**COPPE/UFRJ**

# UBIFEX: UMA ABORDAGEM PARA MODELAGEM DE CARACTERÍSTICAS DE LINHA DE PRODUTOS DE SOFTWARE SENSÍVEIS AO CONTEXTO

Paula Cibele Cavalcante Fernandes

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientadora: Cláudia Maria Lima Werner

Rio de Janeiro
Fevereiro de 2009

UBIFEX: UMA ABORDAGEM PARA MODELAGEM DE CARACTERÍSTICAS DE LINHA DE PRODUTOS DE SOFTWARE SENSÍVEIS AO CONTEXTO

Paula Cibele Cavalcante Fernandes

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) 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:

_____

Prof.ª Cláudia Maria Lima Werner, D.Sc.

_____

Prof. Jano Moreira de Souza, Ph.D.

_____

Prof. Leonardo Gresta Paulino Murta, D.Sc.

_____

Prof.ª Vaninha Vieira dos Santos, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
FEVEREIRO DE 2009

Fernandes, Paula Cibele Cavalcante

UbiFEX: Uma Abordagem para Modelagem de Características de Linha de Produtos de Software Sensíveis ao Contexto / Paula Cibele Cavalcante Fernandes. – Rio de Janeiro: UFRJ/COPPE, 2009.

XIII, 126 p.: il.; 29,7 cm.

Orientadora: Cláudia Maria Lima Werner

Dissertação (mestrado) – UFRJ/ COPPE/ Programa de Engenharia de Sistemas e Computação, 2009.

Referencias Bibliográficas: p. 103-111.

1. Modelagem de Características 2. Linha de Produtos de Software 3. Computação Sensível ao Contexto. I. Werner, Cláudia Maria Lima. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

À minha mãe Lucineide e

à minha avó Creuza (*in memorian*).

# Agradecimentos

A Deus, por me guiar durante toda esta caminhada.

Aos meus pais, por todo carinho e dedicação na minha formação. Por terem apoiado minha vinda para o Rio, apesar da saudade.

À minha família, por todos os pensamentos positivos e força para continuar.

Ao João Gustavo, por ser um grande companheiro em todos os momentos e pela ajuda durante o desenvolvimento deste trabalho.

À minha orientadora, professora Cláudia Werner, pela confiança, pela dedicação na orientação deste trabalho e por todas as oportunidades que contribuíram de forma essencial para o meu amadurecimento na pesquisa.

Ao professor Leonardo Murta, por ter contribuído na proposta desta dissertação e por participar desta banca.

Aos professores Jano Moreira e Vaninha Vieira, por terem aceitado participar desta banca.

Aos amigos do grupo de Reutilização de Software da COPPE/UFRJ, por todas as contribuições para o desenvolvimento deste trabalho. Em especial, Danny, Anderson, Chessman, Hua, Marcelo, Rafael e Rodrigo.

Aos amigos do grupo de pesquisa GREat, em especial o Lincoln e a professora Rossana, por todas as oportunidades de colaboração.

Aos amigos de Fortaleza, em especial Elias, Rafael, Guilherme e Juliana, que, apesar da distância, sempre estiveram presentes.

À CAPES , ao CNPq e ao CEPEL, pelo apoio financeiro.

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

# UBIFEX: UMA ABORDAGEM PARA MODELAGEM DE CARACTERÍSTICAS DE LINHA DE PRODUTOS DE SOFTWARE SENSÍVEIS AO CONTEXTO

Paula Cibele Cavalcante Fernandes

Fevereiro/2009

Orientadora: Cláudia Maria Lima Werner

Programa: Engenharia de Sistemas e Computação

Nos últimos anos, com o desenvolvimento da computação ubíqua, os sistemas buscam cada vez mais ampliar as atividades humanas com novos serviços que possam se adaptar às circunstâncias em que serão utilizados. Nesse cenário, os sistemas sensíveis ao contexto utilizam informações de contexto para prover serviços adaptados e relevantes na realização de tarefas dos seus usuários. O desenvolvimento dessas aplicações pode se beneficiar do paradigma de linha de produtos em termos de reusabilidade e configurabilidade. Um dos desafios na construção de linha de produtos para essa classe de sistemas envolve a representação dessas informações de contexto em modelos de características e a garantia da consistência da configuração dos produtos.

Este trabalho de pesquisa propõe uma abordagem para modelagem de características de linha de produtos sensíveis ao contexto, denominada UbiFEX, cujo objetivo é permitir a representação, de forma explícita, das entidades e informações de contexto relevantes para o domínio e, principalmente, a representação do impacto dessas informações na variabilidade dos produtos em tempo de execução. Além disso, foi proposto um mecanismo para verificação da consistência das configurações dos produtos em diferentes cenários de execução com base nas restrições definidas em modelos de características. Foi realizado um estudo preliminar para avaliar este mecanismo de verificação. Além disso, um protótipo foi desenvolvido para viabilizar a aplicação da abordagem proposta.

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

UBIFEX: AN APPROACH FOR FEATURE MODELING OF CONTEXT-AWARE SOFTWARE PRODUCT LINE

Paula Cibele Cavalcante Fernandes

February/2009

Advisor: Cláudia Maria Lima Werner

Department: Computer and Systems Engineering

In the recent years, with the development of the ubiquitous computing area, systems have to amplify more and more human activities with new services that can adapt to the circumstances in which they are used. In this scenario, context-aware systems use context information to provide adapted and relevant services to the execution of users' tasks. The development of these applications should take advantage from the software product line concept in terms of reusability and configurability. One of the challenges to develop a product line for this class of systems involves the representation of context information in feature models and the guarantee of product configuration consistency.

This work presents an approach for modeling context-aware product lines, called UbiFEX, which aims to represent relevant context entities and information for a domain in an explicit form, in addiction to represent the impact of this information in the product variability at runtime. Also, a mechanism was proposed to verify the product configurations consistency in different execution scenarios based on the specified restrictions in the feature model. A preliminary study was performed to evaluate this verification mechanism. Moreover, a prototype was built to enable the application of the proposed approach.

# Índice

# Índice de Figuras

# Índice de Tabelas

# Capítulo 1 – Introdução

## 1.1 Preâmbulo

Com o passar dos anos, os computadores tornaram-se cada vez mais presentes nas tarefas cotidianas e mais próximos dos usuários. Seja em casa ou no trabalho, para acessar os e-mails, editar textos ou ler as últimas notícias, é difícil imaginar o mundo sem a presença deles.

No início, por volta da década de 40, surgiram os *mainframes*, computadores que geralmente ocupavam um grande espaço e possuíam acesso restrito. Nas décadas de 70 e 80 surgiram os primeiros computadores pessoais (PCs), possibilitando uma maior popularização dos computadores e aproximação com os usuários, além da realização de tarefas gerais como processamento de texto, navegação na Internet, jogos e programação. Com o surgimento da Internet e dos ambientes distribuídos, os computadores passaram a se interconectar e a compartilhar recursos e informações, ampliando os serviços oferecidos e a complexidade das aplicações desenvolvidas. A Figura 1.1 ilustra alguns desses marcos na evolução da Computação.

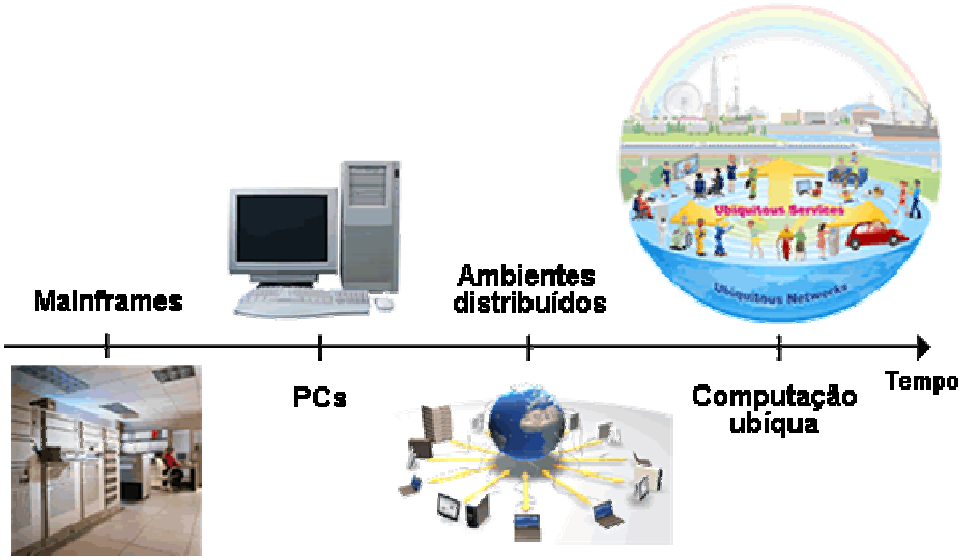



**Figura 1.1. Marcos na evolução da Computação.**

Nos últimos anos, estamos vivendo o surgimento da Computação Ubíqua (WEISER, 1991), que compreende um universo formado por uma grande variedade de dispositivos computacionais que estão se tornando cada vez mais presentes no cotidiano das pessoas, como telefones celulares, PDAs (*Personal Digital Assistants*), entre outros.

Um dos propósitos desse novo paradigma da computação é ampliar as atividades humanas com novos serviços que possam se adaptar às circunstâncias em que serão utilizados (COUTAZ *et al.*, 2005).

Nesse cenário, uma classe de aplicações, chamadas de sensíveis ao contexto (BALDAUF *et al.*, 2007), utiliza informações de contexto[1] para fornecer serviços adaptados e relevantes na realização de tarefas dos seus usuários. O desenvolvimento de software para esse tipo de cenário apresenta muitos desafios (GARLAN e SCHMERL, 2001), entre eles o fato de terem que ser suportados por diferentes dispositivos em diferentes contextos.

Na literatura, podem ser encontradas algumas soluções (HENRICKSEN e INDULSKA, 2004; ROMÁN *et al.*, 2002; SACRAMENTO *et al.*, 2004) que apóiam o desenvolvimento de aplicações sensíveis ao contexto. Entretanto, a grande maioria, embora utilize mecanismos que possibilitem alguma forma de reúso, este não é investigado como foco principal. Nestas soluções, o interesse principal está em resolver problemas de domínios específicos de maneira isolada, não havendo uma preocupação mais sistemática com as etapas de desenvolvimento de software. No entanto, essas aplicações podem compartilhar diversos pontos em comum, principalmente em relação à aquisição e gerenciamento das informações de contexto.

## 1.2 Motivação

De uma maneira geral, a reutilização de software pode ser vista como o uso de software existente ou conhecimento referente a esse software para construir um novo software. Além disso, é considerada uma forma de obter melhores indicadores de qualidade e produtividade na construção de sistemas maiores, mais complexos, confiáveis e baratos (FRAKES e KANG, 2005).

No entanto, para que a reutilização alcance os níveis desejados de produtividade e qualidade na construção de software é necessário que ela seja feita de maneira sistemática, considerando todas as fases do desenvolvimento, desde a análise até a implementação (WERNER e BRAGA, 2005). Tal necessidade dá lugar às técnicas de

[1] O termo contexto pode ser definido como qualquer informação usada para caracterizar a situação de uma entidade. Uma entidade pode ser uma pessoa, um lugar ou um objeto considerado relevante para a interação entre um usuário e uma aplicação, incluindo o próprio usuário e aplicação (DEY, 2001).

reutilização, tais como Linha de Produtos de Software, *Frameworks* e Desenvolvimento Baseado em Componentes.

O paradigma de Linha de Produtos de Software (LPS) tem como proposta a construção sistemática de software baseada em uma família de produtos, guiando as organizações tanto na construção de novas aplicações, a partir de artefatos reutilizáveis (desenvolvimento com reutilização), quanto na construção desses artefatos (desenvolvimento para reutilização) (VAN DER LINDEN *et al.*, 2007). Além disso, permite que as organizações explorem as características comuns e variáveis dos seus produtos de software como forma de alcançar economia em sua produção (CLEMENTS e NORTHROP, 2002).

A motivação deste trabalho consiste em possibilitar que o desenvolvimento de aplicações sensíveis ao contexto possa se beneficiar do paradigma de LPS, de forma que o reúso ocorra de maneira sistemática e que novos produtos possam ser facilmente configurados. O foco deste trabalho está na modelagem de características, uma das primeiras atividades no processo de desenvolvimento de uma LPS, conforme apresentado a seguir.

## 1.3 Problema

A modelagem de características possui como objetivo identificar as características externamente visíveis dos produtos que farão parte da LPS (LEE e MUTHIG, 2006), capturando os pontos comuns e variáveis existentes entre eles. Essas características são organizadas em um modelo denominado modelo de características, que é utilizado como ponto de partida para o recorte necessário à instanciação de novos produtos. Além disso, esse modelo permite uniformizar o entendimento entre todos os envolvidos no processo de desenvolvimento, tais como usuários, especialistas de domínio e desenvolvedores (MASSEN e LICHTER, 2002).

Como mencionado na seção 1.1, as informações de contexto são um conceito chave na definição das aplicações sensíveis ao contexto. Assim, para a construção de uma Linha de Produtos de Software Sensíveis ao Contexto (LPSSC), ou seja, uma linha de produtos que apóia o desenvolvimento de produtos sensíveis ao contexto, é importante que as informações de contexto relevantes para o domínio sejam representadas no modelo de características, além do impacto dessas informações na reconfiguração dos produtos em tempo de execução.

Porém, grande parte das abordagens existentes para LPS trata as variabilidades de cada produto de forma estática, focando apenas em uma adaptação em tempo de desenvolvimento, onde diferentes configurações de um produto são geradas para atender às necessidades dos clientes e às características do ambiente de execução. Essas abordagens não são adequadas para lidar com o dinamismo exigido pelas aplicações sensíveis ao contexto, sendo necessário estendê-las para que sejam capazes de acomodar características de adaptabilidade e sensibilidade ao contexto.

Outras abordagens apresentam essa preocupação de tratar variabilidade de forma dinâmica, porém elas não possuem uma forma de representação explícita das informações de contexto e da forma como elas influenciam na adaptação dinâmica dos produtos com base nessas informações. Além disso, essas abordagens não apresentam mecanismos para verificar se essas adaptações realizadas em tempo de execução estão consistentes com as restrições definidas no modelo em tempo de desenvolvimento.

## 1.4 Objetivo

Diante do problema apresentado na seção 1.3, o objetivo deste trabalho é apresentar uma abordagem para a modelagem de características no desenvolvimento de LPSSC. Dessa forma, será possível representar as informações de contexto relevantes para o domínio desde as fases iniciais de desenvolvimento, facilitando o entendimento dos envolvidos no processo.

Para atingir esses objetivos, foram detectadas algumas etapas que serviram como guia para esse trabalho de pesquisa: (1) analisar as técnicas existentes para modelagem de contexto no desenvolvimento tradicional de aplicações sensíveis ao contexto; (2) identificar os elementos que devem estar presentes no modelo de características para representar esse tipo de informação; (3) definir mecanismos de extensão no modelo de características para acomodar os elementos identificados; e (4) definir um mecanismo para a verificação da configuração dos produtos em relação às variações de contexto com base em modelos de características.

A solução proposta por esse trabalho inclui a extensão da notação para modelagem de características Odyssey-FEX (OLIVEIRA, 2006), de forma a permitir a representação de informações de contexto e de decisões de adaptação com base nessas informações. Para isso, foram definidos novos tipos de características, regras e expressões que representam essa relação entre características e contextos. Além disso, a abordagem proposta, denominada UbiFEX, inclui a definição de um mecanismo para a

verificação da configuração dos produtos em diferentes cenários de execução, considerando as decisões de adaptação e as restrições de composição definidas no modelo de características, como uma tentativa de antecipar falhas que só seriam identificadas durante a execução do produto.

## 1.5 Organização

Esta dissertação está organizada em outros seis capítulos, além deste capítulo de introdução.

O Capítulo 2 apresenta uma revisão da literatura sobre computação sensível ao contexto, discutindo os principais conceitos envolvidos e técnicas utilizadas para o desenvolvimento de aplicações.

O Capítulo 3 apresenta os conceitos relacionados à Linha de Produtos de Software. São também discutidos alguns trabalhos relacionados que levam em consideração o desenvolvimento de aplicações dinâmicas e adaptativas utilizando LPS.

O Capítulo 4 apresenta a abordagem proposta para modelagem de características no desenvolvimento de LPSSC, denominada UbiFEX, que inclui uma notação para representação explícita dos conceitos relacionados à sensibilidade ao contexto e um mecanismo para a verificação da configuração dos produtos em relação às variações de contexto.

O Capítulo 5 descreve um estudo preliminar realizado com alunos de pós-graduação para avaliar o mecanismo de verificação proposto pela abordagem UbiFEX.

O Capítulo 6 discute os detalhes da implementação da abordagem proposta, por meio de um protótipo acadêmico desenvolvido no contexto do ambiente Odyssey (ODYSSEY, 2009). Essa implementação inclui a extensão do ambiente de modelagem de características existente e a criação de um *plug-in* para verificação da configuração dos produtos.

O Capítulo 7 apresenta algumas conclusões, incluindo as contribuições e as limitações deste trabalho, além das possibilidades de trabalhos futuros.

Por fim, os Apêndices incluem os documentos utilizados para aplicação do estudo descrito no Capítulo 5.

# Capítulo 2 – Computação Sensível ao Contexto

## 2.1 Introdução

*"Bom dia! O café está esquentando e seu jornal eletrônico já está disponível - uma voz agradável vinda do refrigerador o cumprimenta enquanto você entra na cozinha. Quando você senta no carro, o assento, os espelhos e o cinto de segurança são automaticamente ajustados - seu filho usou o carro na noite anterior. Na hora do almoço, enquanto você caminha no shopping, mensagens de restaurantes, localizados a menos de 200 metros e que servem sua comida favorita, são enviadas para o seu celular"* (LOKE, 2006).

A situação narrada no parágrafo anterior e muitas outras podem ser imaginadas quando pensamos em sistemas que são cientes do contexto corrente dos seus usuários e podem realizar ações com base nessas informações.

Como apresentada no Capítulo 1, a computação sensível ao contexto envolve o desenvolvimento de aplicações que utilizam informações de contexto para prover serviços e informações relevantes aos seus usuários.

Este capítulo está organizado da seguinte forma: na Seção 2.2, são apresentadas algumas definições para o termo contexto; na Seção 2.3, são apresentados conceitos relacionados à definição e estrutura das aplicações sensíveis ao contexto; na Seção 2.4 são descritas diferentes técnicas para a modelagem de contexto; finalmente, a Seção 2.5 conclui o capítulo com algumas considerações finais.

## 2.2 Contexto

Quando as pessoas interagem umas com as outras, elas são capazes de utilizar implicitamente informações contextuais para auxiliar na comunicação, tais como o humor da pessoa, o idioma utilizado, entre outros. Porém, essa tarefa não é tão simples quando tratamos da interação entre pessoas e computadores. Assim, aumentar o acesso das aplicações ao contexto no qual elas estão sendo executadas possibilita uma melhor interação com os usuários e a capacidade de prover serviços mais úteis.

Para entender o conceito de aplicações sensíveis ao contexto, é necessário definir inicialmente o que se entende por contexto. Uma das primeiras definições no domínio da computação sensível ao contexto foi dada por Schilit e Theimer (1994). Os

autores definem contexto, por enumeração, como: localização; identificação de pessoas e objetos próximos; e mudanças nesses objetos. Além disso, classificam contexto em três categorias: computacional (e.g., capacidade de processamento, memória disponível), do usuário (e.g., localização, pessoas próximas) e físico (e.g., nível de luminosidade, barulho). Outra definição baseada em exemplos é apresentada por Brown *et al.* (1997), que se referem a contexto como localização, identificação das pessoas ao redor do usuário, hora do dia, estação do ano, temperatura, entre outros. Outras definições são dadas utilizando sinônimos, como, por exemplo, contexto compreende aspectos da situação corrente do usuário (HULL *et al.*, 1997), ou ainda, elementos do ambiente do usuário que são reconhecidos pelo computador (BROWN, 1996).

Uma definição mais abrangente, e adotada neste trabalho de pesquisa, é dada por Dey (2001):

"Contexto é qualquer informação que pode ser usada para caracterizar a situação de uma entidade. Uma entidade pode ser uma pessoa, um lugar ou um objeto considerado relevante para a interação entre um usuário e uma aplicação, incluindo o próprio usuário e aplicação".

Essa definição deixa a cargo do projetista da aplicação decidir quais informações podem ser consideradas como contexto para a sua aplicação.

Existem autores que fazem ainda a distinção entre contexto e elemento contextual. Segundo Vieira (2008), contexto representa um conjunto de elementos contextuais instanciados que são necessários para apoiar uma tarefa. Os elementos contextuais, por sua vez, representam qualquer dado ou informação capaz de caracterizar uma entidade em um domínio. Além disso, contexto possui uma dimensão dinâmica e deve ser construído em tempo de execução, enquanto que elementos contextuais são estáveis e podem ser definidos em tempo de projeto.

A tarefa de definir qual informação é importante ou relevante é facilitada utilizando-se as diretrizes: quem (*who*), onde (*where*), o quê (*what*) e quando (*when*), relacionadas a alguma entidade, para se determinar o porquê (*why*) de uma determinada situação estar ocorrendo (ABOWD e MYNATT, 2000).

Alguns autores ainda classificam as informações de contexto em diferentes dimensões, como externo e interno (PREKOP e BURNETT, 2003) ou físico e lógico (HOFER *et al.*, 2003). A dimensão externa ou física se refere ao contexto que pode ser medido por meio de sensores. Já a dimensão interna ou lógica é, em grande parte, especificada pelo usuário ou capturada por monitoração das suas interações. Em relação

à forma como as informações de contexto são adquiridas, outras abordagens as classificam como: obtidas por perfil (*profiled*), sentidas (*sensed*) e derivadas (*derived*) (VIEIRA, 2008). Em relação às questões de persistência, as informações de contexto podem ser classificadas como: estáticas, não variam ao longo do tempo para a entidade a qual eles descrevem, e dinâmicas. Algumas abordagens consideram ainda aspectos de qualidade relacionados às informações de contexto, como, por exemplo, confiabilidade, relevância e validade (ZIMMER, 2006).

Outro ponto importante, além da definição do contexto relevante para um determinado domínio, é o entendimento de como esse contexto pode ser utilizado, o que ajuda a determinar quais comportamentos sensíveis ao contexto serão suportados.

## 2.3 Aplicações Sensíveis ao Contexto

Nesta seção, são discutidos conceitos relacionados às aplicações sensíveis ao contexto, incluindo a definição e a estrutura dessas aplicações. Por fim, são apresentados alguns conceitos sobre adaptação.

### 2.3.1 Definição

Após a apresentação dos conceitos relacionados à definição de contexto, podemos definir aplicações sensíveis ao contexto. Segundo Dey (2001), as definições existentes na literatura podem ser classificadas em dois grupos: as que se adaptam ao contexto e as que simplesmente o utilizam. No primeiro grupo, o comportamento da aplicação deve ser modificado para que a mesma seja considerada sensível ao contexto. No segundo, o sistema deve ser capaz de detectar, interpretar e responder ao contexto, não necessariamente se adaptando. Porém, os autores sugerem uma definição mais genérica que inclui os dois grupos:

"Um sistema é sensível ao contexto se usa contexto para fornecer informações relevantes e/ou serviços para o usuário, onde a relevância depende da tarefa do usuário."

Os guias turísticos móveis (GRÜN *et al.*, 2008) são exemplos de aplicações sensíveis ao contexto. Esses sistemas têm como objetivo auxiliar o turista durante sua viagem, fornecendo informações e serviços com base, por exemplo, na localização e preferências do usuário. Os serviços oferecidos podem ser divididos em categorias como, por exemplo: acomodação, emergência, entretenimento, gastronomia, navegação e orientação, páginas amarelas, notícias, compras, esportes, atrações turísticas, transporte e condições climáticas. Essas aplicações podem, por exemplo, apresentar um

mapa da localização atual do usuário ou fornecer informações sobre o local que está sendo visitado (Figura 2.1). Outros exemplos de sistemas sensíveis ao contexto podem ser encontrados no trabalho de Chen e Kotz (2000).


**Figura 2.1. Exemplo de aplicações sensíveis ao contexto.**

### 2.3.2 Estrutura

Os sistemas sensíveis ao contexto podem ser implementados de diferentes formas, levando-se em consideração as funcionalidades básicas envolvidas no seu funcionamento: "sentir", "pensar" e "agir". A Figura 2.2 ilustra uma arquitetura conceitual em camadas desse tipo de sistema.

As camadas de sensores e recuperação de dados correspondem à capacidade de "sentir" e são responsáveis pela aquisição dos dados contextuais. Um programa normalmente necessita de algumas entradas para executar, as quais, tradicionalmente, são fornecidas manualmente pelos usuários. No entanto, essas informações também podem ser providas por sensores, que podem ser vistos como uma ponte entre o mundo físico e o mundo virtual.

| Aplicação |
|---|
| Gerenciamento |
| Pré-processamento |
| Recuperação de dados |
| Sensores |

**Figura 2.2. Arquitetura conceitual de sistemas sensíveis ao contexto [Adaptada de (BALDAUF *et al.*, 2007)].**

Uma grande variedade de sensores vem sendo desenvolvida, como, sensores de luz, temperatura, movimento, contato, entre outros. Uma tecnologia que está recebendo grande atenção nessa área são as etiquetas RFID (*Radio Frequency Identification*) ou

etiquetas inteligentes (LAHIRI, 2005), que podem ser utilizadas para detectar a presença ou ausência de objetos em um determinado ambiente.

De uma maneira geral, um sensor pode ser definido como qualquer dispositivo de hardware ou software, ou uma combinação dos dois, que pode ser utilizado para adquirir informações de contexto (LOKE, 2006). Por exemplo, um termômetro pode prover leituras de temperaturas, mas uma aplicação que acessa um serviço Web para retornar a temperatura atual, também pode ser vista como um sensor na perspectiva de uma aplicação.

De acordo com a forma que os dados são obtidos, os sensores podem ser classificados em três grupos: físicos, que compreendem os sensores de hardware; virtuais, que correspondem aos sensores de software; e lógicos, que representam uma combinação dos dois anteriores, podendo ser utilizada também outras fontes de dados.

Chen (2004) define três abordagens para adquirir informações contextuais: (1) acesso direto aos sensores, onde o código para esse acesso se encontra dentro da aplicação; (2) uso de uma infra-estrutura de *middleware*, que introduz uma arquitetura em camadas com a intenção de esconder detalhes de baixo nível; e (3) uso de servidores de contexto, que estende a arquitetura baseada em *middleware*, permitindo o acesso de múltiplos clientes a fontes de dados remotas.

É comum esconder das aplicações a forma como uma determinada informação de contexto foi adquirida. Por exemplo, em um ambiente, podem existir diferentes métodos para determinar a localização de uma pessoa (e.g., um conjunto de coordenadas, o nome de um lugar). Neste caso, todos esses métodos são utilizados para retornar a mesma informação de contexto e qualquer um deles pode ser denominado de "sensor de localização", independente do mecanismo de aquisição utilizado. Portanto, uma aplicação que utilize essa abstração pode executar independentemente do método empregado, possibilitando que esses métodos sejam modificados sem a necessidade de modificar a própria aplicação.

A camada de pré-processamento não está presente em todos os sistemas, mas pode ser útil, caso os dados obtidos pela camada de recuperação de dados ainda não estejam adequados para uso. Por exemplo, as exatas coordenadas de uma pessoa obtidas com um GPS (*Global Positioning System*) podem não ter importância para uma determinada aplicação, mas, possivelmente, pode ser relevante o nome da cidade na qual a pessoa se encontra. Essa camada é responsável por inferir e interpretar informações contextuais. Para isso, podem ser utilizadas combinações de dados de

diferentes fontes para que as informações mais precisas sejam fornecidas. Já a camada de gerenciamento é responsável por armazenar e disponibilizar as informações obtidas. Podemos dizer, então, que essas duas camadas são responsáveis pela capacidade de "pensar".

Uma vez que as informações de contexto são adquiridas e interpretadas, ações são executadas. A camada de mais alto nível é a de aplicação, que é responsável por reagir às diferentes mudanças de contexto. Essas mudanças podem ocorrer, por exemplo, em função da tarefa do usuário, do tipo do usuário ou do ambiente. Essa camada representa a capacidade de "agir".

Cada uma dessas capacidades pode ser muito complexa. Assim, elas podem ser disponibilizadas de forma separada ou integradas em um mesmo dispositivo. Além disso, as camadas precisam ser bem definidas e devem prover interfaces genéricas, possibilitando que modificações em uma camada não afetem as outras.

Dessa forma, para a construção de aplicações sensíveis ao contexto, algumas das principais etapas identificadas, a partir da literatura (BALDAUF *et al.*, 2007; LOKE, 2006; COUTAZ *et al.*, 2005; CHEN, 2004), são: a definição de contexto no domínio da aplicação e a escolha de um modelo de captura e representação de contexto. Após essas etapas, é possível tratar as questões relacionadas à adaptação das aplicações baseadas nessas informações.

### 2.3.3 Adaptação

A adaptação em ambientes de computação ubíqua e sensível ao contexto é fundamental para as aplicações. Elas precisam se ajustar às mudanças nesses ambientes sem a necessidade de intervenção do usuário (LYYTINEN e YOO, 2002).

A adaptação é orientada por uma política de adaptação, que pode ter sua escolha influenciada pelo tipo da aplicação e pelos tipos de dados (ARAUJO, 2003). Essas políticas podem ser agrupadas em duas classes: dados e controle (LARA *et al.*, 2001). A adaptação de dados implica em uma transformação dos dados usados pela aplicação para versões que se adaptam aos recursos disponíveis. Por exemplo, as imagens em um documento podem ter sua resolução modificada de acordo com o dispositivo onde estão sendo apresentadas. Já a adaptação de controle, principal foco deste trabalho, implica na modificação do fluxo de controle da aplicação, ou seja, no seu comportamento. Como exemplo, temos uma aplicação que pode, a partir da notificação de aumento do atraso na

rede, introduzir um *buffer* de recepção de dados, em tempo de execução, para diminuir a latência.

Um modelo de adaptação deve levar em consideração as seguintes questões: quando adaptar, o que adaptar e quanto adaptar (BECKER *et al.*, 2004). Dependendo do modelo de adaptação escolhido, a aplicação pode ser adaptada em tempo de construção ou em tempo de execução. No primeiro caso, diferentes versões da aplicação devem ser geradas de acordo com as características específicas do ambiente de execução alvo. Já no segundo caso, a aplicação desenvolvida deve ser capaz de identificar o ambiente de execução onde ela se encontra e adequar seu comportamento a esse novo contexto, sendo esse o caso de maior interesse no desenvolvimento desta dissertação.

Para a implementação desses modelos de adaptação, existem duas abordagens gerais: adaptação por parâmetro e adaptação composicional (MCKINLEY *et al.*, 2004).

Na abordagem por parâmetro, o comportamento da aplicação é determinado por meio da modificação das suas variáveis. Por exemplo, o TCP (*Transmission Control Protocol*) ajusta seu comportamento por meio da mudança de valores que controlam a gerência da janela de controle e as retransmissões em resposta a um aparente congestionamento da rede. Um ponto fraco nessa abordagem é que ela não permite que novos algoritmos e componentes sejam adicionados à aplicação depois do seu projeto e construção originais. Ela pode direcionar uma aplicação a utilizar uma política de adaptação diferente já existente, mas não pode adotar uma nova política (MCKINLEY *et al.*, 2004).

Por outro lado, a adaptação composicional permite que uma aplicação possa, por exemplo, adotar novos algoritmos para atender a requisitos imprevistos durante o seu desenvolvimento. Além disso, permite uma recomposição dinâmica da aplicação em tempo de execução. Por exemplo, ela permite a troca de componentes de uma aplicação para sua execução em dispositivos com memória limitada ou a adição de um novo comportamento nos sistemas já desenvolvidos. Essa composição pode ser realizada de forma estática ou dinâmica. Na composição estática, o desenvolvedor pode combinar vários componentes em tempo de compilação para produzir uma aplicação. Já na composição dinâmica, os componentes podem ser adicionados, removidos e modificados dentro de uma aplicação em tempo de execução, sendo essa estratégia mais adequada para lidar com o comportamento dinâmico dos sistemas sensíveis ao contexto (MCKINLEY *et al.*, 2004).

## 2.4 Modelagem de Contexto

No desenvolvimento de sistemas sensíveis ao contexto, a elaboração de um modelo de contexto é necessária para capturar a estrutura e a semântica das informações de contexto e identificar como essas informações devem ser manipuladas.

Segundo Vieira (2008), os trabalhos relacionados à modelagem de contexto em geral possuem foco em: identificar técnicas de representação que sejam mais adequadas às características das informações de contexto (e.g., (STRANG e LINNHOFF-POPIEN, 2004)); enumerar elementos que podem ser considerados contexto em um domínio ou um conjunto de aplicações (e.g., (SOUZA *et al.*, 2008)); e guiar a modelagem por meio de modelos de contexto genéricos (e.g., (BULCÃO NETO e PIMENTEL, 2005)) e metamodelos (e.g., (VIEIRA, 2008)).

As técnicas para modelagem de contexto podem ser classificadas em diferentes categorias, com base na estrutura de dados utilizada para representar as informações de contexto (STRANG e LINNHOFF-POPIEN, 2004). As subseções a seguir descrevem cada uma dessas técnicas em detalhes. É importante destacar que essas técnicas não são excludentes entre si e as abordagens para modelagem de contexto podem ser baseadas em uma combinação de técnicas de diferentes categorias.

### 2.4.1 Modelos chave-valor

Os modelos chave-valor representam uma das formas mais simples de se representar contexto. As informações de contexto são descritas por meio de pares compostos por uma chave que identifica a informação e um valor associado a essa chave. Nesse modelo, normalmente, as informações de contexto são representadas apenas textualmente ou graficamente por tabelas. Uma instância desse modelo é apresentada na Figura 2.3.

| Chave | Valor |
|---|---|
| Localização | Fortaleza, Ceará |
| Idade | 20 a 30 anos |
| Estado civil | Solteiro |

**Figura 2.3. Exemplo de modelo chave-valor.**

Com base nessas instâncias, podem ser associadas ao sistema ações que devem ser tomadas caso as informações de contexto obtidas apresentem esses valores. Por exemplo, para um sistema de guia turístico móvel, os valores apresentados na Figura 2.3 podem resultar na inclusão de um novo módulo para busca de atrações em Fortaleza

voltadas exclusivamente para solteiros nessa faixa etária. Devido a sua simplicidade, essa técnica é fácil de usar, porém torna a manutenção dos modelos mais difícil e é ineficiente para problemas complexos (VIEIRA, 2008).

### 2.4.2 Modelos baseados em esquemas de marcação

Os modelos que fazem parte desta categoria utilizam uma estrutura de dados hierárquica composta por *tags* com atributos e valores, baseadas no padrão XML (*eXtensible Markup Language*), para representar as informações de contexto.

Os perfis (*profiles*) são representantes dessa classe de modelos. O CSCP (*Comprehensive Structured Context Profiles*) (BUCHHOLZ *et al.*, 2004) é um exemplo de abordagem que utiliza essa técnica. A Figura 2.4 apresenta um exemplo de perfil CSCP. Neste exemplo, temos a descrição do perfil da entidade de contexto Dispositivo Móvel (*DeviceProfile*), por meio da informação de contexto Memória (*Memory*). Apesar de esses modelos permitirem certo grau de validação da sua estrutura, eles não oferecem mecanismos eficientes para representar relacionamento e restrições entre as informações representadas (STRANG e LINNHOFF-POPIEN, 2004).

```
<?xml version="1.0"encoding="UTF-8"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:cscp="context-aware.org/CSCP/CSCPProfileSyntax#"
    xmlns:dev="context-aware.org/CSCP/DeviceProfileSyntax#"
    xmlns:net="context-aware.org/CSCP/NetworkProfileSyntax#"
    xmlns="context-aware.org/CSCP/SessionProfileSyntax#"
    <SessionProfile rdf:ID="Session">
        <cscp:default  rdf:resource=
            http://localContext/CSCPProfile/previous#Session"/>
        <device><dev:DeviceProfile>
            <dev:hardware><dev:Hardware>
                <dev:memory>9216</dev:memory>
            </dev:Hardware></dev:hardware></dev:DeviceProfile>
        </device>
    </SessionProfile>
</rdf:RDF>
```

**Figura 2.4. Exemplo de um perfil CSCP (BUCHHOLZ *et al.*, 2004).**

### 2.4.3 Modelos gráficos

Essa categoria de modelos utiliza elementos gráficos para a representação de contexto e inclui as abordagens baseadas em UML (*Unified Modeling Language*), ORM (*Object Role Modeling*) e grafos contextuais.

As abordagens baseadas em UML utilizam extensões dessa linguagem, como perfis e estereótipos, para representar informações de contexto. Um exemplo é o CMP (*Context UML Profile*) (SIMONS, 2007) ilustrado na Figura 2.5. Neste exemplo, temos a representação do relacionamento entre a entidade de contexto Pessoa e as outras entidades do domínio, chamadas de *ContextItem*.

Nas abordagens baseadas em ORM (HENRICKSEN e INDULSKA, 2006) a modelagem envolve a identificação de fatos e papéis executados pelas entidades. Já os grafos contextuais (BRÉZILLON, 2005) são baseados em redes semânticas e suportam a representação de modelos de tarefas. Eles consideram a relação entre os procedimentos estabelecidos por uma organização para executar uma determinada tarefa e a forma como as informações de contexto influenciam essa execução na prática. Os modelos gráficos possibilitam um melhor entendimento da estrutura das informações contextuais, mas, normalmente, possuem baixo nível de formalidade (STRANG e LINNHOFF-POPIEN, 2004).

