UMA ABORDAGEM DE PROJETO ARQUITETURAL BASEADO EM COMPONENTES NO CONTEXTO DE ENGENHARIA DE DOMÍNIO

Ana Paula Terra Bacelo Blois

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

Aprovada por:

Profa. Cláudia Maria Lima Werner, D.Sc.

Profa. Karin Becker, Ph.D.

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

Prof. Guilherme Horta Travassos, D.Sc.

Profa. Itana Maria de Souza Gimenes, Ph.D.

Prof. Toacy Cavalcante de Oliveira, D.Sc.

RIO DE JANEIRO, RJ – BRASIL

ABRIL DE 2006

BLOIS, ANA PAULA TERRA BACELO

Uma abordagem de Projeto Arquitetural baseado em Componentes no Contexto de Engenharia de Domínio [Rio de Janeiro] 2006

XII, 207 p., 29,7 cm (COPPE/UFRJ, D.Sc., Engenharia de Sistemas e Computação, 2006)

Tese – Universidade Federal do Rio de Janeiro, COPPE

1. Arquitetura de Componentes
2. Desenvolvimento Baseado em Componentes
3. Engenharia de Domínio

I. COPPE/UFRJ II. Título (série)

À Deus.

AGRADECIMENTOS

A Deus e a Virgem Maria. A força da Oração me manteve firme e perseverante.

Aos meus pais, Asdrubal e Maria Enza pelos incentivos, valores ensinados, telefonemas diários e pela confiança sempre creditada a mim. Um agradecimento especial a minha querida mãe que, junto de Deus, deve estar feliz por este momento.

Ao meu marido Henrique, por compreender o meu afastamento durante os últimos anos.

Ao meu irmão André, afilhados, familiares e amigos por compreenderem minha ausência e pelas palavras de carinho e confiança.

À profa. Cláudia Werner, minha orientadora, que me acolheu num momento delicado, me ajudou a trilhar o caminho desta pesquisa, compreendeu meus períodos de ausência e que muito me ajudou no cumprimento de prazos nem sempre viáveis. Pelos elogios, carinhos e críticas construtivas que hoje me fazem uma pessoa melhor.

À profa. Karin Becker, minha co-orientadora, com a qual trabalho desde o Mestrado. Pelas críticas construtivas e pelas palavras de otimismo.

Ao prof. Guilherme Horta Travassos por ter aceito participar desta banca e pelas orientações nas atividades de experimentação desta Tese.

Aos prof. Toacy Cavalcante de Oliveira, Itana Maria de Souza Gimenes e Marta Lima de Queirós Mattoso por terem aceito participar desta banca.

Aos amigos do grupo de reutilização de software, em especial ao Alexandre Correa, Alexandre Dantas, Aline Vasconcelos, Carlos Melo Jr, Cristine Dantas, Guilherme Kummel, Isabella Almeida, Leonardo Murta, Luiz Gustavo, Marco Lopes, Marco Mangan e Rafael Cepêda, pela ajuda e amizade. Aos parceiros diretos desta pesquisa, Artur Barbalho de Souza, Natanael Maia e Regiane Oliveira, pela dedicação e amizade.

As minhas amigas gaúchas Adriana Beiler, Ana Benso, Denise Virti, Mara Lúcia Carneiro, Márcia Campos e Milene Silveira, pelas mensagens de otimismo e pelas informações sobre o Pampa.

À PUCRS e a CAPES, pelo apoio financeiro ao desenvolvimento deste trabalho.

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

UMA ABORDAGEM DE PROJETO ARQUITETURAL BASEADO EM COMPONENTES NO CONTEXTO DE ENGENHARIA DE DOMÍNIO

Ana Paula Terra Bacelo Blois

Abril / 2006

Orientadores: Cláudia Maria Lima Werner e
Karin Becker

Programa: Engenharia de Sistemas e Computação

A reutilização de software é considerada uma forma eficiente de aumentar a produtividade e diminuir custos. A Engenharia de Domínio (ED) e o Desenvolvimento Baseado em Componentes (DBC) são exemplos de abordagens de apoio à reutilização. A maioria dos métodos de ED não privilegia o apoio às atividades de projeto de domínio e tampouco consideram a variabilidade dos requisitos inerentes ao domínio em todas as suas fases do ciclo de vida. Já os métodos de DBC se propõem a oferecer sistemáticas para o desenvolvimento de componentes reutilizáveis, embora, na prática, o apoio à reutilização seja limitado. Esta tese propõe uma abordagem de projeto arquitetural baseada em componentes, num contexto de ED, a qual fornece apoio à modelagem de variabilidades durante todo o ciclo de vida do software, auxiliando na especificação consistente de uma Arquitetura de Referência (AR) de domínio. A abordagem, denominada DECOM, propõe um processo de ED, combinando os princípios inerentes aos métodos de ED e DBC, com maior foco nas atividades de projeto. É utilizada uma notação para representar variabilidades em modelos de domínio e são propostas heurísticas que auxiliam na manutenção da consistência dos diferentes modelos. São propostos critérios para apoio à especificação de elementos arquiteturais, os quais compõe uma AR. É também utilizada uma abordagem de transformação de modelos para tecnologias de componentes específicas. Finalmente, estudos de observação foram desenvolvidos para avaliar algumas das atividades propostas.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

A COMPONENT-BASED ARCHITECTURAL DESIGN APPROACH IN THE DOMAIN ENGINEERING CONTEXT

Ana Paula Terra Bacelo Blois

April / 2006

Advisors: Cláudia Maria Lima Werner and
Karin Becker

Department: Computer and Systems Engineering

Software reuse is considered to be an efficient way to increase productivity and decrease costs. Domain Engineering (DE) and Component Based Development (CBD) are examples of software reuse approaches. Usually, DE methods neither emphasize the design phase activities nor consider the inherently variability of the domain requirements in all life cycle phases. On the other hand, CBD methods propose systematic ways for the development of reusable components, although, in practice, their reuse support is limited. This thesis proposes an approach to support component-based architectural design, in a DE context. This approach supports variability modeling throughout the software life-cycle aiming to specify a consistent Reference Architecture (RA). This approach, called DECOM, proposes a DE process, which combines principles of DE and CBD methods, with its main focus on domain design activities. The approach uses a notation to represent variability in domain models and proposes heuristics that aid in achieving consistency among different models. In order to improve RA reuse, a set of criteria to suggest architectural elements identification is adopted. Moreover, it also incorporates an approach of domain models transformation into a specific component technology. Finally, observational studies to evaluate some proposed activities were performed.

Índice

Índice de Figuras

Índice de Tabelas

Capítulo 1 - Introdução

1.1 - Contexto

O desenvolvimento de um software é uma atividade complexa que deve gerar resultados que satisfaçam as necessidades de seus usuários. Para auxiliá-los nesta construção, Engenheiros de Software utilizam recursos (e.g. processos, técnicas, linguagens de programação), geralmente voltados a algum paradigma de desenvolvimento (e.g. estruturado, orientado a objetos, baseado em componentes).

A construção de um software resulta num conjunto de artefatos, incluindo modelos, que muitas vezes podem ser utilizados em uma outra aplicação semelhante. É baseado neste fato que dentro da Engenharia de Software existe uma área denominada reutilização de software.

A reutilização de software tem sido referenciada como uma das formas mais promissoras para aumentar a produtividade, melhorar a confiança do produto e diminuir custos (2005). Em (TRACZ, 1995), o autor afirma que a produtividade dos projetos aumenta em torno de 50% com o uso de técnicas de reutilização de software**.**

A idéia de reutilização é antiga, mas foi com a Orientação a Objetos (OO) que ela se tornou mais popular (PRESSMAN, 2000). O paradigma de OO enfatiza a criação de classes que encapsulam dados e algoritmos. Se o projeto e a implementação são desenvolvidos adequadamente, classes podem ser reutilizadas por diferentes aplicações. Por esta razão, surgiram algumas iniciativas para a reutilização de soluções para problemas recorrentes, dentre elas: classes abstratas (PREE, 1995), *frameworks* (FAYAD, 1997), padrões de projeto (GAMMA *et al.*, 1995) e padrões arquiteturais (BUSCHMANN *et al.*, 1996).

Um paradigma que também visa apoiar a reutilização de software é o DBC (Desenvolvimento Baseado em Componentes). Embora exista uma discussão quanto ao que venha a ser um componente, é possível dizer que há um consenso de que este artefato deva ser autocontido, claramente identificável e com interfaces bem definidas, com documentação apropriada e um grau de reutilização definido (SAMETINGER, 1997).

As primeiras iniciativas de reutilização de componentes ocorreram no nível de código-fonte, mas atualmente existem propostas para a reutilização destes artefatos no nível de análise e projeto (BOSCH, 2000; ATKINSON *et al.*, 2002; ODYSSEY, 2005).

Continuamente, novas tecnologias de apoio ao DBC aparecem no mercado e se tornam cada vez mais populares (e.g. EJB, .NET , COM). Por esta razão, um desafio enfrentado pelas equipes de desenvolvimento de software é a escolha dessas tecnologias. Muitas empresas precisam adotá-las em função das expectativas de seus clientes, pelos problemas solucionados por essas novas tecnologias, ou porque as tecnologias antigas se tornaram obsoletas. Como conseqüência, componentes existentes precisam ser migrados para uma nova tecnologia, ou para uma nova versão de uma tecnologia existente (KLEPPE, 2003). Visando diminuir esse efeito, a OMG (*Object Management Group*) está desenvolvendo um *framework* conhecido como *Model Driven Architecture* (MDA) (OMG, 2003) que visa transformar modelos independentes de tecnologia em modelos específicos para uma dada tecnologia. Com o uso dessa abordagem também em DBC, os desenvolvedores passam a dar maior importância, em um primeiro momento, à modelagem dos requisitos de negócio da aplicação, diminuindo a preocupação com a plataforma em que serão implementados.

A reutilização de artefatos também aumenta quando a mesma ocorre no contexto de um domínio de aplicações. Neste sentido, a Engenharia de Domínio (ED) se preocupa com a identificação e modelagem de características comuns e variáveis em aplicações de um dado domínio, gerando como resultado modelos de domínio (e.g. casos de uso de domínio, classes do domínio) que podem ser reutilizados em aplicações específicas (PRIETO-DIAZ & ARANGO, 1991).

Dentre os modelos de domínio, o mais conhecido na literatura é o modelo de características, o qual descreve a teoria do domínio (PRIETO-DIAZ & ARANGO, 1991). Este modelo não inclui somente as características do domínio em si, mas também descreve como elas estão relacionadas estruturalmente (BRAGA, 2000). O modelo de características é um elemento chave para relacionar os conceitos de mais alto nível aos demais artefatos de análise e também para os artefatos de projeto, além de ser considerado o melhor caminho para efetuar o recorte do domínio para a construção de uma aplicação.

O Engenheiro de Domínio é quem constrói modelos de domínio com o apoio oferecido por processos de ED. Estes processos, geralmente, compreendem as fases de

análise, projeto e implementação. Na fase de análise, é feito o levantamento dos requisitos do domínio, normalmente especificados através de modelos de características. Na etapa de projeto, modelos de análise são refinados visando a construção de uma arquitetura do domínio, conhecida como Arquitetura de Software Específica para Domínios (DSSA - *Domain Specific Software Architecture*) (SHAW & GARLAN, 1996). Segundo (XAVIER, 2001), uma DSSA representa mais do que uma arquitetura para um determinado domínio de aplicações, mas sim uma coleção de componentes de software, especializados para um determinado tipo de tarefa (domínio) e generalizados para que seja possível seu uso efetivo através do domínio. Além disso, os componentes são disponíveis numa arquitetura padronizada para a construção de aplicações do domínio. Por último, a fase de implementação visa à geração de código fonte para a arquitetura gerada no projeto do domínio.

A identificação de características comuns e variáveis de um domínio é conhecida na literatura técnica como modelagem de variabilidade do domínio (CLAUSS, 2001; KANG *et al.*, 2002; GOMAA & SHIN, 2004; MASSEN & LICHTER, 2004). Na ED, a variabilidade dos requisitos deve ser documentada em todo o ciclo de vida do software, isto é, em todos os artefatos de cada fase da ED, sendo assim indispensável representá-las não só no modelo de características, mas também propagá-las para os demais modelos de domínio de forma consistente (GOMAA & SHIN, 2004).

Durante a ED, em especial durante a fase de projeto, se espera como resultado a construção de uma ou mais Arquiteturas de Referência (AR). Uma AR é definida como uma arquitetura para uma família de aplicações que deve atender a um conjunto de requisitos funcionais e não-funcionais de um domínio (SHAW & GARLAN, 1996). O objetivo dessas arquiteturas é formalizar um conjunto de requisitos funcionais e não funcionais do domínio, organizados sob a forma de elementos arquiteturais (GARLAN, 2000).

A partir da ED, aplicações devem ser instanciadas, considerando os modelos do domínio, através do recorte de um subconjunto de artefatos que atendam a seus requisitos funcionais e não-funcionais. Ao processo envolvido nesta instanciação se dá o nome de Engenharia de Aplicação (EA) (PRIETO-DIAZ & ARANGO, 1991). Este recorte dos artefatos para a EA é fortemente influenciado pelas variabilidades do domínio, uma vez que elas determinam o tipo de aplicação resultante desta instanciação.

Na EA, se espera que a AR gerada para o domínio, possa ser reutilizada e adaptada ao contexto da aplicação instanciada (MILER, 2000).

1.2 - Motivação

Existem diversos métodos de ED e DBC na literatura, tais como: FODA (Feature Oriented Domain Analysis) (KANG et al., 1990), DARE (Domain Analysis Reuse Environment) (FRAKES et al., 1998), FORM (Feature Oriented Reuse Method) (KANG et al., 2002), FODACom (VICI & ARGENTIERI, 1998), FeatuRSEB (Reuse-Driven Software Engineering Business) (GRISS et al., 1998), ODM (Organization Domain Modeling) (SIMOS & ANTHONY, 1998), UML Components (CHEESMAN & DANIELS, 2000), Catalysis (D'SOUZA & WILLS, 1999) e Kobra (ATKINSON et al., 2002).

Em geral, os métodos de ED (e.g. FODA, FORM, RSEB, ODM) dão maior apoio à fase de análise de domínio. Tal característica torna estes métodos restritos pois, do ponto de vista da EA, é esperado que os artefatos de análise e projeto possam ser reutilizados para a construção de uma aplicação específica. A reutilização de artefatos de análise se torna mais efetiva se estiverem associados a artefatos dos níveis mais baixos de abstração (WERNER *et al.*, 1999; WERNER, 2005).

Outro aspecto observado é que estes métodos de ED não apresentam uma semântica completa para a representação das variabilidades dos requisitos do domínio e, tampouco, propõem uma forma de propagação e representação de tais variabilidades para os demais artefatos do domínio (GOMAA & SHIN, 2004). Tais métodos também não dão apoio à criação e, tampouco à reutilização de arquiteturas de referência (ARANGO, 1994).

Por outro lado, embora os métodos de DBC (e.g. Catalysis, UML Components e Kobra) se proponham a gerar artefatos reutilizáveis, tais métodos pouco consideram a variabilidade inerente à modelagem de uma família de aplicações na criação e especificação de um componente. Além disso, eles não propõem sistemáticas para apoio à reutilização dos artefatos produzidos. Também se observa que os métodos de DBC não fornecem apoio à organização dos componentes gerados para a construção de uma arquitetura de componentes.

Um outro aspecto observado é a oportunidade de promover a reutilização de artefatos com a combinação das áreas de ED e DBC. Se por um lado a ED visa identificar aspectos variáveis e invariáveis de um domínio para a sua reutilização, por outro o DBC propõe soluções reutilizáveis para o processo de desenvolvimento de software, disponibilizando aos usuários tais variabilidades através de componentes e interfaces (BROWN, 2000). Neste sentido, um desafio da área de ED é promover o uso de abordagens de DBC, visando a construção de ARs baseadas em componentes. Por outro lado, como desafio da área de DBC podemos citar a modelagem de variabilidades, através de componentes, representando de forma consistente tais aspectos em domínios de aplicações.

Braga (2000) propôs um processo – Odyssey–ED - com a intenção de unir os aspectos de reutilização e entendimento do domínio, inerentes aos métodos de ED, e o desenvolvimento de componentes, fornecido pelos métodos de DBC. Entretanto, o conjunto de atividades, no que tange especificamente ao projeto de domínio com o uso de DBC, é ainda limitado. O modelo de componentes utilizado contempla componentes e suas interfaces. No entanto, não é possível desenvolver a especificação interna desses componentes como previsto no Odyssey-DE. A falta de tal especificação limita o entendimento a respeito do funcionamento do componente como elemento auto-contido, uma vez que não se tem conhecimento sobre o seu funcionamento interno do componente, e exatamente o ponto de comunicação com os demais componentes do modelo, considerando as interfaces providas e requeridas. Além disso, também não existe apoio à especificação de variabilidades, opcionalidades e regras de composição. Tal limitação pode gerar modelos de componentes inconsistentes, pois é possível que componentes, que deveriam ser mutuamente exclusivos, troquem mensagens através de suas interfaces.

Um aspecto positivo é que Odyssey-DE utiliza estilos arquiteturais para a construção de componentes de negócio (TEIXEIRA et al., 2004). No entanto, não apresenta soluções para o agrupamento de componentes visando à construção de elementos arquiteturais para a composição de uma AR. Neste sentido, o apoio à construção de ARs é limitado.

Como conseqüência das limitações detectadas nos métodos de ED e DBC, em especial em sua fase de projeto, pouco se discute na literatura sobre a fase de implementação. Em especial nos métodos de DBC, não são exploradas sistemáticas para

apoiar a geração de componentes implementacionais, segundo uma tecnologia em particular.

1.3 - Objetivos

Tendo em vista as possibilidades de pesquisa ainda existentes nas áreas de ED e DBC, esta tese tem como principal objetivo aproveitar as vantagens oferecidas por estas abordagens e minimizar as limitações encontradas, propondo uma abordagem de projeto arquitetural de componentes, denominada DECOM, no contexto da ED.

Para tanto, são propostos nesta tese:

a) um processo de ED com o foco em DBC, denominado CBD-Arch-DE;

b) uma notação abrangente para a representação de variabilidades e opcionalidades em artefatos do domínio;

c) o estabelecimento de heurísticas para apoio ao mapeamento de artefatos do domínio, para que possam representar de forma consistente os diferentes níveis de abstração de um mesmo requisito do domínio;

d) um conjunto de critérios para a geração de elementos arquiteturais que irão formar uma arquitetura de referência ou arquitetura de aplicação;

e) o apoio à geração de componentes implementacionais através de uma abordagem de transformação de modelos, baseada no *framework* MDA;

f) a construção de ferramentas para apoiar todos os objetivos anteriormente mencionados.

Ao atingir estes objetivos, acredita-se que este trabalho contribui no sentido de dar mais um passo em direção ao apoio à reutilização no desenvolvimento de software, a partir dos seguintes aspectos:

a) A definição do processo CBD-Arch-DE, orientando efetivamente a execução de atividades referentes a todas as fases inerentes à ED;
b) A definição da notação para modelagem de variabilidades e opcionalidades adotada para os diferentes modelos de domínio, através da qual é possível representar adequadamente das características vinculadas a cada modelo;
c) A identificação de heurísticas que auxilia de forma consistente a manutenção das variabilidades, opcionalidades, relacionamentos e regras de composição entre os diferentes modelos do domínio;

d) A manutenção destas variabilidades e opcionalidades em diferentes níveis de abstração influenciando positivamente na consistência do recorte da aplicação derivada do domínio modelado;
e) A especificação e implementação de uma AR, através de sistemáticas para apoio à geração de diferentes categorias de componentes e da organização destes componentes sob a forma de elementos arquiteturais do domínio;
f) O apoio à modelagem do domínio, independente de uma tecnologia de DBC, resultando num conjunto de modelos de domínio reutilizáveis;
g) A possibilidade de explorar tais benefícios através de um ambiente de reutilização real, i.e. Odyssey.

