FORMALIZAÇÃO E VERIFICAÇÃO DE CONSISTÊNCIA NA REPRESENTAÇÃO DE VARIABILIDADES

Regiane Felipe de Oliveira

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:

_____

Profa. Cláudia Maria Lima Werner, D.Sc.

_____

Profa. Marta Lima de Queirós Mattoso, D.Sc.

_____

Profa. Rosana Teresinha Vaccare Braga, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
FEVEREIRO DE 2006

OLIVEIRA, REGIANE FELIPE

Formalização e Verificação de Consistência na Representação de Variabilidades [Rio de Janeiro] 2006

XIII, 133 p., 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia de Sistemas e Computação, 2006)

Dissertação – Universidade Federal do Rio de Janeiro, COPPE

1. Variabilidade
2. Verificação de Consistência
3. Modelo de Características
4. Reutilização de Software

I. COPPE/UFRJ II. Título (série)

*Aos meus pais, Rosária e Raimundo.*

*Ao meu irmão André.*

# AGRADECIMENTOS

Aos meus pais, Rosária e Raimundo, pelo apoio incondicional em todas as horas. Por terem, mesmo contra a vontade, aceitado minha mudança para o Rio de Janeiro, e meus longos períodos de ausência. Sem eles, este trabalho não teria sido concluído.

Ao meu irmão André, pela confiança e admiração.

À Marluce Pereira, por ter me acolhido nesta cidade, pelas muitas horas de companheirismo e experiências trocadas.

À professora Cláudia Werner, não só pela orientação, mas por ter me encorajado desde o começo a não desistir, mesmo quando eu achava que não tinha o suficiente para desenvolver essa dissertação.

Meus mais sinceros agradecimentos aos vários amigos e companheiros que adquiri nesta longa jornada que foi o mestrado. Às companheiras de disciplinas Gladys Lima e Lucia Nigro, por terem me ajudado em um período tão difícil como o período de adaptação. Aos amigos do LENS: Leonardo Murta, Natanael Maia e Artur Barbalho, cujo conhecimento e ajuda foram imprescindíveis no desenvolvimento desse trabalho; Hamilton Oliveira, Cristine Dantas, Alexandre Dantas, Rafael Cepêda, Marco Lopes, Luiz Gustavo Lopes, Carlos Melo Jr.(Beto) e Marco Mangan, pelos diversos momentos de descontração e experiências compartilhadas; à Isabella Almeida, pela amizade, pelas inúmeras caronas para casa e pela valiosa consultoria em questões de "arte gráfica".

Um agradecimento especial à Ana Paula Blois e Aline Vasconcelos, que, por diversas vezes, deixaram suas pesquisas de doutorado para se dedicarem à minha pesquisa. Pela co-orientação, ainda que extra-oficial, me guiando nas diversas "encruzilhadas" do caminho, e principalmente pela amizade e presença de vocês, um simples obrigado não seria suficiente.

À amiga Ana Paula Couto, pelo incentivo constante, mesmo a milhares de quilômetros de distância.

Às secretárias Taísa (ES), Solange e Claudia Prata (PESC) e Carol (LAND), por se mostrarem sempre prestativas.

Às professoras Marta Mattoso e Rosana Braga, por terem aceitado fazer parte desta banca.

À CAPES, pelo apoio financeiro.

Ao Luis Oscar, simplesmente por tudo. Por ter agüentado todas as minhas crises, por ter sido a companhia mais constante, pelas inúmeras manifestações de amor e carinho, e até mesmo pelas diversas discussões que, com certeza, ajudaram a me tornar uma pessoa melhor.

E por último, mas não menos importante, a Deus, sem o qual nada seria possível.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

# FORMALIZAÇÃO E VERIFICAÇÃO DE CONSISTÊNCIA NA REPRESENTAÇÃO DE VARIABILIDADES

Regiane Felipe de Oliveira

Fevereiro / 2006

Orientadora: Cláudia Maria Lima Werner

Programa: Engenharia de Sistemas e Computação

Em abordagens de reutilização de software, como Engenharia de Domínio e Linha de Produtos, as semelhanças e diferenças existentes entre os sistemas de uma família devem ser identificadas e documentadas, constituindo uma atividade denominada modelagem de variabilidades. Para tanto, conceitos inerentes à modelagem de variabilidade devem ser formalizados e representados de maneira coerente em todos os artefatos envolvidos. Caso contrário, falhas na modelagem podem ocorrer, reduzindo o potencial de reutilização.

Através da formalização dos conceitos de variabilidade por meio de um metamodelo, que dá origem a uma notação mais abrangente para a representação de variabilidade em modelos de características (*features*), tais falhas de modelagem são minimizadas. O metamodelo proposto viabiliza o desenvolvimento de um sistema de críticas, que verifica a consistência do modelo de características. Além disso, heurísticas são estabelecidas para a propagação de tais variabilidades para o diagrama de classes da UML (*Unified Modeling Language*). Essa propagação se dá por meio de mapeamento entre o metamodelo de características proposto e o metamodelo da UML.

O trabalho é desenvolvido no contexto do ambiente Odyssey, que visa apoiar a reutilização de software por meio de abordagens complementares, como Engenharia de Domínio e Linha de Produtos e Desenvolvimento Baseado em Componentes.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

FORMALIZATION AND CONSISTENCY CHECKING IN VARIABILITIES MODELING

Regiane Felipe de Oliveira

February / 2006

Advisor: Cláudia Maria Lima Werner

Department: Computer and Systems Engineering

In software reuse approaches, such as Domain Engineering or Software Product Lines, commonality and variability of a system family should be identified and documented through an activity called Variability Modeling. To achieve this goal, it is necessary to formalize and represent variability concepts in a consistent manner within several software models. Otherwise, there might be modeling failures, decreasing the software reuse potential.

This work proposes the decreasing of these modeling faults by formalizing variability concepts using a meta-model, which originates a notation for variability modeling in feature models. The proposed meta-model allows the development of a criticism system, whose purpose is a consistency checking in feature models. In addition, heuristics to represent variability in class models are established. These heuristics are defined based on relations between the feature meta-model and Unified Modeling Language (UML) class meta-model.

This work is developed within Odyssey SDE, a software development environment whose aim is to construct a reuse infrastructure based on domain models, product lines and component based development.

# ÍNDICE

# ÍNDICE DE FIGURAS

# ÍNDICE DE TABELAS

# CAPÍTULO I
# INTRODUÇÃO

## 1.1 – MOTIVAÇÃO

A reutilização, i. e., o processo de criação de software a partir de um software existente, ao invés de construí-lo a partir do zero (KRUEGER, 2002) é uma atividade já realizada na comunidade de desenvolvimento de software. Embora seja feita de uma maneira *ad hoc*, idéias, objetos, argumentos e abstrações sempre foram reutilizados por desenvolvedores de software (PRESSMAN, 1997). No entanto, para que os benefícios esperados, tais como o aumento de produtividade e qualidade e redução de custo e tempo de desenvolvimento do software, sejam alcançados, é necessário que a reutilização seja feita de maneira sistemática. Tal necessidade dá lugar às técnicas de reutilização, tais como Engenharia de Domínio (ED), *Frameworks*, Linha de Produtos de Software (LP) e Desenvolvimento Baseado em Componentes (DBC).

As técnicas de ED e LP se assemelham, particularmente, no desenvolvimento de uma família de sistemas como abordagem de reutilização. Embora cada técnica tenha sua forma de tratar o desenvolvimento dessa família de sistemas, ambas incorporam uma atividade de análise dos aspectos comuns e diferentes entre os sistemas. A análise de tais aspectos dá origem ao conceito de variabilidade.

A variabilidade em uma família de sistemas de software se mostra importante no sentido de deixar explícitos os pontos onde tais sistemas se assemelham, e, portanto podem ser reutilizados, e os pontos onde eles diferem entre si, devendo receber tratamento específico. Para tanto, a variabilidade e conceitos relacionados devem ser documentados durante todo o ciclo de vida do software, nos artefatos inerentes a cada fase. Além disso, deve haver uma coerência na representação da variabilidade nos diferentes artefatos que representam o mesmo requisito do sistema.

Uma análise da literatura apontou trabalhos que se propunham a direcionar a representação da variabilidade por meio de notações gráficas (KANG *et al.*, 1990), (KANG *et al.*, 2002), (RIEBISCH *et al.*, 2002), (MILER, 2000), (BOSCH, 2004), (CECHTICKY *et al.*, 2004), (CZARNECKI *et al.*, 2004). Foram considerados nesta análise os trabalhos que se valiam do artefato de mais alto nível de abstração para a

representação de variabilidades, denominado modelo de características. O modelo de características (*features*) representa as características de uma família de sistemas em um domínio, suas semelhanças e diferenças e as relações entre elas (KANG *et al.*, 1990). No entanto, algumas deficiências foram encontradas nas notações analisadas:

- os conceitos inerentes à variabilidade de uma família de sistemas não eram representados de maneira completa nas notações;
- as notações apresentavam uma semântica variada, que nem sempre era formalizada, levando a interpretações ambíguas;
- a variabilidade era representada em apenas um artefato, sem a preocupação com a sua propagação para outros artefatos relacionados.

Tais deficiências podem ocasionar uma representação inadequada da variabilidade, resultando em uma modelagem incorreta da família de sistemas que será posteriormente reutilizada. Assim, os benefícios almejados com a reutilização de software podem não ser alcançados.

## 1.2 – OBJETIVOS

Tendo em vista os problemas citados anteriormente, essa dissertação tem por objetivo prover uma representação adequada de variabilidades em artefatos reutilizáveis, inerentes a uma família de sistemas, que envolve: a) um metamodelo que formaliza a semântica dos conceitos inerentes à variabilidade, b) uma notação para a representação de variabilidades em modelos de características. e. c) o estabelecimento de heurísticas de mapeamento entre os elementos do modelo de características e do modelo de classes, a fim de nortear a representação coerente das variabilidades entre esses dois artefatos.

O metamodelo direciona a utilização da notação na construção de modelos de características e, por meio de um conjunto de regras de boa formação para ele definidas, possibilita a verificação de consistência nos modelos construídos. Nesse sentido, é proposto um mecanismo de verificação de consistência dos modelos que representam uma família de sistemas, mais especificamente em modelos de características e de classes.

A notação proposta é originada do metamodelo, e é mais completa em relação às notações analisadas na literatura. Além disso, é utilizada no contexto do ambiente Odyssey (ODYSSEY, 2005), que visa prover uma infra-estrutura de suporte à

reutilização baseada em modelos de domínio. Dessa maneira, tem-se a intenção de oferecer um apoio automatizado para a construção de modelos de características, que facilite a representação das variabilidades. Além disso, com a utilização do ambiente Odyssey, a verificação de consistência do modelo de características é também facilitada, por meio de um sistema de críticas para modelos, denominado Oráculo (DANTAS *et al.*, 2001).

Finalmente, as heurísticas de mapeamento entre elementos do modelo de características e o modelo de classes são obtidas com a definição do metamodelo de características, e seu mapeamento para o metamodelo da *Unified Modeling Language* (OMG, 2005), que norteia a construção do modelo de classes.

O objetivo desse trabalho foi atingido através da construção do metamodelo de características, que originou a notação Odyssey-FEX para a modelagem de variabilidades. Além disso, com base em tal metamodelo, foram implementadas regras de consistência para modelos de características, que foram utilizadas na adaptação de um sistema de críticas. A viabilidade de tais regras foram avaliadas por meio de um estudo de observação.

Assim, minimiza-se o problema de propagação da representação de variabilidades entre os modelos e possibilita-se a verificação de consistência intra e inter-modelos, diminuindo a incoerência entre modelos que representem uma mesma família de sistemas, sob diferentes perspectivas, e conseqüentemente, aumentando o potencial de reutilização.

## 1.3 - ORGANIZAÇÃO DA DISSERTAÇÃO

Este trabalho está organizado em seis capítulos. No **segundo** capítulo, são identificados os requisitos desejáveis para uma notação que represente a variabilidade em um modelo de características. Com base nesses requisitos, é feita uma análise comparativa das notações existentes na literatura com esse propósito.

No **terceiro** capítulo, são descritos a proposta da notação e seu metamodelo. São apresentadas ainda, as regras de boa formação do metamodelo.

A verificação de consistência inter e intra-modelos é discutida no **quarto** capítulo. Neste capítulo, é descrito o mapeamento entre os metamodelos de características e de classes, bem como estabelecidas as heurísticas de representação de variabilidades, resultantes desse mapeamento.

No **quinto** capítulo, é detalhada toda a implementação da proposta, no contexto do ambiente Odyssey. A fim de caracterizar a viabilidade das regras de boa formação do metamodelo, foi realizado um estudo de observação, cujos resultados são apresentados também neste capítulo.

No **sexto** e último capítulo, as contribuições e limitações deste trabalho são listadas, bem como apontados os trabalhos futuros.

# CAPÍTULO II
# MODELAGEM DE VARIABILIDADES EM ABORDAGENS DE REUTILIZAÇÃO DE SOFTWARE

## 2.1 - INTRODUÇÃO

É sabido que a reutilização de software é um objetivo perseguido há muito tempo na comunidade de Engenharia de Software. Os históricos encontrados na literatura apontam esforços nessa área desde os idos dos anos 60 e 70, com *McIlroy* e as primeiras noções de componentes. Desde então, muito tem sido feito para se promover a reutilização. Técnicas das mais diversas foram apresentadas, como, por exemplo, Engenharia de Domínio, *Frameworks*, Padrões, Desenvolvimento Baseado em Componentes e Linhas de Produtos de Software.

De acordo com Werner & Braga (WERNER & BRAGA, 2005), para que a reutilização de software seja efetiva, esta deve ser sistematicamente considerada em todas as fases do desenvolvimento, desde a análise até a implementação. Tal sistemática pode ser alcançada por meio de técnicas como, por exemplo, Engenharia de Domínio (ED), ou uma de suas vertentes, a Linha de Produtos de Software (LP).

Embora existam controvérsias na comunidade de Engenharia de Software acerca das semelhanças e diferenças entre ED e LP (WERNER & BRAGA, 2005) (POULIN, 1997), um consenso é que ambas as técnicas incorporam uma etapa de análise de domínio, cujo intuito, entre outras coisas, é explicitar as semelhanças e diferenças existentes entre sistemas de uma família, atividade esta que pode ser denominada modelagem de variabilidades.

A modelagem de variabilidades requer uma modelagem explícita de todos os conceitos inerentes, permitindo o entendimento entre todos os envolvidos no processo, tais como usuários, especialistas de domínio, desenvolvedores, etc (MASSEN & LICHTER, 2002). Além disso, é importante que a variabilidade seja representada em todas as fases do ciclo de vida de um software.

Tal modelagem de variabilidade é abordada na literatura por diversos trabalhos que orientam sua representação em diferentes artefatos do ciclo de vida de um software. A representação da variabilidade em modelos de caso de uso pode ser encontrada em (JOHN & MUTHIG, 2002), (GOMAA & SHIN, 2002 ) ou (BERTOLINO *et al.*, 2002).

Os trabalhos de Mathias Clauβ (CLAUSS, 2001a), (CLAUSS, 2001b) e de Morisio (MORISIO *et al.*, 2000) sugerem extensões da UML para a representação de variabilidades no modelo de classes. Além disso, o método KobrA (ATKINSON *et al.*, 2002) aborda a representação de variabilidades por meio de modelos de decisão[1]. Foram ainda pesquisados na literatura alguns trabalhos que abordam a representação de variabilidades no modelo de características. Tal modelo é considerado um modelo de mais alto nível, bem como o ponto de partida para a reutilização de artefatos em um processo de ED ou LP. Assim, o objetivo deste capítulo é apresentar uma análise comparativa de tais notações, levando em consideração alguns requisitos necessários à representação de variabilidades em um modelo de características.

O capítulo está organizado da seguinte maneira: a seção 2.2 descreve conceitos básicos de ED e LP. Na seção 2.3 são apresentados conceitos relacionados à variabilidade. Os requisitos para a modelagem de variabilidades são explorados na seção 2.4. Tais requisitos servem de base para uma análise das notações existentes, apresentada na seção 2.5. As considerações finais sobre o capítulo podem ser encontradas na seção 2.6.

## 2.2 - ENGENHARIA DE DOMÍNIO E LINHA DE PRODUTOS DE SOFTWARE

Muitos trabalhos encontrados na literatura tendem a comparar ED com LP, muitas vezes gerando controvérsias e questionamentos. O trabalho desenvolvido por Poulin (1997) sugere que Engenharia de Domínio e Linha de Produtos de Software são termos que se referem ao mesmo nível de abstração, direcionados, respectivamente, ao público acadêmico e industrial. Entrar no mérito de tal comparação, porém, não faz parte do escopo deste trabalho.

São apresentados, a seguir, alguns conceitos básicos de cada uma dessas técnicas, de modo a contextualizá-las na reutilização de software.

### 2.2.1 – O PROCESSO DE ENGENHARIA DE DOMÍNIO

O termo Domínio é utilizado para referenciar ou agrupar uma coleção de aplicações, existentes ou futuras, que compartilhem características comuns, i.e., uma

[1] Modelos textuais que consistem em uma hierarquia de decisões que relacionam as opções de configuração de um sistema com os sistemas específicos (ATKINSON *et al.*, 2002).

classe de sistemas que apresentam funcionalidades similares (WERNER & BRAGA, 2005) (SEI, 2005b).

Com o propósito de maximizar a reutilização, por meio da coleta de informações de maneira confiável e sistemática, é definida a Análise de Domínio (AD). Prieto-Diaz define AD como "o processo de identificar e organizar o conhecimento a respeito de uma classe de problemas, de maneira a apoiar a descrição e solução de tais problemas" (PRIETO-DIAZ & ARANGO, 1991). Assim, a AD tem como resultado uma taxonomia de sistemas que demonstrem as similaridades e diferenças entre os sistemas estudados, organizando os componentes e arquiteturas comuns a estes. Tais similaridades e diferenças são conhecidas por **variabilidades**.

Uma vez identificadas tais similaridades e diferenças, é necessário representá-las em um modelo genérico, denominado Modelo de Domínio, que posteriormente será refinado, originando artefatos que fazem parte de uma infra-estrutura de reutilização. Tais atividades fazem parte de um processo ao qual dá-se o nome de **Engenharia de Domínio** (ED) (ARANGO, 1994; WERNER & BRAGA, 2005).

Os artefatos gerados como produto da ED formam a base para que a reutilização ocorra. Assim, sistemas específicos podem ser gerados a partir de tais artefatos do domínio, instanciando-os de acordo com os requisitos da aplicação específica. Tal processo é denominado **Engenharia de Aplicação** (EA) (GRISS *et al.*, 1998; MILER, 2000). A ED e a EA trabalham paralelamente no intuito de prover a reutilização: a ED provê um conjunto de artefatos que serão utilizados, enquanto a EA constrói aplicações com base na instanciação dos artefatos gerados na ED, o que denota respectivamente, o desenvolvimento **para** reutilização e o desenvolvimento **com** reutilização. A ED e a EA são ilustradas na Figura 2.1.

Pode se dizer que existe um consenso da comunidade a respeito das etapas da ED (KANG *et al.*, 1990; GRISS *et al.*, 1998; SIMOS & ANTHONY, 1998; BRAGA, 2000), a saber:

- **Análise do domínio**: atividade onde são identificados o domínio e o seu escopo, e também são determinadas as características comuns e variáveis de uma família de aplicações.
- **Projeto do domínio**: nesta fase, os resultados da primeira etapa são utilizados para a construção de um projeto adaptável, fruto da identificação e generalização de soluções para as características comuns. As oportunidades de reutilização

identificadas na análise do domínio são refinadas de forma a especificar as restrições do projeto.

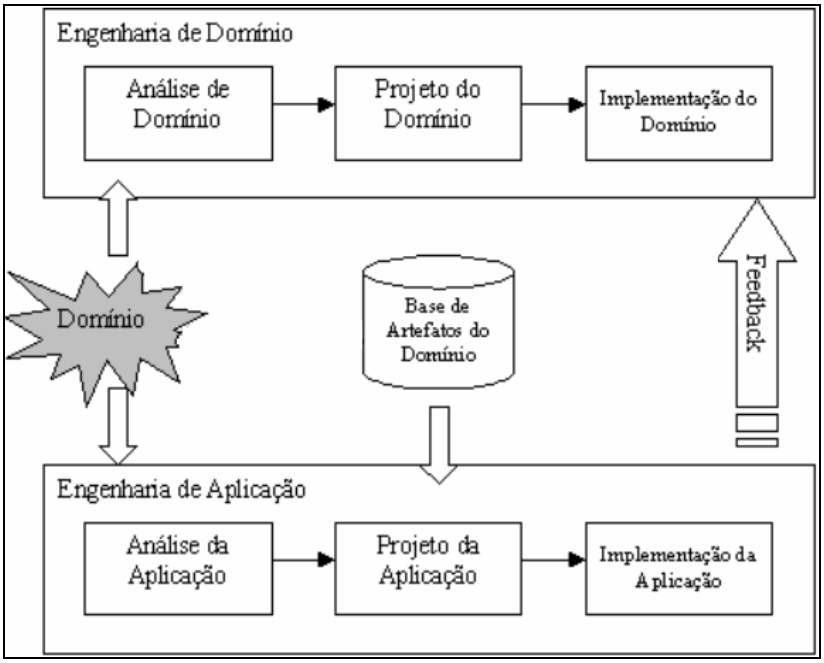



**Figura 2.1 - Ciclo de Vida da Engenharia de Domínio**
**(Adaptado de (ATKINSON *et al.*, 2002))**

- **Implementação do domínio**: nesta atividade, são definidos mecanismos para a tradução dos requisitos (resultados da análise e projeto do domínio) em modelos implementacionais, que incluem a identificação, reengenharia e/ou construção, e manutenção de componentes reutilizáveis que suportam estes requisitos e soluções de projeto.

Para cada uma dessas etapas, existem artefatos e profissionais relacionados, que contribuirão para o desenvolvimento **para** e **com** reutilização. Tal contribuição deve ocorrer por meio de artefatos específicos que ajudem a documentar o resultado das atividades de cada uma das fases de ED. Tais artefatos são modelos de análise, que devem capturar o contexto e o escopo do domínio; modelos de projeto, que devem especificar modelos arquiteturais que sirvam de guia para a instanciação da aplicação; e produtos da fase de implementação do domínio, como linguagens específicas do domínio, geradores de código e código de componentes (SEI, 2005a). Os profissionais relacionados são apresentados em (WERNER & BRAGA, 2005) como: **Fontes** (usuários finais e especialistas de domínio), **Produtores** (Engenheiros de Domínio) e

**Consumidores** (desenvolvedores de aplicações e interessados no entendimento do domínio).

A EA apresenta etapas análogas àquelas apresentadas na ED. No entanto, uma vez que a EA consiste no processo de reutilização de artefatos gerados na ED, as atividades relacionadas à modelagem das aplicações são mais específicas, enfatizando, sobretudo, as características particulares de cada aplicação, e não só as semelhanças entre as aplicações modeladas no domínio.

Durante a EA, é feita a seleção dos componentes necessários à aplicação em um alto nível de abstração, por meio do modelo do domínio, descendo gradualmente em níveis de abstração até conseguir atingir os componentes implementados ou semi-desenvolvidos do domínio (MILER, 2000). A essa seleção, denomina-se "recorte". O recorte na EA é fortemente influenciado pela variabilidade modelada no domínio, uma vez que as características variáveis em artefatos do domínio determinam o tipo de aplicação que será instanciada. Assim, o processo de EA envolve, primeiramente, uma avaliação do domínio, tendo em vista o desenvolvimento da aplicação desejada. Esta avaliação deve levar em conta as funcionalidades já disponibilizadas no domínio e a afinidade da família de sistemas com o produto ou aplicação que será construído. Em seguida, estão as tradicionais fases de análise, projeto e implementação de aspectos específicos da aplicação.

Diversos métodos de ED se apresentam na literatura, dentre eles encontram-se: o FODA (*Feature Oriented Domain Analysis*) (KANG *et al.*, 1990), o FORM (*Feature Oriented Reuse Method*) (KANG *et al.*, 2002), o RSEB (*Reuse-Driven Software Engineering Business*) (GRISS *et al.*, 1998), e o ODM (*Organization Domain Modeling*) (SIMOS & ANTHONY, 1998). Além desses, outro processo existente é o Odyssey-DE (BRAGA, 2000), cujo propósito é unir os aspectos de reutilização e entendimento do domínio providos pelos processos de ED existentes (e.g. FODA, RSEB) e o detalhamento do desenvolvimento de componentes, provido pelos processos de Desenvolvimento Baseado em Componentes (DBC), sobretudo na fase de análise. Um refinamento desse processo, denominado *CBD-Arch-DE* (BLOIS *et al.*, 2004) oferece o apoio necessário à fase de projeto, para que os benefícios da união de ED e DBC sejam alcançados. Todos esses métodos, embora apresentem peculiaridades no tratamento das etapas do processo de ED e de EA, estão de acordo no que diz respeito à sistematização da informação acerca do domínio de um problema, com o propósito de aumentar o potencial da reutilização de software.

### 2.2.2 – A LINHA DE PRODUTOS DE SOFTWARE

O termo **Linha de Produtos de Software** (LP) vem sendo usado para definir um novo paradigma em Engenharia de Software, cujo propósito é guiar organizações no desenvolvimento **para** e **com** reutilização. De fato, é uma vertente da Engenharia de Domínio, cujo foco foi transferido para o âmbito empresarial (LEE *et al.*, 2002).

O SEI (*Software Engineering Institute*) define LP da seguinte maneira (SEI, 2005c):

"Linha de Produtos de Software é um conjunto de sistemas de software que compartilham um conjunto de características comuns e controladas, que satisfazem necessidades de um segmento de mercado em particular, e são desenvolvidos a partir de artefatos (*core assets*), de forma predefinida".

Alguns conceitos são importantes para a definição de uma LP, como, por exemplo, o conceito de **núcleo de artefatos (*core assets*)**, e também o conceito de **variabilidade**. Estes conceitos são detalhados a seguir.

Cada sistema pertencente a uma LP é um sistema como outro qualquer, com a ressalva de que eles são criados a partir de componentes vindos de uma base de artefatos comuns, moldados de acordo com regras definidas pela arquitetura. No entanto, ao longo do processo de desenvolvimento de uma LP, aspectos particulares de cada produto podem ser explicitados, tornando-se necessário adicionar estes novos componentes, ou realizar algum outro tipo de adaptação. Assim, como na ED, os pontos onde tais aspectos se manifestam - isto é, os pontos em que as características dos produtos podem se diferenciar – constituem a variabilidade da linha de produtos.

O núcleo de artefatos forma a base do paradigma de linha de produtos (NORTHROP, 2002) e pode ser definido como o conjunto de elementos que serão reutilizados por cada sistema a ser desenvolvido. Incluem artefatos como a arquitetura do software, componentes de software reutilizáveis, modelos de domínio, requisitos, modelos de desempenho, cronogramas, orçamentos, planos e casos de testes, planejamentos e descrição de processos.

A definição de quais desses artefatos vão compor a base para reutilização é parte da atividade de **Desenvolvimento do Núcleo de Artefatos**, que juntamente com o **Desenvolvimento de Produtos** e **Gerenciamento da Linha de Produtos** formam as três atividades essenciais de uma LP. Tais atividades estão intrinsecamente relacionadas

entre si, de maneira que qualquer alteração em uma delas implica em analisar o impacto causado nas outras. A Figura 2.2 ilustra essa relação.

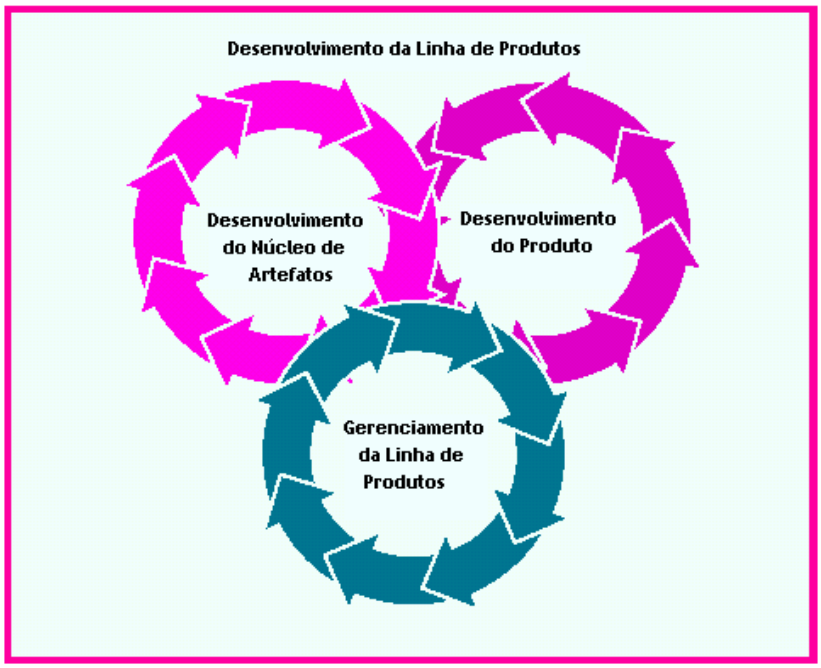



**Figura 2.2 - Atividades Essenciais da Linha de Produtos**
**(Adaptado de(NORTHROP, 2002))**

Na Figura 2.2, as setas circulares representam não só que os artefatos são utilizados pela atividade de desenvolvimento do produto, mas também que estes são constantemente revisados, sejam artefatos já existentes na base ou artefatos candidatos a integrantes da base. Além disso, existe um *feedback* entre a atividade de desenvolvimento do produto e o desenvolvimento dos artefatos, sendo estes atualizados à medida que novos produtos são construídos.

No desenvolvimento de uma LP, é possível identificar os artefatos consumidos e produzidos em cada uma das atividades essenciais. Uma breve descrição dessas atividades é apresentada a seguir.

**Desenvolvimento do Núcleo de Artefatos:** entende-se por "desenvolvimento" do núcleo de artefatos todas as formas de aquisição destes, incluindo a construção ("do zero" ou com reutilização), ou aquisições de tais artefatos por meio de compra ou encomenda de terceiros. Dentre esses artefatos reutilizáveis, pode-se encontrar um *framework*, um componente, uma arquitetura de software, ou alguma documentação relativa a algum componente ou à arquitetura de software. No entanto, a peça-chave do conjunto de artefatos desenvolvidos é a arquitetura, que deve ser comum a todos os produtos relacionados a, ou desenvolvidos por meio da, LP (NORTHROP, 2002).

A atividade de desenvolvimento do núcleo de artefatos tem como objetivo (CLEMENTS, 1999):

- **Escopo da Linha de Produção**: descrição dos produtos iniciais que constituirão a LP, moldada nos termos de suas variabilidades . Define o escopo em que a LP estará incluída.
- **Artefatos do núcleo**: Artefatos que serão a base para a produção dos produtos na LP. A arquitetura comum aos produtos da LP, componentes de software, com seus planos e casos do teste, planos de integração, etc., bem como componentes COTS (*commercial off-the-shelf*), quando adotados, constituem artefatos deste núcleo.
- **Plano de Produção**: Descreve como os produtos serão produzidos a partir dos artefatos do núcleo.

De fato, os produtos resultantes dessa atividade, isto é, o escopo da LP, os artefatos que formarão o núcleo e o plano de produção, são fundamentais para assegurar a viabilidade da produção de uma LP.

**Desenvolvimento do Produto:** a atividade de desenvolvimento do produto se beneficia do resultado da atividade de desenvolvimento dos artefatos do núcleo, além dos requisitos individuais dos produtos específicos que serão desenvolvidos (NORTHROP, 2002). Essa pode ser considerada a principal atividade da abordagem de LP e é ilustrada pela Figura 2.3.

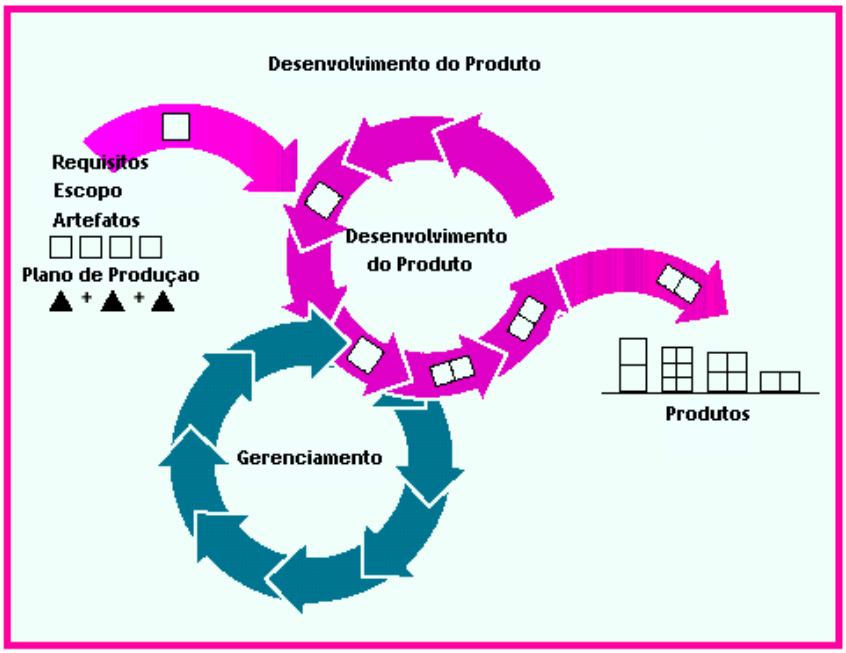



**Figura 2.3 - Desenvolvimento do Produto**

Dentre os artefatos provenientes do núcleo, os mais comuns são, de acordo com (GIMENES & TRAVASSOS, 2002): a) modelo do domínio; b) *framework* de arquitetura de linha de produtos, que especifica a arquitetura genérica para todos os produtos; e c) componentes reutilizáveis que serão integrados à arquitetura para gerar um produto.

Além disso, algumas atividades estão presentes na fase de desenvolvimento do produto, como por exemplo:

- **análise baseada no modelo de domínio**: tem por objetivo analisar um *modelo de domínio* no intuito de verificar o quanto tal modelo representa as necessidades do cliente em termos dos artefatos reutilizáveis do núcleo, no que diz respeito à arquitetura genérica e aos outros componentes disponíveis. Associada a essa atividade pode existir o *modelo de decisão*. Ainda assim, se existirem necessidades do cliente que não forem cobertas pelo modelo de domínio, estas se tornam novos requisitos e podem passar a integrar o núcleo de artefatos. Desse modo, fica explícito o *feedback* que a atividade de desenvolvimento do produto provê à atividade de desenvolvimento do núcleo de artefatos.
- **instanciação da arquitetura do produto:** a partir da arquitetura genérica da LP, chega-se à arquitetura específica do produto a ser desenvolvido, tomando-se as decisões necessárias acerca da seleção dos pontos de variabilidades de maneira incremental.
- **povoamento da arquitetura:** com os componentes adequados, que podem ter origem não só nos artefatos do núcleo, como podem ser adquiridos por terceiros.

Criar produtos pode ter um forte impacto no escopo da linha de produto, em artefatos do núcleo, em planos de produção, e mesmo nos requisitos para produtos específicos. Dessa maneira, é possível inferir que o desenvolvimento do produto pode variar extremamente em função dos artefatos, do plano de produção, e do contexto organizacional (NORTHROP, 2002).

**Gerenciamento da Linha de Produto:** a atividade de gerenciamento do produto é responsável em grande parte pela viabilidade e sucesso da abordagem de LP. É necessário que seja estabelecido um compromisso da organização em supervisionar, coordenar e documentar as atividades anteriormente mencionadas.

Em última instância, deve haver um sincronismo entre o grupo que desenvolve os artefatos e o grupo que gera os produtos. Deve, também, ser garantido o apoio ao desenvolvimento e à evolução dos artefatos da LP (NORTHROP, 2002). Uma descrição

mais detalhada das atividades que constituem uma LP pode ser encontrada em (OLIVEIRA, 2003)

### 2.2.3 – CONSIDERAÇÕES SOBRE ED E LP

Embora a comparação entre as técnicas de ED e LP esteja fora do escopo deste trabalho, é possível verificar algumas semelhanças e diferenças entre as duas abordagens de reutilização. Nota-se que a atividade de Desenvolvimento do Núcleo de Artefatos de uma LP é muito semelhante ao processo de Análise de Domínio na ED, principalmente no que diz respeito à geração de artefatos reutilizáveis. Embora a abordagem de LP possa levar à crença de que exista algum tipo de repositório originado pela atividade de desenvolvimento do núcleo de artefatos, o que não fica explícito na ED, percebe-se que tanto a Análise de Domínio em ED quanto o Desenvolvimento do núcleo em LP apresentam o mesmo objetivo, que é a organização de informações e artefatos que venham a ser reutilizados.

Uma outra semelhança é observada ao se comparar a atividade de Desenvolvimento do Produto, em LP com o processo de Engenharia de Aplicação, na ED. Ambos consideram a variabilidade para desenvolver seu produto, ou instanciar sua aplicação, incorporando nesta atividade elementos que representem requisitos específicos da aplicação ou produto em questão.

Uma diferença que também pode ser observada entre as abordagens de ED e LP é a maneira pela qual ocorre a definição de escopo, i.e., a abrangência dos elementos que serão reutilizados. Enquanto a ED parte de fontes como especialistas ou livros para organizar o conhecimento do domínio e obter os pontos passíveis de reutilização, a LP define seu escopo a partir de produtos pré-determinados, i.e., produtos previamente escolhidos para fazer parte da LP e, a partir destes, as características comuns e variáveis entre eles são abstraídas e generalizadas, de forma a fazerem parte do núcleo de artefatos que serão reutilizados.

Atkinson et al (2002) enxergam a ED como um importante passo no suporte à reutilização. No entanto, eles alegam que, na prática, a definição de escopo (ou de domínio) se apresenta como uma das maiores de suas limitações e/ou dificuldades. Afirmam, ainda, que o sucesso do esforço da ED (ou de técnicas de reutilização em geral) é altamente sensível a uma definição correta do escopo, apresentando a abordagem de LP como uma alternativa para a solução de tal problema, como ilustram a Figura 2.1, apresentada anteriormente, e Figura 2.4.

A Figura 2.1 representa a ED e EA explorando todo o ciclo de vida de um software, se valendo de informações do mundo real para documentar e construir o domínio, enquanto que a Figura 2.4 apresenta uma definição de escopo da LP a partir de produtos pré-selecionados para integrar a LP.

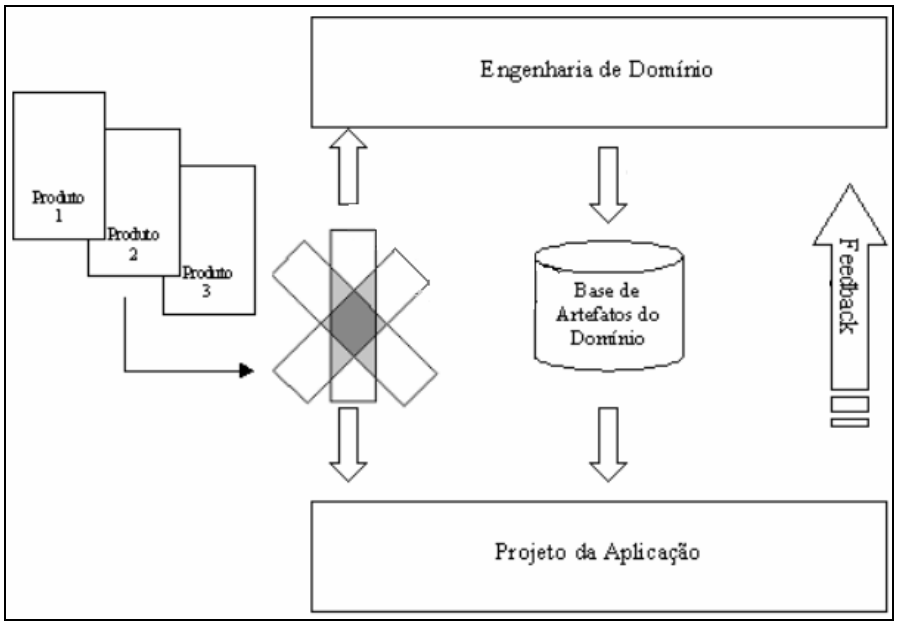



**Figura 2.4 - Reuso com Linha de Produtos**
**(Adaptado de (ATKINSON *et al.*, 2002))**

Enfim, independentemente de tais semelhanças e diferenças, o que se pode constatar é que ambas as abordagens utilizam o conceito de variabilidade para nortear seus processos de reutilização. Detalhes a respeito de tal variabilidade são apresentados na próxima seção.

## 2.3 - VARIABILIDADE NA REUTILIZAÇÃO DE SOFTWARE

O conceito de Variabilidade é reconhecido como um conceito-chave para a reutilização sistemática e bem sucedida. Em abordagens baseadas em famílias de software, como LP ou ED, a variabilidade é uma maneira de se lidar com as inevitáveis diferenças entre os produtos e/ou aplicações, bem como explorar as características comuns entre eles (BECKER, 2003).

A alteração de decisões de projeto já tomadas acarreta um alto custo no desenvolvimento. Para que eventuais alterações sejam evitadas, engenheiros de software tentam adiar ao máximo a tomada de decisão, até as fases mais adiantadas do

desenvolvimento do software. Esse adiamento de decisões de projeto constitui uma das principais motivações para o tratamento de Variabilidade, uma vez que tal tratamento permite documentar as possibilidades de configuração do software até o momento da tomada de decisão. Isso aumenta consideravelmente a capacidade de configuração do software e o seu potencial de reutilização, conforme salientado em (SVAHNBERG *et al.*, 2002).

Trabalhos encontrados na literatura (KANG *et al.*, 1990; GRISS *et al.*, 1998; MASSEN & LICHTER, 2002; RIEBISCH *et al.*, 2002; BOSCH, 2004) apresentam diversos conceitos inerentes à variabilidade e sua modelagem. Tais conceitos são importantes para possibilitar o entendimento entre todos os envolvidos no desenvolvimento de um software (e.g. usuário, especialistas de domínio, desenvolvedores).

Definições encontradas na literatura apresentam o termo Variabilidade como a habilidade que um sistema ou artefato de software possui de ser alterado, customizado, ou configurado para um contexto em particular (BOSCH, 2004). Ou ainda, segundo a terminologia de Svahnberg *et al.* (2002), denota-se por Variabilidade a área que gerencia as partes do processo de desenvolvimento de software e seus artefatos resultantes, que foram feitos para diferenciar produtos, ou, em certas situações, diferenciar características de um mesmo produto.