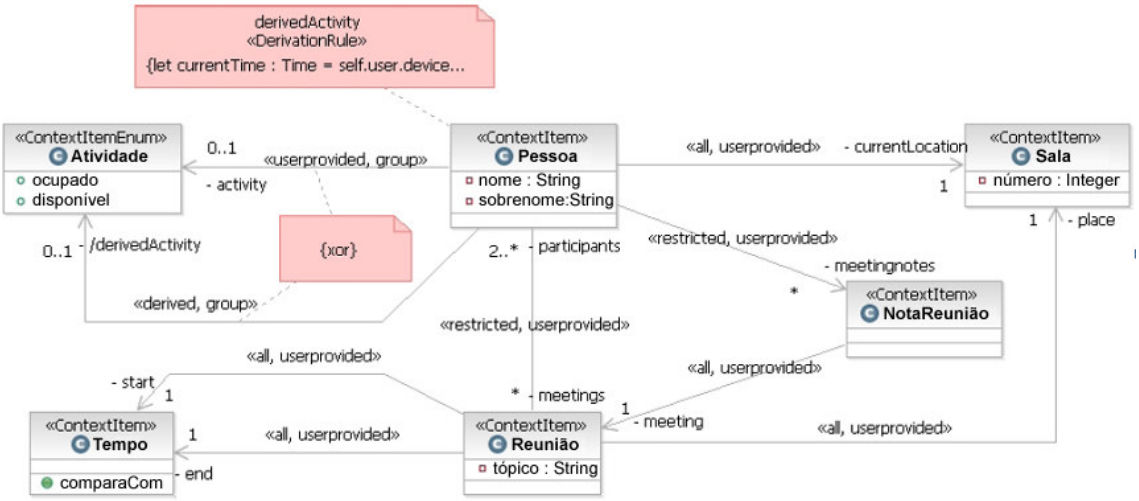



**Figura 2.5. Exemplo de uso do CMP [Adaptada de (SIMONS, 2007)].**

### 2.4.4 Modelos orientados a objetos

As abordagens orientadas a objetos tentam explorar os benefícios desse paradigma, como, por exemplo, encapsulamento e reusabilidade. Os detalhes de processamento de contexto são encapsulados no nível de objetos. O acesso às informações de contexto é feito apenas por meio de interfaces.

Um exemplo dessa abordagem (HENRICKSEN *et al.*, 2002) é apresentado na Figura 2.6. As informações de contexto são representadas por um conjunto de entidades, que, por sua vez, descrevem objetos físicos ou conceituais, como um canal de

comunicação ou uma pessoa. As propriedades das entidades, como o nome da pessoa ou o identificador do canal de comunicação, são representadas por atributos. Uma entidade está relacionada com os seus atributos e outras entidades por meio de relacionamentos de associação. Os modelos orientados a objetos são fortes em relação à possibilidade de composição distribuída, pois novos tipos de informações de contexto podem ser adicionados e instâncias podem ser atualizadas de forma distribuída. Porém, normalmente, as infra-estruturas de execução que suportam esses modelos exigem muitos recursos dos dispositivos computacionais, o que nem sempre pode ser atendido em ambientes de computação ubíqua (STRANG e LINNHOFF-POPIEN, 2004).

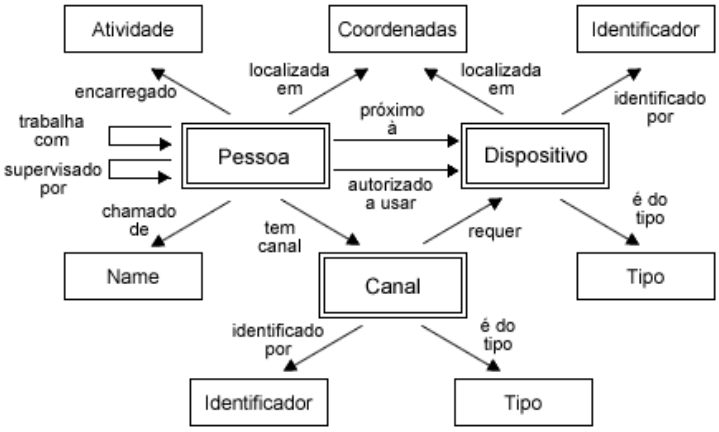



**Figura 2.6. Exemplo de modelo orientado a objetos [Adaptada de (HENRICKSEN *et al.*, 2002)].**

### 2.4.5 Modelos baseados em lógica

Os modelos baseados em lógica possuem um maior grau de formalismo. Os modelos de contexto são definidos por meio de fatos, expressões e regras. Além disso, um processo de inferência pode ser utilizado para derivar novos fatos com base nas regras modeladas. Essa classe de modelos inclui, por exemplo, abordagens baseadas em lógica de primeira ordem (KATSIRI e MYCROFT, 2003).

A Figura 2.7 ilustra parte de um modelo desse tipo relacionado à localização do usuário de uma determinada aplicação. Como mencionado anteriormente, os modelos baseados em lógica possuem alto grau de formalismo, o que implica em uma maior dificuldade de manutenção e compreensão (STRANG e LINNHOFF-POPIEN, 2004).

```
(H_UserAtLocation (uid ?uid)(rid ?rid)
    (start-time ?start-time))
(H_UserColocation (uid-list ?uid1 … ?uidn)
    (rid ?region-id)(start-time ?time-value))
(H_UserIsPresent (uid ?uid)(start-time ?time-value))
(L_UserAtLocation (uid ?uid)(x ?x) (y ?y) (z ?z))
```

**Figura 2.7. Exemplo de parte de um modelo baseado em lógica (KATSIRI e MYCROFT, 2003).**

### 2.4.6 Modelos baseados em ontologia

Essa categoria de modelos utiliza ontologias para representar conceitos e suas relações. Ontologia pode ser definida como uma especificação explícita e formal de uma conceitualização compartilhada (STUDER *et al.*, 1998).

A Figura 2.8 ilustra parte de uma ontologia para modelagem de contexto (ALMEIDA *et al.*, 2006). De acordo com os autores, essa ontologia foi definida utilizando a linguagem OWL (*Web Ontology Language*). Os três principais conceitos utilizados nessa modelagem para representação das informações de contexto são: classes (e.g., usuário), propriedades de classes (e.g, nome do usuário) e relacionamentos entre classes (e.g., localizado em). Entre as vantagens do uso de ontologias estão a possibilidade de reúso e a utilização de mecanismos de inferência existentes. Porém, as ferramentas e padrões para manipular ontologias ainda não são de fácil uso e os mecanismos de inferência impactam diretamente no desempenho das aplicações (VIEIRA, 2008).

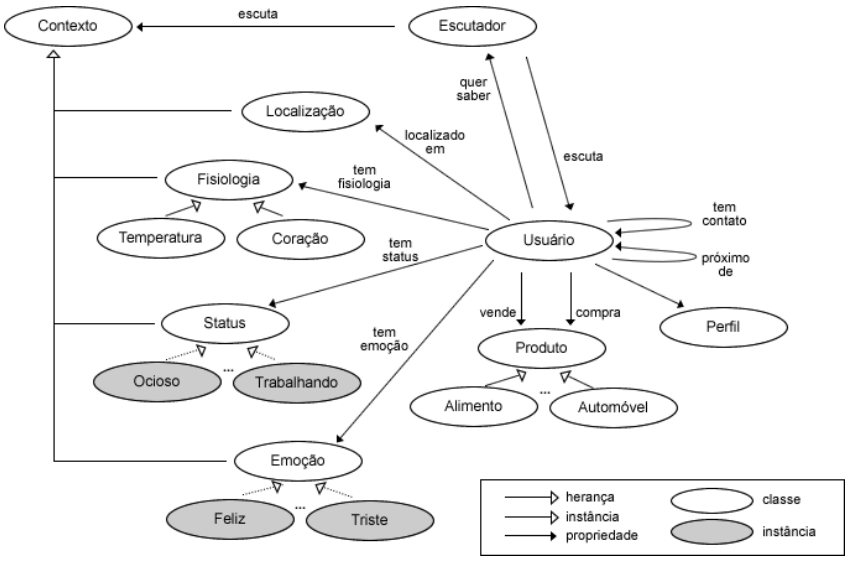



**Figura 2.8. Exemplo de ontologia para modelagem de contexto (ALMEIDA *et al.*, 2006).**

## 2.5 Considerações finais

Neste capítulo, foram apresentados os conceitos relacionados à computação sensível ao contexto relevantes para este trabalho de pesquisa. Foram apresentadas as definições de contexto e aplicações sensíveis ao contexto que serão utilizadas nesta dissertação e discutidas as principais categorias de técnicas para modelagem de contexto, além de algumas das suas vantagens e desvantagens.

Apesar da abordagem proposta nesta dissertação estar mais focada no desenvolvimento de sistemas ubíquos que se adaptam ao contexto, a escolha de uma definição mais ampla de aplicações sensíveis ao contexto, apresentada na Seção 2.3, não restringe o uso e a extensão da abordagem para outras classes de aplicações, incluindo outras áreas de aplicações, como, por exemplo, sistemas colaborativos. Além disso, essa definição não exige que a detecção e a interpretação do contexto de execução sejam feitas pelas próprias aplicações, podendo ser utilizado um mecanismo externo para isso. Dessa forma, esses mecanismos não fazem parte do escopo deste trabalho.

Este trabalho busca contribuir para o desenvolvimento de aplicações sensíveis ao contexto por meio do uso do paradigma de linha de produtos, possibilitando que o reúso ocorra de forma sistemática e que novos produtos sejam facilmente configurados. No próximo capítulo, são apresentados os conceitos relacionados a linhas de produtos de software.

# Capítulo 3 – Linha de Produtos de Software

## 3.1 Introdução

Como apresentado no Capítulo 2, o desenvolvimento de aplicações sensíveis ao contexto é uma atividade complexa, que envolve diferentes etapas, desde a identificação dos contextos relevantes para a aplicação até a implementação de possíveis estratégias de adaptação.

O paradigma de Linha de Produtos de Software (LPS) tem como proposta a construção sistemática de software baseada em uma família de produtos. Um dos principais pontos em uma LPS é a identificação das características comuns e variáveis entre os produtos, possibilitando um aumento na flexibilidade e adaptação às necessidades dos clientes. Dessa forma, o desenvolvimento de aplicações sensíveis ao contexto pode se beneficiar desse paradigma em termos de reusabilidade e configurabilidade.

Este capítulo está organizado da seguinte forma: na Seção 3.2, são apresentadas algumas definições de LPS; na Seção 3.4 são apresentados os conceitos relacionados à modelagem de características; na Seção 3.5 são discutidos os trabalhos relacionados; e finalmente, na Seção 3.6, são apresentadas as considerações finais.

## 3.2 Definição

O conceito de famílias de produtos foi introduzido por PARNAS (1976) na década de 70 e, inicialmente, levava em consideração apenas a variabilidade de requisitos não funcionais entre os produtos. O conceito de Engenharia de Domínio, que, posteriormente, deu origem ao termo LPS, só foi completamente introduzido no início da década de 90, tendo como uma das primeiras contribuições a descrição do método FODA (*Feature-Oriented Domain Analysis*) (KANG *et al.*, 1990).

Na literatura existem várias definições para LPS. No entanto, a definição com maior aceitação na indústria é dada por CLEMENTS e NORTHROP (2002):

"Uma linha de produtos de software é um conjunto de sistemas que usam software intensivamente, compartilhando um conjunto de características comuns e gerenciadas, que satisfazem as necessidades de um segmento em particular de mercado

ou missão, e que são desenvolvidos a partir de um conjunto comum de ativos principais e de uma forma preestabelecida".

Os ativos, também chamados de núcleo de artefatos (*core assets*), são a essência da linha de produtos e correspondem a um conjunto de elementos customizáveis, utilizados na construção das aplicações produzidas. O núcleo inclui artefatos como, por exemplo: modelo do domínio, arquitetura e componentes reutilizáveis que serão integrados à arquitetura para gerar os produtos (GIMENES e TRAVASSOS, 2002). Dentre esses elementos, a arquitetura é o elemento chave (DURSCKI *et al.*, 2004). Além disso, uma LPS atua sobre um segmento particular ou missão, também chamado de domínio, que se refere à área de conhecimento ou especialização em que ela atua. O termo domínio também é utilizado para referenciar uma coleção de aplicações existentes ou futuras que compartilham características comuns (WERNER e BRAGA, 2005).

Um dos benefícios de LPS é o seu alto nível de reutilização, reduzindo o *time-to-market* e requerendo um menor número de pessoas para a produção de software (COALLIER e CHAMPAGNE, 2005).

## 3.3 Ciclo de desenvolvimento

Como mencionado no Capítulo 1, o paradigma de LPS serve como guia para as organizações, tanto na construção de artefatos reutilizáveis (desenvolvimento para reutilização), quanto na construção de novas aplicações a partir desses artefatos (desenvolvimento com reutilização). A Figura 3.1 ilustra o ciclo de desenvolvimento de uma LPS, destacando as atividades de Engenharia de Domínio (ED) e Engenharia de Aplicação (EA).

O processo de ED compreende o desenvolvimento para reutilização, que tem como principal objetivo desenvolver artefatos de software reutilizáveis para uma família de aplicações ou domínio. Esse processo pode ser dividido nas seguintes atividades (VAN DER LINDEN *et al.*, 2007):

- *Análise do domínio*, que é definida como o processo de identificar e organizar o conhecimento a respeito de uma classe de problemas, de maneira a suportar a descrição e solução de tais problemas (PRIETO-DIAZ e ARANGO, 1991). Esta atividade inclui a coleta de informações e conhecimento sobre uma coleção de sistemas, visando explicitar seu conjunto de similaridades e diferenças, organizados em um modelo de domínio;

- *Projeto do domínio*, responsável pelo desenvolvimento da arquitetura da LPS;
- *Implementação do domínio*, que compreende o detalhamento do projeto e a implementação dos componentes de software reutilizáveis;
- *Testes do domínio*, responsável por validar os componentes implementados na atividade anterior; e
- *Gerenciamento dos produtos*, que possui como objetivo definir o escopo da LPS, identificando os produtos que irão constituir a LPS como um todo.

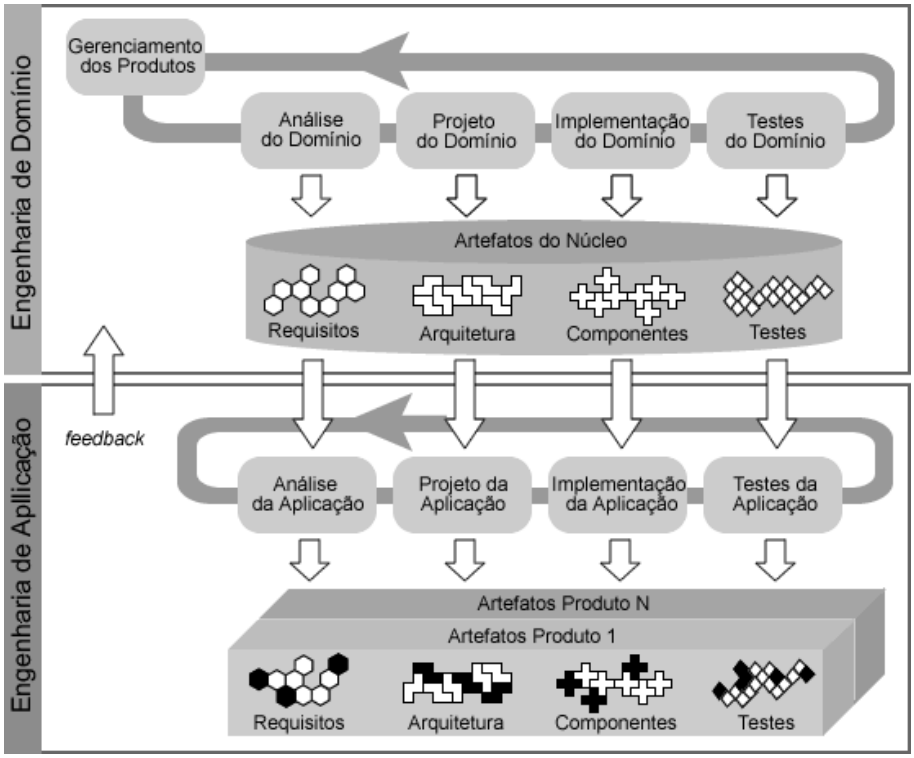



**Figura 3.1. Ciclo de desenvolvimento de uma LPS [Adaptada de (VAN DER LINDEN *et al.*, 2007)].**

Dentre os métodos de ED encontrados na literatura, estão o FODA (*Feature Oriented Domain Analysis*) (KANG *et al.*, 1990), o FORM (*Feature Oriented Reuse Method*) (KANG *et al.*, 2002) e o CBD-Arch-DE (BLOIS, 2006).

O processo de Engenharia de Aplicação (EA) compreende o desenvolvimento com reutilização, que consiste no processo de reutilização dos artefatos gerados durante o processo de ED para o desenvolvimento de produtos. As atividades que compreendem esse processo são (VAN DER LINDEN *et al.*, 2007):

- *Análise da aplicação*, que tem como objetivo identificar os requisitos específicos de um determinado produto, tendo como ponto de partida o modelo de domínio da LPS, avaliando as funcionalidades já disponibilizadas no domínio e o alinhamento do produto com a LPS;

- *Projeto da aplicação*, responsável por instanciar e adaptar a arquitetura da LPS de acordo com os requisitos identificados na atividade anterior;
- *Implementação da aplicação*, que compreende a implementação final do produto baseada nos requisitos e na arquitetura definidos e inclui o reúso e configuração dos componentes existentes, assim como a construção de novos componentes específicos do produto; e
- *Testes da aplicação*, que inclui a validação do produto em relação aos seus requisitos.

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 diferentes níveis de abstração do domínio (MILER, 2000). Essa seleção é denominada "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.

É importante que a EA forneça *feedback* para a ED como forma de melhorar os artefatos do núcleo da LPS e garantir que a infra-estrutura provida se mantenha adequada para a produção de novos produtos.

## 3.4 Modelagem de características

O principal aspecto no desenvolvimento dos sistemas que fazem parte de uma LPS é o fato deles serem criados a partir de componentes vindos de uma base de artefatos comuns, moldados de acordo com regras definidas pela arquitetura. No entanto, como discutido na seção anterior, ao longo do processo de desenvolvimento de uma LPS, aspectos particulares de cada produto podem ser explicitados, constituindo a variabilidade da LPS.

Nesta seção, são apresentados: alguns conceitos relacionados à variabilidade de uma linha de produtos (Seção 3.4.1), detalhes sobre o modelo de características (Seção 3.4.2) e a notação para modelagem de características Odyssey-FEX (Seção 3.4.3).

### 3.4.1 Conceito de variabilidade

Variabilidade pode ser definida como a possibilidade de configuração, ou ainda, como a habilidade que um sistema ou artefato de software possui de ser alterado,

customizado, ou configurado para um contexto em particular (BOSCH, 2004). Esse é um conceito chave para a abordagem de LPS, pois permite tornar explícitos os pontos onde os produtos se assemelham, podendo ser reutilizados, e os pontos onde os produtos diferem entre si, devendo receber tratamento específico (BECKER, 2003). Dessa forma, a variabilidade deve ser gerenciada durante todo o processo de desenvolvimento.

A notação para modelagem de variabilidade pode ser gráfica, textual ou uma combinação das duas formas (MASSEN e LICHTER, 2002). Durante a atividade de análise do domínio, a variabilidade pode ser representada por meio de um modelo de características.

### 3.4.2 Modelo de características

A modelagem de características é uma atividade que trata da complexidade em expressar diversos requisitos em forma de características (MASSEN e LICHTER, 2004). O modelo de características, foco deste trabalho de pesquisa, representa as características de uma família de sistemas em um domínio, suas semelhanças e diferenças e as relações entre elas. Características (*features*) representam as capacidades ou abstrações do domínio obtidas por especialistas, usuários ou a partir de sistemas já existentes (BLOIS, 2006). Essas características podem representar, por exemplo, conceitos ou funcionalidades do domínio, como para o domínio de guias turísticos móveis apresentado no Capítulo 2, poderiam existir as características *Mapa* e *Visualizador de Imagens*.

O modelo de características possui um alto nível de abstração e pode ser considerado um elemento chave para relacionar os conceitos de mais alto nível aos demais artefatos de análise e projeto, sendo utilizado como ponto de partida para o recorte necessário à instanciação de novos produtos (BLOIS, 2006). Além disso, serve para padronizar o entendimento do domínio entre todos os envolvidos no processo de desenvolvimento, tais como usuários, especialistas de domínio e desenvolvedores (MASSEN e LICHTER, 2002).

As características em uma LPS podem ser classificadas quanto à variabilidade em três tipos (OLIVEIRA, 2006):

- *Pontos de variação*, que representam pontos onde decisões são tomadas, refletindo a parametrização no domínio de uma maneira abstrata, e são configuráveis por meio de variantes;

- *Variantes*, que representam características que são alternativas disponíveis para configurar um ponto de variação; e
- *Invariantes*, que compreendem as características "fixas", que não são configuráveis no domínio.

Em relação à opcionalidade, as características podem ser classificadas como (VAN DER LINDEN *et al.*, 2007; OLIVEIRA, 2006):

- *Comuns ou mandatórios*, que devem obrigatoriamente estar presentes em todos os produtos derivados a partir de uma LPS;
- *Opcionais*, que estão presente apenas em parte dos produtos e fazem parte da arquitetura da LPS; e
- *Específicas de produto*, que estão presentes apenas em um produto específico e não fazem parte da arquitetura da LPS.

Durante o ciclo de vida de uma LPS, as características podem variar sua classificação. Por exemplo, uma característica específica de produto pode vir a fazer parte da arquitetura da LPS e ser incorporada em outros produtos, passando a ser uma característica variável, devido à própria evolução da LPS.

Um dos conceitos relacionados à modelagem de características é a cardinalidade, que é utilizada para definir o número mínimo e máximo de características que podem ser escolhidas a partir de um conjunto de alternativas de um ponto de variação.

Outro conceito importante é a especificação de restrições entre as características, que inclui os relacionamentos de dependência e mútua exclusividade. Durante o desenvolvimento de um novo produto em uma LPS, deve haver uma forma de indicar a necessidade ou incompatibilidade da seleção conjunta de características. É importante destacar que tais indicações devem ser verificadas no processo de instanciação dos produtos de uma LPS, como forma de garantir a derivação de produtos mais consistentes em relação às especificações do domínio.

Na literatura, existem diferentes notações que utilizam o modelo de características para representar variabilidade, como, por exemplo, FODA (KANG *et al.*, 1990), notação de Czarnecki (CZARNECKI *et al.*, 2004), notação de Gomaa (GOMAA, 2004) e Odyssey-FEX (OLIVEIRA, 2006). A próxima seção apresenta detalhes sobre a notação Odyssey-FEX, utilizada como base para a abordagem proposta neste trabalho.

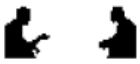
### 3.4.3 Notação Odyssey-FEX

A notação Odyssey-FEX (OLIVEIRA, 2006) foi desenvolvida com o objetivo de suprir algumas deficiências encontradas em outras notações de modelagem de características, como, por exemplo, a representação incompleta dos conceitos inerentes à variabilidade de uma família de produtos. Tais deficiências podem ocasionar uma representação inadequada da variabilidade, resultando em uma modelagem incorreta da família de produtos. Esse foi um dos principais motivos para a escolha dessa notação como base da abordagem proposta, além do apoio ferramental oferecido pela mesma.

Essa notação permite a disposição de características em forma de grafo. As características podem ser classificadas em três dimensões: *categoria*, *variabilidade* e *opcionalidade*.

Em relação à dimensão categoria, os tipos de características estão relacionados às diferentes fases do desenvolvimento de software (Tabela 3.1) e são mutuamente excludentes, não podendo uma característica pertencer simultaneamente a mais de uma categoria.

**Tabela 3.1. Tipos de características na notação Odyssey-FEX [Adaptada de (OLIVEIRA, 2006)].**

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

Quanto à variabilidade, as características podem ser classificadas como: *pontos de variação*, *variantes* ou *invariantes*. Como apresentado no Capítulo 3, pontos de variação representam conceitos ou funcionalidades que expressam alternativas, que são

representadas pelas características variantes. As invariantes representam características "fixas" ou não-configuráveis. Estes tipos também são mutuamente excludentes.

A dimensão opcionalidade indica a obrigatoriedade ou não da presença de determinada característica em um produto a ser desenvolvido dentro do domínio como um todo. De acordo com esse conceito, as características podem ser classificadas como *mandatórias* ou *opcionais*.

A Figura 3.2 apresenta as possíveis combinações na classificação de características na notação Odyssey-FEX. Uma característica pode ser, por exemplo, de análise, variante e opcional.

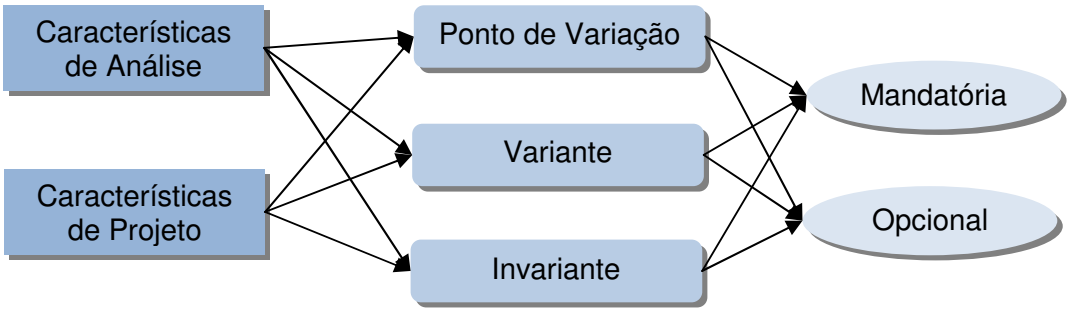



**Figura 3.2. Classificação de características na notação Odyssey-FEX [Adaptada de (OLIVEIRA, 2006)].**

Outro conceito importante, presente na notação Odyssey-FEX, são as regras de composição, utilizadas para expressar restrições existentes entre características. A semântica dessas regras influencia fortemente o recorte para a instanciação dos produtos. Existem dois tipos de regras de composição:

- *Inclusivas*, que definem relações de dependência entre duas ou mais características, indicando que elas devem ser selecionadas em conjunto para um determinado produto;
- *Exclusivas*, que definem relações de mútua exclusividade entre características, onde duas ou mais características não devem ser escolhidas em conjunto em um mesmo produto.

As regras de composição são expressas pela seguinte estrutura: antecedente, palavra-chave e conseqüente. A palavra-chave representa o tipo de regra: "requer" (*requires*), referente às regras inclusivas; e "exclui" (*excludes*), referente às regras exclusivas. Antecedente e conseqüente são expressões, que podem ser literais ou *booleanas* e representam uma característica ou combinação de características do domínio.

A notação Odysse-FEX valoriza a semântica dos relacionamentos em um modelo de características, buscando atingir uma maior capacidade de representação e expressão, combinando relacionamentos próprios com relacionamentos da UML (Tabela 3.2). Esses relacionamentos não são apenas hierárquicos, possibilitando a geração de grafos que permitem uma expansão do modelo em diversas direções.

**Tabela 3.2. Relacionamentos da notação Odyssey-FEX [Adaptada de (OLIVEIRA, 2006)].**

| | Representação | Descrição |
|---|---|---|
| **Relacionamentos UML** |  | **Composição** – Relacionamento em que uma característica é composta por outras, ou seja, uma característica é parte fundamental de outra e a primeira não existe sem a segunda. |
| |  | **Agregação** – Relacionamento em que uma característica representa o todo, e as outras as partes. No entanto, as características existem independentemente umas das outras. |
| |  | **Generalização** – Relacionamento em que há uma generalização/especialização das características. Denota relação de herança entre características. |
| |  | **Associação** – Relacionamento simples entre duas características. Pode ser nomeada, indicando um tipo específico de ligação. |
| **Relacionamentos Específicos** |  | **Alternativo (*Alternative*)** – Relacionamento entre um ponto de variação e as suas variantes, denotando a pertinência de uma variante a um determinado ponto de variação. |
| |  | **Implementado por (*Implemented By*)** – Relacionamento entre características de domínio e características tecnológicas, ou entre características tecnológicas de diferentes categorias. |
| |  | **Ligação de Comunicação (*Communication Link*)** – Relacionamento entre características de entidade e características de domínio. |

O conceito de cardinalidade foi introduzido de forma explícita recentemente na notação Odyssey-FEX (TEIXEIRA *et al.*, 2008). A cardinalidade pode ser definida por meio de um intervalo associado a um ponto de variação. Desta forma, podem ser determinados o número mínimo e o número máximo de variantes que podem ser selecionadas como alternativas de um ponto de variação. Podemos citar como exemplos: a cardinalidade 1-1, que representa um ponto de variação mandatório com variantes mutuamente exclusivas, indicando que somente uma de suas variantes deve ser selecionada na instanciação de aplicações; e a cardinalidade 0-1, que representa um ponto de variação opcional também com variantes mutuamente exclusivas.

A Figura 3.3 ilustra o exemplo de um modelo de características na notação Odyssey-FEX para o domínio de guias turísticos móveis (GRÜN *et al.*, 2008). Esse domínio, como apresentado no Capítulo 2, inclui aplicações que têm como objetivo

auxiliar o turista durante a sua viagem. Nesse exemplo, podemos destacar alguns dos conceitos apresentados nesta seção. A característica *Visualizador Informações Cidade* é uma característica invariante e mandatória. A característica *Visualizador Mapas* é um ponto de variação opcional configurado por duas características variantes do tipo funcional *Visualizador Imagem* e *Visualizador Mapa 3D*. A opcionalidade é indicada pela linha tracejada na representação da característica. Existem ainda relacionamentos da UML, como de composição e de agregação, e relacionamentos do tipo alternativo entre os pontos de variação e suas variantes com cardinalidade 0-1. O tipo da característica é representado tanto pelo ícone em marca d'água quanto pelo estereótipo acima do seu nome.

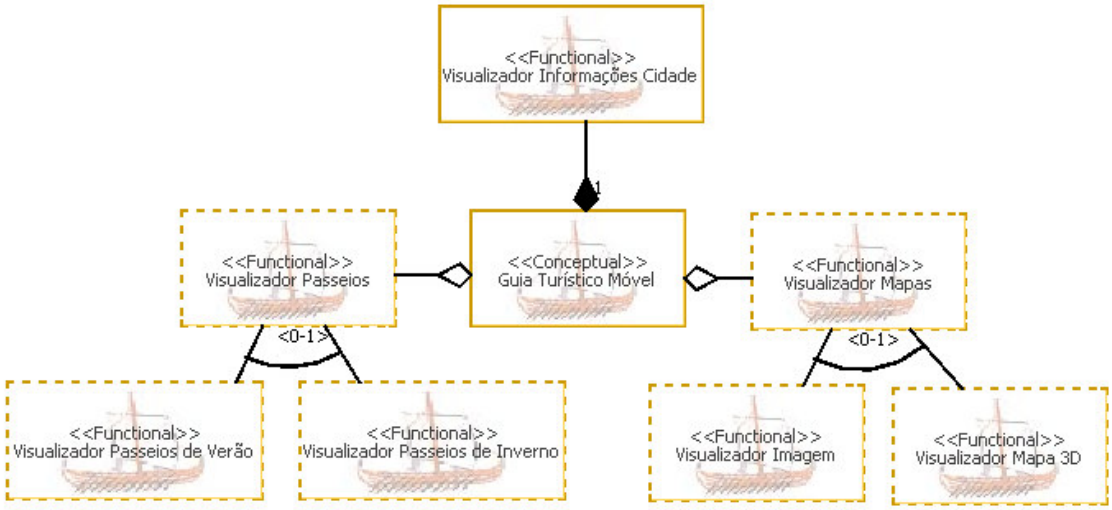



**Figura 3.3. Exemplo de um modelo de características na notação Odyssey-FEX.**

## 3.5 Trabalhos relacionados

Grande parte das abordagens existentes para o desenvolvimento de LPS trata as variabilidades de cada produto de forma estática, focando apenas em uma adaptação em tempo de desenvolvimento. No entanto, como foi apresentado no Capítulo 2, as aplicações sensíveis ao contexto possuem comportamento dinâmico e precisam se adaptar durante sua execução em função das variações de contexto. Além disso, grande parte dessas aplicações é executada em dispositivos móveis com capacidade limitada de recursos. Dessa forma, a decisão relacionada à configuração do produto precisa ser feita também em tempo de execução.

Esta seção está organizada em duas subseções. Na Seção 3.5.1, são discutidas algumas abordagens de LPS que tratam a variabilidade apenas em tempo de desenvolvimento e que, de alguma forma, utilizam informações de contexto, mesmo que

estáticas, para realizar a configuração dos produtos. Na Seção 3.5.2, são discutidas abordagens que já apresentam a preocupação de adaptar técnicas existentes de LPS para suportar a configuração de produtos de forma mais dinâmica e adaptável.

### 3.5.1 Abordagens relacionadas à configuração estática de produtos

Nesta classe de abordagens, as decisões em relação à configuração dos produtos são realizadas em tempo de desenvolvimento. Dessa forma, diferentes configurações de um produto são geradas para atender às necessidades dos clientes e às características do ambiente de execução.

#### 3.5.1.1 Aplicações para dispositivos móveis

ALVES *et al.* (2005) apresentam uma abordagem extrativa e incremental usando linha de produtos para portar jogos para outras plataformas utilizando programação orientada a aspectos (KICZALES *et al.*, 1997). Nesse trabalho, algumas das variações tratadas em relação aos jogos foram: tamanho da tela e tamanho máximo da aplicação. A linha de produtos foi implementada com a parte não variável em classes Java e toda a parte variável em aspectos. A Figura 3.4 apresenta uma visão geral dessa abordagem.

Uma abordagem semelhante é descrita por YOUNG (2005). O processo de construção da linha de produtos proposta envolve um conjunto de classes base para as aplicações e um conjunto de aspectos que implementam as características opcionais, que combinados de diferentes formas geram produtos para diferentes dispositivos. Para validar a abordagem, é apresentado um estudo de caso para a criação de diferentes versões de uma aplicação chamada *MobilePhoto*. Algumas das variabilidades apresentadas se referem à escolha de funcionalidades de e-mail, SMS (*Short Message Service*), MP3 e foto digital, dependendo do dispositivo móvel selecionado.

Nas abordagens apresentadas nesta seção, versões diferentes de um mesmo produto são configuradas estaticamente, durante o processo de desenvolvimento, com base em características específicas do dispositivo no qual serão executadas.

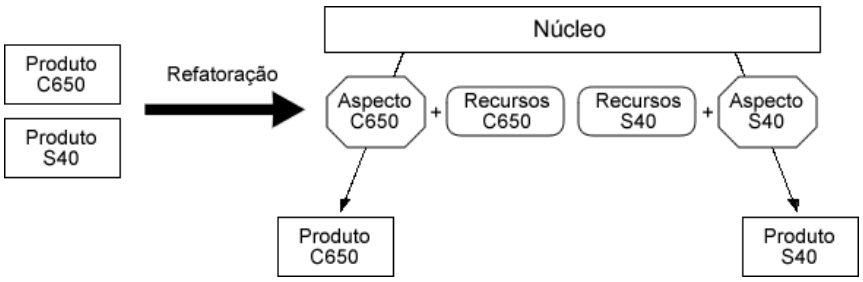



**Figura 3.4. Visão geral da abordagem [Adaptada de (ALVES *et al.*, 2005)].**

#### 3.5.1.2 Modelo de variabilidade de contexto

HARTMANN e TREW (2008) definem o conceito de modelo de variabilidade de contexto, que é utilizado para aplicar restrições no modelo de características, possibilitando a modelagem de múltiplas linhas de produtos. Variabilidade de contexto é definida como a variabilidade do ambiente do produto. O modelo de variabilidade de contexto é formado por: classificadores gerais de contexto (e.g., região geográfica, tipo do usuário), subclassificadores (e.g., Europa, Ásia), cardinalidade e dependências entre classificadores. Além disso, classificadores, também chamados de pontos de variação de contexto, são sempre opcionais. Esse modelo é representado utilizando a notação FODA, mas os classificadores não são considerados como características e os subclassificadores são chamados de subcaracterísticas.

O modelo de variabilidade de contexto é combinado com o modelo de características convencional para criar um novo modelo, denominado modelo de características de múltipas linhas de produtos (*MPL- Feature model*). O relacionamento entre os dois modelos combinados é representado por meio de dependências do tipo requer (*requires*) e exclui (*excludes*) e de um tipo de relacionamento (*sets cardinality*) que determina a cardinalidade de um grupo de características no modelo convencional.

A Figura 3.5 ilustra um exemplo da aplicação da abordagem em um modelo para o domínio da indústria automotiva, mais especificamente para sistemas de informação e entretenimento. Nesse modelo, temos, por exemplo, que o *Carro do Fabricante A* não suporta o protocolo *Flexray*. O *Tipo Econômico* (*Type Budget*) altera a cardinalidade do grupo *Interface*, indicando que não é possível escolher mais de uma interface para essa configuração. Os relacionamentos de dependência também podem ocorrer entre os classificadores de contexto, por exemplo, o *Carro A* nos *Estados Unidos* nunca possui *Tipo Econômico* (*Type Budget*). Dessa forma, à medida que as decisões sobre os classificadores de contexto são tomadas, o modelo se torna mais especializado.