Até o momento, não encontramos na literatura uma abordagem que reúna estes aspectos de forma integrada. Contribuições pontuais sobre alguns dos aspectos atendidos para a abordagem proposta serão apresentadas ao longo dos capítulos desta tese.

1.4 – Projetos Relacionados

Esta tese foi desenvolvida no contexto do Projeto Reuse – COPPE/UFRJ (WERNER, 2005). O objetivo principal deste projeto é explorar técnicas e ferramentas que dêem apoio à Reutilização de Software, particularmente nos contextos da Engenharia de Domínio e do Desenvolvimento Baseado em Componentes. Dentre as técnicas propostas, se encontram: 1) gerência de configuração (MURTA *et al.*, 2004a), 2) recuperação de arquitetura de software (VASCONCELOS & WERNER, 2004), 3) projeto arquitetural baseado em componentes (BLOIS *et al.*, 2004; BLOIS, 2004; BLOIS *et al.*, 2005a) e 4) transformação de software (MAIA *et al.*, 2005a; MAIA *et al.*, 2005b).

O Projeto Reuse tem como origem o Projeto Odyssey (WERNER *et al.*, 1999). O ambiente Odyssey, desenvolvido pela equipe de reutilização de software da COPPE/UFRJ desde 1998, tem como objetivo principal prover mecanismos para reutilização, visando o desenvolvimento de software baseado em componentes. Este ambiente serve como um arcabouço em que modelos conceituais e arquiteturas de software são especificados para domínios de aplicações. Para a representação destes modelos, são utilizadas extensões da UML (OMG, 2005) que representam o conhecimento de um domínio, e que podem ser reutilizados para facilitar a construção de aplicações.

Neste contexto, esta tese propõe uma abordagem que contempla as técnicas 3 e 4 propostas pelo projeto Reuse e, como conseqüência, um conjunto de ferramentas que podem ser utilizadas através do ambiente de reutilização Odyssey.

1.5 - Organização da Tese

Este trabalho está organizado em seis capítulos. Neste **primeiro** capítulo, de introdução, são apresentados a motivação, os objetivos e a organização do trabalho.

No **segundo** capítulo, é feita uma revisão da literatura a respeito de Engenharia de Domínio, Desenvolvimento Baseado em Componentes e Arquiteturas de Software para Domínios e baseadas em Componentes. Além disso, é feita uma análise dos métodos de ED e DBC, baseada num conjunto de requisitos esperados para tais métodos.

No **terceiro** capítulo, é descrita a abordagem proposta, denominada DECOM; a notação para modelagem de variabilidades; o processo de ED com foco em DBC, um conjunto de heurísticas para apoio ao mapeamento de variabilidades, opcionalidades, relacionamentos e outras propriedades especificadas em modelos de características para outros modelos de domínio; os critérios para a identificação de elementos arquiteturais; e a abordagem de transformação de componentes envolvidos nesta pesquisa.

No **quarto** capítulo, é detalhada toda a implementação da DECOM, no contexto do ambiente Odyssey.

No **quinto** capítulo, são detalhados os estudos de observação realizados para avaliar indícios de viabilidade da abordagem de projeto arquitetural de componentes.

Finalmente, o **sexto** e último capítulo apresenta as conclusões e contribuições desta tese, bem como as limitações e perspectivas para pesquisas futuras.

Os **anexos** desta Tese constam no final deste documento, após as referências bibliográficas. O anexo 1 descreve parte da especificação do domínio de Telefonia Móvel, o qual será explorado nesta tese para exemplificar a abordagem proposta. O anexo 2 descreve parte da especificação de componentes prevista na UML 2.0. Os anexos 3 ao 7 consistem em documentos e modelos utilizados no primeiro Estudo de Observação desta tese. Os anexos 8 ao 10 se referem aos documentos e modelos utilizados no segundo estudo de observação. Por último, os anexos 11 ao 14 correspondem a documentações referentes ao terceiro estudo de observação.

Capítulo 2 – Engenharia de Domínio, Desenvolvimento Baseado em Componentes e Arquiteturas de Software: uma Revisão da Literatura

2.1 - Introdução

Neste capítulo são discutidas as áreas de Engenharia de Domínio (ED), Desenvolvimento baseado em Componentes (DBC) e Arquiteturas de Software, que servem de base para a abordagem proposta nesta tese. A maioria das contribuições na área de DBC estão voltadas para as questões de implementação, como tecnologias EJB, .NET, .COM. Já a área de ED não explora a contento todas as fases do ciclo de vida de um domínio de aplicações e pouco consideram o paradigma de DBC. Neste sentido, o uso de uma abordagem de DBC no contexto de ED pode prover sistemáticas para o apoio adequado do desenvolvimento de domínios de aplicações baseadas em componentes, refletindo na melhoria do potencial de reutilização dos artefatos do domínio para a geração de aplicações e, sobretudo para a construção de Arquiteturas de Referência (AR).

O capítulo está organizado da seguinte maneira: a Seção 2.2 apresenta brevemente a área de Engenharia de Domínio e a Seção 2.3 a área de DBC. Na Seção 2.4 é discutido os conceitos que norteiam a construção de arquiteturas de referência baseadas em componentes. Uma análise dos principais métodos de ED e DBC é apresentada na Seção 2.5. A Seção 2.6 discute as considerações finais sobre o capítulo.

2.2 - Engenharia de Domínio

A atividade de desenvolvimento de software pode ser melhor conduzida através de métodos, processos, dentre outras formas de orientação, obtendo como resultado um conjunto de artefatos que atendem às necessidades específicas de uma dada aplicação. No entanto, existem domínios (e.g. ensino, hospitalar, telefonia móvel e petróleo) que possuem características comuns e que, portanto, suas aplicações poderiam ser construídas a partir de um mesmo processo e artefatos, promovendo com isto a reutilização dos conceitos e funcionalidades comuns. A este desenvolvimento de

software para domínios de aplicações é atribuído o termo de Engenharia de Domínio (ED) (KANG *et al.*, 1990; PRIETO-DIAZ & ARANGO, 1991).

Segundo Prietro-Diaz e Arango (1991), "a Engenharia de Domínio é o processo de identificação e organização do conhecimento sobre uma classe de problemas, o domínio do problema, para suportar sua descrição e solução". Enquanto que a Engenharia de Domínio se preocupa com o desenvolvimento de artefatos *para* reutilização, a Engenharia de Aplicação (EA) constrói aplicações *com* base na reutilização de artefatos e modelos gerados pela ED. Segundo SEI/CMU (2005), "a Engenharia de Aplicações desenvolve produtos de software baseado nos artefatos gerados pelo processo de Engenharia de Domínio".

Foram propostos vários métodos de ED, dentre eles FODA (KANG *et al.*, 1990), FORM (KANG *et al.*, 2002), FODAcom (VICI & ARGENTIERI, 1998), FeatuRSEB (GRISS *et al.*, 1998) e ODM (SIMOS & ANTHONY, 1998). Esses métodos de ED propõem sistemáticas para a modelagem de características e outros artefatos de domínio, visando a sua reutilização no contexto da EA. Embora cada um tenha suas devidas particularidades para a obtenção e tratamento de artefatos, existem alguns pontos em comum no tocante às atividades envolvidas na sistemática de cada método.

Em geral, a modelagem de um domínio envolve todas as atividades necessárias para a construção de artefatos que serão reutilizados por aplicações derivadas do domínio, incluindo:

1º.) a identificação de um domínio e seu escopo, bem como a captura de requisitos do domínio em que são representadas suas variabilidades e opcionalidades;

2º.) a construção de um projeto adaptável do domínio, através da especificação de AR que representem os requisitos funcionais e não-funcionais do domínio, devidamente estruturados;

3º.) a definição de mecanismos para a tradução dos requisitos do domínio em sistemas criados a partir de artefatos reutilizáveis.

Os produtos resultantes dessas atividades são modelos de análise, modelos de projeto, arquiteturas de referência, linguagens específicas do domínio, geradores de código e código fonte (SEI/CMU, 2005). Estes produtos são gerados nas diferentes fases do ciclo de vida da ED.

Na EA, as fases que norteiam o seu processo são análogas as da ED. Como a EA baseia-se no processo de reutilização dos artefatos gerados na ED, as atividades compreendidas na modelagem de aplicações são mais específicas, atendendo tanto as

particularidades da própria aplicação quanto as características comuns do domínio. A seguir são apresentadas as principais fases da ED.

2.2.1 – Análise de Domínio

A **análise de domínio** busca identificar, coletar e organizar informações relevantes do domínio, utilizando para tal o conhecimento existente do domínio e métodos para a modelagem de informações (KANG *et al.*, 1990). Estes métodos (e.g. FODA e FeatuRSEB) determinam os limites que norteiam o domínio, os quais envolvem os requisitos comuns e as variáveis de aplicações do domínio, gerando como resultado, modelos para a representação de características opcionais, mandatórias, variáveis e invariáveis entre os sistemas estudados. As características variáveis e invariáveis são denominadas de variabilidade e as opcionais e mandatórias são denominadas de opcionalidades.

Um requisito de um domínio pode, do ponto de vista da variabilidade, representar (CLAUSS, 2001; BOSCH, 2004; OLIVEIRA, 2006):

- Ponto de Variação: reflete a parametrização no domínio de uma maneira abstrata e deve ser configurável através de suas variantes;
- Variantes: funções/conceitos disponíveis que devem necessariamente estar ligados a um ponto de variação, atuando como alternativas de configuração de seu respectivo ponto de variação;
- Invariante: elemento não configurável no domínio.

Ortogonalmente, do ponto de vista da opcionalidade, um requisito do domínio pode ser classificado como:

- Opcional: elemento que pode ou não pertencer a uma aplicação derivada de um domínio.
- Mandatório: elemento que obrigatoriamente deve ser instanciado por todas as aplicações derivadas de um domínio.

Embora as variabilidades e opcionalidades devam ser tratadas em artefatos de todas as fases da ED, a maioria dos métodos aborda estas questões somente através do artefato de análise denominado característica.

Características, também conhecidas pelo seu nome em inglês *features*, são artefatos propostos por Kang *et al.*(1990) no contexto de reutilização de software, mais especificamente em ED. Características representam as capacidades/abstrações do

domínio obtidas por especialistas, usuários e a partir de sistemas já existentes, durante a fase de análise do domínio.

Um modelo de características descreve uma teoria do domínio. Ele não inclui somente as características do domínio, mas também descreve como elas estão relacionadas estruturalmente (BRAGA, 2000). O modelo de características é um elemento chave para relacionar os conceitos de mais alto nível aos demais artefatos de análise e também para os artefatos de projeto. Tal modelo é considerado o melhor caminho para efetuar os pontos de corte para a reutilização de parte de um domínio (um conjunto específico de *features* do domínio) para uma dada aplicação, uma vez que ele é composto de características com suas respectivas variabilidades e opcionalidades (KANG *et al.*, 1990; GRISS *et al.*, 1998; VICI & ARGENTIERI, 1998; MILER, 2000; KANG *et al.*, 2002; CZARNECKI *et al.*, 2005).

Diversos autores (GRISS *et al.*, 1998; MILER, 2000; KANG *et al.*, 2002; SVAHNBERG *et al.*, 2002) propõem diferentes tipos de classificações para características, com o intuito de distinguir o tipo de informação que elas representam (e.g. conceito, funcionalidade, técnica). No entanto, as classificações quanto à variabilidade e opcionalidade apresentadas anteriormente parecem ser um consenso para a modelagem de características, embora especificadas e representadas de diferentes maneiras (KANG *et al.*, 1990; GRISS *et al.*, 1998; MASSEN & LICHTER, 2002; RIEBISCH *et al.*, 2002; BOSCH, 2004). Este consenso para a representação de variabilidades e opcionalidades em características na análise de domínio não se aplica aos demais artefatos do domínio, como será observado ao longo deste Capítulo.

2.2.2 – Projeto do Domínio

O **projeto de domínio** é a etapa em que modelos de projeto são construídos, com base nos modelos de análise, no conhecimento obtido por estudos a respeito de projeto reutilizável e arquiteturas genéricas. O projeto do domínio visa especificar a estrutura arquitetural a ser seguida pelas aplicações provenientes do domínio modelado, utilizando para isto especificações de projeto, padrões de projeto e arquitetural e estratégias para a construção de uma AR (SEI/CMU, 2005) (BRAGA, 2000). Nesta etapa é que são geradas as DSSAs que representam uma coleção de artefatos de software, que pertencem a um domínio em particular, e compostos de uma estrutura padrão para a construção de aplicações (TRACZ, 1995). A Seção 2.4.2 trata das questões referentes à construção de uma DSSA.

Como resultado do projeto de domínio obtêm-se modelos de projeto. Conforme veremos a seguir, estes modelos de projeto, no desenvolvimento baseado em componentes, podem vir a ser modelos de componentes projetados segundo um estilo arquitetural específico, modelos de classes, colaboração e seqüência, referentes à estrutura interna dos componentes e/ou modelos de interação (e.g. colaboração, seqüência).

2.2.3 – Implementação do Domínio

Por último, a fase de **implementação do domínio** visa construir, através de código fonte, os artefatos reutilizáveis, baseado na modelagem do domínio e nas arquiteturas de referência propostas na atividade de projeto.

2.3 - Desenvolvimento Baseado em Componentes

A abordagem de componentes é uma outra alternativa para a obtenção de soluções reutilizáveis para o processo de desenvolvimento de software, buscando uma maior produtividade com menor custo. Tais necessidades, aliadas às mudanças que estão ocorrendo com a tecnologia de computação (e.g. computação distribuída, desenvolvimento para WEB, necessidade de estabelecimento de padrões para a construção de aplicações), são alguns dos principais fatores que motivam o uso da abordagem de DBC (SAMETINGER, 1997).

2.3.1 – Conceitos Básicos

Foram apresentadas e discutidas em (BROWN & WALLNAU, 1998) diferentes definições do que venha a ser um componente de software, em função de questões relativas à dependência de contexto (tipo de aplicação no qual pode ser utilizado) e nível de abstração.

Uma definição bastante abrangente, e que será adotada nesta tese, é a proposta por Sametinger (1997), a saber: “componentes de software reutilizáveis são artefatos autocontidos, claramente identificáveis, que descrevem ou realizam uma função específica e têm interfaces claras, documentação apropriada e um grau de reutilização definido”.

Um componente autocontido é aquele que não precisa de outro componente para ser reutilizável. Quando existe a necessidade de cooperação entre componentes, esta

interação deve ser feita através de suas interfaces, i.e. conjuntos de assinaturas de operações existentes no componente que podem ser invocadas por outro.

Para viabilizar a reutilização de um componente, este deve ser claramente identificável, com o objetivo de facilitar a sua localização e, juntamente com a documentação, poder avaliar o seu potencial de reutilização através das funcionalidades descritas (e.g. descrição das interfaces, casos de uso, localização, etc).

Gimenes *et al.* (2005) identificam alguns benefícios relacionados à reutilização de componentes, tais como: redução de custo e tempo de desenvolvimento; gerenciamento da complexidade; desenvolvimento paralelo de componentes que possuem serviços independentes e bem definidos; aumento da qualidade, partindo do pressuposto que estes componentes sejam cuidadosamente testados para promover a sua reutilização, e facilidade de manutenção e atualização, em função da identificação clara dos serviços prestados pelos mesmos.

Contudo os autores também destacam que, para alcançar este benefícios esperados, é necessário lidar com a complexidade envolvida nas tarefas de desenvolvimento de componentes para e com a reutilização, tanto nos seus aspectos técnicos quanto gerenciais. Para Vitharana et al. (2003), no início do ciclo de vida de um software baseado em componentes, deve-se identificar as estratégias do negócio, estabelecendo os objetivos gerenciais a serem posteriormente traduzidos em características técnicas de componentes. A Tabela 2.1 mostra alguns objetivos gerenciais e características técnicas que podem ser esperadas por aplicações que utilizam um ciclo de vida de DBC. Conforme apresentado nesta tabela, quanto maior o acoplamento entre componentes de uma aplicação o domínio, melhor será o custo associado ao processo de criação destes componentes. Ainda, a coesão de um componente permite que a montagem de uma arquitetura de componentes seja facilitada. Outras características técnicas como tamanho e complexidade de um componente podem interferir na sua capacidade de reutilização e manutenção. Por fim, a quantidade de componentes de um determinado domínio de aplicações pode interferir na facilidade de reutilização e adaptação destes componentes para uma aplicação específica. Estas características técnicas são esperadas para se obter bons resultados durante a construção de uma AR baseada em componentes, as quais serão consideradas para esta abordagem de projeto arquitetural.

Tabela 2.1: Objetivos Gerenciais e Características Técnicas propostas por (VITHARANA *et al.*, 2003)

Objetivos Gerenciais	Características Técnicas Esperadas dos métodos de DBC
Eficácia do custo para a criação dos componentes	Acoplamento inter-componentes
Facilidade de montagem de componentes dentro de aplicações	Coesão intra-componente
Capacidade dos componentes para serem adaptados no desenvolvimento de diferentes aplicações	Número de componentes que representa um domínio de aplicações
Potencial dos componentes para reutilização em múltiplas aplicações	Tamanho do componente
Facilidade de manutenção do componente	Complexidade (número de métodos e complexidade dos parâmetros em objetos de um componente)

Segundo Brown (2000), a partir do conhecimento inicial das necessidades do negócio, um método de DBC deve prever três etapas: entendimento do contexto, definição da arquitetura e proposta de solução, conforme mostra o esquema da Figura 2.1.

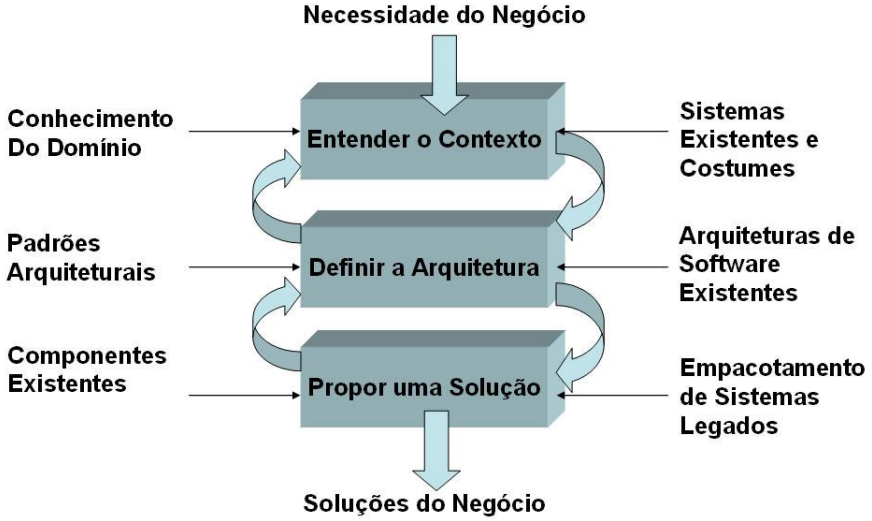


Figura 2.1: Elementos de uma Abordagem de DBC - extraído de (BROWN, 2000)

O esquema da Figura 2.1 apresenta uma perspectiva macro do conjunto de elementos e alternativas de solução para o DBC. Neste sentido, cabe então aos métodos para apoio ao DBC efetuar as devidas extensões nestes elementos e propor soluções de negócio que, ao mesmo tempo, atendam as perspectivas de modelagem independente de uma tecnologia de componentes alvo e, num momento específico, o mapeamento destes componentes para uma tecnologia específica.

