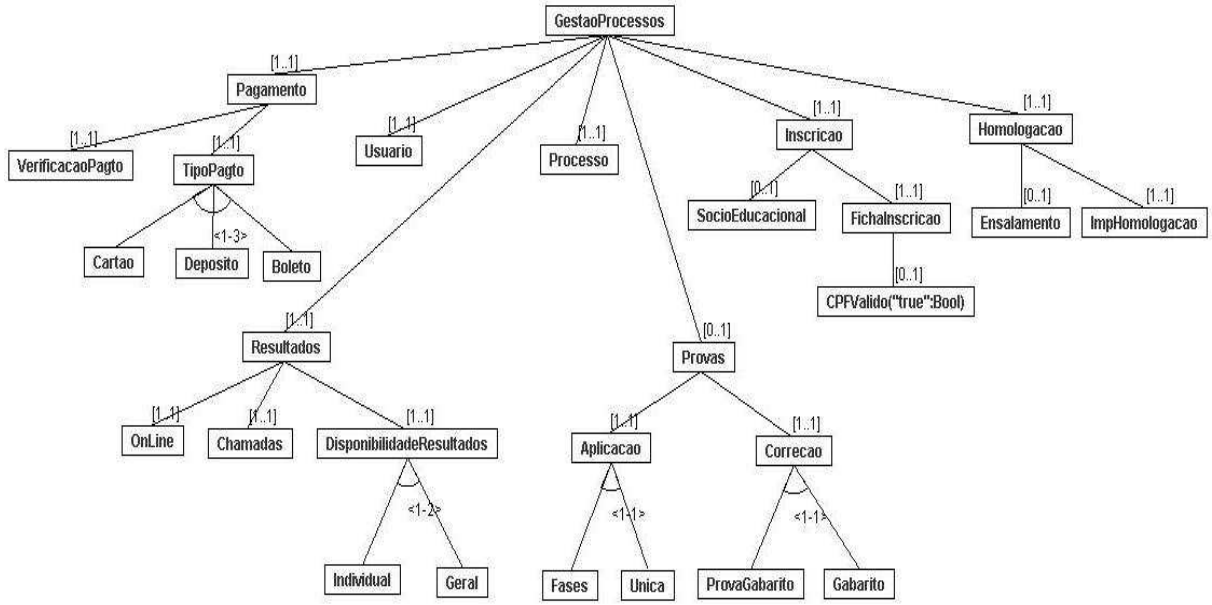



**Figura 23:** Modelo de *features* baseada na notação moderna da ferramenta

A ferramenta tem como opção a especialização do processo, após a geração do diagrama de *features*, em que se tem a estrutura hierárquica da árvore representada na Figura 24. A especialização permite analisar as configurações das *features* de um modo mais estruturado.


**Figura 24:** Especialização do diagrama de *features*

Apesar da *interface* simples da ferramenta, foi encontrada grande dificuldade na criação dos grupos de *features* alternativos inclusivos (*or-group*) e alternativos exclusivos (xor-group). Por exemplo, para a representação da *feature* “aplicação” contendo as sub*features* com relação alternativo exclusivos “fases” e

“única”, é necessário:

1. conectar a *feature* “fases” e *feature* “unica” na *feature* “aplicação”. Para isso deve-se clicar com o botão direito do *mouse* na área de trabalho e selecionar a opção *conect feature* para ligar as *features;*
2. pressionar o botão CTRL do teclado, e clicar sobre as *features* “fases” e ”única”;
3. clicar com o botão direito na área de trabalho, selecionando a opção *group features;*

Por padrão é gerada a relação do tipo alternativa exclusiva, podendo ser modificado clicando-se em uma das sub*features* como o botão direito, selecionando a opção *change group* e a alternativa *or-group.*

### 5.2.4 A Comparação e a Avaliação das Ferramentas

Neste trabalho foram utilizadas três ferramentas de apoio ao gerenciamento de variabilidades em LP. Para a comparação das ferramentas foi utilizado o estudo de caso do domínio para GPSOL, na qual foram realizadas as atividades do processo de gerenciamento, gerando a modelagem de *features,* artefato característico da abordagem de LP.