Essa é uma das poucas abordagens que apresentam o contexto de forma explícita no modelo de características. No entanto, o contexto é tratado apenas de forma estática e o conceito de entidades de contexto não é representado. Além disso, a representação dos modelos em um diagrama unificado para uma LPS de maior escala pode dificultar a visualização dos relacionamentos entre os modelos.

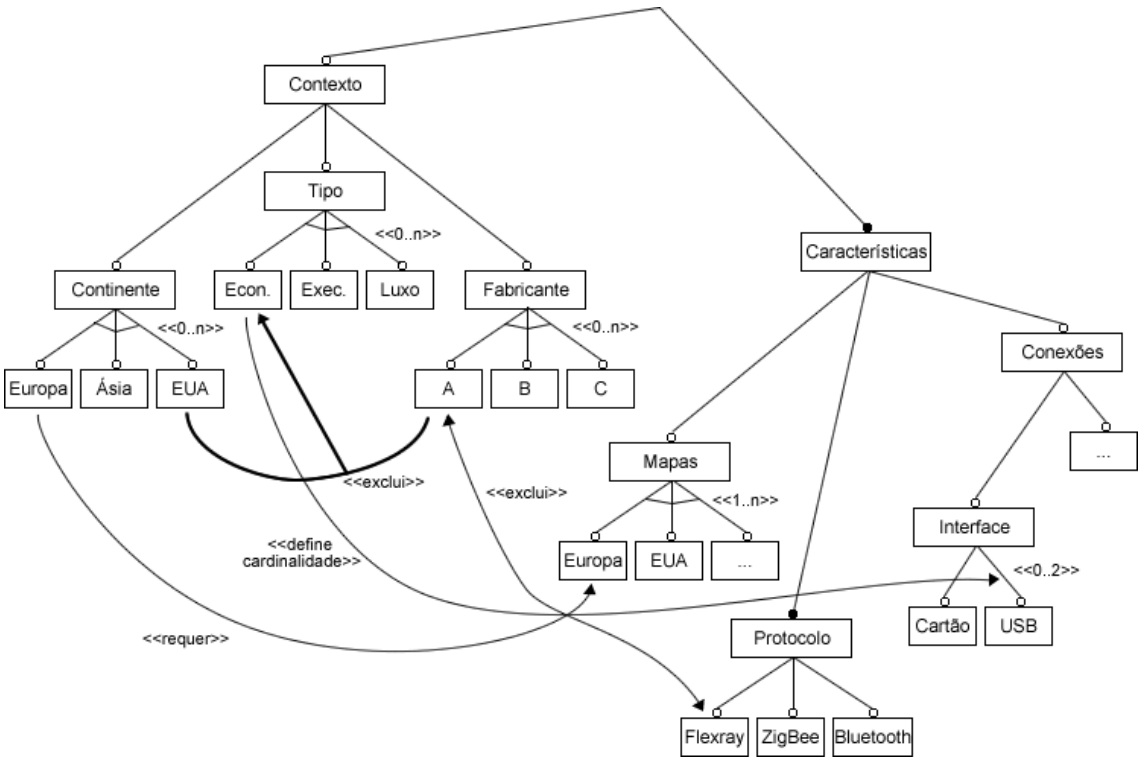



**Figura 3.5. Exemplo de um modelo de características de múltipas linhas de produtos [Adaptada de (HARTMANN e TREW, 2008)].**

### 3.5.2 Abordagens relacionadas à configuração dinâmica de produtos

Essa classe de abordagens busca aumentar a dinamicidade no tratamento das variabilidades em LPS, de forma que as características de um produto possam ser selecionadas em tempo de execução pelo usuário ou pelo próprio produto, quando uma mudança contextual é reconhecida.

#### 3.5.2.1 Variabilidade a qualquer momento

O conceito de variabilidade a qualquer momento (*any-time variability*) é a habilidade de um artefato de software variar seu comportamento em qualquer ponto do seu ciclo de vida, tanto estaticamente em tempo de projeto, quanto dinamicamente, em tempo de execução (VAN DER HOEK, 2004; VAN GURP *et al.*, 2001). O tratamento das variabilidades em tempo de execução permite que o próprio sistema determine sua instância particular, baseado no contexto em que executa.

Segundo VAN DER HOEK (2004), uma solução para suportar esse conceito deve prover quatro elementos principais: uma representação para capturar as variabilidades dos sistemas, uma ferramenta para especificar essas variabilidades, uma ferramenta para resolver variabilidades e um conjunto de ferramentas que apliquem o

resultado dessa resolução em diferentes pontos do ciclo de vida. Nessa abordagem, a arquitetura da LPS é descrita utilizando a linguagem xADL (DASHOFY *et al.*, 2002). Alguns conceitos presentes nessa linguagem são: *opções*, que representam pontos de variação opcionais; *variantes*, que representam pontos de variação obrigatórios, mas que podem ser configurados por um conjunto de alternativas obrigatórias ou opcionais; e *expressões de guarda*, que representam expressões "booleanas" associadas à seleção de cada um dos elementos opcionais. Essas expressões podem ser utilizadas em tempo de execução para determinar a reconfiguração do sistema.

A Figura 3.6 ilustra o processo de reconfiguração utilizado por essa abordagem. Inicialmente, o contexto é detectado e uma configuração do sistema é escolhida. Em seguida, essa configuração é comparada com a configuração corrente e as modificações necessárias são aplicadas para gerar a nova configuração.

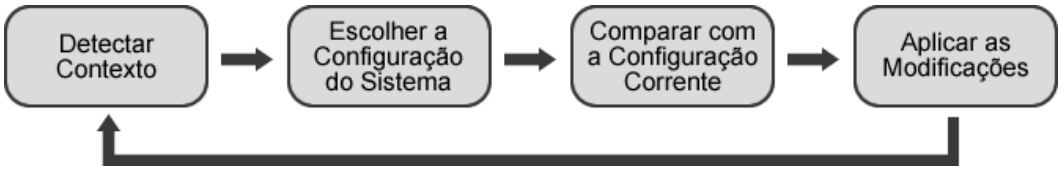



**Figura 3.6. Processo de reconfiguração em tempo de execução [Adaptada de (VAN DER HOEK, 2004)].**

Um sistema colaborativo foi desenvolvido utilizando essa abordagem e a variabilidade tratada era o uso de comunicação segura ou não. Essa decisão era tomada com base na forma como a aplicação estava conectada, se em uma rede fixa ou sem fio. No entanto, essa abordagem não explora a forma como as informações de contexto que formam as expressões de guarda são identificadas e nenhum mecanismo de verificação da consistência das configurações dos produtos é apresentado.

No trabalho de VAN GURP *et al.* (2001), além da utilização do conceito de variabilidade a qualquer momento, é introduzido o conceito de *binding time*, que representa o momento da decisão a respeito da variabilidade dos produtos em uma LPS, que pode ocorrer, por exemplo, em tempo de projeto, compilação ou execução. Porém, não foi apresentado nenhum mecanismo que representasse os elementos que influenciam nessa decisão em tempo de execução. A Figura 3.7 apresenta um modelo de características que ilustra o conceito de *binding time*. O autor classifica os pontos de variação em abertos e fechados, indicando, respectivamente, se novas variantes podem ou não ser adicionadas, mas não são apresentados detalhes de como, por exemplo, novas variantes poderiam ser adicionadas em tempo de execução.

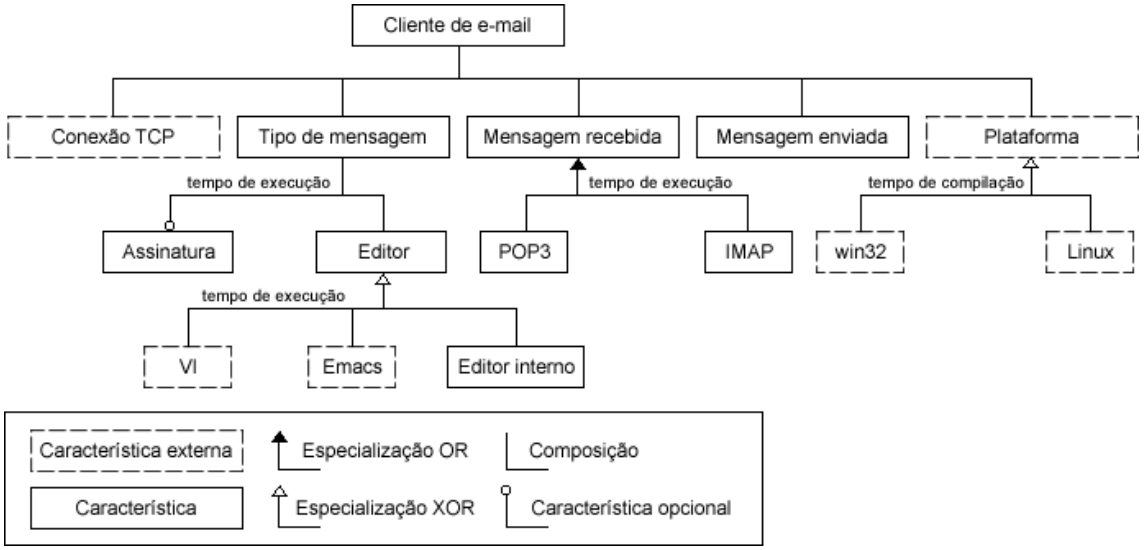



**Figura 3.7. Modelo de características com o conceito de *binding time* [Adaptada de (VAN GURP *et al.*, 2001)].**

### 3.5.2.2 Diagrama de *binding*

LEE e MUTHING (2006) propõem uma abordagem onde o modelo de características é acrescido de um diagrama de *binding* para prover a reconfiguração dos produtos de uma LPS em tempo de execução. Esse diagrama captura as unidades de *binding* (*Feature Binding Units* ou FBUs), que são conjuntos de características relacionadas em um modelo de características para realizar uma tarefa em comum, e os relacionamentos entre essas unidades. Os relacionamentos entre FBUs podem ser estáticos ou dinâmicos, sendo estes últimos acrescidos de pré-condições que indicam a seleção de uma determinada FBU em tempo de execução.

A Figura 3.8 representa um diagrama de *binding* de uma LPS para escritórios virtuais do futuro (VOF). Neste exemplo, a FBU opcional *Impressora Virtual* (*Virtual Printer*) será adicionada à configuração do produto em tempo de execução apenas se a pré-condição representada no diagrama for verdadeira. No entanto, nessa abordagem, não foi identificado nenhum mecanismo de apoio à definição das informações utilizadas para a descrição das pré-condições.

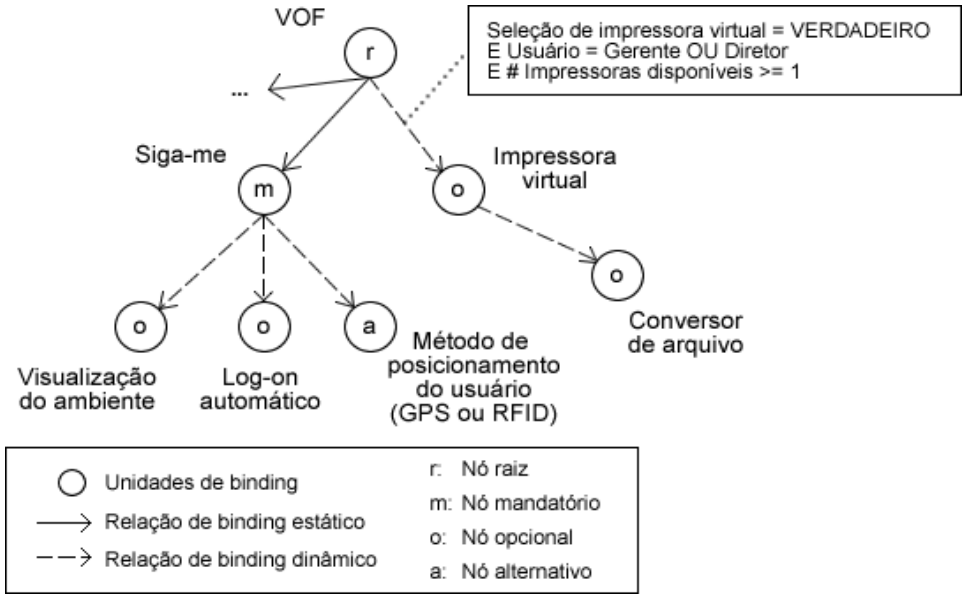



**Figura 3.8. Exemplo de um diagrama de *binding* [Adaptada de (LEE e MUTHIG, 2006)].**

### 3.5.2.3 Modelo de características estendido

BENAVIDES *et al.* (2005) apresentam extensões no modelo de características para introduzir os conceitos de atributos e características extra-funcionais. Atributos são propriedades de características que podem ser medidas. Por exemplo, *latência* e *largura de banda* podem ser atributos da característica *Conexão de Internet*, enquanto *disponibilidade* e *custo* podem ser atributos da característica *Serviços*. O domínio de um atributo representa o espaço de valores que ele pode assumir e pode ser discreto (e.g., inteiros, "booleanos") ou contínuo (e.g., reais). As características extra-funcionais representam a relação entre um ou mais atributos de uma característica. Por exemplo, *largura de banda = 256* ou *latência/disponibilidade >50*. Essas relações são associadas a uma característica. A Figura 3.9 ilustra um modelo de características estendido. Neste exemplo, os mecanismos de extensão foram utilizados com o propósito de representar o tempo e o preço para o desenvolvimento de cada uma das características.

Apesar de não ser o foco dos autores, os mecanismos de extensão apresentados poderiam ser utilizados para representação de contexto. As entidades de contexto seriam representadas por características comuns, as informações de contexto por atributos e as regras de contexto por características extra-funcionais. No entanto, essa representação não apresenta de forma explícita os conceitos relacionados à modelagem de contexto, uma vez que as extensões propostas podem ser utilizadas para diferentes propósitos. No exemplo da Figura 3.9, por exemplo, elas são utilizadas para representar propriedades relacionadas ao desenvolvimento das características. Além disso, os autores não

apresentam nenhum mecanismo para representar restrições entre características e nenhuma ferramenta para apoiar a abordagem proposta.

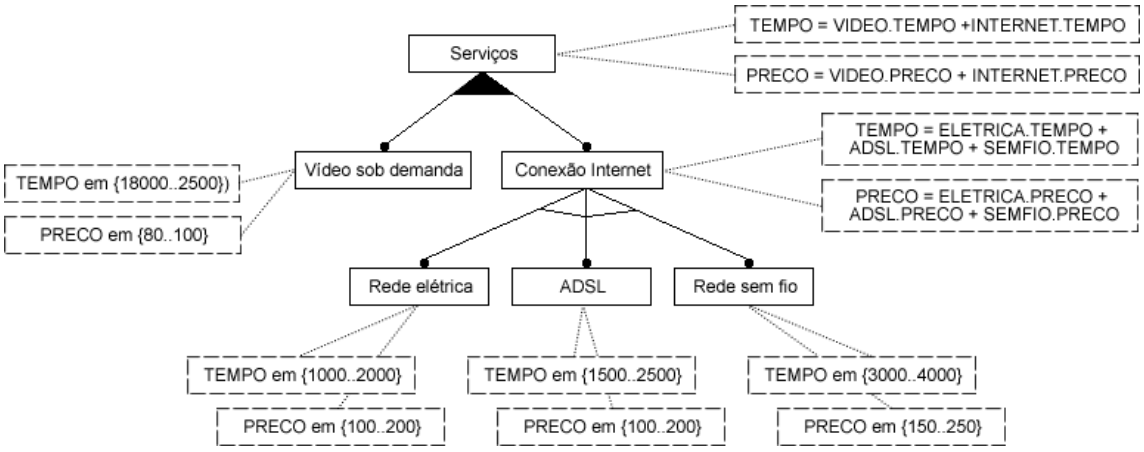



**Figura 3.9. Exemplo de um modelo de características estendido (BENAVIDES *et al.*, 2005).**

### 3.5.2.4 Uso de ontologias

WAGELAAR (2005) afirma que os modelos de características não estão completamente preparados para lidar com o impacto causado por fatores externos ou contexto. Mudanças nesses fatores externos podem causar diferentes interações (dependência ou exclusão) entre características. O autor defende a separação entre as interações internas entre características e as interações causadas por fatores externos, por meio de uma modelagem explícita desses fatores externos. Os fatores externos são descritos em modelos de contexto, enquanto que o modelo de características descreve apenas as dependências em relação a esses fatores. A abordagem proposta nesse trabalho faz uso de ontologias (GRUBER, 1993) para expressar tanto o modelo de contexto, quanto o modelo de características, sendo necessário o mapeamento deste último para ontologias. É também apresentado um mecanismo para determinar a validade da configuração de um produto para um determinado contexto. Porém, não são apresentados detalhes de como são representadas as interações entre os dois modelos.

Apesar dessa abordagem representar os conceitos para modelagem de contexto de forma explícita, ela propõe o uso de ontologias para a representação do modelo de características e contexto. No entanto, ontologia não é um conceito muito comum em LPS. Dessa forma, grande parte do ferramental de apoio ao desenvolvimento de LPS existente não poderia ser utilizado.

## 3.6 Considerações finais

Neste capítulo, foram apresentados os conceitos relacionados à LPS e discutidos os principais trabalhos relacionados.

O desenvolvimento de aplicações sensíveis ao contexto introduz novos desafios para o paradigma de LPS. A adaptação às mudanças contextuais corresponde ao processo de derivação de novos produtos em uma LPS (HALLSTEINSEN *et al.*, 2006). Dessa forma, os produtos devem ser configurados durante sua execução. Nesse novo ambiente, o foco muda da engenharia estática, em tempo de desenvolvimento, para a engenharia dinâmica, em tempo de execução (SUGUMARAN *et al.*, 2006).

O contexto é um elemento chave para aplicações sensíveis ao contexto. Assim, uma LPS para o desenvolvimento desse tipo de aplicação deve fornecer mecanismos para representar as informações de contexto relevantes para o domínio e, principalmente, o impacto dessas informações na reconfiguração dos produtos em tempo de execução. No entanto, como podemos observar nos trabalhos relacionados, grande parte das abordagens, mesmo as relacionadas à configuração dinâmica de produtos, não apresentam mecanismos que tornem explícitos os conceitos relacionados à modelagem de contexto utilizando modelos de características.

Como mencionado anteriormente, o modelo de características é um dos primeiros artefatos desenvolvidos em uma LPS, servindo para apoiar o entendimento da LPS como um todo e é utilizado como ponto de partida para a instanciação de novos produtos. Segundo LEE e MUTHIG (2006), é natural e intuitivo para as pessoas expressar os pontos comuns e variáveis de uma LPS por meio de características. Dessa forma, é de fundamental importância para o desenvolvimento de LPS para essa classe de aplicações, que o contexto também seja representado nesse nível de abstração.

No próximo capítulo é apresentada a abordagem para modelagem de características proposta nesta dissertação, denominada UbiFEX, que busca permitir a representação de contexto de forma explícita e a validação da configuração dos produtos com base em variações contextuais. Além disso, é discutida uma tabela comparativa entre os trabalhos relacionados e a abordagem proposta.

# Capítulo 4 – Abordagem UbiFEX

## 4.1 Introdução

Como apresentado no Capítulo 3, o modelo de características é utilizado como ponto de partida para o recorte necessário a instanciação de novos produtos e serve para apoiar a comunicação e padronizar o entendimento do domínio entre todos os envolvidos no processo de desenvolvimento de uma LPS. Segundo KANG *et al.* (2002), características são abstrações que tanto os clientes quanto os desenvolvedores são capazes de entender. Além disso, é natural e intuitivo expressar a variabilidade de linha de produtos em termos de características (LEE e MUTHIG, 2006).

No Capítulo 3, também foi visto que as abordagens existentes para o desenvolvimento de LPS não apresentam uma forma satisfatória de representar as informações de contexto relevantes para um determinado domínio em modelos de características. No entanto, essas informações são elementos chave no desenvolvimento de sistemas sensíveis ao contexto e, portanto, devem ser identificadas desde as etapas iniciais de desenvolvimento.

A análise dessas abordagens motivou a identificação dos principais requisitos para a abordagem proposta nesta dissertação, que incluem:

1. Identificação de elementos para a representação das entidades e informações de contexto em modelo de características de forma explícita;
2. Identificação de elementos para a representação da influência do contexto na configuração dos produtos;
3. Extensão de uma notação para modelagem de características existente para incluir os elementos identificados;
4. Definição de um mecanismo de verificação de consistência da configuração dos produtos com base em modelo de características em função de variações de contexto.

Como forma de atender aos requisitos listados, foi proposta a abordagem UbiFEX (FERNANDES e WERNER, 2008b; FERNANDES *et al.*, 2008), que possui como principal objetivo fornecer mecanismos para apoiar a modelagem de características de Linha de Produtos de Software Sensíveis ao Contexto (LPSSC). Para

atender a esse objetivo, a abordagem é dividida em dois componentes: UbiFEX-Notation e UbiFEX-Simulation.

UbiFEX-Notation é uma notação para modelagem de características que inclui de forma explícita elementos para modelagem das entidades e informações de contexto relevantes para um determinado domínio. Além disso, inclui elementos para representar a influência dessas informações na configuração dos produtos.

UbiFEX-Simulation é um mecanismo para a verificação de consistência da configuração dos produtos em relação às variações de contexto, com base em modelos de características construídos utilizando UbiFEX-Notation.

Este capítulo está organizado da seguinte forma: a Seção 4.2 descreve o domínio utilizado como exemplo para ilustrar os conceitos apresentados neste capítulo; a Seção 4.3 apresenta em detalhes a notação para modelagem de características UbiFEX-Notation; a Seção 4.4 descreve o mecanismo UbiFEX-Simulation; e a Seção 4.5 apresenta as considerações finais e discute algumas das contribuições e limitações da abordagem proposta.

## 4.2 Domínio Exemplo

O modelo de características utilizado para exemplificar a aplicação da abordagem proposta representa uma LPS no domínio de *middlewares* para aplicações distribuídas[2].

As aplicações distribuídas, em geral, são executadas em ambientes heterogêneos constituídos por computadores que podem apresentar arquiteturas e sistemas operacionais diferentes. Uma das soluções para amenizar os problemas dessa heterogeneidade é interpor entre aplicações e sistemas operacionais uma terceira camada de software, denominada *middleware*. Esta camada permite que os desenvolvedores possam dispor de uma interface de programação uniforme para o desenvolvimento de diferentes aplicações.

Inicialmente, no processo de engenharia de domínio, foi modelada uma LPS com base nas características do *framework* Arcademis (PEREIRA *et al.*, 2006), que é utilizado para apoiar o desenvolvimento de sistemas de *middleware* para aplicações

[2] A criação desse modelo foi realizada em conjunto com um especialista no domínio no desenvolvimento de aplicações sensíveis ao contexto do Grupo de Redes de Computadores, Engenharia de Software e Sistemas (GREat) da Universidade Federal do Ceará (UFC). Essa modelagem foi realizada no contexto do projeto MobiLine (Edital Universal - MCT/CNPq - Nº 15/2007).

distribuídas. No entanto, Arcademis não considera questões relativas à sensibilidade ao contexto, por isso, nessa fase também foram consideradas as características do produto AdaptiveRME (ROCHA, 2007). AdaptiveRME é um middleware adaptativo para ambientes móveis e ubíquos, que foi derivado a partir de RME (*Remote Method Invocation for J2ME*) (PEREIRA *et al.*, 2006), um sistema de *middleware* gerado a partir de Arcademis, e foi estendido para prover reconfiguração dinâmica e sensibilidade ao contexto. Nessa etapa, foram modeladas as características do domínio e identificadas e modeladas as características de contexto relevantes para AdaptiveRME. Além disso, foram especificadas as definições e regras de contexto identificadas a partir das reconfigurações previstas em AdaptiveRME, com base nas variações de contexto.

Em seguida, AdaptiveRME foi derivado por meio do processo de engenharia de aplicação a partir da LPS modelada. A Figura 4.1 ilustra parte do modelo de características de domínio de AdaptiveRME. O modelo completo está disponibilizado no Apêndice I.

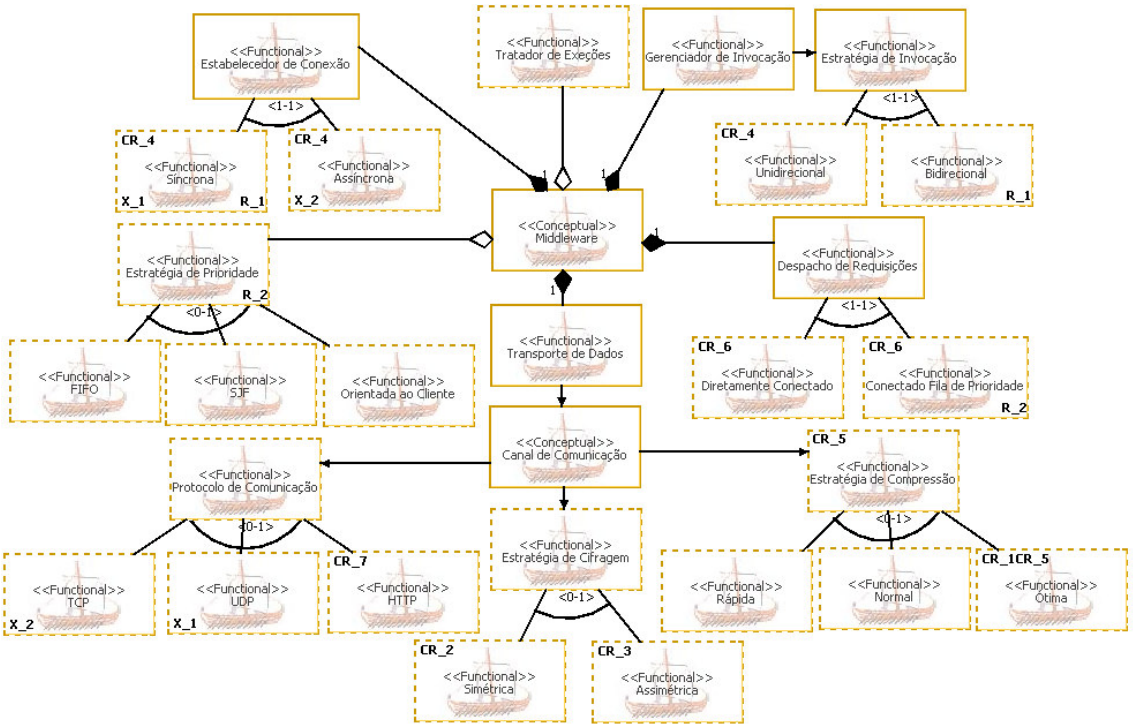



**Figura 4.1. Parte do modelo de características do produto AdaptiveRME.**

Esse modelo inclui características, como, por exemplo: *Estabelecedor de Conexão*, que representa componentes relacionados ao estabelecimento de conexões entre clientes e servidores. Essa característica é um ponto de variação mandatório, que pode ser configurado pelas variantes opcionais *Síncrona* ou *Assíncrona*, que indicam a forma como a conexão será estabelecida. A característica *Transporte de Dados* é uma

invariante, que representa componentes que definem os mecanismos utilizados para transmitir dados entre clientes e servidores.

Podemos observar no modelo da Figura 4.1 algumas marcações nas características. Essas marcações podem ser de três tipos: R_N, que indica que a característica marcada faz parte de uma regra de composição inclusiva; X_N, que indica que a característica faz parte de uma regra de composição exclusiva; e CR_N, que indica a participação em uma regra de contexto. Essas marcações facilitam na identificação das características que fazem parte de alguma dessas regras.

Algumas regras de composição foram definidas para o modelo, como, por exemplo:

- **R_2**: Conectado Fila de Prioridade *requires* Estratégia de Prioridade
- **X_1**: Síncrona *excludes* UDP

A regra de composição inclusiva R_2 indica que, se a característica *Conectado Fila de Prioridade* for selecionada, a característica *Estratégia de Prioridade* também deve estar selecionada. A regra de composição exclusiva X_1 indica que se a característica *Síncrona* for selecionada, a característica *UDP* não deve estar selecionada.

## 4.3 UbiFEX-Notation

Segundo POHL e METZGER (2006), a documentação de variabilidade deve responder pelo menos as seguintes questões:

- *O que varia?* Permite identificar os pontos de variação;
- *Como varia?* Permite identificar as variantes;
- *Por que varia?* Leva ao raciocínio associado aos pontos de variação e variantes; e
- *Para quem ele é documentado?* Indica o grupo alvo de um ponto de variação ou variante.

No entanto, essas perguntas não são suficiente para identificar as informações de contexto relevantes em um domínio, sendo necessário acrescentar a seguinte pergunta:

- *Que entidades e informações de contexto podem influenciar essa variação?* Auxilia na identificação das entidades e informações de contexto que influenciam a configuração dos produtos.

UbiFEX-Notation estende a notação Odyssey-FEX (OLIVEIRA, 2006) com o propósito de permitir a representação de contexto em modelos de características e, conseqüentemente, a modelagem de linhas de produtos sensíveis ao contexto.

Esta seção está organizada da seguinte forma: a Seção 4.3.1 apresenta uma análise da notação Odyssey-FEX; a Seção 4.3.2 detalha as extensões propostas para a criação da notação UbiFEX-Notation; e finalmente, a Seção 4.3.3 apresenta uma visão geral da notação proposta.

### 4.3.1 Análise da notação Odyssey-FEX

Como apresentado na seção anterior, a notação Odyssey-FEX permite a representação dos principais conceitos relacionados a modelos de características. No entanto, por não ser esse o objetivo dessa notação, não foram identificados elementos que tornassem possível a representação explícita dos conceitos necessários para a modelagem de LPSSC.

Inicialmente, foi analisado que tipos de características poderiam ser utilizados para a modelagem de entidades e informações de contexto com base nas categorias existentes. Para a representação das entidades de contexto, as características do tipo entidade na notação Odyssey-FEX são as que mais se aproximaram, pois, segundo a definição dessa categoria, representam entidades do mundo real que atuam sobre o domínio. Porém, elas também representam atores do modelo e, em níveis de abstração posteriores, são mapeadas como atores em diagramas de casos de uso. No entanto, nem todas as entidades de contexto podem ser vistas como atores, pois não interagem diretamente com o domínio.

No caso das informações de contexto, utilizadas para caracterizar as entidades de contexto, a representação mais próxima são as características do tipo ambiente operacional, pois são utilizadas para representar atributos de um ambiente que uma aplicação do domínio pode usar e operar. No entanto, essa categoria leva em consideração apenas atributos do ambiente de operação do produto, o que não possibilitaria, por exemplo, a representação das preferências do usuário. Além disso, não existe nenhuma propriedade para atribuir o tipo do atributo associado a essa característica.

Além dos problemas relatados, a utilização dessas categorias não tornaria explícita a representação das entidades e informações de contexto.

Foi ainda analisada como poderia ser feita a representação da influência desses elementos na configuração dos produtos. Nesse caso, as regras de composição são a representação mais próxima, porém elas incluem apenas relacionamentos entre características, o que não é suficiente para definir as situações, com base nos elementos contextuais, onde a inclusão ou exclusão de uma determinada característica deve ocorrer.

### 4.3.2 Extensões propostas

A partir da análise apresentada na seção anterior e do estudo das diferentes categorias de modelos existentes para modelagem de contexto, apresentadas no Capítulo 2, foi possível identificar as extensões necessárias para possibilitar a modelagem de LPSSC, dando origem a notação UbiFEX-Notation. Essas extensões incluem: a criação de novas categorias de características e novas expressões e regras, denominadas definições e regras de contexto.

#### 4.3.2.1 Novas categorias de características

Para a representação explícita das entidades e informações de contexto, duas novas categorias de características foram definidas, com o mesmo nome dos elementos que representam (Tabela 4.1).

**Tabela 4.1. Novas categorias de características.**

| | Ícone | Categoria |
|---|---|---|
| **Características de Contexto** |  | **Entidade de Contexto** – Características que representam as entidades de contexto relevantes para o domínio. Essas entidades podem ser representadas por lugares (e.g., sala), pessoas (e.g., usuário) ou objetos (e.g., dispositivo móvel). |
| | | **Informação de Contexto** – Características que representam as informações que devem ser coletadas para caracterizar as entidades de contexto do domínio (e.g., a localização do usuário, o sistema operacional do dispositivo móvel). |

Com base nas técnicas e conceitos apresentados no Capítulo 2, foram definidas algumas propriedades para cada um dos tipos de característica definidos.

As características do tipo entidade de contexto possuem como propriedades:

- *Nome*, que é utilizado para identificar a característica; e
- *Descrição*, que permite fornecer mais detalhes sobre a entidade identificada, adicionando uma maior semântica ao modelo.

Por sua vez, as características informação de contexto têm como propriedades:

- *Nome*, que é utilizado para identificar a característica;
- *Descrição*, que permite fornecer mais detalhes sobre a semântica da informação de contexto identificada;
- *Tipo base*, que indica o tipo base da informação que a característica representa (e.g., *string*, inteira, *booleana*);
- *Persistência*, que indica se a característica é estática ou dinâmica, dependendo da freqüência de atualização do seu valor;
- *Composição*, que indica se a informação é simples ou composta por outras informações de contexto; e
- *Aquisição*, que indica se a informação é obtida por perfil, definida pelo usuário, "sentida" ou derivada.

Algumas dessas propriedades, dependendo do domínio, podem ser específicas dos produtos, como, por exemplo, a propriedade de aquisição. Dessa forma, considerando o processo de desenvolvimento de LPS, parte dessas propriedades pode ser definida durante a etapa de engenharia de domínio, e parte durante a engenharia de aplicação.

As características de contexto podem se relacionar utilizando os relacionamentos da UML presentes na notação Odyssey-FEX. Em especial, as entidades de contexto se relacionam com as informações de contexto que as caracterizam por meio de associações nomeadas, quando necessário, para explicitar o tipo específico de relacionamento existente entre elas.

Com base na classificação das características a respeito das dimensões, apresentadas na notação Odyssey-FEX, as características de contexto também podem ser classificadas quanto à opcionalidade como mandatórias ou opcionais. No entanto, nenhuma classificação foi definida em relação à variabilidade para essas características, pois se considerarmos a variabilidade como as diferentes formas que uma informação de contexto pode ser apresentada, o modelo poderia ficar sobrecarregado. Por exemplo, a localização de um usuário pode ser dada por coordenadas GPS ou pelo nome do lugar no qual ele se encontra.

A Figura 4.2 representa parte do modelo de características de contexto de AdaptiveRME, com as entidades e informações de contexto relevantes para a execução desse produto. Nesse modelo, a entidade de contexto *Dispositivo Móvel* é descrita pelas

informações de contexto *Nível da Bateria, Memória Livre, Uso da CPU e Nível do sinal de Conexão.*

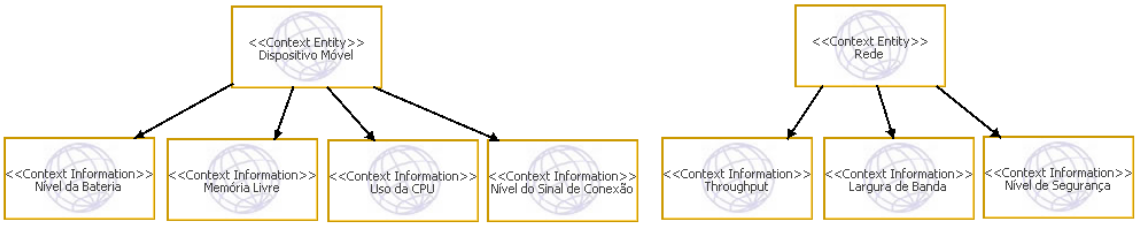



**Figura 4.2. Parte do modelo de características de contexto do produto AdaptiveRME.**

Em relação às propriedades, temos, por exemplo, que a característica *Memória Livre*, para o produto AdaptiveRME, é classificada quanto ao tipo base como inteira, quanto à persistência como dinâmica, quanto à composição como simples e quanto à aquisição como "sentida". As propriedades não são representadas graficamente, pois poderiam sobrecarregar o modelo e dificultar a sua visualização.

Porém, apenas a criação dessas novas categorias de características não é suficiente para representar o impacto dessas informações em relação à variabilidade dos produtos e às decisões de adaptação. Para permitir essa representação, novos elementos foram definidos, os quais foram denominados: definições de contexto e regras de contexto.

#### 4.3.2.2 Definições de contexto

As definições de contexto descrevem situações relevantes para o domínio, tendo como base as entidades e informações de contexto modeladas. Essas situações indicam quando ações relacionadas à configuração dos produtos de uma LPS devem ocorrer. As definições de contexto têm como propriedades: um nome, utilizado como identificador; e uma expressão, que representa a condição para que um contexto definido ocorra.