Embora exista, também, o conceito de Variabilidade no Tempo, que lida com a habilidade de um software suportar evolução e alteração de requisitos, nos seus vários contextos (também conhecida como Gerência de Configuração), o presente trabalho se limita a tratar da chamada Variabilidade no Espaço, que é aquela que lida com as variabilidades entre os produtos ou aplicações individuais no espaço de domínio de uma LP, ou Domínio, em qualquer ponto fixo no tempo (KRUEGER, 2002; BOSCH, 2004).

Assim, conceitos encontrados na literatura compreendem:

- **Pontos de Variação**: determinados pontos em um sistema de software em que decisões são tomadas a respeito, por exemplo, de qual variante será utilizada. Em outras palavras, pontos de variação refletem a parametrização no domínio de uma maneira abstrata e são configuráveis por meio das variantes. De acordo com a descrição encontrada em (SCHMID, 1997), que considera o uso de *frameworks*[2] como técnica de reutilização de software, os pontos de variação podem ser

[2] Projeto reutilizável de um sistema, ou parte dele, representado por um conjunto de classes e os relacionamentos entre suas instâncias. (JOHNSON, 1997)

denominados *hot-spots*. Tais *hot-spots* podem ter diferentes alternativas de configuração, que distinguem as aplicações que serão construídas a partir do *framework*.

- **Variantes**: alternativas de implementação disponíveis para um ponto de variação são denominados Variantes. Em outras palavras, são elementos necessariamente ligados a um ponto de variação, que atuam como alternativas para se configurar aquele ponto de variação.

- **Invariantes**: na literatura pesquisada, os elementos "fixos", que não são configuráveis no domínio, não recebem nenhum tipo de nomenclatura especial. Para diferenciá-los dos outros, estes serão chamados, neste trabalho, de elementos invariantes. Considerando o uso de *frameworks* como técnica de reutilização, tais elementos invariantes são denominados *frozen-spots* (SCHMID, 1997).

- **Elementos Opcionais**: descrevem elementos que podem ou não estar presentes em produtos desenvolvidos em uma LP ou aplicações instanciadas a partir de um domínio.

- **Elementos Mandatórios**: descrevem elementos que devem obrigatoriamente estar presentes em produtos desenvolvidos em uma LP ou aplicações instanciadas a partir de um domínio.

Um conceito que está intimamente ligado aos conceitos de variabilidade descritos anteriormente é o conceito de dependência e mútua exclusividade entre elementos de um domínio. Durante a instanciação de uma aplicação a partir de um domínio ou no desenvolvimento de um novo produto em uma LP, deve haver alguma maneira de se indicar a necessidade de seleção de elementos em conjunto, ou, ao contrário, a incompatibilidade da seleção conjunta de tais elementos. Tais indicações possibilitam a derivação de uma aplicação ou produto mais consistente em relação às especificações do domínio.

Tão importante quanto entender os conceitos inerentes à variabilidade, é modelar tais conceitos correta e explicitamente, de maneira a possibilitar o entendimento de desenvolvedores, projetistas e demais envolvidos no desenvolvimento de software. A modelagem dessa variabilidade é tratada a seguir.

## 2.4 - REQUISITOS PARA A REPRESENTAÇÃO DE VARIABILIDADES

Tratar as diferenças entre os produtos ou aplicações é uma das questões essenciais que devem ser consideradas quando se constrói artefatos reutilizáveis. Tal atividade é conhecida por Tratamento de Variabilidade e se inicia no momento de reconhecimento de um ponto onde seja possível uma configuração no sistema. A partir disso, é possível obter o mapeamento deste ponto para o sistema específico, passando pela modelagem das variabilidades, o que constitui uma das mais importantes tarefas desta atividade (GIMENES & TRAVASSOS, 2002; MYLLYMÄKI, 2002).

Alguns trabalhos direcionam seu foco para a modelagem de variabilidade na fase de análise, como o método FODA (KANG *et al.*, 1990), que utiliza modelos de características, e seus sucessores. Outros estão focados na fase de projeto (GARG *et al.*, 2003; HOEK, 2004) e/ou implementação (ANASTASOPOULOS & GACEK, 2001). Na realidade, o tratamento da variabilidade, com ênfase na sua modelagem, deve ser considerado em todas as fases do desenvolvimento de uma família de aplicações ou produtos.

Massen e Lichter (2002) acrescentam que a notação para a modelagem de variabilidade pode ser gráfica, textual ou um misto das duas formas. Salientam, no entanto, que em uma notação gráfica, os pontos de variação são reconhecidos muito mais facilmente. Em seu trabalho, os autores apresentam uma lista de requisitos desejáveis a uma notação que tenha por objetivo expressar variabilidade em um artefato de software. No entanto, tais requisitos são apresentados de maneira genérica, não especificando os conceitos relacionados à variabilidade identificados na seção anterior. Como um refinamento da lista de Massen e Lichter, é proposta neste trabalho uma nova lista de requisitos que devem ser satisfeitos por uma notação. Tais requisitos são derivados de uma compilação dos conceitos apresentados nos diversos trabalhos na literatura e compreendem:

- **Representação gráfica de Elementos (invariantes) mandatórios**: uma notação deve ser clara ao representar as partes comuns a todos os produtos de uma família de aplicações ou produtos de software, deixando explícito para o especialista quais elementos serão, obrigatoriamente, selecionados em um produto/aplicação específico.

- **Representação gráfica de Elementos (invariantes) opcionais**: uma notação deve deixar claro para o especialista quais elementos podem ser descartados na construção de um produto/aplicação específico.
- **Representação gráfica de Pontos de Variação mandatórios**: é desejável que os pontos onde o software pode ser configurado, para dar origem a diferentes produtos/aplicações, sejam documentados em artefatos e visualizados facilmente em um modelo de software, assim como o fato daquela configuração ser indispensável.
- **Representação gráfica de Pontos de Variação opcionais**: assim como nos pontos de variação mandatórios, é desejável que o especialista saiba que a configuração em um determinado ponto é dispensável.
- **Representação gráfica de Elementos Variantes mandatórios**: as variantes e sua ligação com os pontos de variação devem ser representadas explicitamente, principalmente aquelas que devem ser selecionadas na configuração de um determinado software.
- **Representação gráfica de Elementos Variantes opcionais**: da mesma maneira que as variantes mandatórias, as variantes opcionais e sua ligação com os pontos de variação devem ser explicitamente representadas em um artefato de software.
- **Representação de Dependência/Exclusividade entre elementos**: relações de dependência entre dois ou mais elementos, bem como relações de mútua exclusividade devem ser explicitamente representadas por uma notação, para quaisquer conjuntos de elementos, independentemente de suas posições no diagrama.
- **Representação de Relacionamentos entre elementos:** é desejável que uma notação deixe explícita a relação entre elementos de um modelo. Entre tais relacionamentos podem ser incluídos o relacionamento existente entre variantes e pontos de variação, relacionamentos entre diferentes tipos de características, como por exemplo, características de análise e de projeto, e outros relacionamentos relevantes, que tenham semântica para a instanciação de elementos de uma família de aplicações ou produtos de software.

Com base nesta lista de requisitos foi realizada uma análise em algumas notações existentes na literatura, cujo propósito é a representação de variabilidades em uma família de aplicações ou produtos de software. Face à grande quantidade de trabalhos encontrados, foi feita uma tentativa de definir um escopo, i.e., um subconjunto de notações selecionadas para esta análise. Uma vez que o modelo de características é o

modelo de mais alto nível de abstração, e ponto de partida para o recorte necessário à instanciação de novas aplicações ou produtos, foram analisadas somente notações que se propunham a representar variabilidades no modelo de características. Tal análise é apresentada na próxima seção.

## 2.5 - NOTAÇÕES PARA REPRESENTAÇÃO DE VARIABILIDADE EM MODELOS DE CARACTERÍSTICAS

A Modelagem de Características é uma abordagem que trata da complexidade em expressar diversos requisitos em forma de características e da sua estruturação hierárquica em diagramas de características (MASSEN & LICHTER, 2004).

Diversas definições são atribuídas ao termo características (*features*). No presente trabalho, será adotada a definição de Kang *et al*. (1990), uma das principais referências em modelos de características para a ED. Segundo Kang, uma característica pode ser definida como "um aspecto, uma qualidade, ou uma característica visível ao usuário, proeminente ou distinta, de um sistema (ou sistemas) de software".

Ainda de acordo com Kang, um modelo de características tem como propósito capturar o entendimento de clientes e usuários acerca das capacidades gerais de uma aplicação em um domínio. Tal modelo representa as características-padrão de uma família de sistemas em um domínio e as relações entre elas.

Nesta seção, serão apresentadas algumas notações que utilizam o modelo de características para representar variabilidade.

### 2.5.1 - FEATURE ORIENTED DOMAIN ANALYSIS (FODA) (KANG *et al*., 1990)

O método FODA é um método de ED e foi apresentado por Kang *et al*. em 1990, sendo o precursor das notações baseadas em modelos de características. Tal método estabelece três atividades majoritárias:

- Análise de contexto, cujo propósito é definir o escopo do domínio e avaliar as relações entre os elementos do domínio e os elementos externos a este.
- Modelagem do Domínio, que inclui a subatividade de Análise de Características, e cujo propósito é analisar e representar as semelhanças e diferenças identificadas para o domínio, além de produzir os diversos modelos que representam diferentes aspectos do domínio.
- Modelagem da Arquitetura, cuja finalidade é fornecer uma "solução" aos problemas definidos na fase de modelagem do domínio. Um modelo de arquitetura (conhecido

também como arquitetura de referência) é desenvolvido nesta fase, e nele pode ser feito o detalhamento do projeto.

A Figura 2.5 mostra um exemplo da notação FODA. No método FODA, tais diagramas de características são árvores (representadas em forma de grafo ou de forma hierárquica) que capturam os relacionamentos entre características. A raiz da árvore representa o domínio que está sendo descrito e os nós restantes denotam características e suas sub-características.

Uma característica é considerada mandatória, a menos que um círculo vazio apareça acima de seu nome, indicando que, neste caso, aquela é uma característica opcional, como a característica *Ar Condicionado*, na Figura 2.5. As características variantes no FODA são consideradas como especializações de uma característica mais geral. Assim, as características alternativas, que são filhas da mesma característica-pai e conectadas por um arco, definem o conjunto de características do qual pelo menos uma característica pode ser selecionada, exemplificadas pelas características *Manual* e *Automático*, na Figura 2.5. Dependências e mútua exclusividade entre características podem ser expressas com regras de composição (*Composition Rules*), evitando a sobrecarga gráfica. Ainda na Figura 2.5, observa-se a regra de composição *Ar Condicionado* requer *Cavalo-Vapor*, o que significa que é necessário uma certa potência do motor para que o carro tenha ar condicionado.

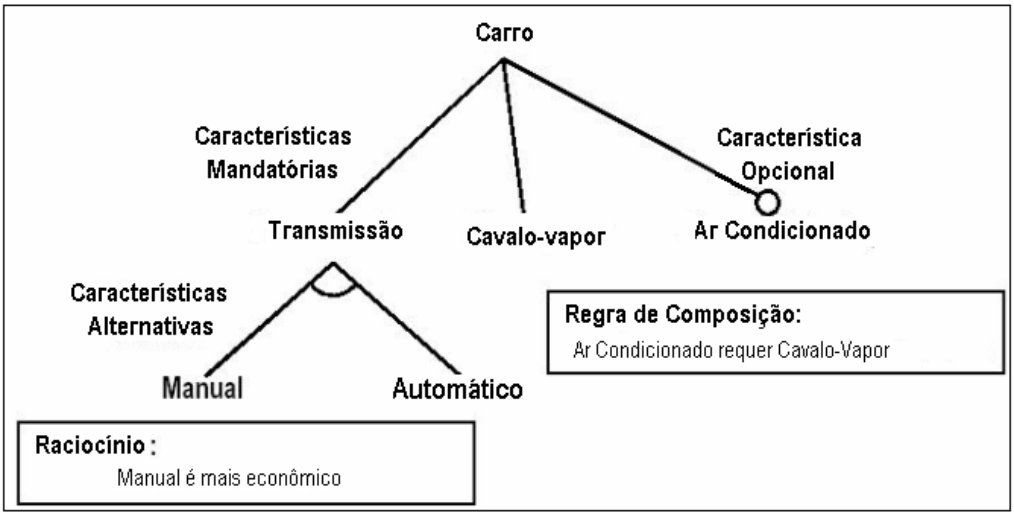



**Figura 2.5 - Notação para variabilidades no método FODA**

**(Adaptado de(KANG *et al.*, 1990))**

### 2.5.2 - FEATURE ORIENTED REUSE METHOD (FORM) (KANG *et al*., 2002; LEE *et al*., 2002)

O método FORM é uma extensão do método FODA para LP, introduzido por Kang *et al.* em 2002. Neste método, são estabelecidos vários tipos de características correspondentes às fases envolvidas em uma LP. As características são classificadas e agrupadas em camadas da seguinte maneira:

- Camada de Capacidades: agrupa características que caracterizam funcionalidades, serviços e operações de um determinado domínio;
- Camada de Ambiente Operacional: as características dessa camada representam atributos de um ambiente no qual uma aplicação do domínio pode operar, incluindo características da plataforma de hardware adotada;
- Camada de Tecnologia de Domínio: composta por características que representam questões de implementação de mais baixo nível, específicos para o contexto de um domínio;
- Camada de Técnica de Implementação: as características desta camada também representam detalhes de implementação de mais baixo nível, no entanto, de maneira mais genérica que as relativas à camada de tecnologia de domínio, no sentido de não serem específicos para um domínio.

Nesta notação, os símbolos do nó são os mesmos que no método FODA, mas todos os nomes das características são descritos em caixas, que representam as camadas. As características mandatórias e opcionais são idênticas ao FODA, bem como a mútua exclusividade das características variantes. Os relacionamentos entre as características são "Composição", "Especialização/Generalização" e "Implementado por". O relacionamento “Implementado por” indica que uma característica é necessária para implementar outra. A Figura 2.6 ilustra a notação do método FORM.

Na Figura 2.6, é representado um modelo de características referente a um sistema integrado de sensores domésticos, denotado pela sigla HIS (*Home Integration System*). A característica HIS pertence à Camada de Capacidades e tem um relacionamento do tipo ”Composição” com as características *Administração*, *Serviços*, etc. Na Camada de Ambiente Operacional, tem-se a característica opcional *Internet*, que tem um relacionamento do tipo “Implementado-por” com a característica *Conexão*, pertencente à Camada de Técnica de Implementação. Tal característica apresenta ainda as alternativas *TCP* e *UDP*, indicando que existe mais de uma forma de conexão para implementar a comunicação de algum evento detectado pelo sistema de sensores. Tais

características alternativas são consideradas especializações de *Conexão*. A Camada de Tecnologia de Domínio é representada pela característica *Monitoramento e detecção*, que implementa o mecanismo de detecção de *Movimento*, *Fumaça* e *Umidade* do sistema *HIS*.

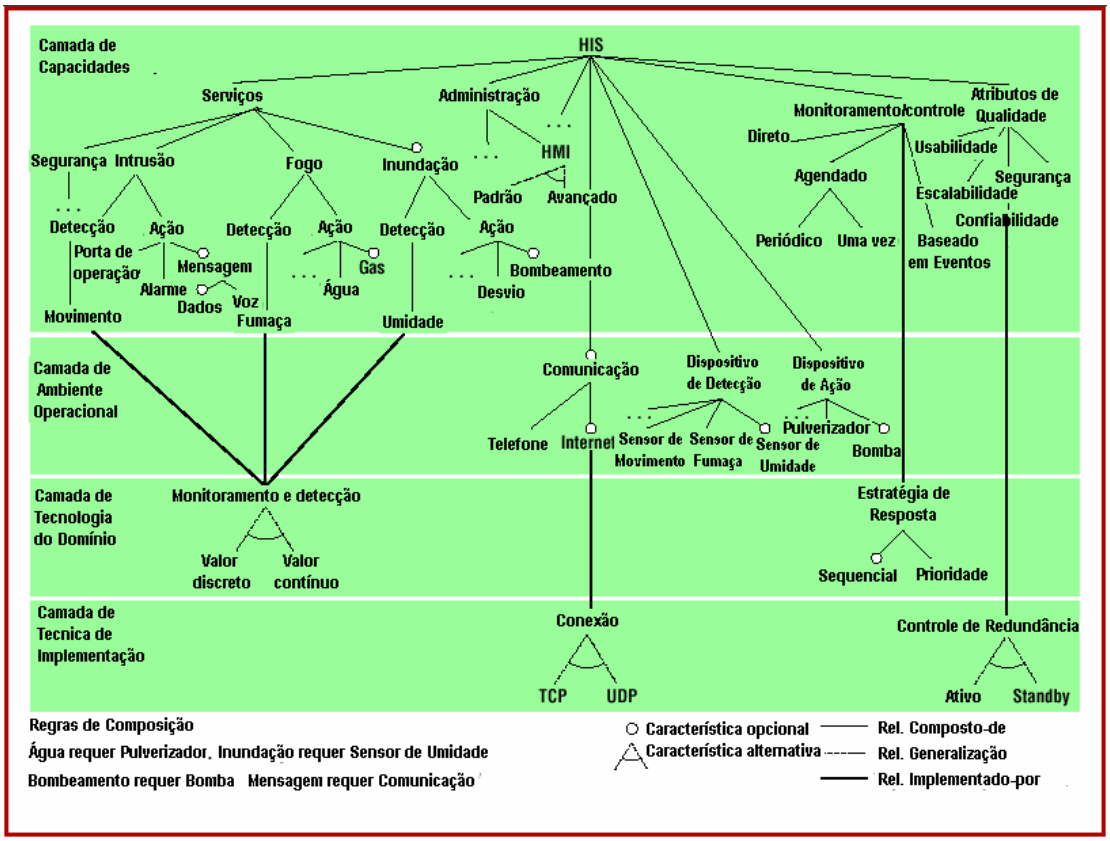



**Figura 2.6 - Notação para variabilidades no FORM**

**(Adaptado de(KANG *et al.*, 2002))**

### 2.5.3 - FEATURSEB (GRISS *et al*., 1998)

O método FeatuRSEB é um método de ED resultante de uma combinação entre os métodos FODA e RSEB (*Reuse-Driven Software Engineering Business*). O RSEB é um processo sistemático de reutilização orientado a casos de uso. A arquitetura e os subsistemas reutilizáveis são descritos, primeiramente, em termos de casos de uso e então transformados em modelos de objeto que são rastreáveis a estes casos de uso. A variabilidade no RSEB é capturada estruturando-se modelos de objetos e casos de uso, utilizando-se explicitamente pontos de variação e variantes.

A notação é apresentada na Figura 2.7 e inclui as seguintes construções:

- O relacionamento "composto_de" indica que uma característica é composta de sub-características e é representado por uma linha simples entre a característica e cada

uma de suas sub características (ex. *Serviço Telefônico* é composto de *Cobrança*, *Modo de Discagem*, etc).

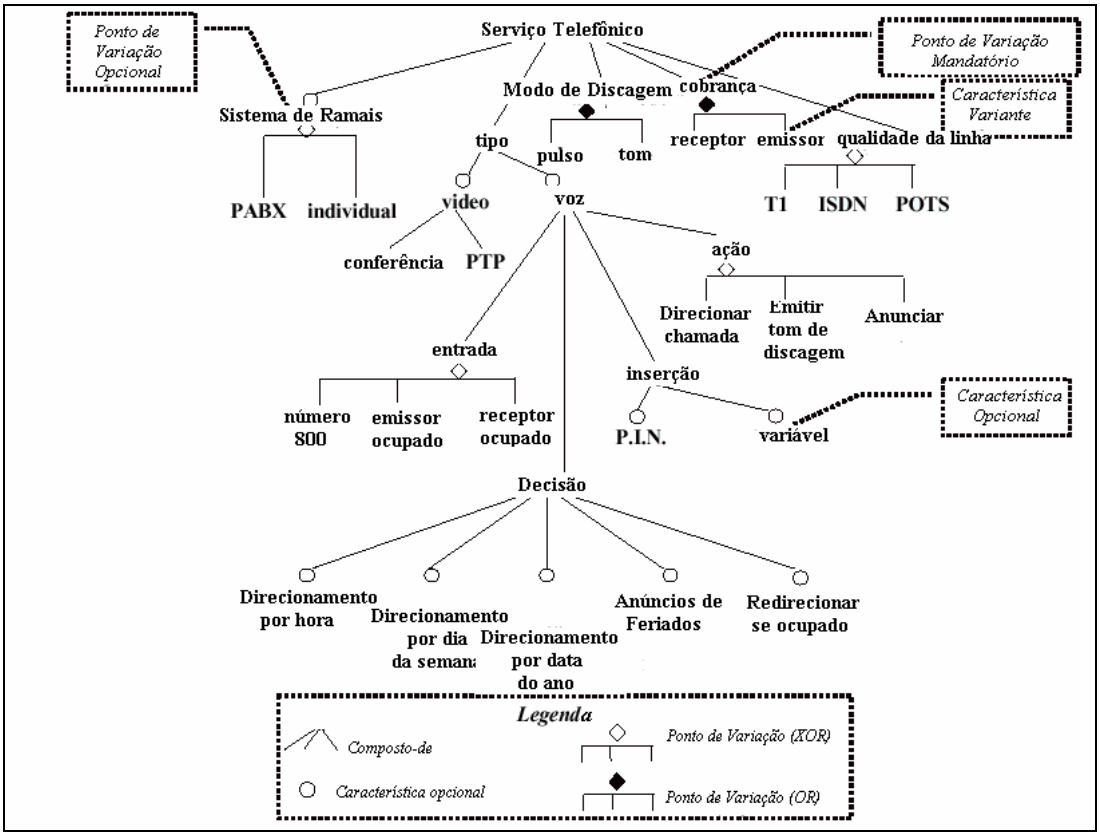



**Figura 2.7 - Notação do método *Featu*RSEB**
**(Adaptado de(GRISS *et al.*, 1998))**

- O atributo de "existência" determina se uma característica é opcional ou mandatória. Uma característica opcional é diferenciada por um círculo simples acima de seu nome. Na Figura 2.7, *Sistema de Ramais* é uma característica opcional e *Cobrança* é uma característica mandatória.
- Pontos de variação são representados por um diamante sob o nome da característica, e suas variantes são representadas por uma linha entre o diamante e o nome da característica variante. No caso de tais variantes serem mutuamente exclusivas, o diamante do ponto de variação é não-preenchido, denotando o relacionamento XOR. Quando as variantes podem ser selecionadas em conjunto, denota-se o relacionamento OR e o diamante do ponto de variação passa a ser preenchido. Na Figura 2.7, *Sistemas de Ramais* é um ponto de variação que tem como variantes *PABX* e *Individual*. Tais variantes são mutuamente exclusivas, como denota o diamante não-preenchido. *Modo de Discagem* é um ponto de variação que tem como

variantes *Pulso* e *Tom*, que possuem um relacionamento do tipo OR, determinado pelo diamante preenchido.

Segundo os autores, a notação apresentada pode alternativamente ser expressa em UML, quando houver a necessidade de expressar uma característica com um maior nível de detalhes. Neste caso, as características são representadas como classes, utilizando o estereótipo <<*feature*>>, e se relacionam por meio de dependências ou associações estereotipadas. Os atributos podem ser usados para representar os tipos de características, ou outras propriedades, enquanto os relacionamentos são utilizados para expressar a conexão e dependência com outras características. Um exemplo da representação em UML é apresentado na Figura 2.8.

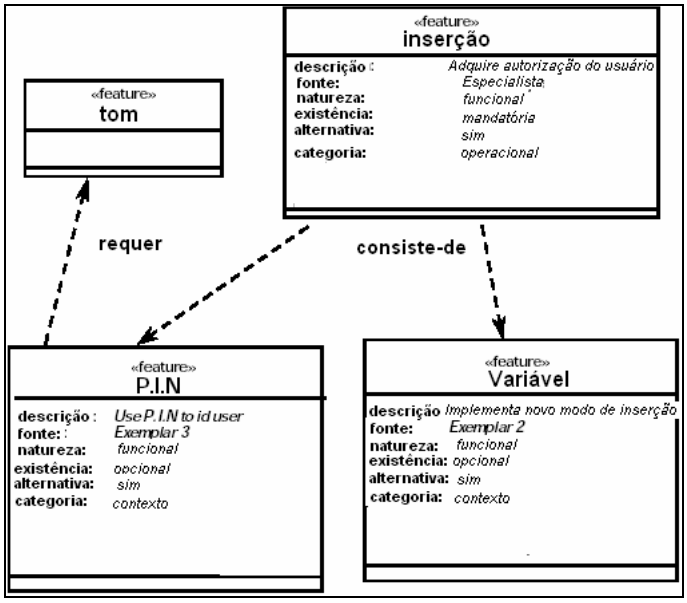



**Figura 2.8 – Representação em UML da notação FeatuRSEB (Adaptado de (GRISS *et al.*, 1998))**

### 2.5.4 - NOTAÇÃO DE SVAHNBERG & BOSCH (SVAHNBERG *et al.*, 2001)

Na notação de Svahnberg & Bosch, descrita em (SVAHNBERG *et al.*, 2001), além das características do tipo Mandatória, Opcional e Variante, já descritas anteriormente em (GRISS *et al.*, 1998), um novo tipo de característica, a Característica Externa, é definida da seguinte maneira: "características externas são características oferecidas pela plataforma alvo do sistema. Mesmo não fazendo parte do sistema diretamente, elas são importantes porque o sistema as usa e delas depende".

Além deste conceito, foi adicionado também o conceito de *Binding Time*, que expressa o momento em que as decisões acerca da variabilidade devem ser tomadas, como por exemplo, no momento de compilação ou de execução.

A notação é ilustrada na Figura 2.9 e apresenta as seguintes propriedades:

- As características são representadas por retângulos.
- O relacionamento com características variantes ou relacionamentos de exclusividade (XOR) são representados por um triângulo não-preenchido (ex. *win32* e *Linux*, na figura).
- O relacionamento OR é representado por um triângulo preenchido em preto (ex. *Pop3* e *IMAP*).
- As características externas são representadas por retângulos pontilhados (ex. *Conexão TCP*).
- A variação entre *binding times* é associada ao relacionamento daquela característica variante (ex. *runtime*, *compiletime*, na figura).

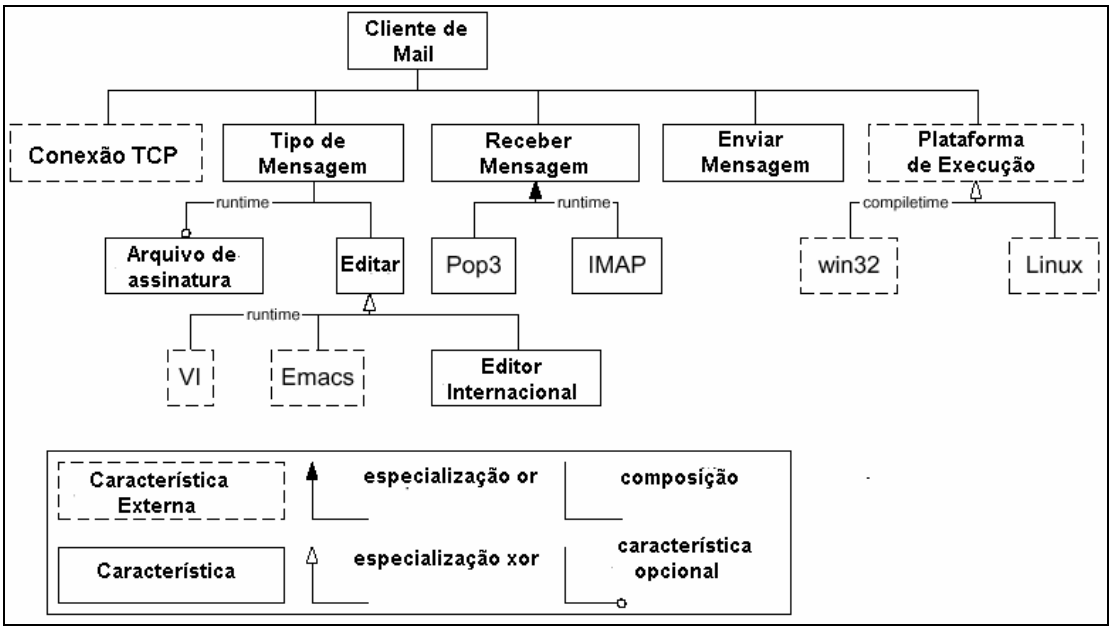



**Figura 2.9 - Exemplo de representação de variabilidades na notação de Svahnberg & Bosch (Adaptado de (SVAHNBERG *et al.*, 2001))**

### *2.5.5* - NOTAÇÃO DE RIEBISCH (RIEBISCH *et al*., 2002)

No trabalho publicado por Riebisch e co-autores, os autores evidenciam o uso de Cardinalidades no diagrama de características, utilizado para modelar uma LP. Cardinalidades são utilizadas para definir o número mínimo e máximo de características

que deve ser escolhido a partir de um conjunto, e, nesta notação, são definidas como sumarizado apresentado a seguir:

**0..1** No máximo uma característica deve ser escolhida a partir de um conjunto de características.

**1** Exatamente uma característica deve ser escolhida a partir de um conjunto de características.

**0..n** Um número qualquer de características (nenhuma, inclusive) pode ser escolhido a partir de um conjunto de características.

**1..n** Pelo menos uma característica deve ser escolhida a partir de um conjunto de características.

No entanto, os autores alegam que, em um caso específico de cardinalidade, como por exemplo, 0..3, ou 3, as notações existentes não são capazes de modelar e propõem uma nova notação para suprir tais necessidades.

Esta notação é apresentada na Figura 2.10 e é caracterizada da seguinte forma: uma característica é um nó em um grafo acíclico e orientado. Os relacionamentos entre as características são expressos por arestas entre estas. Um círculo na extremidade de uma aresta determina o sentido da relação correspondente. Se este círculo for preenchido, a relação entre as características é dita mandatória, ou seja, quando a característica de origem do relacionamento é escolhida, a característica de destino do relacionamento tem que ser escolhida também (ex. *Controle de Leitores*). Se o círculo estiver vazio, então a relação é não mandatória, isto é, a característica de destino do relacionamento não necessariamente deve ser escolhida, e, portanto, é opcional (ex. *Endereço de e-mail*).

Relacionamentos opcionais que são originados de uma mesma característica podem ser combinados em um conjunto. Cada relacionamento pode ser somente parte de um conjunto. Um conjunto tem uma cardinalidade que denota o número mínimo e máximo de características a serem escolhidas do conjunto. As cardinalidades possíveis são: 0..1, 1, 0..n, 1..n, m..n, 0.. *, 1.. *, m.. * (m,n $\in$ N). Um conjunto é mostrado visualmente por um arco que conecte todas as arestas que são parte do conjunto. A cardinalidade é descrita no centro do arco. Por exemplo, a característica *Dados requeridos do leitor* apresenta um conjunto de características opcionais (*Endereço, Aniversário, SSN, Nome de Solteira da Mãe*), de onde no mínimo duas devem ser selecionadas. Tal fato é denotado pela cardinalidade 2..* no centro do arco.

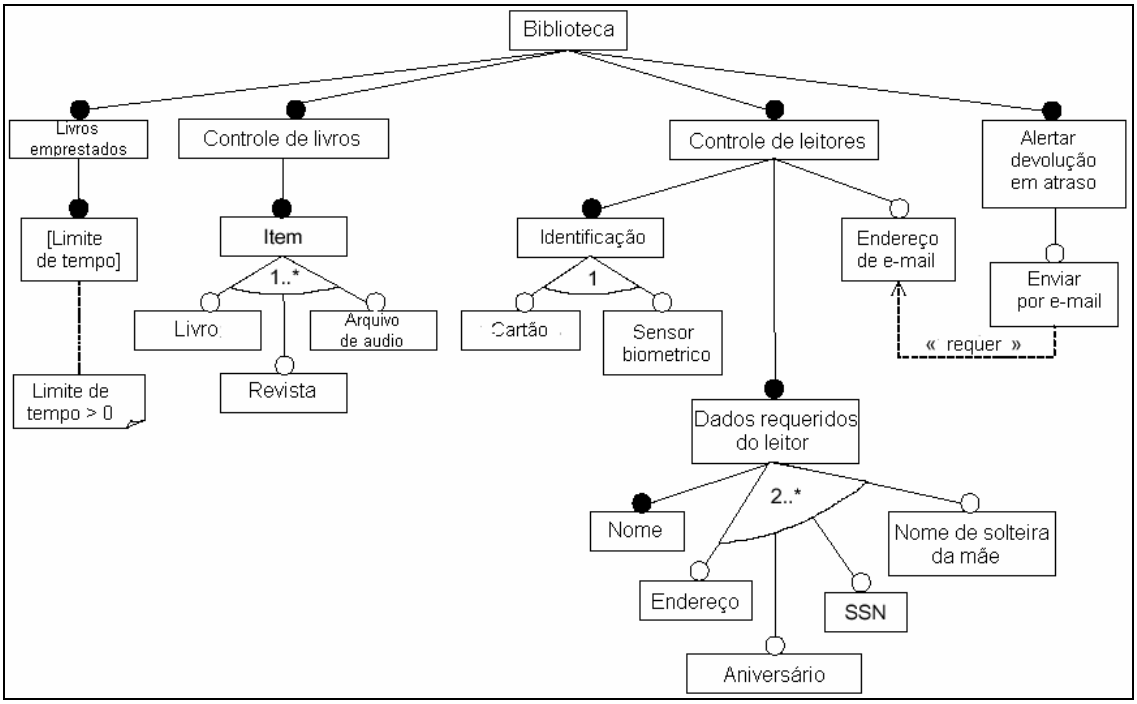



**Figura 2.10 - Exemplo de notação para variabilidades na notação de Riebisch (Adaptado de (RIEBISCH *et al.*, 2002))**

Os relacionamentos entre as características que estão localizadas em diferentes partes, não adjacentes no modelo, não podem ser mostrados no diagrama de características, porque isto reduz a clareza do diagrama. Ao invés disso, tais relacionamentos podem ser descritos em um formulário textual junto com o modelo de características.

### 2.5.6 - NOTAÇÃO DE CECHTICKY (CECHTICKY *et al*., 2004)

O trabalho de Cechticky *et al.* descreve uma técnica de modelagem de características para LP. Tal modelagem é constituída de etapas, cujo propósito final é a criação de um ambiente gerativo para a instanciação de famílias de aplicações ou produtos.

Para fins de comparação, interessa analisar o modelo de características resultante da abordagem, no que diz respeito à representação proposta para modelar os conceitos de variabilidade.

Embora seja destacada a existência de um modelo baseado em XML[3], essa característica é útil quando se trata de geração automática de modelos, o que, como já mencionado, constitui o objetivo final da pesquisa dos autores. Para a análise da notação, tal característica será desconsiderada, sendo levado em consideração apenas o modelo gráfico apresentado. A Figura 2.11 ilustra a notação.

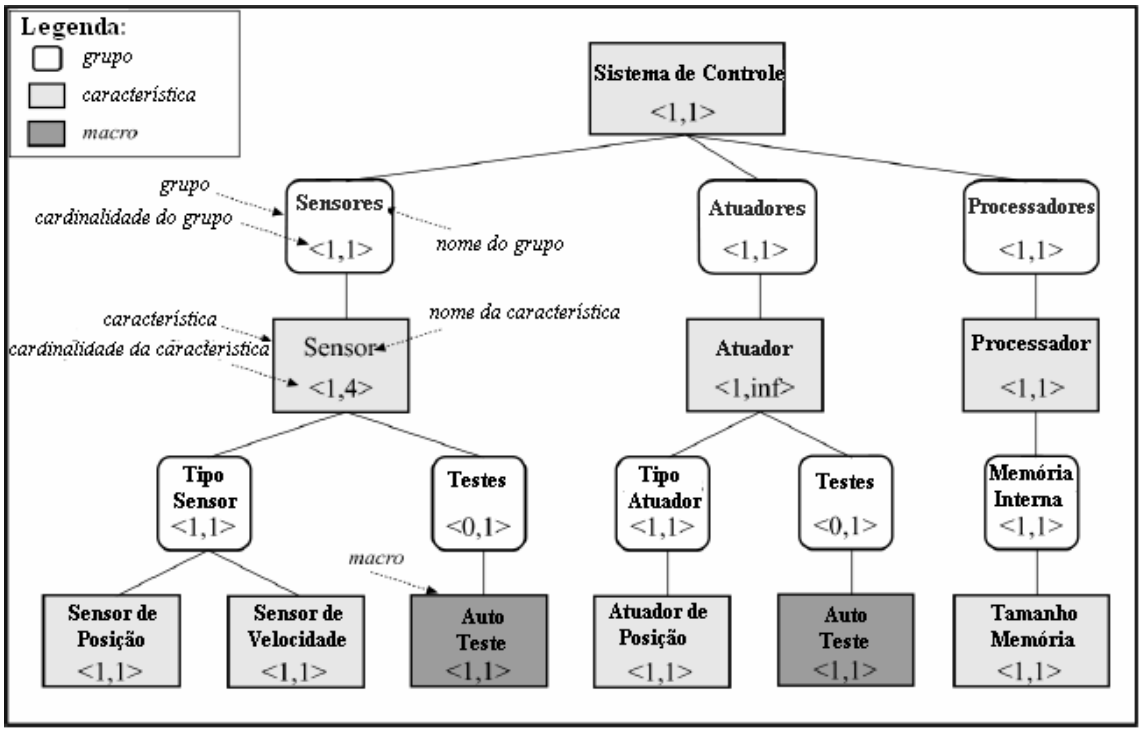



**Figura 2.11- Exemplo de modelo de características na notação de Cechticky**

Nesta notação, as características são representadas por retângulos que contêm o seu nome e a sua cardinalidade. Uma característica não tem relação direta com outra, estando sempre ligadas a um grupo. A mesma característica pode ter diversos grupos unidos a ela. Tanto as características quanto os grupos têm cardinalidade, que podem ser expressas como valores fixos ou como escalas de valores. A cardinalidade de uma característica define o número de instâncias da característica que pode aparecer em uma aplicação ou produto, enquanto que a cardinalidade de um grupo define o número de características daquele grupo que podem ser selecionadas na instanciação de uma aplicação. Na Figura 2.11, o grupo *Sensores* apresenta cardinalidade <1,1>, o que indica que somente uma característica daquele grupo pode ser instanciada. Essa característica é *Senso*r, que por sua vez apresenta cardinalidade <1,4>, indicando que o sensor pode ser instanciado de uma a quatro vezes na aplicação.

[3] *Extensible Markup Language* – formato de texto flexível utilizado para descrição de dados. (XML, 2005)

Os grupos fazem o papel do ponto de variação e são representados por caixas de borda arredondada. Tais caixas contêm o nome e a cardinalidade do grupo. As características ligadas ao grupo são as variantes. É a cardinalidade do grupo que determina se a característica é mandatória ou opcional. Por exemplo, uma característica mandatória está ligada a um grupo que tem cardinalidade <1,1>, enquanto que uma característica opcional está ligada a um grupo que tem cardinalidade <0,1>. Se o grupo tiver cardinalidade <1,1>, mas agrupar mais de uma característica, significa que estas são mutuamente exclusivas. Na Figura 2.11, tal fato é exemplificado pelo grupo *Tipo de Sensor* e as características *Sensor de Posição* e *Sensor de Velocidade*. A cardinalidade <1,1> no grupo *Tipo de Sensor* indica que *Sensor de Posição* e *Sensor de Velocidade* são características variantes mutuamente exclusivas.

Um outro elemento existente no modelo de Cechticky é a característica macro, que representa uma abstração de um conjunto de outras características.

As regras de composição são expressas em um modelo próprio, denominado Modelo de Restrições, separado do modelo de características. Nele, as restrições podem ser do tipo "Requer", "Exclui" ou "Restrição customizada". Tal restrição customizada pode expressar condições como expressões lógicas ou aritméticas envolvendo características. Por exemplo, uma restrição que indica que o sistema de controle deve ter um sensor de posição com capacidade de auto-teste, ou ao menos dois sensores de posição e um processador com um mínimo de 4 kilobytes da memória é representada pela expressão:

```
((count($E1)>0) and (count($E2)>0))
or (count($E2)>2) and ($E3>=4096))
```

onde E1, E2 e E3 são elementos do modelo de características e representam, respectivamente, as características Sensor de Posição com Auto-teste, Sensor de Posição e Tamanho da Memória Interna.

Além destes modelos, ainda existe um outro modelo de características denominado modelo da aplicação, que representa a aplicação instanciada, i.e., o modelo de características onde todas as características são mandatórias e todas as variabilidades foram resolvidas.

### 2.5.7 – NOTAÇÃO DE CZARNECKI (CZARNECKI *et al.*, 2004)

O trabalho de Czarnecki propõe uma notação para a modelagem de características baseada em cardinalidades. Segundo os autores, tal notação integra uma

série de extensões de notações previamente propostas na literatura como por exemplo, (RIEBISCH *et al.*, 2002) e (GRISS *et al.*, 1998) .

A Figura 2.12 ilustra a notação. Nesta notação, a especificação de cardinalidade consiste em uma seqüência de intervalos. No entanto, a cardinalidade é uma propriedade do relacionamento entre as características e não das características em si. Assim, a notação não permite que sejam atribuídas cardinalidades às características.