O Quadro 5 apresenta as características técnicas das ferramentas analisadas.

| Ferramentas | Versão | Linguagem | Extensão | Licença | Disponibilidade |
|---|---|---|---|---|---|
| **Jude** | 5.0.2 | Java | JUDE | *Free* | Internet/CD Instalação |
| **Captain** | 1.0 | Java | XML | *Free* | Internet |
| **FeaturePlugin** | 0.0.6 | Java | FMP | *Free* | Internet |

**Quadro 5:** Características técnicas das ferramentas de gerenciamento

Nota-se que as ferramentas têm como características comuns a linguagem Java utilizada na sua criação e a licença *free*, indicando a predominância do software livre.

Para a etapa de comparação das ferramentas envolvendo o processo de gerenciamento de variabilidades, quanto a seus pontos positivos e negativos, é

necessário analisar e comparar as principais características e funcionalidades encontradas no processo de gerenciamento (Quadro 6).

| | FERRAMENTAS | | |
|---|---|---|---|
| | **Jude Community** | **Captain Feature** | **FeaturePlugin** |
| **FUNCIONALIDADES/CARACTERISTICAS** | | | |
| **Exportação** | HTML/JPEG/JAVA | JPEG, XML* | XML |
| **Importação** | JAVA | Não Possui | Não Possui |
| **Notação** | UML | Czarnecki e Eisenecker | Czarnecki e Eisenecker |
| **Modelo de *Feature*** | Árvore | Árvore/Estrutural | Estrutural |
| **Geração de Artefatos** | Casos de Uso, Diagamas de Classe, Compontes | Modelo de *feature*/especialização do modelo de *features* | Modelo de *features* |
| **Tipos de *features* (representação)** | Estereótipos da UML (ex. <<mandatory>>) | [1..1] ou símbolos próprios da notação (ex.  ) | Símbolos próprios da notação (ex.  ) |
| **Configuração das *features*** | Não possui | Possui | Possui |
| **Interface** | Modelagem | Modelagem/estrutura | Estrutura |
| **Delimitação dos pontos e variação** | Através de notas da UML | Através da notação própria da linguagem | Através da notação própria da linguagem |
| **Relações de dependência** | *Requires* e *mutex* | Não possui | Não possui |

**Quadro 6:** Comparação qualitativa das funcionalidades das ferramentas de gerenciamento

Para a análise comparativa das ferramentas os critérios adotados foram:

- **exportação:** os tipos de arquivos que a ferramenta permite gerar além da própria extensão;
- **importação:** tipos de arquivos aceitos como entrada diferentes da própria extensão;
- **notação:** características da modelagem que a ferramenta desenvolve;
- **modelo de *features:*** forma como a modelagem é representada, podendo ser em forma de diagramas ou estruturas;
- **geração de artefatos:** artefatos da LP que podem ser aplicados o processo de gerenciamento de variabilidades;
- **tipos de *features* (representação)**: forma como é mostrado a identificação dos tipos de *features*;

* Somente na geração de especialização do modelo de *features.*

- **configuração das *features*:** usado para selecionar o tipo de produto a ser gerado na LP, ou seja, é gerado a configuração especifica de um sistema concreto;
- **interface:** formas como os artefatos da LP são visualizados;
- **delimitação dos pontos de variação:** indica se a ferramenta suporta gerenciamento envolvendo cardinalidades, e a forma como são representadas;
- **relações de dependência:** indica a relação entre os grupos de *features*.