Na etapa de entendimento do contexto, são definidos os requisitos iniciais da aplicação e modelados os casos de uso que definem os usuários e as atividades por eles desempenhadas. Além disso, devem ser especificados os tipos de negócio, ou seja, os principais elementos do negócio e seus relacionamentos no domínio. Para o desenvolvimento desta primeira etapa, o conhecimento do domínio e dos sistemas já existente é fator chave.

A etapa posterior envolve a definição da arquitetura, a qual é desenvolvida a partir dos aspectos estáticos e dinâmicos modelados na fase anterior. As atividades previstas na fase de definição de arquitetura são:

- **modelagem da arquitetura de componentes**, em que uma coleção de componentes relacionados são propostos e definidos. A arquitetura de componentes gerada pode organizá-los de acordo com o tipo de serviço oferecido (e.g. infra-estrutura, negócio);
- **modelagem do contexto**, para entender o escopo do software que está sendo desenvolvido. Em geral, este entendimento compreende uma especificação mais exata das responsabilidades dos componentes em relação aos casos de uso da aplicação;
- **modelagem de interface**, para obter um conjunto de interfaces candidatas e descrevê-las em detalhe. Sugere-se que o projetista considere os resultados da modelagem de contexto e dos tipos do negócio para eleger o primeiro conjunto de interfaces candidatas;
- **definição de interface**, na qual interfaces obtidas como resultado da atividade anterior são descritas em detalhe (e.g. com o uso da OCL – Object Constraint Language) (CHEESMAN & DANIELS, 2000).

A última etapa é a proposta de solução, onde os componentes são efetivamente implementados. Deve-se viabilizar a comunicação entre componentes gerados com os sistemas já existentes e ainda componentes de terceiros. Esta etapa encerra com a entrega do sistema.

2.3.2 - Arquiteturas baseadas em Componentes

Do ponto de vista da organização dos componentes, alguns autores (JACOBSON *et al.*, 1997; SZYPERSKI, 1998; D'SOUZA & WILLS, 1999; BROWN, 2000; XIA, 2000) sugerem que uma arquitetura de componentes deve ser organizada em diferentes camadas, em que os componentes possuem diferentes níveis de abstração,

e as camadas inferiores prestam serviços para as camadas superiores. Embora a estrutura em camadas represente uma boa alternativa para a estruturação de arquiteturas de componentes, outras formas de estruturação de arquiteturas podem ser avaliadas para o contexto de DBC. Nesta Seção será discutida a organização de arquiteturas de componentes proposta por Brown (2000), conforme mostra a Figura 2.2, e que será também utilizada pela abordagem de projeto arquitetural apresentada nesta tese.

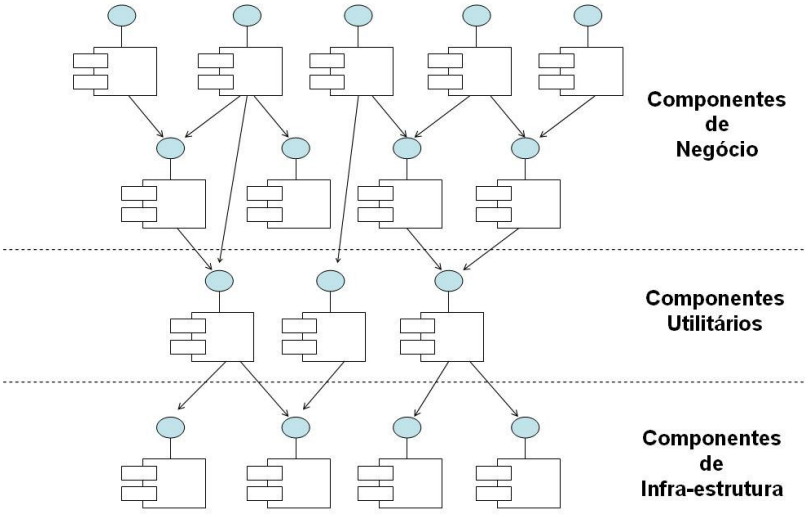


Figura 2.2: Componentes de uma Arquitetura em Camadas

Segundo Brown (2000), a primeira camada compreende os componentes de negócio, que implementam um conceito ou processo de negócio autônomo. Estes componentes são artefatos necessários para representar, implementar e implantar um conceito ou processo do negócio. O modelo de componentes deve especificar as dependências entre os componentes por meio de suas interfaces providas e requeridas. Alguns autores, como Herzum e Sims (2000) e Cheesman e Daniels (2000), assim como algumas tecnologias, como Enterprise Javabeans (SUN, 2005a), identificam duas categorias de componentes de negócio: componentes de entidade e componentes de processo. Os componentes de entidade representam os principais conceitos de um domínio, onde os processos de negócio operam. Eles correspondem a conceitos normalmente identificados pela análise orientada a objetos ou pela análise de entidade-relacionamento da aplicação. Os componentes de processo, também denominados por Cheesman e Daniels (2000) de componentes transacionais, representam os processos e atividades relativas ao domínio da aplicação, executando ou apoiando usuários na execução de tarefas específicas do negócio. Na tecnologia EJB, os componentes de

negócio correspondem aos *Entity Beans* e os componentes de processo os *Session Beans*.

A camada intermediária possui os componentes utilitários, que prestam serviços genéricos necessários ao desenvolvimento das aplicações (BROWN, 2000). Componentes utilitários não pertencem a nenhum domínio específico, mas são utilizados por inúmeras aplicações de negócio (e.g. controle de calendários, agenda de endereços, formulários genéricos). Do ponto de vista de reutilização, os componentes utilitários são mais reutilizados, se comparados aos componentes de negócio, por exemplo.

A última camada envolve os componentes de infra-estrutura. Estes componentes são responsáveis por estabelecer a comunicação com as plataformas de execução do software. Eles estão mais vinculados às questões de tecnologia de desenvolvimento do que os componentes das camadas superiores (BROWN, 2000). Dentre os serviços providos por componentes de infra-estrutura, estão: comunicação entre componentes distribuídos, persistência, tratamento de exceções e portabilidade. A camada de componentes de infra-estrutura possui uma forte relação com a tecnologia de componentes adotada na aplicação gerada.

Esta estrutura de divisão em camadas visa padronizar em três grandes categorias os componentes gerados/utilizados num processo de DBC, de acordo com o tipo de apoio oferecido e nível de abstração. Ao se adotar uma tecnologia de componentes para a construção da aplicação, esta estrutura de camadas deverá sofrer adaptações para as características da tecnologia (e.g. J2EE, JavaBeans, COM). Cada tecnologia tem sua própria forma de distribuir os serviços dos componentes de uma aplicação, bem como uma forma particular de estabelecer a comunicação entre os diferentes componentes.

Nesta seção, o termo arquitetura baseada em componentes foi tratado num contexto específico, sem qualquer perspectiva de apoio à sua construção visando à reutilização futura. A construção de arquiteturas de componentes para domínios de aplicações será tratada na Seção 2.4.

2.4 – Arquiteturas de Software para Domínios e baseadas em Componentes

Num contexto de reutilização de software, uma arquitetura de software deve atender aos requisitos de uma família de aplicações. Neste sentido, esta arquitetura deve representar uma estrutura de referência para o domínio modelado, a qual será reutilizada por diferentes aplicações derivadas deste domínio.

Se construir uma arquitetura para uma única aplicação não é uma tarefa trivial, a complexidade aumenta bastante quando a arquitetura construída deve representar uma família de aplicações de um domínio. Tal complexidade reflete em particular: no custo, tempo e esforço de construção desta arquitetura, e na formação esperada da equipe de desenvolvimento para lidar com tal nível de complexidade (MENDES, 2002).

Tendo em vista a importância da atividade de construção de uma arquitetura, é que a Arquitetura de Software tem sido considerada como uma disciplina da Engenharia de Software.

Nas subseções que se seguem, esta disciplina é discutida com o intuito de identificar os problemas, complexidades e soluções para a construção de arquiteturas para um domínio de aplicações, construídas a partir da abordagem de componentes.

2.4.1 – Conceitos Básicos

Existem várias definições de Arquitetura de Software na literatura (PFLEEGER, 1998; FIELDING, 2000; GARLAN, 2000). Nesta tese, será adotada a definição de Shaw e Garlan (1996) que definem: "uma arquitetura de software envolve a descrição de elementos arquiteturais dos quais os sistemas serão construídos, interações entre esses elementos, padrões que guiam suas composições e restrições sobre estes padrões". A arquitetura de um software define em termos computacionais quais são seus elementos arquiteturais e como ocorre a interação entre eles. Elementos arquiteturais neste contexto podem ser bancos de dados, servidores, clientes, filtros, um ou mais componentes, dentre outros, e a interação entre eles pode ocorrer através de chamadas de procedimentos, acesso a variáveis, uso de protocolos para acesso a clientes e servidores, bancos de dados, e outros eventos quaisquer (GARLAN, 2000) .

Esta definição se preocupa com a questão da estrutura em que os artefatos estão relacionados e as restrições envolvendo fatores internos e externos do sistema. Existem propostas que se preocupam com a organização dos artefatos do domínio em diferentes

visões arquiteturais (CLEMENTS, 1995; KRUCHTEN, 1995; HOFMEISTER *et al.*, 1999), dentre elas a de Kruchten (1995), que define o modelo de visões arquiteturais conhecido como 4+1. Cada uma das quatro visões arquiteturais (lógica, de processo, de desenvolvimento e física) trata de um conjunto de objetivos específicos do projeto arquitetural. A quinta visão, de cenários, ilustra o comportamento das demais.

O apoio a estas questões estruturais de uma arquitetura varia em relação as diferentes alternativas de projeto de um sistema (e.g. estruturado, OO, DBC). Tais alternativas podem utilizar o apoio oferecido por abordagens que propõem soluções reutilizáveis como, por exemplo, *frameworks* (FAYAD, 1997; JOHNSON, 1997), padrões de projeto e arquiteturais (GAMMA *et al.*, 1995; BUSCHMANN *et al.*, 1996; SCHMIDT, 2000), estilos arquiteturais (SHAW & GARLAN, 1996), linguagens de descrição arquiteturais (MEDVIDOVIC & TAYLOR, 2000) e métodos de DBC (D'SOUZA & WILLS, 1999; CHEESMAN & DANIELS, 2000). Tais abordagens apóiam de alguma forma a construção de elementos arquiteturais de um sistema os quais, por sua vez, devem também estar em consonância com seus respectivos requisitos não funcionais (e.g. escalabilidade e desempenho).

Para alcançar os objetivos esperados de um bom projeto arquitetural, é também necessário um conjunto de habilidades do arquiteto de software. Ele precisa ter um conhecimento profundo dos requisitos das aplicações, das tecnologias disponíveis para apoio à construção da arquitetura e dos processos de desenvolvimento adequados para a aplicação a ser desenvolvida. Tais habilidades devem estar alinhadas aos objetivos organizacionais que influenciam nas decisões técnicas (VITHARANA *et al.*, 2003), os quais foram descritos na Seção 2.3.1.

2.4.2 - Arquiteturas de Software Específicas para Domínio

Num contexto de ED, uma arquitetura de software deve se preocupar em propor uma solução estruturada para os requisitos que correspondem a uma família de aplicações de um determinado domínio, a qual é conhecida como Arquiteturas de Software Específicas para Domínio, ou *Domain Specific Software Architecture* (DSSA). Uma DSSA fornece uma base estrutural para a interoperabilidade de componentes dentro de um domínio (MENDES, 2002).

É consenso na literatura (TRACZ, 1995; XAVIER, 2001; MENDES, 2002), que uma DSSA compreende quatro elementos principais, conforme mostra a Figura 2.3:

1. Um modelo de domínio (e.g. modelo de características do domínio);

2. Os requisitos de referência, isto é, requisitos funcionais e não-funcionais do domínio, sendo que estes últimos estabelecem restrições sobre a arquitetura e a implementação;
3. Uma AR, que satisfaça os requisitos funcionais e não-funcionais e,
4. Um processo de desenvolvimento que apóia a Engenharia de Domínio como um todo.

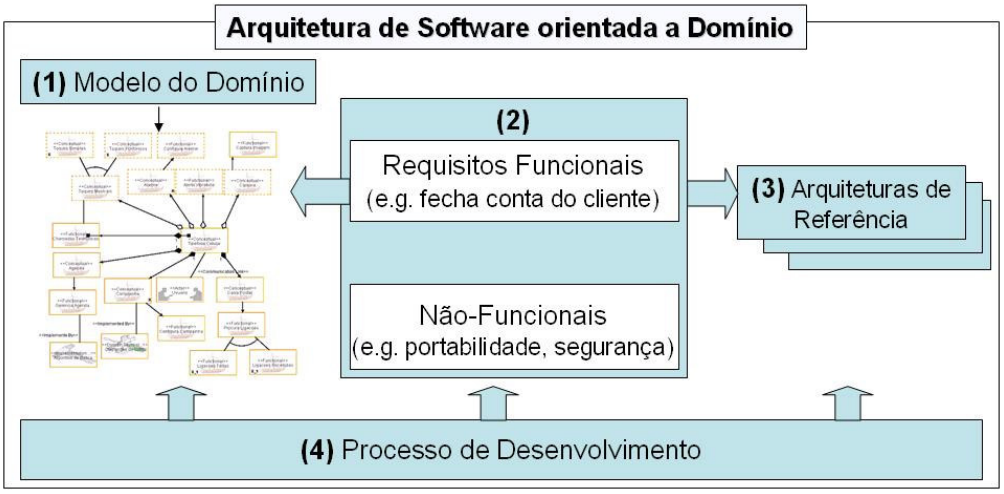


Figura 2.3: Elementos de uma DSSA

Uma AR não necessariamente satisfaz todos os requisitos funcionais e não-funcionais de um domínio. Neste sentido, podem existir uma ou mais arquiteturas de referência, permitindo maior flexibilidade para o Engenheiro durante a construção de uma arquitetura de aplicação.

Existe uma distinção entre AR e a arquitetura de uma aplicação. Uma AR é definida como uma arquitetura para uma família de aplicações construída na fase de projeto do domínio. Já uma arquitetura de aplicação é definida como uma instância e/ou refinamento de uma arquitetura de referência, ou seja, representa uma única aplicação, obtida através da EA (MILER, 2000). Exemplos de AR para domínios específicos podem ser encontrados em (CRISPEN *et al.*, 1993; COLLINS-COPE & MATTHEWS, 2000; GROSSKURTH & GODFREY, 2005).

2.4.2.1 – Linguagens, Estilos e Padrões Arquiteturais.

A maioria das iniciativas para a construção de arquiteturas de referência destaca o apoio das linguagens de descrição arquitetural, estilos arquiteturais e padrões arquiteturais (BUSCHMANN *et al.*, 1996; SHAW & GARLAN, 1996; MEDVIDOVIC & TAYLOR, 2000).

Uma Linguagem de Descrição Arquitetural (*Architecture Description Language* (ADL)) , representa uma notação formal para representação de arquiteturas de software num alto nível de abstração. Um *framework* para a classificação e comparação de ADLs pode ser encontrado em (MEDVIDOVIC & TAYLOR, 2000). Existem propostas de ADL aplicadas ao contexto de arquiteturas baseadas em componentes (e.g. ABC/ADL (CHEN *et al.*, 2002), CDL (MENCL, 1998) e a xADL (DASHOFY *et al.*, 2002)). Na literatura pesquisada não foi identificado o uso de ADLs para a formalização de AR.

Segundo BUSCHMANN *et al.* (1996), padrões arquiteturais expressam um esquema de organização estrutural de sistemas. Através desta organização, são definidos os subsistemas, seus relacionamentos, bem como regras e orientações para um bom uso do padrão arquitetural. Padrões são geralmente descritos através de uma linguagem de padrões (e.g. nome do padrão, contexto, problema, solução, estrutura, usos conhecidos e padrões relacionados). Os padrões arquiteturais mais conhecidos são os relatados por (BUSCHMANN *et al.*, 1996), como, por exemplo, os padrões MVC (*Model-View-Controller*) e Camadas.

Um estilo arquitetural define um conjunto de regras de projeto que identificam tipos de componentes, seus conectores e restrições existentes para a sua composição, os quais podem ser usados para compor uma família de sistemas e subsistemas. Conectores são o meio de comunicação entre componentes de um estilo arquitetural. Cada conector possui a especificação de um protocolo que define suas propriedades, dentre elas: os tipos de interfaces que ele pode intermediar e a ordem em que os eventos acontecem (SHAW & GARLAN, 1996). Em geral, os estilos arquiteturais compreendem um vocabulário comum (e.g. tipos de componentes, tipos de conectores, estruturas de controle, comunicação de dados, interação entre dados e controle, regras e restrições). Exemplos de estilos arquiteturais conhecidos são *Pipes and Filters*, Camadas, Baseado em Eventos e Organização Orientada a Objetos.

Em função da ampla aceitação e maturidade da abordagem, será dado maior ênfase às propostas de estilos e padrões arquiteturais para DBC. Embora o uso de ADLs não esteja contemplado nesta tese, a sua utilização futura pode ser considerada como um recurso adicional para a formalização das ARs. No entanto, mesmo sem esta formalização, a abordagem dá o apoio necessário para a construção de arquiteturas de referências.

A literatura apresenta várias propostas de estilos arquiteturais. Alguns considerados genéricos e que podem, a princípio, ser utilizados por qualquer domínio de

aplicação (e.g. camadas, *pipe and filter* (SHAW & GARLAN, 1996)). Outros estilos são orientados a domínio (EMMERICH *et al.*, 2001) ou orientados ao DBC (HERZUM & SIMS, 2000). Embora exista uma certa dificuldade prática em diferenciar padrões de estilos arquiteturais, observa-se que os estilos são bem mais abstratos. Por exemplo, Camadas é tanto considerado como um estilo arquitetural como um padrão, porém estão representados em níveis de abstração distintos.

Métodos de DBC ((D'SOUZA & WILLS, 1999; BROWN, 2000; HERZUM & SIMS, 2000)) e propostas para arquiteturas de referência baseadas em componentes (COLLINS-COPE & MATTHEWS, 2000) sugerem o uso do estilo arquitetural em camadas. Neste caso, os componentes, que representam diferentes abstrações do domínio (e.g. negócio, processo, infra-estrutura, utilitário), são distribuídos pelas camadas da arquitetura, sugerindo uma organização dos componentes de acordo com o serviço que eles oferecem.

Também podem ser encontrados estilos arquiteturais para DBC, documentados em (HERZUM & SIMS, 2000), e que são conhecidos como estilos baseados em tipos do negócio e estilos baseados em instância. Os componentes gerados a partir destes estilos arquiteturais correspondem aos componentes de negócio discutidos em (D'SOUZA & WILLS, 1999; BROWN, 2000; HERZUM & SIMS, 2000).

O estilo arquitetural baseado em tipos leva a encapsular em um componente todas as instâncias de um tipo de negócio, fornecendo acesso às suas informações através de suas interfaces. Estes componentes são chamados de gerentes de instância. Estes estilos podem ser detectados nos métodos de DBC Catalysis (CHEESMAN & DANIELS, 2000) e UML Components (D'SOUZA & WILLS, 1999).

Teixeira (2003) estende a proposta de Herzum *et al.* (2000) considerando que tipos de negócio relacionados através de herança, composição e agregação podem ser agrupados num único componente de negócio, de acordo com a semântica da relação existente entre os tipos. Neste caso, o componente de negócio pode gerenciar um ou mais tipos de negócio do domínio ou aplicação.

Já o estilo baseado em instâncias (HERZUM & SIMS, 2000), como já diz o próprio nome, permite o acesso diretamente as instâncias dos tipos de negócio. Para este acesso são necessários dois tipos de componentes: o componente de entidade e o componente coleção. Os componentes entidades, que são as instâncias propriamente ditas, e os componentes coleção servem como arcabouço para a criação e gerência de componentes entidade, oferecendo serviços de busca de outros componentes

pertencentes a mesma coleção. No estilo baseado em instâncias, o mapeamento de tipo de negócio para componente é de um-para-um, ou seja, cada tipo terá o seu componente coleção e pelo menos um componente entidade gerado. Neste sentido, as relações existentes entre os tipos de negócio não implicam em uma política de agrupamento de tipos para os componentes gerados.