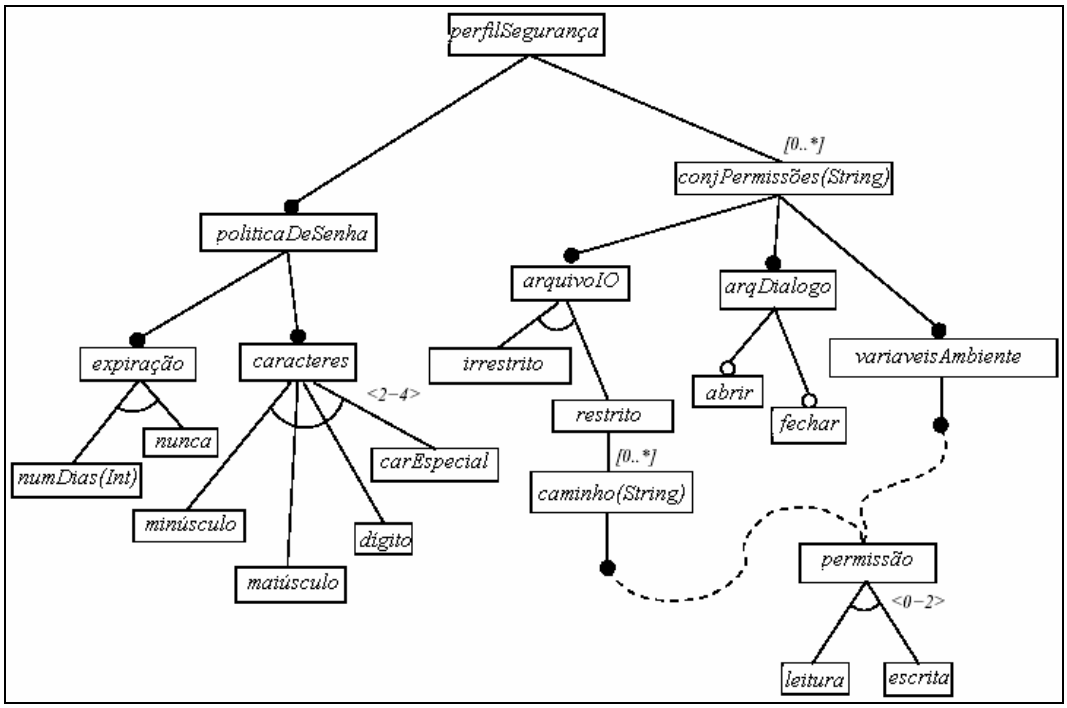



**Figura 2.12 – Exemplo de modelo de características na notação de Czarnecki (Extraído de (CZARNECKI *et al.*, 2004))**

As características são organizadas em forma de árvore e são representadas por retângulos contendo seu nome. Atributos podem, também, ser modelados como características, e neste caso podem possuir um tipo, indicado entre parênteses na frente do nome da característica (ex. *númeroDias(int)*).

O conceito de grupo também está presente nesta notação. Um grupo pode representar um ponto de variação e as características a ele pertencentes são conectadas por um arco e representam variantes. Na Figura 2.12, a característica *Caracteres* representa um grupo (ou ponto de variação) cujos membros (ou variantes) são *Maiúsculo, Minúsculo, Dígito e CarEspecial*. O relacionamento entre um grupo e as características a ele relacionadas pode receber cardinalidade, denotando o número mínimo e máximo de variantes que podem ser instanciadas para aquele ponto de variação (na figura, <2-4>). Quando um grupo possui cardinalidade <1-1>, significa que

as características a ele relacionadas são mutuamente exclusivas, e é mandatória a instanciação de uma das variantes. No entanto, a notação da cardinalidade <1-1> pode ser suprimida, assumindo-se que um grupo sem notação de cardinalidade possui cardinalidade <1-1>. É o caso do grupo *Expiração*, cujos membros, *númeroDias* e *Nunca*, são mutuamente exclusivos. Um grupo que possua cardinalidade [0..*] representa um ponto de variação opcional.

Quando uma característica é mandatória, recebe um pequeno círculo preenchido sobre o retângulo que a representa (ex. *Política de Senha*). Caso o círculo seja não-preenchido, a característica é opcional (ex. *Abrir*, relacionado a *ArqDiálogo*). A opcionalidade também pode ser expressa pela cardinalidade [0..1].

A notação possui ainda o conceito de diagrama de referência, que significa basicamente que uma característica pode estar conectada com outro diagrama de características. Tal conexão é representada por uma linha tracejada entre as características relacionadas e a característica raiz do novo diagrama.

A representação de dependência entre características é apontada como trabalho futuro. No entanto, um trabalho mais recente de Czarnecki (CZARNECKI *et al.*, 2005) continua apresentando o tratamento de dependências como trabalho futuro.

### 2.5.8 - MODELO DE CARACTERÍSTICAS DO AMBIENTE ODYSSEY (MILER, 2000)

O ambiente Odyssey (ODYSSEY, 2005) possui um modelo de características, proposto no contexto do processo de engenharia de aplicação deste ambiente, denominado Odyssey-EA. Tal modelo tem o propósito de não só documentar as características de um domínio, como também servir de terminologia para conceitos e funcionalidades deste.

A Figura 2.13 ilustra a notação. Na notação de Miler, existem diversos tipos de características que são representadas por um retângulo com seu nome, e ícones que representam os tipos. Na figura, é possível observar Características Externas (*LDB* e *Avaliação Externa*), Conceituais (*Curso, Disciplina*), de Entidade (*Diretor* e *Professor*), Adicional (*Monitor*) e Organizacional (*Material Instrucional*). As características mandatórias são representadas por um retângulo de linha contínua (ex. *Disciplina*), enquanto que as características opcionais são representadas por um retângulo com linha tracejada (ex. *Monitor*). O modelo é complementado por um *template* de informações sobre cada característica denominado Padrão de Domínio. Com esse *template*, é possível, entre outras coisas, informar se a característica é uma variante. É possível

observar ainda diversos tipos de relacionamentos, como o de "Dependência", entre as características *LDB* e *Curso*, "Associação", entre *Professor* e *Curso,* "Agregação", entre *Curso* e *Disciplina* e "Composição", entre *Grupo* e *Usuário*.

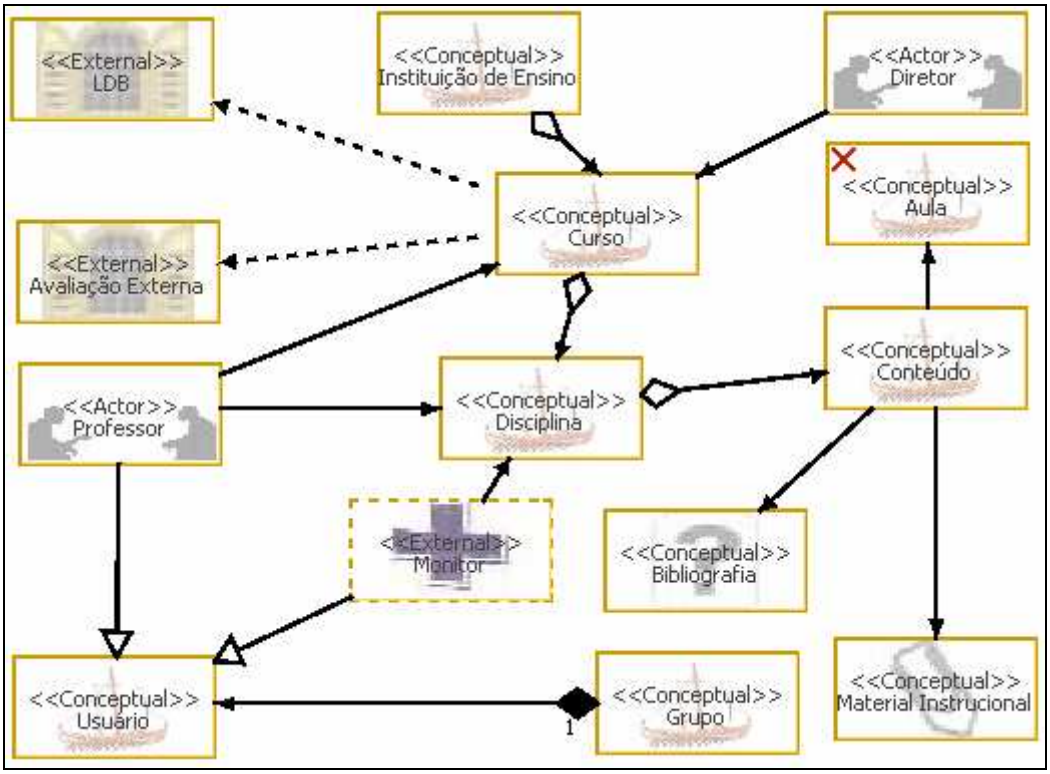



**Figura 2.13 – Modelo de características do ambiente Odyssey**

Os relacionamentos são muito valorizados na notação. No modelo de características podem ser empregados os relacionamentos-padrão da UML (herança, composição, agregação e associação), com o propósito de auxiliar a utilização do modelo de características como uma organização da terminologia do domínio, evidenciando a relação existente entre suas características.

As regras de composição são representadas por restrições inclusivas e exclusivas, também por meio do padrão de domínio: para cada característica é possível informar que características devem ou não ser incluídas em conjunto, por meio do padrão de domínio. Os pares de características envolvidos em uma restrição recebem uma marcação no diagrama. Como exemplo, tem-se a marcação na característica *Aula*, indicando a restrição inclusiva "Aula requer Disciplina".

### 2.5.9 – CONSIDERAÇÕES SOBRE AS NOTAÇÕES ANALISADAS

As notações apresentadas previamente têm por objetivo representar as variabilidades inerentes a uma família de aplicações ou produtos, utilizando modelos de características. Com base nos requisitos para uma notação gráfica que visa a

representação de variabilidades, apontados na seção 2.3, são feitas as seguintes considerações:

- **FODA**: Embora seja uma das primeiras notações para a modelagem de variabilidade e tenha como característica a clareza e facilidade de entendimento, o método não possui expressividade para modelar relacionamentos entre variantes, nem representar explicitamente os pontos de variação. Além disso, o relacionamento generalização/especialização não pode ser representado graficamente. Regras de composição para expressar dependência e exclusividade são oferecidas, mas não há nenhuma consideração explícita a respeito de regras de composição que envolvam um conjunto maior de características. Dessa maneira, o requisito de representação de dependências para quaisquer conjuntos de características não é atendido.
- **FORM**: Por ser derivado do método FODA, apresenta as mesmas deficiências. Além disso, embora tenha introduzido o relacionamento Implementado-Por (*Implemented by*), que é relevante para a modelagem das características, não apresenta nenhum relacionamento que denote a ligação dos pontos de variação e variantes, pelo fato de não representar explicitamente o ponto de variação. Assim, deixa de atender os requisitos de representação explícita de pontos de variação, bem como de relacionamentos relevantes para instanciação do domínio.
- **FeatuRSEB**: Seu ponto mais forte é a representação explícita dos pontos de variação e suas variantes. No entanto, embora os autores da notação classifiquem pontos de variação e outras características como opcionais e mandatórias, não há classificação desse tipo para as características variantes. Além disso, no FeatuRSEB, as relações de dependência são expressas como atributos das características, ou restrições UML, e não explicitamente no diagrama. A princípio, percebe-se que essas relações não existem no diagrama de características. Dessa forma, os requisitos de representação explícita de pontos de variação e variantes opcionais e mandatórios, bem como a representação de dependência não são satisfeitos pela notação.
- **Notação de Svahnberg & Bosch**: Nesta notação, as variantes não são claramente representadas. Subentende-se que variantes são as características sob a representação de especialização, levando a interpretações ambíguas, uma vez que uma característica externa também pode estar sob a representação da especialização. Nesse caso, uma característica externa ao domínio poderia ser uma variante do domínio, o que não parece lógico. Além disso, não há representação explícita de

pontos de variação e nem de dependência entre características, o que claramente deixa de satisfazer à lista de requisitos.

- **Notação de Riebisch**: Esta notação não representa explicitamente os pontos de variação, e nem permite relacionamentos de dependência entre características não adjacentes no diagrama. A representação de dependências em forma de Regras de Composição é prevista utilizando OCL, e não há indícios explícitos da possibilidade de Regras de Composição para um conjunto maior de características. Além disso, não é clara a semântica dos relacionamentos existentes. Um outro ponto fraco é que somente características opcionais podem ser combinadas em conjuntos. Se a representação de conjuntos for utilizada para denotar variantes e pontos de variação, isso pode ocasionar uma modelagem incorreta, uma vez que características fora do conjunto não seriam consideradas variantes, mesmo se o forem.
- **Notação de Cechticky**: Nesta notação, relacionamentos entre características não são permitidos, devendo ser mediados por um grupo. Isso pode se tornar uma desvantagem, à medida que o modelo for crescendo: a introdução de elementos, que a princípio podem ser desnecessários, aumentaria muito a complexidade de leitura e manutenção do modelo. É previsto na notação a construção de regras de composição compostas (ou com operadores booleanos), para expressar dependência e exclusividade entre características. No entanto, da forma como é apresentada, a definição de tais regras é confusa, além da necessidade de um modelo adicional para que sejam expressas. Mais uma vez, isso pode gerar um aumento de complexidade, proporcional ao tamanho do modelo.
- **Notação de Czarnecki:** Nesta notação, um dos pontos fracos é a ausência de padronização para representação de elementos opcionais e mandatórios. Tais elementos podem ser representados de maneira gráfica ou por meio de cardinalidades, porém não é claro quando utilizar uma ou outra opção. Além disso, adota diferentes representações para as cardinalidades (entre <> e entre [] ), não deixando claro se existe uma diferença entre tais representações e quando utilizá-las. Outro ponto fraco é a não-representação de dependências ou mútua exclusividade no diagrama, o que reduz a expressividade do modelo. Relacionamentos entre características também não são considerados. Os autores alegam que relacionamentos devem ser modelados em outro tipo de diagrama, que não o diagrama de características. Isso apresenta uma desvantagem no sentido de que

decisões que poderiam ser tomadas a partir do modelo de características terão de ser transferidas para outros momentos posteriores no ciclo de vida.

- **Modelo de Características do Ambiente Odyssey**: A notação de Miler é útil no sentido de documentar uma terminologia do domínio, de maneira icônica. Uma outra vantagem é a disposição das características, que não necessariamente precisa ser em forma de árvore hierárquica. Os relacionamentos da UML dão uma semântica mais rica às relações existentes entre as características do modelo. No entanto, seus pontos fracos são os seguintes: embora exista a opção de característica variante no padrão de domínio, graficamente, não há nenhuma diferença na representação da variante; pontos de variação não são expressos explicitamente, nem tampouco por meio de classificação no padrão de domínio. A dependência e mútua exclusividade são previstas na notação, no entanto, graficamente não são representadas de maneira intuitiva, utilizando a mesma simbologia para os dois tipos de restrições. Além disso, não é possível expressar dependência ou exclusividade entre mais de duas características. Assim, a função do modelo de características de auxiliar o desenvolvedor por meio de facilidades visuais fica restrita, uma vez que vários dos requisitos para modelagem de variabilidade não são satisfeitos.

A análise descrita previamente é sumarizada na Tabela 2.1, na qual, percebe-se mais claramente as deficiências de cada notação, em relação à lista de requisitos para a modelagem de variabilidade apresentada anteriormente na seção 2.4. Um dos principais problemas encontrados, em relação aos requisitos estabelecidos, foi a falta de representação explícita do ponto de variação, sejam estes opcionais ou mandatórios. Em função dessa não-representação, relacionamentos entre os pontos de variação e suas variantes (representado na tabela pelo relacionamento Alternativo) também não são considerados pela maioria das notações analisadas.

Um outro grande problema encontrado foi a representação de dependência e mútua exclusividade. A maioria das notações prevê algum tipo de representação, porém, satisfazem apenas parcialmente aos requisitos, seja por não permitir tal representação no próprio modelo de características, seja por não permitir que tais dependências envolvam mais de duas características, reduzindo a expressividade do modelo.

**Tabela 2.1 – Quadro comparativo entre as notações analisadas**

| Notação / Requisito | FODA | FORM | Featu-RSEB | Svanhberg & Bosch | Riebisch | Cechticky | Czarnecki | Odyssey |
|---|---|---|---|---|---|---|---|---|
| **Elemento Invariante** | Sim | Sim | Sim | Sim | Sim | Sim | Sim | Sim |
| **Elemento Invariante Opcional e Mandatório** | Sim | Sim | Sim | Sim | Sim | Sim | Sim | Sim |
| **Ponto de Variação** | Não | Não | Sim | Não | Não | Sim | Sim | Não |
| **Ponto de Variação Opcional e Mandatório** | Não | Não | Sim | Não | Não | Sim | Sim | Não |
| **Variante** | Sim | Sim | Sim | Sim | Sim | Sim | Sim | Parcial-mente |
| **Variante Opcional/ Mandatória** | Sim | Sim | Não | Sim | Sim | Sim | Sim | Parcial-mente |
| **Dependên-cia** | Parcial-mente | Parcial-mente | Parcial-mente | Não | Sim | Parcial-mente | Não | Parcial-mente |
| **Mútua Exclusivi-dade** | Parcial-mente | Parcial-mente | Parcial-mente | Sim | Sim | Parcial-mente | Parcial-mente | Parcial-mente |
| **Relaciona-mento "Implemen-tado Por"** | Não | Sim | Não | Não | Não | Não | Não | Não |
| **Relaciona-mento "Alternati-vo"** | Não | Não | Sim | Não | Não | Parcial-mente | Sim | Não |

## 2.6 – CONSIDERAÇÕES FINAIS

A partir da análise das notações realizada, pôde-se perceber que é muito difícil encontrar uma notação que seja suficientemente abrangente, no sentido de representar as variabilidades em um modelo de características. Tal representação é importante no sentido de possibilitar uma melhor compreensão da família de sistemas que está sendo modelada, deixando explícitos os pontos passíveis de originar aplicações ou produtos específicos e diferenciados. Embora as notações selecionadas para a análise, bem como

a lista de requisitos apresentada, estejam longe de representar uma lista completa, a constatação das deficiências das notações serve como motivação para que se tente criar uma notação mais completa, que atenda ao menos os requisitos apresentados nesse capítulo.

Embora tenham sido analisadas somente notações que representam variabilidade em modelo de características, é importante que a variabilidade representada neste modelo seja propagada para outros modelos do ciclo de vida do software. Isso só é possível por meio de uma formalização dos conceitos representados.

Uma alternativa para se "formalizar" tal representação é a definição de um metamodelo. Das notações analisadas, apenas as notações de Cechticky e de Czarnecki apresentam um metamodelo. As outras notações não apresentam tal formalização, o que dificulta o seu entendimento por parte do usuário.

Face a estas questões, é apresentada, no próximo capítulo, uma proposta de notação que atende aos requisitos aqui explicitados. Além disso, tal notação tem sua semântica "formalizada" por meio de um metamodelo, com o intuito de possibilitar uma utilização mais consistente do modelo de características. Assim, é possível um aumento do potencial de reutilização em uma família de aplicações ou produtos.

# CAPÍTULO III
# ODYSSEY-FEX: UMA NOTAÇÃO PARA A MODELAGEM DE VARIABILIDADE EM REUTILIZAÇÃO DE SOFTWARE

## 3.1 – INTRODUÇÃO

No capítulo 2, foram apresentados os requisitos para a representação de variabilidades em uma LP ou ED, bem como as notações existentes com esse propósito, baseadas em modelos de características. De acordo com a análise apresentada, nota-se que tais requisitos não são completamente cobertos pelas notações em questão.

Um outro problema apresentado pelas notações analisadas foi o fato de a grande maioria delas não possuir sua semântica "formalmente" definida por meio de um metamodelo. A formalização dessa semântica permite, entre outras: a) a integração entre diferentes ambientes de modelagem de domínio baseados em modelos de características; b) a validação da consistência do modelo de características; e c) a facilidade de compreensão dos modelos e comunicação entre desenvolvedores.

Em vista disso, é apresentada, neste capítulo, uma proposta de notação mais abrangente para representação das variabilidades referentes às características do domínio, denominada *Odyssey-FEX*[4]. Tal notação, que é integrada ao ambiente Odyssey (ODYSSEY, 2005), define a semântica dos elementos que representam conceitos, funcionalidades e tecnologias utilizadas em um domínio, incluindo sua variabilidade, bem como os relacionamentos entre eles.

Junto com a notação, é proposto um metamodelo com regras de boa formação, cujo objetivo é formalizar a semântica e servir de auxílio na construção e entendimento do modelo de características desenvolvido.

O capítulo está organizado da seguinte maneira: a seção 3.2 apresenta uma descrição de parte do domínio de Telefonia Móvel, que será utilizado como exemplo ao longo do capítulo. Conceitos encontrados em tal descrição são formalizados por meio do metamodelo descrito na seção 3.3. A notação Odyssey-FEX é detalhada na seção 3.4. Por fim, considerações finais são apresentadas na seção 3.5.

[4] *Odyssey-FEX: **F**eature **EX**tended*

## 3.2 – DOMÍNIO DE TELEFONIA MÓVEL: UMA BREVE DESCRIÇÃO

Para auxiliar o leitor no entendimento dos conceitos presentes em um modelo de características, é descrito a seguir parte do domínio de Telefonia Móvel, que é utilizado como exemplo ao longo desta dissertação, definido a partir da análise de informações oferecidas pelos fabricantes de aparelho celular (LG, 2005), (NOKIA, 2005), (GRADIENTE, 2005).

O domínio de Telefonia Móvel abrange conceitos e funcionalidades que podem estar presentes em um software desenvolvido para um telefone celular. Atualmente, pode ser encontrada uma grande diversidade de telefones celulares, com software que apresenta variadas funcionalidades. No entanto, alguns conceitos e funcionalidades são intrínsecos e estão presentes em todos os tipos de software. Dentre esses podem ser citados: *Campainha*, que representa o toque do telefone, *Agenda*, onde são armazenados números de telefone, *Chamada Telefônica*, que é a funcionalidade de se efetuar uma ligação, e *Caixa Postal*, onde mensagens de voz são deixadas para o usuário. De acordo com as definições apresentadas no capítulo anterior, tais conceitos e funcionalidades são denominados **mandatórios**. Além disso, alguns conceitos, embora sejam mandatórios, e portanto, presentes em todos os tipos de software para telefone celular, apresentam alternativas, dando ao usuário a opção de escolha na hora da compra. A esses conceitos denomina-se **pontos de variação**, e a suas alternativas, denomina-se **variantes**. Como exemplo, tem-se o conceito de *Cores do Visor*. Este conceito representa um ponto de variação do tipo de visor do aparelho celular, que tem como variantes as opções "monocromático" ou "colorido".

Outros conceitos e funcionalidades são oferecidos no domínio de telefonia celular, mas podem ser encontrados em apenas alguns aparelhos, constituindo um diferencial. Tais conceitos são denominados **opcionais**. Dentre eles podem ser citados: *Câmera Fotográfica*, *Alerta Vibratório*, *Alarme*, *Acesso à Internet*, *Envio de mensagens de texto*, como e-mails e recados, *Jogos* e *Recebimento de toques musicais*, que representa a funcionalidade de alteração do tipo de campainha, reproduzindo-se um toque especial, como uma música conhecida, geralmente oferecido pela operadora de telefonia móvel.

Alguns desses conceitos também podem apresentar alternativas, isto é, os conceitos opcionais também podem ser pontos de variação e/ou variantes. Como

exemplo de pontos de variação, tem-se *Recebimento de toques musicais* e *Jogos*. Os toques musicais recebidos podem possuir variantes, i. e., podem ser do tipo *monofônicos*, *polifônicos* ou *MP3*. *Toques monofônicos* são toques musicais que a maioria dos aparelhos celulares pode reproduzir. O celular só precisa executar uma nota por vez para reproduzir um toque monofônico. Toques *polifônicos* são toques musicais que podem consistir em várias notas de uma vez, reproduzidas por meio de um alto-falante em vez de uma campainha. São toques mais elaborados, com som muito próximo das músicas reais. Toques do tipo *MP3* são toques que utilizam tecnologia e formato padrão para compactar uma seqüência de sons em um arquivo extremamente pequeno, preservando o nível original da qualidade do som ao ser reproduzido, i. e., músicas reais podem ser utilizadas como campainha de um celular (NOKIA, 2005). No caso dos jogos, existe uma variedade deles que podem estar presentes em um ou outro aparelho de telefone celular. Como exemplo, tem-se o *Car Racer*, que é um jogo de corrida de carros, o *Snake*, jogo cujo objetivo é guiar uma cobra para que seja alimentada, o jogo de Tênis, e outros.

Embora não visíveis em um aparelho de telefone celular, outros conceitos são importantes para o desenvolvimento de um software de telefonia, sendo então incluídos como parte do domínio de Telefonia Móvel. Como exemplo, tem-se a tecnologia *JAVA*, que implementa o jogo *Car Racer*, ou a tecnologia *WAP* (*Wireless Application Protocol*), que é um padrão internacional aberto para aplicativos que utilizam comunicação sem fio. O aplicativo principal baseado em WAP é o acesso à Internet de um dispositivo móvel, como um celular, que pode ser usado, por exemplo, em serviços de notícias, compra de ingressos, troca de e-mail e transações bancárias. Alguns aparelhos celulares oferecem a funcionalidade de transferência de dados para um computador pessoal (PC), sendo necessário para isso algum tipo de conexão. Esta conexão também constitui um ponto de variação, tendo como variantes os seguintes exemplos: a) *Bluetooth*, que é uma tecnologia que permite que uma conexão sem fio de curto alcance seja estabelecida com outro dispositivo compatível (celulares, laptops, câmeras digitais); b) *Infravermelho*, que faz a comunicação sem fio, por meio de sinais de luz que são captados por um sensor instalado no destinatário; e c) conexão via *USB* (*Universal Serial Bus*), que é uma interface de comunicação entre um computador e um dispositivo compatível, como um aparelho celular, câmera digital ou impressora. No entanto, uma conexão via USB necessita de um *Cabo de Dados* compatível para efetuar a transferência dos dados (NOKIA, 2005).

Adicionalmente, têm-se os *Planos da Operadora*, que influenciam na organização das funcionalidades de um telefone celular, determinando quais dessas estarão disponíveis para o usuário, de acordo com o contrato estabelecido com este.

Além de todos esses elementos do domínio, é importante destacar os *Serviços de Operadoras*, que embora possam ser classificados como parte do domínio de Operadoras de Telefonia Móvel, tem uma estreita relação com o domínio de Telefonia Móvel, aqui detalhado.

Uma vez descrito o domínio, é apresentado, a seguir, o metamodelo proposto para formalizar a semântica de um modelo de características que represente tais informações.

## 3.3 – FORMALIZAÇÃO DE CONCEITOS POR MEIO DE METAMODELOS

Um metamodelo define uma sintaxe abstrata para a construção de modelos, isto é, uma estrutura que norteia a representação de elementos em um modelo (MATULA, 2003). O objetivo de um metamodelo é fornecer uma maior compreensão sobre o modelo a ser construído e/ou interpretado.

A construção de metamodelos tem sido uma estratégia utilizada para sistematizar a semântica de modelos. A UML (OMG, 2004) estabelece um metamodelo que especifica a semântica para cada um dos modelos por ela definidos. Este metamodelo é especificado por meio do próprio modelo de classes da UML. No entanto, o modelo de características não é parte integrante do conjunto de modelos definidos pela UML, além de não existir uma especificação para tal definida pela OMG.

No conjunto das notações analisadas no Capítulo 2, apenas duas delas (CECHTICKY *et al.*, 2004), (CZARNECKI *et al.*, 2004) apresentam uma formalização via metamodelo. Além dessas, foi encontrado apenas um trabalho (GOMAA & SHIN, 2004) que apresenta uma abordagem de gerenciamento de variabilidade por meio das diversas visões de uma linha de produtos de software, representadas pelos múltiplos diagramas da UML. Os autores propõem a definição de um metamodelo, que é uma extensão do metamodelo da UML, que facilite a verificação de consistência entre essas múltiplas visões, nas quais está incluída a visão representada pelo modelo de características.

Dentre as motivações para a definição de um metamodelo, podemos listar as seguintes:

- Compreensão de modelos: Um modelo de software pode não ser de fácil compreensão à primeira vista, ocasionando interpretações incorretas da semântica ali representada. Tal fato pode se refletir na modelagem equivocada dos conceitos do domínio, ou na introdução de erros na modelagem. Assim, um metamodelo que formalize a semântica dos elementos presentes em um modelo é fundamental para que se possa documentar e disseminar o conhecimento acerca do domínio.
- Verificação de consistência inter-modelos: Devido à diversidade de modelos construídos durante o desenvolvimento de um software (modelo de classes, caso de uso etc.), correspondentes às diferentes visões de um software, um domínio, ou uma Linha de Produtos, e onde cada modelo tem sua própria semântica, a verificação de consistência entre esses modelos tende a ser mais complexa. Um metamodelo que possa padronizar a semântica dos modelos envolvidos é uma alternativa para lidar com tal complexidade (GOMAA & SHIN, 2004).
- Verificação de consistência intra-modelos: A verificação de consistência dentro de um modelo de software é igualmente importante. Assim, a existência de um metamodelo que estabeleça diretrizes que auxiliem na construção de um modelo é útil na diminuição do número de erros de modelagem.

Nesta seção, é proposto um metamodelo para a notação Odyssey-FEX, com o intuito de formalizar os conceitos existentes em um modelo de características, bem como alcançar as motivações citadas acima.

### 3.3.1 - O METAMODELO PARA A NOTAÇÃO ODYSSEY-FEX

O metamodelo desenvolvido para a notação Odyssey-FEX tem por objetivo reduzir a complexidade de construção do modelo de características, com a definição de diretrizes que auxiliem o desenvolvedor e/ou usuário da notação.

Tal metamodelo é representado por meio de diagramas de classes, onde cada classe representa um elemento presente no modelo de características, ou um relacionamento entre esses elementos. Ele é dividido em três pacotes: **a) Principal**, que representa a taxonomia das características, **b) Relacionamento**, que especifica

propriedades dos relacionamentos existentes entre as características em um modelo de características; e **c) Regras de Composição**, que especifica as regras de dependência e exclusividade entre as características de um modelo.

#### 3.3.1.1 - PACOTE PRINCIPAL

O pacote Principal (*Core*) do metamodelo da Odyssey-FEX é representado na Figura 3.14.

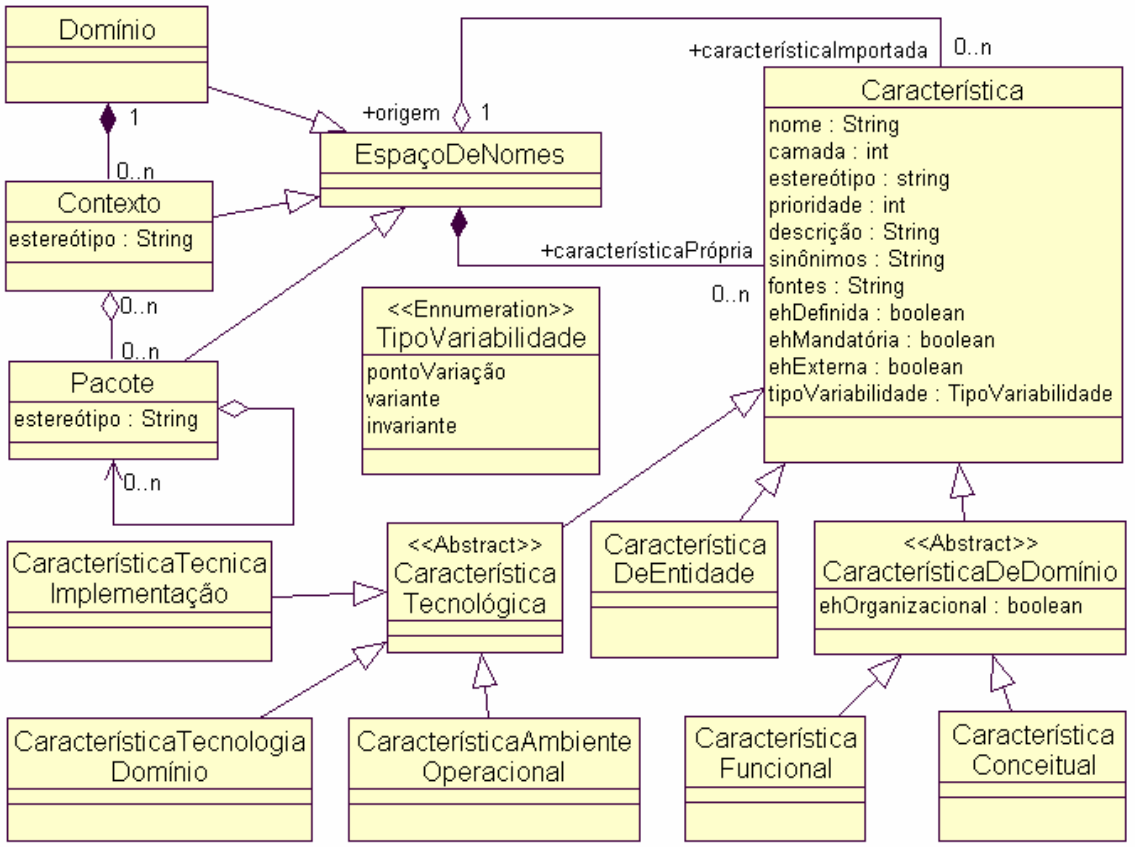



**Figura 3.14 - Metamodelo Odyssey-FEX: Principal**

O pacote Principal é composto de características e seus diversos tipos, bem como seus atributos. Na notação Odyssey-FEX, tem-se as *Características de Domínio*, *Características Tecnológicas*, e *Características de Entidade* como tipos de Característica. As Características de Domínio podem ser especializadas em *Característica Conceitual* e *Característica Funcional*, que representam, respectivamente, conceitos e funcionalidades do domínio que está sendo modelado. Características Tecnológicas podem ser especializadas em Característica de Tecnologia de Domínio, Característica de Ambiente Operacional e Característica de Técnica de Implementação e representam as tecnologias utilizadas no domínio. Tais características

são classificadas em camadas, como é detalhado na seção 3.4.1.1. Características de Entidade representam atores que interagem com o domínio em questão.

No domínio de Telefonia Móvel, as características *Agenda*, *Câmera* ou *Acesso à Internet* são exemplos de Características de Domínio, as características *WAP* e *JAVA* são exemplos de Características Tecnológicas, e *Usuário* é um exemplo de Característica de Entidade.

As propriedades essenciais de uma característica são definidas por meio dos atributos da classe *Característica,* como descrito a seguir:

- nome: atributo que define o nome da característica.
- camada: atributo que define a camada de tecnologia (Camada de Capacidades, Camada de Ambiente Operacional, Camada de Tecnologia de Domínio ou Camada de Técnica de Implementação) à qual a característica pertence. Tal atributo é relevante na modelagem de características, uma vez que existe uma ordem entre as camadas de tecnologia, e esta deve ser respeitada no relacionamento "ImplementadoPor" (ver seção 3.3.1.2). Características de Domínio e de Entidade pertencem à primeira camada, Características de Ambiente Operacional pertencem à segunda camada, Características de Tecnologia de Domínio pertencem à terceira camada e Características de Técnica de Implementação pertencem à quarta camada de tecnologia do domínio.
- ehDefinida: atributo booleano que estabelece a condição de uma característica ser ou não definida naquele domínio. Uma característica não-definida é uma característica já identificada no domínio, porém ainda não definida por meio de outros artefatos.
- ehMandatória: o atributo booleano que define se a característica é ou não obrigatória em um domínio.
- ehExterna: atributo que evidencia uma característica pertencente a um domínio diferente do que está sendo modelado.
- tipoVariabilidade: atributo que define o tipo de variabilidade: Variante, Invariante ou Ponto de Variação, que é representado pela lista enumerada *TipoVariabilidade.*

As Características de Domínio possuem ainda o seguinte atributo:

- ehOrganizacional; atributo booleano que estabelece se uma característica está organizando um domínio. Características organizacionais não possuem ligações concretas com o uso real do domínio, e são utilizadas com o intuito de facilitar o entendimento deste.

Além disso, as características pertencem a um EspaçoDeNomes (*Namespace*). Um EspaçoDeNomes é um elemento do modelo que contém um conjunto de elementos denomináveis, onde elementos são distinguíveis entre si pelo nome (OMG, 2004). A classe *EspaçoDeNomes* tem como especialização as classes *Pacotes*, *Contextos* e *Domínios*. Pacotes são elementos usados para agrupar outros elementos do modelo, no caso, características, fornecendo um EspaçoDeNomes para os elementos agrupados (OMG, 2004). Contextos representam um contexto da organização e podem ser organizados em diagramas, que têm por objetivo situar o domínio em relação ao seu escopo, limites, relacionamentos com outros domínios de aplicação e principais atores envolvidos (BRAGA, 2000). Domínios descrevem uma coleção de problemas reais e/ou uma coleção de aplicações que compartilham características comuns, de acordo com o consenso de uma comunidade de software (ARANGO & PRIETO-DIAZ, 1991).

Se o *EspaçoDeNomes* se refere a um outro domínio, então a característica é Externa. Convém ressaltar que Contextos e Pacotes são mecanismos de organização de um conjunto de características, que diferem em seu nível de abrangência. Domínios abrangem Contextos, que por sua vez abrangem Pacotes.

#### 3.3.1.2 - PACOTE RELACIONAMENTOS

O pacote Relacionamentos (*Relationships*) do metamodelo da Odyssey-FEX é representado na Figura 3.15. Nele, é possível observar a classe abstrata *Relacionamento*, e suas especializações: *ImplementadoPor, LigaçãoDeComunicação, Alternativo, Associação* e *Generalização.*

A classe *ImplementadoPor* representa o relacionamento existente entre duas características, que denota a tecnologia utilizada para implementar determinada característica. As associações entre as classes *ImplementadoPor* e *Característica* denotam essa relação, onde uma característica é a origem e a outra é o destino do relacionamento. Como é apresentado nas regras de boa formação do metamodelo, mais adiante, as características envolvidas em tal relacionamento devem pertencer a camadas de tecnologia diferentes. Como exemplo, tem-se, no domínio de Telefonia Móvel, o

relacionamento entre as características *Acesso à Internet* e *WAP*, uma vez que a primeira é implementada pela tecnologia representada pela segunda.

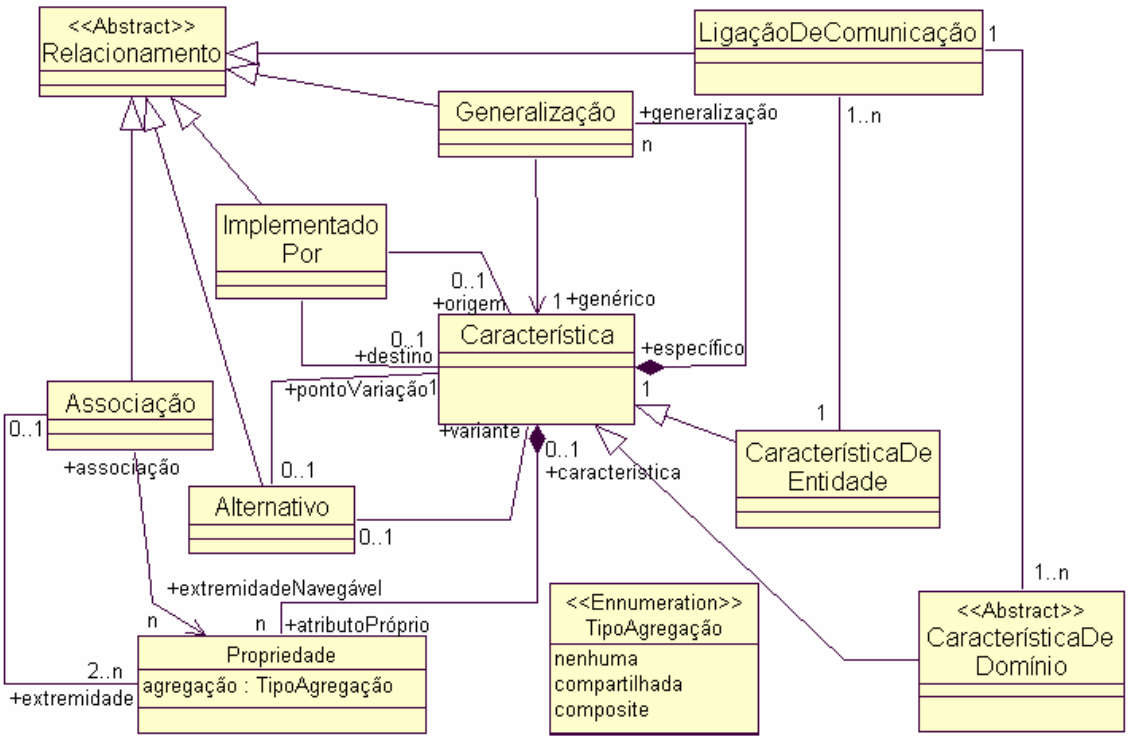



**Figura 3.15 - Metamodelo Odyssey-FEX: Relacionamentos**

A classe *LigaçãoDeComunicação* representa o relacionamento existente entre as instâncias das classes *CaracterísticaDeDomínio* e *CaracterísticaDeEntidade*, que são especialização da classe *Característica*, como apresentado na seção 3.3.1.1. Tal relacionamento cumpre o mesmo papel do relacionamento de associação entre atores e casos de uso na UML (OMG, 2004) . Como exemplo, tem-se o relacionamento entre um Usuário e o Telefone Celular.

A classe *Alternativo* representa o relacionamento existente entre duas características que atuem, respectivamente, como ponto de variação e variante. Essa relação é expressa por meio das associações existentes entre as classes *Alternativo* e *Característica*. Como exemplo, tem-se o relacionamento existente entre a funcionalidade *Recebimento de toques musicais*, e suas variações *Toque Monofônico*, *Toque Polifônico* e *MP3*.

Os relacionamentos de *Generalização* e *Associação*, denotados pelas classes homônimas no metamodelo, estão baseadas na especificação da UML 2.0 (OMG, 2004), e utilizam um subconjunto das funcionalidades propostas nesta especificação. Os

relacionamentos de Composição e Agregação são identificados por meio do atributo tipoAgregação da classe *Propriedades*, que, segundo a especificação da UML, pode atuar como extremidade da associação, que está associado a uma instância da classe Caraterística. Tal atributo assume o valor "*nenhuma*" se for uma associação, "*compartilhada*" quando é agregação ou "*composite*" quando é composição, conforme listado na lista enumerada *TipoAgregação*.

O relacionamento de herança é representado pelas associações entre as classes *Generalização* e *Característica*, indicando que uma das características envolvidas é a mais genérica e as outras são mais específicas.

#### 3.3.1.3 - PACOTE REGRAS DE COMPOSIÇÃO

O pacote Regras de Composição (*Composition Rules*) do metamodelo da notação Odyssey-FEX contém classes que definem a semântica das regras de dependência e mútua exclusividade entre características.

Como mencionado anteriormente no Capítulo 2, um dos requisitos para uma notação que represente variabilidade é a necessidade de representação de dependência e exclusividade para quaisquer conjuntos de elementos do modelo.