Com base na comparação entre as características funcionais do processo de gerenciamento envolvendo as ferramentas, representados no Quadro 6, outras serão adicionadas a fim de enriquecer o estudo qualitativo. O Quadro 7 apresenta os principais diferenciais, ponto fortes e pontos fracos encontrados na análise das ferramentas.

| Ferramentas | Diferenciais | Pontos Fortes | Pontos Fracos |
|---|---|---|---|
| **Jude Community** | • permite gerenciar artefatos da LP que não fazem parte da abordagem (casos de uso, diagrama de classes, componentes);<br>• exportação JAVA | • usabilidade intuitiva;<br>• fácil instalação e configuração;<br>• exporta a modelagem para o tipo figura (JPEG);<br>• permite gerenciar variabilidades em outros artefatos da UML. | • documentação disponível apenas *on-line;*<br>• a modelagem das *features* não se dá de maneira totalmente satisfatória conforme citado na seção 5.2.1 |
| **Captain Feature** | • ferramenta com tamanho muito pequeno (aproximadamente 2,24 Mb e que compactada pode chegar a 1,2 Mb, cabendo num disquete 1,44Mb) | • fácil instalação e configuração;<br>• possui modelo de *features* na base de dados como exemplo. | • poucas funcionalidades configurativas;<br>• documentação com pouco conteúdo explicativo;<br>• problemas com relacionamentos entre *features*, citado na seção 5.2.3 |

**Quadro 7:** Comparação qualitativa dos diferenciais, pontos fortes e fracos das ferramentas.

| Ferramentas | Diferenciais | Pontos Fortes | Pontos Fracos |
|---|---|---|---|
| **FeaturePlugin** | • disponível na forma de *plugin* do Eclipse;<br>• permite configurar os produtos da LP. | • boa usabilidade;<br>• boa visualização da estrutura em árvores do modelo de *features*;<br>• bom ambiente de configuração e edição de *features*. | • difícil de instalar e configurar;<br>• a versão só é aceita pelo Eclipse 3.0.2;<br>• pouca documentação disponível. |

**Quadro 7:** continuação da comparação qualitativa dos diferenciais, pontos fortes e fracos das ferramentas.

Através da real utilização das ferramentas pode-se realizar o estudo qualitativo envolvendo suas características funcionais em realização ao processo de gerenciamento de variabilidades.

As ferramentas são dotadas de características próprias para esse processo, envolvendo a modelagem de *features,* porém todas levam ao mesmo objetivo: gerenciar a identificação e o relacionamento do conjunto de características envolvidas em um domínio da LP.

A ferramenta FeaturePlugin possui como característica a configuração das *features*, mostrando-se bastante eficiente no processo de modelagem, por sua estrutura hierárquica de fácil compreensão. No entanto para uma melhor utilização das suas funções seria necessário uma melhor documentação disponível.

A ferramenta Capatin Feature apesar de possuir uma interface muito básica se comparado com as demais ferramentas, possui uma boa representação quanto à modelagem de *features.* A função de especialização da ferramenta gera uma estrutura muito parecida com a modelagem do FeaturePlugin, devido ao fato da notação também ser baseada em Czarnecki e Eisenecker (2000).

As ferramentas Capatain Feature e FeaturePlugin permitem apenas a modelagem e configuração das *features,* não estendendo o gerenciamento para outras etapas do processo. O que não acontece com a ferramenta Jude Community, que na realização do processo de gerenciamento, foi a que se mostrou mais complexa e útil na sua utilização, pois apesar de não ser uma ferramenta característica da abordagem de LP e sim da modelagem de dados da UML, ela permite o gerenciamento das *features* nos demais artefatos envolvendo a notação UML (casos de uso, diagrama de classes e componentes). A representação do modelo de *features* fornece uma melhor visão das *features* e dos relacionamentos

entre ela, para a criação do modelo de rastreamento de variabilidades. Devido à utilização de “notas” da UML, a delimitação das variabilidades, além de basear-se nas cardinalidades da *feature,* gerencia também o tempo de resolução, de compilação e permite ainda identificar se o ponto de variação permite ou não a adição de mais variantes. Também, através da notação UML, é possível criar vínculos de dependência (*requires* e *mutex*) entre os grupos de *features* envolvidas no modelo.

No entanto, observa-se também, a possibilidade do uso dessas ferramentas em conjunto para melhor satisfazer processos de gerenciamento em LP. Por exemplo, o uso da ferramenta Jude Community para a representação gráfica das *features* e da ferramenta FeaturePlugin para a configuração dos produtos que farão parte de determinada LP.