A utilização do estilo arquitetural em camadas combinado com os estilos baseados em tipos ou instâncias pode representar uma boa alternativa para a construção de uma arquitetura, pois aumenta a flexibilidade do sistema pela definição de diversos níveis de abstração. Por outro lado, geralmente o estilo em camadas diminui o desempenho da arquitetura gerada (SHAW & GARLAN, 1996).

Outros estilos arquiteturais genéricos podem ser considerados na construção de uma arquitetura de componentes. Neste caso, a sua utilização poderia ser auxiliada pelo uso de alguns padrões arquiteturais como os propostos em (SCHMIDT, 2000), os quais dão apoio à persistência entre objetos concorrentes, tratamento de exceções e comunicação síncrona e assíncrona.

Um modelo genérico para a construção de arquiteturas de referência baseado em estilos arquiteturais é proposto em (ROSSAK *et al.*, 1997). De acordo com este modelo, somente depois de escolhido o estilo arquitetural é que devem ser identificados os elementos arquiteturais do domínio. Tais elementos representam a categorização e interconexão dos requisitos do domínio, obtidas com base em princípios e regras de um domínio em questão. Embora tal proposta destaque a importância da escolha do estilo arquitetural, não existe qualquer indicação sobre o processo de escolha deste estilo, a relação entre o estilo e os requisitos da aplicação e, após a escolha, como é conduzida a instanciação do mesmo.

Em (XAVIER, 2001), o autor destaca que a maioria das iniciativas utilizam padrões arquiteturais sem qualquer tipo de critério para justificar a escolha dos mesmos. Na proposta de Xavier (2001), cada padrão arquitetural tem associado um conjunto de requisitos não funcionais, baseados na ISO 9126 (1992), tais como interoperabilidade, tolerância a falhas, modificabilidade, testabilidade, adaptabilidade, etc. Cada padrão arquitetural é avaliado segundo tais requisitos, usando uma escala de "muito bom" a "desconhecido". Esta avaliação partiu de um estudo da literatura e que foi refinado neste contexto em função das experiências acumuladas. Assim, nesta escala, "muito bom" significa que o requisito não funcional é muito bem contemplado pelo padrão arquitetural e "desconhecido" denota que não existem indicadores de que este padrão

possa atender a este requisito não funcional. O resultado desta avaliação é uma lista sugerindo os padrões arquiteturais que melhor atendem aos requisitos não-funcionais escolhidos pelo Engenheiro de Domínio. A partir desta lista de padrões arquiteturais o Engenheiro de Domínio pode escolher um ou mais deles para a sua instanciação em ARs.

Da mesma forma, Bosch (2000) propõe um projeto arquitetural de linhas de produto, indicando a necessidade de avaliação dos requisitos não-funcionais para a imposição de um determinado padrão arquitetural. No entanto, o autor não detalha como esta avaliação é realizada.

2.4.3 - Técnicas para o Refinamento de Componentes em Elementos Arquiteturais numa Arquitetura de Referência

Os estilos arquiteturais discutidos anteriormente propõem soluções para a estruturação de arquiteturas num alto nível de abstração. Cada elemento arquitetural sugerido pelo estilo escolhido deverá coordenar um conjunto representativo de componentes do domínio os quais provavelmente trocam muitas mensagens entre si.

A coesão de componentes através de elementos arquiteturais auxilia a compreensão e organização dos componentes da arquitetura e, sobretudo, facilita a sua manutenção e futura reutilização para diferentes aplicações.

Pesquisas que discutem como avaliar o grau de acoplamento e coesão de seus artefatos visam exatamente melhorar a compreensão do problema, organizando os artefatos que são mais dependentes uns dos outros através de um único artefato agregador (e.g. componente, pacote). Propostas para avaliar o grau de acoplamento no nível de classes (e.g. métricas para avaliar a coesão, acoplamento e complexidade ciclomática (PRESSMAN, 2000) não são suficientes em arquiteturas de componentes, pois neste caso, é necessário avaliar não só classes que compõem a arquitetura interna de um único componente mas, principalmente, a arquitetura que contempla dos os componentes de uma aplicação ou domínio.

Existem algumas propostas na literatura para a avaliação de acoplamento de componentes. Lee *et al.* (2001) propuseram métricas para auxiliar a construção de componentes. Estas métricas avaliam o acoplamento, coesão, herança, dependência, interfaces, granularidade e relacionamentos estruturais de classes candidatas a fazer parte da especificação interna do componente. Cho *et al.* (1998) propõem métricas de acoplamento dinâmico, as quais avaliam o número de mensagens trocadas em tempo de

execução, entre funções que fazem parte de objetos específicos. Enquanto as métricas propostas por Lee *et al.* (2001) são obtidas no nível de classe, as métricas dinâmicas propostas por Cho *et al.* (1998) estão relacionadas ao nível de objeto.

Tendo em vista que num processo de ED existem requisitos que correspondem a diferentes variabilidades e opcionalidades, são necessárias métricas que avaleam o acoplamento de componentes, considerando também estas questões. As métricas de utilização de serviços propostas por Hoek *et al.* (2003) foram apresentadas num contexto de linha de produtos de software, podendo também ser aplicadas em ED. Com estas métricas, componentes são avaliados individualmente pelos serviços providos e requeridos, e depois coletivamente, no contexto da arquitetura da linha de produto[1] , considerando as características de opcionalidade e variabilidade dos componentes.

As métricas propostas por Hoek *et al.* (2003) são as mais adequadas para avaliar o acoplamento de componentes numa arquitetura de referência de um domínio, uma vez que consideram as variabilidades e opcionalidades dos componentes para sugerir o agrupamento de componentes na arquitetura. Embora a avaliação de interfaces fornecidas e requeridas por um componente gere informações importantes para a tomada de decisão de agrupamento de componentes, existem outras relações entre os artefatos do domínio (e.g. rastros entre os diferentes artefatos do domínio os quais são gerados durante de cada artefato) que também podem representar um critério para avaliar o acoplamento de componentes de domínios de aplicações ou LP. Porém estas outras relações sobre a semântica do domínio não são tratadas pelos autores.

Observa-se a complementaridade entre estilos arquiteturais e métricas para refinar uma arquitetura de componentes: os estilos apóiam a geração e organização dos componentes, e as métricas oferecem formas de refinamento que podem estar baseadas em critérios de acoplamento de componentes. Como resultado, a compreensão da arquitetura de referência pode ser facilitada, bem como sua manutenção, aumentando o potencial e a facilidade para sua reutilização em futuras aplicações.

[1] "Linha de Produtos de Software é um conjunto de sistemas de software que compartilham um conjunto de características comuns e controladas, que satisfazem necessidades de um segmento de mercado em particular, e são desenvolvidos a partir de artefatos (*core assets*), de forma predefinida". (SEI/CMU, 2005)

2.5 - Métodos de Engenharia de Domínio e Desenvolvimento Baseado em Componentes

Na literatura técnica podem ser encontradas algumas análises de métodos de ED. Arango (1994) faz uma comparação de métodos de ED com base nos seguintes requisitos: a) o propósito da análise de domínio; b) se os resultados obtidos através dos métodos são confiáveis e de que forma podem ser repetidos e validados para diferentes domínios; c) quais são os passos necessários para a obtenção de sucesso no uso dos métodos; e d) quais são as contribuições, limitações e se existem falhas, quais são os motivos da sua ocorrência. Embora tais critérios sejam indiscutivelmente importantes durante a análise de métodos de ED, nesta comparação o autor não considera a variabilidade inerente à modelagem de domínio. Além disso, não questiona o apoio ao projeto de domínio baseado em componentes, visando a construção de uma arquitetura de referência baseada neste paradigma.

Uma outra análise comparativa de métodos de ED, posterior a de Arango (1994), é apresentada em (FERRÉ & VEGAS, 1999). Os autores avaliam cada método considerando a sua capacidade de reutilização em nível de requisitos, artefatos, processos, experiências e arquiteturas geradas através de cada um, considerando as técnicas empregadas e a representação das variabilidades do domínio para estes fins. Ao considerar a variabilidade como um critério para avaliação dos métodos, foi possível observar que tal requisito não é contemplado em todos os artefatos do domínio, oferecendo maior apoio, quando existente, à análise de domínio, em particular através de modelos de características. Também foi possível observar que as técnicas disponíveis para a geração de arquiteturas se concentram, quando tal apoio é existente, no uso de padrões de projeto. Portanto, nenhuma contribuição de abordagens de DBC é mencionada pelos autores.

Em (BRAGA, 2000), é feita uma análise de métodos de ED e DBC baseado no *framework* de comparação utilizado no estudo de viabilidade do método FODA (KANG *et al.*, 1990). Este *framework* apresenta os três principais aspectos que, segundo os autores, devem ser considerados na avaliação de um método: a) a existência de um *processo*; b) os *produtos* gerados pelo método, como são representados e aplicáveis ao desenvolvimento de um domínio de aplicações e; c) as *ferramentas para automatizar o processo*. Observa-se que este conjunto de critérios é bastante genérico, não permitindo uma análise mais precisa quando ao apoio oferecido à representação de variabilidades e opcionalidades de um domínio e a construção de arquiteturas de referência. A avaliação

a respeito dos métodos de DBC também foi limitada, pois os aspectos do *framework* de comparação não permitem evidenciar o apoio oferecido ao desenvolvimento de componentes reutilizáveis, e tampouco ao desenvolvimento de arquiteturas de componentes para a reutilização.

Tendo em vista a importância da representação de variabilidades e opcionalidades em ED, Van der Massen e Lichter (2002) propõem um conjunto de critérios para analisar a especificação destas propriedades em modelos de características. No entanto, também é necessário representar tais variabilidades e opcionalidades nos demais artefatos do domínio e, sobretudo, dar apoio à manutenção destas propriedades nestes outros artefatos que são também considerados na criação de arquiteturas de referência, na instanciação de novas aplicações derivadas do domínio.

Uma limitação comum às propostas discutidas anteriormente é a inexistência de um critério que considere a independência de tecnologia nos modelos gerados pelos métodos. Além disso, também não existem critérios que avaliem como os métodos dão apoio à geração de artefatos implementacionais, a partir daqueles criados durante as atividades previstas pelo método.

Tendo em vista as propostas de análise de métodos de ED e DBC anteriormente citadas, e suas limitações quanto ao suporte ao DBC e modelagem de variabilidades, é proposto um novo conjunto de critérios para avaliação de métodos de ED e DBC.

Os critérios para apoio de DBC no contexto de ED são:

- Modelagem de variabilidades e opcionalidades em diferentes níveis de abstração. A modelagem de variabilidades e opcionalidades é especificada na fase de levantamento de requisitos, geralmente através de modelos de características. Além disso, esta modelagem deve ser também especificada nos demais artefatos do domínio, mantendo esta semântica nestes diferentes níveis de abstração;
- Manutenção da ligação e consistência entre os artefatos do domínio de diferentes níveis de abstração. Durante a ED, diversos artefatos são gerados com base num mesmo conjunto de requisitos. Neste sentido, deve ser mantida esta ligação existente entre os artefatos de diferentes níveis de abstração do domínio, relacionados a um mesmo conjunto de requisitos. Ao manter esta ligação, é possível indicar que tanto a consistência de um requisito do domínio, bem como sua respectiva variabilidade e

opcionalidade, são mantidos nos demais artefatos do domínio a ele relacionado.

- Organização dos artefatos para a composição de elementos arquiteturais. Métodos de ED e DBC devem fornecer sistemáticas para a organização de seus artefatos em elementos arquiteturais, os quais compõem uma arquitetura de componentes. Para isso é esperado o uso de critérios, métricas ou outras técnicas para apoio a criação destes elementos arquiteturais.
- Uso de princípios de DBC na modelagem do domínio. É indicado o uso de uma abordagem de DBC por métodos de ED. Tal indicação se deve ao fato de que o DBC dá apoio a uma série de características técnicas que promovem à reutilização, conforme proposto por Vitharana *et al.* (2003) e discutido na Seção 2.3.1 desta Tese. Dentre as características para apoio à reutilização, se destacam o acoplamento entre componentes de um domínio e aplicações, e a coesão de um componente em relação ao seu domínio de componentes.
- Necessidade de processos de ED com apoio ao DBC. Este requisito é semelhante ao proposto no framework de avaliação de Kang *et al.* (1990), denotando a necessidade de um processo que dê efetivamente apoio a todas as fases da ED (i.e. análise, projeto e implementação do domínio), com o uso de princípios de DBC. Além disso, tais atividades, incluindo aquelas desenvolvidas durante a análise e projeto do domínio, devem ser previstas de tal forma que nenhuma especificação de artefatos seja feita com base em alguma tecnologia de componentes específica. Através de um processo, a ED pode tornar-se mais produtiva, uma vez que são indicadas a execução de atividade que efetivamente apóiam a construção dos artefatos do domínio.
- Geração de código fonte na linguagem de programação empregada por uma tecnologia de DBC. Conforme o requisito anterior, modelos de análise e projeto de domínio devem ser gerados sem vínculo tecnológico, visando aumentar o seu potencial de reutilização. Neste sentido, este requisito indica a necessidade de utilizar alguma sistemática que apóie a geração de código fonte, de acordo com uma tecnologia de componentes escolhida. Assim, através deste requisito é possível contemplar a fase de implementação do domínio, prevista na ED.

- Modelagem e reutilização de artefatos e modelos através de ferramental apropriado. Este requisito sugere a necessidade de uma infra-estrutura de reutilização que possa dar apoio integrado ao ferramental possivelmente existente de apoio aos critérios anteriores. É importante destacar esta necessidade de integração, uma vez que, do ponto de vista de reutilização, os artefatos gerados em todas as fases do processo estão relacionados a um conjunto de requisitos do domínio, que por sua vez poderá ser reutilizado para uma dada aplicação. Assim, caso não exista esta integração entre as diferentes ferramentas, a reutilização dos artefatos poderá ser prejudicada, limitando-se à alguma fase ou atividade específica de um processo.

A seguir é apresentada uma análise dos principais métodos de ED e DBC, considerando tais critérios.

2.5.1 – Análise dos Métodos de ED e DBC

Tendo em vista os critérios anteriormente identificados são analisados os principais métodos de ED e DBC existentes na literatura técnica.

O método mais conhecido e considerado como precursor dos demais é o FODA (KANG *et al.*, 1990). O ponto forte do método é o apoio à fase análise de domínio, contemplando atividades de análise de contexto e a modelagem do domínio. No entanto, não existe ferramental apropriado para o desenvolvimento das atividades desta fase. Conceitualmente, é previsto o apoio à construção de DSSA, mas a literatura disponível trata desta questão de maneira superficial. Neste sentido, os requisitos estabelecidos que abordam a construção de DSSAs não são atendidos pelo FODA. Outro aspecto negativo do método é a falta de apoio ao DBC e tampouco implementação de código fonte baseado em alguma tecnologia específica.

Conforme citado anteriormente, vários métodos foram surgindo a partir do FODA e, dentre eles, o que mais tem se destacado é o FORM (KANG *et al.*, 2002), o qual tem foco em Linha de Produto de Software (LP) baseado em componentes. Assim como o método FODA, o FORM dá maior ênfase às atividades de análise de domínio. Embora não existam relatos de experiência no uso do FORM, a literatura ressalta uma pequena melhora do apoio ao ciclo completo de ED comparado ao método original FODA (WERNER & BRAGA, 2005). No método FORM, existe a preocupação na modelagem de especificação das diferentes características de um domínio, além de um maior detalhamento na representação de variabilidades no domínio, através de regras de

composição. É disponibilizada uma ferramenta para apoio à modelagem de características. No entanto, as variabilidades especificadas neste modelo não são consideradas durante a geração de componentes no projeto da LP. Também não está disponível na literatura como podem ser gerados os rastros existentes entre os artefatos do domínio. Uma outra observação quanto ao FORM é o apoio parcial à fase de projeto. Não é claro como ocorre a especificação dos componentes gerados a partir do modelo de componentes da LP e também como são estruturados os componentes para a composição de elementos arquiteturais. Neste caso, é impossível afirmar o apoio existente para a construção de ARs. Um outro problema identificado através da análise de exemplos é que a especificação de componentes é feita de forma dependente de tecnologia, e não existe uma descrição de como o processo orienta o Engenheiro a obter tal especificação. Por último, o FORM não possui um ambiente específico que possa dar apoio integrado às atividades previstas pelo método, restringindo a sua reutilização.

Um outro processo de ED é o proposto em (BRAGA, 2000). O processo, denominado Odyssey-DE (*Domain Engineering*), foi desenvolvido no contexto da infra-estrutura de reutilização Odyssey (WERNER *et al.*, 1999) e, se comparado aos demais, é o processo que contempla o maior número de requisitos estabelecidos na Seção 2.5. Este processo tem o propósito de unir os aspectos de reutilização e entendimento do domínio que são providos pelos processos de ED existentes e o detalhamento do desenvolvimento de componentes, provido pelos processos de DBC. Ainda que aborde o ciclo de vida completo de ED, observa-se que o foco do processo ainda é na análise de domínio. No tocante à modelagem de variabilidades, apresenta as seguintes limitações:

a) a notação gráfica do modelo de características é limitada. Não existe diferença na representação de pontos de variação, variantes e invariantes.

b) as dependências e exclusão mútua, previstas na notação de Miler (2000), não são graficamente representadas de maneira intuitiva, utilizando a mesma simbologia para os dois tipos de restrições. Assim, a função do modelo de características de auxiliar o Engenheiro de Domínio através de facilidades visuais fica restrita.

Outra limitação do Odyssey-DE é que ele não apóia a especificação e representação de variabilidades para os demais artefatos de análise do domínio, o que pode tornar os demais artefatos do domínio inconsistentes sob este ponto de vista. Embora existam estas limitações na análise de domínio, para as funcionalidades disponíveis existe um ferramental de apoio, através do ambiente Odyssey.

O apoio ao projeto de domínio no Odyssey-DE é parcial. Embora o processo se proponha a dar apoio ao DBC, existem algumas restrições quanto ao projeto arquitetural em ED. O modelo de componentes contempla componentes e suas interfaces. No entanto, não é possível desenvolver a especificação interna desses componentes previsto no processo. Além disso, também não existe apoio à especificação de variabilidades, opcionalidades e regras de composição.

Outro aspecto é com relação ao apoio oferecido para a geração dos componentes, disponibilizado pelo Odyssey-DE, através da proposta de Teixeira (2003). Apesar de serem utilizadas regras de acoplamento dos tipos de negócio para a geração de componentes de negócio, em casos de domínios grandes, a quantidade de componentes gerados tende a ser elevada. A proposta de projeto arquitetural de Braga (2000) não apresenta soluções para o agrupamento de componentes visando à construção de elementos arquiteturais para composição de arquiteturas de referência. Neste sentido, o apoio à construção de ARs é limitado.

Um aspecto positivo do Odyssey-DE é que os componentes gerados são independentes de tecnologia de implementação, podendo auxiliar numa reutilização futura. Por outro lado, não existe qualquer apoio à implementação de componentes para uma tecnologia específica. Existe apoio à geração de classes em linguagem Java para cada componente do domínio. No entanto, considerando uma tecnologia de componentes, este mapeamento nem sempre ocorre de um componente independente de tecnologia, para somente um outro componente de uma tecnologia específica.

Além de métodos de ED, existem métodos de DBC que visam dar apoio à reutilização, dentre eles, se destacam Catalysis, UML Components e Kobra.