No método FODA (KANG *et al*., 1990), Kang propõe a representação de tais conceitos por meio do uso de Regras de Composição (*Composition Rules*). As Regras de Composição são regras que definem restrições existentes entre características e que não são explicitamente expressas no modelo para não dificultar sua visualização. Tais regras são expressas por meio de cláusulas do tipo "é requerido", que indicam a dependência entre duas ou mais características, e cláusulas do tipo "mutuamente exclusivo com", que indicam que duas ou mais características não devem ser selecionadas em conjunto em um mesmo produto ou aplicação. No entanto, somente com a representação proposta para os conceitos de dependência e exclusividade não é possível representar toda a semântica necessária em um modelo de características. Por exemplo, considere as regras expressas na Figura 3.16, tomando como base um domínio de Telefonia Móvel. Tais regras são elaboradas assumindo a descrição do domínio de Telefonia Móvel apresentada na seção 3.2.

Na Figura 3.16, podem ser observadas duas regras de composição. A primeira regra representa o fato de que aparelhos celulares podem possuir a funcionalidade de transferência de dados para um PC por meio de uma conexão que pode ser do tipo Infravermelho, ou por *Bluetooth* ou via USB. A segunda regra indica que, para que a

transferência seja feita via USB, um cabo de dados é necessário. A primeira regra poderia ser representada nas outras notações de maneira equivalente, se fosse desmembrada em uma das três regras: "Transferência de Dados para PC **requer** *Bluetooth*", "Transferência de Dados para PC **requer** USB" ou "Transferência de Dados para PC **requer** Infravermelho". No entanto, esse desmembramento não representaria a segunda regra de forma correta, pois "Transferência de Dados para PC **requer** Cabo de Dados" e "USB **requer** Cabo de Dados" tem a semântica diferente de (Transferência de Dados para PC AND USB) **requer** (Cabo de Dados), uma vez que esta última regra indica que, somente quando as características Transferência de Dados para PC e USB forem incluídas em conjunto na aplicação, a característica Cabo de Dados será requerida

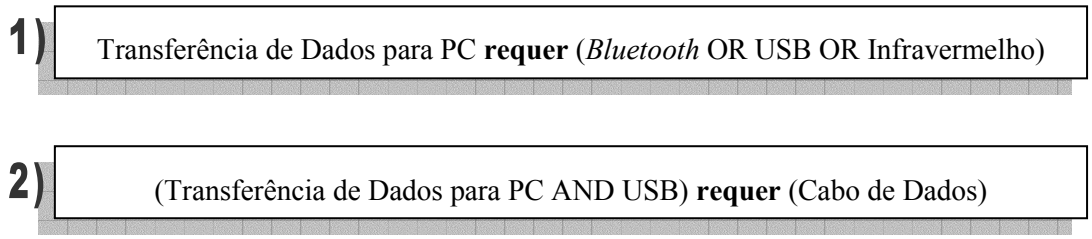



**Figura 3.16 – Exemplos de Regras de Composição no domínio de Telefonia Móvel**

Assim, com base na necessidade de expressão de regras como as descritas previamente, e nas deficiências apresentadas pelas notações analisadas na literatura, é proposta a idéia de Regras de Composição Complexas, com suas expressões combinadas por meio de operadores booleanos, no intuito de permitir uma representação mais expressiva da semântica de um modelo de características. Além disso, as Regras de Composição Complexas constituem uma alternativa que satisfaz o requisito de representar as dependências e exclusividade para quaisquer conjuntos de características, o que não era possível na maioria das notações analisadas.

Na notação Odyssey-FEX, uma Regra de Composição tem a seguinte estrutura:

Antecedente + **palavra-chave** + Conseqüente

A Regra de Composição que expressa dependência entre características é denominada Regra de Composição Inclusiva, enquanto a regra que expressa mútua exclusividade é denominada Regra de Composição Exclusiva. A palavra-chave apresentada na estrutura define o tipo de Regras de Composição: Regras Inclusivas são denotadas pela palavra-chave "requer", enquanto que Regras Exclusivas são denotadas pela palavra-chave "exclui".

O antecedente e o conseqüente de uma regra são expressões, que podem ser literais ou booleanas, e denotam uma característica do domínio ou uma combinação entre estas. Esta combinação de expressões tem por finalidade intensificar a expressividade de uma regra de composição. O pacote Regras de Composição pode ser observado na Figura 3.17.

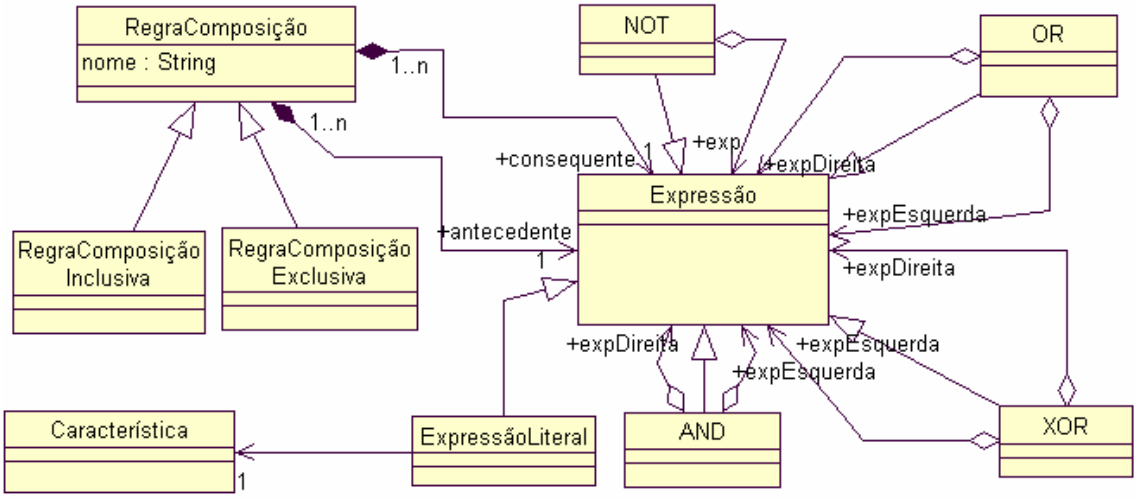



**Figura 3.17 – Metamodelo Odyssey-FEX: Regras de Composição**

Uma regra de composição é representada pela classe *RegraComposição* que tem como especializações as classes *RegraComposiçãoInclusiva* e *RegraComposiçãoExclusiva.* e cujos atributos são:

- nome: define o nome pelo qual uma Regra de Composição pode ser identificada e referenciada. Este nome aparece nas características envolvidas em uma regra no modelo de características.
- antecedente: representado por uma das associações entre as classes *RegraComposição* e *Expressão*, tal atributo denota a expressão que forma o antecedente da regra.
- conseqüente: representado pela outra associação entre *RegraComposição* e *Expressão*, tal atributo denota a expressão que forma o conseqüente da regra.

A classe *Expressão* é abstrata, tendo como subclasses as classes *AND*, *NOT*, *XOR*, *OR* e *ExpressãoLiteral*, segundo o padrão de projeto *Interpreter* (GAMMA *et al.*, 1995). Expressões representam uma característica ou combinações de características de um domínio. As classes *AND*, *XOR* e *OR* são ainda agregações de outras duas expressões, denominadas neste metamodelo expEsquerda e expDireita. A classe NOT

contém uma expressão, aqui denominada exp. A classe *ExpressãoLiteral* contém um atributo do tipo *Característica*, representado pela associação entre essas duas classes, significando que uma Característica é a expressão mais elementar de uma Regra de Composição.

Retomando o domínio de Telefonia Móvel, tome-se, como exemplo, a segunda regra da Figura 3.16: "(Transferência de Dados para PC AND USB) **requer** (Cabo de Dados)". Neste caso, a regra em questão é uma instância da classe *RegraComposiçãoInclusiva*, onde seu antecedente é uma instância da classe AND e seu conseqüente é uma instância da classe *ExpressãoLiteral*, e portanto instâncias da classe abstrata *Expressão*. Além disso, a instância da classe AND é composta por duas instâncias de ExpressãoLiteral : Transferência de Dados para PC e USB.

### 3.3.2 - REGRAS DE BOA-FORMAÇÃO DO METAMODELO ODYSSEY-FEX

O metamodelo da notação Odyssey-FEX, a exemplo da especificação da UML, possui regras de boa formação, que direcionam a construção e a verificação de consistência do modelo de características. Cada classe descrita no metamodelo tem o seu conjunto de restrições e propriedades, que dão origem a regras de boa formação do modelo. A seguir, são descritas na Tabela 3.2, Tabela 3.3 e Tabela 3.4 as restrições de cada pacote.

Algumas restrições descritas nas tabelas serão detalhadas, a fim de assegurar uma maior compreensão da notação. Outras propriedades, bem como a descrição completa, notação e exemplos de cada classe do metamodelo podem ser encontradas em (OLIVEIRA *et al.*, 2005a).

**Restrições da Tabela 3.2**

- **Restrição 5**: *Características que sejam Não-Definidas não podem ser Organizacionais e vice-versa*.

De acordo com a definição, que é detalhada na seção 5.2.2 , uma característica Organizacional não tem ligação concreta com o uso real do domínio, portanto não é expressa em outros artefatos. Uma vez que uma característica Não-definida pode dar origem a novos artefatos, não faz sentido existir uma característica organizacional não-definida.

- **Restrição 8:** Características *que tenham classificação de Variantes e Mandatórias devem ter um relacionamento do tipo Alternativo com outra característica classificada como Ponto de Variação NECESSARIAMENTE mandatória.*

Uma variante está necessariamente ligada a um ponto de variação. Uma vez que tal variante seja mandatória, ela deve obrigatoriamente existir na instanciação da aplicação. Se tal variante pertencer a um ponto de variação opcional, este ponto de variação pode não ser selecionado na instanciação de uma aplicação. Assim, uma inconsistência será gerada, uma vez que, de acordo com a Restrição 7, não pode haver variantes desassociadas de um ponto de variação.

**Tabela 3.2 - Restrições do metamodelo Odyssey-FEX: Pacote *Principal***

| **Pacote Principal** | |
|---|---|
| **Classe** | **Restrições** |
| **EspaçoDeNomes** | 1. Todos os membros de um EspaçoDeNomes são distinguíveis dentro daquele EspaçoDeNomes. (OMG, 2004) |
| **Domínio** | 2. Domínios contêm Contextos |
| **Contexto** | 3. Contextos pertencem obrigatoriamente a um Domínio.<br>4. Contextos contêm Pacotes. |
| **Característica** | 5. Características que sejam Não-Definidas não podem ser Organizacionais e vice-versa.<br>6. Características que tenham a classificação de Variantes devem obrigatoriamente estar associadas a UMA e SOMENTE UMA característica que tenha classificação de Ponto de Variação.<br>7. Não deve haver características que tenham a classificação de Variantes desassociadas de um Ponto de Variação<br>8. Características que tenham classificação de Variantes e Mandatórias devem ter um relacionamento do tipo Alternativo com outra característica classificada como Ponto de Variação NECESSARIAMENTE mandatória.<br>9. Características que tenham classificação de Variantes e Opcionais podem ter um relacionamento do tipo Alternativo com outra característica classificada como Ponto de Variação que seja mandatória ou opcional. Neste caso, a obrigatoriedade do ponto de variação indica que pelo menos uma das variantes ligadas a ele deve ser selecionada na aplicação.<br>10. Características que tenham classificação de Ponto de Variação e Opcionais devem ter um relacionamento do tipo Alternativo com outras características classificadas como Variantes NECESSARIAMENTE opcionais. |
| **CaracterísticaDe Entidade** | 11. Características de Entidade se relacionam somente via Ligação de Comunicação com as Características de Domínio. |

- **Restrição 10:** *Características que tenham classificação de Ponto de Variação e Opcionais devem ter um relacionamento do tipo Alternativo com outras características classificadas como Variantes NECESSARIAMENTE opcionais*.

De modo semelhante à Restrição 8, se um ponto de variação for opcional, este pode não ser selecionado na instanciação da aplicação, e portanto não pode ter subordinadas a ele variantes que tenham de ser selecionadas obrigatoriamente.

As restrições do pacote Relacionamentos são apresentadas na Tabela 3.3 a seguir.

**Tabela 3.3 - Restrições do metamodelo Odyssey-FEX: Pacote *Relacionamentos***

| **Pacote Relacionamentos** | |
|---|---|
| **Classe** | **Restrições** |
| **Alternativo** | 1. O relacionamento Alternativo só poderá ocorrer entre características do tipo Variante e Ponto de Variação.<br>2. O relacionamento Alternativo só poderá ter características do tipo Ponto de Variação como origem. |
| **LigaçãoDeComunicação** | 3. O relacionamento Ligação de Comunicação só poderá ocorrer entre Características de Domínio e Características de Entidade. |
| **ImplementadoPor** | 4. No relacionamento Implementado Por, a característica de origem deve pertencer a camadas superiores à da característica de destino, isto é, ter valor do atributo camada menor do que o da característica de destino. |
| **Herança** | 5. Não deve haver herança entre características de tipos diferentes.<br>6. Não deve haver herança circular. |
| **Associação** | 7. Se o relacionamento de Associação for uma Composição, então este não pode ocorrer entre características quando a característica que representa o todo é opcional e a característica que representa a parte é mandatória. |
| **Propriedade** | 8. A multiplicidade em uma extremidade que seja Composição não deve ter limite superior maior que 1.<br>9. O relacionamento de Composição é determinado pelo valor "verdadeiro" para o atributo ehComposição. |

**Restrições da Tabela 3.3**

- **Restrição 4:** *No relacionamento Implementado Por, a característica de origem deve pertencer a camadas superiores à da característica de destino, isto é, ter valor do atributo camada menor do que o da característica de destino*.

De acordo com (LEE *et al.*, 2002), o relacionamento Implementado Por é usado quando uma característica é necessária para a implementação de outra. Tal relação de implementação entre características representa uma diferença de abstração entre estas, o que é caracterizado pela organização em diferentes camadas. Portanto, a característica que será implementada, e que representa a origem do relacionamento, deve pertencer a

camadas superiores que a característica que representa a tecnologia usada na implementação, destino do relacionamento.

- **Restrição 7:** *Se o relacionamento de Associação for uma Composição, então este não pode ocorrer entre características quando a característica que representa o todo é opcional e a característica que representa a parte é mandatória.*

De acordo com a UML (OMG, 2005), somente associações binárias podem ser composição. Além disso, quando um relacionamento de composição é removido, ambas as partes envolvidas são removidas. Isto significa que, em um relacionamento entre características, se uma característica participante de uma composição deixar de existir na instanciação da aplicação - o que pode ocorrer se a característica for opcional - a outra característica envolvida deve também deixar de existir. Desse modo, só faz sentido haver composição entre características em que aquela que representa o "todo" seja mandatória, o que garante que a "parte" não existirá sem o "todo" na instanciação da aplicação, ou duas características opcionais, em que, uma vez que uma delas não seja selecionada, a outra poderá também deixar de ser selecionada na instanciação da aplicação.

**Restrições da Tabela 3.4**

- **Restrição 4**: *Em uma Regra de Composição Inclusiva, o conseqüente só poderá ser opcional se o antecedente for opcional*

Uma vez que uma característica atua como conseqüente de uma regra inclusiva, ela é uma característica requerida. Se esta característica requerida for opcional, ela pode não existir na instanciação da aplicação, gerando uma inconsistência caso a característica que representa o antecedente desta regra seja selecionada na instanciação da aplicação. Assim, para que o conseqüente de uma regra seja opcional, o antecedente desta regra deve obrigatoriamente ser opcional, possibilitando que este não seja selecionado, caso a característica por ele requerida também não o seja.

A partir da descrição do metamodelo da Odyssey-FEX, é apresentada a seguir a notação proposta para representar os conceitos descritos, de maneira a contemplar de forma mais abrangente a representação de variabilidades em um Domínio, ou uma Linha de Produtos de Software.

**Tabela 3.4 - Restrições do metamodelo Odyssey-FEX: Pacote *Regras de Composição***

| Pacote Regras de Composição | |
|---|---|
| **Classe** | **Restrição** |
| **RegraComposição** | 1. Características dependentes entre si não podem ser mutuamente exclusivas, e vice versa.<br>2. Uma regra de composição não pode ter antecedente definido e conseqüente nulo ou vice versa.<br>3. Regras de Composição não são bidirecionais. Por exemplo, se uma característica A requer a característica B, e a característica B requer a característica A, devem existir duas Regras de Composição. |
| **RegraComposiçãoInclusiva** | 4. Em uma Regra de Composição Inclusiva, o conseqüente só poderá ser opcional se o antecedente for opcional. |
| **RegraComposiçãoExclusiva** | 5. Uma Regra de Composição Exclusiva só pode envolver características opcionais. |
| **AND** | 6. Uma expressão do tipo AND é composta de duas expressões, denominadas expEsquerda e expDireita, que são instancias de qualquer das subclasses da classe Expressão. |
| **OR** | 7. Uma expressão do tipo OR é composta de duas expressões, denominadas expEsquerda e expDireita, que são instancias de qualquer das subclasses da classe Expressão. |
| **XOR** | 8. Uma expressão do tipo XOR é composta de duas expressões, denominadas expEsquerda e expDireita, que são instancias de qualquer das subclasses da classe Expressão. |
| **NOT** | 9. Uma expressão do tipo NOT é composta de uma expressão, denominada exp, que é instancia de qualquer das subclasses da classe Expressão. |
| **ExpressãoLiteral** | 10. Uma expressão do tipo ExpressãoLiteral contém um único elemento, que é instância da classe Característica. |

## 3.4 - NOTAÇÃO ODYSSEY-FEX

A notação Odyssey-FEX é proposta no intuito de representar os conceitos formalizados pelo metamodelo descrito na seção anterior, de acordo com os requisitos para modelagem de variabilidades, apresentados no Capítulo 2. Para tanto, assim como diversas outras notações propostas na literatura para a modelagem de variabilidades, a notação tem como base o trabalho de Kang (KANG *et al.*, 1990). No entanto, a notação Odyssey-FEX incorpora ainda elementos do modelo de características do ambiente Odyssey, definidos na proposta de Miler (2000).

Das propostas do trabalho de Miler, a notação Odyssey-FEX mantém o detalhamento das características por meio de padrões de domínio, os relacionamentos da

UML e a representação visual por meio de ícones que definem os diversos tipos de característica. No entanto, tais tipos foram redefinidos, conforme apresentado na seção 3.4.1.1. A seguir, é apresentada a classificação das características, segundo a notação Odyssey-FEX.

### 3.4.1 - CLASSIFICAÇÃO DAS CARACTERÍSTICAS NA NOTAÇÃO ODYSSEY-FEX

Na notação Odyssey-FEX as características podem ser classificadas quanto à sua categoria, variabilidade e opcionalidade, além de possuírem qualificações adicionais que podem fortalecer a semântica do modelo a ser construído.

#### 3.4.1.1 – CLASSIFICAÇÃO QUANTO À CATEGORIA

Os tipos de características definidos na Odyssey-FEX estão relacionados às diferentes fases de desenvolvimento do software. Assim, as características de Entidade e de Domínio, que podem ser subdivididas em conceituais e funcionais, estão relacionadas à fase de análise do domínio, e as características tecnológicas, são relativas à fase de projeto do domínio. A descrição de cada tipo de característica e seu respectivo ícone, de acordo com a Categoria, podem ser observados na Tabela 3.5.

As características funcionais e conceituais estão incluídas na fase de análise do domínio. No nível de análise, encontram-se as funcionalidades e os conceitos de alto nível, que serão primeiramente utilizados na aplicação. Estas funcionalidades são representadas na notação Odyssey-FEX pelas características funcionais. Tais características estão ligadas aos casos de uso existentes no domínio. Por sua vez, as características conceituais representam os conceitos que correspondem aos elementos de um determinado domínio modelado na notação Odyssey-FEX. A Figura 3.18 mostra as características *Envio de Mensagens de Texto* e *Agenda*, respectivamente, como exemplo de característica funcional e conceitual, pertencentes ao domínio de Telefonia Móvel descrito na seção 3.2.

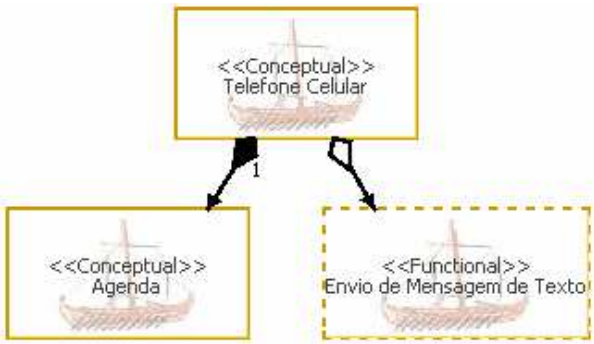
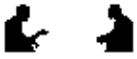



**Figura 3.18 – Características conceituais e funcionais da notação Odyssey-FEX**

**Tabela 3.5 – Tipos de Características na notação Odyssey-FEX**

| Ícone | Tipo de Característica | |
|---|---|---|
|  | **Características de Domínio** – Características intimamente ligadas à essência do domínio. Representam as funcionalidades e/ou os conceitos do modelo e correspondem a casos de uso e componentes estruturais concretos. | **Características de Análise** |
|  | **Características de Entidade** – São os atores do modelo. Entidades do mundo real que atuam sobre o domínio. Podem, por exemplo, expor a necessidade de uma interface com o usuário ou de procedimentos de controle. | |
|  | **Características de Ambiente Operacional -** Características que representam atributos de um ambiente que uma aplicação do domínio pode usar e operar. Ex: tipo de terminal, sistemas operacionais, bibliotecas etc. | **Características de Projeto (Tecnológicas)** |
|  | **Características de Tecnologia de Domínio -** Características que representam tecnologias utilizadas para modelar ou implementar questões específicas de um domínio. Ex: métodos de navegação em um domínio de aviões. | |
|  | **Características de Técnicas de Implementação –** Características que representam tecnologias utilizadas para implementar outras características, podendo ser compartilhadas por diversos domínios. Ex: técnicas de sincronização. | |

A classificação das características tecnológicas em camadas, proposta originalmente por Lee, Kang e Lee (LEE *et al.*, 2002) e Kang, Lee e Donahue (KANG *et al.,* 2002) foi mantida na notação Odyssey-FEX, com o objetivo de possibilitar a representação mais completa das características de um domínio, permitindo que aspectos tecnológicos sejam também considerados.

As tecnologias JAVA e WAP são exemplos de características tecnológicas no domínio de Telefonia Móvel, como apresentado na Figura 3.19.

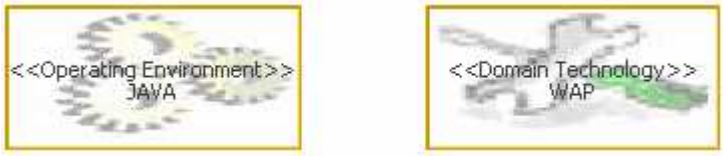



**Figura 3.19 - Características tecnológicas da notação Odyssey-FEX**

Na notação Odyssey-FEX, a Camada de Capacidades compreende as características de Domínio e as características de Entidade. As demais camadas correspondem, respectivamente, às suas características homônimas.

Os tipos de características descritos acima são mutuamente excludentes. Uma característica não pode pertencer simultaneamente a mais de uma categoria.

#### 3.4.1.2 – CLASSIFICAÇÃO QUANTO À VARIABILIDADE

De acordo com os conceitos de variabilidade apresentados no capítulo 2, as características na notação Odyssey-FEX possuem a classificação de **Ponto de Variação**, **Variantes** ou **Invariantes**.

A classificação de uma característica entre Ponto de Variação e Variante é abordada em outros trabalhos da literatura, como em (KANG *et al*., 1990), (GRISS *et al*., 1998), (SVAHNBERG *et al*., 2002) e (RIEBISCH *et al*., 2002). No entanto, as características "fixas", ou não-configuráveis, não recebem nenhum tipo de nomenclatura especial na literatura pesquisada. A nomenclatura "Invariante" foi introduzida pela notação Odyssey-FEX para diferenciar tais características das demais.

Uma característica não pode receber simultaneamente dois tipos diferentes de classificação quanto à variabilidade. No entanto, essa classificação é ortogonal em relação aos outros tipos de classificação apresentados na Tabela 3.5. Desse modo, uma característica de qualquer categoria descrita na Tabela 3.5 pode receber uma classificação de variabilidade. Por exemplo, uma característica de Domínio pode ser Variante, ou uma Característica de Ambiente Operacional pode ser um Ponto de Variação. A Figura 3.20 apresenta a característica de Domínio *Recebimento de Toques Musicais* que também é um ponto de variação, e outras características de Domínio *Toque Polifônico* e *Toque Monofônico*, que por sua vez são também variantes.

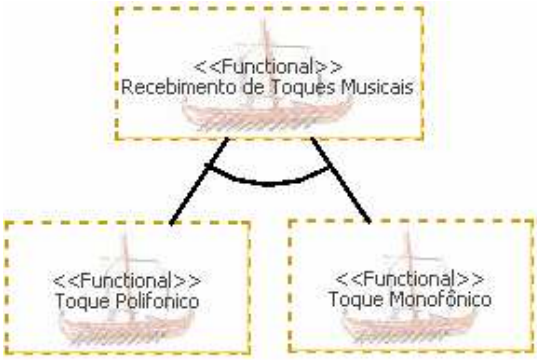



**Figura 3.20 – Classificação ortogonal Categoria x Variabilidade**

#### 3.4.1.3 – CLASSIFICAÇÃO QUANTO À OPCIONALIDADE

Como mencionado no capítulo 2, a variabilidade em um Domínio ou Linha de Produtos de Software é representada também por meio da opcionalidade, que envolve a classificação de elementos em mandatórios ou opcionais. A classificação das características quanto à sua opcionalidade indica justamente a obrigatoriedade ou não da presença de um determinado elemento nas aplicações ou produtos a serem desenvolvidos. Vale ressaltar que a opcionalidade é referente ao domínio como um todo. Características que são opcionais em relação ao domínio, mas que por ventura venham a ser mandatórias em relação a outras características selecionadas, devem expressar essa informação por meio de Regras de Composição, que são detalhadas na seção 3.4.4

Na notação Odyssey-FEX, as características opcionais são evidenciadas no modelo por um contorno pontilhado, conforme apresentado na Figura 3.21.


**Figura 3.21 - Representação das características opcionais na Notação Odyssey-FEX**

A classificação quanto à opcionalidade, também, é ortogonal às outras classificações. Isso significa que Características de Análise (Características de Domínio e Entidade) podem ser pontos de variação opcionais, ou variantes mandatórias, bem como Características de Projeto (Tecnológicas) podem ser pontos de variação, ou variantes, invariantes, sejam estes opcionais ou mandatórios. A Figura 3. 22 apresenta as possíveis combinações na classificação das características da notação Odyssey-FEX.

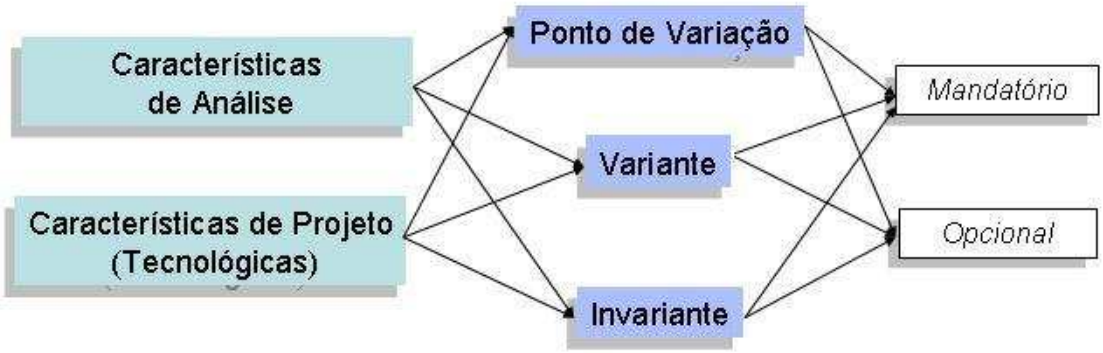



**Figura 3. 22 - Classificação ortogonal das características na notação Odyssey-FEX**

Como exemplo, no domínio de Telefonia Móvel, *Jogos* é uma característica de Análise, uma vez que representa um conceito, e também é um ponto de variação, uma vez que possui as variantes *Car Racer* e *Snake*. No caso, esse ponto de variação é opcional, uma vez que não tem a obrigatoriedade de ser selecionado na instanciação de uma aplicação.

#### 3.4.1.4 - PROPRIEDADES ADICIONAIS DAS CARACTERÍSTICAS

Na notação proposta por Miler, as características de análise podem ser classificadas com uma maior variedade de tipos em relação à taxonomia proposta pela notação Odyssey-FEX. No entanto, após uma análise detalhada, percebeu-se que tais tipos não seriam necessariamente excludentes entre si, como originalmente proposto. Em vista disso, esses tipos adicionais foram incorporados à notação Odyssey-FEX como propriedades das características já definidas. Tais peculiaridades são apresentadas na Tabela 3.6.

**Tabela 3.6 - Propriedades das características na notação Odyssey-FEX**

| Representação | Descrição |
|---|---|
| <<*from Another Domain* >> | **Externas** – Representam a ligação com outros domínios. Podem ou não ser refinadas pelo modelo. Mostram a fronteira do domínio e como ela se comporta. |
|  | **Não-Definidas** – Características de um domínio já identificadas, porém ainda não definidas por meio de casos de uso, modelos conceituais, ou características tecnológicas. |
|  | **Organizacionais** – Características do modelo que têm apenas o intuito de facilitar o entendimento ou organizar o domínio. Não possuem ligações concretas com o uso real do domínio. |

A classificação "**Externa**" pode ser atribuída a qualquer uma das categorias que classificam as características na notação Odyssey-FEX, uma vez que em domínios

externos pode haver tanto características de análise quanto tecnológicas. A classificação "**Não-Definida**" se aplica a todas as categorias de características, exceto à característica de Entidade, uma vez que esse tipo de característica não é detalhado nos modelos de mais baixo nível de abstração.

A classificação "**Organizacional**" só se aplica às características de Domínio, uma vez que esse tipo de característica representa os conceitos e funcionalidades do domínio, cujo entendimento precisa ser organizado. Uma vez que, como definido no metamodelo, as características Organizacionais não possuem ligações concretas com o uso real do domínio, entende-se que as características tecnológicas não devem receber a classificação de Organizacionais, devido ao fato de que estas representam uma tecnologia utilizada no domínio, e não uma organização do mesmo.

A Figura 3.23 apresenta um exemplo das propriedades descritas acima: *Planos de Telefonia* é uma característica Organizacional, *Snake* é uma característica Não-Definida, enquanto *Serviços da Operadora* é uma característica Externa, pertencente ao domínio de Operadoras de Telefonia Móvel.

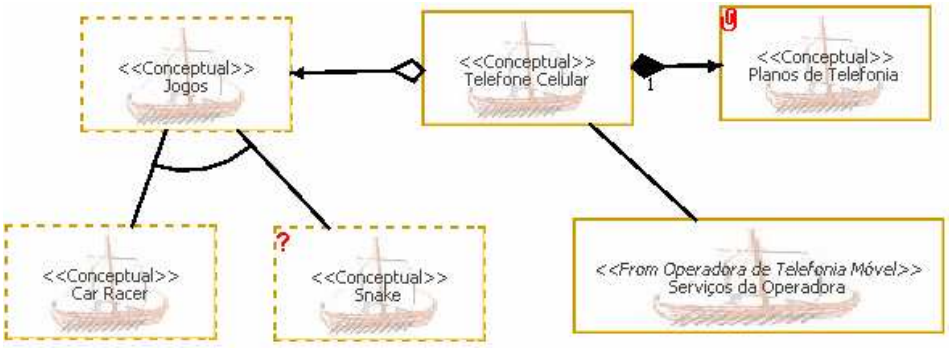



**Figura 3.23 – Propriedades das características na notação Odyssey-FEX**

### 3.4.2 – RELACIONAMENTOS

A notação Odyssey-FEX valoriza a semântica dos relacionamentos em um modelo de características, oferecendo uma maior capacidade de representação e expressão. Relacionamentos da UML (OMG, 2005), cuja incorporação à notação de modelagem de características foi anteriormente proposta por Miler, foram mantidos na notação Odyssey-FEX. Além desses, relacionamentos que deixam explícita a representação da variabilidade no modelo de características foram propostos, baseados em trabalhos da literatura (KANG *et al*., 1990), (GRISS *et al*., 1998), (KANG *et al*., 2002). A representação e descrição de tais relacionamentos podem ser visualizados na Tabela 3.7. A descrição dos relacionamentos da UML foram adaptadas de (MILER, 2000).

**Tabela 3.7 - Relacionamentos da notação Odyssey-FEX**

| Representação | Descrição | |
|---|---|---|
|  | **Composição** – Relacionamento em que uma característica é composta de várias outras. Denota relação na qual uma característica é parte fundamental de outra, de forma que a primeira não existe sem a segunda. | Relacionamentos da UML |
|  | **Agregação** – Relacionamento em que uma característica representa o todo, e as outras as partes. Similar à composição, porém as características envolvidas existem independentemente uma da outra. | |
|  | **Generalização** – Relacionamento em que há uma generalização/especialização das características. Este tipo de relacionamento indica que as características mais especializadas (filhas) herdam as propriedades e atributos de características mais generalizadas (antecessores). | |
|  | **Associação** – Relacionamento simples entre duas características. Denota algum tipo de ligação entre seus membros. Pode ser nomeada, indicando um tipo específico de ligação. | |
|  | **Alternativo** (*Alternative*) - Relacionamento entre um ponto de variação e suas variantes, denota a pertinência de uma variante a um determinado ponto de variação. | Relacionamentos Específicos na Odyssey-FEX |
|  | **Implementado por** *(Implemented By)* - Relacionamento entre Características de Domínio e Características Tecnológicas, ou entre Características Tecnológicas que se encontrem em camadas diferentes. | |
|  | **Ligação de Comunicação** *(Communication Link)* - *R*elacionamento existente entre Características de Entidade e Características de Domínio. Cumpre o mesmo papel do relacionamento de associação entre atores e casos de uso na UML (OMG, 2004) | |

Diferentemente do modelo de características de Kang (KANG *et al.*, 1990), os relacionamentos na notação Odyssey-FEX não são apenas hierárquicos, mas sim um grafo acíclico, permitindo a expansão do modelo de características em várias direções, como pode ser visto mais adiante, no exemplo de utilização da seção 3.4.5.

### 3.4.3– REGRAS DE COMPOSIÇÃO

Conforme descrito na seção 3.3.1.3, a notação Odyssey-FEX define Regras de Composição Complexas, que podem ser combinadas por meio de expressões booleanas. Tais expressões booleanas podem ainda ser compostas por mais de duas características,

denotando uma combinação de expressões, tanto no antecedente quanto no conseqüente da regra.

É importante observar a semântica das regras de composição com operadores booleanos, pois tal semântica influencia fortemente a interpretação das regras no recorte para a instanciação da aplicação e/ou produto. Por exemplo, considerando que A e B sejam características, uma regra "NOT(A) requer B" significa que a característica B é requerida sempre que a característica A não for incluída na instanciação da aplicação.

Desse modo, se evidencia o poder de expressividade das regras de composição complexas, atendendo ao requisito da representação de dependência e mútua exclusividade para quaisquer conjuntos de características do domínio.

### 3.4.4 - EXEMPLO DE UTILIZAÇÃO DA NOTAÇÃO ODYSSEY-FEX

A Figura 3.24 mostra um modelo de características construído utilizando-se a notação Odyssey-FEX. Neste exemplo, é considerado o domínio de Software para Telefonia Móvel descrito na seção 3.2.

No modelo apresentado, a característica *Usuário* é uma característica de Entidade, e tem uma Ligação de Comunicação com a característica conceitual *Telefone Celular*. Tal característica representa o domínio como um todo, e dela partem as ligações com as outras características do modelo. As características *Envio de Mensagem de Texto* e *Alerta Vibratório* são características funcionais e opcionais, o que significa que tais funcionalidades podem ou não estar presentes em um produto ou aplicação, no caso, um telefone celular. Da mesma forma, *Câmera* é uma característica opcional.

É possível ainda observar quatro pontos de variação: *Jogos*, *Cores no Visor, Conexão* e *Recebimento de Toques Musicais*, que indicam pontos onde o software pode ser configurado para a geração de uma aplicação ou produto específico. Nota-se que tais características podem ser classificadas como pontos de variação por meio do relacionamento Alternativo existente entre elas e suas variantes. O ponto de variação *Recebimento de Toques Musicais* tem por variantes *Toques Simples, Toques Polifônicos e MP3*, todos opcionais (desenhados com linha tracejada). Isto significa que a funcionalidade de recebimento de toques musicais é opcional no produto, e uma vez presente, pode ser configurada por meio de qualquer das variantes, pois nenhuma delas é mandatória.

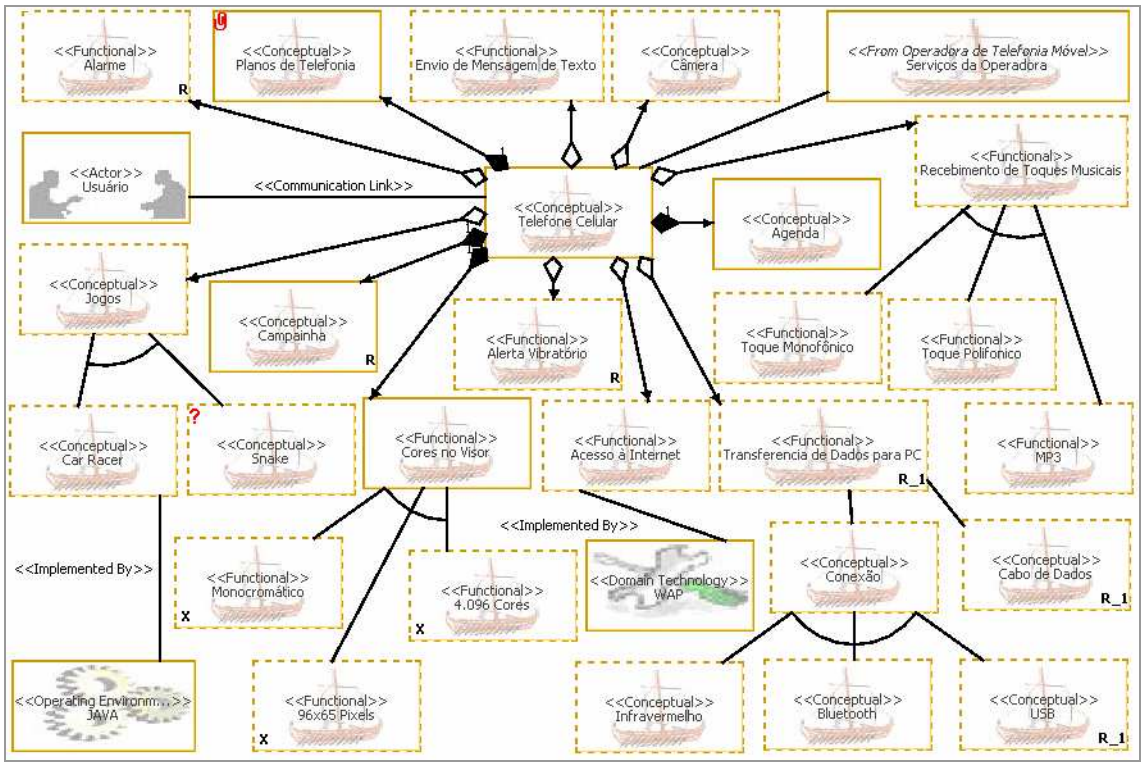



**Figura 3.24 - Exemplo de Utilização da Notação Odyssey-FEX**

O ponto de variação *Jogos* é opcional e tem por variantes *CarRacer* e *Snake*, também opcionais. Assim como no caso anterior, o conceito "Jogo" pode ou não estar presente em um telefone celular. No entanto, uma vez presente, tal ponto de variação deve ser configurado por meio de uma das variantes. Como todas as suas variantes são opcionais, qualquer uma delas pode ser selecionada para tal configuração, ficando a escolha a critério do desenvolvedor. Além disso, como no modelo não há a marcação X nas variantes *CarRacer* e *Snake*, pode-se afirmar que elas não participam de nenhuma regra de composição exclusiva, não havendo problemas na sua seleção em conjunto para um mesmo produto ou aplicação. Na característica *Snake*, pode-se observar uma interrogação no canto superior esquerdo, o que significa que essa característica ainda não foi definida por meio de outros artefatos de software, tais como casos de uso ou classes. As características *Alarme e Campainha* possuem uma marcação R, indicando que ambas fazem parte de uma regra de composição inclusiva. No caso, a regra R é "Alarme **requer** Campainha", o que significa que só é possível selecionar a característica Alarme se a característica *Campainha* for selecionada. Do mesmo modo, as características *Transferência de Dados para PC*, *USB* e *Cabo de Dados* estão envolvidas na regra de composição inclusiva "(Transferência de Dados para PC AND USB) **requer** Cabo de Dados", representada no modelo pela marcação R_1. Além

disso, *CarRacer* possui um relacionamento ImplementadoPor com a característica de Ambiente Operacional *JAVA*, o que significa que tal jogo, se existente no produto, será implementado na plataforma JAVA.

O ponto de variação *Cores no Visor* é mandatório - propriedade que pode ser vista por meio da linha não-tracejada na característica – e tem como variantes as características *4.096Cores*, *Monocromático* e *96x65Pixels*. No entanto, essas variantes são opcionais. Isto significa que um telefone celular deve possuir uma configuração para a característica *Cores no Visor*, porém nenhuma das alternativas oferecidas por suas variantes é obrigatória, cabendo ao desenvolvedor escolher qual a variante utilizar. O pequeno X existente em cada uma dessas variantes indica que elas fazem parte de uma regra de composição exclusiva, cujo nome é X. Neste exemplo, a regra X é "Monocromático **exclui** (4.096 cores AND 96x65 Pixels)". As regras X e R descritas neste exemplo não são visualizadas, além de seus nomes, no modelo de características, para que este não fique visualmente poluído. No entanto, como este modelo é construído com o apoio de um ferramental automatizado, como é descrito no capítulo 5, as regras de composição podem ser facilmente visualizadas dentro do ambiente de desenvolvimento.

Adicionalmente, tem-se a característica *Planos de Telefonia*, que é uma característica organizacional, como indica o pequeno clipe no seu canto superior esquerdo. Tal característica não tem ligação concreta com o software do telefone celular, mas organiza informações a respeito da disponibilidade de alguns serviços, de acordo com as operadoras de telefonia celular.