A notação BNF (*Backus-Naur Form*) que descreve a expressão que representa uma definição de contexto é apresentada na Figura 4.3. Essa expressão pode ser formada da seguinte maneira:

- Uma Característica do tipo Informação de Contexto (CIC) previamente modelada, um operador relacional e um valor; ou
- A negação de uma expressão previamente definida; ou
- Uma combinação, por meio de operadores lógicos, de expressões já definidas.

```
<exp-def-contexto> ::= <CIC><operador-relacional><valor>
            | NÃO <exp-def-contexto>
            | <exp-def-contexto><operador-lógico><exp-def-contexto>
<operador-relacional> ::=  > | < | >= | <= | = | <>
<operador-lógico> ::=  E | OU
<valor> ::= <string> | <inteiro> | <float> | <booleano>
```

**Figura 4.3. BNF para a expressão de uma definição de contexto.**

Este último caso permite que uma nova definição de contexto seja definida a partir de outras previamente definidas, promovendo o reúso e facilitando a atividade de manutenção. A idéia de composição de situações também é apresentada por YE *et al.* (2008), que afirmam que esse mecanismo ajuda a manter a consistência entre as situações definidas.

Os operadores relacionais definidos são: *maior que* (>), *menor que* (<), *maior ou igual que* (>=), *menor ou igual que* (<=), *igual* (=) e *diferente* (<>). Os operadores lógicos podem ser do tipo: OU (*OR*) ou E (*AND*). Os valores podem ser do tipo: *string*, inteiro, *float* ou *booleano*.

Um contexto está ativo para um determinado cenário de execução do produto, quando a avaliação da expressão que o descreve retorna um valor verdadeiro. O termo cenário, neste caso, se refere ao conjunto de valores das informações de contexto em um determinado instante. Para o produto AdaptiveRME, temos como exemplos as definições de contexto apresentadas na Figura 4.4. Neste caso, a definição de contexto Alta Carga de Processamento está ativa para um determinado cenário, se o valor da informação de contexto Uso da CPU da entidade de contexto Dispositivo Móvel for maior que oitenta.

**Alta Carga de Processamento**

(Dispositivo Móvel.Uso da CPU > 80)

**Throughput Baixo**

((Rede.Throughput < 64) AND (Rede.Largura de Banda >= 1024))

**Figura 4.4. Exemplo de definições de contexto.**

As unidades de medida e o domínio de valores possíveis para uma informação de contexto não possuem uma propriedade particular para descrevê-los, porém eles podem ser detalhados na propriedade de descrição da característica. Dessa forma, para a especificação das definições de contexto, os domínios e unidades de medida utilizados

devem considerar essa descrição. No exemplo da Figura 4.4, o *Uso da CPU* é dado em porcentagem, o *Throughput* e a *Largura de Banda* em Kbps.

Após a definição dos contextos relevantes para o domínio, as ações que devem ser tomadas para uma determinada situação podem ser especificadas por meio das regras de contexto.

#### 4.3.2.3 Regras de contexto

As regras de contexto representam como um contexto, previamente especificado, impacta na configuração dos produtos de uma LPS em tempo de execução, indicando, por exemplo, decisões a respeito da seleção de variantes em um ponto de variação. Essas regras têm como propriedades: um identificador e uma expressão.

A notação BNF (*Backus-Naur Form*) que descreve a expressão que representa uma regra de contexto é apresentada na Figura 4.3. Essa expressão é formada por um antecedente, o operador "implica" (*implies*) e um conseqüente. O antecedente pode ser formado por uma:

- Definição de contexto; ou
- Combinação de definições de contexto e operadores lógicos; ou
- Combinação, por meio de operadores lógicos, de definições de contexto e características.

Os operadores lógicos utilizados no antecedente podem ser do tipo E (*AND*), OU (*OR*), OU-Exclusivo (*XOR*) ou NÃO (*NOT*).

O conseqüente, por sua vez, é formado por uma característica ou uma combinação de características. Os operadores lógicos utilizados no conseqüente podem ser do tipo E (*AND*) ou NÃO (*NOT*).

```
<exp-regra-contexto> ::= <antecedente>implies<conseqüente>
<antecedente> ::= <def-contexto>
            |<def-contexto><operador-lógico-ant><característica>
            |<antecedente ><operador-lógico-ant><antecedente>
<operador-lógico-ant> ::= E | OU | OU-Exclusivo | NÃO
<conseqüente> ::= <característica>
            |<conseqüente><operador-lógico-con>< conseqüente>
<operador-lógico-con> ::= E | NÃO
```

**Figura 4.5. BNF para a expressão de uma regra de contexto.**

A presença de uma característica no conseqüente indica a sua adição. No entanto, a presença do operador lógico NÃO (*NOT*) indica a remoção da característica ao qual ele é aplicado.

Para uma característica poder fazer parte do conseqüente de uma regra de contexto ela deve atender à seguinte classificação: não ser uma característica de contexto, pois deve ser adicionada ou removida da configuração do produto; ser uma característica opcional, pois todas as mandatórias já devem estar presentes na configuração do produto; e ser uma invariante, ponto de variação ou variante.

Para que características classificadas como ponto de variação pudessem ser adicionadas no conseqüente de uma regra de contexto, foi introduzido na notação Odyssey-FEX o conceito de variante padrão (GOMAA, 2004). Esse conceito está presente em outras notações de modelagem de características e indica qual opção deve ser escolhida caso nenhuma outra variante seja selecionada para um determinado ponto de variação.

O operador "implica" (*implies*) determina que se a expressão que define o antecedente for avaliada como verdadeira, então, as características presentes no conseqüente devem ser ou não selecionadas para a nova configuração do produto em função dos operadores lógicos envolvidos.

As regras de contexto podem ser utilizadas tanto para representar mudanças essenciais na configuração do produto quanto para prover funcionalidades melhor adaptadas, com base nos contextos que estão ativos em um determinado instante no ambiente de execução do produto. Dessa forma, essas regras podem ser divididas em duas categorias: obrigatórias e opcionais.

As regras obrigatórias indicam que para uma determinada condição, representada pelo seu antecedente, se as ações indicadas pelo conseqüente não ocorrem, possivelmente o produto não irá funcionar de forma adequada. Já as regras opcionais representam apenas ações de melhoria no funcionamento do produto em um determinado cenário, por exemplo, para prover serviços mais relevantes, e, portanto, a não realização das mesmas não implicaria em problemas na execução do produto.

A Figura 4.6 apresenta exemplos de regras de contexto do produto AdaptiveRME. A regra de contexto CR_5 indica que, para um determinado cenário de execução, quando a definição de contexto *Throughput Baixo* for verdadeira, o ponto de variação *Estratégia de Compressão* e a sua variante *Ótima* devem ser adicionados à

configuração atual do produto. Para este exemplo, todas as regras de contexto foram consideradas como obrigatórias.

**CR_5**

Throughput Baixo *implies* (Estratégia de Compressão AND Ótima)

**CR_6**

Alta Carga de Processamento *implies* (Conectado Fila de Prioridade AND (NOT Diretamente Conectado))

**Figura 4.6. Exemplos de regras de contexto.**

Na representação gráfica do modelo de características (ver Figura 4.1), as características que fazem parte do conseqüente de uma regra de contexto são marcadas no canto superior esquerdo com o identificador da regra, permitindo de forma clara a identificação das características que sofrem influência de contexto.

### 4.3.3 Visão geral

Com a inclusão das extensões apresentadas nas seções anteriores, o modelo de características de uma LPSSC engloba (Figura 4.7):

- *Modelo de características do domínio*, que inclui as características conceituais, funcionais e tecnológicas do domínio, além de representar a variabilidade da LPSSC;
- *Regras de composição*, que representam as restrições de dependência e mútua exclusividade entre as características do domínio;
- *Modelo de características de contexto*, que inclui as características entidades de contexto e informações de contexto, representando os elementos contextuais relevantes para o domínio;
- *Definições de contexto*, que representam situações onde possivelmente decisões de adaptação devem ser tomadas; e
- *Regras de contexto*, que representam ações a serem realizadas na configuração dos produtos em função de variações de contexto.

Como podemos observar, as regras de contexto são responsáveis por fazer a ligação entre os dois tipos de modelos de características em uma LPSSC, representando a influência de um modelo no outro.

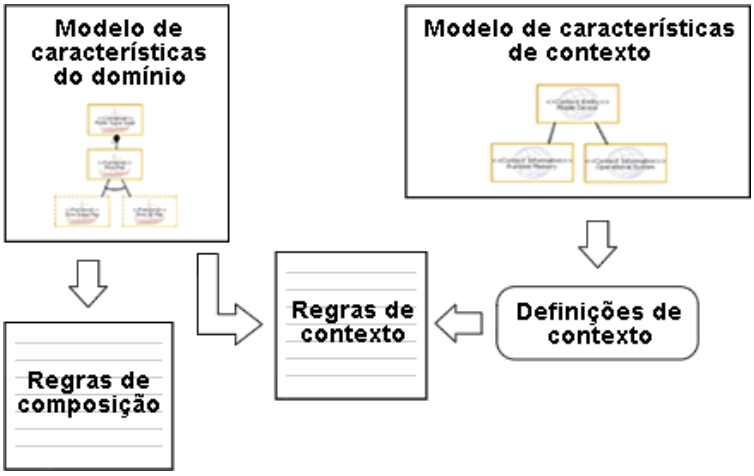



**Figura 4.7. Visão geral do modelo de características de uma LPSSC.**

A abordagem proposta sugere dois modelos distintos para os diferentes tipos de características como forma de explicitar ainda mais os conceitos relacionados aos elementos de contexto. No entanto, essa decisão deve ser tomada pelo engenheiro de domínio responsável pela modelagem de características da LPSSC, dependendo, por exemplo, da complexidade da LPSSC.

## 4.4 UbiFEX-Simulation

Como apresentado no Capítulo 3, após a definição do modelo de características na etapa de engenharia de domínio, os produtos são derivados pelo processo de engenharia de aplicação, que utiliza como ponto inicial o modelo de características da LPS. Nas abordagens tradicionais de LPS, todos os pontos de variação e características que fazem parte de um produto são definidos nessa etapa, ainda em tempo de desenvolvimento. No entanto, para LPSSC, algumas dessas decisões devem ser realizadas apenas em tempo de execução. Um recorte inicial do modelo de características pode ser realizado durante a engenharia de domínio e novos recortes também podem ocorrer em tempo de execução, como forma do produto se adaptar às variações de contexto.

Nas abordagens que tratam a configuração dos produtos apenas de forma estática, a consistência dessa configuração, em relação às restrições definidas no modelo de características, é verificada no processo de derivação do produto. Porém, quando consideramos os produtos sensíveis ao contexto, novas configurações podem ser geradas em função das variações de contexto em tempo de execução.

Como forma de verificar a consistência dessas novas configurações com base nos modelos de características desenvolvidos utilizando UbiFEX-Notation, a

abordagem UbiFEX define um mecanismo de verificação, denominado UbiFEX-Simulation.

UbiFEX-Simulation possui como objetivo verificar, em tempo de desenvolvimento, a consistência entre regras de contexto e, principalmente, a consistência dessas regras em relação às restrições de configuração dos produtos.

As principais restrições definidas em modelos de características e, normalmente, verificadas no processo de engenharia de aplicação, estão relacionadas às:

- Regras de composição, tanto inclusivas quanto exclusivas;
- Cardinalidade de pontos de variação; e
- Opcionalidade das características.

Dessa forma, cada nova configuração gerada do produto, em função das variações de contexto e aplicação das regras de contexto, deve satisfazer a esse conjunto de restrições.

Em relação à verificação de consistência das regras de contexto definidas, dois tipos de inconsistência podem ser identificados:

- *Intra-regra*, que ocorre quando uma mesma regra é responsável por incluir e excluir uma mesma característica. Esse tipo de inconsistência independe dos valores das informações de contexto, uma vez que, sempre que essa regra for executada, ações contraditórias serão realizadas; e
- *Inter-regra*, que ocorre quando, para uma mesma característica, uma regra implica na sua inclusão e outra regra na sua remoção. Se ambas as regras forem obrigatórias, uma delas terá a sua ação desfeita, podendo ocasionar uma falha na execução do produto. Esse tipo de inconsistência, por sua vez, depende dos valores das informações de contexto em um determinado momento, pois a inconsistência só ocorre se as duas regras forem executadas para um mesmo cenário.

A verificação de consistência em relação às restrições definidas no modelo também depende dos valores das informações de contexto, pois uma nova configuração do produto é gerada em função das regras de contexto executadas.

Dessa forma, UbiFEX-Simulation define um processo para auxiliar essas verificações para um determinado cenário. No entanto, é importante destacar que para garantir uma verificação abrangente da consistência das configurações do produto, esse

processo deve ser executado para cada um dos cenários que levam a situações de reconfiguração.

Como pré-requisitos para a execução do processo proposto por UbiFEX-Simulation, a configuração inicial do produto, que compreende as características que farão parte do produto na sua inicialização, deve ser selecionada. Além disso, a configuração inicial deve estar consistente em relação às regras de composição, cardinalidade e opcionalidade definidas pelo modelo de características do produto. Para exemplificar, podemos considerar a seleção das seguintes características do produto AdaptiveRME (ver Figura 4.1): *Middleware*, *Canal de Comunicação*, *Tratador de Exceções*, *Protocolo de Comunicação*, *UDP*, *Transporte de Dados*, *Gerenciador de Invocação*, *Estratégia de Invocação*, *Unidirecional*, *Despacho de Requisições*, *Diretamente Conectado*, *Estabelecedor de Conexão* e *Assíncrona*.

O processo proposto por UbiFEX-Simulation é ilustrado na Figura 4.8 e suas atividades são descritas nos tópicos a seguir.

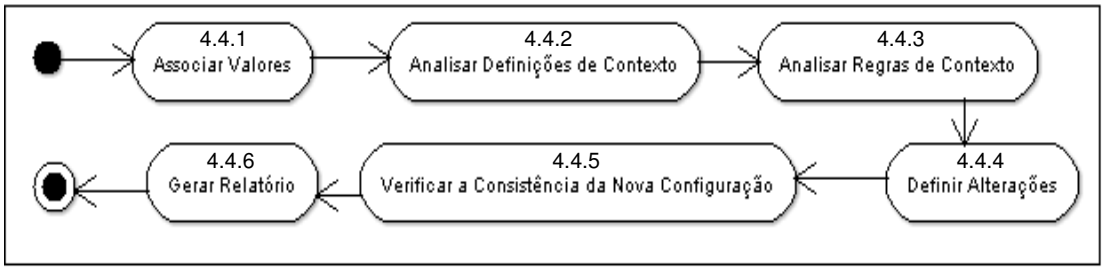



**Figura 4.8. Processo de verificação proposto por UbiFEX-Simul.**

### 4.4.1 Associar valores

A primeira atividade do processo consiste na associação dos valores para cada característica do tipo informação de contexto de acordo com o cenário para o qual se deseja realizar as verificações.

**Tabela 4.2. Valores associados às informações de contexto.**

| CENÁRIO 1 | | |
|---|---|---|
| **Entidade de Contexto** | **Informação de Contexto** | **Valor** |
| Rede | *Throughput* | 56Kbps |
| | Nível de Segurança | 4 |
| | Largura de Banda | 512Kbps |
| Dispositivo Móvel | Nível de Bateria | 75% |
| | Memória Livre | 256k |
| | Nível do Sinal de Conexão | 5 |
| | Uso da CPU | 85% |

Esses valores podem ser associados, por exemplo, como mostrado na Tabela 4.2. Neste caso, a informação de contexto *Uso da CPU* da entidade de contexto *Dispositivo Móvel* possui valor de 85%.

### 4.4.2 Analisar definições de contexto

Em seguida, as definições de contexto especificadas no modelo de características de contexto devem ser analisadas com base nos valores definidos na atividade anterior. Dessa forma, as expressões que representam cada uma das definições de contexto são avaliadas, com o objetivo de identificar as definições de contexto que estão ativas, ou seja, que tiveram a expressão avaliada como verdadeira para o cenário especificado. Para este exemplo de cenário, vamos considerar as definições apresentadas na Figura 4.4: *Alta Carga de Processamento* e *Throughput Baixo*. Avaliando cada uma das expressões, é possível identificar que apenas *Alta Carga de Processamento* está ativa para os valores apresentados na Tabela 4.2.

### 4.4.3 Analisar regras de contexto

A próxima atividade consiste em analisar as regras de contexto especificadas para o produto. Essa atividade tem o objetivo de verificar quais regras serão executadas com base nas definições de contexto ativas e nas características selecionadas na configuração inicial. Para que uma regra seja executada, a expressão que representa o seu antecedente deve ser verdadeira. Neste exemplo, vamos considerar as regras de contexto apresentadas na Figura 4.6: CR_5 e CR_6. Dessa forma, analisando o antecedente de cada uma das regras, temos que apenas CR_6 será executada, pois possui como antecedente a definição de contexto ativa *Alta Carga de Processamento*.

### 4.4.4 Definir alterações

O próximo passo é definir as alterações que devem ser realizadas na configuração inicial do produto para gerar uma nova configuração, adaptada às informações de contexto. Para isso, dois conjuntos devem ser definidos: *características adicionadas* e *características removidas*. Esses conjuntos são povoados de acordo com as ações especificadas por cada uma das regras de contexto que são executadas, definidas na atividade anterior. Além disso, a indicação de qual regra foi responsável pela inclusão ou remoção de uma determinada característica também deve ser especificada. No exemplo apresentado, temos os seguintes conjuntos: Características

adicionadas = {*Conectado Fila Prioridade* (*CR_6*)} e Características removidas = {*Diretamente Conectado* (*CR_6*)}.

### 4.4.5 Verificar a consistência da nova configuração

Depois que os conjuntos de características adicionadas e removidas são definidos, é possível realizar a verificação da consistência das regras de contexto e da nova configuração, em relação às restrições do modelo. Essa atividade é dividida em quatro etapas:

1. Verificar se os conjuntos são vazios;
2. Identificar inconsistências inter e intra-regra;
3. Identificar inconsistências relacionadas ao conjunto de características adicionadas; e
4. Identificar inconsistências relacionadas ao conjunto de características removidas.

A etapa 1 consiste em verificar se ambos os conjuntos são vazios, pois se isso ocorrer, a nova configuração gerada é igual à configuração inicial. Esse fato indica que o cenário verificado não possui influência na configuração do produto. Dessa forma, nenhuma inconsistência é identificada e o processo de verificação é finalizado.

A etapa 2 tem como objetivo identificar possíveis inconsistências apenas entre as regras de contexto. Como mencionado anteriormente, elas podem ser classificadas em intra-regra e inter-regra. Para identificar esse tipo de inconsistência, é necessário verificar se alguma característica está presente tanto no conjunto de características adicionadas quanto no conjunto de características removidas. Se isso ocorrer, dois casos devem ser verificados: se a característica foi incluída em ambos os conjuntos por uma mesma regra de contexto, uma inconsistência intra-regra é identificada; caso contrário, uma inconsistência inter-regra é identificada.

Quando inconsistências inter ou intra-regras são identificadas, recomenda-se que o processo de verificação seja finalizado, pois os conjuntos gerados incluem modificações obtidas a partir de regras inconsistentes. Porém, no caso de inconsistências inter-regras, também pode ser considerado o fato das regras de contexto poderem ser classificadas como obrigatórias ou opcionais. Se a inconsistência tiver sido gerada entre regras opcionais ou entre uma regra obrigatória e outra opcional, dependendo do funcionamento do mecanismo de reconfiguração dos produtos, essa inconsistência pode ser descartada. Por exemplo, sempre que ocorrer uma inconsistência

desse tipo entre uma regra obrigatória e uma opcional, apenas a primeira é executada, ou, no caso de duas regras opcionais, alguma política de prioridade pode ser aplicada entre elas. No entanto, a definição dessa política está fora do escopo deste trabalho.

Para o exemplo apresentado nesta seção, nas etapas 1 e 2, os conjuntos não são vazios e nenhuma característica faz parte dos dois conjuntos, portanto, nenhuma inconsistência foi identificada e o processo não foi finalizado.

A etapa 3 consiste na identificação de inconsistências relacionadas ao conjunto de características adicionadas. Nesse caso, para cada elemento desse conjunto, as seguintes verificações devem ser realizadas, em relação às regras de composição:

- Se a característica adicionada faz parte do antecedente de uma regra de composição inclusiva, avaliado como verdadeiro, é necessário verificar se o conseqüente está sendo satisfeito; e
- Se a característica adicionada faz parte do conseqüente ou do antecedente de uma regra de composição exclusiva, avaliados como verdadeiros, é necessário verificar, respectivamente, se o antecedente ou conseqüente da regra está sendo satisfeito.

Em relação às restrições de cardinalidade, a seguinte verificação deve ser realizada:

- Se a característica adicionada é uma variante, é necessário verificar se a cardinalidade máxima do ponto de variação ao qual ela pertence está sendo respeitada com a sua inclusão.

Na etapa 3, temos que a característica adicionada *Conectado Fila de Prioridade* faz parte do antecedente de uma regra de composição (R_2) que não está sendo respeitada, pois o conseqüente dessa regra, formado pela característica *Estratégia de Prioridade*, não está selecionado na configuração inicial do produto e nem pertence ao conjunto de características adicionadas. Quanto à cardinalidade, *Conectado Fila de Prioridade* é uma variante, mas a cardinalidade máxima de 1 está sendo respeita, uma vez que a variante *Diretamente Conectado*, selecionada na configuração inicial, pertence ao conjunto de características removidas.

A etapa 4 consiste na identificação de inconsistências relacionadas ao conjunto de características removidas. Nesse caso, para cada elemento desse conjunto, as seguintes verificações devem ser realizadas, em relação às regras de composição:

- Se a característica removida faz parte do conseqüente de uma regra de composição inclusiva, avaliado como verdadeiro, é necessário verificar se o conseqüente está sendo satisfeito.

Em relação às restrições de cardinalidade, a seguinte verificação deve ser realizada:

- Se a característica removida é uma variante, é necessário verificar se a cardinalidade mínima do ponto de variação ao qual ela pertence está sendo respeitada com a sua remoção.

Na etapa 4, temos que, a característica removida *Diretamente Conectado* não faz parte de nenhuma regra de composição. Quanto à cardinalidade, o valor mínimo 1 está sendo respeitado, pois apesar da remoção dessa característica, a variante *Conectado Fila de Prioridade* pertence ao conjunto de características adicionadas.

Nas etapas 3 e 4, caso uma regra de composição não esteja sendo respeitada, uma inconsistência entre regras de composição e regras de contexto é identificada. No caso de regras de composição inclusiva, isso ocorre quando apenas o antecedente ou o conseqüente é avaliado como verdadeiro. Para as regras de composição exclusivas, isso ocorre quando tanto o conseqüente quanto o antecedente são verdadeiros. Da mesma forma, caso uma cardinalidade de um ponto de variação não esteja sendo respeitada, uma inconsistência entre cardinalidade e regra de contexto é identificada.

É importante destacar que, nas etapas 3 e 4, as verificações realizadas devem sempre levar em consideração o conjunto global de alterações, incluindo todas as características adicionadas e removidas, além das características selecionadas na configuração inicial.

### 4.4.6 Gerar relatório

A última atividade do processo proposto por UbiFEX-Simulation consiste na geração de um relatório com as inconsistências identificadas.

No exemplo apresentado, apenas uma inconsistência foi encontrada, envolvendo a regra de composição R_2 e a regra de contexto CR_6. Como CR_6 é uma regra obrigatória, ela seria executada e a característica *Conectado Fila de Prioridade* adicionada. Porém, o funcionamento do produto estaria comprometido, uma vez que as dependências dessa característica não estão sendo satisfeitas. Uma forma de corrigir essa inconsistência seria, durante a modelagem, adicionar a característica *Estratégia de Prioridade* ou uma de suas variantes no conseqüente da regra CR_6.

## 4.5 Considerações finais

Neste capítulo, foi apresentada a abordagem UbiFEX, que visa apoiar a modelagem de características de LPSSC. UbiFEX é dividida em dois componentes, UbiFEX-Notation e UbiFEX-Simulation, que aplicados em conjunto permitem a representação de contexto e do impacto deste na configuração dos produtos de forma explícita. Além disso, UbiFEX-Simulation possibilita a verificação da configuração dos produtos com base em diferentes cenários de execução. Essa verificação é importante, pois tem como objetivo antecipar possíveis falhas que só seriam detectadas durante a execução dos produtos.

A Tabela 4.3 apresenta um resumo, considerando alguns dos trabalhos relacionados apresentados no Capítulo 3, que tratam mais diretamente de modelagem de características, e a abordagem proposta, em relação aos requisitos apresentados no início deste capítulo. Esses requisitos incluem: a representação explícita das entidades de contexto, informações de contexto e da influência do contexto na configuração dos produtos; e a verificação da consistência das configurações dos produtos.

**Tabela 4.3. Resumo dos trabalhos relacionados em relação aos requisitos da abordagem proposta.**

| | **HARTMANN e TREW (2008)** | **LEE e MUTHIG (2006)** | **BENAVIDES *et al.* (2005)** | **WAGELAAR (2005)** | **UbiFEX** |
|---|---|---|---|---|---|
| **Entidades de contexto** | | | Parcial | X | X |
| **Informações de contexto** | X | Parcial | Parcial | X | X |
| **Influência do contexto na configuração dos produtos** | X | X | | | X |
| **Verificação da consistência das configurações** | | | | X | X |

Todas as abordagens apresentam alguma forma de representação das informações de contexto, mesmo que não seja de forma explícita, sendo classificadas na tabela como parcial. As entidades de contexto e a influência do contexto na configuração dos produtos são representadas apenas em parte das abordagens. Além da abordagem UbiFEX, apenas WAGELAAR (2005) inclui um mecanismo para verificação da consistência das configurações do produto. No entanto, essa abordagem

não apresenta detalhes dos tipos de inconsistências identificadas, além de ser baseada na transformação do modelo de características em ontologias.

Algumas limitações identificadas na abordagem UbiFEX, assim como em outras abordagens, em relação às definições de contexto se referem ao fato delas não conseguirem representar todo o domínio de situações possíveis em sistemas sensíveis ao contexto. No entanto, foram consideradas as situações mais comumente observadas. Além disso, UbiFEX permite que novos tipos de valores ou novos operadores sejam adicionados na definição dessas expressões, por exemplo, para tratar informações e operadores baseados em conjuntos.

A abordagem proposta é aplicada apenas em modelos de características, mas poderia ser aplicada também em níveis de abstração posteriores, permitindo, por exemplo, um refinamento das regras de contexto, com elementos de menor granularidade do que as características.

A aplicação de UbiFEX-Simulation tem como objetivo antecipar a identificação, ainda em tempo de desenvolvimento, de possíveis falhas que ocorreriam durante a execução do produto, simulando as ações de reconfiguração para um determinado cenário de um produto derivado a partir de uma LPSSC. Dessa forma, é possível corrigir em as inconsistências identificadas antes da execução do produto.

As aplicações sensíveis ao contexto precisam se adaptar a diferentes e inúmeros cenários, o que torna difícil garantir a verificação de todos os cenários possíveis. Além disso, existe a possibilidade de que novas características sejam adicionadas e novas regras introduzidas apenas durante a execução dos produtos. Apesar disso, com a aplicação dessa abordagem, é possível garantir que o produto funcionará da forma adequada para os cenários verificados, além disso, permite que erros e inconsistências de modelagem não sejam propagados para níveis de abstração posteriores. No entanto, a solução para a eliminação dessas inconsistências é de responsabilidade do engenheiro de domínio.

No próximo capítulo é apresentado um estudo executado para avaliar a aplicação do processo proposto por UbiFEX-Simulation.

# Capítulo 5 – Avaliação do Mecanismo UbiFEX-Simulation

## 5.1 Introdução

O mecanismo UbiFEX-Simulation, apresentado no Capítulo 4, possui como objetivo auxiliar a verificação de consistência da configuração dos produtos com base em modelo de características, em função de variações de contexto. Um estudo preliminar, descrito neste capítulo, foi realizado para avaliar a aplicação desse mecanismo.

Este capítulo está organizado da seguinte forma: a Seção 5.2 apresenta o objetivo do estudo; a Seção 5.3 define o estudo realizado; as Seções 5.4 e 5.5 descrevem, respectivamente, a primeira e a segunda etapa do estudo; a Seção 5.6 discute a validade do estudo; e finalmente, a Seção 5.7 apresenta as considerações finais.

## 5.2 Meta

O estudo realizado para avaliar a aplicação do mecanismo UbiFEX-Simulation pode ser classificado como um estudo de observação, no qual o participante realiza alguma tarefa enquanto é observado por um experimentador. Esse tipo de estudo tem como finalidade coletar dados sobre como uma determinada tarefa é realizada (SHULL *et al.*, 2001). Dessa forma, pode-se obter uma melhor compreensão de como um novo processo é utilizado. A meta do estudo de observação realizado pode ser descrita de acordo com a Tabela 5.1, seguindo o modelo proposto por WOHLIN *et al.* (2000).

## 5.3 Definição

No total, participaram do estudo seis alunos de pós-graduação da linha de Engenharia de Software da COPPE/UFRJ, divididos em dois grupos. A seleção dos participantes foi realizada de forma aleatória por meio de um convite enviado por e-mail. O participante tinha total liberdade para decidir sobre a sua colaboração no estudo. O estudo foi dividido em duas etapas, denominadas Etapa 1 e Etapa 2. Cada etapa foi realizada por grupos distintos, com três participantes cada.

**Tabela 5.1 Descrição da meta do estudo de observação.**

| |
|---|
| **Analisar** o processo definido em UbiFEX-Simulation |
| **Com a finalidade de** caracterizar sua aplicação |
| **Em relação à** verificação de consistência da configuração dos produtos com base em modelos de características, considerando variações nos contextos de execução dos produtos |
| **Do ponto de vista do** engenheiro de domínio |
| **No contexto de** alunos de pós-graduação, analisando a configuração de um produto sensível ao contexto em diferentes cenários de execução, com e sem o uso do processo definido em UbiFEX-Simulation |

Independente do grupo ao qual pertencesse, o participante deveria, inicialmente, preencher: o Formulário de Consentimento (Apêndice II), que descreve alguns tópicos relacionados à participação no estudo e, ao ser assinado, confirma a concordância do participante com o mesmo; e o Formulário de Caracterização do Participante (Apêndice III), que avalia o nível de conhecimento do participante em diferentes tópicos relacionados ao estudo, como, por exemplo, linha de produtos de software, sistemas sensíveis ao contexto e modelagem de características. Essas informações foram coletadas como o intuito de auxiliar a interpretação dos resultados obtidos.

Na situação proposta aos participantes, eles deveriam exercer o papel de um engenheiro de domínio contratado por uma empresa de desenvolvimento de software. Essa empresa possuía uma linha de produtos para o desenvolvimento de sistemas de *middlewares* para aplicações distribuídas. Originalmente, essa linha de produtos não era sensível ao contexto, mas por exigências do mercado, foi necessário incluir tal característica. Para isso, foi feita uma nova modelagem de características dessa linha de produtos utilizando a notação UbiFEX-Notation. Em seguida, foi derivado o produto AdaptiveRME, por meio do processo de engenharia de aplicação. Dessa forma, a tarefa do participante era verificar a consistência da configuração desse novo produto em diferentes cenários de execução e identificar possíveis inconsistências em relação ao modelo de características do produto.

Os participantes da Etapa 1 deveriam realizar essa verificação de forma *ad-hoc*, seguindo uma estratégia própria. Já os participantes da Etapa 2 deveriam realizar essa verificação utilizando o processo presente no mecanismo UbiFEX-Simulation. O objetivo dessa divisão foi comparar os resultados obtidos com e sem a aplicação desse processo.

Uma descrição mais detalhada do produto AdaptiveRME e da tarefa a ser realizada pode ser encontrada nos Apêndices IV e V, utilizados, respectivamente, para o Grupo 1, que inclui os participantes que realizaram a Etapa 1, e para o Grupo 2, que inclui os participantes que realizaram a Etapa 2. Como forma de minimizar o esforço dos participantes, apenas parte do modelo geral de características de AdaptiveRME foi fornecido para análise (Apêndice VI), incluindo parte do modelo de características de domínio, das regras de composição, do modelo de características de contexto, das definições e das regras de contexto.

Como mencionado anteriormente, os participantes deveriam verificar a configuração de AdaptiveRME em diferentes cenários de execução. Foram definidos seis cenários com base nas características de contexto presentes nesse produto, que incluíam: a entidade de contexto *Rede*, caracterizada pelas informações de contexto *Throughput*, *Nível de Segurança* e *Largura de Banda*; e a entidade de contexto *Dispositivo Móvel*, caracterizada pelas informações de contexto *Nível de Bateria*, *Memória Livre*, *Nível do Sinal de Conexão* e *Uso da CPU*. A Tabela 5.2 ilustra um dos cenários de execução que foi apresentado aos participantes.

**Tabela 5.2. Exemplo de um cenário de execução.**

| **CENÁRIO X** | | |
|---|---|---|
| **Entidade de Contexto** | **Informação de Contexto** | **Valor** |
| Rede | *Throughput* | 32 Kbps |
| | Nível de Segurança | 7 |
| | Largura de Banda | 256 Kbps |
| Dispositivo Móvel | Nível de Bateria | 90% |
| | Memória Livre | 512 KB |
| | Nível do Sinal de Conexão | 5 |
| | Uso da CPU | 60% |

A descrição completa de todos os cenários utilizados no estudo pode ser encontrada no Apêndice VII. O modelo geral de características de AdaptiveRME entregue aos participantes sofreu algumas alterações em relação ao modelo original, apresentado no Capítulo 4, com o intuito de introduzir as inconsistências que deveriam ser identificadas pelos participantes.

A configuração inicial do produto deveria ser considerada sempre a mesma para todos os cenários analisados e as características que faziam parte desta configuração foram destacadas na representação gráfica do modelo entregue aos participantes. Além

disso, o participante era informado de que a configuração inicial não apresentava nenhuma inconsistência.

Cada cenário foi apresentado de forma individual aos participantes, pois não era desejado que a análise de um cenário tivesse influência na análise de outro. Isso poderia ocorrer, pois os valores de algumas informações de contexto eram comuns para diferentes cenários.

A definição dos cenários levou em consideração os diferentes tipos de inconsistências descritos no Capítulo 4. Dessa forma, o resultado esperado para cada cenário é descrito na Tabela 5.3. Nessa tabela temos: o número do cenário; se existe ou não alguma inconsistência; caso exista inconsistência, qual a descrição da mesma; e os elementos envolvidos nessa inconsistência, representados pelos identificadores das regras descritos no modelo exemplo (Apêndice VI) ou pelo nome das características envolvidas. Outros elementos envolvidos poderiam ser considerados, como por exemplo, as características envolvidas nas inconsistências entre regras de contexto ou composição, o que não invalida a resposta dos participantes.

**Tabela 5.3. Resultados esperados para cada cenário.**

| Cenário | Inconsistência? | Descrição das Inconsistências | Elementos Envolvidos |
|---|---|---|---|
| **1** | Não | - | **-** |
| **2** | Sim | Inconsistência inter-regra de contexto | RX1 e RX5 |
| **3** | Sim | Inconsistência entre regra de composição inclusiva e regra de contexto | RC2 e RX6 |
| **4** | Não | - | - |
| **5** | Sim | Inconsistência entre regra de composição exclusiva e regra de contexto | RC3 e RX4 |
| **6** | Sim | Inconsistência entre cardinalidade e regra de contexto | Característica *Protocolo de Comunicação* e RX7 |

No Cenário 1, nenhuma regra de contexto era executada, logo a configuração inicial do produto não era modificada, mantendo a sua consistência. Esse cenário foi definido apenas para o participante ir se familiarizando com os termos e documentos utilizados no estudo.

No Cenário 2, duas regras de contexto são executadas. Uma regra adiciona e a outra remove uma mesma característica (*Ótima*), resultando em uma inconsistência inter-regra de contexto.

O Cenário 3 apresenta um caso onde ocorre uma inconsistência entre uma regra de composição inclusiva e uma regra de contexto. Nesse cenário, uma característica adicionada à configuração do produto (*Conectado Fila de Prioridade*) requer outra característica (*Estratégia de Prioridade*) que não está presente na nova configuração.

No Cenário 4, apesar de algumas regras de contexto serem executadas, nenhuma inconsistência deve ser identificada. Esse cenário foi definido apenas para não deixar o participante ter a impressão errada de que sempre que alguma regra de contexto é executada, ocorre uma inconsistência.

O Cenário 5 é semelhante ao Cenário 3, mas neste caso ocorre uma inconsistência entre uma regra de composição exclusiva e uma regra de contexto. Isso ocorre pois uma característica adicionada à configuração do produto (*Síncrona*) exclui uma das características que está presente nessa nova configuração (*UDP*).

Finalmente, o Cenário 6 apresenta um caso de inconsistência entre cardinalidade e regra de contexto. Nesse caso, uma característica adicionada à configuração do produto (*HTTP*) é variante de um ponto de variação (*Protocolo de Comunicação*) que possui cardinalidade máxima igual a um e já possui outra variante selecionada, desrespeitando assim o número máximo de variantes que podem ser selecionadas para esse ponto de variação.

Após a avaliação de cada um dos cenários, esse era o resultado geral esperado dos participantes. No entanto, variações na descrição das inconsistências e elementos envolvidos eram possíveis, uma vez que os participantes poderiam descrever com as próprias palavras as inconsistências identificadas. Dessa forma, foi feito um mapeamento dessas descrições com as apresentadas na Tabela 5.3.

Além dos dados coletados durante a realização da tarefa proposta, foi entregue, ao final, um questionário de avaliação, com o objetivo de confirmar os dados observados e coletar a opinião dos participantes sobre outros aspectos investigados neste estudo.