Catalysis é um método de DBC que utiliza a notação da UML, principalmente para a representação de classes, casos de uso, interação, colaboração e de pacotes (D'SOUZA e WILLIS, 1999). Para tanto, o método procura dar apoio: a) ao DBC, na definição das interfaces dos componentes e nos refinamentos sucessivos deste processo de construção; b) ao processo de integração dos subsistemas de um mesmo modelo de negócio; c) à integridade do processo de refinamento dos requisitos do negócio através das especificações dos componentes, pré e pós-condições e; d) ao projeto orientado a objeto, mais especificamente, no uso coerente e consistente da notação UML no processo de projeto.

No que se refere à reutilização, o Catalysis não apresenta um conjunto de elementos que definam o quê e como reutilizar. Não existe apoio à especificação de

variabilidades e opcionalidades nos componentes especificados. Neste sentido, o método dá mais apoio ao desenvolvimento de uma aplicação específica mas não a um domínio de aplicações. Embora pareça estar contemplado no estágio de projeto da arquitetura, não existe apoio explícito à especificação de agrupamentos dos componentes para a geração de elementos arquiteturais.

Um outro aspecto também negativo do Catalysis é a vinculação tecnológica sugerida no estágio de projeto da arquitetura, através da escolha de uma plataforma. Com base nesta escolha é que o estágio posterior, de projeto interno dos componentes, será desenvolvido. Neste sentido, o projeto interno dos componentes pode vir a se tornar obsoleto com o tempo, caso a plataforma escolhida se torne também obsoleta, ou o usuário necessite trocar a plataforma ou inda outros fatores que podem comprometer a reutilização destes artefatos. Em função deste vínculo tecnológico, o método apóia a fase de implementação, embora não exista explicitamente qualquer indicação de geração de código fonte na literatura pesquisada.

A última limitação identificada é que o Catalysis não propõe ferramentas que apóiem especificamente o método, mas é sugerido o uso de ferramentas CASE para o seu desenvolvimento. No entanto, não se pode garantir que tais ferramentas possuam funcionalidades para à construção de todos os modelos, tal e qual foram sugeridos pelo método.

UML Components é um outro método para DBC que utiliza os recursos da UML para o desenvolvimento de sistemas (CHEESMAN & DANIELS, 2000), o qual contempla etapas de Requisitos, Especificação, Provisão, Montagem, Teste e Utilização. Observa-se que o método UML Components possui maior foco nas atividades que envolvem a especificação de componentes e interfaces, mas não detalha o processo de montagem da arquitetura de componentes e composição destes componentes através de elementos arquiteturais. Do ponto de vista dos critérios apresentados no início da Seção 2.5, se observa que o UML Components não foi proposto visando o desenvolvimento para reutilização. Não existe qualquer preocupação em identificar quais são as variabilidades e opcionalidades dos requisitos do sistema. Assim, se conclui que o foco é no desenvolvimento de um sistema, sem preocupação com a reutilização dos artefatos gerados. O apoio oferecido à especificação dos componentes de um sistema é independente de tecnologia de implementação, mas por outro lado não é observada qualquer preocupação na etapa de montagem em auxiliar a geração de componentes em outros, já especificados numa tecnologia de DBC. Embora seja prevista a criação da

arquitetura do sistema, não existem indícios de organização destes componentes em elementos arquiteturais.

Embora UML Components não proponha um ferramental específico para apoio às suas etapas, podem ser utilizadas ferramentas CASE que dão apoio a UML. De qualquer forma, não é garantido que ferramentas genéricas possuam funcionalidades que contemplem o nível de detalhe requerido pelo método, em especial nas atividades de especificação, provisão e montagem.

O último método de DBC analisado, voltado para LP é o Kobra (ATKINSON *et al.*, 2002). Uma das principais características de Kobra é a separação da descrição abstrata de um componente de sua implementação. Outra característica que o difere dos demais métodos de DBC, é a premissa de que todo o artefato UML (e.g. requisitos, arquiteturas, projeto) usado no desenvolvimento de um sistema deve ser estruturado e organizado em torno de componentes essenciais de um sistema, os quais são denominados de *komponents*. Tal premissa não é um consenso na literatura, pois algumas abordagens partem do pressuposto que outros artefatos de análise devem ser utilizados para identificar componentes para a aplicação. Por exemplo, Braga (2000), através do processo Odyssey-DE, sugere que componentes de negócio sejam criados a partir de tipos de negócio do domínio.

Dentre os métodos de DBC, o Kobra é aquele que mais detalha o apoio ao projeto arquitetural de seus componentes, com orientações que guiam a construção de suas especificações internas e, de uma maneira geral, contempla o maior número de requisitos identificados para a avaliação dos métodos de ED e DBC.

No entanto, algumas limitações foram detectadas. O conjunto de informações, registradas em modelos de decisão que orientam a especificação de variabilidades da LP, não são de fácil compreensão e, em conseqüência, não fica claro como as informações registradas estão contempladas nos diagramas que compõem um *kcomponent*.

Observa-se uma forte ênfase do método na construção de cada *komponent*. Porém, não fica claro de que forma estes *komponents* se relacionam no contexto de uma arquitetura de *komponents* para uma LP (e.g. uso de padrões arquiteturais, de projeto, estilos arquiteturais). O método utiliza vários modelos da UML para a representação da especificação de *komponents* (e.g. modelos de classe, colaboração), mas não utiliza o modelo de componentes para representação da arquitetura de *komponents* da LP. O único artefato que pode indicar os produtos existentes na LP é o modelo de decisão, o

qual consiste de um conjunto de tabelas em que cada requisito é descrito, indicando qual ou quais produtos da linha utilizam tal requisito e que outros requisitos são necessários para a sua utilização.

Não existe qualquer orientação a respeito da implementação de código fonte dos *komponents* desenvolvidos pelo método, considerando a linguagem utilizada por uma tecnologia de DBC específica.

Tendo em vista a análise sobre estes métodos (i.e. FODA, FORM, Odyssey-DE, Catalysis, UML Components e Kobra), se observa que tanto a ED quanto o DBC e, sobretudo, a combinação destas duas abordagens, são carentes de soluções que privilegiam o apoio à reutilização.

2.6 - Considerações Finais

Este capítulo apresentou os conceitos básicos sobre as abordagens de reutilização de ED, Arquitetura de Software e DBC que são importantes para contextualizar a abordagem de projeto arquitetural apresentada nesta Tese. Além disso, foi apresentada uma análise dos principais métodos propostos para tais abordagens. A análise destes métodos foi baseada num conjunto de critérios estabelecidos para apoio ao projeto baseado em componentes, com apoio à reutilização.

A Tabela 2.2 mostra um resumo da análise desenvolvida na Seção 2.5, organizando os métodos em ordem cronológica de publicação e classificadas quanto ao tipo de abordagem (i.e. ED, DBC, ED com DBC). O objetivo desta Tabela é não exatamente avaliar quanto ao suporte existente ou não com relação aos critérios, mas sobretudo mostrar o quanto tais abordagens têm se preocupado com o apoio oferecido a tais critérios. A Tabela mostra que desde o período em que o método FODA foi proposto até as últimas propostas (e.g. Kobra, FORM), pouco evoluiu o suporte indicado pelos critérios de avaliação. Considerando que a Tabela mostra contribuições apresentadas num período de um pouco mais de uma década, se conclui que pesquisas que indiquem suporte aos critérios de avaliação devam ser consideradas como contribuições importantes nas áreas de ED e DBC.

Odyssey-DE, FODA e FORM dão apoio parcial à modelagem de variabilidades, pois limitam-se ao modelo de características do domínio e, ainda, com as restrições descritas na Seção 2.5. UML Components e Catalysis não dão apoio à modelagem de variabilidades e opcionalidades, pois se limitam à construção de aplicações específicas. Kobra apóia a modelagem de variabilidades e opcionalidades através dos modelos de

decisão. No entanto, não contempla tais especificações através de seus *komponents.* Contudo, é esperado que a modelagem de variabilidades, opcionalidades, e relacionamentos dos requisitos funcionais do domínio sejam devidamente especificados em todos os seus artefatos, desde aqueles gerados durante a fase de análise até o projeto do domínio.

Tabela 2.2: Critérios para avaliação de Métodos de ED e DBC

Critérios de Avaliação		Abordagens de ED		Abordagens de DBC		Abordagens de ED com suporte ao DBC	
		FODA (1990)	FORM (2002)	Catalysis (1999)	UML Components (2000)	Odyssey-DE (2000)	Kobra (2000)
1	Modelagem de variabilidades e opcionalidades em diferentes níveis de abstração	Parcial	Parcial	Não	Não	Parcial	Parcial
2	Manutenção das ligações e consistência entre os diferentes artefatos do domínio	Não	Não	Não	Não	Parcial	Parcial
3	Organização dos artefatos para a composição de elementos arquiteturais	Não	Não	Parcial	Parcial	Não	Sim
4	Uso de princípios de DBC na modelagem do domínio	Não	Parcial	Parcial	Parcial	Parcial	Sim
5	Necessidade de processos de ED com apoio ao DBC	Sim	Sim	Não	Não	Sim	Sim
6	Geração de código na linguagem de programação empregada por uma tecnologia de DBC	Não	Não	Não	Não	Parcial	Não
7	Modelagem e Reutilização de Artefatos e Modelos através de um ferramental apropriado	Não	Parcial	Não	Parcial	Parcial	Não

Quanto ao segundo requisito da Tabela 2.2, somente Odyssey-DE e Kobra apresentam apoio parcial à manutenção de rastros e sua consistência entre os diferentes artefatos do domínio. Em Odyssey-DE é dado apoio ao rastreamento entre os artefatos. No entanto, não existe uma sistemática que automatize este rastreamento, restringindo a garantia de consistência do mesmo em relação à semântica do domínio. Kobra dá apoio ao rastro entre artefatos entre um *komponent* e os demais artefatos estruturais e comportamentais a ele relacionados. Porém não é garantido o rastro entre elementos estruturais e comportamentais relacionados a um mesmo *komponent*. Os demais métodos não dão apoio a este requisito. No entanto, é importante a manutenção dos rastros entre os artefatos do domínio para obter-se um registro dos diferentes níveis de abstração de um requisito funcional durante a modelagem do domínio. Além disso, a consistência deste rastro, através de sistemáticas apropriadas, garante que as

propriedades de um determinado requisito do domínio sejam mantidas em seus diferentes níveis de abstração.

Com relação ao terceiro requisito da Tabela 2.2, Kobra é o único método de DBC que se preocupa parcialmente com a geração de elementos arquiteturais. Isso porque os demais artefatos do domínio estão associados a seu respectivo *komponent* que, por sua vez, pode vir a representar ou não um elemento arquitetural da linha de produto. Os demais métodos de ED e DBC não contemplam sistemáticas para tal apoio. Contudo, a geração e composição de elementos arquiteturais tende a melhorar a organização e estruturação da arquitetura de componentes de um domínio. Em nenhum dos métodos analisados existe explicitamente um apoio a identificação de requisitos não-funcionais de um domínio e, também, não estabelecem atividades e orientações para a construção de arquiteturas de referência.

Kobra também é o único método que contempla o requisito de apoio à modelagem baseada em componentes. Catalysis, UML Components dão apoio ao DBC, porém para a EA e não a ED. Odyssey-DE contempla parcialmente tal requisito, pois se limita a apoiar a geração de componentes de negócio, sem considerar a sua especificação interna, a geração de outras categorias de componentes do domínio e tampouco a relação de componentes de negócio com estas outras categorias. Uma abordagem de projeto arquitetural de domínio, baseada em componentes, deve prover sistemáticas para apoiar a geração de todas as categorias de componentes do domínio e seus respectivos relacionamentos.

Todos os métodos de ED sugerem atividades para apoio às suas fases. Porém, alguns deles com foco nas atividades de análise como discutido anteriormente. UML Components e Catalysis não possuem processo para ED, porém dão apoio a EAs especificas. Esta avaliação confirma a necessidade de se propor processos que orientem tanto o Engenheiro de domínio quanto o de Aplicações nas suas respectivas tarefas de modelagem.

Conforme descrito na Tabela 2.2, a maioria dos métodos não se preocupa em gerar componentes de uma tecnologia específica a partir dos componentes por ele gerados. Somente Odyssey-DE gera codificação em linguagem Java para componentes do domínio, mas tal codificação não é gerada de tal forma que seja compatível à codificação esperada por uma tecnologia de DBC. Contudo, é importante a utilização de sistemáticas que apóiem o mapeamento de componentes gerados através destes métodos em outros componentes, especificados conforme uma tecnologia específica. Tais

sistemáticas podem auxiliar na consistência das especificações dos componentes gerados pelos métodos, além do que a devida transformação destes componentes em um ou mais artefatos da tecnologia que correspondam ao mesmo. Por exemplo, considere um componente de negócio gerado pelo Odyssey-DE e a tecnologia de componentes EJB. Tal componente deve ser mapeado, em EJB, para um componente Entity Bean e mais quatro interfaces: três locais (EJB Local Interface, EJB Home Interface e EJB Local Home Interface) e uma remota (EJB Remote Interface).

Não existe apoio oferecido pelos métodos de ED e DBC, bem como uma infra-estrutura de reutilização para apoio a todas as atividades previstas no ciclo de vida. Somente o Odyssey-DE, como relatado anteriormente, possui este apoio à reutilização, a qual contempla um conjunto de ferramentas para apoio *com* e *para* reutilização (MILER, 2000; DANTAS *et al.*, 2001; DANTAS *et al.*, 2002; BLOIS *et al.*, 2005b). Embora FORM e UML Components possuam apoio parcial para ferramentas de análise, tais propostas não estão contempladas numa infra-estrutura de reutilização.

Tendo em vista as lacunas detectadas nos métodos de ED e DBC, nos próximos Capítulos será apresentada uma abordagem para apoio à ED com o foco no DBC. Esta abordagem propõe apoio ao projeto arquitetural de componentes, visando à construção de uma arquitetura de software específica do domínio que seja consistente com as variabilidades dos requisitos funcionais do domínio e que atendam a um conjunto de requisitos não-funcionais esperados. Para tal, é proposta uma notação para a modelagem de variabilidades e opcionalidades e um conjunto de heurísticas que norteiam o mapeamento de tais variabilidades e opcionalidades para os diferentes artefatos do ciclo de vida do domínio. Além disso, é proposto um conjunto de critérios para a organização de componentes em termos de elementos arquiteturais do domínio, visando a construção de arquiteturas de referência. Além disso, é proposta no contexto desta pesquisa uma abordagem de transformação para o mapeamento dos modelos gerados durante o projeto de domínio, originalmente independentes de tecnologia, para uma tecnologia de componentes específica.

Capítulo 3 – DECOM: uma Abordagem de ED com suporte ao DBC

3.1 - Introdução

No capítulo 2, foram analisados métodos de ED propostos na literatura. Estes métodos possuem sistemáticas para a modelagem de artefatos na ED, visando a reutilização na EA. Embora existam diferenças entre os métodos, eles são semelhantes quanto às fases envolvidas no processo de ED (e.g. identificação e escopo do domínio, análise, projeto e implementação do domínio). No entanto, algumas restrições foram identificadas nos processos definidos, conforme relatado, através de um conjunto de requisitos, nas Seções 2.5 e 2.6.

Se por um lado a ED visa identificar aspectos variáveis e invariáveis de um domínio para sua reutilização, por outro o DBC propõe soluções reutilizáveis para o processo de desenvolvimento de software, podendo disponibilizar aos usuários os aspectos variáveis e invariáveis do domínio através de componente e suas interfaces (BROWN, 2000).

Na Seção 2.4.2 foram discutidas as propostas existentes na literatura que apóiam a construção de arquiteturas de software e, em especial, para arquiteturas baseadas em componentes. Na ED, o que os autores têm pesquisado é a possibilidade de criação de uma arquitetura de referência do domínio, ou possíveis ARs.

Conforme já discutido, se pressupõe que uma AR contemple requisitos funcionais e não funcionais do domínio. No entanto, é difícil ter certeza de que uma arquitetura atende a todos os requisitos de um domínio, uma vez que os produtos, processos, enfim, um domínio como um todo está em constante evolução. A geração de uma AR é uma tarefa genuinamente complexa e, por mais que existam propostas para este fim, é difícil uma solução que seja completa, para qualquer domínio de aplicações.

Baseado na análise da literatura sobre construção de AR e nas lacunas ainda existentes nas abordagens de ED e DBC, a abordagem DECOM propõe um apoio ao projeto arquitetural baseado em componentes para o contexto de ED, a qual contempla cinco elementos principais, conforme apresenta a Figura 3.1.

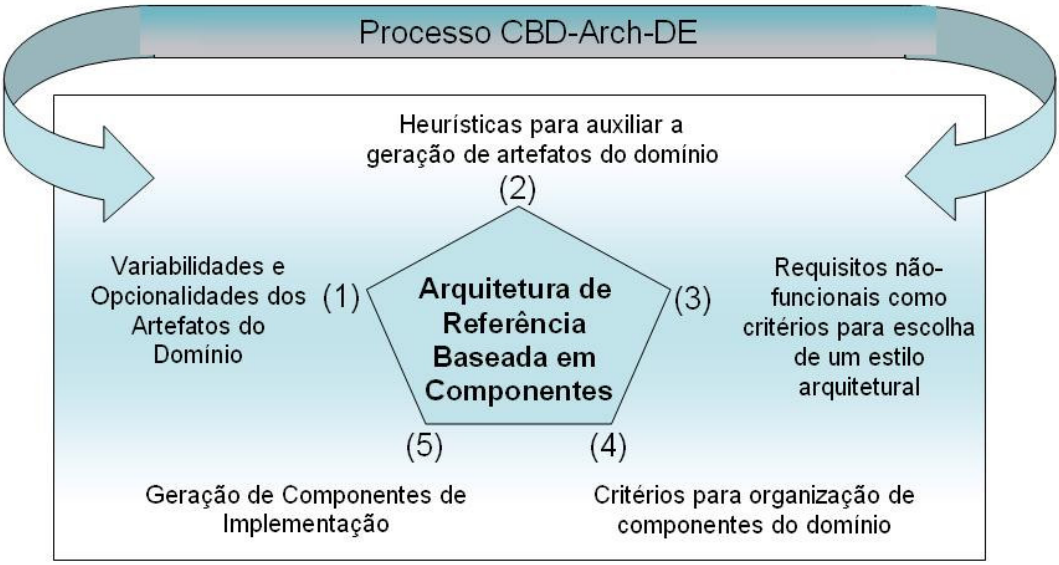


Figura 3.1: Elementos da Abordagem DECOM

A numeração de 1 a 5 na Figura 3.1 não exige uma ordem esperada para contemplar o apoio à construção de uma AR. No entanto, por questões de níveis de abstração, esta indicação numérica muitas vezes faz sentido durante a sua construção.

O elemento número 1 da Figura 3.1 indica a necessidade de manter a consistência das variabilidades e opcionalidades dos requisitos funcionais do domínio. Assim, a cada decisão de geração de componentes e distribuição dos mesmos em elementos arquiteturais, devem ser respeitadas tais propriedades que determinam, muitas vezes, os diferentes produtos (aplicações) derivados de um domínio. Para oferecer tal apoio, esta Tese utiliza a notação Odyssey-FEX, proposta por (OLIVEIRA, 2006), para a representação de variabilidades e opcionalidades em modelos de características. Além disso, no contexto desta tese, é feita uma extensão das notações dos demais artefatos no domínio para dar apoio à representação destas propriedades. A notação Odyssey-FEX e demais extensões são apresentadas na Seção 3.2. Maiores detalhes da notação Odyssey-FEX podem ser obtidos em (OLIVEIRA *et al.*, 2005; OLIVEIRA, 2006).

O segundo elemento da Figura 3.1 indica a necessidade de sistemáticas que auxiliem a criação e especificação de artefatos do domínio. Para tal, esta Tese propõe um conjunto de heurísticas que auxiliam no mapeamento das variabilidades, opcionalidades e relacionamentos existentes de um artefato do domínio para o outro, até a construção dos componentes (BLOIS *et al.*, 2005c) e, finalmente, dos elementos

arquiteturais (BLOIS *et al.*, 2005b) que devem compor uma AR. Estas heurísticas serão detalhadas na Seção 3.4.