# 6 CONCLUSÃO E TRABALHOS FUTUROS

LP é vista como uma área muito promissora, pois oferece uma maneira sistemática, prática e planejada de reutilização de software. Muito desse sucesso deve-se ao fato da indústria enxergar no enfoque de LP uma maneira de resolver problemas como *time-to-market*, redução de custo e tempo de produção, aumentando assim a sua produtividade.

O gerenciamento de variabilidades é uma das atividades mais importantes da LP, pois através desse processo é possível a identificação dos pontos nos quais um produto se diferencia em uma LP. No entanto, nota-se a complexidade do processo, comprovando-se a importância do envolvimento de alguma ferramenta que auxilie no gerenciamento de variabilidades.

Para a realização desse trabalho, analisou-se qualitativamente o uso de três ferramentas de apoio ao gerenciamento, sendo elas: Jude Community, Captain Feature e FeaturePlugin. Para o uso real e para a realização do estudo qualitativo das ferramentas no processo de gerenciamento, foi utilizado o domínio GPSOL da LP.

Através do estudo de caso qualitativo envolvendo as ferramentas conclui-se que o gerenciamento não seria possível sem a utilização de uma ferramenta de apoio, pois a notação utilizada para a geração de artefatos característicos da abordagem de LP como a modelagem de *features,* devem ser representadas de forma que seja possível analisar pontos de variação e variantes, tal como identificar e delimitar as *features.*

Conclui-se também, que apesar do estudo comparativo das ferramentas realizado nesse trabalho, apontar a Jude Community, como a mais completa e útil para as atividades envolvendo o gerenciamento de variabilidades, percebe-se que ela não satisfaz totalmente o processo, necessitando por vezes, o auxílio de outras ferramentas automatizadas. Isso se deve ao fato de que, independentemente as ferramentas conhecidas não auxiliam totalmente todas as fases do processo de gerenciamento, sendo às vezes necessário fazer uso conjunto com outras.

Como trabalhos futuros pode-se citar:

- a criação de uma ferramenta automatizada de apoio ao gerenciamento de variabilidades que satisfizesse totalmente a complexidade do processo, e que permitiria por exemplo, a análise de configurações de produtos de uma LP com base na remoção e adição de *features* e a análise dos artefatos afetados com as ações;

- um estudo quantitativo das ferramentas de gerenciamento, afim de analisar por meio de técnicas de ponderação a real eficácia das ferramentas nas atividades;

- um estudo qualitativo das ferramentas no processo de gerenciamento de variabilidades para LP orientada a aspectos, por ser uma linha de pesquisa em constante evolução.

## REFERÊNCIAS BIBLIOGRÁFICAS


AUDY, Jorge. Método de Pesquisa em Sistemas de Informação. In MINICURSO I – MÓDULO 2, 2002, Rio Grande do Sul. **Proceedings...** Rio Grande do Sul: PUCRS, 2002. p. 20.

AMARAL, E. A. G. G. **Empacotamento de experimentos em engenharia de softwar**e. 2003. 158 p. Dissertação (Mestrado em Ciências em Engenharia de Sistemas e Computação) – Programa de Engenharia de Sistemas e Computação, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2003.

ANTKIEWICZ, M.; CZARNECKI, K. FeaturePlugin: Feature modeling plug-in for Eclipse, *OOPSLA'04 Eclipse Technology eXchange (ETX) Workshop,* 2004. Paper available from <http://www.swen.uwaterloo.ca/~kczarnec/etx04.pdf>. Software available from gp.uwaterloo.ca/fmp – Disponível em: <http://www.eclipse.org> - Acessado em: outubro de 2007.

ARAGÓN, Cesar Rosas. **Processo de desenvolvimento de uma linha de produtos para sistemas de gestão de bibliotecas**. 2004. Dissertação (Mestrado) – Universidade Federal do Mato Grosso do Sul (UFMS), Campo Grande, 2004.