As principais questões investigadas neste estudo são:

1. A aplicação do processo definido em UbiFEX-Simulation possui impacto no número de inconsistências identificadas?

2. A aplicação do processo definido em UbiFEX-Simulation antecipa a identificação de inconsistências que só seriam percebidas em tempo de execução?
3. A aplicação do processo definido em UbiFEX-Simulation é viável se realizada de forma manual?

## 5.4 Estudo de observação – Etapa 1

Esta seção apresenta a primeira etapa do estudo realizado. A Seção 5.4.1 descreve o estudo e os principais dados a serem coletados. A Seção 5.4.2 apresenta o procedimento adotado para a realização dessa etapa. Os resultados obtidos são apresentados na Seção 5.4.3. Por fim, uma análise geral do estudo é realizada na Seção 5.4.4.

### 5.4.1 Descrição

Nesta etapa do estudo, os participantes deveriam analisar a configuração dos produtos nos seis cenários propostos sem a utilização do processo definido em UbiFEX-Simulation. Assim, eles deveriam encontrar uma estratégia própria para realizar essa verificação.

O principal objetivo dessa etapa foi observar a estratégia utilizada pelos participantes e o total de inconsistências encontradas. Os principais dados coletados nessa etapa envolveram: a descrição da estratégia utilizada pelo participante; as inconsistências identificadas em cada cenário; e o tempo para a realização da tarefa.

### 5.4.2 Procedimento

Nessa etapa do estudo, participaram três alunos de pós-graduação. Cada sessão foi realizada com o experimentador e um dos participantes, totalizando três sessões.

Como descrito na Seção 5.3, inicialmente, o participante deveria preencher o Formulário de Consentimento (Apêndice II) e o Formulário de Caracterização do Participante (Apêndice III).

Em seguida, com o intuito de uniformizar o conhecimento e treinar todos os participantes em relação aos tópicos abordados no estudo, foi realizada uma apresentação de aproximadamente 15 minutos, incluindo os seguintes tópicos: sistemas sensíveis ao contexto, linha de produtos de software, modelagem de características, a notação UbiFEX-Notation e o modelo exemplo do estudo.

Após a apresentação, foi reservado um tempo para o participante tirar possíveis dúvidas sobre o conteúdo da apresentação com o experimentador. Depois disso, foram fornecidos os documentos: Descrição da Tarefa – Grupo 1 (Apêndice IV) e Descrição do Modelo Exemplo (Apêndice VI). Após a leitura desses documentos, foi entregue ao participante o primeiro cenário de execução do produto a ser analisado.

Em posse desses documentos, para cada novo cenário entregue ao participante, ele deveria realizar a tarefa de verificação proposta e preencher o Formulário de Identificação de Inconsistências (Apêndice IX).

Concluída a análise dos seis cenários, o participante deveria responder às questões do Formulário de Avaliação – Grupo 1 (Apêndice XI), como forma de auxiliar a coleta de dados do estudo.

O tempo total para a realização de cada sessão desta etapa do estudo foi, em média, de 1 hora e 8 minutos.

### 5.4.3 Resultados

A Tabela 5.4 apresenta um resumo da caracterização dos participantes da Etapa 1. Nos itens da questão 2, os níveis de experiência variam de 0 (nenhuma) a 4 (usei em projetos na indústria), e, no item da questão 3, variam de 0 (nenhuma familiaridade) a 2 (muito familiar).

**Tabela 5.4. Resumo da caracterização dos participantes da Etapa 1 do estudo.**

| **Pergunta** | | | **P1** | **P2** | **P3** |
|---|---|---|---|---|---|
| **1** | | Formação acadêmica | Mestrando | Mestrando | Mestrando |
| **2** | **2.1** | Sistemas sensíveis ao contexto (0-4) | 1 | 1 | 1 |
| | **2.2** | Modelagem de contexto (0-4) | 0 | 0 | 0 |
| | **2.3** | Linhas de produtos de software (0-4) | 2 | 1 | 2 |
| | **2.4** | Modelagem de características (0-4) | 2 | 0 | 1 |
| **3** | | Experiência no domínio de *middlewares* (0-2) | 0 | 0 | 1 |

Durante o estudo, foram medidos os tempos de cada participante para a realização da tarefa proposta. Os números apresentados na Tabela 5.5 levam em consideração apenas o intervalo entre a entrega do primeiro cenário de execução para análise até o término do preenchimento do Formulário de Identificação de Inconsistências para o último cenário apresentado.

**Tabela 5.5. Tempo de cada participante da Etapa 1 do estudo.**

| Participante | Tempo da tarefa (em minutos) |
|---|---|
| P1 | 50 |
| P2 | 33 |
| P3 | 25 |

Nas seções seguintes, são detalhados os resultados obtidos em cada uma das três sessões realizadas.

#### 5.4.3.1 Participante P1

As respostas do participante P1 apresentadas no Formulário de Identificação de Inconsistências estão resumidas na Tabela 5.6. O participante identificou as inconsistências dos Cenários 3 e 6. No entanto, as inconsistências presentes nos Cenários 2 e 5 não foram identificadas.

**Tabela 5.6. Respostas de P1 apresentadas no Formulário de Identificação de Inconsistências.**

| Cenário | Inconsistência? |
|---|---|
| **1** | Não |
| **2** | Não |
| **3** | Sim |
| **4** | Não |
| **5** | Não |
| **6** | Sim |

Segundo os dados coletados a partir da observação da realização da tarefa e das respostas fornecidas pelo participante no Formulário de Avaliação, a estratégia utilizada para a identificação das inconsistências foi a seguinte: inicialmente, o participante P1 analisou os valores associados às informações de contexto para o cenário entregue e verificou quais definições de contexto estavam ativas. Em seguida, o participante analisou cada uma das regras de contexto, verificando de forma individual as ações tomadas para cada regra executada. Assim, para cada conjunto de ações determinado por uma regra de contexto, o participante verificava as regras de composição e a cardinalidade, caso a característica adicionada fosse uma variante.

O participante afirmou que sentiu dificuldades na realização da tarefa, por ela ser muito trabalhosa. Além disso, ele considera que a utilização de um processo como guia tornaria a tarefa mais fácil.

Segundo o participante, as inconsistências não seriam identificadas antes da execução do produto sem uma atividade de verificação, pois, geralmente, os modelos de características são bastante complexos. Ele também respondeu que não seria viável realizar manualmente essa tarefa de verificação para sistemas mais complexos, justificando que "além de ser uma atividade demorada, o índice de erros seria alto também".

#### 5.4.3.2 Participante P2

As respostas do participante P2 estão resumidas na Tabela 5.7. O participante identificou as inconsistências dos Cenários 2, 5 e 6. No entanto, a inconsistência presente no cenário 3 não foi identificada.

**Tabela 5.7. Respostas de P2 apresentadas no Formulário de Identificação de Inconsistências.**

| Cenário | Inconsistência? |
|---|---|
| **1** | Não |
| **2** | Sim |
| **3** | Não |
| **4** | Não |
| **5** | Sim |
| **6** | Sim |

Segundo os dados coletados, a estratégia utilizada para a identificação das inconsistências pode ser definida da seguinte forma: para cada um dos valores das informações de contexto em um determinado cenário, o participante P2 analisou as definições de contexto e as regras de contexto que eram impactadas diretamente, verificando as regras executadas e o conjunto de ações a serem realizadas. Em seguida, o participante analisou a representação gráfica do modelo de características e as regras de composição com o objetivo de identificar as possíveis inconsistências.

O participante afirmou que não sentiu dificuldades na realização da tarefa. No entanto, considera que a utilização de um processo como guia seria útil, pois ajudaria na verificação das inconsistências sem considerar a subjetividade dos engenheiros de domínio.

Segundo o participante, as inconsistências não seriam completamente identificadas antes da execução do produto sem uma atividade de verificação. Além disso, ele considera que tarefas desse tipo são importantes para agregar qualidade aos produtos, sobretudo quando se trata de produtos derivados de linha de produtos. O

participante respondeu que não seria viável realizar manualmente essa tarefa de verificação para sistemas mais complexos e fez a seguinte afirmação: “creio que esta tarefa deve, no mínimo, ser semi-automatizada, utilizando uma ferramenta que contemple um processo, pois o nível de detalhes (regras e informações) aumenta a sua complexidade”.

#### 5.4.3.3 Participante P3

De acordo com o Formulário de Identificação de Inconsistências, as respostas do participante estão resumidas na Tabela 5.8. O participante identificou as inconsistências dos Cenários 3, 5 e 6. No entanto, a inconsistência presente no Cenário 2 não foi identificada.

**Tabela 5.8. Respostas de P3 apresentadas no Formulário de Identificação de Inconsistências.**

| Cenário | Inconsistência? |
|---|---|
| **1** | Não |
| **2** | Não |
| **3** | Sim |
| **4** | Não |
| **5** | Sim |
| **6** | Sim |

Segundo os dados coletados, a estratégia utilizada pelo participante P3 para a identificação das inconsistências foi a seguinte: para cada cenário entregue, o participante percorreu cada uma das regras de contexto, verificando qual seria executada, analisando as definições de contexto presentes no antecedente dessas regras e tendo como base os valores associados a cada uma das informações de contexto. Para cada uma das ações executadas por cada uma das regras, o participante verificava, na representação gráfica do modelo de características, se a característica que estava sendo adicionada ou removida possuía alguma marcação que indicasse a participação da mesma em uma regra de composição. Se isso ocorresse, a regra era verificada. Se a característica fosse uma variante, a cardinalidade do ponto de variação também era verificada.

O participante afirmou que não sentiu dificuldades na realização da tarefa. No entanto, considera que a utilização de um processo como guia seria útil, pois por não se tratar de uma tarefa comum, possíveis entendimentos alternativos, poderiam levar a execução da tarefa de forma errada.

Para o participante P3, as inconsistências não seriam completamente identificadas antes da execução do produto sem uma atividade de verificação, pois seria necessário considerar os diferentes cenários de execução. O participante também respondeu que não seria viável realizar de forma manual essa tarefa para sistemas mais complexos e afirmou que "à medida que cresce a quantidade de regras de contexto aplicadas simultaneamente, o esforço para identificar as inconsistências e verificar todas as restrições aplicáveis seria grande".

Outro ponto destacado pelo participante foi que as marcações relacionadas às regras de contexto na representação gráfica do modelo de características não tiveram nenhuma influência na execução da atividade.

### 5.4.4 Análise geral

É possível perceber pela Tabela 5.4, que os participantes não possuíam muita experiência nos tópicos envolvidos no estudo. No entanto, a apresentação realizada no início de cada sessão foi suficiente para a execução da tarefa proposta.

Nenhum dos participantes conseguiu encontrar todas as inconsistências. É importante destacar que nenhuma falsa inconsistência foi identificada. No caso dos participantes P1 e P3 a inconsistência presente no Cenário 2 não foi identificada. No entanto, esse cenário pode ser considerado um caso especial, pois não foi explicitado para o participante que todas as regras de contexto eram obrigatórias. Além disso, tanto o participante P1 quanto P3 aplicaram as ações definidas pelas regras de contexto de forma individual, o que não ocorreu com P2. Isso possibilitou que P2 percebesse que duas regras apresentavam conflitos no modelo e destacasse esse fato como uma inconsistência.

No estudo, não foi definido nenhum cenário que explorasse a questão da necessidade de se analisar o conjunto global de alterações, mas se isso tivesse sido feito, possivelmente, P1 e P3 encontrariam falsas inconsistências, pois na estratégia utilizada por eles, as ações realizadas por cada uma das regras eram analisadas individualmente.

A estratégia mais próxima do processo definido em UbiFEX-Simulation foi apresentada pelo participante P2. Porém, o participante não identificou a inconsistência presente no Cenário 3, possivelmente, por ter esquecido de verificar as regras de composição para este caso.

Em relação aos tempos, apresentados na Tabela 5.5, podemos perceber que P1 demorou mais tempo para a realização da tarefa, o que pode ser explicado pelo fato de

ter sido o único participante dessa etapa do estudo que afirmou ter sentido dificuldades na realização da tarefa. O participante P3 foi o mais rápido, o que se deve, principalmente, ao fato desse participante ter explorado bastante a representação gráfica do modelo de características.

O participante P3 fez uma observação sobre o fato das marcações, relacionadas às regras de contexto, presentes na representação gráfica das características não terem influenciado na realização da tarefa. Nesse caso, para a aplicação manual do processo, talvez fosse interessante realizar marcações também relacionadas às definições de contexto envolvidas nas regras. Dessa forma, após analisar as definições de contexto ativas, o participante já poderia definir que características poderiam ser influenciadas por essas definições e qual a regra associada. Porém, essas marcações adicionais poderiam poluir e dificultar a visualização do modelo.

No geral, todos os participantes responderam que um processo seria útil e facilitaria a realização da tarefa e que as inconsistências não seriam completamente identificadas antes da execução do produto sem uma atividade de verificação. Além disso, podemos destacar que todos os participantes responderam que não seria viável realizar essa tarefa para sistemas mais complexos de forma manual.

## 5.5 Estudo de observação – Etapa 2

Esta seção apresenta a segunda etapa do estudo realizado. A Seção 5.5.1 descreve o estudo. Na Seção 5.5.2, é apresentado o procedimento adotado para a realização dessa etapa. Os resultados obtidos são apresentados na Seção 5.4.3. Finalmente, uma análise geral do estudo é realizada na Seção 5.5.4.

### 5.5.1 Descrição

Essa etapa do estudo é semelhante à Etapa 1. No entanto, os participantes deveriam analisar a configuração dos produtos nos seis cenários seguindo o processo definido em UbiFEX-Simulation.

O principal objetivo dessa etapa foi observar a aplicação do processo pelos participantes e o total de inconsistências encontradas. Os principais dados coletados nessa etapa envolveram: as dificuldades na aplicação do processo; sugestões de melhoria para o processo; as inconsistências identificadas em cada cenário; e o tempo para a realização da tarefa.

### 5.5.2 Procedimento

Participaram dessa etapa do estudo três alunos de pós-graduação. Cada sessão foi realizada com o experimentador e um dos participantes, totalizando três sessões.

Como descrito na Seção 5.3, inicialmente, o participante deveria preencher o Formulário de Consentimento (Apêndice II) e o Formulário de Caracterização do Participante (Apêndice III).

Em seguida, semelhante ao procedimento da Etapa 1, foi realizada uma apresentação para o participante de aproximadamente 20 minutos, abordando os tópicos necessários para a realização do estudo. Neste caso, além dos tópicos mencionados na Seção 5.4.2, também foi apresentado ao participante o processo definido em UbiFEX-Simulation para verificação das configurações dos produtos.

Após a apresentação, foi destinado um tempo para o participante tirar possíveis dúvidas sobre o conteúdo da apresentação com o experimentador. Depois disso, foram fornecidos os documentos: Descrição da Tarefa – Grupo 2 (Apêndice V), Descrição do Modelo Exemplo (Apêndice VI) e Descrição do Processo de Verificação (Apêndice VIII). Além desses documentos, também foi entregue ao participante o Formulário de Apoio à Aplicação do Processo (Apêndice X), que tinha como objetivo auxiliar nas atividades de execução do processo para cada cenário. Após a leitura desses documentos, foi entregue ao participante o primeiro cenário de execução do produto a ser analisado.

Em posse desses documentos, para cada novo cenário entregue ao participante, ele deveria realizar a tarefa de verificação proposta e preencher o Formulário de Identificação de Inconsistências (Apêndice IX).

Concluída a análise dos seis cenários, o participante deveria responder às questões do Formulário de Avaliação – Grupo 2 (Apêndice XII), como forma de auxiliar a coleta de dados do estudo.

O tempo total para a realização de cada sessão desta etapa foi, em média, de 1 hora e 25 minutos.

### 5.5.3 Resultados

A Tabela 5.9 apresenta um resumo da caracterização dos participantes da Etapa 2. Nos itens da questão 2, os níveis de experiência variam de 0 (nenhuma) a 4 (usei em projetos na indústria), e, no item da questão 3, variam de 0 (nenhuma familiaridade) a 2 (muito familiar).

**Tabela 5.9. Resumo da caracterização dos participantes da Etapa 2 do estudo.**

| Pergunta | | | P4 | P5 | P6 |
|---|---|---|---|---|---|
| **1** | | Formação acadêmica | Mestrando | Mestrando | Mestrando |
| **2** | **2.1** | Sistemas sensíveis ao contexto (0-4) | 1 | 1 | 0 |
| | **2.2** | Modelagem de contexto (0-4) | 1 | 1 | 0 |
| | **2.3** | Linhas de produtos de software (0-4) | 2 | 2 | 1 |
| | **2.4** | Modelagem de características (0-4) | 2 | 1 | 0 |
| **3** | | Experiência no domínio de *middlewares* (0-2) | 1 | 0 | 0 |

Os tempos de cada participante para a realização da tarefa na Etapa 2 são apresentados na Tabela 5.10. Da mesma forma que na etapa anterior, os números levam em consideração apenas o intervalo entre a entrega para o participante do primeiro cenário de execução até o término do preenchimento do Formulário de Identificação de Inconsistências para o último cenário apresentado.

**Tabela 5.10. Tempo de cada participante da Etapa 2 do estudo para realização da tarefa proposta.**

| Participante | Tempo da tarefa (em minutos) |
|---|---|
| P4 | 47 |
| P5 | 40 |
| P6 | 55 |

Nas seções seguintes, são detalhados os resultados obtidos em cada uma das três sessões realizadas.

#### 5.5.3.1 Participante P4

A partir dos dados coletados do Formulário de Identificação de Inconsistências, as respostas do participante P4 estão resumidas na Tabela 5.11. O participante identificou as inconsistências dos Cenários 2, 3, 5 e 6, ou seja, todas as inconsistências esperadas.

De acordo com o que foi observado durante a execução da tarefa, o participante seguiu todas as atividades definidas no processo. Esse fato foi reforçado pela análise do Formulário de Apoio à Aplicação do Processo, que foi preenchido de forma correta para todos os cenários.

**Tabela 5.11. Respostas de P4 apresentadas no Formulário de Identificação de Inconsistências.**

| Cenário | Inconsistência? |
|---|---|
| **1** | Não |
| **2** | Sim |
| **3** | Sim |
| **4** | Não |
| **5** | Sim |
| **6** | Sim |

O participante afirmou que não sentiu dificuldades na realização da tarefa. Além disso, considerou que o processo foi útil para a realização da tarefa e destacou que o processo estava bastante claro. Como sugestão de melhoria para o processo, o participante afirmou que o detalhamento em um fluxograma da atividade de verificação de consistência da nova configuração facilitaria o seu entendimento.

Segundo o participante, as inconsistências não seriam identificadas antes da execução do produto sem uma atividade de verificação, pois mesmo um especialista poderia não identificar alguma inconsistência durante a modelagem. Ele também respondeu que não seria viável realizar de forma manual a verificação para sistemas mais complexos e afirmou que essa tarefa se torna extremamente desgastante, suscetível a erros e que tomaria um tempo que poderia ser utilizado em outra tarefa.

#### 5.5.3.2 Participante P5

De acordo com os dados coletados, as respostas do participante P5 estão resumidas na Tabela 5.12. O participante identificou todas as inconsistências esperadas.

Da mesma forma que foi observado na execução da tarefa por P4, o participante P5 seguiu todas as atividades definidas no processo. Esse fato também foi reforçado pela análise do Formulário de Apoio à Aplicação do Processo, que foi preenchido de forma correta para todos os cenários.

**Tabela 5.12. Respostas de P5 apresentadas no Formulário de Identificação de Inconsistências.**

| Cenário | Inconsistência? |
|---|---|
| **1** | Não |
| **2** | Sim |
| **3** | Sim |
| **4** | Não |
| **5** | Sim |
| **6** | Sim |

O participante afirmou que não sentiu dificuldades na realização da tarefa e considerou que o processo foi útil, destacando que cada uma das etapas do processo estava bem definida.

Como sugestão de melhoria para o processo, o participante indicou que também seria necessário verificar os casos onde poderiam ocorrer inconsistências relacionadas à cardinalidade na remoção de características. Na descrição do processo utilizada no estudo, apenas os casos de adição de novas características variantes era verificado. Essa sugestão foi implementada e está presente no processo apresentado no Capítulo 4.

Segundo o participante, as inconsistências não seriam identificadas antes da execução do produto sem uma atividade de verificação, pois algumas regras apresentam uma lógica muito complexa. Ele também respondeu que não seria viável realizar de forma manual essa tarefa de verificação para sistemas mais complexos e afirmou que "sistemas muito complexos eventualmente acarretariam em definições de contexto e regras logicamente muito complexas e passíveis de erro".

#### 5.5.3.3 Participante P6

As respostas do participante P6 estão resumidas na Tabela 5.13. O participante identificou as inconsistências dos Cenários 2, 3 e 5. No entanto, a inconsistência presente no Cenário 6 não foi identificada.

O participante P6, mesmo utilizando o processo definido em UbiFEX-Simulation, não conseguiu identificar todas as inconsistências. De acordo com o que foi observado durante a execução da tarefa, especialmente para o Cenário 6, o participante não seguiu todas as atividades definidas no processo, provavelmente por cansaço, uma vez que esse foi o último cenário a ser analisado. Esse fato foi comprovado pela análise do Formulário de Apoio à Aplicação do Processo, que não foi totalmente preenchido e apresentou algumas informações incorretas quanto às características adicionadas. No entanto, para os outros cenários, todos os dados foram preenchidos de forma correta.

**Tabela 5.13. Respostas de P6 apresentadas no Formulário de Identificação de Inconsistências.**

| Cenário | Inconsistência? |
|---|---|
| **1** | Não |
| **2** | Sim |
| **3** | Sim |
| **4** | Não |
| **5** | Sim |
| **6** | Não |

Quanto à dificuldade na realização da tarefa, o participante afirmou que sentiu dificuldades no início, por não conhecer o processo definido em UbiFEX-Simulation. Além disso, considerou que o processo foi útil para a realização da tarefa e destacou que o processo permitiu a identificação de inconsistências sem a necessidade de se ter muito conhecimento sobre o assunto.

O participante não apresentou nenhuma sugestão de melhoria para o processo. Segundo o participante, as inconsistências não seriam identificadas antes da execução do produto sem uma atividade de verificação, pois essa não era uma tarefa tão simples e, sem um processo, seria muito difícil fazer essa identificação. Ele também respondeu que não seria viável realizar essa tarefa de forma manual para sistemas mais complexos e afirmou que "apesar da aplicação do processo ir se tornando mais fácil à medida que vai sendo repetido, leva algum tempo para ser executado, além de ser trabalhoso".

### 5.5.4 Análise geral

Pela Tabela 5.9, é possível perceber que os participantes desta etapa também não possuíam muita experiência nos tópicos envolvidos no estudo. No entanto, a apresentação realizada no início de cada sessão foi suficiente para a execução da tarefa proposta.

Em relação aos tempos, apresentados na Tabela 5.10, a variação máxima entre dois participantes foi de 15 minutos. O participante P6 demorou mais tempo para a realização da tarefa, o que pode ser explicado pelo fato de ter sido o único participante dessa etapa que afirmou ter sentido dificuldades na aplicação do processo. Além disso, P3 é o participante menos experiente nos tópicos envolvidos no estudo.

Nessa etapa do estudo, nenhuma falsa inconsistência foi identificada. Além disso, um dos participantes não conseguiu encontrar todas as inconsistências esperadas. Esse fato ocorreu, pois o participante não seguiu todas as atividades definidas no processo para o Cenário 6, o que o levou a uma definição incorreta das características adicionadas e a não identificação de inconsistências para esse caso. Como esse foi o último cenário analisado, alguns fatores podem ter influenciado esse resultado, como o cansaço ou o tempo.

No geral, todos os participantes responderam que o processo auxiliou na realização da tarefa. Além disso, responderam que as inconsistências não seriam completamente identificadas antes da execução do produto sem uma atividade de

verificação e que não seria viável realizar de forma manual essa tarefa para sistemas mais complexos.

## 5.6 Validade

Este estudo preliminar de avaliação foi executado com o objetivo de caracterizar a aplicação do processo definido em UbiFEX-Simulation. A análise dos resultados obtidos foram úteis para ressaltar alguns pontos de melhoria do processo proposto, assim como os benefícios da sua aplicação. No entanto, devido às restrições deste estudo, estes resultados não podem ser generalizados.

Os participantes do estudo foram selecionados de forma aleatória, dentro de um grupo de alunos que fazem parte do mesmo grupo de pesquisa do experimentador, devido às restrições de tempo e pessoal disponível. Além disso, apenas três participantes realizaram cada etapa do estudo. Dessa forma, os resultados podem ter sido influenciados pelos tamanhos dos grupos e pela relação entre os participantes e o experimentador.

O uso de alunos de pós-graduação, não tão familiarizados com os temas envolvidos no estudo e não tão experientes quanto profissionais da indústria, também restringe a generalização dos resultados. Outra limitação é o fato de que a análise dos cenários foi realizada logo após as apresentações dos conceitos que seriam abordados no estudo. Assim, os resultados também podem ter sido influenciados pela falta de tempo de maturação do conhecimento necessário para a realização da tarefa proposta.

No estudo, foi utilizado um modelo de características relativamente pequeno. Isso foi necessário devido a limitações de tempo para cada sessão do estudo. Dessa forma, é possível que os resultados não sejam aplicáveis a produtos de maior porte. Porém, o processo de verificação definido em UbiFEX-Simulation talvez seja ainda mais útil para a identificação de inconsistências em projetos maiores.

## 5.7 Considerações Finais

Neste capítulo, foi apresentado o estudo de avaliação do processo definido em UbiFEX-Simulation, que possui como objetivo auxiliar a verificação de consistência da configuração dos produtos em função de variações no contexto de execução.

Os resultados obtidos nesse estudo apresentam alguns indícios para as respostas de cada uma das questões apresentadas na Seção 5.3.

Comparando os resultados obtidos nas duas etapas do estudo, em relação à identificação de inconsistências (Tabela 5.14), podemos perceber que o número de inconsistências encontradas pelo grupo de participantes que executou a tarefa utilizando o processo (Grupo 2), em geral, foi maior. Se fizermos uma média da quantidade de inconsistências encontradas, temos que o Grupo 1 encontrou aproximadamente 68% das inconsistências existentes, enquanto que o Grupo 2 encontrou 93%. Esses resultados mostram indícios de que a aplicação do processo impacta no número de inconsistências encontradas. No entanto, esse resultado é limitado, uma vez que, a amostra de participantes foi reduzida.

**Tabela 5.14. Resultados gerais da identificação de inconsistências.**

| Cenário | Inconsistência? | | | | | | |
|---|---|---|---|---|---|---|---|
| | Grupo 1 | | | Grupo 2 | | | Gabarito |
| | P1 | P2 | P3 | P4 | P5 | P6 | |
| **1** | Não | Não | Não | Não | Não | Não | Não |
| **2** | Não | Sim | Não | Sim | Sim | Sim | Sim |
| **3** | Sim | Não | Sim | Sim | Sim | Sim | Sim |
| **4** | Não | Não | Não | Não | Não | Não | Não |
| **5** | Não | Sim | Sim | Sim | Sim | Sim | Sim |
| **6** | Sim | Sim | Sim | Sim | Sim | Não | Sim |
| **Total de Inconsistências** | **2** | **3** | **3** | **4** | **4** | **3** | **4** |

No geral, todos os participantes responderam que as inconsistências não seriam completamente identificadas, antes da execução do produto, sem uma atividade de verificação. Entre as justificativas levantadas temos que os modelos de características são, geralmente, muito complexos. Esse resultado apresenta indícios de que a aplicação de um processo de verificação antecipa a identificação de inconsistências que só seriam percebidas em tempo de execução. No entanto, estudos mais específicos poderiam ser realizados para uma comprovação mais forte deste fato.

Todos os participantes responderam que não seria viável realizar manualmente a identificação de inconsistências para sistemas mais complexos, mesmo no caso em que o processo foi utilizado. Como justificativa, eles alegaram que essa era uma atividade muito desgastante, demorada e suscetível a erros. Inclusive, um dos participantes sugeriu que, no mínimo, essa atividade fosse semi-automatizada. Esse resultado permite responder a terceira pergunta relacionada à viabilidade da aplicação manual do processo, uma vez que, mesmo para um modelo relativamente pequeno, esses

problemas foram identificados. Além disso, esse resultado pode ser reforçado pelos tempos levados para a realização da tarefa nas duas etapas, que variaram de 25 a 55 minutos.

É importante lembrar que esse tempo foi gasto para a análise de apenas seis cenários de execução do produto. No entanto, os sistemas sensíveis ao contexto, estão sujeitos a centenas ou até mesmo milhares de cenários (BALDAUF *et al.*, 2007), o que tornaria inviável uma verificação de todos os casos. Por outro lado, com uma ferramenta que automatizasse esse processo, certamente essa tarefa seria mais fácil e rápida de ser executada. Esse fato motivou o desenvolvimento de um protótipo para simulação dos diferentes cenários de execução de um produto derivado a partir de uma linha de produtos e verificação da sua configuração nesses diferentes cenários. Esse protótipo é apresentado no próximo capítulo.

O estudo descrito neste capítulo apresentou um primeiro passo para a avaliação da abordagem UbiFEX, proposta neste trabalho de pesquisa. Como discutido na Seção 5.6, os resultados desse estudo são limitados em diferentes aspectos, restringindo a sua generalização.

# Capítulo 6 – Protótipo UbiFEX

## 6.1 Introdução

No Capítulo 5, foi descrito um estudo de avaliação do mecanismo UbiFEX-Simulation. Os resultados desse estudo motivaram o desenvolvimento de um protótipo (FERNANDES e WERNER, 2008a) para apoiar a aplicação da abordagem proposta.

Como apresentado no Capítulo 4, a abordagem UbiFEX tem como principal objetivo possibilitar a modelagem de características de linha de produtos de software sensíveis ao contexto. Para esse fim, foram definidos dois componentes: UbiFEX-Notation e UbiFEX-Simulation.

O modelo de características não é um artefato isolado no processo de desenvolvimento de uma LPS. Dessa forma, é interessante que a abordagem UbiFEX seja construída e aplicada em um ambiente de reutilização, que forneça o suporte necessário ao desenvolvimento e integração dos artefatos envolvidos em todas as etapas. Assim, UbiFEX foi implementada no contexto do ambiente de reutilização Odyssey (ODYSSEY, 2009), desenvolvido pelo Grupo de Reutilização de Software da COPPE/UFRJ. Isso possibilitou que as extensões e mecanismos propostos fossem inseridos dentro de uma infra-estrutura mais abrangente, que oferece suporte ao desenvolvimento de linha de produtos como um todo.

O componente UbiFEX-Notation foi incluído no ambiente por meio de extensões na implementação da notação para modelagem de características existente. Por sua vez, UbiFEX-Simulation foi desenvolvido na forma de uma ferramenta carregada por demanda (*plug-in*).

Este capítulo está organizado da seguinte forma: a Seção 6.2 apresenta uma visão geral do ambiente Odyssey; a Seção 6.3 descreve os detalhes da implementação de UbiFEX-Notation; a Seção 6.4 discute as decisões tomadas na implementação de UbiFEX-Simulation; e, finalmente, a Seção 6.5 apresenta algumas considerações finais.

## 6.2 Ambiente Odyssey

O ambiente Odyssey (ODYSSEY, 2009) é uma infra-estrutura que possui como principal objetivo apoiar a reutilização de software por meio de técnicas de linha de produtos e desenvolvimento baseado em componentes. Ele oferece suporte para as

atividades das etapas de engenharia de domínio (desenvolvimento para reutilização) e engenharia de aplicação (desenvolvimento com reutilização). Para auxiliar o desenvolvedor, o ambiente oferece seus próprios processos de reutilização.

A estrutura interna do Odyssey é organizada em níveis decrescentes de abstração, onde os elementos mais abstratos, situados no topo da estrutura, possuem rastros ou ligações para os elementos menos abstratos. O nível de abstração foco deste trabalho de pesquisa é o de características (*features*), representado no Odyssey pela visão *Features View*. É nesse nível que são representados todos os elementos referentes à atividade de modelagem de características.

O ambiente Odyssey é composto por: um núcleo, denominado Odyssey-Light, que contém as funcionalidades identificadas como essenciais para a modelagem baseada em reutilização; e um conjunto de ferramentas de apoio, implementadas na forma de *plug-ins* e carregadas por demanda, utilizando-se um mecanismo de carga dinâmica (FERNANDES *et al.*, 2007). Esses *plug-ins* auxiliam na automatização das diversas atividades definidas em um processo de reutilização. Como exemplo dessas ferramentas, podemos citar: FrameDoc (MURTA, 2001), para a documentação de componentes; Ares (VERONESE *et al.*, 2002), para engenharia reversa; Oráculo (DANTAS *et al.*, 2001), para notificação de críticas em modelos UML; e Odyssey-MDA (MAIA *et al.*, 2005), para transformação de modelos. Toda essa infra-estrutura é implementada utilizando a linguagem Java (SUN, 2009).

A modelagem de características é uma das funcionalidades implementadas no núcleo do ambiente Odyssey. A notação padrão utilizada é a Odyssey-FEX (OLIVEIRA, 2006). Outras notações são suportadas na etapa de modelagem, como as notações de GOMAA (2004) e CZARNECKI (2004). Além disso, é fornecido um mecanismo para a transição de modelos entre as notações (TEIXEIRA *et al.*, 2008). No entanto, algumas funcionalidades estão disponíveis apenas para a notação padrão, como, por exemplo, a transformação de modelos entre diferentes níveis de abstração. Dessa forma, apenas a notação padrão foi considerada como foco deste trabalho.

## 6.3 Implementação de UbiFEX-Notation

Como mencionado anteriormente, a implementação da notação UbiFEX-Notation foi realizada no núcleo do ambiente Odyssey por meio de extensões na implementação da notação Odyssey-FEX.

Esta seção está organizada da seguinte forma: a Seção 6.3.1 apresenta a estrutura semântica do ambiente Odyssey; a Seção 6.3.2 discute as extensões propostas neste trabalho na estrutura do Odyssey; e a Seção 6.3.3 detalha a representação dessas extensões no ambiente.

### 6.3.1 Estrutura semântica do ambiente Odyssey

Para um melhor entendimento das extensões implementadas, é necessário conhecer alguns detalhes da estrutura semântica do ambiente Odyssey. A Figura 6.1 ilustra um diagrama de classes que representa parte da estrutura interna do Odyssey, com o foco nas classes do núcleo que estão relacionadas com a atividade de modelagem de características.

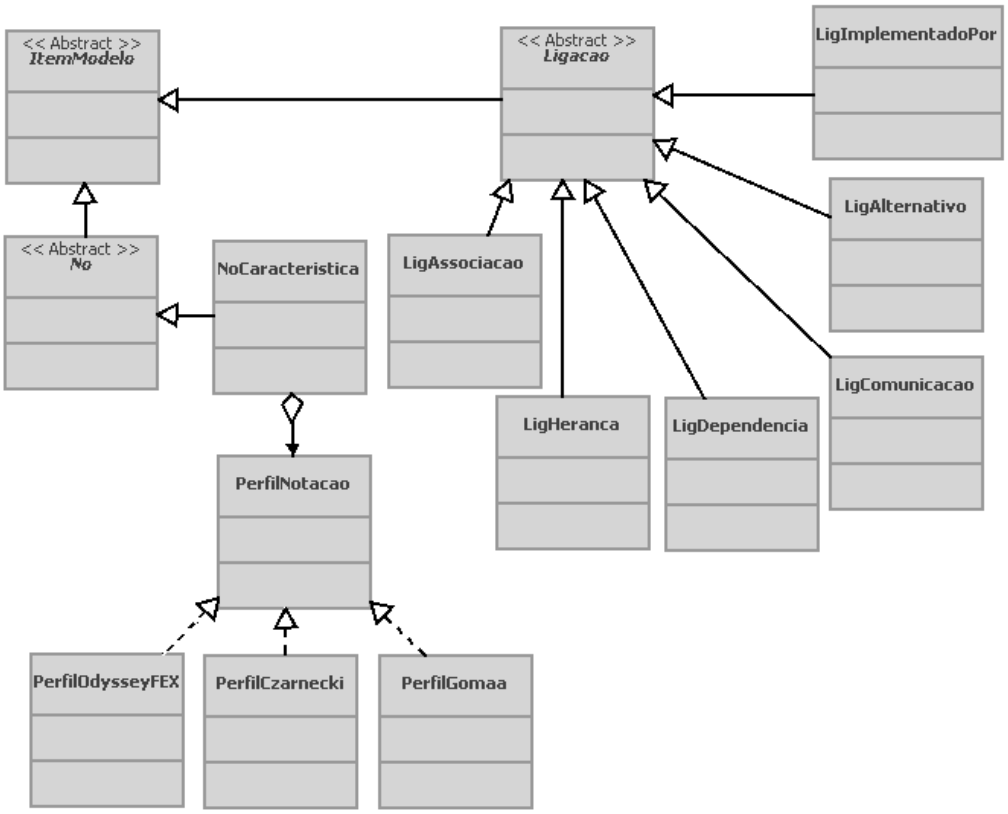



**Figura 6.1. Parte da estrutura interna do ambiente Odyssey.**

A representação interna do Odyssey é organizada hierarquicamente por meio de uma árvore semântica de objetos, que, por sua vez, é organizada em categorias de modelos. Uma dessas categorias representa o modelo de características. 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, por exemplo,

características, representadas pela classe *NoCaracteristica*. A classe abstrata *Ligacao* representa as ligações entre os diversos nós como, por exemplo, herança e associação, representadas respectivamente pelas classes *LigHerenca* e *LigAssociacao*.