O elemento número 3 da Figura 3.1 evidencia a necessidade de um apoio a identificação de estilos arquiteturais a serem utilizados na especificação da AR, com base nos requisitos não-funcionais do domínio. Embora esta abordagem não apóie por completo as atividade de criação de AR, algumas iniciativas são observadas para tal suporte. Tais iniciativas serão discutidas na Seção 3.3.2.3.

O quarto elemento da Figura 3.1 propõe critérios para a identificação e organização dos componentes modelados no domínio (BLOIS *et al.*, 2005b) através de elementos arquiteturais, visando uma melhor estruturação dos componentes do domínio através da AR. Estes critérios estão detalhados na Seção 3.5.

Por último, o quinto elemento da Figura 3.1 indica a necessidade de apoio a transformação dos componentes lógicos, criados durante as atividades anteriormente mencionadas, em outros componentes, implementados numa tecnologia de componentes em particular. Para esta transformação é utilizada a abordagem Odyssey-MDA, desenvolvida por (MAIA, 2006). O uso desta abordagem de transformação é descrito na Seção 3.3.3, no contexto do processo CBD-Arch-DE, e na Seção 3.6 que contempla uma breve descrição sobre a mesma. Maiores detalhes da abordagem são obtidos em (MAIA *et al.*, 2005a; MAIA *et al.*, 2005b; MAIA, 2006).

Conforme destaca a Figura 3.1, estes cinco elementos estão organizados em função das atividades principais previstas no processo CBD-Arch-DE (BLOIS *et al.*, 2004), proposto nesta Tese, o qual é detalhado na Seção 3.3.

Para auxiliar o leitor no entendimento de toda a abordagem DECOM será utilizado o domínio de Telefonia Móvel como exemplo, o qual está brevemente descrito no Anexo 1. Através do exemplo o leitor terá uma visão completa e integrada dos elementos que compõem esta abordagem de projeto arquitetural.

Finalmente na Seção 3.7 apresentamos as considerações finais sobre este capítulo.

3.2 - Notação para a Modelagem de Variabilidades e Opcionalidades de um Domínio

Em (OLIVEIRA, 2006), foi identificado um conjunto de requisitos necessários para uma representação completa das variabilidades e opcionalidades de um domínio.

Através destes requisitos, foi feita uma análise das propostas de representação de variabilidades existentes na literatura, em particular, no contexto de modelos de características, a saber, (KANG *et al.*, 1990; GRISS *et al.*, 1998; CLAUSS, 2001; KANG *et al.*, 2002; RIEBISCH *et al.*, 2002; BOSCH, 2004)), obtendo-se os seguintes resultados:

- Em geral, as notações apresentam formas diversificadas para a representação de um mesmo requisito de variabilidade e opcionalidade.
- Falta de integração desta notação a um ambiente de reutilização de software.
- Falta de propagação da variabilidade e opcionalidade para os demais artefatos do domínio relacionados a um dado requisito. Tais representações limitam-se aos artefatos de análise, em particular, modelos de características.

Com base nestes requisitos, foi proposto um meta-modelo de características (OLIVEIRA *et al.*, 2005), o qual é composto de um conjunto de regras de boa formação para modelos de características. Este meta-modelo deu origem à notação Odyssey-FEX (BLOIS *et al.*, 2005a), detalhada em (OLIVEIRA, 2006), que será utilizada nesta Tese para a modelagem de características do domínio.

Ainda, no contexto desta Tese, é proposta uma extensão da notação original (OMG, 2005) dos modelos de tipos de negócio, casos de uso e componentes com o intuito de permitir a especificação de suas variabilidades e opcionalidades em um domínio.

Estas notações são apresentadas nas próximas Subseções 3.2.1 e 3.2.2.

3.2.1 – Notação para Modelagem de Variabilidade no Modelo de Características do Domínio

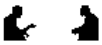
A notação Odyssey-FEX (OLIVEIRA, 2006) é voltada à atividade de levantamento de requisitos. Esta notação define uma semântica completa das características que representam conceitos, funcionalidades e tecnologias de um domínio, incluindo sua variabilidade, opcionalidade, bem como os relacionamentos entre eles.

As características na notação Odyssey-FEX podem ser classificadas segundo três critérios ortogonais, a saber, Categoria, Variabilidade e Opcionalidade, além de possuírem qualificações adicionais que podem fortalecer a semântica do modelo a ser construído.

As características podem ser categorizadas como: de domínio, de entidade, ambiente operacional, tecnologia de domínio e técnica de implementação, sendo que

estas três últimas são também classificadas como características tecnológicas. A descrição de cada tipo de característica e sua representação gráfica, através de ícone e estereótipo, está descrita na Tabela 3.1.

Tabela 3.1: Categorias de Características na notação Odyssey-FEX

Ícone	Estereótipo	Categorias de Características	
	<<functional>> ou <<conceptual>>	**Características de Domínio** – Características intimamente ligadas à essência do domínio. Representam as funcionalidades, denominadas de **Características Funcionais**, que correspondem a casos de uso, ou conceitos do domínio, denominadas de **Características Conceituais**, e que correspondem a artefatos estruturais concretos.	
	<<actor>>	**Características de Entidade** – São os atores do modelo. Entidades do mundo real que atuam sobre o domínio. Podem, por exemplo, expor a necessidade de uma interface com o usuário ou de procedimentos de controle.	
	<<operational environment>>	**Características de Ambiente Operacional -** Características que representam atributos de um ambiente que uma aplicação do domínio pode usar e operar. Ex: tipo de terminal, sistemas operacionais, bibliotecas etc.	**Características Tecnológicas**
	<<domain technology>>	**Características de Tecnologia de Domínio -** Características que representam detalhes de implementação de mais baixo nível, específicos para o contexto de um domínio. Ex: métodos de navegação em um domínio de aviões.	
	<<implementation technique>>	**Características de Técnicas de Implementação –** Características que representam detalhes de implementação de mais baixo nível, contudo de cunho mais genérico que as outras características tecnológicas de domínio. Ex: técnicas de sincronização.	

As características de domínio existem em todas as notações estudadas. Porém, a classificação de características funcionais, conceituais e de entidade são originárias da notação proposta por Miler (MILER, 2000). Características de ambiente operacional, tecnologias de domínio e técnicas de implementação foram originalmente propostas por Lee, Kang e Lee (LEE *et al.*, 2002) e Kang, Lee e Donahue (KANG *et al.*, 2002). Tal classificação foi mantida na notação Odyssey-FEX, permitindo que tais aspectos sejam também considerados no modelo de características durante a análise de domínio.

Além da categoria de características apresentada na Tabela 3.1, existem características adicionais que são consideradas como propriedades daquelas já definidas. Tais propriedades são apresentadas na Tabela 3.2.Tabela 3.2

A propriedade "**Externa**" pode ser atribuída a qualquer tipo de característica definida anteriormente, uma vez que em domínios externos pode haver tanto características de análise quanto tecnológicas.

A propriedade "**Não-Definida**" se aplica a todos os tipos de características, exceto à característica de Entidade, uma vez que esse tipo de característica não é detalhado nos modelos de mais baixo nível de abstração.

Tabela 3.2: Propriedades das características na notação Odyssey-FEX

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

A propriedade "**Organizacional**" só se aplica às características de Domínio, pois esse tipo de característica representa os conceitos e funcionalidades do domínio, cujo entendimento precisa ser organizado. Uma vez que, por definição, as características Organizacionais não possuem ligações concretas com o uso real do domínio, as características de ambiente operacional, tecnologia de domínio e técnica de implementação não devem receber a classificação de Organizacionais.

Quanto à variabilidade, as características na notação Odyssey-FEX recebem a seguinte classificação:

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

A representação gráfica de um ponto de variação e suas variantes é expressa através dos estereótipos <<variation point>> e <<variant>>, respectivamente, juntamente com o relacionamento Alternativo que é descrito na Tabela 3.3. A ausência de estereótipo quanto à variabilidade significa que a característica é invariante.

Por último, uma característica de um domínio pode ser classificada quanto à sua opcionalidade, como mandatória ou opcional. Tal classificação indica justamente a opcionalidade de uma determinada característica do domínio em qualquer aplicação desenvolvida naquele domínio. Vale ressaltar que a opcionalidade é em relação ao domínio como um todo. Características que são opcionais em relação ao domínio, mas que venham a ser mandatórias em relação a outras características a serem selecionadas, deverão expressar essa informação através de Regras de Composição, mencionadas a

seguir. Na notação Odyssey-FEX, as características opcionais são evidenciadas por um retângulo com contorno pontilhado.

Cada característica deve estar classificada segundo cada um dos três critérios, os quais são ortogonais entre si, definindo uma tripla <categoria, variabilidade, opcionalidade>. Por exemplo, <característica de domínio conceitual, variante, mandatória> e <Característica de Ambiente Operacional, Ponto de Variação, Opcional> ilustram possíveis classificações de características.

No que diz respeito ao relacionamento entre as características descritas anteriormente, a notação Odyssey-FEX incorpora relacionamentos da UML e outros relacionamentos que tornam explícita a representação da variabilidade no modelo de características. Na Tabela 3.3 são descritos tais relacionamentos, bem como a sua representação visual no modelo de características.

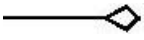
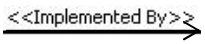
Um dos requisitos de uma notação para modelagem de características é a necessidade de representação de dependência e exclusividade entre as mesmas. Kang *et al.* (1990) propõe a representação de tais conceitos através do uso de regras de composição (*Composition Rules*), as quais definem restrições existentes entre características e que não são expressas através de relacionamentos por questões visuais do modelo, somente através de cláusulas.

A notação Odyssey-FEX propõe Regras de Composição que estendem a proposta de Kang, contemplando operadores booleanos que permitem uma representação mais expressiva da semântica de um modelo de características. Assim, regras de composição como "(A AND B) requer C" podem ser construídas. Uma Regra de Composição tem a seguinte estrutura:

< Antecedente + palavra-chave + Conseqüente>

A Regra de Composição que expressa dependência entre características é denominada Regra de Composição Inclusiva (palavra chave "requer"), enquanto a regra que expressa mutua exclusividade é denominada Regra de Composição Exclusiva (palavra chave "exclui"). O antecedente e o conseqüente de uma regra são expressões que denotam uma combinação ou não de características de um domínio. Esta combinação de expressões tem por finalidade intensificar a expressividade de uma regra de composição.

Tabela 3.3: Relacionamentos da notação Odyssey-FEX

Representação	Descrição	
	**Composição** – Relacionamento em que uma característica é composta de outra. Denota relação na qual uma característica é parte fundamental de outra, de forma que a segunda, que representa o todo, requer a primeira, ou seja, a sua parte.	Relacionamentos da UML
	**Agregação** – Relacionamento em que uma característica representa o todo, e a outra uma parte. Similar à composição, porém as características envolvidas existem independentemente uma da outra.	
	**Herança** – Relacionamento em que há uma generalização/especialização das características. Este tipo de relacionamento indica que as características mais especializadas (filhas) herdam as peculiaridades e atributos de características mais generalizadas (pais).	
	Associação – Relacionamento simples entre duas características. Denota algum tipo de ligação entre seus membros. Pode ser nomeada, indicando um tipo específico de ligação.	
	**Alternativo** (*Alternative*) - Relacionamento entre um ponto de variação e suas variantes, denota a pertinência de uma variante a um determinado ponto de variação.	Relacionamentos propostos na Odyssey-FEX
	**Implementado por** *(Implemented By)* - Relacionamento entre Características de Domínio e Características de Ambiente Operacional, Tecnologia de Domínio e Técnicas de Implementação, ou entre estas três últimas características.	
	**Ligação de Comunicação** *(Communication Link)* - *R*elacionamento existente entre Características de Entidade e Características de Domínio.	

Além da notação Odyssey-FEX, a abordagem DECOM utiliza uma sistemática para verificação de inconsistências em modelos UML, proposta em (DANTAS *et al.*, 2001). Esta sistemática foi estendida por (OLIVEIRA, 2006) para a verificação de tais inconsistências em modelos de características. No contexto do modelo de características gerado a partir da notação Odyssey-FEX, esta sistemática deve verificar inconsistências, tais como as apresentadas na Tabela 3.4. Nesta Tabela, o ponto de interrogação (?) significa que este pode ser substituído por qualquer classificação associada àquela categoria.

A Figura 3.2, mostra um exemplo do modelo de características do domínio de Telefonia Móvel utilizando a notação Odyssey-FEX. Esta Figura pode ser melhor visualizada no Anexo 1.

Tabela 3.4: Regras de consistência do modelo de características (extraído de (OLIVEIRA, 2006))

Elemento do Modelo	Regra
Características de Domínio	Características Não-Definidas não podem ser Organizacionais e vice-versa.
Pontos de Variação e Variantes	<?, Variante,?> deve obrigatoriamente estar associada um e somente um <?, Ponto de variação,?>
	<?, Variante, Mandatória> deve ter um relacionamento do tipo Alternativo com outra <?, Ponto de Variação, Mandatória>
	<?, Ponto de Variação, Opcional> deve ter um relacionamento do tipo Alternativo com outras < ?, Variante, Opcional>
Relacionamento de Composição	Não deve existir relacionamento de composição entre <?, ? , Opcional> e <?, ? , Mandatório>.
Relacionamento Alternativo	O relacionamento Alternativo só poderá ocorrer entre <?, Ponto de variação, ?> e <?, Variante, ?>.
	O relacionamento Alternativo só poderá ter <?, Ponto de variação, ?> como origem.
Relacionamento Ligação de Comunicação	O relacionamento Ligação de Comunicação só poderá ocorrer entre <Característica de Domínio, ?, ?> e <Característica de Entidade, ?, ?>
Relacionamento Implementado Por	No relacionamento "Implementado Por", a característica de origem pertence a uma categoria diferente da característica de destino.
Regras de Composição	Características dependentes entre si não podem ser mutuamente exclusivas, e vice versa.
	Numa regra inclusiva, o conseqüente só poderá ser <?, ? , Opcional> se o antecedente for <?,?,Opcional>.
	Uma regra exclusiva só pode envolver <?, ? , Opcional>.
	Uma regra não pode ter antecedente definido e conseqüente nulo ou vice versa.

As características descritas na Tabela 3.1 estão identificadas pelos estereótipos <<conceptual>>, <<functional>>, <<actor>>, <<operating environment>>, <<domain technology>> e <<implementation technique>>. As características com retângulos tracejados representam características opcionais e as demais, obrigatórias. A característica Jogos representa um ponto de variação opcional, e as características Car Racer e Snake representam suas variantes opcionais. Para representar esta relação é utilizado o relacionamento alternativo. A mesma relação existe entre Procura Ligações, Ligações Recebidas e Ligações Feitas. Porém, são consideradas características obrigatórias. Telefone Celular está relacionado à característica de entidade Usuário através de um <<communication link>>. Car Racer está relacionada à característica de ambiente operacional Java através de um <<implemented by>>. Telefone Celular está relacionado com as características Chamadas Telefônicas, Agenda, Campainha e Caixa Postal através de composição. As características Alarme, Alerta Vibratório, Câmera, Jogos e Toques Musicais estão relacionados a Telefone Celular por agregação. A regra de exclusividade é expressa no modelo através do caractere X (e.g. Toques Simples ou

Toques Polifônicos) e a regra de dependência através do caractere R (e.g. Alarme e Campainha).

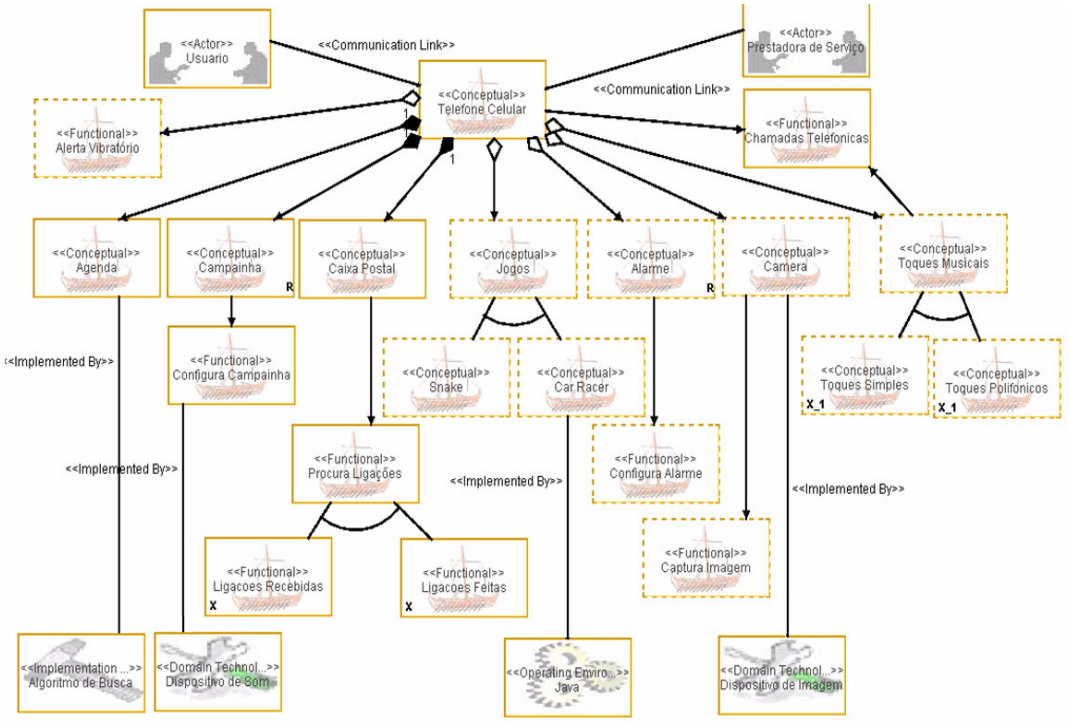


Figura 3.2: Modelo de Características do Domínio de Telefonia Móvel

3.2.2 – Notação para a Modelagem de Variabilidades, Opcionalidades e Regras de Composição nos demais artefatos do domínio.

Tipos de Negócio, Casos de Uso e Componentes são artefatos do domínio, no contexto da abordagem DECOM, os quais são derivados do modelo de características. Portanto, estes artefatos também devem manter a representação das variabilidades, opcionalidades e regras de composição provenientes das respectivas características. Nesta Seção, serão apresentadas as notações para estes artefatos do domínio, em especial no que diz respeito à representação de variabilidades, opcionalidades e regras de composição. Outros detalhes referentes à notação destes artefatos do domínio podem ser encontrados na especificação da UML 2.0 (OMG, 2005) e em (D'SOUZA & WILLS, 1999; CHEESMAN & DANIELS, 2000). As notações apresentadas nestas subseções serão adotadas no modelagem do domínio, orientada pelo processo de ED, proposto nesta abordagem arquitetural, o qual está descrito na Seção 3.3.

3.2.2.1 - Modelagem em Tipos de Negócio

A notação de tipos de negócio equivale à notação de classes proposta pela UML (OMG, 2005). Tipos de negócio são considerados classes, porém especificados na fase de análise. As principais diferenças de notação entre tipos de negócio e classes são: a especificação de tipos de negócio com estereótipo <<business type>> e a inexistência de métodos em tipos, somente atributos.

Neste caso, para a representação de variabilidades, opcionalidades e regras de composição optou-se por uma extensão que alterasse o mínimo possível a notação original e que, ao mesmo tempo, atendesse a semântica necessária. Neste caso, as variabilidades, opcionalidades e regra de exclusão estão especificadas através de estereótipos. Já a regra de inclusão, que representa uma dependência entre características, será representada em tipos de negócio através do relacionamento de associação. Neste caso, a navegabilidade do relacionamento é necessária para representar qual o tipo de negócio depende do outro. Embora dependência e associação representem semânticas distintas, quando estes relacionamentos são analisados para a geração de um determinado componente, ambos serão avaliados da mesma forma. Assim, com intuito de alterar o mínimo possível a semântica do modelo de tipos de negócio, optou-se por utilizar a associação com navegabilidade, ao invés de criar um novo relacionamento de dependência no modelo.