Finalmente, pode-se observar as características *Acesso à Internet* e *WAP*. A primeira é mais uma funcionalidade que pode ou não estar presente no produto a ser desenvolvido. Uma vez presente, o relacionamento ImplementadoPor entre esta e a característica tecnológica *WAP* indica que esta funcionalidade será implementada pela tecnologia de domínio WAP.

Os relacionamentos de composição e agregação que podem ser visualizados no modelo indicam a relação entre a característica *Telefone Celular* e as demais características. De acordo com as definições da UML (OMG, 2005), em uma composição, a "parte" não pode existir sem o "todo", enquanto na agregação, "todo" e "parte" podem existir independentemente. Sendo assim, é razoável que uma característica opcional não esteja ligada à outra característica por meio de composição. Nestes casos, o relacionamento passa a ser de agregação.

## 3.5 – CONSIDERAÇÕES FINAIS

Este capítulo apresentou a notação Odyssey-FEX, cujo propósito é prover mecanismos de representação mais abrangentes dos conceitos envolvidos na variabilidade de um Domínio ou Linha de Produtos de Software. Além da notação, é apresentado um metamodelo que formaliza os conceitos envolvidos, com regras de boa formação que servem como diretrizes para a utilização da notação Odyssey-FEX.

A Tabela 3.8 reapresenta os requisitos para representação de variabilidades, identificados no capítulo 2, desta vez, comparando a notação proposta neste trabalho com as notações analisadas na literatura.

**Tabela 3.8 - Comparação entre as notações existentes e a notação Odyssey-FEX**

| Notação / Requisito | FODA | FORM | Featu-RSEB | Svanhberg & Bosch | Riebisch | Cechticky | Czarnecki | Odyssey | Odyssey-FEX |
|---|---|---|---|---|---|---|---|---|---|
| **Elemento Invariante** | Sim | Sim | Sim | Sim | Sim | Sim | Sim | Sim | Sim |
| **Elemento Invariante Opcional e Mandatório** | Sim | Sim | Sim | Sim | Sim | Sim | Sim | Sim | Sim |
| **Ponto de Variação** | Não | Não | Sim | Não | Não | Sim | Sim | Não | Sim |
| **Ponto de Variação Opcional e Mandatório** | Não | Não | Sim | Não | Não | Sim | Sim | Não | Sim |
| **Variante** | Sim | Sim | Sim | Sim | Sim | Sim | Sim | Parcial-mente | Sim |
| **Variante Opcional/ Mandatória** | Sim | Sim | Não | Sim | Sim | Sim | Sim | Parcial-mente | Sim |
| **Dependên-cia** | Parcial-mente | Parcial-mente | Parcial-mente | Não | Sim | Parcial-mente | Não | Parcial-mente | Sim |
| **Mútua Exclusivi-dade** | Parcial-mente | Parcial-mente | Parcial-mente | Sim | Sim | Parcial-mente | Parcial-mente | Parcial-mente | Sim |
| **Relaciona-mento "Implemen-tado Por"** | Não | Sim | Não | Não | Não | Não | Não | Não | Sim |
| **Relaciona-mento "Alternati-vo"** | Não | Não | Sim | Não | Não | Parcial-mente | Sim | Não | Sim |

Pode-se constatar que a notação Odyssey-FEX atende a tais requisitos, o que pode ser interpretado como uma contribuição para uma modelagem mais abrangente das características e suas variabilidades.

Algumas notações analisadas no capítulo 2 apresentavam cardinalidade nos relacionamentos ou nas suas características, no intuito de indicar o número mínimo e máximo de características que podem ser instanciadas. Na notação Odyssey-FEX, tais cardinalidades não são representadas explicitamente, uma vez que é possível inferir a cardinalidade máxima e mínima por meio das combinações das características, como descrito na Tabela 3.9:

**Tabela 3.9 – Cardinalidades na notação Odyssey-FEX**

| | |
|---|---|
| **0..1** | representada pela característica opcional; |
| **1** | representada pela característica mandatória, ou por um ponto de variação mandatório com variantes mutuamente exclusivas |
| **0..n** | representada pelo ponto de variação opcional, quando suas variantes não são mutuamente exclusivas |
| **1..n** | representada pelo ponto de variação mandatório quando suas variantes não são mutuamente exclusivas |

Assim sendo, optou-se por não acrescentar representações gráficas para as cardinalidades, a fim de não causar uma sobrecarga de informações visuais ao usuário.

A formalização da representação da variabilidade por meio do metamodelo permite que sejam estabelecidos mecanismos de verificação de consistência inter e intra-modelos. Tais mecanismos são apresentados no próximo capítulo.

# CAPÍTULO IV
# VERIFICAÇÃO DE CONSISTÊNCIA DE MODELOS BASEADOS NA NOTAÇÃO ODYSSEY-FEX

## 4.1 – INTRODUÇÃO

Assim como no desenvolvimento de um software, a qualidade dos artefatos produzidos durante as fases iniciais de um processo de reutilização é fundamental para o seu sucesso e sua viabilidade econômica. Desse modo, é desejável que os artefatos resultantes, principalmente da fase de análise (i.e., modelos de características, casos de uso, classes), estejam modelados o mais corretamente possível, no que diz respeito à notação utilizada. Além disso, deve-se considerar que em um processo de reutilização, o uso de modelos de domínio que contenham inconsistências, ambigüidades e erros multiplica os efeitos desastrosos sobre a produtividade e qualidade, pois afeta toda uma família de sistemas sendo desenvolvidos a partir de componentes comuns.

Desse modo, é necessária uma verificação de consistência dos modelos construídos, consistência esta que deve estar presente não só em cada modelo (consistência intra-modelos), mas também entre os diversos modelos que representam o software, ou família de software, sob diferentes perspectivas (consistência inter-modelos).

A verificação de consistência pode ser efetuada de maneira formal ou informal. No entanto, embora o uso de formalismos seja motivado pela diminuição de ambigüidades, ganhos na consistência de diagramas e especificações e a possibilidade de correção por meio de provas formais, as especificações formais são complexas em função do rigor matemático das linguagens, além de dificuldade de integração com ferramentas de desenvolvimento com suporte gráfico para a criação de modelos orientados a objetos (DANTAS *et al.*, 2001).

Uma alternativa às especificações formais é a utilização de um sistema de críticas. Um sistema de críticas pode ser entendido como "um mecanismo que atua sobre ferramentas de modelagem, oferecendo correções e sugestões sobre os modelos sendo projetados" (SOUZA *et al.*, 2000; DANTAS *et al.*, 2001).

No entanto, para que haja uma verificação de consistência, regras de consistência devem ser estabelecidas, no intuito de atuar como diretrizes para o

mecanismo de verificação que será utilizado.

Nesse sentido, o presente capítulo tem como objetivo contextualizar as regras de boa formação, definidas no capítulo anterior, com o uso de um sistema de críticas para a verificação de consistência intra-modelos. Além disso, são apresentadas heurísticas para a verificação de consistência inter-modelos, no intuito de manter a coerência entre a representação da variabilidade no modelo de características e no modelo de classes.

O capítulo está organizado da seguinte maneira: na seção 4.2 são discutidas questões referentes à verificação de consistência intra-modelos. A consistência inter-modelos é discutida na seção 4.3. Na seção 4.4 é descrito o mapeamento entre os metamodelos de características e de classes. As heurísticas resultantes deste mapeamento são apresentadas na seção 4.5. Por fim, algumas considerações sobre o capítulo são encontradas na seção 4.6.

## 4.2 – VERIFICAÇÃO DE CONSISTÊNCIA INTRA-MODELOS

Quando não há verificação da consistência e da correção dos modelos sendo construídos, em virtude de notações imprecisas ou suscetíveis a interpretações incorretas, aliadas ao baixo conhecimento do domínio do problema, eleva-se o grau de incerteza sobre as especificações e chances de ambigüidades e erros serem introduzidos despercebidamente. Assim, tão importante quanto a representação de variabilidades na modelagem de um domínio, também são os mecanismos de verificação da consistência dos modelos que estão sendo construídos.

A atuação de um sistema de críticas sobre uma ferramenta de modelagem é motivada por vários aspectos, tais como (ROBBINS *et al.*, 1998):

- erros decorrentes do conhecimento limitado sobre o domínio de aplicação;
- baixos custos de revisão imediata contra altos custos de retrabalho para a eliminação de erros;
- possibilidade de aprendizado contínuo;
- redução de tempo de desenvolvimento;
- melhor gerenciamento de riscos.

A verificação de consistência de modelos UML é freqüentemente discutida na literatura (SOUZA *et al.*, 2000; DANTAS *et al.*, 2001; EGYED, 2001). No entanto, poucos trabalhos discutem a consistência intra-modelos, incluindo o modelo de características (GOMAA & SHIN, 2002 ), (SOUZA *et al.*, 2003). Assim, considerando

o modelo de características como o modelo base para a representação de variabilidades, e ponto de partida para o recorte do domínio na instanciação de uma aplicação, é interessante que seja estabelecido algum mecanismo de verificação de consistência para esse modelo.

Para a verificação de consistência intra-modelos, a existência de um metamodelo que apresente diretrizes para modelagem, i. e., regras de boa formação, se mostra importante, uma vez que tais regras servem de base para o estabelecimento de um sistema de críticas sobre o modelo. Dessa forma, as regras de boa formação da notação Odyssey-FEX, apresentadas no capítulo anterior, são resgatadas aqui com a proposta de servirem de base para um sistema de críticas intra-modelos, no contexto do ambiente Odyssey (ODYSSEY, 2005).

O ambiente Odyssey possui um mecanismo de críticas, denominado Oráculo (DANTAS *et al.*, 2001), que é uma ferramenta auxiliar à ferramenta de diagramação do ambiente. Tal mecanismo verifica a consistência de modelos UML, seguindo as regras de boa formação do seu metamodelo. Assim, com a definição do metamodelo da notação Odyssey-FEX, as regras de boa formação são incorporadas à base de regras existente no Oráculo, de modo a estender o uso do mecanismo para o modelo de características. A implementação desta proposta é detalhada no capítulo 5.

## 4.3 - VERIFICAÇÃO DE CONSISTÊNCIA INTER-MODELOS

Um software em desenvolvimento pode ser observado e modelado sob diversas visões. Tais visões representam diferentes aspectos do software e se complementam, no sentido de fornecer um maior entendimento do sistema como um todo. Da mesma forma, domínios podem ser modelados por meio de artefatos que representem seus diferentes aspectos, de maneira a fornecer informação mais completa sobre eles (GRISS *et al.*, 1998). De um modo geral, tais artefatos compreendem diagramas, que embora não sejam substituíveis uns pelos outros, também não são independentes entre si. Tal dependência entre diagramas implica em uma quantidade substancial de informação redundante, o que infelizmente, implica na possibilidade de inconsistências (EGYED, 2001). Em vista disso, os diferentes artefatos que representam um domínio devem estar em conformidade entre si, inclusive no que diz respeito à representação de variabilidades. Para tanto, deve haver um mecanismo de verificação de consistência

inter-modelos.

Assim como na verificação de consistência intra-modelos, a verificação inter-modelos deve ser baseada em regras que estabeleçam a correspondência dos elementos dos diversos modelos. Mais uma vez considerando a reutilização e o modelo de características, deve haver uma correspondência não só das características, mas também da variabilidade representada neste modelo com elementos dos modelos de mais baixo nível de abstração.

Embora as notações analisadas no capítulo 2 se proponham a representar as variabilidades de um domínio somente por meio de características, existem na literatura outras notações que propõem a representação de variabilidades em outros artefatos, tais como caso de uso (BERTOLINO *et al.*, 2002), (GOMAA & SHIN, 2002 ), ou diagrama de classes (CLAUSS, 2001a), (MORISIO *et al.*, 2000). No entanto, nenhuma dessas notações se preocupa com a propagação das variabilidades para outros artefatos, de maneira que estas sejam representadas de forma coerente nas diversas visões do domínio em questão.

Em vista disso, são propostas, neste trabalho, algumas heurísticas de mapeamento entre os metamodelos de características e de artefatos de mais baixo nível de abstração, no intuito de orientar a representação de variabilidades em tais artefatos, possibilitando uma verificação de consistência entre estes. Apesar da necessidade se mapear as variabilidades para todos os artefatos do domínio, o presente trabalho se propõe a dar apenas um primeiro passo nesse sentido, estabelecendo diretrizes para a representação de variabilidades somente no modelo de classes, uma vez que, segundo Gomaa & Shin (2002), embora seja um artefato de mais baixo nível de abstração que o modelo de características, o modelo de classes também pertence à fase de análise. O mapeamento entre modelos é apresentado a seguir.

## 4.4 – MAPEAMENTO MODELO DE CARACTERÍSTICAS - MODELO DE CLASSES

As diretrizes de mapeamento da representação de variabilidades do modelo de características para um modelo de classes foram obtidas a partir da análise da modelagem de três diferentes domínios: Ambientes de Desenvolvimento de Software (ADS), Máquinas de Processo e Sistemas de Controle Leiteiro.

O procedimento inicial para a obtenção dessas diretrizes, no caso dos domínios de ADS e Máquinas de Processo, foi o seguinte: por meio de uma ferramenta de engenharia reversa, foi extraído, a partir de sistemas já construídos, o modelo de classes. Em ambos os casos, foi possível o acesso aos desenvolvedores do sistema, de maneira que estes puderam orientar a construção do modelo de características correspondente. De posse do modelo de classes e do modelo de características, os desenvolvedores dos sistemas, e especialistas nos respectivos domínios, realizaram um mapeamento, de forma a relacionar elementos correspondentes nos dois modelos. Assim, foi identificada a forma pela qual as características que representavam conceitos e funcionalidades do domínio foram contempladas no modelo de classes.

No caso do domínio de Sistemas de Controle Leiteiro, embora o mapeamento tenha sido obtido também a partir da experiência de especialistas, o procedimento foi um pouco diferente. Um grupo de pesquisadores do domínio de Sistemas de Controle Leiteiro da Universidade Federal de Juiz de Fora foi consultado, no intuito de se observar a maneira como o domínio foi modelado. A partir dessa consulta, foram construídos os modelos de características e de classes do domínio. Em seguida, foi solicitado aos próprios pesquisadores que estabelecessem a correspondência das características do domínio com as classes modeladas. Assim, foi possível obter um mapeamento a partir de modelos construídos de acordo com uma abordagem *top-down* (modelo de características construído antes do modelo de classes), em contrapartida ao mapeamento obtido no caso dos outros domínios, onde a abordagem seguida foi a *bottom-up* (modelo de classes construído antes do modelo de características), com a utilização de engenharia reversa.

Embora provenientes de mapeamentos tão distintos entre si, a partir da análise desses mapeamentos, algumas correspondências puderam ser evidenciadas, conforme sumarizado na Tabela 5.10.

A Tabela 5.10 mostra a relação estabelecida entre os elementos dos dois modelos. Uma vez que o modelo de características obtido foi construído utilizando-se a notação Odyssey-FEX, foi realizada, a partir de tais correlações, uma abstração para os níveis de metamodelo, no intuito de relacionar os elementos do metamodelo de características com o metamodelo da UML, que norteia o modelo de classes. Para essa abstração, foi considerada a versão 2.0 da UML (OMG, 2005).

**Tabela 5.10 - Correspondência entre elementos dos modelos de Características e Classes**

| Modelo de Características | Modelo de Classes |
|---|---|
| Característica Conceitual | Classe, Atributo |
| Característica Funcional | Classe, Método, Pacote |
| Regra Composição Inclusiva | Dependência, Associação |
| Regra Composição Exclusiva | Classe, Método, Pacote (estereotipados com <<xor>>) |
| Ponto de Variação | Classe (*pai*), Interface, Classe com atributo "Tipo", Método parametrizado |
| Variante | Classe (*filha* ou que realize interface), Classe simples,Método simples, Parâmetros |
| Relacionamento Alternativo | Herança ou Implementação |

Considerando-se tal abstração, pode-se obter as seguintes correlações:

1. Uma instância da meta-classe *Característica* no metamodelo de características pode ser mapeada para uma ou mais instâncias das meta-classes *Classe, Operação*[5] ou *Pacote* no metamodelo da UML.

2. Uma instância da meta-classe *CaracterísticaConceitual* no metamodelo de características pode ser mapeada para uma ou mais instâncias das meta-classes *Classe* e *Propriedade*[6] no metamodelo da UML.

3. Uma instância da meta-classe *CaracterísticaFuncional* no metamodelo de características pode ser mapeada para uma ou mais instâncias das meta-classes *Operação* ou *Pacote* no metamodelo da UML.

4. Uma instância das meta-classes derivadas de *CaracterísticaTecnologica*, i.e., *CaracterísticaTecnicaImplementação*, *CaracterísticaAmbienteOperacional* e *CaracteristicaTecnologiaDominio*, no metamodelo de características, pode ser mapeada

[5] Na especificação da UML 2.0, métodos são representados pela classe *Operação*

6 Na especificação da UML 2.0, atributos são representados pela classe P*ropriedade*.

para uma ou mais instâncias das meta-classes *Classe*, *Operação* ou *Pacote* no metamodelo da UML.

5. Instâncias da meta-classe *CaracterísticaDeEntidade*, no metamodelo de características, não são mapeadas para instâncias da meta-classes do diagrama de classes no metamodelo da UML. Seu mapeamento é feito somente para a meta-classe **Ator** no diagrama de casos de uso da UML.

6. Uma instância da meta-classe *RegraComposiçãoInclusiva*, no metamodelo de características, pode ser mapeada para uma ou mais instâncias das meta-classes *Dependência* ou *Associação* no metamodelo da UML.

7. Uma instância da meta-classe *RegraComposiçãoExclusiva*, no metamodelo de características, pode ser mapeada para uma ou mais instâncias das meta-classes *Classe*, *Operação* ou *Pacote* no metamodelo da UML, utilizando o estereótipo <<xor>>.

8. Uma instância da meta-classe *Característica*, cujo valor do atributo *tipoVariabilidade* seja *pontoVariação*, no metamodelo de características pode ser mapeada para uma ou mais instâncias das meta-classes *Interface*, *Operação* ou *Classe* no metamodelo da UML, desde que a classe seja do tipo *Pai* em um relacionamento de generalização, ou possua um atributo "tipo". As instâncias das classes da UML devem possuir estereótipos <<vp>>.

9. Uma instância da meta-classe *Característica*, cujo valor do atributo *tipoVariabilidade* seja *variante*, no metamodelo de características, pode ser mapeada para uma ou mais instâncias das meta-classes *Operação* ou *Classe* no metamodelo da UML. Além disso, as seguintes condições podem ocorrer:

a. A classe ser do tipo Filha em um relacionamento de generalização; ou

b. A classe realizar uma interface que suporte uma instância da meta-classe Classe no metamodelo de características, cujo valor do atributo *tipoVariabilidade* seja *pontoVariação*

Neste caso, no diagrama de classes, a classe que suporta uma característica variante deve possuir estereótipo do tipo <<variant>>.

10. Uma instância da meta-classe *Pacote* no metamodelo de características pode ser mapeada para uma ou mais instâncias da meta-classe *Pacote* no metamodelo da UML.

11. Uma instância da meta-classe *ImplementadoPor*, no metamodelo de características, pode ser mapeada para uma ou mais instâncias da meta-classe *Associação* ou *Implementação* no metamodelo da UML.

12 Instâncias da meta-classe *LigaçãoDeComunicação*, no metamodelo de características, pode ser mapeada para uma ou mais instâncias da meta-classe *Associação* no metamodelo da UML.

13 . Instâncias da meta-classe *Alternativo*, no metamodelo de características, podem ser mapeadas para uma ou mais instâncias da meta-classe *Generalização* ou *Implementação* no metamodelo da UML.

14. Instâncias da meta-classe *Característica*, cujo valor do atributo *ehDefinida* é igual a "*falso*", não são mapeadas para o metamodelo da UML.

15. Instâncias da meta-classe *CaracterísticaDeDominio*, cujo valor do atributo *ehOrganizacional* é igual a "*verdadeiro*", não são mapeadas para o metamodelo da UML.

16. Uma instância das meta-classes *EspaçoDeNomes*, *Domínio* ou *Contexto*, no metamodelo de características, pode ser mapeada para uma ou mais instâncias da meta-classe *EspaçodeNomes (Namespace)* no metamodelo da UML.

A definição de tais correlações entre os metamodelos possibilita que um conjunto de heurísticas sejam estabelecidas, no intuito de nortear não só a construção de um modelo de classes a partir de um modelo de características, mas também de verificar a consistência entre os modelos, como apresentado a seguir.

## 4.5 – HEURÍSTICAS PARA VERIFICAÇÃO DE CONSISTÊNCIA INTER-MODELOS

Conforme mencionado nas seções anteriores, a utilização de um mecanismo de verificação de consistência deve ser norteado por regras. De acordo com Gomaa & Shin (2002, 2004), regras resolvem inconsistências entre as múltiplas visões do domínio, sejam essas visões correspondentes, ou não, a uma mesma fase de desenvolvimento da família de sistemas, além de permitir o mapeamento entre as múltiplas visões, quando em fases diferentes.

No entanto, uma vez que a modelagem do domínio é fortemente influenciada pela experiência de especialistas e pela maneira pela qual o domínio é visto por esses, ela pode ser considerada uma atividade subjetiva. Dessa forma, não são estabelecidas regras de mapeamento, mas heurísticas, cujo propósito é possibilitar uma representação mais coerente dos requisitos do domínio nos diferentes níveis de abstração.

Assim, a partir do mapeamento entre os metamodelos de características e de classes descrito na seção anterior, é possível estabelecer um conjunto de heurísticas, com base na observação dos domínios modelados, como exemplificado a seguir. As figuras representam exemplos ilustrativos de como as características poderiam ser mapeadas.

**Heurística 1**

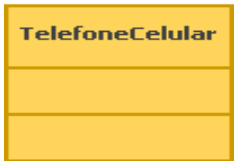
| *Se existir, no modelo de características, uma característica **conceitual**, então, pode existir no diagrama de classes uma **classe** ou **atributo** que suporte tal característica* | |
|---|---|
| **Característica Conceitual** | **Elemento Correspondente** |
|  |  |

**Figura 4.25 - Mapeamento sugerido pela Heurística 1**

**Heurística 2**

| *Se existir, no modelo de características, um elemento **pacote**, então, pode existir no diagrama de classes, um **pacote** que suporte tal elemento.* |
|---|

**Heurística 3**

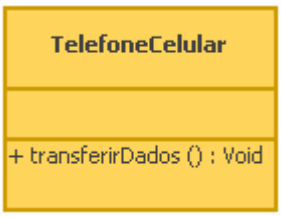
| *Se existir, no modelo de características, uma característica **funcional**, então, pode existir no diagrama de classes uma **classe**, um **método** ou **pacote** que suporte tal característica.* | |
|---|---|
| **Característica Funcional** | **Elemento Correspondente** |
|  |  |

**Figura 4.26 - Mapeamento sugerido pela Heurística 3**

**Heurística 4**

| *Se existir, no modelo de características, uma característica **tecnológica**, então, pode existir no diagrama de classes uma **classe,** um **método** ou **pacote** que suporte tal característica.* | |
|---|---|
| **Característica Tecnológica** | **Elemento Correspondente** |
|  |  |

**Figura 4.27 - Mapeamento sugerido pela Heurística 4**

**Heurística 5**

*Características classificadas como **Organizacionais** ou **Não definidas** não são mapeadas para o diagrama de classe*

**Heurística 6**

*Características de **Entidade** não são mapeadas para o diagrama de classe.*

**Heurística 7**

*Se existir, no modelo de características, um **ponto de variação**, pode existir no diagrama de classes uma **classe** ou **interface** que suporte tal elemento, utilizando-se do estereótipo <<vp>>*

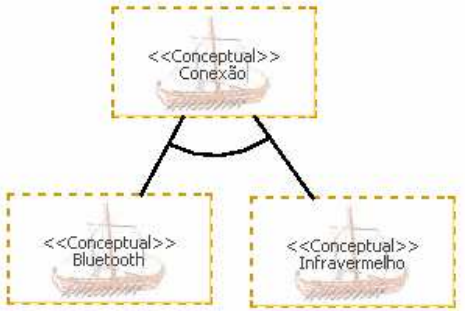
| Ponto de Variação | Elemento Correspondente |
|---|---|
|  |  |



**Figura 4.28 - Mapeamento sugerido pela Heurística 7**

**Heurística 8**

*Se existir, no modelo de características, uma característica **variante**, então podem existir, no diagrama de classes, **classes** ou **métodos** que suportem tal elemento, utilizando o estereótipo <<variant>>*

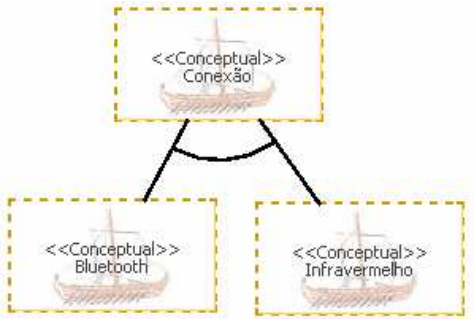
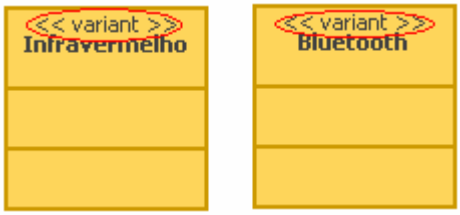
| Variante | Elemento Correspondente |
|---|---|
|  |  |



**Figura 4.29 - Mapeamento sugerido pela Heurística 8**

**Heurística 9**

*O relacionamento do tipo **LigaçãoDeCommunicação** entre duas características no modelo de características **não é mapeado** no diagrama de classes, uma vez que tal relacionamento envolve pelo menos uma característica do tipo **Entidade**, que não é mapeada.*

**Heurística 10**

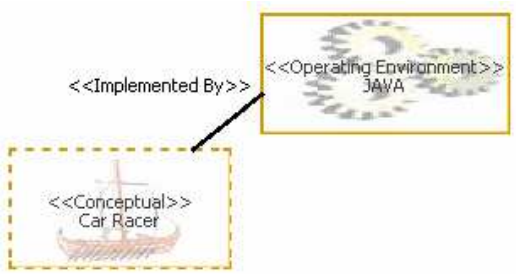
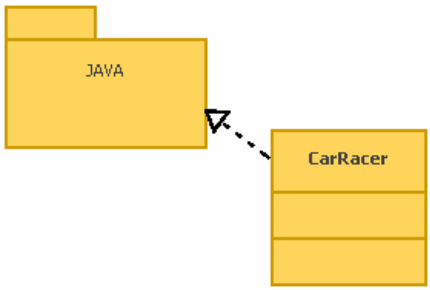
| *Se existir, no modelo de características, um relacionamento do tipo **ImplementadoPor** entre duas características, então pode existir no diagrama de classes um relacionamento de **associação** ou **implementação** entre as classes que suportam tais características.* | |
|---|---|
| **ImplementadoPor** | **Elemento Correspondente** |
|  |  |

**Figura 4.30 - Mapeamento sugerido pela Heurística 10**

**Heurística 11**

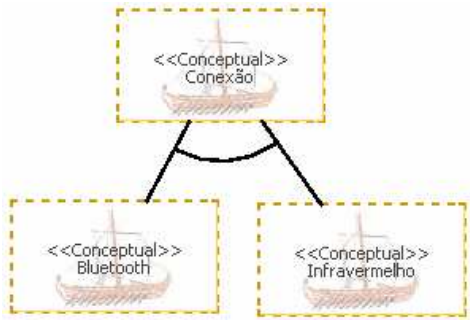
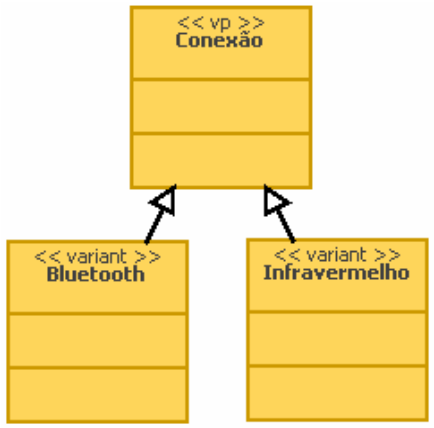
| *Se existir, no modelo de características, um relacionamento do tipo **Alternativo** entre um ponto de variação e suas variantes, então, pode existir no diagrama de classes um relacionamento de **herança** entre as classes que suportam tais características, se o ponto de variação estiver mapeado para uma **classe**. No caso de o ponto de variação estar mapeado para uma **interface**, pode existir no diagrama de classes um relacionamento de **realização** entre as classes que suportam as variantes e a interface que suporta o ponto de variação.* | |
|---|---|
| **Alternativo** | **Elemento Correspondente** |
|  |  |

**Figura 4.31 - Mapeamento sugerido pela Heurística 11**

**Heurística 12**

*Se existir, no modelo de características, uma **Regra de Composição Inclusiva** entre duas ou mais características, então, pode existir no diagrama de classes um relacionamento de **dependência** ou **associação**, (com multiplicidade mínima igual a 1 para a classe requerida) entre as classes que suportam tais características.*

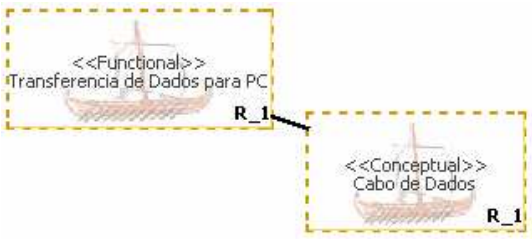
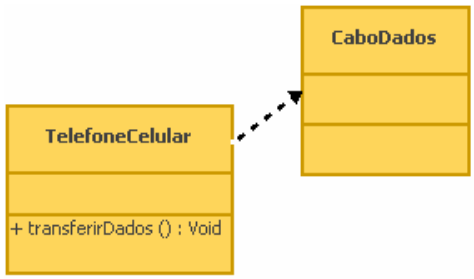
| Regra de Composição Inclusiva | Elemento Correspondente |
|---|---|
|  |  |



**Figura 4.32 - Mapeamento sugerido pela Heurística 12**

**Heurística 13**

*Se existir, no modelo de características, uma **Regra de Composição Exclusiva** entre duas características, então, pode existir no diagrama de classes **estereótipos** do tipo <<xor>> nas classes, métodos ou pacotes que suportam tais características.*

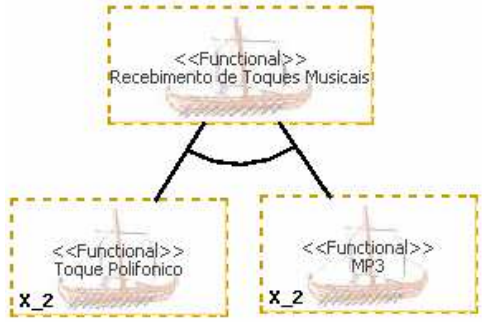
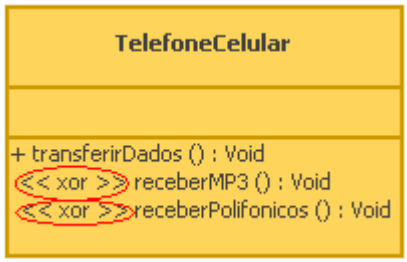
| Regra de Composição Exclusiva | Elemento Correspondente |
|---|---|
|  |  |



**Figura 4.33 - Mapeamento sugerido pela Heurística 13**

**Heurística 14**

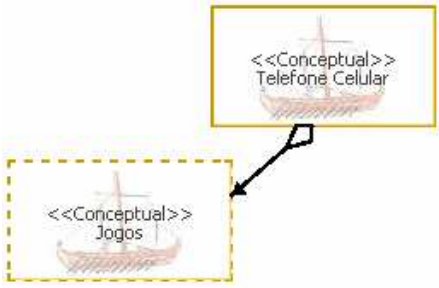
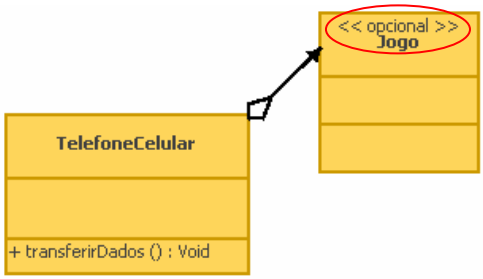
| *Em todos os casos, deve ser observada a condição de opcionalidade do elemento no modelo de características: elementos não-mandatórios no modelo de características devem ser suportados por classes, métodos, atributos e pacotes não-mandatórios no diagrama de classe, representados graficamente e devidamente diferenciados dos elementos mandatórios.* | |
|---|---|
| **Características Opcional e Mandatória** | **Elementos Correspondentes** |
|  |  |

**Figura 4.34 - Mapeamento sugerido pela Heurística 14**

As heurísticas de mapeamento podem ser muito mais úteis quando suportadas por um ferramental automatizado, possibilitando a sua utilização na verificação de consistência entre modelos existentes, ou até mesmo na geração de modelos. No entanto, é importante observar as ligações entre tais modelos. Para utilização de tais heurísticas, é necessário que exista uma granularidade pequena o suficiente dos rastros entre os elementos inter-modelos. Por exemplo, deve existir a possibilidade de mapeamento entre uma característica e um método ou atributo de uma classe.

É interessante notar ainda que, embora as heurísticas de mapeamento tenham sido obtidas a partir de exemplos que seguiam processos distintos para a modelagem do domínio, a utilização dessas heurísticas na construção do modelo de classes está inserida dentro de um processo, implicitamente conduzindo o desenvolvedor a uma seqüência pré-determinada de atividades de modelagem de domínio. No entanto, tais heurísticas podem ser reutilizadas em outros processos, não sendo obrigatório que o diagrama de classes seja construído imediatamente após o modelo de características. Como exemplo, tem-se o refinamento do processo *CBD-Arch-DE* (OLIVEIRA *et al.*, 2005b) que apresenta um mapeamento das características para o modelo de tipos de negócios, com o intuito de propagar as variabilidades até o nível de componentes.

Espera-se com esse mapeamento definir orientações para a propagação da

representação das variabilidades para modelos de mais baixos níveis de abstração, diminuindo inconsistências e dificuldades de evolução do software.

## 4.6 – CONSIDERAÇÕES FINAIS

Este capítulo apresentou questões referentes à verificação de consistência intra e inter-modelos de domínio. Foi proposta a utilização das regras de boa formação da notação Odyssey-FEX como base para um sistema de críticas existente no ambiente Odyssey, por meio de sua extensão para contemplar o modelo de características.

É proposto ainda um mapeamento do metamodelo de características Odyssey-FEX para o diagrama de classes da UML. Tal mapeamento tem o intuito de estabelecer heurísticas para a representação das variabilidades em artefatos de níveis de abstração mais baixos, bem como verificar a consistência inter-modelos. Assim, se torna possível a propagação desta representação de uma maneira mais coerente por todos os artefatos de uma Linha de Produtos de Software, ou domínio.

Dos trabalhos encontrados na literatura a respeito de sistemas de críticas para verificar consistência intra e inter-modelos, o trabalho de Souza *et al* (2003) é o que apresenta uma abordagem mais próxima da proposta deste trabalho. Tal proximidade se deve ao uso da notação de Miler (2000) como base para um sistema de críticas, inserido em um ambiente denominado DAISY. No entanto, a abordagem descrita pelos autores é diferente da abordagem adotada pelo Oráculo, no ambiente Odyssey. Analisando comparativamente as propostas do sistema de críticas existente no ambiente DAISY e do Oráculo, alguns pontos devem ser levados em consideração.

A abordagem descrita em (SOUZA *et al.*, 2003) é dependente da notação utilizada no diagrama de características. Assim, como utiliza a notação de Miler, o sistema de críticas pode não ser satisfatório em relação a críticas de variabilidade, uma vez que a notação de Miler apresentava deficiências nesse sentido. Levando em consideração a flexibilidade do Oráculo na inclusão de regras e as deficiências da notação de Miler que foram sanadas pela notação Odyssey-FEX, pode-se dizer que o Oráculo possibilita uma verificação de consistência mais completa, permitindo que um menor número de erros seja propagado para outros modelos do domínio.

Além disso, o sistema de críticas do ambiente DAISY apresenta alguns exemplos de críticas utilizadas na edição/construção de um diagrama de características. No entanto, não deixa claro como essas críticas foram criadas, isto é, as críticas parecem

ter sido estabelecidas informalmente, a partir de observações da notação utilizada. Nesse sentido, a proposta do Oráculo possui a vantagem de estar embasada em um metamodelo de características, com regras de boa formação bem definidas, que servem como guia para a elaboração das críticas e detecção de inconsistências no diagrama de características.

Embora proponha a verificação de consistência entre modelos de características e de classes, o trabalho de Souza *et al* (2003) propõe essa verificação como uma atividade de Engenharia de Aplicação (EA), i. e., a consistência é verificada somente no momento da instanciação da aplicação. Na verdade, essa verificação de consistência se faz necessária ainda na Engenharia de Domínio, quando os diagramas de classes são construídos como artefatos que fazem parte do conjunto de modelos de domínio. O sistema de críticas deveria, assim, verificar a consistência entre os modelos de características e o modelo de classes do domínio, não postergando tal verificação até o momento do recorte, na EA.

Dessa maneira, acredita-se no apoio oferecido à detecção de inconsistências, por meio de regras para verificação intra-modelo, bem como de heurísticas para verificação inter-modelos. Assim, incluindo o modelo de mais alto nível de abstração, evita-se, entre outras coisas, a propagação de erros de modelagem para as diversas fases de modelagem, bem como aplicações de um domínio. Espera-se, com essa proposta, um maior apoio à construção dos artefatos de domínio, de maneira coerente, aumentando assim a produtividade e a qualidade do desenvolvimento de software para e com reutilização.

# CAPÍTULO V
# IMPLEMENTAÇÃO DA NOTAÇÃO ODYSSEY-FEX EM UM AMBIENTE DE REUTILIZAÇÃO

## 5.1- INTRODUÇÃO

No Capítulo 3, foi apresentada a proposta da notação Odyssey-FEX, cujo objetivo era oferecer uma maior gama de recursos para a representação das variabilidades nos artefatos de uma estrutura de reutilização, como um Domínio ou uma Linha de Produtos de Software. Para tornar viável a sua utilização na prática, é necessário o apoio de um ferramental automatizado que auxilie o desenvolvedor na construção do modelo de características e outros artefatos a ele relacionados. Uma vez que, em um processo de desenvolvimento de um domínio, ou de uma Linha de Produtos de Software, o modelo de características não é um artefato isolado, é interessante construí-lo e utilizá-lo em um ambiente de reutilização que forneça subsídios necessários à construção e integração dos artefatos envolvidos.

Para que a reutilização seja efetiva, é necessário que o modelo de características seja modelado corretamente, e esteja coerente com os outros artefatos relacionados, no que diz respeito, por exemplo, à representação de variabilidade. Assim, um mecanismo de verificação de consistência é de grande utilidade, para que erros eventualmente introduzidos na modelagem de características não sejam propagados para outros artefatos e fases do ciclo de vida.

Nesse sentido, é apresentado neste capítulo o ambiente de reutilização Odyssey (ODYSSEY, 2005), que foi estendido para apoiar a utilização da notação Odyssey-FEX no desenvolvimento de um domínio ou Linha de Produtos de Software, bem como o sistema de críticas para modelos de software denominado Oráculo (DANTAS *et al.*, 2001), e sua atuação no modelo de características.

O capítulo está organizado da seguinte maneira: a seção 5.2 apresenta uma descrição do ambiente Odyssey, como contexto de utilização da notação Odyssey-FEX. A seção 5.3 descreve a implementação e adaptações realizadas, a fim de viabilizar a modelagem de características, de acordo com a proposta apresentada nos capítulos anteriores. A adaptação do sistema de críticas é descrita na seção 5.4. Na seção 5.5, é descrito um estudo de observação que caracteriza a viabilidade das regras de boa

formação da notação Odyssey-FEX. Por fim, na seção 5.6, são feitas algumas considerações sobre a implementação descrita neste trabalho.

## 5.2- CONTEXTO DE UTILIZAÇÃO - O AMBIENTE ODYSSEY

O ambiente Odyssey (ODYSSEY, 2005) é um ambiente de desenvolvimento de software que oferece ferramentas de apoio à construção de aplicações, baseado em reutilização. Para tanto, contempla atividades do desenvolvimento *para* reutilização, conhecido como Engenharia de Domínio (ED), e do desenvolvimento *com* reutilização, conhecido como Engenharia de Aplicação (EA). Para auxiliar o desenvolvedor, o ambiente oferece seus próprios processos de reutilização: Odyssey–ED (BRAGA, 2000), e *CBD-Arch-DE* (BLOIS *et al.*, 2004), para ED, e Odyssey-EA (MILER, 2000), para EA.

O ambiente Odyssey é composto de ferramentas que ajudam na automatização das diversas etapas definidas em um processo de reutilização. Podemos citar, como exemplo, ferramentas para a documentação de componentes (MURTA *et al.*, 2001), especificação e instanciação de arquiteturas específicas de domínios (XAVIER, 2001), apoio à engenharia reversa (VERONESE *et al.*, 2002), ferramenta de notificação de críticas em modelos UML (DANTAS *et al.*, 2001), suporte a padrões de projeto (DANTAS *et al.*, 2002), ferramentas de modelagem e execução de processos (MURTA *et al.*, 2002), ferramenta para controle de versões de modelos (OLIVEIRA *et al.*, 2004), apoio à edição concorrente de modelos (LOPES *et al.*, 2004), ferramenta de transformação de modelos (MAIA *et al.*, 2005), entre outras. Toda essa infra-estrutura é implementada utilizando-se a linguagem Java (SUN, 2005).

A estrutura interna do ambiente Odyssey tem uma forte relação com a maneira como os artefatos são reutilizados. Esta estrutura apresenta níveis decrescentes de abstração, onde os elementos mais abstratos, situados no topo da estrutura, possuem rastros, i.e., ligações, para os elementos menos abstratos, situados mais abaixo. De acordo com Braga (2000) e Miler (2000), os elementos mais abstratos da estrutura são os contextos, seguidos pelas características (*features*). É a partir da seleção de contextos e características que é feito o recorte do domínio, i.e., a seleção de artefatos do domínio para a instanciação de uma aplicação ou produto. O recorte possibilita que outros artefatos (ex. casos de uso, classes e componentes), relacionados por meio dos rastros, sejam reutilizados em uma nova aplicação. Neste procedimento, as características e os

relacionamentos anteriormente especificados no domínio são mantidos, permitindo que ajustes sejam feitos para refletir detalhes da nova aplicação (MILER, 2000).

As características do ambiente Odyssey, descritas acima, justificam a sua escolha como contexto de utilização da notação Odyssey-FEX.