A implementação das diferentes notações suportadas pelo Odyssey se baseia no padrão *State* (GAMMA *et al.*, 1995). A classe *NoCaracteristica* guarda uma instância do estado corrente, também chamado de perfil, que representa a notação que está sendo utilizada em um determinado momento no ambiente Odyssey. A classe *PerfilNotacao* corresponde a uma interface de encapsulamento do comportamento comum associado aos perfis. Cada perfil agrega as particularidades de cada notação. Por exemplo, a classe *PerfilOdysseyFEX* guarda informações relativas apenas a notação Odyssey-FEX, como, por exemplo, as suas categorias de classificação, os valores de cardinalidades, entre outros.

### 6.3.2 Extensões propostas

A primeira extensão proposta pela abordagem UbiFEX para a representação das entidades e informações de contexto foi a inclusão de dois novos tipos de categorias para as características. Dessa forma, a classe que representa o perfil referente à notação Odyssey-FEX foi renomeada para *PerfilUbiFEX*.

Cada perfil possui como extensão os diferentes tipos de características disponíveis em uma determinada notação. A Figura 6.2 apresenta a estrutura do *PerfilUbiFEX*, que representa a notação UbiFEX-Notation, destacando as modificações realizadas no perfil original da notação Odyssey-FEX. Esse perfil é estendido pelas classes: *NoCaracteristicaTecnicaImplementacional*; *NoCaracteristicaAmbienteOperacional*; *NoCaracteristicaTecnologiaDominio*; *NoCaracteristicaDominio*, que possui como especializações as classes *NoCaracteristicaFuncional* e *NoCaracteristicaConceitual*; *NoCaracteristicaEntidade;* e *NoCaracteristicaContexto*, especializada pelas classes *NoCaracteristicaEntidadeContexto* e *NoCaracteristicaInformacaoContexto*.

Além das novas categorias adicionadas, foi incluída a definição de novos elementos para a representação da influência das informações de contexto na configuração dos produtos, que são as definições e regras de contexto. A construção desses elementos se assemelha à construção das regras de composição, uma vez que todos possuem uma expressão como propriedade principal. Dessa forma, foram acrescentados à estrutura semântica do Odyssey os pacotes *DefinicoesContexto* e *RegrasContexto*, que contém as classes que representam esses novos elementos.

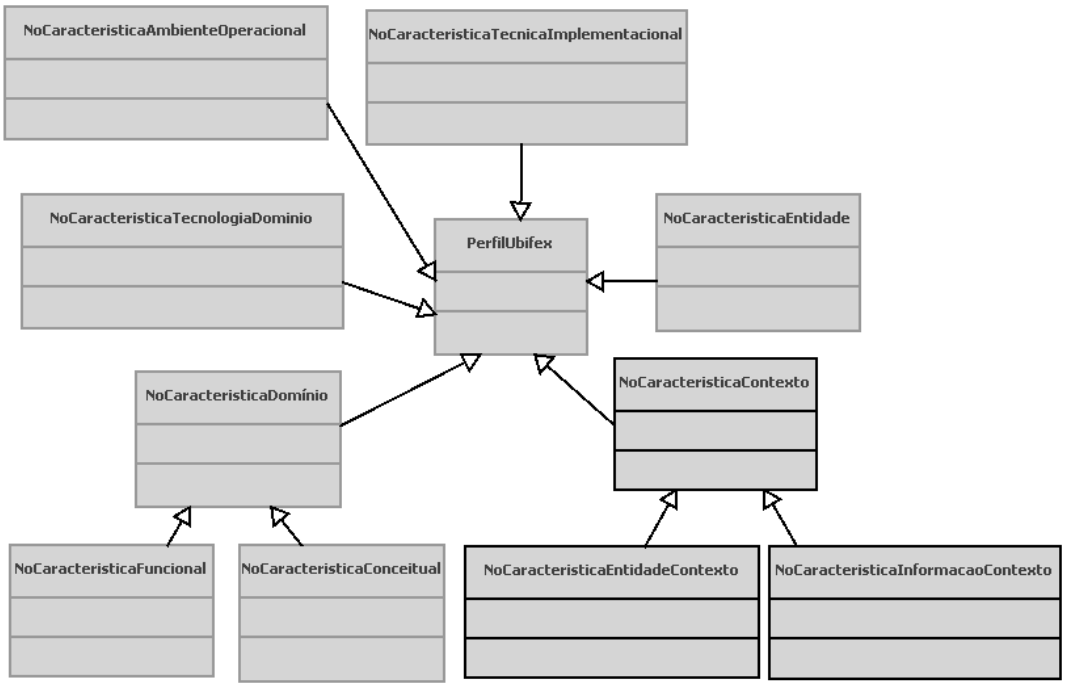



**Figura 6.2. Estrutura do perfil da notação UbiFEX-Notation**

A Figura 6.3 apresenta um detalhamento da interação entre as classes desses pacotes e o núcleo do Odyssey, destacando os novos elementos envolvidos. Semelhante ao papel da classe *GerenteRegraComposicao*, as classes *GerenteDefinicaoContexto* e *GerenteRegraContexto*, representam, respectivamente, a interface entre as definições de contexto e regras de contexto com o ambiente. Essas classes também são responsáveis por manter o rastro entre os elementos, como, por exemplo, o rastro para as características que fazem parte de uma determinada definição de contexto.

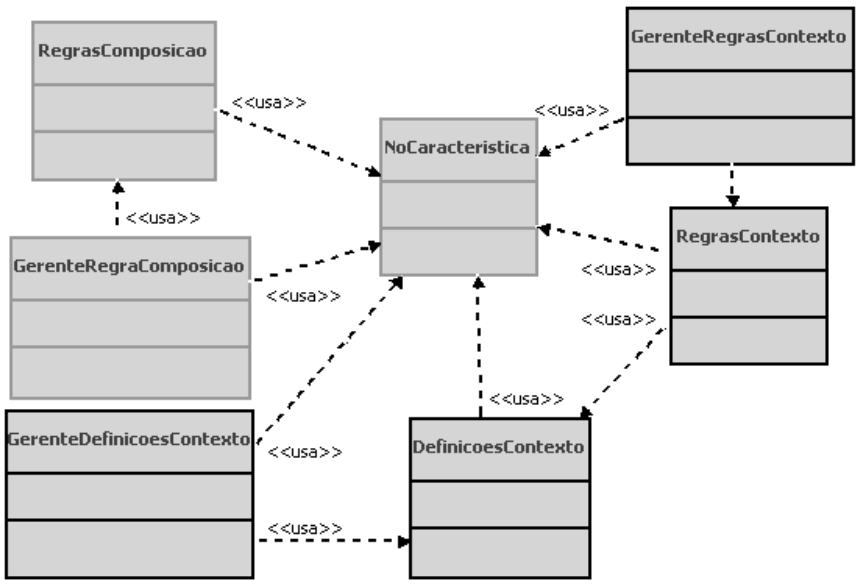



**Figura 6.3. Classes dos pacotes *DefinicoesContexto*, *RegrasContexto* e *RegrasComposicao* no ambiente Odyssey**

As expressões associadas às definições e regras de contexto são formadas por uma combinação de expressões e operadores. Um pacote denominado *Expressoes* foi criado para agrupar esse conjunto de operadores e expressões. A Figura 6.4 apresenta as classes que fazem parte desse pacote. As classes *AND*, *OR*, *XOR* e *NOT* representam, respectivamente, as expressões do tipo E, OU, OU-Exclusivo e NÃO. As classes *ExpressaoLiteralCaracterística* e *ExpressaoLiteralContexto* representam, respectivamente, expressões formadas apenas por uma característica ou definição de contexto. A classe *ExpressaoBooleana* se refere às expressões que envolvem operadores relacionais, utilizadas para representar as definições de contexto. Essas expressões são interpretadas segundo o padrão *Interpreter* (GAMMA *et al.*, 1995).

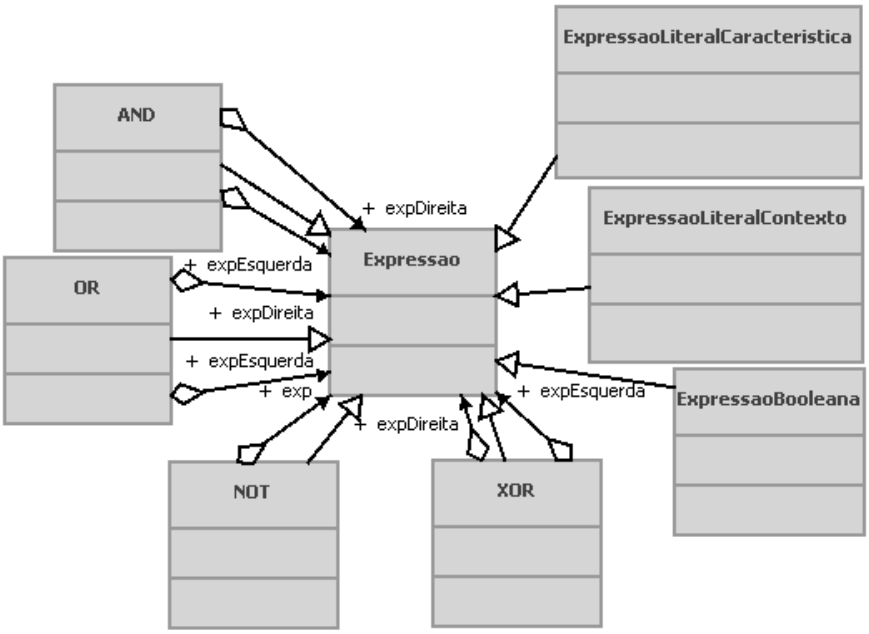



**Figura 6.4. Classes do pacote *Expressoes*.**

### 6.3.3 Representação das extensões propostas no ambiente Odyssey

Esta seção descreve como cada uma das extensões propostas é representada no ambiente Odyssey, incluindo as novas categorias de características criadas (Seção 6.3.3.1), as definições (Seção 6.3.3.2) e regras de contexto (Seção 6.3.3.3). Ao final, é apresentada uma visão geral do ambiente após as extensões (Seção 6.3.3.4) e as adaptações realizadas no processo de engenharia de aplicação (Seção 6.3.3.5).

#### 6.3.3.1 Novas categorias de características

No ambiente Odyssey, as características são detalhadas em um padrão de documentação, denominado padrão de domínio (MILER, 2000). Esse padrão foi

adaptado para descrever cada uma das novas categorias de características propostas em UbiFEX-Notation. O detalhamento das características pode ser realizado por meio da aba *Feature*, que contém todos os campos necessários para descrever cada uma das categorias. A Figura 6.5 e a Figura 6.6, ilustram, respectivamente, as telas que representam o padrão de domínio utilizado para descrever características do tipo entidade de contexto e informação de contexto.

No padrão de domínio de uma característica, os campos referentes à notação utilizada e à classificação quanto à categoria, presentes em ambas as telas, não permite edição, sendo definidos com a notação e o tipo da característica criada. Além disso, existem outros campos comuns, como: nome, descrição, prioridade, opcionalidade e sinônimos.

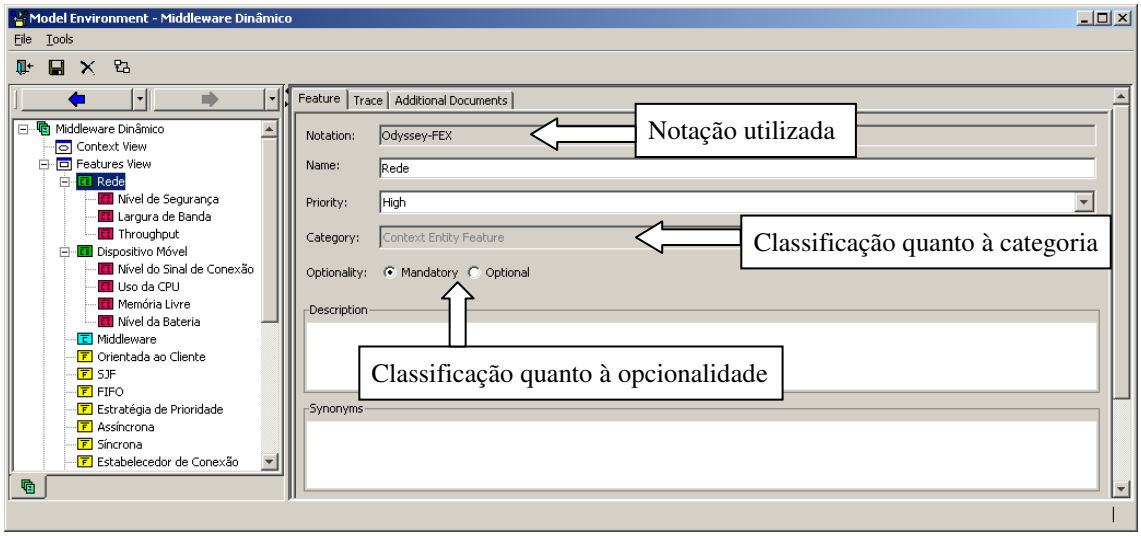



**Figura 6.5. Tela para descrição de características do tipo entidade de contexto.**

Na Figura 6.6, existem ainda campos específicos para descrever as outras propriedades definidas para as características do tipo informação de contexto, descritas no Capítulo 4, como: tipo base, persistência, composição e aquisição. Nesta tela, foram adicionados ainda alguns campos relacionados ao mecanismo UbiFEX-Simulation para incluir os dados da simulação, os quais serão detalhados na Seção 6.4.

O padrão de domínio também possibilita estabelecer a rastreabilidade de uma característica, ou seja, estabelecer uma ligação da característica com os demais elementos do modelo ou artefatos do domínio, como regras de composição, definições de contexto, tipos de negócio, casos de uso, componentes, entre outros. Essa rastreabilidade é o ponto central do processo de Engenharia de Aplicação e é definida por meio da aba *Trace*, presente nas telas apresentadas em ambas as figuras.

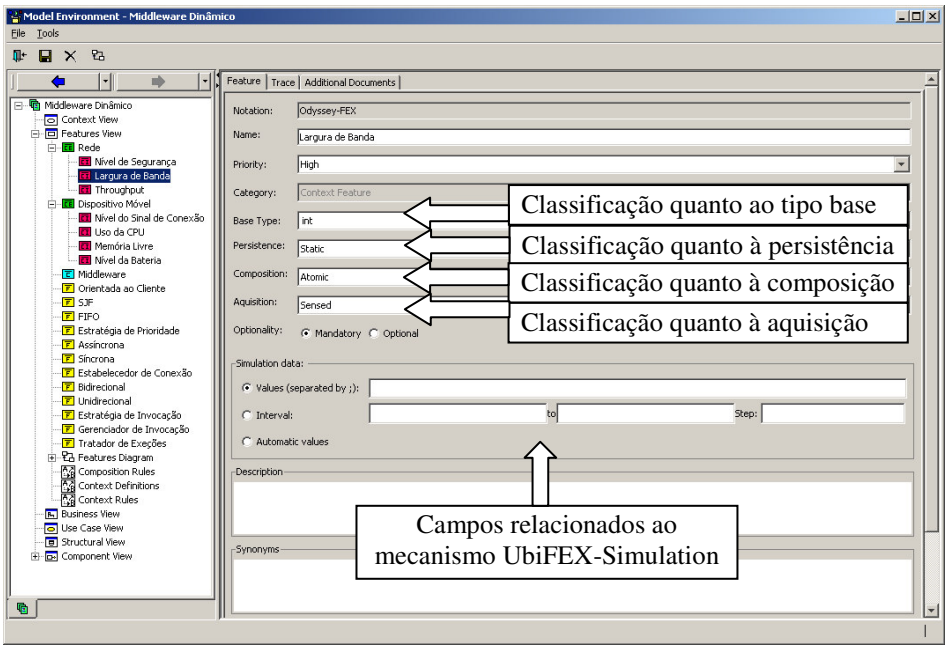



**Figura 6.6. Tela para descrição de características do tipo informação de contexto.**

### 6.3.3.2 Definições de contexto

As definições de contexto foram propostas para descrever as situações ou contextos relevantes para o domínio. Elas são criadas a partir das características do tipo informação de contexto presentes no ambiente, ou seja, para a criação de uma definição de contexto, é necessária a modelagem prévia das características envolvidas.

No ambiente Odyssey, o processo de criação de uma definição de contexto está inserido na visão de características, denominada *Features View*, por meio da opção *Context Definitions* (ver Figura 6.7). Cada definição de contexto instanciada se torna parte da árvore semântica do Odyssey. A Figura 6.7 apresenta a tela para criação da expressão que representa uma determinada definição de contexto.

A expressão final pode ser formada apenas por uma expressão que envolve: uma característica do tipo informação de contexto, previamente especificada no modelo; um operador relacional (*maior*, *maior que*, *menor*, *menor que*, *igual* ou *diferente*); e um valor. Ou ainda, pela combinação de várias expressões por meio dos operadores lógicos: OU (*OR*), E (*AND*) ou NÃO (*NOT*).

No exemplo apresentado na Figura 6.7, inicialmente foram criadas as expressões (*Rede.Throughput < 64*) e (*Rede. Largura de Banda >= 1024*) e, em seguida, elas foram combinadas utilizando o operador E (*AND*). Cada expressão adicionada (*Add*) se torna visível nas caixas de seleção que podem ser combinadas pelos operadores lógicos. Isso possibilita que uma nova definição de contexto seja criada a partir de outras

previamente definidas. Os parênteses são colocados automaticamente a cada expressão criada, evitando que ocorram ambigüidades na interpretação das expressões. A opção *Accept Expression* associa a expressão que está sendo criada à expressão final.

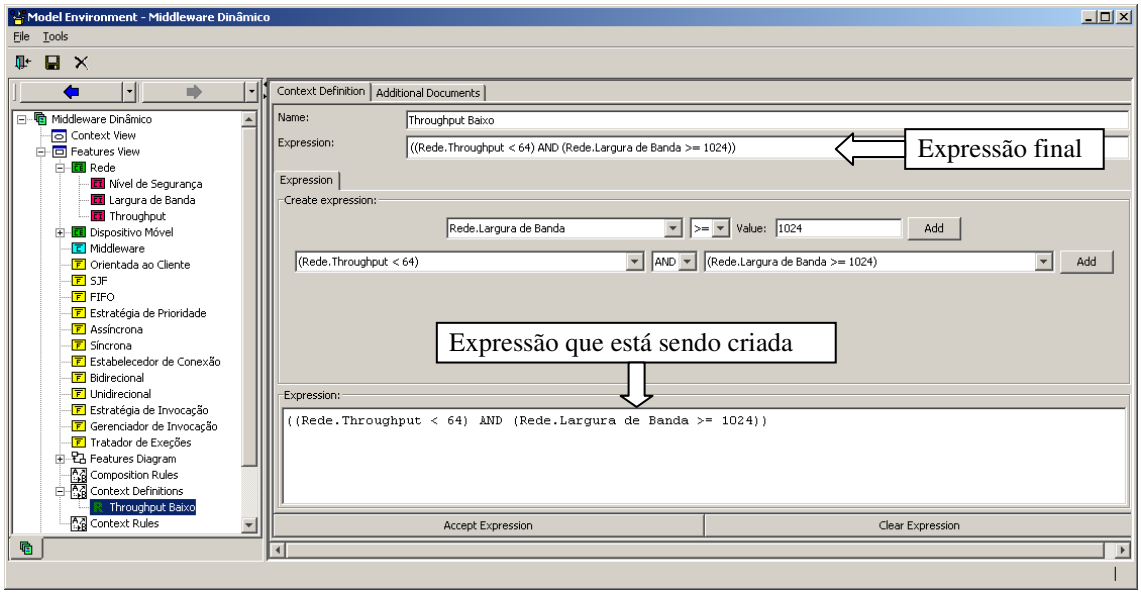



**Figura 6.7. Tela para criação da expressão que representa uma definição de contexto.**

### 6.3.3.3 Regras de contexto

As regras de contexto representam como um contexto previamente definido impacta na configuração dos produtos de uma LPS em tempo de execução, indicando, por exemplo, qual decisão a ser tomada em um determinado ponto de variação.

No ambiente Odyssey, o processo de criação de uma regra de contexto está inserido na visão *Features View*, por meio da opção *Context Rules* (ver Figura 6.8). Cada regra de contexto instanciada se torna parte da árvore semântica do Odyssey e recebe como identificador CR_X, onde X é incrementado seqüencialmente.

A expressão que representa uma regra de contexto é formada por um antecedente, o operador "implica" (*implies*) e um conseqüente. Quando uma regra de contexto é criada, seu antecedente e conseqüente são nulos e apresentados na definição da regra, respectivamente, por <*undefined antecedent*> e <*undefined consequent*>. Como o antecedente e o conseqüente são formados por elementos diferentes, eles são definidos em painéis distintos, representados pelas abas *Antecedent* e *Consequent*.

A Figura 6.8 ilustra a tela utilizada para a definição de uma regra de contexto, destacando o painel para a definição do antecedente da regra. A Figura 6.9, por sua vez, destaca a definição do conseqüente da regra.

O antecedente de uma regra de contexto pode ser formado por apenas uma definição de contexto, que depois de escolhida na primeira caixa de seleção, é adicionada na expressão que está sendo criada por meio do botão *Context*. Além disso, pode ser formado por uma combinação de definições de contexto e características, por meio de operadores lógicos. Para garantir que o antecedente tenha no mínimo uma definição de contexto envolvida, a primeira caixa de seleção não inclui características para seleção. No exemplo apresentado na Figura 6.8, o antecedente é formado apenas pela definição de contexto *Throughput Baixo*. Para atribuir essa expressão efetivamente ao antecedente da regra, deve ser utilizado o botão *Accept Expression*.

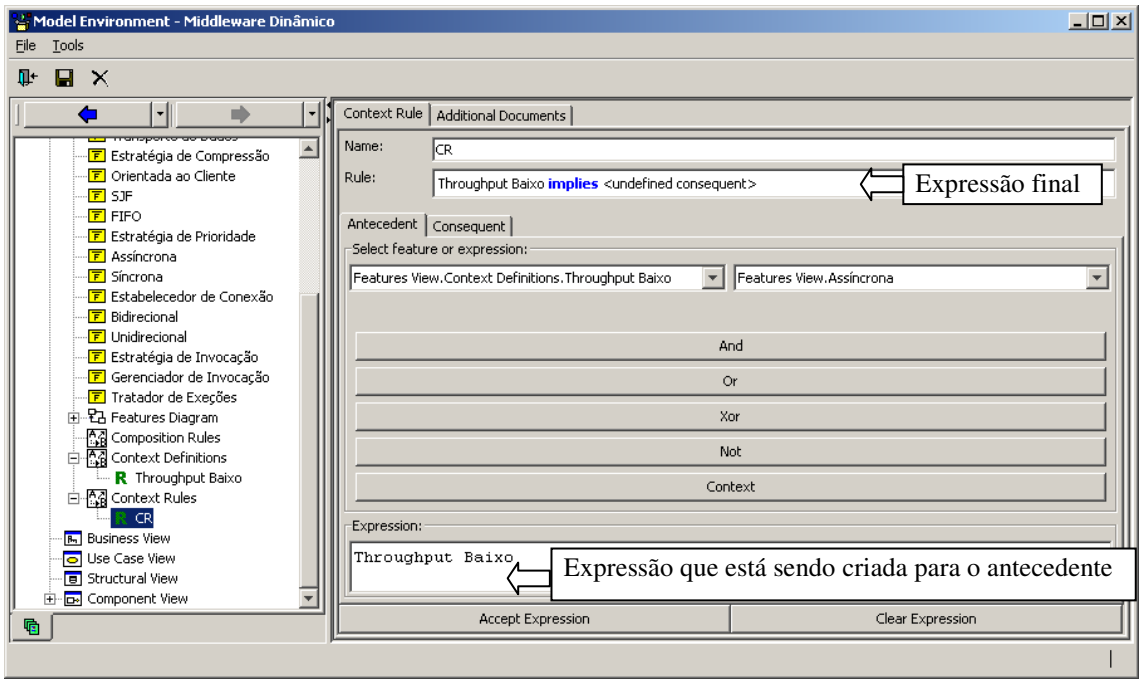



**Figura 6.8. Tela para definição de uma regra de contexto com destaque para o antecedente.**

O conseqüente pode ser formado por uma característica que, depois de escolhida na primeira caixa de seleção, é adicionada por meio do botão *Feature* na expressão que está sendo criada. Além disso, pode ser formado por um conjunto de características combinadas utilizando os operadores lógicos E (*AND*) e NÃO (*NOT*). As características apresentadas nas caixas de seleção para a construção das expressões atendem às restrições de classificação definidas pela abordagem no Capítulo 4, que incluem: não ser uma característica de contexto; ser uma característica opcional; e ser uma invariante, ponto de variação ou variante.

Como mencionado no Capítulo 4, a presença de uma característica no conseqüente de uma regra de contexto indica a sua adição na configuração do produto. No entanto, a aplicação do operador NÃO (*NOT*) indica a remoção do conjunto de

características ao qual ele é aplicado. No exemplo apresentado na Figura 6.9, o conseqüente é formado pela expressão (*Estratégia de Compressão AND Ótima*), indicando a adição do ponto de variação, já com a decisão de qual variante deve ser selecionada. Para os casos onde a variante não é incluída no conseqüente da regra, foi introduzido na notação Odyssey-FEX o conceito de variante padrão, que é configurada na definição de cada ponto de variação e indica qual variante deve ser escolhida em um determinado ponto de variação, caso nenhuma outra tenha sido selecionada. Para atribuir essa expressão efetivamente ao conseqüente da regra, deve ser utilizado o botão *Accept Expression*.

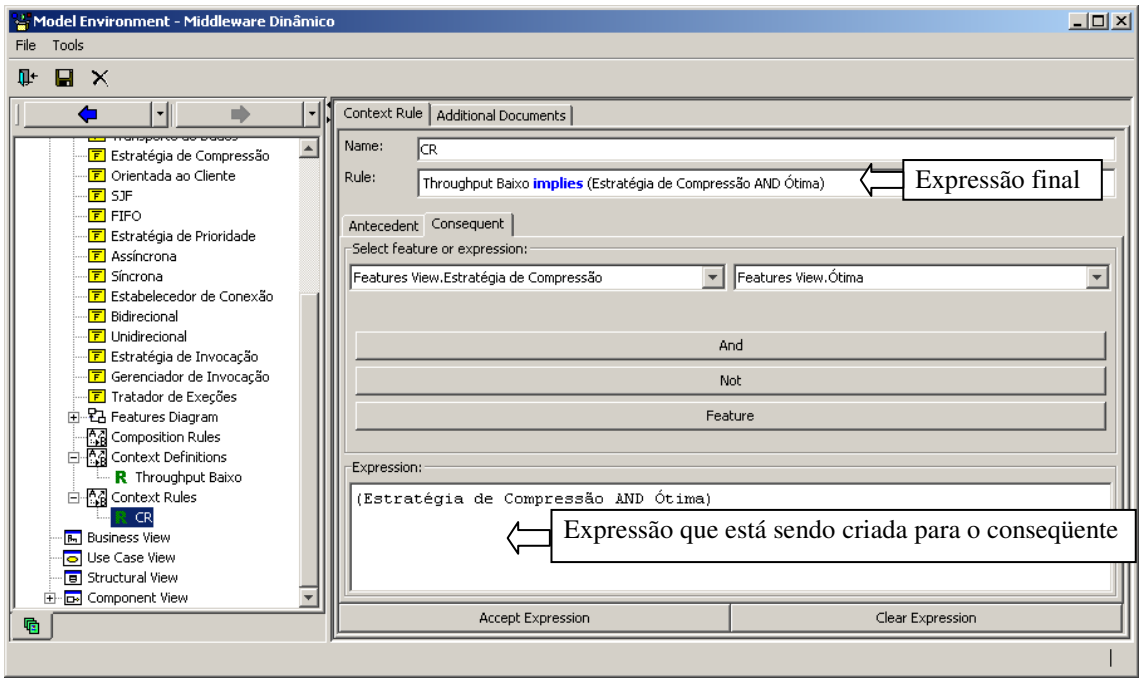



**Figura 6.9. Tela para definição de uma regra de contexto com destaque para o conseqüente.**

A regra de contexto criada a partir da definição do antecedente e do conseqüente descritos anteriormente é a seguinte: Throughput Baixo *implies* (Estratégia de Compressão *AND* Ótima), indicando que: se a definição de contexto *Throughput Baixo* estiver ativa para um determinado ambiente, as características *Estratégia de Compressão* e *Ótima* devem ser adicionadas à configuração do produto.

Cada expressão criada por meio dos botões que representam as operações lógicas se torna visível nas caixas de seleção, podendo ser selecionada como parte de outra expressão. Dessa forma, é possível definir expressões mais complexas, tanto para o antecedente, quanto para o conseqüente de uma regra de contexto. Os parênteses são colocados automaticamente quando a expressão é criada, como forma de evitar ambigüidades na interpretação da regra.

Apesar da notação UbiFEX-Notation classificar as regras de contexto em obrigatórias e opcionais, por limitação na implementação de UbiFEX-Simulation, que trata todas as regras como obrigatórias, essa funcionalidade não foi incluída no painel de descrição das regras.

As regras de contexto também podem ser visualizadas por meio das características que fazem parte do seu conseqüente, que são marcadas no canto superior esquerdo com o identificador da regra, permitindo a identificação das características que sofrem influência de contexto de forma explícita. Por exemplo, no diagrama apresentado na Figura 6.10, a característica Assimétrica faz parte do conseqüente da regra de contexto cujo identificador é CR_2.

#### 6.3.3.4 Diagramador de Características

Além da estrutura semântica apresentada nas seções anteriores, o ambiente Odyssey possui uma estrutura léxica formada por diagramas específicos a cada modelo representado. Para todo item de modelo semântico, existem zero ou mais itens léxicos, nós e ligações que formam os desenhos dos diagramas. Cada um desses itens léxicos, por sua vez, está associado a um único item semântico. Esses modelos são construídos em uma ferramenta de diagramação, denominada Editor de Diagramas.

Cada uma das categorias de diagramas presente no ambiente Odyssey (e.g., Diagrama de Características, Diagrama de Classes, Diagrama de Componentes e outros) é representada por um painel específico, incluindo os elementos pertencentes àquele diagrama.

A janela do diagramador de características do Odyssey, após a implementação das extensões propostas, pode ser visualizada na Figura 6.10. À esquerda, identificamos a árvore de itens semânticos instanciados, e à direita, o diagrama com o desenho que representa as características (nós) e suas ligações (arestas). A exibição desse diagrama corresponde ao item semântico *Features Diagram*, que está selecionado na árvore.

Na parte superior da janela, existe uma barra de ferramentas que auxilia o usuário no processo de modelagem. Nessa barra, encontram-se as ações necessárias para a construção do modelo, como, por exemplo, as ações relativas à criação de características e das ligações entre elas.

Na implementação das extensões propostas em UbiFEX-Notation, foram adicionados na árvore da esquerda os itens semânticos que representam as definições e as informações de contexto. Além disso, na barra de ferramentas foram adicionados os

ícones referentes às ações de criação das características do tipo entidade e informação de contexto. Essas alterações estão destacadas na Figura 6.10.

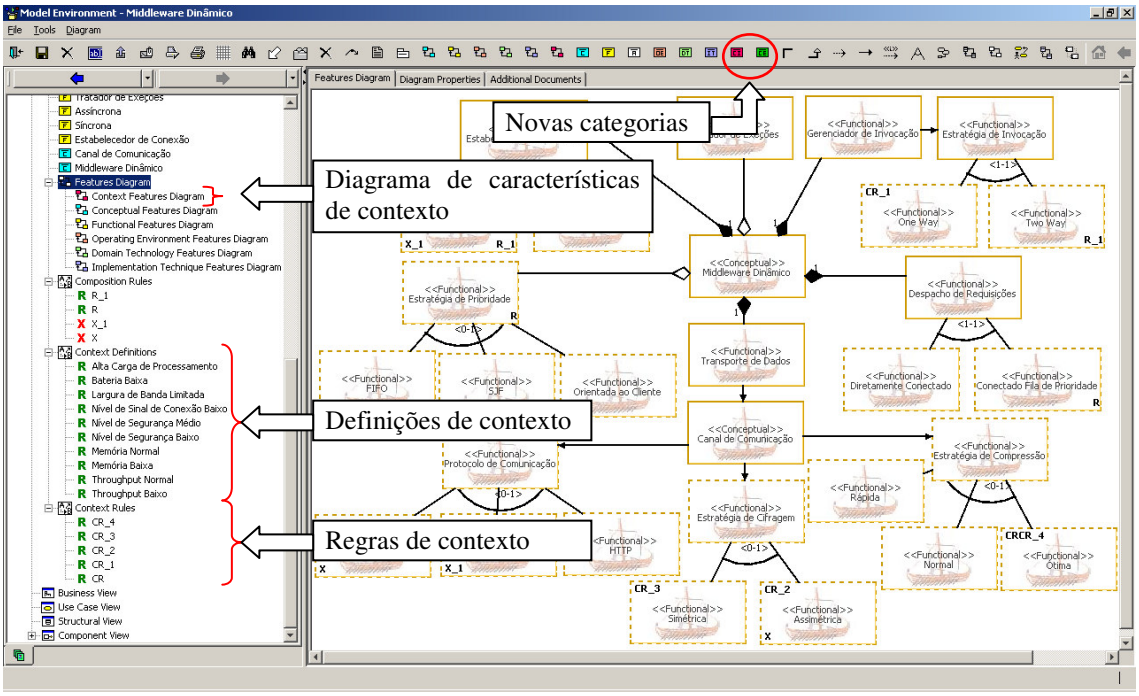



**Figura 6.10. Diagramador de características do ambiente Odyssey.**

### 6.3.3.5 O processo de engenharia de aplicação

Como foi definido no Capítulo 3, o processo de Engenharia de Aplicação (EA) compreende o desenvolvimento com reutilização, que consiste no processo de reutilização dos artefatos gerados durante o processo de engenharia de domínio para o desenvolvimento de produtos. Durante a EA, é feita a seleção dos componentes necessários à aplicação em um alto nível de abstração (MILER, 2000).

No ambiente Odyssey, o modelo de características é um dos principais elementos utilizado na etapa de EA. Essa etapa é iniciada por meio de um guia (*wizard*), que inclui todos os passos para a seleção dos artefatos a partir da engenharia de domínio. Dessa forma, foi necessário realizar algumas adaptações no processo existente para possibilitar a derivação de produtos a partir de linha de produtos sensíveis ao contexto modeladas utilizando UbiFEX-Notation.

Na implementação dessas adaptações era desejado manter o suporte do Odyssey ao desenvolvimento de linha de produtos tradicionais. Dessa forma, uma primeira extensão foi realizada para indicar o tipo de aplicação que estava sendo derivada, se era uma aplicação tradicional ou uma aplicação sensível ao contexto, como ilustrado na Figura 6.11.

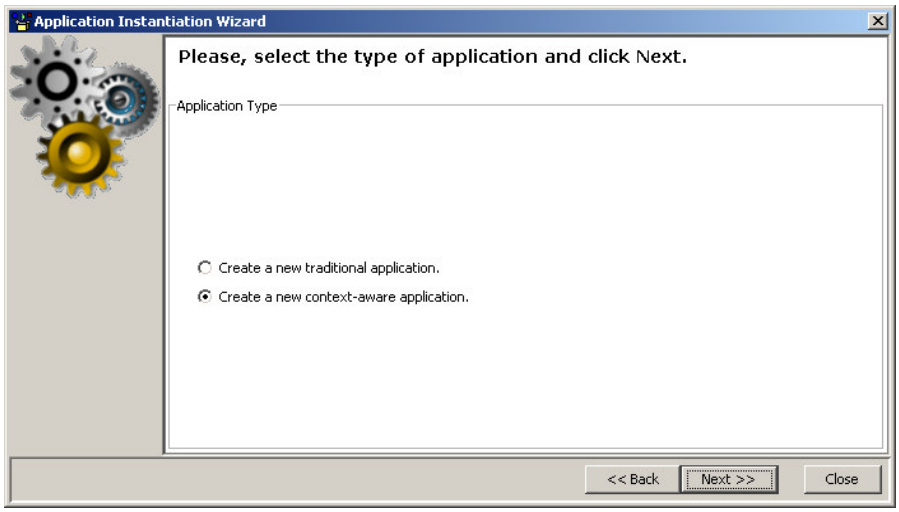



**Figura 6.11. Janela para seleção do tipo de aplicação derivada.**

No caso de uma aplicação tradicional, o processo continuou o mesmo. No entanto, para a criação de uma aplicação derivada a partir de uma linha de produtos sensíveis ao contexto, além da seleção das características do domínio, foi necessário adicionar um painel para a seleção das características de contexto relevantes para o produto (Figura 6.12).