Na Tabela 3.5 são descritas as representações de variabilidades, opcionalidades e regras de composição em tipos de negócio.

Tabela 3.5: Extensões na Notação de Tipos de Negócio para a Modelagem de Variabilidades, Opcionalidades e Regras de Composição

Categoria	Propriedade	Forma de representação
Variabilidade	Ponto de Variação	Estereótipo <<variation point>>
	Variante	Estereótipo <<variant>> no tipo de negócio ou atributo de tipo de negócio.
	Invariante	Ausência de estereótipo
	Relação entre Ponto de Variação e Variantes	Relacionamento de Herança, onde o Tipo de Negócio <<variation point>> representa o pai da relação, e os tipos de negócio <<variant>> representam os filhos.
Opcionalidade	Opcional	Estereótipo <<optional>>
	Mandatório	Ausência de Estereótipo
Regras de Composição	Requer	Relacionamento de Associação. A origem da associação representa o antecedente da regra, e o destino representa o conseqüente. *ou* Especificação de dependência numa propriedade do tipo de negócio que corresponde ao antecedente da regra
	Exclui	Estereótipo <<XOR>> entre os tipos de negócio envolvidos.

A Figura 3.3 mostra um exemplo de Modelo de Tipos de Negócio do domínio de Telefonia Móvel, considerando a modelagem de variabilidade, opcionalidade e regra de composição.

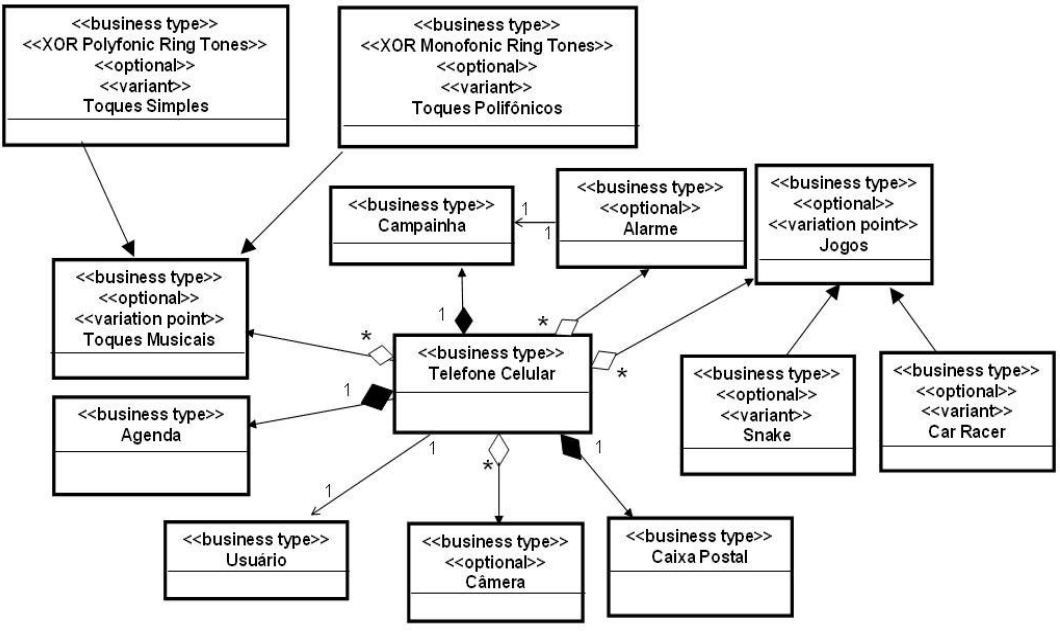


Figura 3.3: Modelo de Tipos de Negócio com Notação para a Modelagem de Variabilidades, Opcionalidades e Regras de Composição

O Tipo de Negócio Toques Musicais representa um ponto de variação opcional e está identificado pelos estereótipos <<variation point>> e <<optional>>. Toques Simples e Toques Polifônicos são tipos de negócio variantes e opcionais que fazem parte de uma mesma regra de composição excludente. Portanto, possuem os estereótipos <<variant>>, <<optional>> e <<XOR "Toques Simples">>. Campainha e Alarme, que originalmente fazem parte de uma regra de composição do tipo requer, estão relacionados por meio de associação, em que o primeiro, que representa o antecedente, está associado ao segundo, que representa o conseqüente. Os demais relacionamentos são semelhantes àqueles explorados na notação para modelagem de características.

3.2.2.2 - Modelagem em Casos de Uso

Casos de Uso do Domínio possuem a mesma semântica e notação dos casos de uso da UML (OMG, 2005). Com o intuito de efetuar o menor número de extensões possível, foram utilizados, em sua maioria, recursos de estereótipos e relacionamentos já existentes no modelo. A Tabela 3.6 descreve as extensões efetuadas. Embora semanticamente sejam distintos, o relacionamento Alternativo do modelo de características será representado por herança entre casos de uso. Tal decisão se deve ao

fato de que, em termos de projeto baseado em componentes, ambos os relacionamentos são mapeados, semanticamente, da mesma forma (e.g. componente que representa o caso de uso filho da relação de herança ou variante do relacionamento alternativo deve requerer o componente que representa o pai da relação de herança ou ponto de variação do relacionamento Alternativo). Neste caso, não é necessário criar mais um relacionamento no modelo de casos de uso. Para a representação de regras de inclusão, é utilizado o recurso de pré-condição da UML (OMG, 2005). O caso de uso que representa o antecedente da regra deve ter registrado na sua especificação, uma pré-condição de existência de um ou mais casos de uso do domínio.

Tabela 3.6: Extensões na Notação de Casos de Uso para a Modelagem de Variabilidades, Opcionalidades e Regras de Composição

Categoria	Propriedade	Forma de representação
Variabilidade	Ponto de Variação	Estereótipo <<variantion point>>
	Variante	Estereótipo <<variant point>>
	Invariante	Ausência de estereótipo
	Relação entre Ponto de Variação e Variantes	Relacionamento de Herança, onde o Caso de Uso <<variation point>> representa o pai da relação, e os Casos de Uso<<variant>> representam os filhos.
Opcionalidade	Opcional	Estereótipo <<optional>>
	Mandatório	Ausência de Estereótipo
Relacionamentos	Relação de Associação	Relacionamento com navegabilidade contendo estereótipo <<association >>
Regras de Composição	Requer	Uso de pré-condição especificado na descrição do caso de uso que corresponde ao antecedente da regra.
	Exclui	Estereótipo <<XOR>> nos casos de uso envolvidos.

A Figura 3.4 mostra um exemplo de Modelo de Casos de Uso do domínio de Telefonia Móvel, considerando a modelagem de variabilidade. O Caso de Uso Procura Ligações representa um ponto de variação e está identificado pelo estereótipo <<variation point>>. Ligações Recebidas e Ligações Feitas representam variantes de Procura Ligações e estão identificadas pelo estereótipo <<variant>>. Além disso, fazem parte de uma mesma regra de exclusão, estando assim identificados pelos estereótipos <<XOR Ligações Recebidas>> e <<XOR Ligações Feitas>>. Configura Chamadas é um caso de uso Opcional, representado pelo estereótipo <<optional>>. Os casos de uso Chamadas Telefônicas e Procura Ligações fazem parte de uma regra de inclusão. Assim, Procura Ligações deve ter na sua especificação uma pré-condição de existência referente ao caso de uso Chamadas Telefônicas. Gerencia Compromissos e Gerencia Telefones estão relacionados originalmente a Gerencia Agenda através de composição, sendo assim, no modelo de casos de uso, proposto o estereótipo <<include>>.

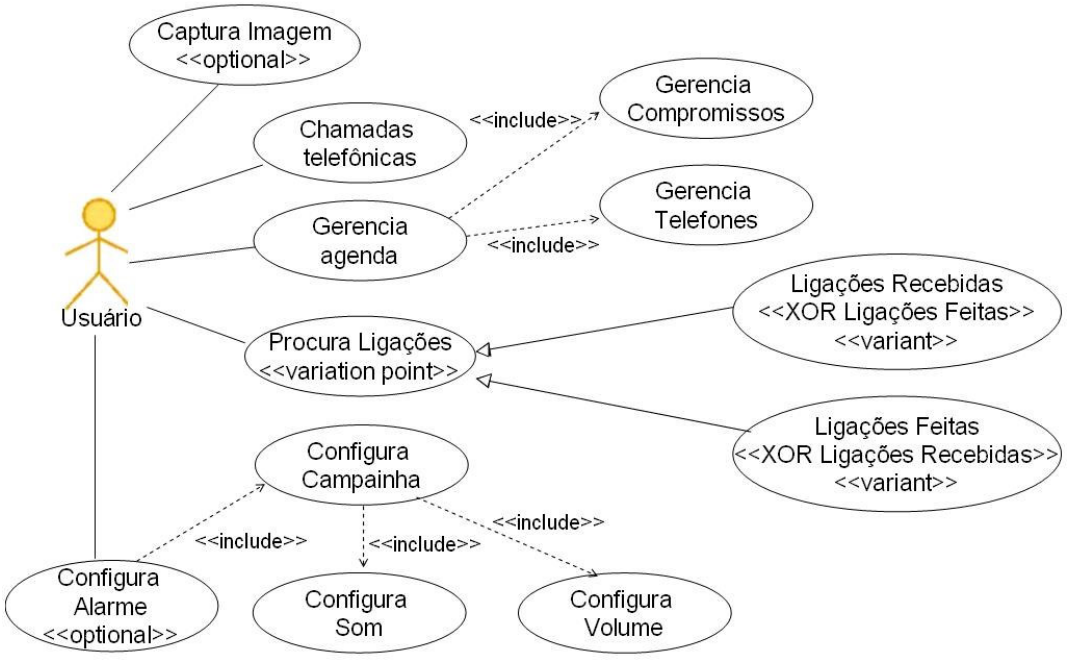


Figura 3.4: Parte de um Modelo de Casos de Uso do Domínio de Telefonia Móvel

3.2.2.3 - Modelagem de Componentes

A notação de componentes adotada é a proposta pela UML 2.0. No Anexo 2 é apresentada uma breve descrição da UML 2.0, em especial no que se refere à especificação de componentes. A representação de variabilidades, opcionalidades e regras de exclusão são expressas através de estereótipos. A regra de composição de inclusão é expressa através de uma interface requerida. A Tabela 3.7 descreve as representações de variabilidades, opcionalidades e regras de composição em componentes.

Tabela 3.7: Extensões na Notação de Componentes para Modelagem de Variabilidades, Opcionalidades e Regras de Composição

Categoria	Propriedade	Forma de representação
Variabilidade	Ponto de Variação	Estereótipo <<variation point>>
	Variante	Estereótipo <<variant point>>
	Invariante	Ausência de estereótipo
	Relação entre Ponto de Variação e Variantes	Os componentes com estereótipo <<variant>> deve requerer a interface fornecida pelo componente com estereótipo <<variation point>>.
Opcionalidade	Opcional	Estereótipo <<optional>>
	Mandatório	Ausência de Estereótipo
Regras de Composição	Requer	O componente que representa o antecedente da regra deve requerer a interface fornecida que conseqüente da regra.
	Exclui	Estereótipo <<XOR>> nos componentes envolvidos.

A Figura 3.5 mostra um exemplo de Modelo de Componentes de Negócio do domínio de Telefonia Móvel, considerando a modelagem de variabilidades, opcionalidades e relacionamentos.

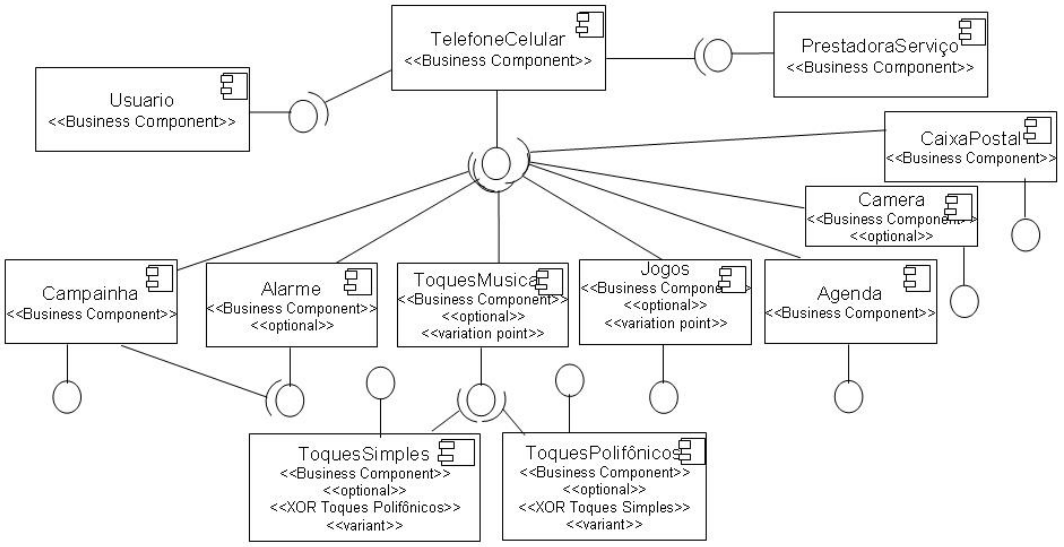


Figura 3.5: Parte de um Modelo de Componentes do Domínio de Telefonia Móvel

O componente de negócio Toques Musicais representa um ponto de variação opcional e está representado pelos estereótipos <<variation point>> e <<optional>>. Toques Simples e Toques Polifônicos são componentes de negócio variantes, opcionais e que fazem parte de uma mesma regra de exclusão mútua. Portanto, possuem os estereótipo <<variant>>, <<optional>> e <<XOR "nome do componente de negócio que exclui">>. Campainha e Alarme, que originalmente fazem parte de uma regra de composição do tipo requer, estão relacionados através de suas interfaces, onde o primeiro, que representa o antecedente, requer a interface do segundo, que representa o conseqüente.

3.3 - Processo de ED com apoio de DBC

Nesta tese é proposto um processo de ED com apoio de DBC, visando preencher as lacunas existentes nos métodos analisados. Este processo, denominado CBD-Arch-DE (**C**omponent-**B**ased **D**evelopment – **Arch**itecture – **D**omain **E**ngineering) propõe 4 grandes fases na ED, conforme mostra a Figura 3.6. Esta figura mostra a modelagem das fases do processo, utilizando a notação para a modelagem de atividades da UML (OMG, 2005). O processo inicia pelo planejamento do domínio, seguido da análise,

projeto e implementação. Entre cada fase existe um ponto de decisão, cada qual questionando sobre uma possível revisão das atividades que envolvem a respectiva fase.

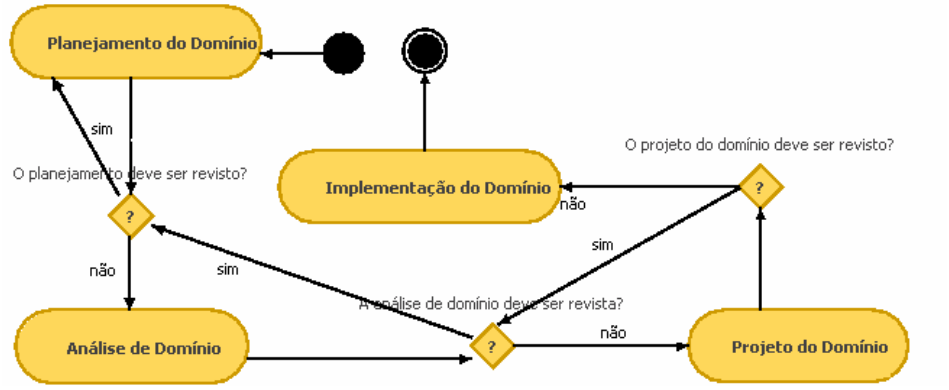


Figura 3.6: Fases do processo CBD-Arch-DE

Durante a execução do processo, uma série de artefatos do domínio são gerados, que representam diferentes abstrações do domínio, começando pela geração dos artefatos de mais alto nível de abstração (e.g. características do domínio) até os artefatos de mais baixo nível (e.g. elementos arquiteturais). À medida que os artefatos de mais baixo nível de abstração são gerados, a abordagem DECOM registra os rastros para os artefatos correspondentes de mais alto nível. Por exemplo, todo componente do negócio está relacionado com os seus respectivos tipos de negócio e casos de uso, que por sua vez estão relacionados às características do domínio dos quais foram originados.

No CBD-Arch-DE, a fase de planejamento do domínio é desenvolvida de forma análoga à proposta do processo Odyssey-DE, pois não é o foco desta tese tal apoio. Maiores detalhes desta atividade podem ser obtidos em (BRAGA, 2000).

Concluída a fase de planejamento, dá-se início às fases de análise, projeto e implementação do domínio, respectivamente, de forma interativa, ou seja, são previstos retornos às fases anteriores para refinamento do domínio. Por serem o foco desta Tese, estas fases são discutidas em mais detalhes no restante desta Seção.

3.3.1 - Fase de Análise do Processo CBD-Arch-DE

Conforme apresenta a Figura 3.7, a fase Análise de Domínio começa pela identificação dos contextos. Um contexto situa o domínio em relação ao seu escopo. Através de um diagrama de contextos é possível modelar os relacionamentos entre contextos do domínios, representando sub-áreas do domínio, bem como os atores envolvidos (BRAGA, 2000). Quanto maior for a abrangência do domínio, maior será a

importância do contexto. A identificação de contextos auxilia a organização dos demais artefatos do domínio e, sobretudo a sua reutilização durante a Engenharia de Aplicação.

No exemplo de telefonia móvel, são sugeridos dois contextos: básico e adicionais. O contexto básico deve organizar as características que existem em qualquer aparelho e, o contexto adicionais deve organizar as funcionalidades específicas de um aparelho ou plano de telefonia celular.

Em seguida, o Engenheiro começa a atividade de identificação dos requisitos do domínio, os quais podem estar relacionados a um ou mais contextos e que nesta tese são representadas usando a notação Odyssey-FEX (BLOIS *et al.*, 2005a), descrita na Seção 3.2.1. A Figura 3.2 mostra um exemplo de modelo de características do domínio de telefonia móvel.

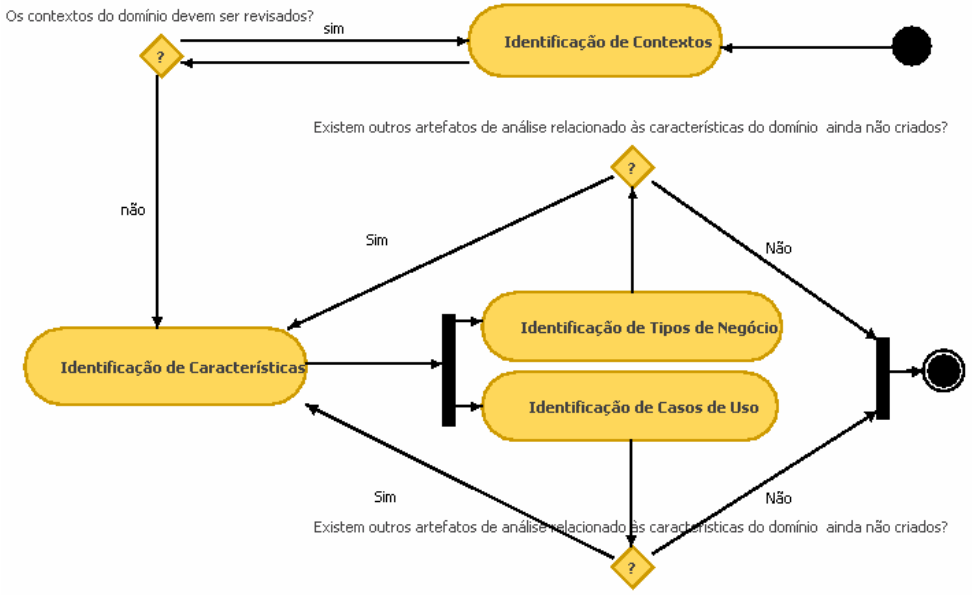


Figura 3.7: Atividades de análise do domínio do processo CBD-Arch-DE

A modelagem de um domínio é muito complexa, e para apoiá-la, o CBD-Arch-DE sugere a utilização de uma abordagem para verificação de possíveis inconsistências existentes no modelo, decorrentes do mal uso da notação (DANTAS *et al.*, 2001), tais como as apresentadas na Tabela 3.4. Além de identificar inconsistências, tal abordagem permite que as mesmas possam ser corrigidas.