BASILI, V. R.; SELBY, R. W.; HUTCHENS, D. H. Experimentation in software engineering. **IEEE Transactions on Software Engineering**, Piscataway, v. 12, n. 7, p. 733-743, 1986.

BEDNASCH, T.; ENDLER, C.; LANG, M. CaptainFeature. Tool available on SourceForge at <https://sourceforge.net/projects/captainfeature/> (2002-2004) – Acessado em: Outubro, 2007.

BEUCHE, D.; PAPAJEWSKI, H.; SCHRÖDER-PREIKSCHAT, W. Variability management with feature models. In: SOFTWARE VARIABILITY MANAGEMENT WORKSHOP, 2003, Portland. **Proceedings…** Portland, 2003. p. 72-83.

BECKER, M. Towards a general model of variability in product families. In: SOFTWARE VARIABILITY MANAGEMENT WORKSHOP, 2003, Portland. **Proceedings…** Portland, 2003. p. 19-27.

BOSCH, J. **Design & use of software architectures**: adopting and evolving a product-line approach. Boston: Addison Wesley, 2000.

CASTRO, Cristiane Y. H.. **Uma abordagem para o desenvolvimento de linha de produto orientada a aspecto.** 2007. Proposta de Mestrado – Universidade Estadual de Maringá (UEM), Maringá, 2007.

CIRILO, Elder; KULESZA, Uirá; LUCENA, C. J. P.. GenArch: Uma Ferramenta baseada em Modelos para Derivação de Produtos de Software. In: Anais da Sessão de Ferramentas do SBCARS2007. **Anais**... Rio de Janeiro, 2007.

CHASTEK, G. Object technology and product lines. In: ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications (Addendum). Atlanta, Georgia, USA: ACM Press, 1997.p. 105-109.

CLEMENTS, P.; NORTHROP, L. **Software product lines: practices and patterns**. 1. ed. Boston: Addison-Wesley, 2001.

COHEN, Sholom. Product Line State of the Practice Report. Disponível em <http://www.sei.cmu.edu/publications/documents/02.reports/02tn017.html>, 2002 – Acessado em: novembro de 2007.

CONRADI, R.; BASILI, V.; CARVER, J.; SHULL, F.; TRAVASSOS, G.. “A Pragmatic Document Standards for a Experience Library: Roles, Documents, Contents e Structure.”, Technical Report CS-TR-4235, University of Maryland, Abril 2001.

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

_____; HELSEN, S.; EISENECKER, U.: Staged configuration through specialization and multi-level configuration of feature models. Software Process Improvement and Practice 10, 2005.

ECLIPSE PROJECT – Disponível em: <http://www.eclipse.org> - Acessado em: Outubro de 2007.

FRITSCH, C.; LEHN, A.; STROHM, T. Evaluating variability implementation mechanisms. In: INTERNATIONAL WORKSHOP ON PRODUCT LINE ENGINEERING, 2., 2002, Seattle. *Proceedings*… Seattle, 2002. p. 59-64.

GIMENES, I.M.S. Técnicas de Reutilização de Software. In MINICURSO I – MÓDULO II, 2006, Maringá. **Proceedings...** Maringá: UEM, 2006. p. 43-44.

_____; LAZILHA, F. R.; PRICE, R. T. Uma proposta de arquitetura de linha de produtos para workflow management systems. In: XVI SIMPÓSIO BRASILEIRO DE ENGENHARIA DE SOFTWARE, 2002, Gramado. **Anais...** Gramado, 2002.

_____; TRAVASSOS, G. H. O enfoque de linha de produto para desenvolvimento de software. In: JORNADA DE ATUALIZAÇÃO EM INFORMÁTICA DA SBC, 22., 2002, Florianópolis. **Anais...** Florianópolis, 2002. p. 34

GRISS, M., “Implementing Product-Line *Features* with Component Reuse”, 5th International Conference on Software Reuse, ACM/IEEE, Vienna, Austria, Junho 2000.