## 5.3- IMPLEMENTAÇÃO DA NOTAÇÃO ODYSSEY-FEX

Assim como as notações analisadas na literatura, o ambiente Odyssey, enquanto ferramental de apoio à construção de modelos, apresentava deficiências no que diz respeito à representação dos conceitos de variabilidade. Dessa maneira, uma extensão de funcionalidades já existentes no ambiente fez-se necessária.

É no nível das características - denominado, no Odyssey, "*Features View*" - que se concentra o maior volume de trabalho no sentido de adaptar o ambiente à utilização da nova notação. Esta adaptação é detalhada a seguir.

### 5.3.1– ESTRUTURA SEMÂNTICA DO AMBIENTE ODYSSEY

Para compreender a implementação da notação Odyssey-FEX , é necessário que se conheça um pouco da estrutura semântica do ambiente Odyssey. Na Figura 5.35, pode ser visto um diagrama de classes que representa parte da estrutura interna do Odyssey. Foram contempladas, neste diagrama, o conjunto de classes que apresentam maior relevância para este trabalho, que será chamado de *kernel*.

A representação interna do ambiente Odyssey é organizada hierarquicamente por meio de uma árvore semântica de objetos. Todo objeto da árvore semântica é chamado de *ModeloAbstrato*, e representa um elemento de modelagem. A partir de um desses elementos, é possível percorrer a árvore hierarquicamente e as relações entre os objetos, a fim de obter as informações necessárias.

A árvore semântica é organizada em categorias de modelos, representados, na figura, pela classe *Modelo*. São definidas no Odyssey as categorias Contextos, Características, Casos de Uso, Tipos de Negócio, Classes e Componentes. Cada modelo é composto por diversos itens de modelagem, instâncias da classe *ItemModelo*, que representam pacotes, diagramas, nós e ligações específicos à categoria do modelo. A classe abstrata *No* é uma superclasse para elementos como classes, características, componentes, etc, enquanto a classe *Ligacao* representa as ligações entre os diversos nós, tais como Herança, Associação, etc.

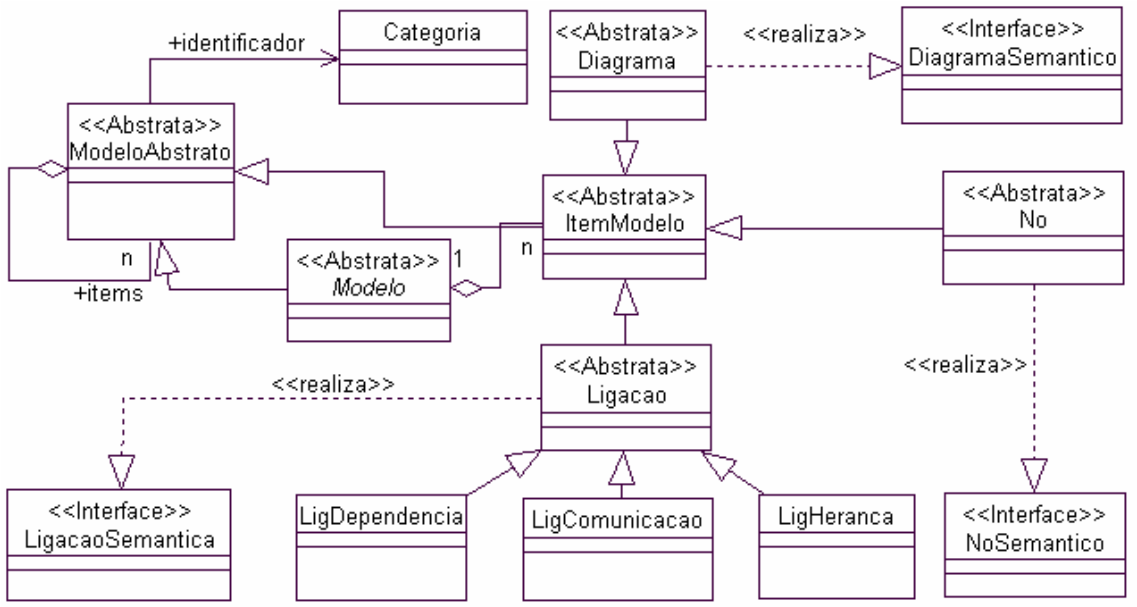



**Figura 5.35 - Representação interna do ambiente Odyssey: *kernel***

Para a implementação da notação Odyssey-FEX, as classes *No* e *Ligação* foram estendidas, no intuito de abranger a nova taxonomia, conforme apresentado anteriormente pelo metamodelo. Na Figura 5.36, é apresentada a parte da estrutura interna atual do ambiente Odyssey, referente à extensão das classes *No* e *Ligacao*. As classes destacadas em cinza representam as novas classes implementadas em função da notação Odyssey-FEX.

A classe *NoCaracterística* representa uma característica genérica, que é a superclasse para as outras características existentes. Cada categoria proposta na notação Odyssey-FEX é representada por uma classe no modelo. Além das novas classes, novos atributos foram acrescentados à classe *NoCaracterística* para determinar as condições de característica mandatória ou opcional, externa, não definida e organizacional. A classe *NoCaracterísticaEntidade* já existia no ambiente Odyssey, não sendo necessária nenhuma alteração para sua adequação à notação proposta.

Da mesma maneira, a classe *Ligacao* representa uma generalização para os tipos de ligação presentes no *kernel* do Odyssey, bem como para as ligações acrescentadas pela notação Odyssey-FEX. Os relacionamentos Alternativo, ImplementadoPor e LigaçãoDeComunicação são representados, no modelo, também por classes.

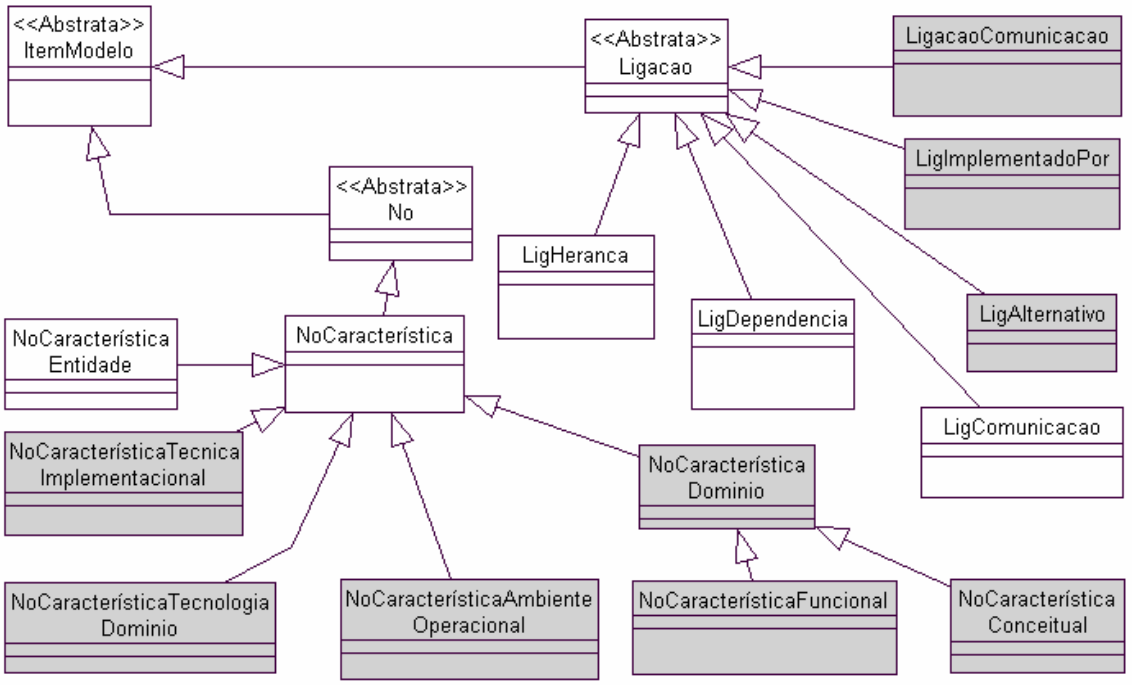



**Figura 5.36 – Extensão das estrutura interna do Odyssey de acordo com a notação Odyssey-FEX**

Além desta extensão da estrutura semântica do ambiente Odyssey, outras classes referentes às Regras de Composição foram introduzidas e serão melhor detalhadas na seção 5.3.3.

### 5.3.2 - PADRÃO DE DOMÍNIO DAS CARACTERÍSTICAS

Um dos aspectos do modelo de características do ambiente Odyssey é o detalhamento das características em um padrão de documentação, denominado **padrão de domínio** (MILER, 2000). Para contemplar a taxonomia definida na notação Odyssey-FEX, algumas adaptações neste padrão de domínio foram realizadas. A Figura 5.37 mostra a janela do padrão de domínio atual de uma característica, depois da implementação da notação. Na aba "*Feature*", é possível visualizar os campos necessários à definição da característica, sob a taxonomia da notação Odyssey-FEX.

No padrão de domínio de uma característica, o campo que apresenta a classificação quanto à categoria não permite edição, sendo apresentado ao usuário de acordo com o tipo de característica previamente criada. Do mesmo modo, caso a característica em questão seja um ponto de variação, é possível visualizar as variantes correspondentes, porém a edição é feita somente por meio do diagrama de características, como é detalhado na seção 5.3.4.

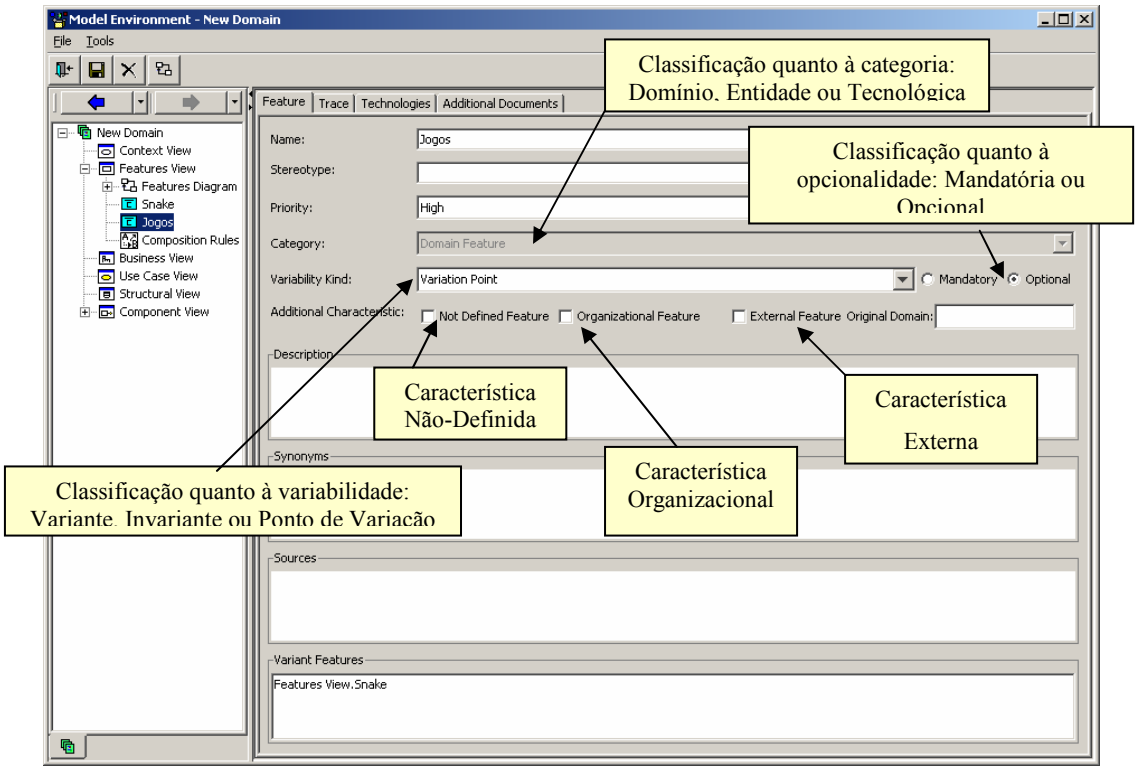



**Figura 5.37 – Padrão de domínio da características adaptado para a notação Odyssey-FEX**

As outras classificações descritas na notação Odyssey-FEX, tais como tipo de variabilidade, característica não definida, organizacional ou externa, podem ser configuradas de acordo com a necessidade do usuário. Além desses campos, é possível, por meio do padrão de domínio, fornecer outras informações a respeito da característica, tais como Nome, Prioridade, Descrição, Sinônimos e Fontes, como inicialmente previsto.

Por meio do padrão de domínio, é possível estabelecer a rastreabilidade de uma característica, i. e, estabelecer uma ligação da característica com os demais artefatos de software do domínio, ou Linha de Produtos, tais como tipos de negócio, casos de uso, componentes, etc. Esta rastreabilidade é o ponto central do processo de Engenharia de Aplicação, que é detalhado na seção 5.3.5.

### 5.3.3 - CRIAÇÃO DAS REGRAS DE COMPOSIÇÃO

Conforme já mencionado no capítulo 3, é proposto neste trabalho o uso de Regras de Composição com operadores booleanos. Para implementar essa proposta, foi acrescentado à estrutura semântica do Odyssey o pacote *RegrasComposicao*, que contém as classes especificadas pelo metamodelo apresentado no capítulo 3. A Figura 5.38 mostra como o pacote *RegrasComposicao* está relacionado com o *kernel* do ambiente Odyssey, representado pelas classes não destacadas em cinza na figura.

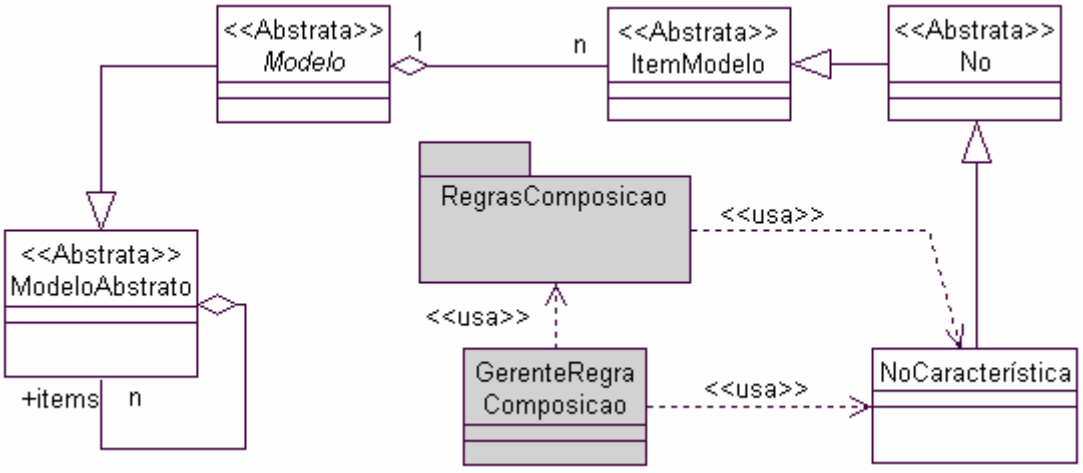



**Figura 5.38 – Pacote *RegrasComposicao* no ambiente Odyssey**

Na Figura 5.39, é apresentado um detalhamento da interação entre o pacote *RegrasComposicao* e o *kernel* do Odyssey. A classe *GerenteRegraComposicao* presente no modelo representa a interface entre as regras de composição e o ambiente. No Odyssey, instâncias da classe *RegraComposicaoInclusiva* ou *RegraComposicaoExclusiva* são criadas com antecedente e conseqüente nulos, que depois são associados a uma expressão. Essa associação é feita por meio da classe *GerenteRegraComposicao*. Além disso, informações a respeito das regras de composição, como por exemplo a quais regras determinada característica está associada, são também de responsabilidade desta classe. Isso explica as dependências de *GerenteRegraComposicao* em relação às classes *RegraComposicaoInclusiva*, *RegraComposicaoExclusiva* e *NoCaracterística*.

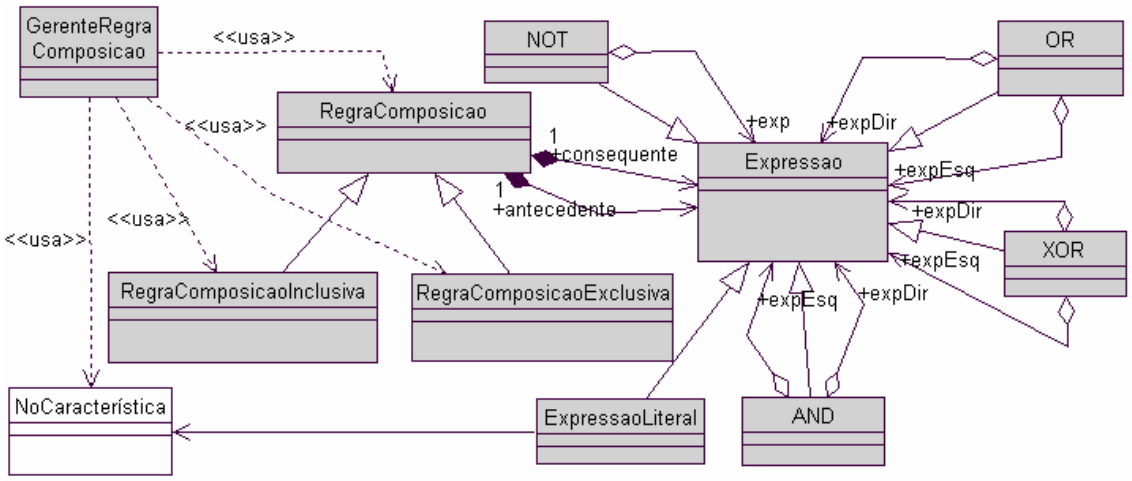



**Figura 5.39 - Detalhamento do pacote *RegrasComposicao***

A dependência entre o pacote *RegrasComposicao* e a classe *NoCaracterística* apresentada na Figura 5.38 se justifica pela associação entre as classes

*ExpressaoLiteral*, pertencente ao pacote *RegrasComposicao*, e *NoCaracterística*, como apresentado na Figura 5.39.

As regras de composição são criadas a partir das características já existentes no ambiente, i. e., para a criação de uma regra de composição é necessária a criação prévia das características envolvidas.

No Odyssey, o processo de criação de uma regra de composição começa na visão de características, chamada "*Features View*". Ao se expandir a árvore semântica, a opção *Composition Rules* é visualizada. A Figura 5.40 apresenta a janela inicial de criação de uma regra de composição.

As regras de composição inclusivas e exclusivas são criadas separadamente, por meio de botões na barra de ferramenta. Cada regra instanciada se torna parte da árvore semântica do Odyssey, e recebe um nome – **R** para regras inclusivas e **X** para regras exclusivas – incrementado seqüencialmente.

Ao selecionar uma regra na árvore semântica, uma janela como a apresentada na Figura 5.41 será mostrada.

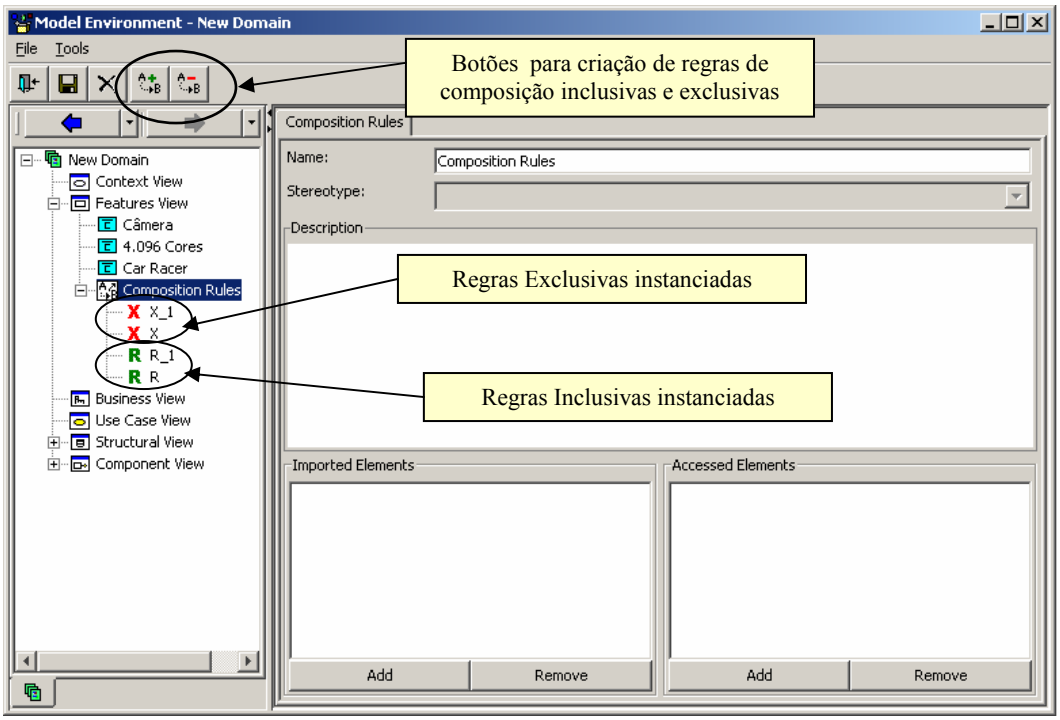



**Figura 5.40 – Painel de criação de Regras de Composição**

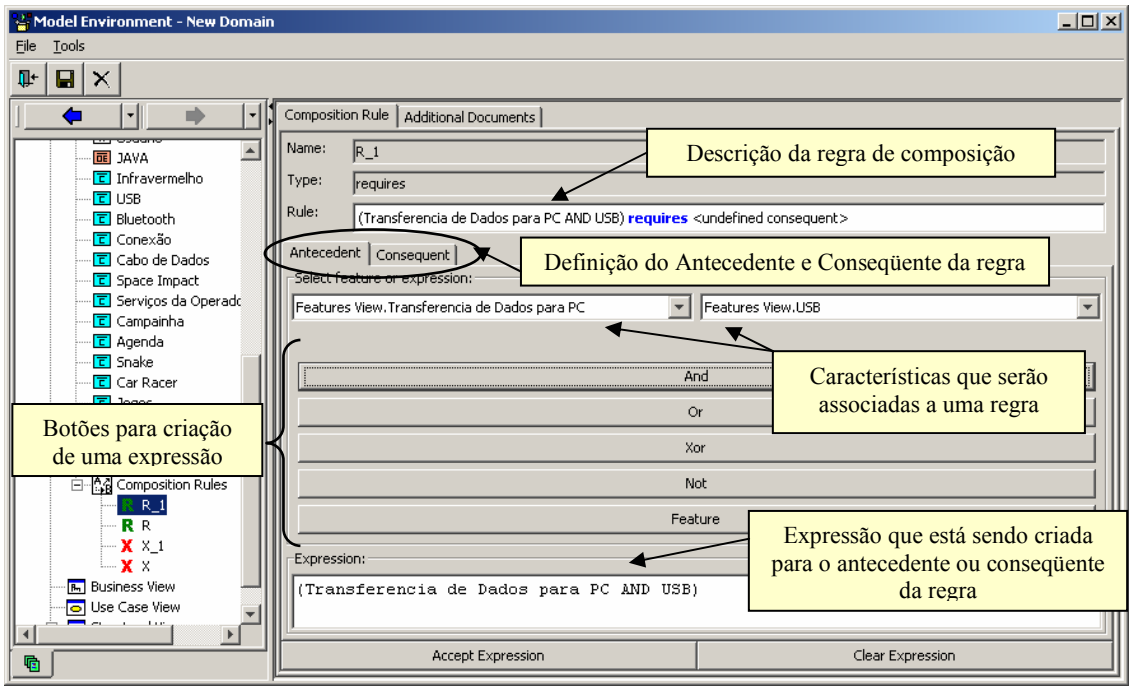



**Figura 5.41 – Definição dos componentes de uma regra de composição – Antecedente e Conseqüente**

É por meio desta janela que o antecedente e conseqüente de uma regra são definidos ou editados. Logo que a regra é instanciada, seu antecedente e conseqüente são nulos e apresentados, respectivamente, como <*undefined antecedent*> e <*undefined consequent*>. Embora o processo de definição do antecedente e conseqüente de uma regra seja o mesmo, estes são criados distintamente, por meio da escolha da janela correspondente, representada pelas abas *Antedecent* e *Consequent*. É apresentado, na Figura 5.41, o exemplo de uma regra de composição inclusiva parcialmente criada. Neste exemplo, apenas o antecedente da regra está definido, representado pela expressão (Transferência de Dados para PC AND USB). Para a criação desta expressão, são utilizadas as caixas de seleção (*combo boxes*), que contém as características e expressões previamente criadas. Desse modo, são selecionadas, uma em cada caixa de seleção, as características Transferência de Dados para PC e USB. Em seguida, utilizando-se no botão AND, a expressão é criada. O próximo passo é efetivamente atribuir a expressão criada ao antecedente da regra, por meio do botão *Accept Expression*.

Embora tenham sido usadas apenas duas características, neste exemplo, para a criação de uma expressão, é possível criar expressões mais complexas, por meio da combinação entre expressões e características, ou mesmo por meio da combinação entre expressões. Por exemplo, é possível a criação de expressões do tipo (Transferência de

Dados para PC AND (USB OR Infravermelho)), ((*Bluetooth* OR Infravermelho) OR (USB AND Cabo de Dados)). Isto é possível porque cada expressão criada se torna visível na caixa de seleção, podendo ser selecionada como parte de outra expressão. Os parênteses são colocados automaticamente a cada expressão criada, como pode ser acompanhado por meio da área de texto intitulada *Expression*. Isso implica na ordem de criação das expressões, que devem ser iniciadas pelos parênteses mais internos. No entanto, colocando-se os parênteses corretamente, evita-se ambigüidades na interpretação da regra.

Apesar da expressividade oferecida pelas regras de composição com operadores booleanos, a criação de regras de composição simples, envolvendo apenas duas características pode ser necessária. Para a notação Odyssey-FEX, uma regra de composição simples é aquela cujo antecedente e conseqüente são definidos por expressões literais, que por sua vez são formadas, respectivamente, por apenas uma característica. Portanto, é utilizado o mesmo mecanismo para criação de regras compostas, implementado no Odyssey. O processo de criação se dá da seguinte maneira: uma característica é selecionada dentre as características disponíveis na caixa de seleção da esquerda, na aba *Antecedent*. Conforme já mencionado, o antecedente da regra é uma expressão literal. Esta é instanciada por meio do botão *Feature* e visualizada na área de texto *Expression*, passando efetivamente a fazer parte da regra após a confirmação por meio do botão *Accept Expression*. Alternando-se para a aba *Consequent*, o processo é então repetido, completando assim a criação da regra de composição.

É importante observar que a implementação da funcionalidade de criação das regras de composição com operadores booleanos permite a criação de regras do tipo "(USB AND USB)", "(Infravermelho OR Infravermelho)". Tais expressões, embora redundantes, não representam erro.

Uma vez criadas as regras de composição, estas podem ser visualizadas, por meio das próprias características no modelo de características. Os nomes dados às regras de composição não são passíveis de alteração, uma vez que, como o objetivo é mostrá-los no modelo de características, uma flexibilidade muito grande na escolha do nome poderia poluir visualmente o modelo.

Além de serem visualizadas, as regras de composição também são levadas em consideração no recorte feito no processo de Engenharia de Aplicação, como é visto nas próximas seções.

### 5.3.4 - O DIAGRAMADOR DE CARACTERÍSTICAS

O Odyssey, enquanto ambiente de desenvolvimento de software, possui sua ferramenta de diagramação, denominada Editor de Diagramas. Esta ferramenta permite a construção de modelos segundo um subconjunto da linguagem UML (OMG, 2005). Além dos modelos especificados pela UML, é construído o modelo de características, foco de atenção neste trabalho.

Até este momento, foi apresentada neste capítulo a estrutura semântica do ambiente Odyssey. Porém, além da estrutura semântica dos elementos, existe uma estrutura léxica formada pelos diagramas específicos a cada modelo. Para todo item de modelo semântico, existem zero ou mais itens léxicos correspondentes, nós e arestas que formam os desenhos dos diagramas. Um diagrama semântico contém apenas um diagrama léxico.

No Odyssey, todas as categorias de diagramas (ex. Diagrama de Classes, Diagrama de Componentes, etc) são representadas por um painel específico, que apresenta o desenho do diagrama léxico, i.e., os nós e arestas léxicos pertencentes àquele diagrama. Isso inclui o diagrama de características. Neste painel, os elementos léxicos sabem se os desenhos estão de acordo com a notação determinada pela especificação da linguagem UML, e no caso do modelo de características, de acordo com a notação Odyssey-FEX. Cada um desses itens léxicos, por sua vez, está associado a um único item semântico e todas as suas propriedades, seja este um nó ou uma ligação.

A Figura 5.42 mostra a janela do diagramador de características do Odyssey, onde, à esquerda, consta a árvore de itens semânticos instanciados e, à direita, o diagrama com o desenho das características (nós) e ligações (arestas) nele contidos. A exibição desse diagrama corresponde ao item semântico *Features Diagram*, que está atualmente selecionado na árvore. Na parte superior da janela, existe uma barra de ferramentas que auxilia o usuário nas tarefas de modelagem. Esta barra, que muda dinamicamente conforme a categoria do item selecionado na árvore, corresponde às ações específicas da categoria de características do Odyssey.

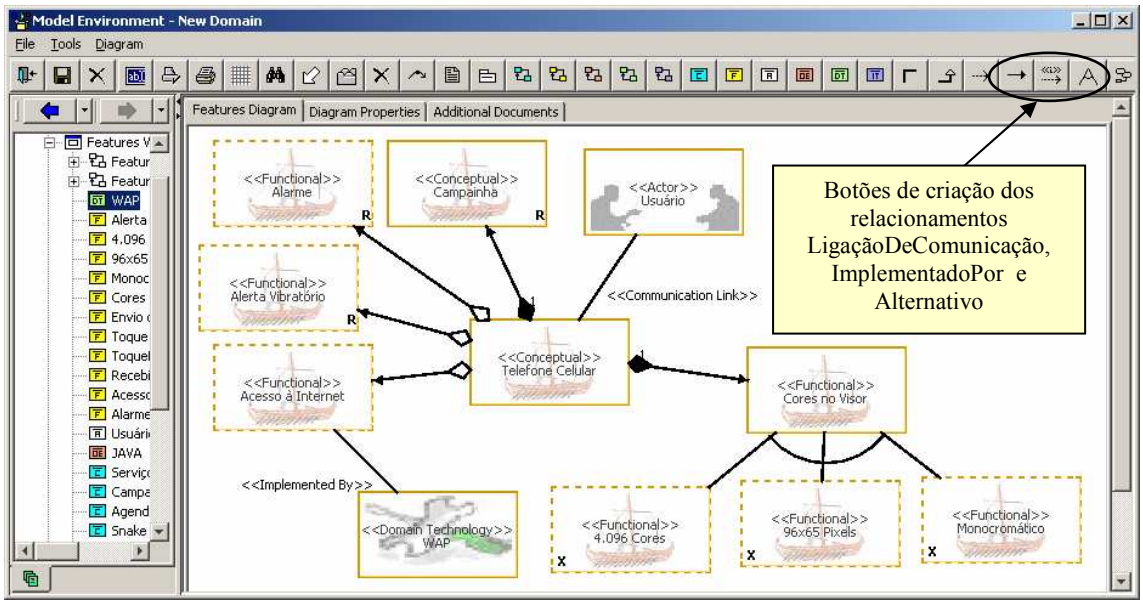



**Figura 5.42 – Janela de diagramação do ambiente Odyssey**

É por meio desta barra de ferramentas que elementos são inseridos no diagrama no qual será construído o modelo de características. A fim de contemplar a notação Odyssey-FEX, foram acrescentadas às ações previamente existentes no diagramador as seguintes ações: criação dos relacionamentos ImplementadoPor, LigaçãoDeComunicação e Alternativo. Ao selecionar uma dessas ações, são criados os elementos semântico e léxico referentes ao relacionamento, porém somente a ligação léxica é exibida no Odyssey.

Uma vez que os elementos semânticos relativos aos relacionamentos não têm representação na árvore semântica do Odyssey, é por meio do diagramador que estes são manipulados. Por exemplo, somente por meio do diagramador é possível associar ou desassociar um ponto de variação de suas variantes, bem como remover relacionamentos ou editar propriedades, como multiplicidades, no caso de associações.

No diagrama de características, é possível visualizar as características envolvidas em regras de composição previamente criadas. Na Figura 5.42, se observa que as características *4.096 Cores*, *Monocromático* e *96x65Pixels* estão assinaladas com um **X** em seu canto inferior direito. Isso significa que tais características estão envolvidas na mesma regra de composição exclusiva, denominada **X**. Do mesmo modo, as características *Alarme*, *Campainha* e *Alerta Vibratório* estão sinalizadas com um R no seu canto inferior direito, significando que estão envolvidas na mesma regra inclusiva denominada **R**.

### 5.3.5 - O PROCESSO DE ENGENHARIA DE APLICAÇÃO

Para se desenvolver um produto em uma Linha de Produtos a partir de um núcleo de artefatos, ou para se derivar uma aplicação de um domínio, é utilizado um processo de Engenharia de Aplicação (EA) (ATKINSON *et al.*, 2002). No entanto, é necessário que nesse processo as variabilidades presentes em uma Linha de Produtos, ou em um Domínio, sejam levadas em consideração, uma vez que tais variabilidades nortearão o desenvolvimento de produtos ou aplicações específicos.

No ambiente Odyssey, o processo de EA utiliza o recurso de rastreabilidade existente entre os artefatos construídos no domínio. Esta rastreabilidade ocorre por meio dos artefatos dos diferentes níveis de abstração, que se relacionam entre si, como apresentado na Figura 5.43.

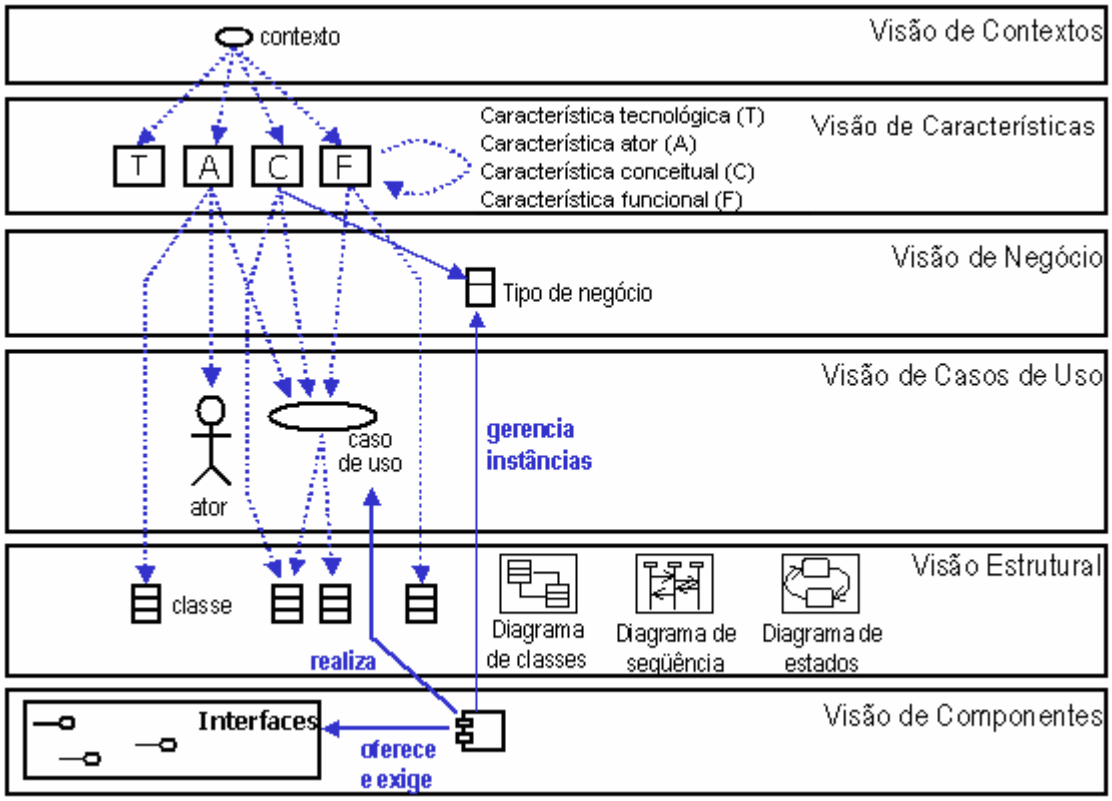



**Figura 5.43 – Rastreabilidade no ambiente Odyssey**
**(Adaptado de (TEIXEIRA, 2003))**

Conforme já mencionado na seção 5.2, os elementos de mais alto nível de abstração no Odyssey são os Contextos, seguidos das Características. Um Contexto subdivide um domínio, e está diretamente relacionado às características. A partir das características, outros elementos são rastreados (ex. casos de uso, tipos de negócio, classes, componentes).

A partir da seleção de contextos e características, é possível fazer um recorte no domínio, trazendo todos os elementos que são rastreáveis a partir deles, como apresentado na Figura 5.44. Assim, as propriedades e relacionamentos dos artefatos reutilizados são mantidos da mesma forma que foram especificados no modelo do domínio, possibilitando que, posteriormente, ajustes sejam feitos para refletir outros detalhes, específicos da aplicação.

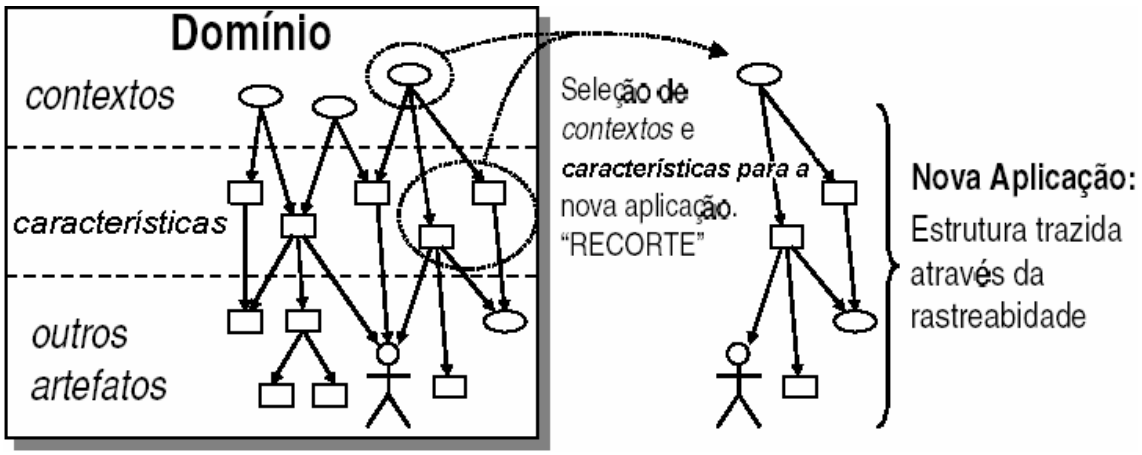



**Figura 5.44 – Reutilização de artefatos por meio da EA**
**(TEIXEIRA, 2003)**

Com o uso da notação Odyssey-FEX, surgiu a necessidade de adaptação do processo de EA no ambiente Odyssey, para contemplar a variabilidade dos elementos, bem como a interpretação das regras de composição.

A Figura 5.45 apresenta algumas classes relevantes para a implementação do processo de EA. Tais classes fazem parte do pacote *Application* do Odyssey, e foram modificadas para contemplar a instanciação correta de uma aplicação, de acordo com a notação Odyssey-FEX.

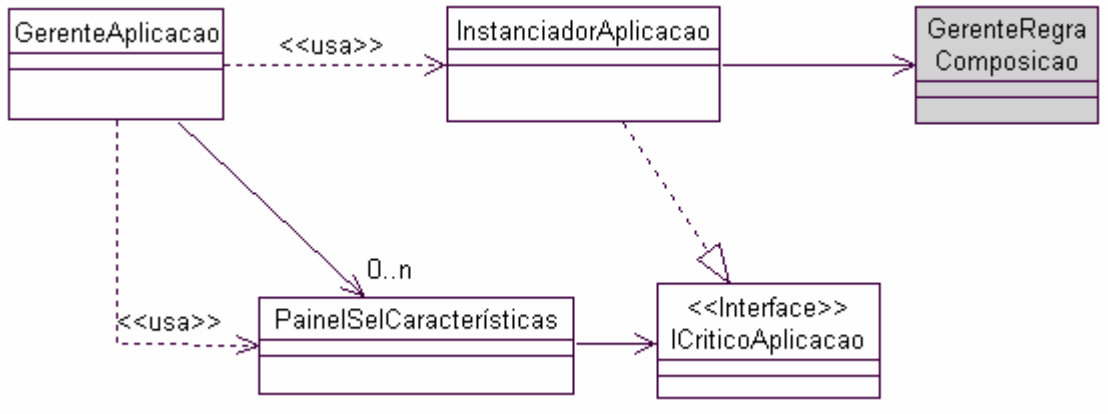



**Figura 5.45 – Classes do pacote *Application* no ambiente Odyssey**

A classe *GerenteAplicacao* é responsável pela identificação das características de acordo com o(s) contexto(s) escolhido(s). Uma vez escolhido o contexto, as características correspondentes são apresentadas ao desenvolvedor, para que sejam selecionadas. Esta apresentação é de responsabilidade da classe *PainelSelCaracteristicas*. No entanto, a maior responsabilidade no processo de EA é atribuída à classe *InstanciadorAplicacao*. Esta classe é responsável por realizar o recorte no domínio, i.e., encontrar e copiar os artefatos que são rastreáveis a partir das características selecionadas. Esta classe implementa, ainda, a interface *ICriticoAplicacao,* que é uma Interface para chamadas de críticas na política de seleção de características. Por meio da referência da classe *InstanciadorAplicacao* para a classe *GerenteRegraComposicao*, é possível chamar o método de interpretação das regras nas quais as características selecionadas estão envolvidas Desse modo, *InstanciadorAplicacao* é responsável por validar a seleção de características do domínio, ativando o processo de interpretação das regras de composição, detectando eventuais quebras de regras e relacionamentos, bem como inconsistências como a não-seleção de uma característica mandatória.

A Figura 5.46. apresenta a seleção de contextos, que dá início ao processo de seleção de artefatos do domínio no ambiente Odyssey.