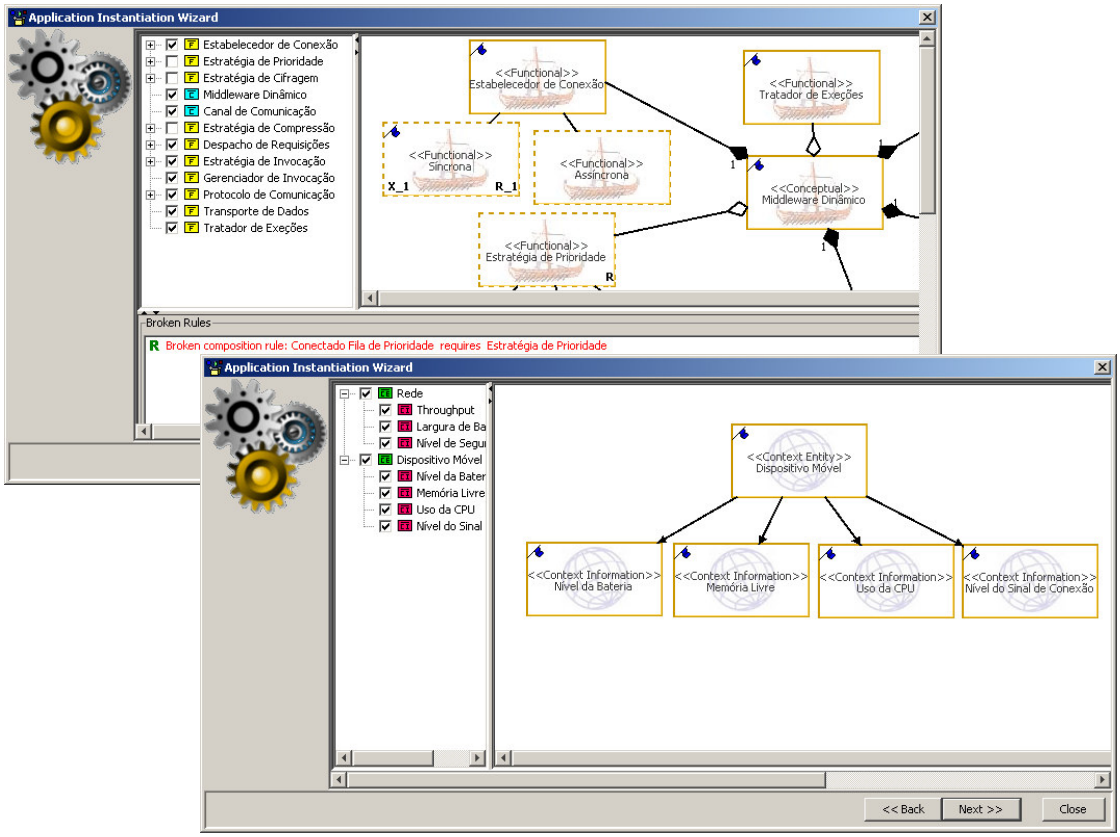



**Figura 6.12. Seleção das características do produto.**

Na Figura 6.12, as características que são mandatórias no domínio já se encontram selecionadas. As características não selecionadas representam as características opcionais no domínio, cabendo ao desenvolvedor selecionar as características desejadas. À medida que as características são selecionadas, elas são marcadas no modelo apresentado na parte direita da tela, como pode ser observado nas características *Estabelecedor de Conexão* e *Síncrona*.

Nas aplicações tradicionais, as características do domínio selecionadas já fazem parte da configuração inicial do produto, portanto, nessa etapa, todas as verificações quanto às restrições definidas entre as características como, por exemplo, as regras de composição inclusivas e exclusivas, são realizadas. Caso alguma inconsistência seja encontrada, elas são listadas no final do painel de seleção das características do domínio, na área denominada *Broken Rules*. Porém, para os produtos derivados a partir de uma linha de produtos sensíveis ao contexto, essa seleção indica apenas um "pré-recorte" do domínio, uma vez que as configurações dos produtos podem ser definidas apenas em tempo de execução ou, no caso da configuração inicial, a partir de uma seleção posterior a esse "pré-recorte".

Dessa forma, essa etapa de verificação foi adaptada para excluir as inconsistências quanto às regras de composição exclusivas, uma vez que a seleção das características, nessa etapa, não indica que elas estarão presentes ao mesmo tempo na configuração do produto. No entanto, a verificação quanto às regras inclusivas foi mantida. No exemplo apresentado na Figura 6.12, temos que: a regra de composição Conectado Fila Prioridade *requires* Estratégia de Prioridade está sendo quebrada, pois a característica do antecedente está selecionada, mas a do conseqüente não.

No Odyssey, o processo de EA utiliza o recurso de rastreabilidade existente entre os artefatos construídos no domínio. A seleção de uma característica da árvore implica na inclusão desta e dos artefatos a ela relacionados no novo produto a ser desenvolvido. Assim, a partir da seleção das características de contexto, os outros elementos especificados, como as definições e as regras de contexto, são automaticamente selecionados para o produto.

Ao término do processo de seleção dos artefatos para o produto, ajustes podem ser realizados ou aspectos específicos da aplicação podem ser modelados, possibilitando assim o desenvolvimento de um novo produto. É também na EA que o mecanismo UbiFEX-Simulation pode ser aplicado, como descrito na seção a seguir.

## 6.4 Implementação de UbiFEX-Simulation

Como apresentado no Capítulo 4, UbiFEX-Simulation possui como objetivo verificar, em tempo de desenvolvimento, a consistência da configuração de um determinado produto para um determinado cenário de execução. Esse mecanismo envolve a verificação da consistência entre as regras de contexto presentes no modelo de características que representa o produto e a verificação da consistência dessas regras em relação às restrições de configuração dos produtos.

O mecanismo UbiFEX-Simulation foi implementado na forma de um *plug-in* para o ambiente Odyssey e pode ser carregado por demanda. A Figura 6.13 apresenta um modelo de classes simplificado do *plug-in* UbiFEX-Simulation. Esse *plug-in* é baseado na simulação dos diferentes cenários de execução de um produto, partindo de uma determinada configuração inicial.

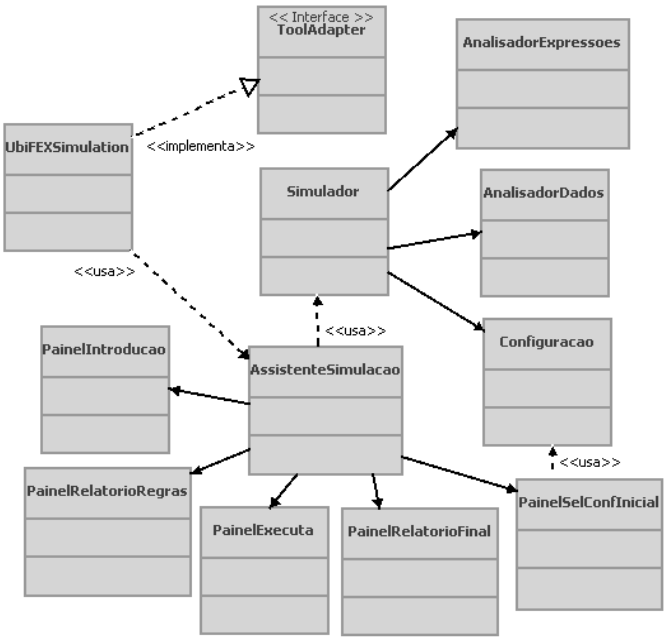



**Figura 6.13. Diagrama de classes simplificado do *plug-in* UbiFEX-Simulation.**

Para a implementação desse *plug-in*, a interface *ToolAdapter*, disponibilizada pelo ambiente Odyssey para a inclusão de novos *plug-ins*, foi implementada pela classe *UbiFEXSimulation*. A classe *AssistenteSimulacao* é responsável pelos elementos de interface com o usuário e é apresentada na forma de um *wizard*, com os passos para a execução do processo proposto em UbiFEX-Simulation. As classes *Simulador*, *AnalisadorExpressoes* e *AnalisadorDados* são responsáveis pelo funcionamento interno do mecanismo de simulação, incluindo a associação dos dados e a análise das regras. Por fim, a classe *Configuracao* representa as configurações do produto.

Uma das atividades anteriores à execução do *plug-in* é a definição dos valores que serão simulados para cada característica do tipo informação de contexto especificada. Como forma de facilitar a definição desses valores, foi utilizado o mesmo painel de descrição desse tipo de característica, como destacado na Figura 6.14. Eles podem ser definidos de três formas: uma lista de valores, um intervalo ou de forma automática. Neste último caso, os valores são calculados no processo de simulação com base nas expressões das definições de contexto que a característica faz parte, incluindo valores que testem as expressões tanto como verdadeiras quanto como falsas. Por exemplo, para a seguinte expressão *Dispositivo Móvel.Memória Livre < 128*, que representa a definição de contexto *Memória Baixa*, no modo automático, são gerados três valores para a característica *Memória Livre*: um maior, um menor e um igual a 128. Todas as características possuem o modo automático selecionado por padrão.

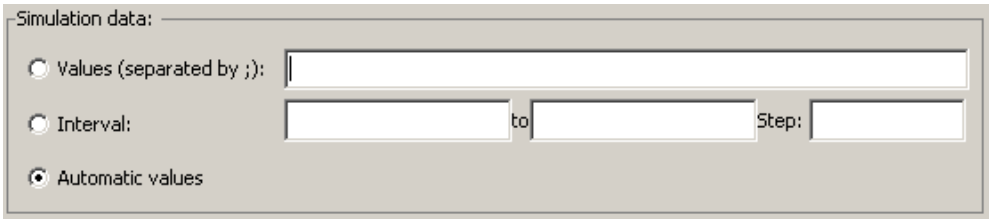



**Figura 6.14. Diferentes formas de definir os dados de simulação.**

Para utilização do *plug-in*, o engenheiro de domínio deve acessar o menu *Tools→UbiFEX-Simulation→Context Simulation Wizard* (Figura 6.15). Ao selecionar essa opção, é aberta a janela do *wizard* para execução da simulação.

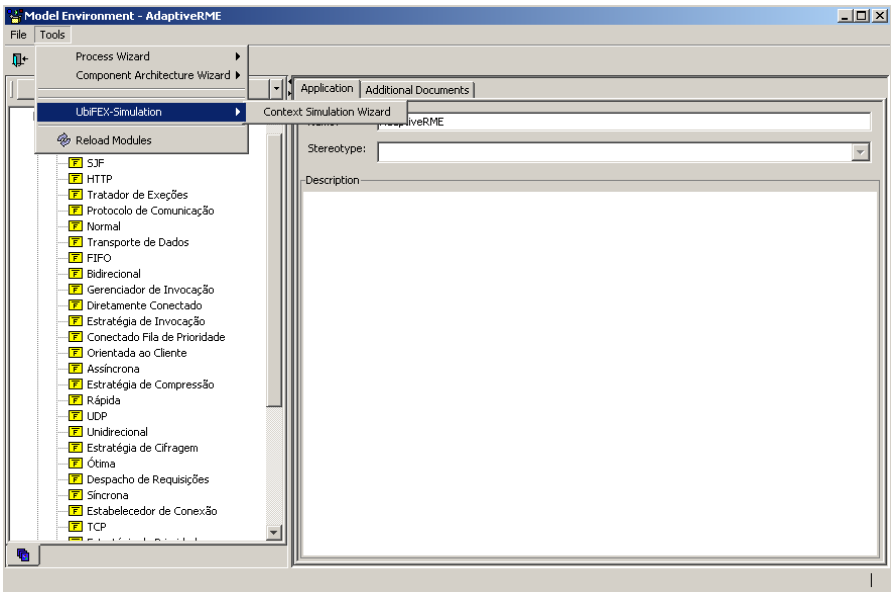



**Figura 6.15. Acesso ao *plug-in* UbiFEX-Simulation.**

O primeiro passo do *wizard* é a verificação da consistência apenas entre as regras de contexto, que envolvem as inconsistências do tipo intra e inter-regra. Um relatório é apresentado com o resultado dessa verificação. As inconsistências inter-regras são exibidas apenas como forma de alertar o engenheiro de domínio que as regras apresentadas não devem ser executadas em um mesmo cenário, como ilustrado na Figura 6.16. Nesse caso, as regras de contexto CR e CR_4 incluem e excluem a característica *Ótima*.

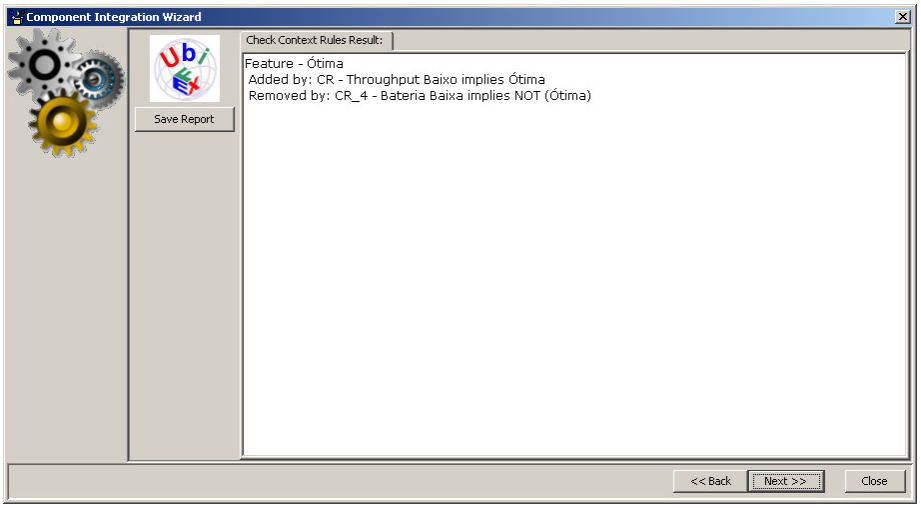



**Figura 6.16. Relatório de inconsistências entre regras de contexto.**

O passo seguinte é a seleção da configuração inicial do produto (Figura 6.17), a partir da qual as novas configurações serão geradas. Essa atividade é semelhante à que ocorre no momento da instanciação de um produto. No entanto, nesse caso, todas as restrições definidas no modelo devem ser verificadas, garantindo a consistência da configuração inicial do produto. Caso alguma regra ou restrição esteja sendo quebrada, ela é listada na área de texto denominada *Broken Rules*.

Após a definição dos valores que serão simulados e da configuração inicial do produto, o próximo passo é a execução do processo de verificação das novas configurações, que pode ser acompanhada por meio de uma barra de progresso. Inicialmente, são gerados todos os cenários de execução da simulação, por meio da combinação entre os valores definidos para cada uma das características do tipo informação de contexto. Essa combinação é feita por meio do produto cartesiano entre todos os conjuntos de valores definidos. Por exemplo, para duas características com dois valores de simulação cada uma, quatro cenários seriam gerados.

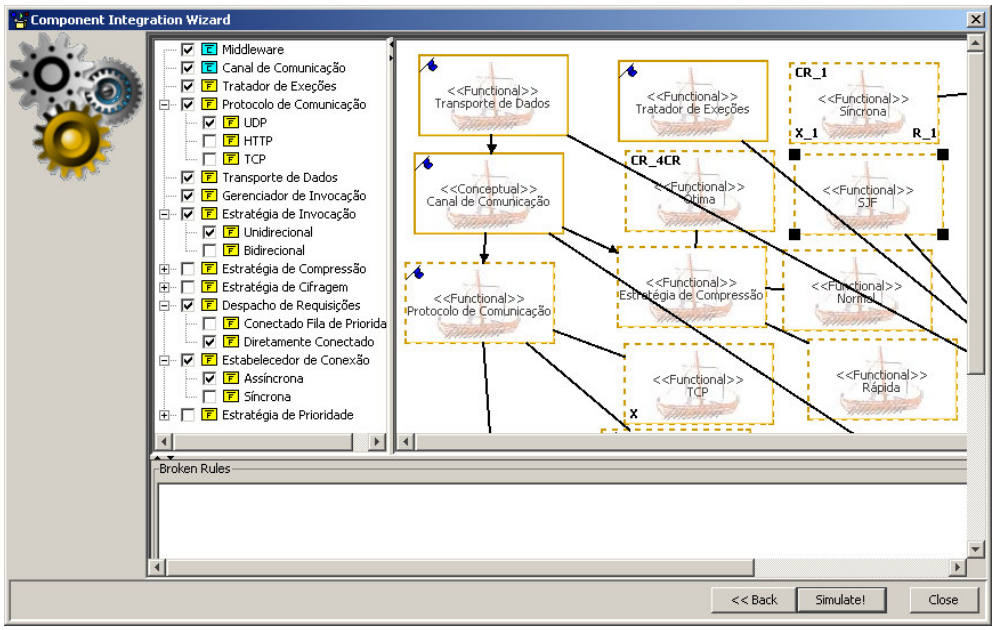



**Figura 6.17. Seleção da configuração inicial do produto.**

Em seguida, o processo proposto em UbiFEX-Simulation é executado para cada um dos cenários. Dessa forma, os valores do cenário são associados às informações de contexto, as definições de contexto ativas são determinadas e as regras de contexto executadas. Por fim, uma nova configuração é gerada e a sua consistência, em relação às restrições das regras de composição e cardinalidade, é verificada. Ao final do processo de verificação, é exibido um relatório com os resultados obtidos (Figura 6.18).

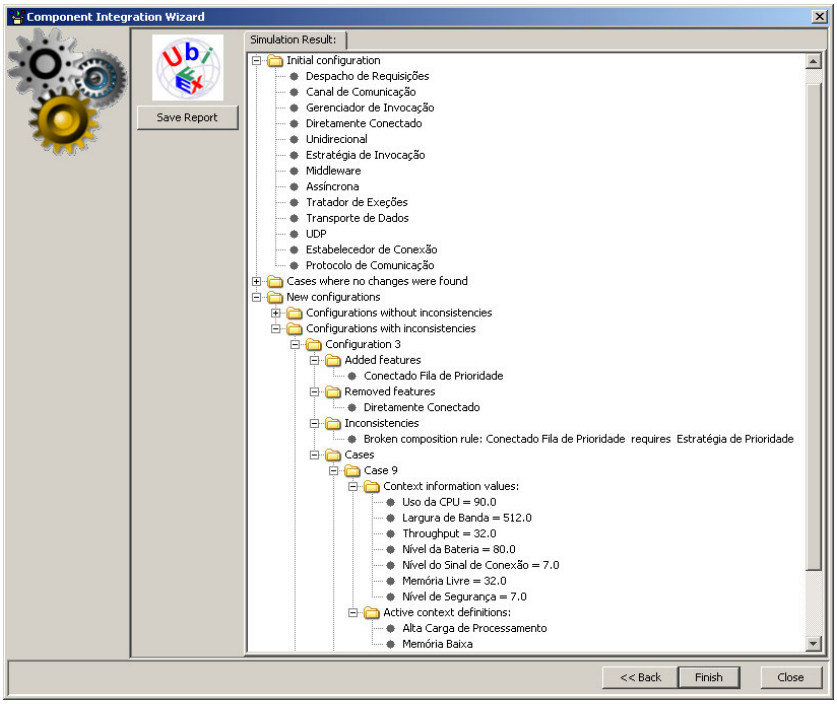



**Figura 6.18. Relatório final da simulação.**

O resultado apresentado no relatório final da simulação inclui as seguintes informações: a configuração inicial do produto, selecionada na etapa anterior; os cenários onde não ocorreram mudanças na configuração do produto, ou seja, o produto permaneceu com a configuração inicial; e as novas configurações geradas, que são agrupadas em dois grupos, dependendo se foram ou não identificadas inconsistências durante o processo de verificação. Para cada nova configuração, são destacadas: as características adicionadas e removidas em relação à configuração inicial; as inconsistências identificadas, quando houver; e os cenários que deram origem àquela configuração. Além disso, para cada cenário, são apresentados os valores das informações de contexto e as definições de contexto ativas.

No exemplo apresentado na Figura 6.18, temos que a Configuração 3 foi gerada a partir da adição da característica *Conectado Fila Prioridade* e a remoção da característica *Diretamente Conectado*. Essas modificações levaram o produto a uma configuração inconsistente, pois a regra de composição inclusiva *Conectado Fila Prioridade requires Estratégia de Prioridade* está sendo quebrada e a característica *Estratégia de Prioridade* não está presente nessa configuração. É possível ainda identificar os cenários que levam a essa configuração inconsistente. No caso do Cenário 9, essa situação ocorreu quando as definições de contexto *Alta Carga de Processamento* e *Memória Baixa* estavam ativas.

De posse desses resultados, o engenheiro de domínio pode realizar as alterações necessárias no modelo de características para evitar que essas configurações inconsistentes se propaguem até a execução do produto. No exemplo apresentado, seria necessário adicionar a característica requerida *Estratégia de Prioridade* no conseqüente da regra de contexto responsável pela inclusão da característica *Conectado Fila Prioridade*. Como forma de reduzir esse tipo de problema, foi adicionado no ambiente de modelagem um mecanismo de alerta na criação das regras de contexto, para indicar se alguma das características que fazem parte do conseqüente dessas regras, também fazem parte de alguma regra de composição (Figura 6.19).

## 6.5 Considerações finais

Neste capítulo, foi apresentado um protótipo que implementa a abordagem UbiFEX no contexto do ambiente Odyssey. A notação UbiFEX-Notation foi implementada no núcleo do Odyssey e o mecanismo UbiFEX-Simulation foi disponibilizado na forma de um *plug-in*.

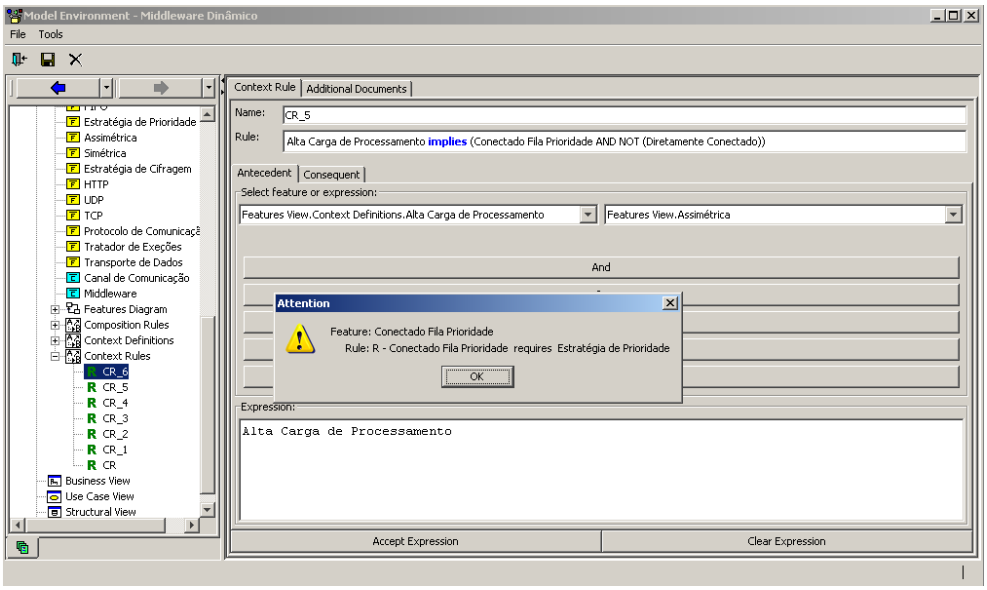



**Figura 6.19. Mecanismo de alerta na criação de regras de contexto.**

O protótipo desenvolvido busca apoiar a aplicação da abordagem e foi motivado pelos resultados obtidos no estudo de avaliação apresentado no Capítulo 5. Nesse estudo, os participantes realizaram a verificação da configuração de um produto em apenas seis cenários e levaram um tempo entre 25 a 55 minutos. Utilizando o ferramental desenvolvido, para o mesmo modelo de característica desse estudo, um total de mil cenários foram verificados em aproximadamente 5 minutos, considerando um computador com CPU Core 2 Duo 1.66GHz e memória de 2GB. Dessa forma, a redução de tempo e esforço com a aplicação do protótipo é consideravelmente alta.

Uma das limitações do *plug-in* que implementa UbiFEX-Simulation é o fato de todas as regras de contexto serem consideradas como obrigatórias. Além disso, não foi desenvolvido nenhum mecanismo para calcular o valor de uma informação de contexto composta, com base nas informações de contexto atômicas que a compõe, e apenas informações de contexto numéricas e booleanas podem ser simuladas. A definição dessa expressão de composição não foi incluída na implementação de UbiFEX-Notation.

A configuração dos produtos é verificada para os diferentes cenários, com base em uma configuração inicial determinada pelo usuário. Para garantir uma verificação mais abrangente, algumas melhorias poderiam ser implementadas no *plug-in,* para possibilitar a execução do processo de verificação de forma recursiva, fazendo com que cada configuração gerada se tornasse uma nova configuração inicial. Dessa forma, seria possível ter uma visão geral de todas as configurações derivadas. Além disso, poderia ser implementado um mecanismo para representar a relação entre os valores das informações de contexto, como forma de minimizar a quantidade de cenários simulados.

# Capítulo 7 – Conclusão

## 7.1 Epílogo

O paradigma de linha de produtos de software tem como proposta a construção sistemática de software baseada em uma família de produtos, apoiando o desenvolvimento com e para reutilização. Entre os seus benefícios estão um alto nível de reúso, redução do *time-to-market*, entre outros.

A computação sensível ao contexto pode explorar esse paradigma de forma que o reúso ocorra de maneira sistemática e que novos produtos possam ser facilmente configurados. No entanto, novos desafios podem ser identificados no desenvolvimento de linha de produtos sensíveis ao contexto. Esses desafios estão relacionados principalmente aos aspectos de sensibilidade ao contexto e reconfiguração dinâmica envolvidos nessa classe de produtos.

As informações de contexto são elementos chave na definição de sistemas sensíveis ao contexto. Dessa forma, uma das primeiras preocupações consiste na identificação e na representação dessas informações desde os estágios iniciais de uma LPS, além da influência das mesmas na configuração dos produtos em tempo de execução.

A modelagem de características é uma das primeiras atividades no desenvolvimento de uma LPS e busca capturar os pontos comuns e variáveis existentes entre os produtos. O modelo de características, gerado a partir dessa atividade, é utilizado como ponto de partida para o recorte necessário à instanciação de novos produtos. Além disso, permite uniformizar o entendimento entre todos os envolvidos no processo de desenvolvimento.

Neste trabalho de pesquisa, foi apresentada a abordagem UbiFEX, que visa apoiar a modelagem de características de linha de produtos de software sensíveis ao contexto. Essa abordagem foi dividida em dois componentes: UbiFEX-Notation, uma notação para a representação dos conceitos relacionados à sensibilidade ao contexto; e UbiFEX-Simulation, um mecanismo para a verificação da configuração dos produtos em relação às variações de contexto, com base em modelos de características construídos utilizando UbiFEX-Notation.

Um estudo preliminar foi realizado para avaliar o processo de verificação proposto em UbiFEX-Simulation. Esse estudo contribuiu para a melhoria da abordagem e motivou o desenvolvimento de um protótipo para apoiar a aplicação da abordagem. Esse protótipo foi desenvolvido no contexto do ambiente Odyssey, envolvendo as etapas de engenharia de domínio e engenharia de aplicação.

Neste último capítulo, são destacadas as principais contribuições deste trabalho na Seção 7.2. A Seção 7.3 apresenta as limitações identificadas em relação à abordagem proposta e ao protótipo desenvolvido. Por fim, na Seção 7.4, são discutidos alguns dos possíveis trabalhos futuros.

## 7.2 Contribuições

O trabalho de pesquisa apresentado nesta dissertação teve como objetivo principal propor uma abordagem para modelagem de características de linha de produtos de software sensíveis ao contexto. Entre as suas principais contribuições, podemos destacar:

- Definição de uma abordagem, denominada UbiFEX, para a modelagem de características de LPSSC, que inclui: (1) uma notação para a representação explícita dos conceitos relacionados à sensibilidade ao contexto em modelos de características; e (2) um mecanismo para a verificação da configuração dos produtos em relação às variações de contexto ainda em tempo de desenvolvimento. É importante destacar que a abordagem proposta é baseada no mesmo tipo de modelo comumente utilizado para a identificação de características e variabilidade em LPS;
- Avaliação preliminar do mecanismo de verificação UbiFEX-Simulation, por meio de um estudo realizado com alunos de pós-graduação. Os resultados obtidos foram utilizados para melhoria da abordagem e poderão servir como base para o planejamento de uma avaliação mais completa;
- Desenvolvimento de um protótipo que implementa a abordagem proposta no contexto do ambiente Odyssey. Sendo a notação implementada no núcleo do ambiente, como extensão da notação existente, e o mecanismo de verificação implementado como um *plug-in,* que pode ser carregado por demanda.

Podemos ainda ressaltar as seguintes contribuições secundárias:

- Estudo das áreas de Computação Sensível ao Contexto e Linha de Produtos de Software, buscando identificar as oportunidades de pesquisa existentes;
- Identificação das extensões necessárias para possibilitar a representação em modelos de características de sistemas sensíveis ao contexto;
- Identificação dos diferentes tipos de inconsistências possíveis na reconfiguração dos produtos em relação às variações de contexto com base em modelos de características.

## 7.3 Limitações

Algumas limitações deste trabalho foram identificadas a partir de uma análise crítica da abordagem proposta e do protótipo desenvolvido. As principais são listadas a seguir:

- UbiFEX-Simulation busca antecipar a identificação de possíveis inconsistências que só seriam percebidas em tempo de execução. No entanto, a aplicação desse mecanismo não garante que o produto não irá falhar, pois cenários não previstos podem ocorrer durante a execução dos produtos;
- A aplicação de UbiFEX-Simulation é restrita a modelo de características desenvolvidos utilizando UbiFEX-Notation;
- Apesar de ter sido construído um exemplo de modelo de características para uma LPSSC baseada em sistemas reais, não foi realizada nenhuma avaliação em relação às extensões propostas na notação UbiFEX-Notation. Dessa forma, não é possível confirmar se essa notação é satisfatória na representação dos sistemas sensíveis ao contexto;
- Não foi realizado nenhum estudo para avaliar o desempenho do protótipo desenvolvido, principalmente em relação ao *plug-in* que implementa UbiFEX-Simulation. É possível que o desempenho do *plug-in* seja influenciado pelo aumento do número de cenários a serem verificados e pela complexidade do modelo;
- A parte do protótipo relacionada à implementação de UbiFEX-Simulation, desenvolvida como *plug-in* para o ambiente Odyssey, suporta apenas a simulação de valores para informações de contexto atômicas, ou seja, que não são compostas por outras informações. Além disso, essas informações devem ser numéricas ou booleanas.

- O plug-in desenvolvido verifica a configuração com base apenas em uma configuração inicial determinada pelo usuário. No entanto, para garantir uma verificação mais abrangente, seria necessário executar o processo de forma recursiva, fazendo com que cada configuração gerada se tornasse uma nova configuração inicial.

## 7.4 Trabalhos Futuros

Ao longo do desenvolvimento deste trabalho, algumas oportunidades de melhoria, tanto da abordagem proposta quanto do protótipo desenvolvido, foram destacadas. Além disso, novas oportunidades de pesquisa foram identificadas. Os possíveis trabalhos futuros envolvem:

- Aplicação da abordagem UbiFEX e do protótipo no desenvolvimento de LPSSC reais e de maior porte. Dessa forma, será possível identificar oportunidades de evolução e melhoria da abordagem e das ferramentas propostas;
- Planejamento e execução de estudos de avaliação mais completos, que envolvam a abordagem proposta como um todo e a utilização do protótipo;
- Representação e mapeamento das extensões definidas no modelo de características para níveis de abstração mais próximos da implementação, como, por exemplo, o modelo de componentes;
- Integração da abordagem UBIFEX com mecanismos para a obtenção das informações de contexto, modeladas no ambiente de execução dos produtos, como sistemas de *middleware* e gerenciadores de sensores;
- Integração da abordagem proposta com tecnologias que suportem a reconfiguração dinâmica de produtos, como, por exemplo, OSGi (OSGI ALLIANCE, 2009);
- Construção de um ambiente de execução que suporte a reconfiguração dos produtos com base nas extensões propostas e que permita a inclusão de novas características e regras de adaptação dos produtos em tempo de execução;

# Referências Bibliográficas


ABOWD, G.D., MYNATT, E.D., 2000, "Charting past, present and future research in ubiquitous computing", *ACM Transactions on Computer-Human Interaction*, v. 7, n. 1 (March), pp. 29-58.

ALMEIDA, D.R., BAPTISTA, C.S., ANDRADE, F.G., 2006, "Using Ontologies in Context-Aware Applications". In: *Proceedings of the 17th International Conference on Database and Expert Systems Applications (DEXA'06)*, pp. 349-353, Krakow, Poland, September.

ALVES, V., JÚNIOR, P.M., COLE, L., *et al.*, 2005, "Extracting and Evolving Mobile Games Product Lines". In: *Proceedings of the 9th International Software Product Line Conference (SPLC'05)*, pp. 70-81, Rennes, France, September.

ARAUJO, R.B., 2003, "Computação Ubíqua: Princípios, Tecnologias e Desafios". In: *XXI Simpósio Brasileiro de Redes de Computadores*, pp. 1-71, Natal, Rio Grande do Norte, Brasil, Maio.

BALDAUF, M., DUSTDAR, S., ROSENBERG, F., 2007, *A Survey on Context-Aware Systems*, Relatório Técnico TUV-1841-2004-24, Information Systems Institute of the Technical University of Vienna.

BECKER, C., HANDTE, M., SCHIELE, G., 2004, "PCOM - A Component System for Pervasive Computing". In: *Proceedings of the Second IEEE International Conference on Pervasive Computing and Communications* , pp. 67-76, Orlando, USA, March.

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.

BENAVIDES, D., TRINIDAD, P., RUIZ-CORTÉS, A., 2005, "Automated Reasoning on Feature Models". In: *17th Conference on Advanced Information Systems Engineering (CAiSE'05)* , pp. 491-503, Porto, Portugal, June.

BLOIS, A., 2006, *Uma abordagem de Projeto Arquitetural baseado em Componentes no Contexto de Engenharia de Domínio*, Tese de D.Sc., COPPE Sistemas, UFRJ, Rio de Janeiro, Brasil.

BOSCH, J., 2004, "Software Variability Management". In: *Proceedings of the 26th international Conference on Software Engineering*, pp. 720-721, Scotland, UK, May.

BROWN, P., BOVEY, J., CHEN, X., 1997, "Context-Aware Applications: From the Laboratory to the Marketplace", *IEEE Personal Communications*, v. 4, n. 5 (October), pp. 58-64.

BROWN, P.J., 1996, "The Stick-e Document: a Framework for Creating Context-Aware Applications". In: *Proceedings of the Electronic Publishing*, pp. 259-272, Palo Alto, September.

BRÉZILLON, P., 2005, "Task Realization Models in Contextual Graphs". In: *CONTEXT 2005*, pp. 55-68, Paris, France, .

BUCHHOLZ, S., HAMANN, T., HUBSCH, G., 2004, "Comprehensive Structured Context Profiles (CSCP): Design and Experiences". In: *Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops*, pp. 43-47, Florida, EUA, March.

BULCÃO NETO, R., PIMENTEL, M., 2005, "Toward a Domain-Independent Semantic Model for Context-Aware Computing". In: *Proceedings of the Third Latin American Web Congress*, pp. 61-70, Buenos Aires, Argentina, November.

CHEN, G., KOTZ, D., 2000, *A Survey of Context-Aware Mobile Computing Research*, Relatório Técnico TR2000-381, Dept. of Computer Science, Dartmouth College.

CHEN, H., 2004, *An Intelligent Broker Architecture for Pervasive Context-Aware Systems*, Ph.D Thesis. Department of Computer Science, University of Maryland, Baltimore County, USA.

CLEMENTS, P., NORTHROP, L., 2002, *Software Product Lines: Practices and Patterns*, Addison-Wesley.

COALLIER, F., CHAMPAGNE, R., 2005, "A Product Line engineering practices model", *Science of Computer Programming*, v. 57, n. 1 (July), pp. 73-87.

COUTAZ, J., CROWLEY, J.L., DOBSON, S., *et al.*, 2005, "Context is Key", *Communications of the ACM*, v. 48, n. 3 (March), pp. 49-53.

CZARNECKI, K., HELSEN, S., EISENECKER, U., 2004, "Staged Configuration Using Feature Models". In: *Third Software Product Line Conference (SPLC 2004)*, pp. 266-283, Boston, MA, USA, August.

DANTAS, A.R., CORREA, A.L., WERNER, C., 2001, "Oráculo: Um Sistema de Críticas para a UML". In: *XV Simposio Brasileiro de Engenharia de Software,*

*Caderno de Ferramentas*, pp. 398-403, Rio de Janeiro, Rio de Janeiro, Brasil, Outubro.

DASHOFY, E.M., VAN DER HOEK, A., TAYLOR, R.N., 2002, "An Infrastructure for the Rapid Development of XML-Based Architecture Description Languages". In: *Proceedings of the 24th International Conference on Software Engineering*, pp. 266-276, Orlando, Florida, USA, May.

DEY, A., 2001, "Understanding and Using Context", *Personal and Ubiquitous Computing*, v. 5, n. 1 (February), pp. 4-7.

DURSCKI, R.C., SPINOLA, M.M., BURNETT, R.C., *et al.*, 2004, "Linhas de Produto de Software: riscos e vantagens de sua implantação". In: *VI Simpósio Internacional de Melhoria de Processos de Software*, pp. 155-166, São Paulo, Brasil, Novembro.

FERNANDES, P., PRUDÊNCIO, J.G., MARINHO, A., *et al.*, 2007, "Carga Dinâmica de Componentes via Biblioteca Brechó". In: *Simpósio Brasileiro de Componentes, Arquiteturas e Reutilização de Software (SBCARS 2007), Sessão de Ferramentas*, pp. 1-8, Campinas, São Paulo, Brasil, Agosto.

FERNANDES, P., WERNER, C., 2008a, "UbiFEX: Modelagem de Características para Linhas de Produtos de Software Sensíveis ao Contexto". In: *Simpósio Brasileiro de Componentes, Arquiteturas e Reutilização de Software (SBCARS 2008), Sessão de Ferramentas*, pp. 25-32, Porto Alegre, Rio Grande do Sul, Brasil, Agosto.