_____; FAVARO, J.; D´ALESSANDRO, M. Integrating feature modeling with the RSEB. In: INTERNATIONAL CONFERENCE ON SOFTWARE REUSE, 5., 1998, Washington. **Proceedings...** Washigton, 1998.p.76-85.

GOMAA, H. **Designing software product lines with UML**: from use cases to pattern-based software architectures. Boston: Addison-Wesley, 2005.

_____; WEBBER, D. Modeling adaptive and evolvable software product lines using the variation point model. In: HAWAII INTERNATIONAL CONFERENCE ON SYSTEM SCIENCES, 37., 2004, Hawaii. **Proceedings…** Hawaii, 2004. p. 01-10.

HEYMANS, P.; TRIGAUX, J. C. **Software product line:** state of the art. Technical report for PLENTY project, Institut d’Informatique FUNDP, Namur, 2003.

HALMANS, G.; POHL, K. Communicating the variability of a software-product family to customers. **Journal on Software and Systems Modeling**, New York, v. 2, n. 1, p. 15-36, mar. 2003.

JUDE UML MODELING TOOL. Disponível em http://jude.change-vision.com/jude-web/product/community.html – Acessado em: Outubro de 2007.

KANG, K. **Feature-oriented domain analysis (FODA) - feasibility study**. Technical Report CMU/SEI-90-TR-21, SEI/CMU, Pittsburgh, 1990.

KITCHENHAM, B. DESMET: a method for evaluating software engineering methods and tools. Technical Report TR96-09, Keele, United Kingdom, 1996. 49 p.

_____; PICKARD, L.; PFLEEGER, S. L. Case studies for method and tool evaluation. **IEEE Software,** v.11, p. 52-62, 1995.

NORTHROP, L. M. SEI's Software product line tenets. **IEEE Software**, v. 19, n. 14, 2002.

OLIVEIRA JUNIOR, E. A. de. **Um processo de gerenciamento de variabilidade para linha de produto de software.** 2005, 155 f. Dissertação (Mestrado) – Universidade Estadual de Maringá (UEM), Maringá, 2005.

PFLEEGER, S. L. Experimental design and analysis in software engineering – how to set up na experiment. **ACM SIGSOFT – software Engineering Notes.** v. 20, n. 1, p. 22-26, 1995.

SEI - SOFTWARE ENGINEERING INSTITUTE. **Capability Maturity Model for Software (SW-CMM)**. out. 2004. Disponível em: <http://www.sei.cmu.edu/PLP>

SELBIG, Mario. AmiEddi. Tool available at <http://www.generative-programming.org>, 2002-2004 – Acessado em: novembro de 2007.

_____. A feature diagram editor — analysis, design, and implementation of its core functionality. Diplomarbeit, Fachbereich Informatik, Fachhochschule Kaiserslautern, Standort Zweibr¨ucken, Germany, Out. 2000.

SOURCEFORGE. Captain Feature. Disponível em <https://sourceforge.net/projects/captainfeature/> - Acessado em: Outubro de 2007.

SUCCI, G.; YIP, J.; PEDRYCZ, W. Holmes: an intelligent system to support software product line Development. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, 23., 2001, Toronto. **Proceedings…** Toronto, 2001, p.829-832.

TRAVASSOS, G. H.; GUROV, D.; AMARAL, E. A. G. G. Introdução à engenharia de software experimental. Relatório Técnico RT-ES-590/02. Programa de Engenharia de Sistemas e Computação, COPPE/UFRJ, 2002. 52 p.

VAN GURP, J.; BOSCH, J. On the notion of variability in software product lines. In: THE WORKING IEEE/IFIP CONFERENCE ON SOFTWARE ARCHITECTURE, 2001, Amsterdam. **Proceedings...** Amsterdam, 2001.

VAN DER HOEK, A. Capturing product line architectures. In: INTERNATIONAL SOFTWARE ARCHITECTURE WORKSHOP, 4., 2000, Limerick. **Proceedings…** Limerick, 2000. p. 95-99.

WEISS, D.; CHI TAU, R. L. **Software product-line engineering:** a family-based software development process. Boston: Addison Wesley, 1999.