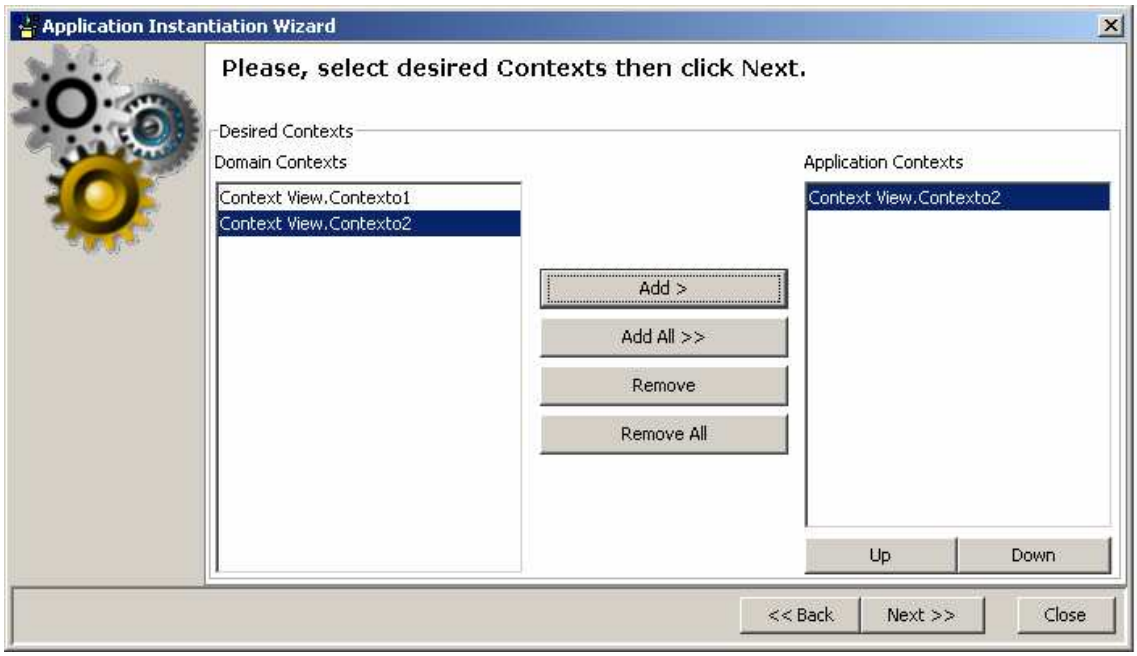



**Figura 5.46 – Seleção de contextos para instanciação de aplicação no ambiente Odyssey**

Uma vez selecionado um ou mais contextos, as características a ele(s) relacionadas serão apresentadas. Neste momento, o desenvolvedor recebe informações a

respeito de tais características, como mostra a janela apresentada na Figura 5.47. Na Figura 5.47 é possível observar, à esquerda, uma árvore de características que podem ser selecionadas. A seleção de uma característica da árvore implica na inclusão desta - e conseqüentemente, dos artefatos a ela relacionados - na nova aplicação ou produto a ser desenvolvido. Por convenção, as características que são mandatórias no domínio já se encontram selecionadas. As características não selecionadas na árvore representam as características opcionais no domínio, cabendo ao desenvolvedor selecionar dentre estas as características desejadas.

As características que representam um ponto de variação são apresentadas em um nível hierárquico (na figura, característica *Jogos*), tendo suas variantes como elementos subordinados a estas na árvore (características *Car Racer* e *Snake*). Assim como nas outras características, pontos de variação ou variantes que sejam mandatórios já se encontram selecionados.

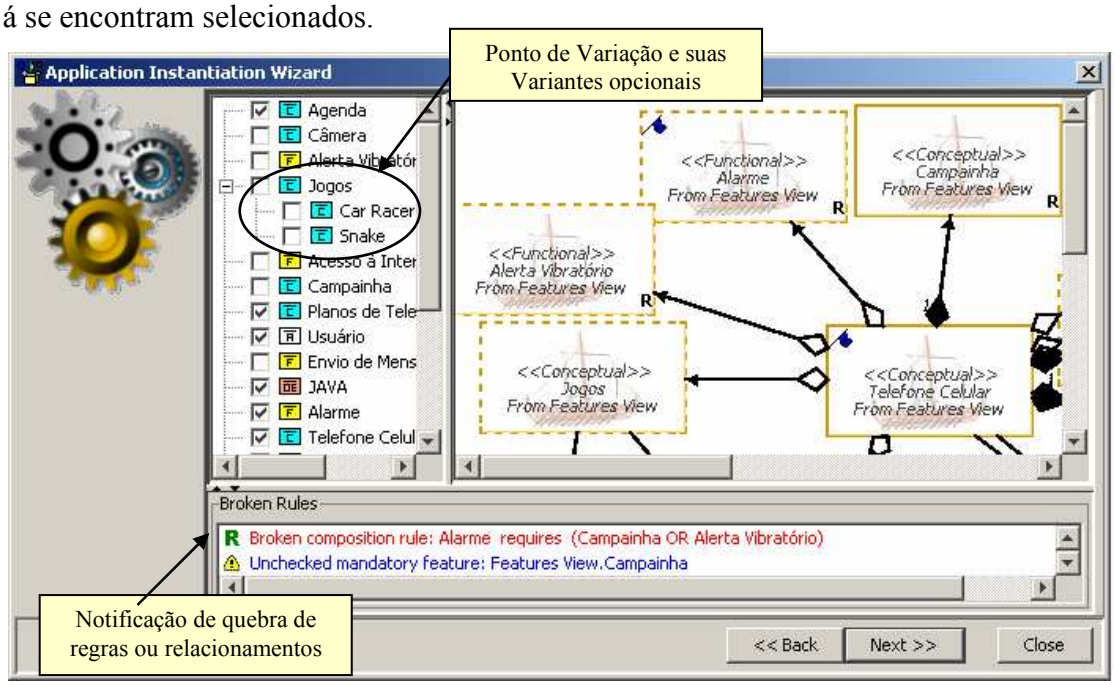



**Figura 5.47 – Seleção das características do domínio para instanciação de aplicação**

Na mesma tela, à direita, é apresentado o modelo de características que está sendo avaliado. À medida que as características são selecionadas na árvore, as características do modelo são assinaladas - como pode ser observado nas características *Alarme* e *Telefone Celular*, na figura - oferecendo, assim, maior visibilidade ao desenvolvedor. É possível ainda observar, na parte inferior da tela, uma área de texto, intitulada *Broken Rules*, onde são apresentadas possíveis quebras de regras de composição, ou quebras de relacionamento entre características, ocasionadas pela seleção ou não-seleção de características em conjunto. Esta área pode ser ocultada, de

acordo com a vontade do desenvolvedor. Dessa maneira, é possível apresentar informações úteis ao processo de EA, porém de forma a se evitar a sobrecarga de informações ao desenvolvedor. Como exemplo, tem-se a quebra da regra de composição inclusiva "Alarme **requires** (Campainha OR Alerta Vibratório)", causada pela seleção da característica *Alarme* e a não-seleção da característica *Campainha* ou *Alerta Vibratório*. Além disso, neste domínio a característica *Campainha* é mandatória, e esta não foi selecionada para a aplicação, ocasionando uma notificação para o desenvolvedor (mensagem "*Unchecked mandatory feature: Features View.Campainha*").

Ao término do processo de seleção de artefatos, uma nova e específica aplicação pode ser modelada, tendo como base os artefatos, propriedades e relacionamentos oriundos do modelo de domínio. Neste momento, ajustes podem ser feitos, ou mesmo aspectos específicos da aplicação podem ser modelados, possibilitando assim a instanciação de uma nova aplicação, ou seja, o desenvolvimento de um novo produto.

As regras de composição são interpretadas a cada seleção de características na árvore. Isto oferece ao desenvolvedor a informação de que existem características dependentes entre si que não estão sendo incluídas em conjunto na aplicação, ou características mutuamente exclusivas que estão sendo incluídas em conjunto na aplicação. Além disso, é necessário verificar outras inconsistências, tais como: uma característica mandatória não estar sendo incluída na aplicação e quebra de relacionamentos entre características. Assim, a interpretação das regras de composição, juntamente com a interpretação da variabilidade das características e a instanciação correta da aplicação, constituem o principal esforço de implementação no que diz respeito ao processo de EA, no contexto deste trabalho.

## 5.4 - IMPLEMENTAÇÃO DO SISTEMA DE CRÍTICAS

No capítulo 4, foram discutidos os conceitos e necessidades de um sistema de críticas que verifique a consistência entre elementos de um modelo (consistência intra-modelos), bem como a consistência entre os diversos modelos que participam do desenvolvimento de uma Linha de Produtos (consistência inter-modelos).

No intuito de viabilizar a proposta apresentada no capítulo anterior, foi adaptado, neste trabalho, um sistema de críticas intra-modelo, para a atuação no modelo de características. No contexto do ambiente Odyssey, existe um mecanismo de críticas, denominado Oráculo (DANTAS *et al.*, 2001), que é uma ferramenta auxiliar à

ferramenta de diagramação do ambiente. O Oráculo monitora a inclusão, remoção e edição de elementos em modelos no ambiente Odyssey, enviando notificações aos desenvolvedores a cada vez que as regras de boa formação de um modelo são desrespeitadas. Inicialmente, ele verificava modelos UML seguindo as regras de boa formação do seu metamodelo. Com a proposta da notação Odyssey-FEX, o Oráculo foi estendido para, também, verificar a consistência dos modelos de características.

Na Figura 5.48, é apresentada a estrutura básica do Oráculo, em nível de projeto. A classe *GerenteCriticas* é a entidade responsável pelo controle de execução do mecanismo, delegando a verificação individualmente para cada instância da classe *Critica* cadastrada e ativa. Cada crítica, por sua vez, contém um nome genérico e um conjunto de regras que caracterizam aquela categoria de crítica, além do agente que irá realizar a verificação. Um *Agente* é uma classe que contém a implementação das regras pertencentes a uma crítica. Cada instância da classe *Regra* representa a definição de uma heurística ou regra de consistência que deve ser verificada. Possui ainda um nome, conselhos associados à regra e o método que implementa a sua verificação. Relacionado ao gerente de críticas, existe um objeto especial responsável por carregar e instanciar as regras e críticas, representado, no modelo, pela classe *CriticaLoader* (DANTAS *et al*., 2001).

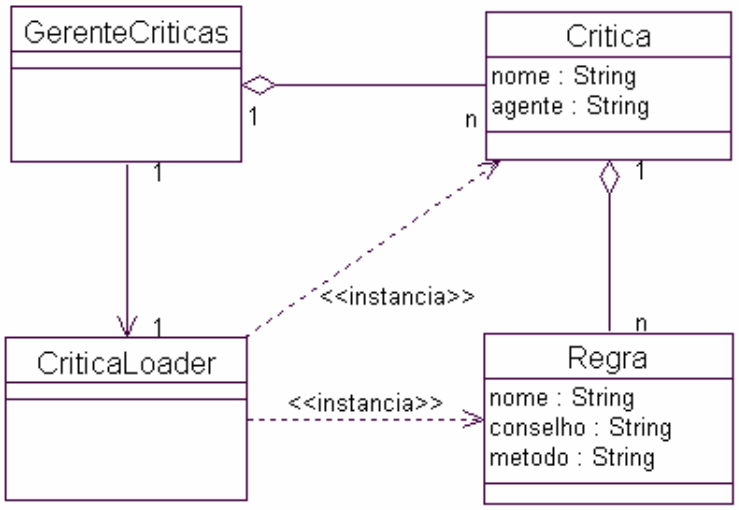



**Figura 5.48 - Estrutura básica do Oráculo (DANTAS *et al.*, 2001)**

A fim de implementar a verificação de consistência em modelos, foi selecionado um subconjunto de regras da UML, cujas críticas associadas foram divididas em categorias. Cada categoria é implementada por um agente: *UMLClasse, UMLEstado, UMLUseCases* e *UMLVisibilidade*. Para a verificação de consistência do modelo de características foi criado um novo agente, denominado *Features*, que contém a

implementação dos métodos relativos às críticas deste modelo. As regras utilizadas nesta implementação representam um subconjunto de regras extraídas do metamodelo da notação Odyssey-FEX, como apresentado no Capítulo 3 e em (OLIVEIRA *et al.*, 2005a).

As regras foram incluídas na estrutura do Oráculo por meio do mecanismo de cadastramento de regras já existente. Ao invés da criação de críticas e regras amarradas ao código, no Oráculo é utilizado um arquivo descritor do tipo XML (*Extended Markup Language*) (XML, 2005), que contém todas as informações sobre as críticas e regras. Este arquivo tem formatação especial, permitindo que críticas e regras sejam facilmente reconfiguradas, bem como facilidades de adição e remoção desses elementos do sistema. Assim, para modificar o conselho associado a uma regra, ou incluir novas regras, por exemplo, é necessário apenas modificar ou adicionar texto no arquivo descritor, sem a necessidade de alteração do código fonte. As novas entradas incluídas no arquivo são, automaticamente, reconhecidas pelo mecanismo de críticas durante a leitura do arquivo descritor. A Figura 5.49 mostra um trecho do arquivo XML, já com as críticas relativas ao modelo de características.


**Figura 5.49 – Arquivo em XML de elementos do Oráculo, relativos ao modelo de características**

O Oráculo interage com o desenvolvedor por meio de mensagens enviadas no momento da ação que gerou a inconsistência. A Figura 5.50 exemplifica a atuação do mecanismo. Neste caso, o desenvolvedor está tentando associar uma variante mandatória a um ponto de variação opcional. Isso gera uma inconsistência, uma vez que se um ponto de variação não estiver presente na aplicação, as variantes a ele associadas também não poderão estar. Neste caso, nenhuma de suas variantes pode ser mandatória.

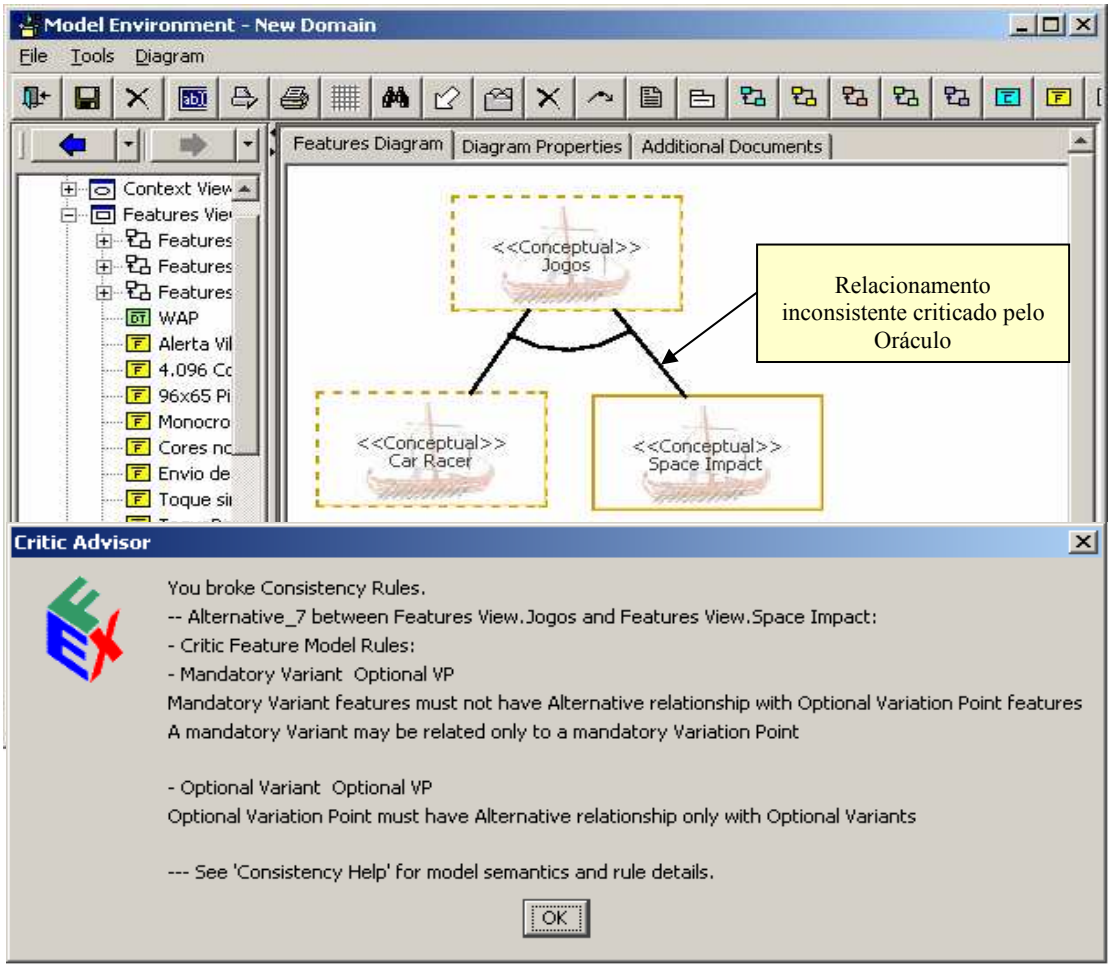



**Figura 5.50 – Janela de resposta do Oráculo**

É importante salientar que o desenvolvedor pode decidir se o mecanismo de notificação estará ativo, ou mesmo se determinadas críticas ou conjunto de críticas estarão ativos, no momento da modelagem. Desta forma, o desenvolvedor tem a opção de não receber as mensagens no momento da ação, mas gerar um relatório ao final da modelagem, com informações sobre inconsistências existentes no modelo.

A partir desta implementação, o Oráculo oferece apoio à detecção de inconsistências intra-modelos. Assim, evita-se, entre outras coisas, a propagação de erros de modelagem para as diversas fases de modelagem, bem como aplicações de um domínio. Espera-se, com essa implementação, prover um maior apoio à construção dos

artefatos de domínio, de maneira coerente, aumentando assim a produtividade e a qualidade do desenvolvimento de software **para** e **com** reutilização.

## 5.5- ESTUDO DE OBSERVAÇÃO

Com o objetivo de caracterizar a viabilidade das regras de boa formação do metamodelo da notação Odyssey-FEX, foi realizado um estudo observacional. De acordo com (SHULL *et al*., 2001), o termo "observacional" é usado para definir o tipo de estudo onde o participante realiza alguma tarefa enquanto é observado por um experimentador. A finalidade da observação é coletar dados sobre como determinada tarefa em particular é realizada. As técnicas observacionais podem ser usadas para se obter uma compreensão detalhada de como um processo novo é aplicado.

O objetivo do estudo pode ser apresentado de acordo com o modelo descrito em (WOHLIN *et al.*, 2000), e apresentado na Tabela 5.11 :

**Tabela 5.11 – Definição de objetivos do estudo de observação**

| |
|---|
| **Analisar** as regras de boa formação da notação Odyssey-FEX |
| **Com o propósito de** caracterizar a viabilidade as regras |
| **Com respeito à** diminuição de inconsistências no modelo de características |
| **Do ponto de vista do** Engenheiro de Software |
| **No contexto** do ambiente Odyssey |

Neste estudo, os indivíduos selecionados utilizaram as regras de boa formação propostas, no intuito de minimizar as inconsistências no modelo de características do domínio. Pretendia-se observar se, com o uso das regras de boa formação, seria criado um modelo de características mais consistente com a especificação de requisitos apresentada.

Os passos definidos para este estudo de observação foram: a) definição dos participantes do estudo, b) treinamento dos participantes a respeito da notação Odyssey-FEX, definida nesta dissertação; c) treinamento dos participantes a respeito das regras de boa formação do metamodelo da notação Odyssey-FEX ; d) utilização das regras de boa formação; e e) avaliação do estudo de observação.

### 5.5.1 – DEFINIÇÃO DOS PARTICIPANTES

Para a realização deste estudo, foram convidados quatro voluntários. Dois são alunos do curso de Ciência da Computação da Universidade Federal do Rio de Janeiro.

Os outros dois são alunos do Programa de Engenharia de Sistemas e Computação, da Coordenação dos Programas de Pós-Graduação em Engenharia - COPPE/UFRJ, sendo um de mestrado e outro de doutorado. Os participantes, embora já tivessem algum contato com a abordagem de características, não tinham experiência em modelagem de características, em especial utilizando a notação Odyssey-FEX, nem na utilização do ferramental apresentado. Foram utilizados os documentos dos Anexos I e V para, respectivamente, obter o consentimento e a caracterização dos participantes do estudo.

### 5.5.2 – TREINAMENTO DA NOTAÇÃO ODYSSEY-FEX

A notação Odyssey-FEX foi distribuída previamente por e-mail aos participantes do estudo. Tal distribuição foi importante para que os participantes tivessem o conhecimento a respeito desta taxonomia para um melhor entendimento das regras de boa formação propostas. Juntamente com a notação, foi disponibilizado parte de um modelo de características do domínio de ensino colaborativo (*Computer Supported Cooperative Learning - CSCL*), visando auxiliar os participantes na compreensão das propriedades definidas pela taxonomia utilizada no estudo de observação. O material distribuído aos participantes pode ser encontrado no Anexo II.

### 5.5.3 – TREINAMENTO DAS REGRAS DE BOA FORMAÇÃO

Conforme descrito na seção 3.3.2, foi definido um conjunto regras de boa formação para a modelagem das características utilizando a notação Odyssey-FEX. Um conjunto destas regras foi utilizado neste estudo de observação.

Os participantes do estudo tiveram acesso a estas regras por meio do material apresentado no Anexo IV.

### 5.5.4 – UTILIZAÇÃO DAS REGRAS DE BOA FORMAÇÃO

Foi apresentado aos participantes do estudo um modelo de características do domínio de telefonia móvel, desenvolvido no ambiente de reutilização Odyssey, baseado na notação Odyssey-FEX, como apresentado em capítulos anteriores. O modelo contempla características conceituais, tecnológicas, funcionais, cada qual com suas obrigatoriedades, variabilidades, relacionamentos e, em alguns casos, regras de composição, conforme apresenta o Anexo III.

O estudo consistiu nas seguintes atividades: a) apresentação aos participantes de uma parte do modelo de características do domínio de telefonia móvel, b) apresentação de uma especificação que representa o refinamento do domínio de novos produtos a

serem incluídos neste domínio (Anexo III), e c) apresentação das regras de boa formação da notação Odyssey-FEX (Anexo IV). De posse destes documentos, os participantes devem modelar o domínio, a fim de contemplar os novos requisitos em características que eventualmente sejam reutilizadas. Os participantes podem consultar a Lista de Regras de Boa Formação, distribuída com o material do estudo.

O resultado esperado desta atividade é um modelo de características do domínio de telefonia móvel, com características que correspondem ao refinamento do domínio, de acordo com o documento entregue aos participantes. O modelo esperado é apresentado na Figura 5.51.

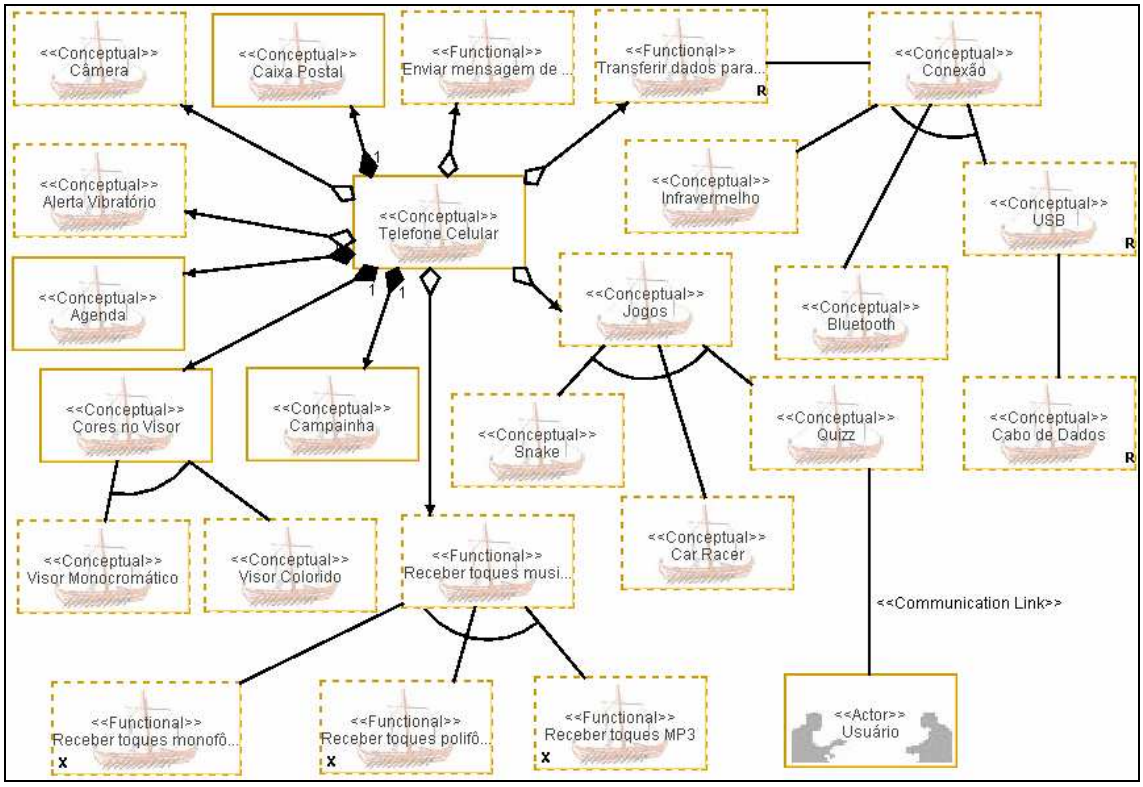



**Figura 5.51 – Modelo de características esperado ao final do estudo de observação**

A ação dos participantes é gravada em vídeo para análise posterior. A partir do vídeo gravado e do modelo resultante, é analisada a viabilidade das regras de boa formação do metamodelo da notação Odyssey-FEX. Tal análise é feita comparando-se o número de inconsistências geradas pelos participantes durante a atividade de modelagem, o número de inconsistências que são notificadas ao participante pelo sistema de críticas, e o número de inconsistências mantidas no modelo de características após a notificação. Para auxiliar tal observação, é preenchido o formulário de avaliação apresentado no Anexo V.

### 5.5.5 – AVALIAÇÃO DO ESTUDO DE OBSERVAÇÃO

O estudo de observação foi executado em quatro dias, um para cada participante, individualmente. Os resultados obtidos a partir da análise dos vídeos são sumarizados na Tabela 5.12. É importante observar que as inconsistências contabilizadas na Tabela 5.12 se referem à notação e não à semântica do modelo.

**Tabela 5.12 – Resultado da análise dos modelos de características**

| | Participante 1 | Participante 2 | Participante 3 | Participante 4 |
|---|---|---|---|---|
| **Inconsistências geradas** | 6 | 4 | 5 | 8 |
| **Inconsistências não notificadas** | 2 | 1 | 0 | 1 |
| **Inconsistências notificadas** | 4 | 3 | 5 | 7 |
| **Inconsistências corrigidas após notificação** | 4 | 3 | 2 | 4 |
| **Inconsistências remanescentes no modelo** | 1 | 1 | 3 | 4 |

Os modelos finais analisados ficaram bem próximos do modelo esperado, apresentado na Figura 5.51. No entanto, embora seja previsto na notação Odyssey-FEX a criação de características de entidade, não foi criada, em nenhum dos casos, a característica *Usuário*, nem tampouco o relacionamento Ligação de Comunicação com outras características, conforme previsto na especificação. Assim, não foi possível avaliar as regras relativas ao relacionamento Ligação de Comunicação.

As regras de composição foram criadas corretamente, no entanto, houve divergência na interpretação dos participantes: um dos participantes modelou uma regra de composição inclusiva, como previsto na especificação, outro modelou um relacionamento direto entre as características. Em ambos os casos, a modelagem está correta, do ponto de vista da notação. O terceiro participante não incluiu no modelo a característica necessária para a criação da regra. O quarto participante não modelou corretamente a relação entre as características que estariam envolvidas na regra de composição, cometendo um erro semântico (relacionamento de composição entre as características *USB* e *Cabo de Dados*) e um notacional (composição onde o "todo" é opcional e a "parte" mandatória). O erro de notação foi devidamente notificado pelo sistema de críticas. No entanto, mesmo tendo sido notificado, o participante não corrigiu o erro e nem justificou sua atitude.

Pôde-se observar, durante a análise dos vídeos, que existiam determinadas inconsistências relativas à notação que não haviam sido previstas pelo sistema de críticas. No entanto, tais inconsistências foram levadas em consideração na Tabela 5.12, na linha de inconsistências não notificadas.

Dentre os erros previstos pelo sistema de críticas, pôde-se observar a devida notificação em cada caso. Os participantes 1 e 2 corrigiram os erros notificados pelo sistema de críticas, deixando o modelo totalmente isento de inconsistências de modelagem, do ponto de vista da notação. As inconsistências remanescentes no modelo, nesses dois casos, são provenientes de inconsistências não notificadas pelo sistema de críticas. O participante 3 não corrigiu algumas das inconsistências notificadas, permanecendo estas no modelo. O participante justificou sua atitude, alegando ter julgado que, a exemplo de outros sistemas de críticas, o usuário seria impedido de cometer o erro, e não apenas avisado. Por ter pensado de tal forma, julgou que, uma vez não sendo impedido de prosseguir na modelagem, não havia necessidade de corrigir os erros notificados. O participante 4, cujo modelo final menos se aproximou do modelo esperado, alegou ter tido dificuldades de entendimento dos conceitos de ponto de variação e variante, e as regras referentes à opcionalidade destes. No entanto, embora não tenha corrigido no modelo, o participante foi notificado pelo sistema de críticas em todos os momentos em que as inconsistências relativas à notação ocorreram.

O questionário apresentado no Anexo V foi utilizado para a obtenção da opinião dos participantes. O resultado é sumarizado na Tabela 5.13.

De acordo com os resultados apresentados na Tabela 5.13, nota-se que as regras foram entendidas pela maioria dos participantes. O participante que respondeu "parcialmente" na questão 1, justificou que a dificuldade de entendimento das regras se deve à inexistência de exemplos. Na questão 4, embora o participante tenha respondido "sim", o vídeo gravado não mostra, em nenhum momento, a criação de um ponto de variação mandatório com variantes opcionais. Na questão 8, dois participantes expressaram a necessidade da criação das regras e dois não expressaram tal necessidade. No entanto, um participante, que respondeu "não", criou as regras. Na questão 10, o participante que respondeu "parcialmente" justificou que achou inconveniente uma das mensagens do sistema de críticas, que notifica um erro que seria consertado no passo seguinte da modelagem. De fato, o sistema de críticas possui uma limitação de implementação, no caso desta mensagem específica, que notifica que uma variante não pode estar desassociada de um ponto de variação. Tal notificação ocorre no momento

que o usuário informa que a característica é variante, antes que o usuário possa associá-la ao respectivo ponto de variação. Na questão 11, o participante que respondeu "não" alega que, para ele, o entendimento da regra ficou comprometido por não ter impedido a ação do usuário, embora o erro tenha sido mostrado na tela.

**Tabela 5.13 – Avaliação do questionário preenchido pelos participantes do estudo**

| Questão | Sim | Não | Parcialmente |
|---|---|---|---|
| 1) As regras de boa formação da notação Odyssey-FEX são de fácil entendimento? | 3 | 0 | 1 |
| 2) Em algum momento você sentiu a necessidade de associar uma variante a mais de um ponto de variação? | 0 | 4 | 0 |
| 3) Em algum momento você desejou relacionar uma variante MANDATÓRIA a um ponto de variação OPCIONAL? | 0 | 4 | 0 |
| 4) Em algum momento você desejou relacionar um ponto de variação MANDATÓRIO com uma variante OPCIONAL? | 1 | 3 | 0 |
| 5) Em algum momento você desejou criar uma composição entre um elemento mandatório e um elemento opcional? | 0 | 4 | 0 |
| 6) Em algum momento você desejou utilizar o relacionamento Ligação de Comunicação entre características diferentes de Entidade e Domínio? | 0 | 4 | 0 |
| 7) Em algum momento você desejou criar uma regra de composição exclusiva envolvendo características dependentes entre si ou vice versa? | 0 | 4 | 0 |
| 8) Em algum momento você desejou criar uma regra de composição exclusiva envolvendo características opcionais? | 2 | 2 | 0 |
| 9) Em algum momento você desejou criar uma regra de composição inclusiva cujo antecedente era mandatório e o conseqüente opcional? | 0 | 4 | 0 |
| 10) O suporte computacional existente contemplou adequadamente a utilização das regras de boa formação do modelo de características? | 3 | 0 | 1 |
| 11) Durante a construção do modelo de características foi fácil identificar qual regra de boa formação estava sendo violada? | 3 | 1 | 0 |

A realização desse estudo de observação foi importante, pois identificou regras de boa formação que não haviam sido previstas anteriormente, nem implementadas no sistema de críticas, a saber:

- Não deve existir relacionamento de Associação entre um ponto de variação e uma variante. Apenas o relacionamento Alternativo deve ocorrer entre tais características.
- Não deve existir mais de um relacionamento do tipo Alternativo entre as mesmas características.
- Uma variante deve ser da mesma categoria de seu ponto de variação, i.e., não deve existir relacionamento Alternativo entre características de categorias distintas (conceitual e funcional, por exemplo).

Com o estudo de observação, pôde-se observar que as regras de boa formação da notação Odyssey-FEX são viáveis para auxiliar a diminuição de inconsistências em modelos de características. No entanto, as regras não garantem essa diminuição, uma vez que cabe ao usuário corrigir o erro quando da notificação pelo sistema de críticas.

## 5.6- CONSIDERAÇÕES FINAIS

Neste capítulo foi descrita a implementação realizada no contexto do ambiente Odyssey, com o intuito de viabilizar as propostas apresentadas neste trabalho. Foram mostradas as adaptações realizadas no ambiente, em nível de modelo conceitual, bem como o funcionamento do sistema, quanto ao apoio automatizado à utilização da notação Odyssey-FEX.

As regras de composição com operadores booleanos, sua criação e impacto no processo de Engenharia de Aplicação, ou desenvolvimento do produto, também foram apresentados, bem como o diagramador utilizado para a construção do modelo de características. Por fim, foi apresentado o sistema de críticas para a verificação de consistência intra-modelo, denominado Oráculo, e a adaptação necessária para sua utilização no modelo de características.

Na implementação das regras de composição, expressões redundantes não são simplificadas. Embora tais expressões pudessem ser simplificadas, optou-se por oferecer a maior expressividade possível ao especialista que esteja modelando as regras, ao invés de impedir a modelagem. Além disso, não há como afirmar que tal tipo de regra não faria sentido para nenhum domínio específico. Assume-se, portanto, que o especialista do domínio tem conhecimento do que está sendo feito, e se uma regra redundante é modelada, o especialista deve ter razões que justifiquem tal fato.

Embora a implementação aqui descrita possibilite a utilização da notação Odyssey-FEX, de acordo com a proposta apresentada no capítulo 3, o ambiente Odyssey necessita ainda de uma implementação que contemple a verificação de consistência inter-modelos, apresentada no capítulo 4. Devido à amplitude da proposta, priorizou-se a implementação da notação Odyssey-FEX por completo, incluindo a criação e interpretação das Regras de Composição, e do sistema de críticas para verificação da consistência intra-modelos. No entanto, um subconjunto das heurísticas de mapeamento para a verificação de consistência inter-modelos foram propostas e

implementadas para apoiar o mapeamento entre características e tipos de negócio, no contexto do processo *CBD-Arch-DE* (OLIVEIRA *et al.*, 2005b).

Dessa maneira, acredita-se possibilitar ao desenvolvedor uma modelagem de características completa e consistente em relação à notação proposta, bem como ampliar a utilização do ambiente Odyssey para o desenvolvimento de uma Linha de Produtos de Software, ou de um Domínio.

# CAPÍTULO VI
# CONCLUSÕES E TRABALHOS FUTUROS

## 6.1 – CONTRIBUIÇÕES

Técnicas de reutilização de software, como Engenharia de Domínio e Linha de Produtos, incorporam em seus processos uma etapa de análise, que visa explicitar a variabilidade, i.e, aspectos comuns e específicos inerentes a uma família de sistemas. Para tanto, existe a necessidade de representação dessa variabilidade em modelos que representam uma família de sistemas, durante todo o desenvolvimento do software.

A partir da análise de notações para a representação de variabilidades em modelos de características existentes na literatura, percebeu-se a existência de deficiências em tais notações, no que diz respeito à representação dos conceitos relacionados à variabilidade. Essa representação se mostrou incompleta e, por vezes, ambígua, podendo ocasionar uma modelagem equivocada da família de sistemas representados por tais notações.

Com o objetivo de minimizar os problemas ocasionados pela representação incompleta das variabilidades nos diversos artefatos inerentes à reutilização de uma família de sistemas, algumas propostas foram apresentadas nesta dissertação. Ao final da pesquisa, algumas contribuições foram obtidas, como descrito a seguir.

a) A d*efinição de um conjunto de requisitos para a representação da variabilidade em modelos de características*. Conceitos inerentes à modelagem de variabilidades foram organizados em um conjunto mínimo de requisitos necessários para que os aspectos comuns e específicos de uma família de sistemas sejam evidenciados e documentados. Dessa forma, o presente trabalho oferece subsídios para que outras notações possam vir a ser analisadas.

b) *Formalização da semântica dos conceitos de variabilidade por meio de um metamodelo*. O metamodelo, que dá origem à notação Odyssey-FEX, vem minimizar possíveis ambigüidades e erros de interpretação na modelagem de características, ao documentar de maneira sistemática as informações necessárias à construção do modelo. Além disso, a existência de um metamodelo possibilita uma série de outros benefícios, como, por exemplo, verificação de consistência entre modelos, apoio à transformação de modelos ou troca de informação entre ferramentas.

c) *Definição de uma notação para a modelagem de características mais abrangente, no que diz respeito à representação de variabilidades*. A notação Odyssey-FEX, proposta neste trabalho, foi elaborada a partir do refinamento da notação proposta por Miler (2000), e de elementos das notações analisadas que melhor representavam os conceitos envolvidos. Além disso, apresenta uma semântica mais rica, no sentido de ser útil para a representação da terminologia do domínio, a partir do uso de ícones e relacionamentos mais expressivos. No que diz respeito à representação de variabilidade, a notação Odyssey-FEX traz, como contribuição, as Regras de Composição Complexas, que permitem o uso de operadores booleanos para representar dependência e mútua exclusividade entre quaisquer conjuntos de características do domínio, aumentando ainda mais o poder de expressividade do modelo de características.

d) *Definição de um conjunto de regras para a verificação de consistência intra-modelo*. As regras de boa formação do metamodelo da notação Odyssey-FEX serviram de base para a adaptação do sistema de críticas Oráculo (DANTAS *et al.*, 2001), para a verificação de consistência do modelo de características. Dessa maneira, a introdução de erros de modelagem, tão comum, principalmente na fase de análise, é minimizada, conforme evidenciado pelo estudo observacional apresentado nesta dissertação. Assim, evita-se que tais erros sejam propagados para momentos posteriores no ciclo de vida do software.

e) *Definição de heurísticas para a propagação das variabilidades, representadas no modelo de características, para o modelo de classes.* A representação da variabilidade deve estar coerente entre os modelos que representam a mesma família de sistemas sob diferentes perspectivas. Assim, é fundamental que a variabilidade representada no modelo de características, considerado o modelo de mais alto nível de abstração, seja propagada para outros modelos de nível de abstração mais baixo. Neste trabalho, foi dado um passo inicial nesse sentido, por meio da proposta de heurísticas para a propagação da variabilidade para o modelo de classes, originadas do mapeamento entre os elementos do metamodelo de características e os elementos do metamodelo da UML (OMG, 2005). Além disso, o presente trabalho serviu de base para a definição de novos conjuntos de heurísticas para a propagação de variabilidades para outros modelos, tais como casos de uso, tipos de negócio e componentes, no contexto do processo *CBD-Arch-DE*, desenvolvido em (BLOIS, 2006). Dessa forma, a definição dessas heurísticas é uma contribuição no sentido de orientar uma representação mais coerente nos diversos modelos que constituem uma família de sistemas.

Toda a implementação da proposta foi realizada no contexto do ambiente Odyssey, viabilizando a concretização das idéias propostas em um ambiente de reutilização e está disponível em (ODYSSEY, 2005).

## 6.2 – LIMITAÇÕES E TRABALHOS FUTUROS

No decorrer desta pesquisa, algumas limitações foram detectadas. Uma delas diz respeito à **cardinalidade na notação Odyssey-FEX.** A notação proposta não utiliza cardinalidades para as características, seus pontos de variação e variantes. Isso se deve ao fato de que é possível inferir as cardinalidades **0..1, 1, 0..n, 1..n** por meio da combinação de classificações, como descrito anteriormente, na seção 3.5. No entanto, a notação não cobre cardinalidades representadas por intervalos fixos, (ex. **0..3, 1..4**). Esse fato é considerado uma limitação da abordagem, uma vez que, na tentativa de não ocasionar uma sobrecarga de informações visuais ao usuário da notação, optou-se por não implementar as cardinalidades graficamente. No entanto, um estudo pode ser realizado no sentido de se identificar a melhor maneira de representar os intervalos fixos na notação.

Uma outra limitação da abordagem é referente às **Regras de Composição com Expressões Valoradas.** As regras de composição definidas neste trabalho expressam dependência e mútua exclusividade entre características, e podem ser classificadas de "simples", quando existem somente expressões literais envolvidas, ou "complexas", quando há um operador booleano. No entanto, não é possível representar expressões que contenham valores, dando origem a restrições do tipo "Ar Condicionado **requer** Cavalo-Vapor > 100". Essa limitação da abordagem se deve à ênfase que foi dada à criação de regras que atendessem ao requisito de expressar dependência entre quaisquer conjuntos de características, cobrindo uma deficiência das outras notações analisadas.

Embora a pesquisa seja ampla, é sabido que a modelagem de variabilidades é um assunto que ainda permite uma certa exploração. Desse modo, alguns pontos passíveis de continuidade deste trabalho foram identificados, a saber:

- **Estabelecimento de heurísticas para propagação de variabilidade entre outros artefatos de software:** Com as heurísticas para a representação de variabilidades do modelo de características para o modelo de classes, um primeiro passo foi dado no sentido de auxiliar a representação coerente das variabilidades nos diversos artefatos inerentes a uma família de sistemas. No entanto, não são estabelecidas, nesta

dissertação, heurísticas para a representação de variabilidades para outros artefatos de software. Heurísticas para a propagação das variabilidades para outros artefatos (ex. diagramas de casos de uso, tipos de negócio e componentes) são encontradas em (BLOIS, 2006), complementando esta abordagem. No entanto, heurísticas para artefatos, como por exemplo, diagrama de seqüência, podem ainda ser estabelecidas, tornando ainda mais consistente a representação da família de sistemas reutilizáveis.