FERNANDES, P., WERNER, C., 2008b, "UbiFEX: Modeling Context-Aware Software Product Lines". In: *2nd International Workshop on Dynamic Software Product Lines (DSPL 2008)*, pp. 3-8, Limerick, Ireland, September.

FERNANDES, P., WERNER, C., MURTA, L., 2008, "Feature Modeling for Context-Aware Software Product Lines". In: *Twentieth International Conference on Software Engineering and Knowledge Engineering (SEKE'08)*, pp. 758-763, Redwood City, San Francisco Bay, USA, July.

FRAKES, W.B., KANG, K., 2005, "Software Reuse Research: Status and Future", *IEEE Transactions on Software Engineering*, v. 31, n. 7 (July), pp. 529-536.

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

GARLAN, D., SCHMERL, B., 2001, "Component-based Software Engineering in Pervasive Computing Environments". In: *Proceedings of the 4th ICSE*

*Workshop on Component-Based Software Engineering, Component Certification and System Prediction*, pp. 1-4, Toronto, Canada, May.

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, Julho.

GOMAA, H., 2004, *Software Product Lines with UML: From Use Cases to Pattern-Based Software Architectures*, Addison-Welsey.

GRUBER, T.R., 1993, "A Translation Approach to Portable Ontology Specifications", *Knowledge Acquisition*, v. 5, n. 2 (June), pp. 199-220.

GRÜN, C., WERTHNER, H., PRÖLL, B., *et al.*, 2008, "Assisting Tourists on the Move - An Evaluation of Mobile Tourist Guides". In: *7th International Conference on Mobile Business*, pp. 171-180, Barcelona, Spain, July.

HALLSTEINSEN, S., STAV, E., SOLBERG, A., *et al.*, 2006, "Using Product Line Techniques to Build Adaptive Systems". In: *Proceedings of the 10th International on Software Product Line Conference*, pp. 141-150, Washington, DC, USA, August .

HARTMANN, H., TREW, T., 2008, "Using Feature Diagrams with Context Variability to Model Multiple Product Lines for Software Supply Chains". In: *Proceedings of the 12th International Software Product Line Conference*, pp. 12-21, Limerick, Ireland, September.

HENRICKSEN, K., INDULSKA, J., 2004, "A Software Engineering Framework for Context-Aware Pervasive Computing". In: *Proceedings of the Second IEEE international Conference on Pervasive Computing and Communications*, pp. 77-86, Florida, USA, March.

HENRICKSEN, K., INDULSKA, J., 2006, "Developing Context-Aware Pervasive Computing Applications: Models and Approach", *Pervasive and Mobile Computing*, v. 2, n. 1 (February), pp. 37-64.

HENRICKSEN, K., INDULSKA, J., RAKOTONIRAINY, A., 2002, "Modeling Context Information in Pervasive Computing Systems". In: *Proceedings of the First International Conference on Pervasive Computing*, pp. 167-180, Zurich, Switzerland, August.

HOFER, T., SCHWINGER, W., PICHLER, M., *et al.*, 2003, "Context-Awareness on Mobile Devices - the Hydrogen Approach". In: *Proceedings of the 36th Hawaii International Conference on System Sciences* , p. 292, Hawaii, USA, January.

HULL, R., NEAVES, P., BEDFORD-ROBERTS, J., 1997, "Towards Situated Computing". In: *Proceedings of the 1st IEEE international Symposium on Wearable Computers*, pp. 146-153, Cambridge, MA, USA, October.

KANG, K., LEE, J., DONOHOE, P., 2002, "Feature-Oriented Product Line Engineering", *IEEE Software*, v. 19, n. 4 (July/August), pp. 58-65.

KANG, K., COHEN, S., HESS, J., *et al.*, 1990, *Feature-Oriented Domain Analysis (FODA) Feasibility Study*, Relatório Técnico CMU/SEI-90-TR-021, Software Engineering Institute.

KATSIRI, E., MYCROFT, A., 2003, "Knowledge Representation and Scalable Abstract Reasoning for Sentient Computing Using First-Order Logic". In: *First Workshop on Challenges and Novel Applications for Automated Reasoning*, pp. 73-93, Miami, USA, July.

KICZALES, G., LAMPING, J., MENDHEKAR, A., *et al.*, 1997, "Aspect-Oriented Programming". In: *Proceedings European Conference on Object-Oriented Programming*, pp. 220-242, Jyväskylä, Finland, June.

LAHIRI, S., 2005, *RFID Sourcebook*, IBM Press.

LARA, E., WALLACH, D.S., ZWAENEPOEL, W., 2001, "Puppeteer: Component-Based Adaptation for Mobile Computing". In: *Proceedings of the 3rd Conference on USENIX Symposium on Internet Technologies and Systems*, p. 14, San Francisco, California, USA, March.

LEE, J., MUTHIG, D., 2006, "Feature-Oriented Variability Management in Product Line Engineering ", *Communications of the ACM*, v. 49, n. 12 (December), pp. 55-59.

LOKE, S., 2006, *Context-Aware Pervasive Systems: Architectures for a New Breed of Applications*, Auerbach Publications.

LYYTINEN, K., YOO, Y., 2002, "Issues and Challenges in Ubiquitous Computing", *Communications of the ACM* , v. 45, n. 12 (December), pp. 62-65.

MAIA, N.E.N., BLOIS, A.P.B., WERNER, C., 2005, "Odyssey-MDA: Uma Ferramenta para Transformação de Modelos UML". In: *XIX Simpósio Brasileiro de Engenharia de Software, Sessão de Ferramentas*, pp. 51-56, Uberlândia, Minas Gerais, Brasil, Outubro.

MASSEN, T.V.D., LICHTER, H., 2002, "Modeling Variability by UML Use Case Diagrams". In: *Proceedings of the International Workshop on Requirements Engineering for Product Lines (REPL'02)*, pp. 19-31, Essen, Germany, September.

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

MCKINLEY, P.K., SADJADI, S.M., KASTEN, E.P., *et al.*, 2004, "Composing Adaptive Software", *IEEE Computer*, v. 37, n. 7 (July), pp. 56-64.

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

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*, pp. 241-259, Curitiba, Paraná, Brasil, Agosto.

ODYSSEY, 2009, "Projeto Odyssey". In: http://reuse.cos.ufrj.br/odyssey, acessado em 10/01/2009.

OLIVEIRA, R.F., 2006, *Formalização e Verificação de Consistência na Representação de Variabilidades*, Dissertação de M.Sc., COPPE Sistemas, UFRJ, Rio de Janeiro, Brasil.

OSGI ALLIANCE, 2009, "OSGi - The Dynamic Module System for Java". In: http://www.osgi.org, acessado em 23/01/2009.

PARNAS, D., 1976, "On the Design and Development of Program Families", *IEEE Transactions on Software Engineering*, v. SE-2, n. 1 (March), pp. 1-9.

PEREIRA, F.M.Q., VALENTE, M.T.O., BIGONHA, R., *et al.*, 2006, "Arcademis: a Framework for Object-Oriented Communication Middleware Development", *ACM Software-Practice & Experience*, v. 36, n. 5 (April), pp. 495-512.

POHL, K., METZGER, A., 2006, "Variability Management in Software Product Line Engineering". In: *Proceeding of the 28th International Conference on Software Engineering*, pp. 1049-1050, Shanghai, China, May.

PREKOP, P., BURNETT, M., 2003, "Activities, Context and Ubiquitous Computing", *Computer Communications*, v. 26, n. 11 (July), pp. 1168-1176.

PRIETO-DIAZ, R., ARANGO, G., 1991, *Domain Analysis and Software Systems Modeling*, IEEE Computer Society Press.

ROCHA, L.S., 2007, *AdaptiveRME e AspectCompose: Um Middleware Adaptativo e um Processo de Composição Orientado a Aspectos para o Desenvolvimento de Software Móvel e Ubíquo*, Dissertação de M.Sc., Departamento de Computação, Universidade Federal do Ceará (UFC), Ceará, Brasil.

ROMÁN, M., HESS, C., CERQUEIRA, R., *et al.*, 2002, "A Middleware Infrastructure for Active Spaces", *IEEE Pervasive Computing*, v. 1, n. 4 (October), pp. 74-83.

SACRAMENTO, V., ENDLER, M., RUBINSZTEJN, H., *et al.*, 2004, "MoCA: A Middleware for Developing Collaborative Applications for Mobile Users", *IEEE Distributed Systems Online*, v. 5, n. 10 (oCTOBER), p. 2.

SCHILIT, B., THEIMER, M., 1994, "Disseminating Active Map Information to Mobile Hosts", *IEEE Network*, v. 8, n. 5 (September/October), pp. 22-32.

SHULL, F., CARVER, J., TRAVASSOS, G.H., 2001, "An Empirical Methodology for Introducing Software Processes", *ACM SIGSOFT Software Engineering Notes*, v. 26, n. 5 (September), pp. 288-296.

SIMONS, C., 2007, "CMP: A UML Context Modeling Profile for Mobile Distributed Systems". In: *Proceedings of the 40th Annual Hawaii international Conference on System Sciences*, p. 289b, Hawaii, EUA, January.

SOUZA, D., BELIAN, R., SALGADO, A.C., *et al.*, 2008, "Towards a Context Ontology to Enhance Data Integration Processes". In: *4th ODBIS Workshop on Ontologies-based Techniques for DataBases in Information Systems and Knowledge Systems*, pp. 49-56, Auckland, New Zealand, August.

STRANG, T., LINNHOFF-POPIEN, C., 2004, "A Context Modeling Survey". In: *First International Workshop on Advanced Context Modelling, Reasoning And Management*, pp. 33-40, Nottingham, UK, September.

STUDER, R., BENJAMINS, V.R., FENSEL, D., 1998, "Knowledge Engineering: Principles and methods", *Data and Knowledge Engineering* , v. 25, n. 1 (March), pp. 161-197.

SUGUMARAN, V., PARK, S., KANG, K.C., 2006, "Software Product Line Engineering", *Communications of the ACM*, v. 49, n. 12 (December), pp. 29-32.

SUN, 2009, "Java Technology". In: http://www.java.sun.com, acessado em 10/01/2009.

TEIXEIRA, E.N., WERNER, C., VASCONCELOS, A.P.V., 2008, "Flexibilização para Representação de Características no Ambiente Odyssey". In: *II Simpósio*

*Brasileiro de Componentes, Arquiteturas e Reutilização de Software (SBCARS 2008), Sessão de Ferramentas*, pp. 9-16, Porto Alegre, Rio Grande do Sul, Brasil, Agosto.

VAN DER HOEK, A., 2004, "Design-Time Product Line Architectures for Any-Time Variability", *Science of Computer Programming*, v. 53, n. 3 (December), pp. 285-304.

VAN DER LINDEN, F., SCHMID, K., ROMMES, E., 2007, *Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering*, Springer-Verlag New York, Inc.

VAN GURP, J., BOSCH, J., SVAHNBERG, M., 2001, "On the Notion of Variability in Software Product Lines". In: *Proceedings of the Working IEEE/IFIP Conference on Software Architecture*, pp. 45-54, Amsterdam, Netherlands, August.

VERONESE, G., CORREA, A., WERNER, C., *et al.*, 2002, "ARES: Uma Ferramenta de Engenharia Reversa Java-UML". In: *VI Simpósio Brasileiro de Engenharia de Software, Sessão de Ferramentas*, pp. 347-352, Gramado, Rio Grande do Sul, Brasil, Outubro.

VIEIRA, V., 2008, *CEManTIKA: A Domain-Independent Framework for Designing Context-Sensitive Systems*, Tese de D.Sc., Centro de Informática, UFPE, Pernambuco, Brasil.

WAGELAAR, D., 2005, "Towards Context-Aware Feature Modelling using Ontologies". In: *MoDELS 2005 Workshop on MDD for Software Product Lines: Fact or Fiction?*, , Montego Bay, Jamaica, October.

WEISER, M., 1991, "The Computer for the 21st Century", *Scientific American*, v. 265, n. 3 (September), pp. 94-104.

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*, Kluwer Academic Publishers.

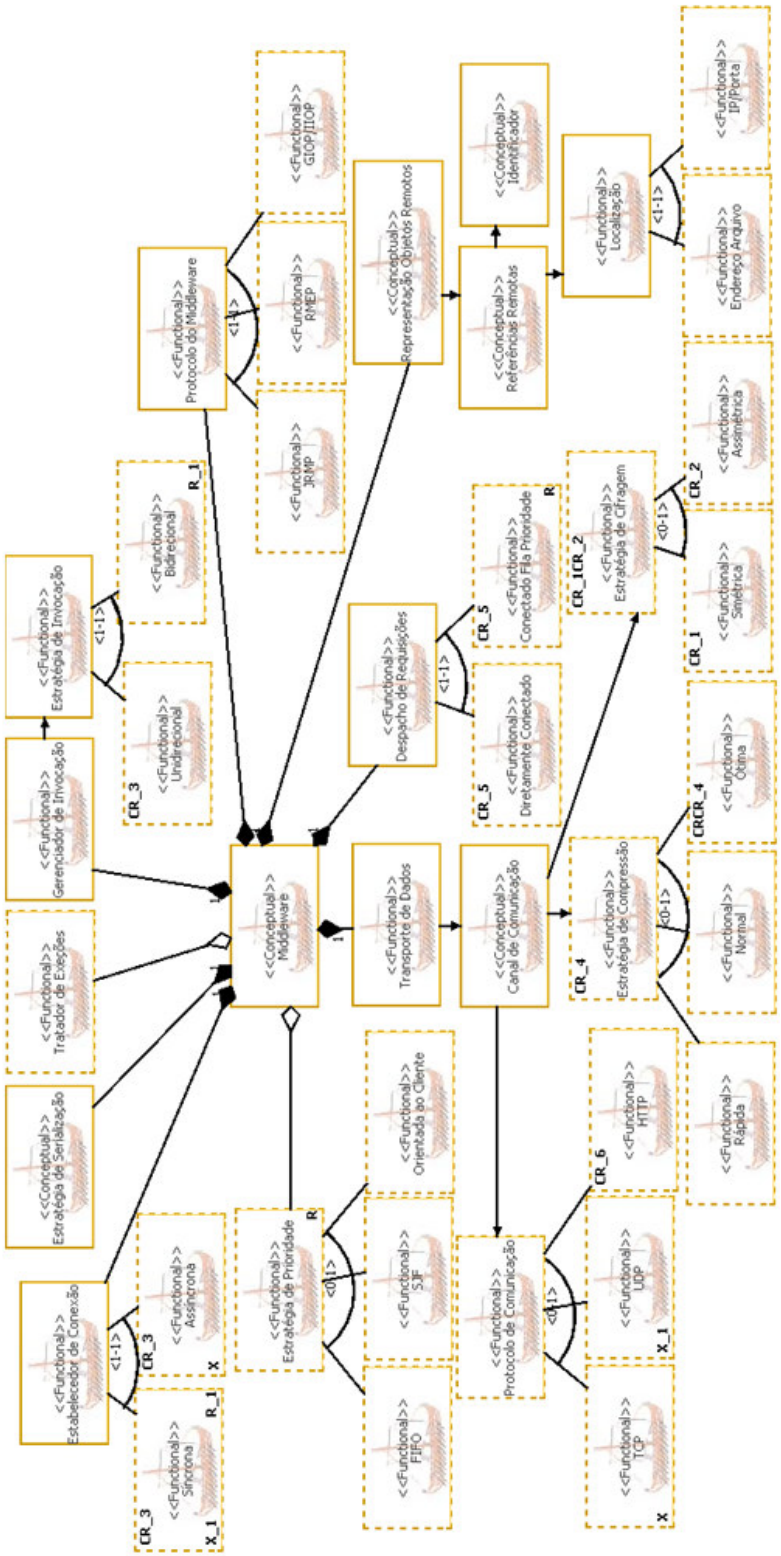
YE, J., COYLE, L., DOBSON, S., *et al.*, 2008, "Representing and Manipulating Situation Hierarchies using Situation Lattices", *Revue d'Intelligence Artificielle*, v. 22, n. 5 (October), pp. 647-667.

YOUNG, T.J., 2005, *Using AspectJ to Build a Software Product Line for Mobile Devices*, Dissertação de M.Sc., The University of British Columbia, Vancouver, Canada.

ZIMMER, T., 2006, "QoC: Quality of Context – Improving the Performance of Context-Aware Applications". In: *Adjunct Proceedings of 4th International Conference on Pervasive Computing*, pp. 209-214, Dublin, Ireland, May.

# Apêndice I

## Modelo Completo de AdaptiveRME



# Apêndice II

## Formulário de Consentimento

**Estudo**

Este estudo visa caracterizar a aplicação do mecanismo de verificação de consistência da configuração de produtos da abordagem UbiFEX no apoio ao desenvolvimento de linha de produtos de software sensíveis ao contexto.

**Idade**

Eu declaro ter mais de 18 anos de idade e concordar em participar de um estudo conduzido por Paula Cibele Cavalcante Fernandes na Universidade Federal do Rio de Janeiro.

**Procedimento**

Este estudo acontecerá em uma única sessão, que incluirá a análise da consistência da configuração de um produto em diferentes cenários de execução,com base no seu modelo de característica. Eu entendo que, uma vez o experimento tenha terminado, os trabalhos que desenvolvi serão estudados visando analisar a aplicação dos procedimentos e técnicas propostos.

**Confidencialidade**

Toda informação coletada neste estudo é confidencial, e meu nome não será divulgado. 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 e liberdade de desistência**

Eu entendo que os benefícios que receberei deste estudo são limitados ao aprendizado do material que é distribuído e apresentado. Eu entendo que sou livre para realizar perguntas a qualquer momento, solicitar que qualquer informação relacionada a minha pessoa não seja incluída no estudo ou comunicar minha desistência de participação. 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.

**Pesquisador responsável**

Paula Cibele Cavalcante Fernandes
Programa de Engenharia de Sistemas e Computação - COPPE/UFRJ

**Professor responsável**

Profª. Cláudia Maria Lima Werner
Programa de Engenharia de Sistemas e Computação - COPPE/UFRJ

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

**Assinatura: _____**

**Data**:_____

# Apêndice III

## Formulário de Caracterização do Participante

Este formulário contém algumas perguntas sobre sua experiência acadêmica e profissional.

### 1) Formação acadêmica

( ) Doutorado
( ) Doutorando
( ) Mestrado
( ) Mestrando
( ) Graduação
Ano de ingresso: _____ Ano de conclusão (ou previsão de conclusão): _____

### 2) Formação geral

Por favor, indique o grau de sua experiência nesta seção, seguindo a escala de 5 pontos abaixo:

*0 = nenhum*
*1 = estudei em aula ou em livro*
*2 = pratiquei em projetos em sala de aula*
*3 = usei em projetos pessoais*
*4 = usei em projetos na indústria*

| | | | | | |
|---|---|---|---|---|---|
| 2.1) Sistemas sensíveis ao contexto | 0 | 1 | 2 | 3 | 4 |
| 2.2) Modelagem de contexto | 0 | 1 | 2 | 3 | 4 |
| 2.3) Linhas de produtos de software | 0 | 1 | 2 | 3 | 4 |
| 2.4) Modelagem de características | 0 | 1 | 2 | 3 | 4 |

### 3) Experiência no domínio exemplo

Esta seção será utilizada para compreender quão familiar você é com o domínio que será utilizado para as atividades durante o experimento. Por favor, indique o grau de experiência nesta seção, seguindo a escala de 3 pontos abaixo:

*0 = Eu não tenho familiaridade com este domínio. Eu nunca fiz isto.*
*1 = Eu utilizo isto algumas vezes, mas não sou um especialista.*
*2 = Eu sou muito familiar com este domínio. Eu me sentiria confortável fazendo isto.*

| | | | |
|---|---|---|---|
| Domínio de *Middlewares* | 0 | 1 | 2 |

**Obrigada por sua colaboração!**

# Apêndice IV

## Descrição da Tarefa – Grupo 1

### Instruções

Este é um estudo de observação, por isso, sempre que possível, verbalize seus pensamentos, para que o experimentador possa melhor avaliar os resultados obtidos. Pergunte e comente tudo que achar necessário.

### Contextualização

A demanda por aplicações distribuídas tem crescido continuamente. Por exemplo, sobre a Internet são executadas várias aplicações distribuídas diferentes, desde as muito simples até outras extremamente complexas, como os sistemas bancários. Essas aplicações, em geral, são executadas em ambientes heterogêneos constituídos por computadores que podem apresentar arquiteturas e sistemas operacionais diferentes. Uma das soluções para amenizar os problemas dessa heterogeneidade é interpor entre aplicações e sistemas operacionais uma terceira camada de software, denominada *middleware*. Esta camada permite que os desenvolvedores possam dispor de uma interface de programação uniforme.

Você recebeu parte do modelo de características do produto AdaptiveRME, que foi gerado a partir de uma linha de produtos no domínio de *middlewares*. Este produto possui a capacidade de se adaptar às variações de contexto durante sua execução. As adaptações previstas estão descritas no modelo recebido.

Sua tarefa é analisar o modelo de características desse produto e verificar a consistência da configuração do mesmo em diferentes cenários de execução de AdaptiveRME. Para cada um dos cenários você recebeu uma tabela indicando os valores das informações de contexto e um formulário de identificação de inconsistências.

# Apêndice V

## Descrição da Tarefa – Grupo 2

**Instruções**

Este é um estudo de observação, por isso, sempre que possível, verbalize seus pensamentos, para que o experimentador possa melhor avaliar os resultados obtidos. Pergunte e comente tudo que achar necessário.

**Contextualização**

A demanda por aplicações distribuídas tem crescido continuamente. Por exemplo, sobre a Internet são executadas várias aplicações distribuídas diferentes, desde as muito simples até outras extremamente complexas, como os sistemas bancários. Essas aplicações, em geral, são executadas em ambientes heterogêneos constituídos por computadores que podem apresentar arquiteturas e sistemas operacionais diferentes. Uma das soluções para amenizar os problemas dessa heterogeneidade é interpor entre aplicações e sistemas operacionais uma terceira camada de software, denominada *middleware*. Esta camada permite que os desenvolvedores possam dispor de uma interface de programação uniforme.

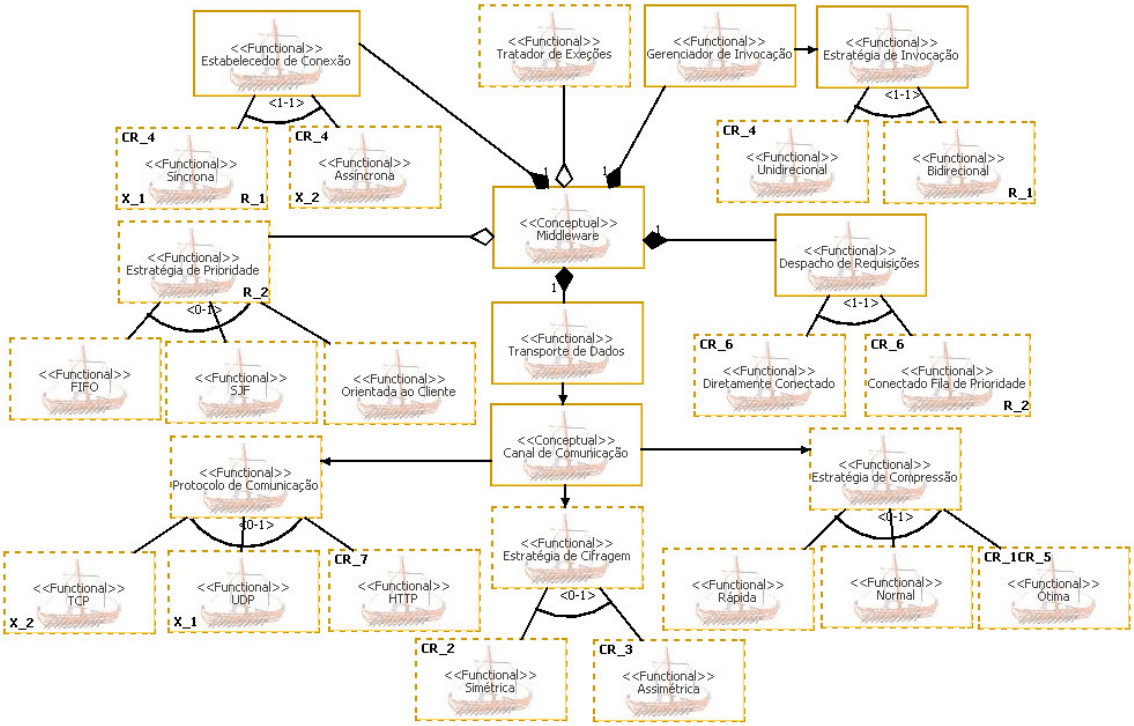
Você recebeu parte do modelo de características do produto AdaptiveRME, que foi gerado a partir de uma linha de produtos no domínio de *middlewares*. Este produto possui a capacidade de se adaptar às variações de contexto durante sua execução. As adaptações previstas estão descritas no modelo recebido.

Sua tarefa é analisar o modelo de características desse produto e verificar a consistência da configuração do mesmo em diferentes cenários de execução de AdaptiveRME utilizando o processo de verificação proposto por UbiFEX-Simulation. Para cada um dos cenários você recebeu uma tabela indicando os valores das informações de contexto e um formulário de identificação de inconsistências.

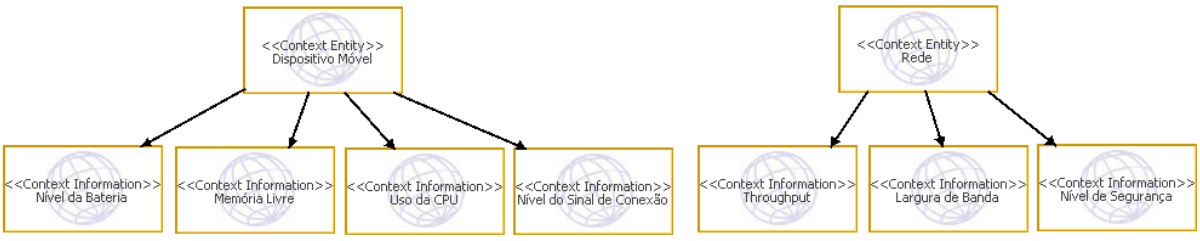
# Apêndice VI

## Descrição do Modelo Exemplo

### Modelo de Características – Produto AdaptiveRME




### Modelo de Características de Contexto – Produto AdaptiveRME



## Regras de Composição

RC1 - **R_1**: Bidirecional **requires** Síncrona

RC2 - **R_2**: Conectado Fila de Prioridade **requires** Estratégia de Prioridade

RC3 - **X_1**: Síncrona **excludes** UDP

RC4 - **X_2**: Assíncrona **excludes** TCP

## Definições de Contexto

DC1 - **Alta Carga de Processamento:** (Dispositivo Móvel.Uso da CPU > 80)

DC2 - **Bateria Baixa:** (Dispositivo Móvel.Nível da Bateria < 30)

DC3 - **Largura de Banda Limitada:** (Rede.Largura de Banda < 512)

DC4 - **Nível de Sinal de Conexão Baixo:** (Dispositivo Móvel.Nível do Sinal de Conexão <= 3)

DC5 - **Nível de Segurança Médio:** ((Rede.Nível de Segurança >= 3) AND (Rede.Nível de Segurança < 5))

DC6 - **Nível de Segurança Baixo:** (Rede.Nível de Segurança < 3)

DC7 - **Memória Normal:** (Dispositivo Móvel.Memória Livre >= 128)

DC8 - **Memória Baixa:** (Dispositivo Móvel.Memória Livre < 128)

DC9 - **Throughput Normal:** ((Rede.Throughput >= 128) AND (Rede.Largura de Banda >= 1024))

DC10 - **Throughput Baixo:** ((Rede.Throughput < 64) AND (Rede.Largura de Banda >= 1024))

## Regras de Contexto

RX1 - **CR_1**: Bateria Baixa **implies** NOT (Ótima)

RX2 - **CR_2**: Nível de Segurança Baixo **implies** (Estratégia de Cifragem AND Simétrica)

RX3 - **CR_3**: Nível de Segurança Médio **implies** (Estratégia de Cifragem AND Assimétrica)

RX4 - **CR_4**: (Nível de Sinal de Conexão Baixo AND Throughput Baixo) **implies** ((NOT (Assíncrona)) AND (Unidirecional AND Síncrona))

RX5 - **CR_5**: Throughput Baixo **implies** (Estratégia de Compressão AND Ótima)

RX6 - **CR_6**: Alta Carga de Processamento **implies** (Conectado Fila de Prioridade AND (NOT (Diretamente Conectado)))

RX7 - **CR_7**: Throughput Normal **implies** HTTP

# Apêndice VII

## Descrição dos Cenários de Execução

### Cenário 1

| Informação de Contexto | Valor |
|---|---|
| Throughput | 32Kbps |
| Nível de Segurança | 7 |
| Largura de Banda | 256Kbps |
| Nível de Bateria | 90% |
| Memória Livre | 512KB |
| Nível do Sinal de Conexão | 5 |
| Uso da CPU | 60% |

### Cenário 2

| Informação de Contexto | Valor |
|---|---|
| Throughput | 48Kbps |
| Nível de Segurança | 6 |
| Largura de Banda | 1024Kbps |
| Nível de Bateria | 20% |
| Memória Livre | 128KB |
| Nível do Sinal de Conexão | 4 |
| Uso da CPU | 45% |

### Cenário 3

| Informação de Contexto | Valor |
|---|---|
| Throughput | 56Kbps |
| Nível de Segurança | 4 |
| Largura de Banda | 512Kbps |
| Nível de Bateria | 75% |
| Memória Livre | 256KB |
| Nível do Sinal de Conexão | 5 |
| Uso da CPU | 85% |

### Cenário 4

| Informação de Contexto | Valor |
|---|---|
| Throughput | 32Kbps |
| Nível de Segurança | 7 |
| Largura de Banda | 1024Kbps |
| Nível de Bateria | 95% |
| Memória Livre | 512KB |
| Nível do Sinal de Conexão | 4 |
| Uso da CPU | 60% |

### Cenário 5

| Informação de Contexto | Valor |
|---|---|
| Throughput | 56Kbps |
| Nível de Segurança | 5 |
| Largura de Banda | 2048Kbps |
| Nível de Bateria | 70% |
| Memória Livre | 64KB |
| Nível do Sinal de Conexão | 2 |
| Uso da CPU | 60% |

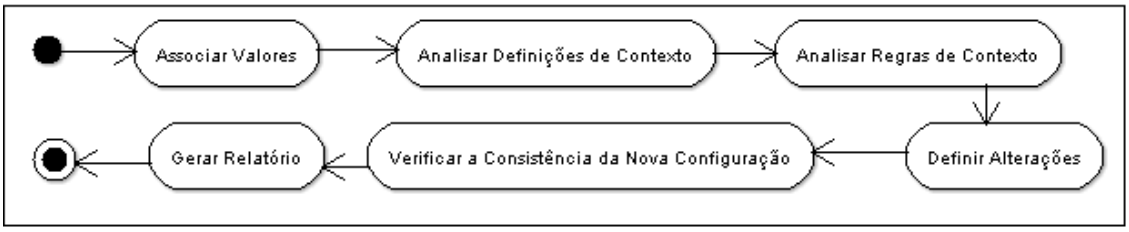
### Cenário 6

| Informação de Contexto | Valor |
|---|---|
| Throughput | 128Kbps |
| Nível de Segurança | 2 |
| Largura de Banda | 1024Kbps |
| Nível de Bateria | 80% |
| Memória Livre | 32KB |
| Nível do Sinal de Conexão | 7 |
| Uso da CPU | 50% |

# Apêndice VIII

## Descrição do Processo de Verificação

O processo de verificação da consistência da configuração dos produtos em relação às variações de contexto proposto por UbiFEX-Simulation é apresentado na figura abaixo. Cada uma das atividades é descrita a seguir:




- **Associar valores**
  - ✓ Associar valores às informações de contexto para o cenário atual.
- **Analisar definições de contexto**
  - ✓ Analisar as definições de contexto ativas com base nos valores do cenário atual.
- **Analisar regras de contexto**
  - ✓ Analisar as regras de contexto que serão executadas com base nas definições de contexto ativas.
- **Definir alterações**
  - ✓ Montar dois conjuntos: características **adicionadas** e **removidas** com base nas regras de contexto executadas, indicando que regra inclui ou removeu determinada característica.
- **Verificar a consistência da nova configuração**
  - ✓ Se os conjuntos forem vazios, a nova configuração gerada é igual à configuração inicial, então, não ocorreu nenhuma inconsistência.
  - ✓ Se não:
    - Verifique se alguma característica está presente em ambos os conjuntos
      - Se sim → **INCONSISTÊNCIA** entre regras de contexto. **Parar a verificação!**
    - Para cada elemento do conjunto de características **removidas**, verifique:
      - Faz parte do conseqüente de alguma regra de composição inclusiva (*requires*)? Se sim, verifique se a regra não está sendo quebrada, levando em consideração todas as alterações definidas.
        - Se alguma regra foi quebrada → **INCONSISTÊNCIA** entre regras de contexto e regras de composição.

- Para cada elemento do conjunto de características **adicionadas**, verifique:
  - Faz parte de alguma regra de composição exclusiva (*excludes*) ou do antecedente de alguma regra de composição inclusiva (*requires*)? Se sim, verifique se a regra não está sendo quebrada, levando em consideração todas as alterações definidas.
    - Se alguma regra foi quebrada → **INCONSISTÊNCIA** entre regras de contexto e regras de composição.
  - É uma variante? Se sim, verifique se a cardinalidade do grupo a qual ela pertence está sendo respeitada, levando em consideração todas as alterações definidas.
    - Se a cardinalidade não foi respeitada → **INCONSISTÊNCIA** de cardinalidade.

➢ **Gerar relatório**

✓ Preencha o questionário com as inconsistências encontradas.

# Apêndice IX

## Formulário de Identificação de Inconsistências

### Identificação de Inconsistências

Utilizando o modelo de características do produto AdaptiveRME e os cenários apresentados, identifique possíveis inconsistências quanto às restrições definidas no modelo que podem ocorrer em cada um dos cenários. Preencha a tabela abaixo indicando se existe ou não inconsistência, uma breve descrição da mesma e os elementos envolvidos, sejam eles regras ou características. Utilize a numeração apresentada no documento de descrição do modelo exemplo para facilitar a identificação das regras.

| Cenário | Inconsistência? (S/N) | Descrição das Inconsistências | Elementos Envolvidos |
|---|---|---|---|
| **1** | | | |
| **2** | | | |
| **3** | | | |
| **4** | | | |
| **5** | | | |
| **6** | | | |

# Apêndice X

## Formulário de Apoio à Aplicação do Processo

| Cenário | Definições de Contexto Ativas | | Regras de Contexto Executadas | | Características Adicionadas | Características Removidas |
|---|---|---|---|---|---|---|
| | DC1 | | RX1 | | | |
| | DC2 | | RX2 | | | |
| | DC3 | | RX3 | | | |
| | DC4 | | RX4 | | | |
| | DC5 | | RX5 | | | |
| | DC6 | | RX6 | | | |
| | DC7 | | RX7 | | | |
| | DC8 | | | | | |
| | DC9 | | | | | |
| | DC10 | | | | | |

# Apêndice XI

## Formulário de Avaliação – Grupo 1

| 1) Qual estratégia você utilizou para a identificação de inconsistências? Descreva as etapas seguidas. |
|---|
| |
| |
| |
| |
| |

| 2) Você sentiu dificuldades na realização das tarefas? Especifique. | □ Sim □ Não<br>□ Parcialmente |
|---|---|
| | |
| | |
| | |

| 3) Você acredita que essas inconsistências seriam identificadas antes da execução do produto sem a realização de uma tarefa de verificação? Justifique. | □ Sim □ Não<br>□ Parcialmente |
|---|---|
| | |
| | |
| | |

| 4) Você considera que a utilização de um processo como guia facilitaria sua tarefa? Justifique. | □ Sim □ Não<br>□ Parcialmente |
|---|---|
| | |
| | |
| | |

| 5) Você acredita que é viável a identificação manual de inconsistências na configuração de sistemas mais complexos? Justifique. | □ Sim □ Não<br>□ Parcialmente |
|---|---|
| | |
| | |
| | |

**Obrigada por sua colaboração!**

# Apêndice XII

## Formulário de Avaliação – Grupo 2

| 1) Você considera que a utilização do processo proposto por UbiFEX-Simulation auxiliou na realização das tarefas? Justifique. | □ Sim □ Não<br>□ Parcialmente |
|---|---|
| | |
| | |
| | |

| 2) Você sentiu dificuldades na realização das tarefas? Especifique. | □ Sim □ Não<br>□ Parcialmente |
|---|---|
| | |
| | |
| | |

| 3) Quais sugestões você teria para melhorar o processo? |
|---|
| |
| |
| |

| 4) Você acredita que essas inconsistências seriam identificadas antes da execução do produto sem a realização de uma tarefa de verificação? Justifique. | □ Sim □ Não<br>□ Parcialmente |
|---|---|
| | |
| | |
| | |

| 5) Você acredita que é viável a identificação manual de inconsistência na configuração de sistemas mais complexos? Justifique. | □ Sim □ Não<br>□ Parcialmente |
|---|---|
| | |
| | |
| | |

**Obrigada por sua colaboração!**