- **Aplicação da proposta em projetos reais e mais complexos:** A complexidade presente nos modelos reais poderia validar de forma mais abrangente as regras de boa formação do modelo de características, as heurísticas de mapeamento e a eficiência da notação apresentadas neste trabalho. Assim, é relevante ressaltar a importância dessa aplicação em casos reais de desenvolvimento de uma família de sistemas.

Espera-se, contudo, que os resultados deste trabalho possam representar um passo a mais na busca por uma reutilização de software mais efetiva, possibilitando assim, alcançar os benefícios almejados pela comunidade de Engenharia de Software.

# REFERÊNCIAS BIBLIOGRÁFICAS


ANASTASOPOULOS, M., GACEK, C., 2001, "Implementing Product Line Variabilities". In: *Proceedings of the Symposium on Software Reusability* pp. 109 - 117 Toronto, Ontario, Canada

ARANGO, G., 1994, "Domain Analysis Methods". In: SCHÄFER, W., PRIETO-DIAZ, R., MATSUMOTO, M. (eds), *Software Reusability*, Chichester, Ellis Horwood.

ARANGO, G., PRIETO-DIAZ, R., 1991, "Introduction and Overview: Domain Analysis Concepts and Research Directions". In: G.ARANGO, R.P.-D.A. (eds), *Domain Analysis and Software Systems Modeling*, IEEE Computer Society Press.

ATKINSON, C., BAYER, J., BUNSE, C.*, et al.*, 2002, *Component-based product line engineering with UML*, Boston, Addison-Wesley Longman Publishing Co., Inc.

BECKER, M., 2003, "Towards a general model of variability in product families". In: *Proceedings of the 1st Workshop on Software Variability Management*, pp. 19–27, Groningen, The Netherlands., February.

BERTOLINO, A., FANTECHI, A., GNESI, S.*, et al.*, 2002, "Use Case Description of Requirements for Product Lines". In: *Proceedings of the International Workshop on Requirements Engineering for Product Lines (REPL'02)*, pp. 12–18, Essen, Germany, September, 2002.

BLOIS, A.P., BECKER, K., WERNER, C.M.L., 2004, "Um Processo de Engenharia de Domínio com foco no Projeto Arquitetural Baseado em Componentes". In: *IV Workshop de Desenvolvimento Baseado em Componentes*, pp. 15-20, João Pessoa, Paraíba, Brasil, Setembro.

BLOIS, A.P.T.B., 2006, *Uma Abordagem de Projeto Arquitetural Baseado em Componentes no Contexto da Engenharia de Domínio*, Tese de DSc., COPPE, UFRJ, Rio de Janeiro, Brasil (Em conclusão).

BOSCH, J., 2004, "Software Variability Management". In: *Proceedings of the 26th International Conference on Software Engineering (ICSE'04)*, pp. 720-721, Scotland, UK.

BRAGA, R.M.M., 2000, *Busca e Recuperação de Componentes em Ambientes de Reutilização de Software*, Tese de DSc., COPPE, UFRJ, Rio de Janeiro, Brasil.

CECHTICKY, V., PASETTI, A., ROHLIK, O.*, et al.*, 2004, "XML-based feature modelling". In: *Software Reuse: Methods, Techniques and Tools: 8th*

*International Conference, ICSR, Proceedings*, v. 3107, pp. 101–114, Madrid, Spain, July.

CLAUSS, M., 2001a, "Generic Modeling using UML Extensions for Variability". In: *DSVL 2001 (OOPSLA Workshop on Domain Specific Visual Languages), Proceedings*, pp. 11-18, Finland.

CLAUSS, M., 2001b, "Modeling variability with UML". In: *GCSE2001, Young researchers Workshop Proceedings*, pp. 226–230, Erfurt, Germany.

CLEMENTS, P.C., 1999, "Essential Product Line Practices". In: *Proceedings of the Ninth Workshop on Institutionalizing Software Reuse - WISR9*, Austin, TX, USA, January.

CZARNECKI, K., HELSEN, S., EISENECKER, U., 2004, "Staged configuration using feature models". In: *Software Product Lines: Third International Conference, SPLC 2004, Proceedings*, v. 3154, pp. 266–283, Boston, MA, USA, August 30-September 2.

CZARNECKI, K., HELSEN, S., EISENECKER, U.W., 2005, "Formalizing cardinality-based feature models and their specialization", *Software Process: Improvement and Practice*, v. 10, n. 1 (March), pp. 7-29.

DANTAS, A.R., CORREA, A.L., WERNER, C.M.L., 2001, "Oráculo: Um Sistema de Críticas para a UML". In: *XV Simposio Brasileiro de Engenharia de Software - SBES, Caderno de Ferramentas*, pp. 398-403, Rio de Janeiro, RJ , Brasil.

DANTAS, A.R., VERONESE, G.O., CORREA, A.L.*, et al.*, 2002, "Suporte a Padrões no Projeto de Software". In: *XVI Simpósio Brasileiro de Engenharia de Software*, pp. 450-455, Gramado, RS, Brasil, Outubro.

EGYED, A., 2001, "Scalable Consistency Checking Between Diagrams-The ViewIntegra Approach". In: *Proceedings of the 16th IEEE International Conference on Automated Software Engineering*, pp. 387-390, San Diego, USA, November.

GAMMA, E., HELM, R., JOHNSON, R.*, et al.*, 1995, *Padrões de Projeto - Soluções Reutilizáveis de Software Orientado a Objetos* Ed. Bookman.

GARG, A., CRITCHLOW, M., CHEN, P.*, et al.*, 2003, "An Environment for Managing Evolving Product Line Architectures". In: *Proceedings of International Conference on Software Maintenance*, pp. 358-369, Amsterdam, The Netherlands.

GIMENES, I.M.S., TRAVASSOS, G.H., 2002, "O Enfoque de Linha de Produto para Desenvolvimento de Software". In: *XXI Jornada de Atualização em Informática (JAI) – Evento Integrante do XXII Congresso da SBC*, pp. 1-31, Florianópolis, Brasil.

GOMAA, H., SHIN, M.E., 2002 "Multiple-View Meta-Modeling of Software Product Lines". In: *Proceedings of the Eighth IEEE International Conference on Engineering of Complex Computer Systems*, pp. 238 - 246, Maryland, USA.

GOMAA, H., SHIN, M.E., 2004, "A Multiple-View Meta-modeling Approach for Variability Management in Software Product Lines". In: *Software Reuse: Methods, Techniques and Tools: 8th International Conference, ICSR 2004, Proceedings*, v. 3107, pp. 274-285, Madrid, Spain, July.

GRADIENTE, 2005. In: http://www.gradiente.com/inc/ExibeGlossario.asp, acessado em 16/12/2005.

GRISS, M.L., FAVARO, J., D'ALESSANDRO, M., 1998, "Integrating feature modelling with the RSEB". In: *Proceedings of Fifth International Conference on Softwre Reuse - ICSR5*, pp. 76-85 Victoria, British Columbia, Canada.

HOEK, A.V.D., 2004, "Design-time product line architectures for any-time variability", *Science Computer Program*, v. 53, n. 3 (December 2004), pp. 285 - 304.

JOHN, I., MUTHIG, D., 2002, "Product Line Modeling with Generic Use Cases". In: *Workshop on Techniques for Exploiting Commonality Through Variability Managemen, Second Software Product Line Conference*, San Diego, USA, August.

JOHNSON, R.E., 1997, "Frameworks = (components + patterns) ", *Communications of the ACM* v. 40 n. 10 (October 1997), pp. 39-42

KANG, K.C., COHEN, S.G., HESS, J.A., *et al.*, 1990, *Feature-Oriented Domain Analysis (FODA) - Feasibility Study*, Software Engineering Institute (SEI), CMU/SEI-90-TR-21.

KANG, K.C., LEE, J., DONOHOE, P., 2002, "Feature-Oriented Product Line Engineering", *IEEE Software*, v. 9, n. 4 (Jul./Aug 2002), pp. 58-65.

KRUEGER, C.W., 2002, "Variation Management for Software Product Lines". In: *Proceedings of Second Software Product Line Conference, SPLC 2*, pp. 257-271, San Diego, CA, USA.

LEE, K., KANG, K.C., LEE, J., 2002, "Concepts and Guidelines of Feature Modeling for Product Line Software Engineering". In: *Software Reuse: Methods,*

*Techniques, and Tools : 7th International Conference, ICSR-7, Proceedings* pp. 62 - 77, Austin, TX, USA, April.

LG, 2005, "LG Brasil - Produtos". In: http://br.lge.com/md/product/prodcategorylist.do?actType=comp&currPage=1&categoryId=0000060102&parentCategoryId=0000000601&categoryLevel=4&productId=&productImage=&selectModel=, acessado em 16/12/2005.

LOPES, M.A.M., MANGAN, M.A.S., WERNER, C.M.L., 2004, "MAIS: Uma Ferramenta de Percepção para apoiar a Edição Concorrente de Modelos de Análise e Projeto". In: *XVIII Simpósio Brasileiro de Engenharia de Software, Sessão de Ferramentas*, pp. 61-66, Brasília, DF, Brasil, Outubro.

MAIA, N.E.N., BLOIS, A.P.B., WERNER, C.M., 2005, "Odyssey-MDA: Uma Ferramenta para Transformação de Modelos UML". In: *XIX Simpósio Brasileiro de Engenharia de Software, Sessão de Ferramentas*, Uberlândia, MG, Brasil, Outubro.

MASSEN, T.V.D., LICHTER, H., 2002, "Modeling Variability by UML Use Case Diagrams". In: *Proceedings REPL02 - International Workshop on Requirements Engineering for Product Lines*, pp. 19-31, Essen, Germany, September.

MASSEN, T.V.D., LICHTER, H., 2004, "Deficiencies in Features Models". In: *Workshop on Software Variability Management for Product Derivation - Towards Tool Support*, pp. 59-72, Boston, MA, USA.

MATULA, M., 2003, "NetBeans Metadata Repository". In: http://mdr.netbeans.org/MDR-whitepaper.pdf, acessado em 21/12/2005.

MILER, N., 2000, *A Engenharia de Aplicações no Contexto da Reutilização baseada em Modelos de Domínio*, Dissertação de M.Sc, COPPE, UFRJ, Rio de Janeiro, Brasil.

MORISIO, M., TRAVASSOS, G.H., STARK, M.E., 2000, "Extending UML to Support Domain Analysis". In: *Proceedings of the The Fifteenth IEEE International Conference on Automated Software Engineering (ASE'00)*, pp. 321-324, Grenoble, France.

MURTA, L.G.P., BARROS, M.O., WERNER, C.M.L., 2001, "FrameDoc: Um Framework para a Documentação de Componentes Reutilizáveis". In: *IV International Symposium on Knowledge Management/Document Management (ISKM/DM'2001)*, pp. 241-259, Curitiba, PR, Brasil, Agosto.

MURTA, L.G.P., BARROS, M.O., WERNER, C.M.L., 2002, "Charon: Uma Ferramenta para a Modelagem, Simulação, Execução e Acompanhamento de Processos de Software". In: *XVI Simpósio Brasileiro de Engenharia de Software*, pp. 366-371, Gramado, RS, Outubro.

MYLLYMÄKI, T., 2002, *Variability Management in Software Product Lines*, Institute of Software Systems, Tampere University of Technology, Report 30.

NOKIA, 2005, "Nokia Support Glossary". In: http://europe.nokia.com/support/glossaryCDAMaster/0,,lang_id=49&letter=0,00.html, acessado em 19/10/2005.

NORTHROP, L., 2002, "SEI's Software Product Line Tenets", *IEEE Software*, v. 19, n. 4 (July/August, 2002), pp. 32-40.

ODYSSEY, 2005, "Odyssey SDE Homepage". In: http://reuse.cos.ufrj.br/odyssey.

OLIVEIRA, H., MURTA, L.G.P., WERNER, C.M.L., 2004, "Odyssey-VCS: Um Sistema de Controle de Versões Para Modelos Baseados no MOF". In: *XVIII Simpósio Brasileiro de Engenharia de Software, Sessão de Ferramentas*, pp. 85-90, Brasília, DF, Brasil, Outubro

OLIVEIRA, R.F., 2003, *O Papel da Engenharia de Software na Reutilização com Linha de Produtos*, COPPE/UFRJ, Projeto Odyssey Share - Relatório Técnico 3/2003.

OLIVEIRA, R.F., BLOIS, A.P.T.B., VASCONCELOS, A.P.V., *et al.*, 2005a, *Metamodelo de Características da Notação Odyssey-FEX - Descrição de Classes*, COPPE/UFRJ, Projeto Reuse - Relatório Técnico 2/2005. Disponível em: http://reuse.cos.ufrj.br/odyssey/.

OLIVEIRA, R.F., BLOIS, A.P.T.B., VASCONCELOS, A.P.V., *et al.*, 2005b, "Representação de Variabilidades em Componentes de Negócio no Contexto da Engenharia de Domínio". In: *Anais do V Workshop de Desenvolvimento Baseado em Componentes*, pp. 73-80, Juiz de Fora, MG, Brasil, Novembro.

OMG, 2004, "Unified Modeling Language Specification Version2.0". In: http://www.omg.org/cgi-bin/doc?ptc/2004-10-02, acessado em 05/04/2005.

OMG, 2005, "Unified Modeling Language Specification Version2.0". In: http://www.omg.org/cgi-bin/doc?formal/05-07-04, acessado em 04/10/2005.

POULIN, J.S., 1997, "Software Architectures, Product Lines, and DSSAs: Choosing the Appropriate Level of Abstraction ". In: *Workshop on Institutionalizing Software Reuse 8, Proceedings*, Columbus, USA, Mar 23-26.

PRESSMAN, R.S., 1997, "Software Reuse". In: MCGRAW-HILL (eds), *Software Engineering: a Practitioner's Approach* 4th ed.

PRIETO-DIAZ, R., ARANGO, G., 1991, "Domain Analysis Concepts and Research Directions". In: PRIETO-DIAZ, R., ARANGO, G. (eds), *Domain Analysis and Software Systems Modeling*, IEEE Computer Society Press.

RIEBISCH, M., BÖLLERT, K., STREITFERDT, D., *et al.*, 2002, "Extending Feature Diagrams with UML Multiplicities". In: *Proceedings of 6th Conference on Integrated Design & Process Technology*, pp. 1-7, Pasadena,California, USA, June.

ROBBINS, J.E., HILBERT, D.M., REDMILES, D.F., 1998, "Software architecture critics in Argo ". In: *Proceedings of the 3rd International Conference on Intelligent User Interfaces* pp. 141 - 144, San Francisco, California, USA, January 06 - 09.

SCHMID, H.A., 1997, "Systematic Framework Design by Generalization ", *Communications of the ACM* v. 40, n. 10 (October, 1997), pp. 48-51

SEI, 2005a, "Domain Engineering". In: http://www.sei.cmu.edu/domain-engineering/domain_eng.html, acessado em 14/11/2005.

SEI, 2005b, "Domain Engineering and Domain Analysis - Software Technology Roadmap". In: http://www.sei.cmu.edu/str/descriptions/deda_body.html, acessado em 14/11/2005.

SEI, 2005c, "A Framework for Software Product Line Practice - Version 4.2". In: http://www.sei.cmu.edu/productlines/frame_report/what.is.a.PL.htm, acessado em 15/11/2005.

SHULL, F., CARVER, J., TRAVASSOS, G.H., 2001, "An Empirical Methodology for Introducing Software Processes ". In: *Proceedings of the 8th European Software Engineering Conference Held Jointly with 9th ACM SIGSOFT international Symposium on Foundations of Software Engineering* pp. 288-296 Vienna, Austria, September.

SIMOS, M., ANTHONY, J., 1998, "Weaving the Model Web: A Multi-Modeling Approach to Concepts and Features in Domain Engineering". In: *Proceedings of 5th International Conference of Software Reuse (ICSR-5)*, pp. 94-102, Vitória, Canadá, June.

SOUZA, C.R.B., JR., J.S.F., GONCALVES, K.M., *et al.*, 2000, "A Group Critic System for Object-Oriented Analysis and Design". In: *Proceedings of the The*

*Fifteenth IEEE International Conference on Automated Software Engineering (ASE'00)* pp. 313-316, Grenoble, France, September.

SOUZA, C.R.B., OLIVEIRA, H.L.R., ROCHA, C.L.P.D., *et al.*, 2003, "Using Critiquing Systems for Inconsistency Detection in Software Engineering Models". In: *Proceedings of the Fifteenth International Conference on Software Engineering and Knowledge Engineering (SEKE'2003)*, pp. 196-203, San Francisco, California, USA, July.

SUN, 2005, "Java Technology". In: http://www.java.sun.com, acessado em 29/09/2005.

SVAHNBERG, M., GURP, J.V., BOSCH, J., 2001, "On the Notion of Variability in Software Product Lines". In: *Proceedings of The Working IEEE/IFIP Conference on Software Architecture (WISCA'01) pp*. 45-55, Amsterdam, The Netherlands.

SVAHNBERG, M., GURP, J.V., BOSCH, J., 2002, *A Taxonomy of Variability Realization Techniques*, Blekinge Institute of Technology, Technical paper ISSN: 1103-1581.

TEIXEIRA, H.V., 2003, *Geração de Componentes de Negócio a partir de Modelos de Análise*, Dissertação de M.Sc., COPPE, UFRJ, Rio de Janeiro, Brasil.

VERONESE, G.O., NETTO, F.J., CORREA, A., *et al.*, 2002, "ARES - Uma Ferramenta de Engenharia Reversa Java-UML". In: *XVI Simpósio Brasileiro de Engenharia de Software - Caderno de Ferramentas*, pp. 347-352, Gramado, RS, Outubro.

WERNER, C.M.L., BRAGA, R.M.M., 2005, "A Engenharia de Domínio e o Desenvolvimento Baseado em Componentes". In: GIMENES, I.M.S., HUZITA, E.H.M. (eds), *Desenvolvimento Baseado em Componentes: Conceitos e Técnicas*, Rio de Janeiro.

WOHLIN, C., RUNESON, P., HÖST, M., *et al.*, 2000, *Experimentation in Software Engineering: An Introduction*, USA, Kluwer Academic Press.

XAVIER, J.R., 2001, *Criação e Instanciação de Arquiteturas de Software Específicas de Domínio no Contexto de uma Infra-Estrutura de Reutilização*, Dissertação de M.Sc., COPPE, UFRJ, Rio de Janeiro, Brasil.

XML, 2005, "Extended Markup Language". In: http://www.xml.org, acessado em 12/10/2005.

# ANEXO I

# FORMULÁRIO DE CONSENTIMENTO

**MODELAGEM DE CARACTERÍSTICAS DO DOMÍNIO**

Eu declaro ter mais de 18 anos de idade e concordar em participar de um estudo conduzido por Regiane Felipe de Oliveira, como parte das atividades para obtenção do título de Mestre, no Programa de Engenharia de Sistemas e Computação da COPPE/UFRJ. Este estudo visa compreender a viabilidade de aplicação de regras de boa formação do metamodelo da notação Odyssey-FEX, cujo propósito é modelar características do domínio.

**PROCEDIMENTO**

Este estudo acontecerá em duas etapas. Primeiro é apresentado o ferramental necessário para a modelagem de características de domínio e parte de um modelo de características do domínio, bem como a especificação do domínio que deverá ser modelado, documentos que serão utilizados neste estudo. Na segunda etapa os participantes deverão utilizar o ferramental apresentado previamente para a modelagem de características. Eu entendo que, uma vez o experimento tenha terminado, os trabalhos que desenvolvi, serão estudados visando entender a eficiência dos procedimentos e as técnicas que me foram ensinadas.

**CONFIDENCIALIDADE**

Toda informação coletada neste estudo é confidencial, e meu nome não será identificado em momento algum. Da mesma forma, me comprometo a não comunicar os meus resultados enquanto não terminar o estudo, bem como manter sigilo das técnicas e documentos apresentados e que fazem parte do experimento.

**BENEFÍCIOS, LIBERDADE DE DESISTÊNCIA**

Eu entendo que os benefícios que receberei deste estudo são limitados ao aprendizado do material que é distribuído e ensinado, independente de participar ou não deste estudo, mas que os pesquisadores esperam aprender mais sobre quão eficiente é a utilização das regras de boa formação para o modelo de características, que o experimento se propõe a avaliar.

Eu entendo que sou livre para realizar perguntas a qualquer momento ou solicitar que qualquer informação relacionada a minha pessoa não seja incluída no estudo. Eu entendo que participo de livre e espontânea vontade com o único intuito de contribuir para o avanço e desenvolvimento de técnicas e processos para a Engenharia de Software.

**EQUIPE:**

**PESQUISADOR RESPONSÁVEL**

Regiane Felipe de Oliveira

Programa de Engenharia de Sistemas e Computação - COPPE/UFRJ

**PROFESSOR RESPONSÁVEL**

Profa. Cláudia M.L. Werner

Programa de Engenharia de Sistemas e Computação - COPPE/UFRJ

**Nome (em letra de forma):**_____

**Assinatura:** _____ **Data:**_____

# ANEXO II

# LEITURA INTRODUTÓRIA

A Engenharia de Domínio (ED) é uma área da Engenharia de Software cujo principal objetivo é desenvolver artefatos de software para uma família de aplicações, de modo que estes possam ser reutilizados em aplicações deste domínio.

A ED incorpora uma etapa de Análise de Domínio onde o engenheiro de domínio deve obter os requisitos do domínio os quais podem representar características funcionais, conceituais, tecnológicas, que por sua ver podem ser ou não obrigatórias para todas as aplicações derivadas deste domínio, e ainda podem ter diferentes formas de implementação. Neste sentido, é importante que na análise de domínio se tenha uma forma sistemática de representar estas diferentes características, bem como suas obrigatoriedades e variabilidades e, não menos importante, propagar estas propriedades para os diferentes níveis de abstração do domínio.

No estudo de observação que você irá participar é utiliza uma notação (Odyssey-FEX) para representar todas as características acima mencionadas. Nesta notação as características podem ser classificadas quanto à sua categoria, variabilidade e opcionalidade.

Quanto à sua categoria, as características podem ser classificadas conforme apresenta a Tabela 1. Estas características são mutuamente excludentes, ou seja, não podem pertencer simultaneamente a mais de uma categoria.

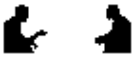
As características na notação Odyssey-FEX possuem a seguinte classificação quanto a sua variabilidade:

- **Ponto de Variação**: são as características que refletem a parametrização no domínio de uma maneira abstrata. São configuráveis por meio das variantes.
- **Variantes**: são características que atuam como alternativas para se configurar um ponto de variação.
- **Invariantes**: são as características fixas, que representam elementos não configuráveis em um domínio.

A classificação das características quanto à opcionalidade indica justamente a obrigatoriedade ou não-obrigatoriedade da presença de um determinado elemento nas aplicações ou produtos a serem desenvolvidos. Vale ressaltar que a opcionalidade é referente ao domínio como um todo. Características que são opcionais em relação ao domínio, mas que por ventura venham a ser mandatórias em relação a outras características a serem selecionadas, deverão expressar essa informação por meio de

Regras de Composição. Características opcionais são evidenciadas no modelo por um contorno pontilhado.

**Tabela 1 – Tipos de Características na notação Odyssey-FEX**

| Ícone | Tipo de Característica | |
|---|---|---|
|  | **Características de Domínio** – Características intimamente ligadas à essência do domínio. Representam as funcionalidades e/ou os conceitos do modelo e correspondem a casos de uso e componentes estruturais concretos. | **Características de Análise** |
|  | **Características de Entidade** – São os atores do modelo. Entidades do mundo real que atuam sobre o domínio. Podem, por exemplo, expor a necessidade de uma interface com o usuário ou de procedimentos de controle. | |
|  | **Características de Ambiente Operacional -** Características que representam atributos de um ambiente que uma aplicação do domínio pode usar e operar. Ex: tipo de terminal, sistemas operacionais, bibliotecas etc. | **Características Tecnológicas** |
|  | **Características de Tecnologia de Domínio -** Características que representam detalhes de implementação de mais baixo nível, específicos para o contexto de um domínio. Ex: métodos de navegação em um domínio de aviões. | |
|  | **Características de Técnicas de Implementação** – Características que representam detalhes de implementação de mais baixo nível, contudo de cunho mais genérico que as relativas à camada de tecnologia de domínio. Ex: técnicas de sincronização. | |

As classificações quanto à categoria, opcionalidade e variabilidade são ortogonais. Como exemplo, uma característica de Domínio pode ser variante e opcional ao mesmo tempo. As características definidas no domínio podem estar relacionadas conforme mostra a Tabela 2.

Um dos requisitos para uma notação que represente variabilidade é a necessidade de representação de dependência e exclusividade entre os elementos do modelo.

As Regras de Composição são regras que definem restrições existentes entre características e que não são expressas no modelo, por meio de relacionamentos, por questões visuais. Tais regras podem ser expressas por meio de cláusulas do tipo “é requerido”, que indicam a dependência entre duas ou mais características (representado no modelo por um **R**, no canto inferior direito da característica), e cláusulas do tipo “mutuamente exclusivo com”, que indicam que duas ou mais características não devem ser selecionadas em conjunto em um mesmo produto ou aplicação (representado no modelo por um **X**, no canto inferior direito da característica).

**Tabela 2: Relacionamentos da notação Odyssey-FEX**

| Representação | Descrição |
|---|---|
|  | **Composição** – Relacionamento em que uma característica é composta de várias outras. Denota relação na qual uma característica é parte fundamental de outra, de forma que a primeira não existe sem a segunda. |
|  | **Agregação** – Relacionamento em que uma característica representa o todo, e as outras as partes. Similar à composição, porém as características envolvidas existem independentemente uma da outra. |
|  | **Herança** – Relacionamento em que há uma generalização/especialização das características. Este tipo de relacionamento indica que as características mais especializadas (filhas) herdam as peculiaridades de características mais generalizadas (antecessores). |
|  | **Associação** – Relacionamento simples entre duas características. Denota algum tipo de ligação entre seus membros. Pode ser nomeada, indicando um tipo específico de ligação. |
|  | **Alternativo** (*Alternative*) - Relacionamento entre um ponto de variação e suas variantes, denota a pertinência de uma variante a um determinado ponto de variação. |
| <<Implemented By>> | **Implementado por** *(Implemented By)* - Relacionamento entre Características de Domínio e Características Tecnológicas, ou entre Características Tecnológicas que se encontrem em camadas diferentes. Indicam que uma determinada tecnologia é utilizada para implementar a característica relacionada. |
| <<Communication Link>> | **Ligação de Comunicação** *(Communication Link)* - *R*elacionamento existente entre Características de Entidade e Características de Domínio. Cumpre o mesmo papel do relacionamento de associação entre atores e casos de uso na UML |

### Exemplo de Uso da Notação – Domínio de Ensino Colaborativo

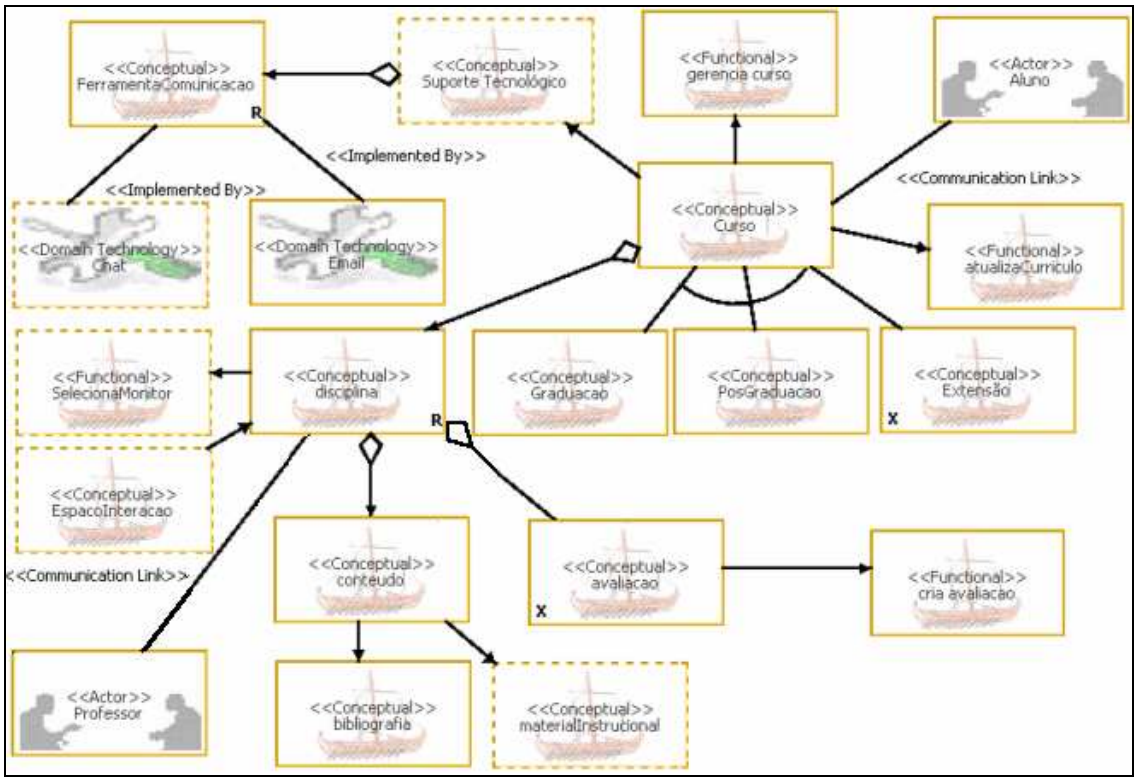
Para auxiliar os conceitos acima apresentados será utilizado um pequeno modelo de características do domínio de ensino colaborativo.

As características conceituais, funcionais, de entidade e tecnológicas citadas na Tabela 1 estão identificadas pelos estereótipos <<Conceptual>>, <<Functional>>, <<Actor>> e <<Domain Technology>>, respectivamente.

As características com retângulos tracejados representam características opcionais e as demais, obrigatórias para qualquer aplicação derivada deste domínio. A característica Curso representa um ponto de variação mandatório, e as características Graduação, Pós-Graduação e Extensão representam suas variantes também mandatórias. Para representar esta relação é utilizado o relacionamento alternativo.

A característica conceitual Curso está relacionada a característica de entidade Aluno por meio de um <<Communication Link>>. A característica conceitual FerramentaComunicação está relacionada às características de Tecnologia de Domínio Chat e Email por meio de um <<Implemented by>>. Disciplina está relacionado com as características Conteúdo e Avaliação, por meio de agregação.

A relação de exclusividade é expressa no modelo por meio da notação **X** (e.g. Extensão **exclui** Avaliação) e a relação de dependência por meio da notação **R** (Disciplina **requer** FerramentaComunicação).



# ANEXO III

# ESPECIFICAÇÃO DO REFINAMENTO DO DOMÍNIO

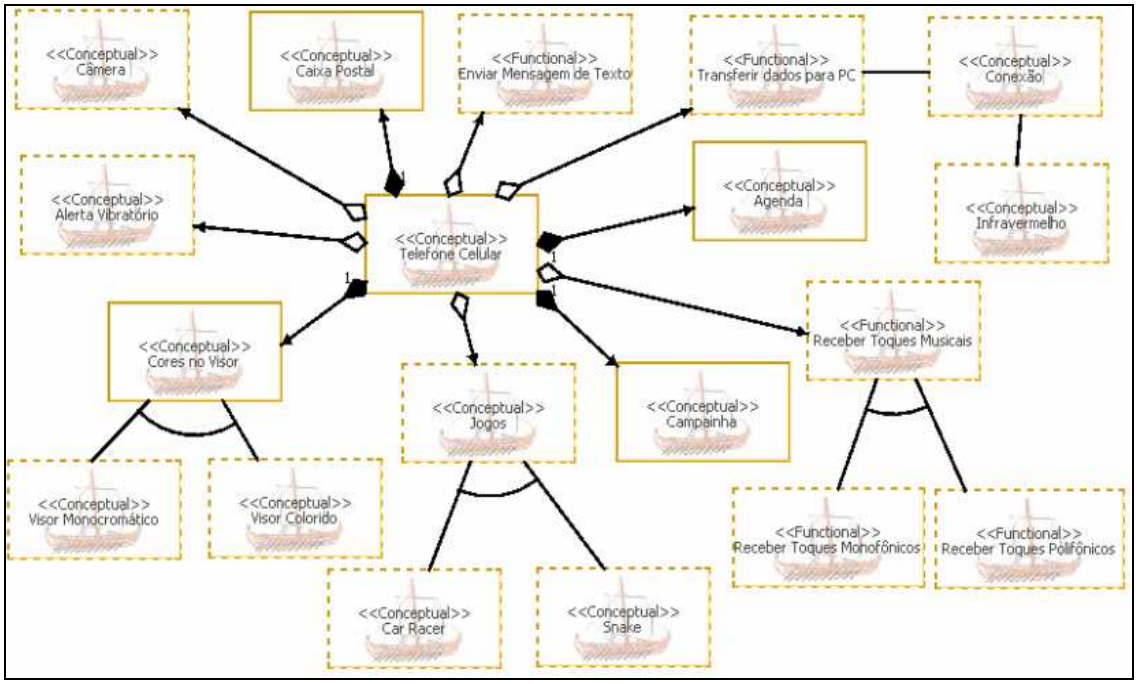
O domínio de Telefonia Móvel abrange conceitos e funcionalidades que podem estar presentes em um software desenvolvido para um telefone celular. Atualmente, pode ser encontrada uma grande diversidade de telefones celulares, que apresentam as mais variadas funcionalidades. A descrição a seguir representa parte deste domínio está modelada em características, como apresenta a figura em anexo.

Alguns conceitos e funcionalidades são intrínsecos e estão presentes a todos os tipos de software. Dentre estes podem ser citados: Campainha, que representa o toque do telefone, Agenda, onde são armazenados números de telefone, Chamada Telefônica, que é a funcionalidade de se efetuar uma ligação, e Caixa Postal, onde mensagens de voz são deixadas para o usuário. Além disso, alguns conceitos apresentam alternativas, dando ao usuário a opção de escolha na hora da compra. Como exemplo, tem-se o conceito de Cores do Visor. Este conceito representa o tipo de visor do aparelho celular, que tem como alternativas as opções “monocromático” ou “colorido”.

Outros conceitos e funcionalidades são oferecidos no domínio de telefonia celular, mas podem ser encontrados em apenas alguns aparelhos, constituindo um diferencial. Dentre estes podem ser citados: Câmera fotográfica, Alerta Vibratório, Acesso à Internet, Envio de mensagens de texto, como e-mails e recados, Jogos e Recebimento de toques musicais, que representa a funcionalidade de alteração do tipo de campainha, reproduzindo-se um toque especial, como uma música conhecida, geralmente oferecido pela operadora de telefonia móvel. Os toques musicais recebidos podem ser de vários tipos, como monofônicos ou polifônicos.

No caso dos jogos, existe uma variedade deles que pode estar presente em um ou outro aparelho de telefone celular. Como exemplo, tem-se *Car Racer*, que é um jogo de corrida de carros, jogo de Tênis, entre outros.

Alguns aparelhos celulares oferecem a funcionalidade de transferência de dados para um computador pessoal (PC), sendo necessário para isso algum tipo de conexão. Esta conexão pode ser efetuada de várias maneiras, como por exemplo, por meio de Infravermelho, que faz a comunicação sem fio, por meio de sinais de luz que são captados por um sensor instalado no destinatário. A Figura 1 ilustra a modelagem do domínio de telefonia móvel.




O domínio de um problema está em constante evolução, sendo necessário que refinamentos sejam efetuados nos modelos. O domínio de telefonia celular não é diferente. Assim, considere que um “novo” aparelho celular esteja na iminência de ser lançado no mercado. Tal aparelho apresenta, além das características presentes em todos os aparelhos, a seguinte especificação: visor colorido, transferência de dados para PC por meio de Bluetooth, e por meio de USB. No entanto, para conexão via USB um cabo de dados é necessário. O novo modelo permite ainda uma maior iteração com o usuário, devido ao jogo Quizz, onde o usuário participa ativamente. Além disso, o novo aparelho tem a opção de recebimento de toque no formato MP3, no entanto, a escolha dessa opção impede que ele tenha outros tipos de toque.

Assim, o domínio deve ser refinado para incorporar os novos requisitos, de modo que estes possam eventualmente ser reutilizados posteriormente. Para tanto, as regras de boa formação da notação Odyssey-FEX deverão ser observadas.

**Glossário**:

**Bluetooth***:* tecnologia que permite que uma conexão sem fio de curto alcance seja estabelecida com outro dispositivo compatível (celulares, laptops, câmeras digitais)

**Infravermelho**: tecnologia para transferência de dados, cuja comunicação é sem fio, por meio de sinais de luz que são captados por um sensor instalado no destinatário.

**USB** - *Universal Serial Bus*: interface de comunicação entre um computador e um dispositivo compatível, como um aparelho celular, câmera digital ou impressora

# ANEXO IV

# NOTAÇÃO ODYSSEY-FEX : REGRAS DE BOA FORMAÇÃO

Foi definido um conjunto regras de boa formação para a modelagem das características utilizando a notação Odyssey-FEX. Tais regras visam manter a consistência das variabilidades, obrigatoriedades e relacionamentos entre características do domínio, que provavelmente venham a ser utilizadas tanto na definição de novos artefatos do domínio, como no recorte para a instanciação de aplicações.

As regras que você deverá observar são:

**R1:** Características que tenham a classificação de Variantes devem obrigatoriamente estar associadas a UMA e SOMENTE UMA característica que tenha classificação de Ponto de Variação

**R2**: Não deve existir características que tenham a classificação de Variantes desassociadas de um Ponto de Variação

**R3**: Características que tenham classificação de Variante e Mandatória devem ter um relacionamento do tipo Alternativo com outra característica classificada como Ponto de Variação NECESSARIAMENTE mandatória

**R4**: Características que tenham classificação de Ponto de Variação e Opcionais devem ter um relacionamento do tipo Alternativo com outras características classificadas como Variantes NECESSARIAMENTE Opcionais

**R5**: Não deve haver relacionamento de composição entre features quando a feature que representa o todo é opcional e a feature que representa a parte é mandatória.

**R6**: O relacionamento Alternativo só poderá ocorrer entre características do tipo Variante e Ponto de Variação

**R7**: O relacionamento Alternativo só poderá ter características do tipo Ponto de Variação como origem.

**R8**: O relacionamento Ligação de Comunicação só poderá ocorrer entre Características de Domínio e Características de Entidade

**R9**: Características dependentes entre si não podem ser mutuamente exclusivas, e vice versa.

**R10**: Numa regra de composição inclusiva, o conseqüente só poderá ser opcional se o antecedente for opcional

**R11**: Uma regra de composição exclusiva só pode envolver características opcionais.

**R12**: Uma regra de composição não pode ter antecedente definido e conseqüente nulo ou vice versa.

**R13**: Não deve haver herança circular entre características.

**R14**: Não deve haver herança entre características de tipos diferentes

# ANEXO V

## AVALIAÇÃO GERAL DAS REGRAS DE BOA FORMAÇÃO

| | |
|---|---|
| 1) As regras de boa formação da notação Odyssey-FEX são de fácil entendimento? | ☐ Sim ☐ Não ☐ Parcialmente |
| Justifique sua resposta caso a opção escolhida seja "não" ou "parcialmente": _____ _____ | |
| 2) Em algum momento você sentiu a necessidade de associar uma variante a mais de um ponto de variação? | ☐ Sim ☐ Não ☐ Parcialmente |
| Justifique sua resposta caso a opção escolhida seja "sim" ou "parcialmente": _____ _____ | |
| 3) Em algum momento você desejou relacionar uma variante MANDATÓRIA a um ponto de variação OPCIONAL? | ☐ Sim ☐ Não ☐ Parcialmente |
| Justifique sua resposta caso a opção escolhida seja "sim" ou "parcialmente": _____ _____ | |
| 4) Em algum momento você desejou relacionar um ponto de variação MANDATÓRIO com uma variante OPCIONAL? | ☐ Sim ☐ Não ☐ Parcialmente |
| Justifique sua resposta caso a opção escolhida seja "sim" ou "parcialmente": _____ _____ | |
| 5) Em algum momento você desejou criar uma composição entre um elemento mandatório e um elemento opcional? | ☐ Sim ☐ Não ☐ Parcialmente |
| Justifique sua resposta caso a opção escolhida seja "sim" ou "parcialmente": _____ _____ | |
| 6) Em algum momento você desejou utilizar o relacionamento Ligação de Comunicação entre características diferentes de Entidade e Domínio? | ☐ Sim ☐ Não ☐ Parcialmente |
| Justifique sua resposta caso a opção escolhida seja "sim" ou "parcialmente": _____ _____ | |

| | |
|---|---|
| 7) Em algum momento você desejou criar uma regra de composição exclusiva envolvendo características dependentes entre si ou vice versa?<br>Justifique sua resposta caso a opção escolhida seja "sim" ou "parcialmente":<br>_____<br>_____ | ☐ Sim ☐ Não<br>☐ Parcialmente |
| 8) Em algum momento você desejou criar uma regra de composição exclusiva envolvendo características opcionais?<br>Justifique sua resposta caso a opção escolhida seja "sim" ou "parcialmente":<br>_____<br>_____ | ☐ Sim ☐ Não<br>☐ Parcialmente |
| 9) Em algum momento você desejou criar uma regra de composição inclusiva cujo antecedente era mandatório e o conseqüente opcional?<br>Justifique sua resposta caso a opção escolhida seja "sim" ou "parcialmente":<br>_____<br>_____ | ☐ Sim ☐ Não<br>☐ Parcialmente |

## Suporte Computacional para as Regras de Boa Formação

| | |
|---|---|
| 10) O suporte computacional existente contemplou adequadamente a utilização das regras de boa formação do modelo de características?<br>Justifique sua resposta caso a opção escolhida seja "não" ou "parcialmente":<br>_____<br>_____ | ☐ Sim ☐ Não<br>☐ Parcialmente |
| 11) Durante a construção do modelo de características foi fácil identificar qual regra de boa formação estava sendo violada?<br>Justifique sua resposta caso a opção escolhida seja "não" ou "parcialmente":<br>_____<br>_____ | ☐ Sim ☐ Não<br>☐ Parcialmente |