Uma vez definidos os requisitos, o processo sugere a execução de duas atividades que podem ocorrer em paralelo, a saber: identificação de tipos de negócio e identificação de casos de uso. A identificação de tipos de negócio também é recomendada em outras abordagens de DBC (D'SOUZA & WILLS, 1999) (CHEESMAN & DANIELS, 2000). Os tipos de negócio (*business types*) são classes no

nível de especificação que descrevem apenas dados (atributos), sem operações de software. Tipos de negócio podem se relacionar por meio de associação, dependência, agregação, composição e herança. Diagramas de tipos de negócio representam artefatos que possivelmente venham a ser persistidos numa aplicação ou para um domínio de aplicações.

Contudo, num processo de ED, os tipos de negócio devem ser relacionados às suas respectivas características do domínio, visando aumentar a sua coesão e a integridade da modelagem.

Assim como proposto por Teixeira (2003), tipos de negócio são artefatos de análise gerados a partir das características conceituais e de entidade do domínio. Este mapeamento se deve ao fato de que características conceituais e de entidade e tipos de negócio representam conceitos do domínio, porém em diferentes níveis de abstração. Uma característica conceitual representa um conceito de um domínio de forma genérica, que nem sempre é tratado pelas aplicações do domínio, servindo somente para o entendimento do domínio. Por outro lado, o tipo de negócio também representa um conceito do domínio, mas considerando suas propriedades, muitas vezes parametrizáveis, servindo assim de base para a geração de aplicações derivadas deste domínio.

Variabilidades, opcionalidades e relacionamentos também devem ser avaliados neste mapeamento. A DECOM propõe um conjunto de heurísticas que auxiliam o mapeamento de características conceituais e de entidade, propostas na notação Odyssey-FEX, em tipos de negócio do domínio. Tais heurísticas estão detalhadas na Seção 3.4.1, as quais utilizam a notação para a modelagem de tipos de negócio, discutido na Seção 3.2.2.1, Tabela 3.5.

Com o uso das heurísticas propostas, o Engenheiro de Domínio, utilizando o processo CBD-Arch-DE, irá criar um modelo de tipos de negócios, os quais podem representar variabilidades, opcionalidades e regras de composição através de estereótipos, e os relacionamentos existentes entre estes. O modelo de tipos de negócio obtido a partir do modelo de características da Figura 3.2 para o domínio de Telefonia Móvel é apresentado na Figura 3.3.

A outra atividade da análise do domínio no CBD-Arch-DE é a definição dos casos de uso. Casos de uso do domínio descrevem possíveis operações sobre os tipos de negócio previamente criados. Casos de uso do domínio devem ser derivados a partir das características funcionais do domínio, já que casos de uso descrevem funcionalidades a

respeito dos requisitos do domínio. A opcionalidade, variabilidade, as regras de composição e relacionamentos existentes entre características funcionais devem, também, ser mapeadas para os casos de uso do domínio. A fim de manter tais consistências, também foram definidas heurísticas para apoiar este mapeamento, as quais estão detalhadas na Seção 3.4.2, utilizando a notação estendida para a modelagem de casos de uso discutida na Seção 3.2.2.2, Tabela 3.6.

Apoiado pelo uso destas heurísticas, o modelo de casos de uso do domínio resultante será composto por um conjunto de casos de uso, que correspondem às características funcionais e de entidade, bem como suas respectivas propriedades quanto a opcionalidade, variabilidade, regras de composição, representadas através de estereótipos, e relacionamentos entre casos de uso. A Figura 3.4 mostra um exemplo de modelo de caso de uso obtido a partir das heurísticas propostas, considerando o modelo de características da Figura 3.2.

Conforme apresentado na Figura 3.7, a fase de análise de domínio termina quando o Engenheiro de Domínio estimar que todas as características, tipos de negócio e casos de uso foram devidamente identificados e especificados para o domínio em questão, com o apoio das heurísticas propostas para sua geração.

3.3.2 - Fase de Projeto do Processo CBD-Arch-DE

O Projeto de Domínio compreende as atividade de criação de componentes, o agrupamento destes e a geração da arquiteturas de referência do domínio, conforme ilustra a Figura 3.8. Cada uma das atividades será detalhada nas próximas Seções.

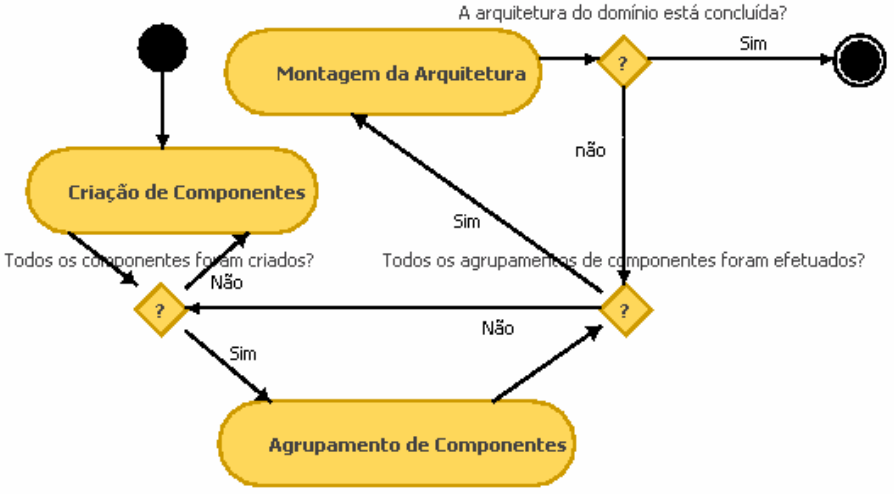


Figura 3.8: Atividades de projeto do domínio do processo CBD-Arch-DE

3.3.2.1 - Atividade de Criação de Componentes

Durante esta atividade, diferentes tipos de componentes são criados com base em características específicas do domínio. Nas subseções que se seguem, são apresentados os diferentes tipos de componentes, junto com os artefatos da análise de domínio que dão origem aos mesmos. Além disso, será apresentado como o processo CBD-Arch-DE conduz o Engenheiro do Domínio na geração de cada um dos tipos de componentes discutidos na Seção 2.3.1.

3.3.2.1.1 - Categorização dos Componentes do Domínio

Conforme discutido no Capítulo 2, as abordagens de DBC em geral identificam alguns tipos de componentes os quais atendem aos requisitos de aplicações ou de domínios de aplicações. Na DECOM serão gerados os seguintes componentes: de negócio, de processo, utilitário e de infra-estrutura. Tais componentes combinam as categorias propostas por Brown (2000) e Cheesman & Daniels (2000).

A Figura 3.9 esquematiza a relação entre os artefatos de análise e componentes do domínio.

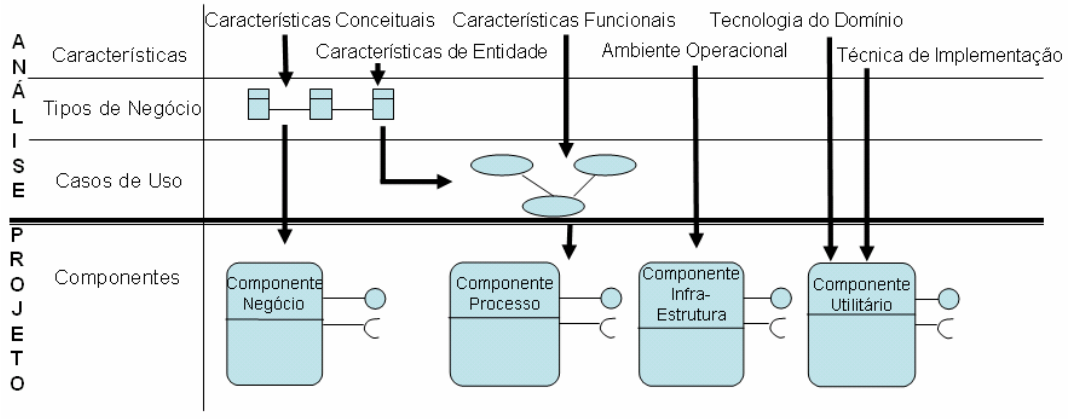


Figura 3.9: Artefatos de Análise que originam componentes de um domínio

Componentes de negócio representam conceitos do domínio e são gerados através dos tipos de negócio, os quais estão relacionados às características conceituais e de entidade do domínio. Componentes de processo representam o comportamento do domínio e estão relacionados aos casos de uso, sendo que estes últimos representam as características funcionais do domínio e realizam ações sobre tipos de negócio. Componentes utilitários correspondem a serviços genéricos e são criados a partir das características de ambiente operacional e tecnologia de domínio, as quais possuem

propósito semelhante, porém num alto nível de abstração. Por último, os componentes de infra-estrutura são responsáveis por estabelecer a comunicação com as plataformas de execução de software, e devem ser criados a partir das características de ambiente operacional.

Como apresentado no Capítulo 2, a literatura (ATKINSON *et al.*, 2002) (CHEESMAN & DANIELS, 2000) (D'SOUZA & WILLS, 1999) dá maior ênfase no apoio a geração de componentes de negócio e processo. No entanto, uma arquitetura de componentes de um domínio deve também contemplar os serviços prestados pelos componentes utilitários e de infra-estrutura. Se por um lado estes serviços devem ser contemplados na arquitetura de componentes do domínio, por outro existe uma dúvida quanto ao nível de especificação destes componentes, uma vez que estes dependem fortemente da tecnologia de hardware e software sobre o qual serão implementados, e que geralmente são serviços oferecidos pela própria tecnologia de componentes (e.g. EJB, .NET).

Em termos de arquitetura de domínio, os componentes utilitários e de infra-estrutura possuem serviços autocontidos, com um propósito específico (e.g. comunicação num serviço de mensagens), podendo, em sua maioria, serem disponibilizados na arquitetura através de um único componente, com interfaces definidas que viabilizem a comunicação com os demais componentes do domínio que requerem seus serviços.

Componentes de Negócio, Utilitários e de Infra-estrutura podem ser criados independentemente uns dos outros, pois são provenientes de artefatos distintos. Já os componentes de processo necessitam da geração prévia de componentes de negócio, mais especificamente de suas interfaces fornecidas para a geração do próprio componente e do seu respectivo diagrama de interação.

Para cada tipo de componente gerado, deve ser também avaliado a variabilidade, opcionalidade, relacionamentos e regras de composição dos seus respectivos artefatos de origem, visando a manutenção de consistência de tais propriedades no próprio componente. Por esta razão, são propostas heurísticas que auxiliam a obtenção desta consistência nos componentes gerados. Tais heurísticas, categorizadas por tipo de artefato gerado, estão descritas na Seção 3.4.

As atividades sobre a criação de componentes do domínio, especificadas com detalhe nas subseções que seguem, podem ser visualizadas através da Figura 3.10.

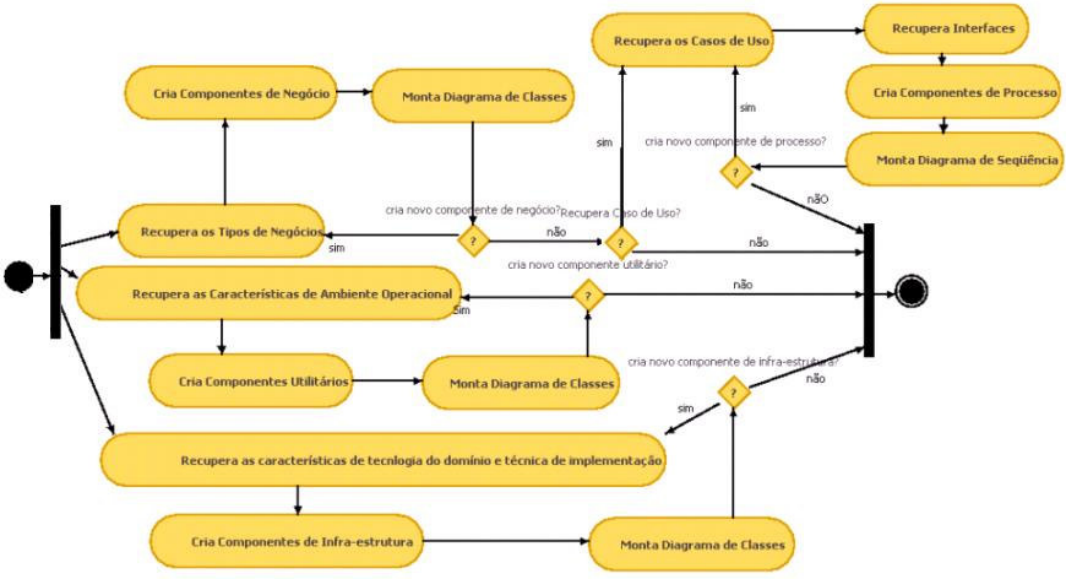


Figura 3.10: Atividades de criação de componentes do processo CBD-Arch-DE

3.3.2.1.2 - Criação de Componentes de Negócio

No CBD-Arch-DE, componentes de negócio são gerados a partir de tipos de negócio, segundo o estilo arquitetural baseado em tipo (HERZUM & SIMS, 2000). Considerando as extensões de (TEIXEIRA, 2003) para redução do número de componentes gerados, estes componentes podem agrupar dois ou mais tipos de negócio, de acordo com a relevância de relacionamentos, a saber: é obrigatório o agrupamento de tipos de negócio relacionados por herança; é recomendado o agrupamento de tipos de negócio relacionados por composição e agregação; é sugerido o agrupamento de tipos de negócio, relacionados por meio de associação.

Esta tese estende a proposta de Teixeira (2003) de modo que tipos de negócio sejam agrupados não somente com base em seus relacionamentos (i.e. associação, herança, composição, agregação), como também em função de suas variabilidades, opcionalidades e regras de composição. Tais propriedades são representadas no modelo de componentes através dos estereótipos discutidos na Seção 3.2.2.3, Tabela 3.7.

Como resultado deste processo são gerados componentes de negócio com interfaces providas e requeridas.

Além da geração do próprio componente, é gerada uma especificação interna do componente, através de um diagrama de classes. Este diagrama de classes preliminar

contém uma classe para cada tipo de negócio que está relacionado a este componente, com seus respectivos atributos, extraídos dos respectivos tipos de negócio e um conjunto de operações padrão (*create*, *update*, *delete*), também disponibilizadas nas interfaces fornecidas pelos componentes. Além disso, também são gerados os relacionamentos entre classes, os quais são análogos aos relacionamentos entre os tipos de negócio de origem.

3.3.2.1.3 - Criação de Componentes de Processo

A geração dos componentes de processo inicia pela identificação de todos os casos de uso do domínio. Para cada caso de uso selecionado, são recuperados todos os passos que realizam o caso de uso. Como comentado na Seção 3.3.2.1.1, os passos dos casos de uso especificam a interação existente entre os tipos de negócio do domínio. Obtendo-se os tipos de negócio que fazem parte desta interação, é possível identificar qual o componente de negócio que realiza um determinado tipo de negócio e, por sua vez, quais das interfaces fornecidas pelo componente gerenciam os dados do tipo de negócio em avaliação. Depois de detectadas todas as interfaces necessárias para a realização dos casos de uso selecionados, são então gerados componentes de processo, cada qual com uma interface fornecida e uma ou mais interfaces requeridas. A interface fornecida é composta por todas as operações especificadas no caso de uso. As interfaces requeridas correspondem àquelas fornecidas pelos componentes de negócio utilizados pelo componente de processo. Neste ponto do CBD-Arch-DE, não é sugerido o agrupamento de casos de uso em um único componente de processo, já que casos de uso em ED são bastante genéricos. Contudo, o agrupamento de componentes de processo pode ser recomendado posteriormente, quando da definição da arquitetura de referência. Para este último caso, podem então ser utilizados os critérios de agrupamentos de componentes a serem apresentados na Seção 3.5.

Outras propriedades a serem garantidas no componente de processo são os relacionamentos, variabilidades, opcionalidades e regras de composição atribuídas ao caso de uso. Para tal, é proposto um conjunto de heurísticas que orienta a criação destes componentes, discutido na Seção 3.4.3.

Para cada componente de processo é gerado um diagrama de seqüência. Através deste diagrama, um cliente, ao requerer um serviço fornecido pela interface do componente de processo, solicita conseqüentemente os serviços relacionados à sua

interface requerida, ou seja, as providas pelos componentes de negócio dos quais dependem.

3.3.2.1.4 - Criação de Componentes Utilitários e de Infra-Estrutura

O CBD-Arch-DE sugere que seja gerado um componente para cada característica tecnológica (i.e. ambiente operacional, tecnologia de domínio e técnica de implementação), sem sugerir agrupamentos. Existem duas justificativas para não agrupar características tecnológicas:

- Características tecnológicas já são originalmente de alta granularidade, e o seu agrupamento para a geração de componentes utilitários ou de infra-estrutura pode restringir a sua reutilização em novas aplicações do domínio, ou até mesmo em outros domínios (BROWN, 2000);
- A categorização de componentes utilitários e de infra-estrutura serve exatamente para distinguir os tipos de serviços oferecidos pelos diferentes tipos de características tecnológicas do domínio.

Tais componentes devem ser previstos no projeto do domínio, uma vez que componentes de negócio e de processo podem utilizar seus serviços para executar atividades específicas do domínio. Embora tais componentes sejam gerados num nível de projeto, durante a implementação do domínio eles podem ser obtidos por terceiros, pois a maioria representa serviços genéricos. Tais componentes podem ainda ser disponibilizados pela própria tecnologia de componentes escolhida para a implementação do domínio.

Assim como os componentes de negócio, a geração dos componentes utilitários e de infra-estrutura inclui, além da própria geração do componente e suas respectivas interfaces providas e requeridas, variabilidades, opcionalidades, relacionamentos, regras de composição e sua especificação interna. Para tal, também é proposto um conjunto de heurísticas, discutidas na Seção 3.4.3, as quais utilizam a notação apresentada na Seção 3.2.2.3. A especificação destes componentes consiste de um diagrama de classes composto inicialmente por uma classe, que representa a característica tecnológica que originou o componente, sem atributos e com os métodos previstos na interface fornecida pelo componente. Este diagrama de classes preliminar deve necessariamente ser refinado pelo engenheiro do domínio ou de aplicação.

3.3.2.1.5 - Considerações Finais sobre a Geração de Componentes do Domínio

A Figura 3.11 mostra parte do que seria um modelo de componentes no domínio de Telefonia Móvel, considerando o modelo de características da Figura 3.2, o modelo de tipos de negócio da Figura 3.3 e o modelo de casos de uso da Figura 3.4. Este modelo foi gerado a partir das atividades previstas para a geração dos diferentes tipos de componentes do domínio, usando a notação para modelagem de variabilidades, opcionalidades e regras de composição em componentes. Tais componentes estão organizados conforme a arquitetura em camadas sugerida por Brown (2000).

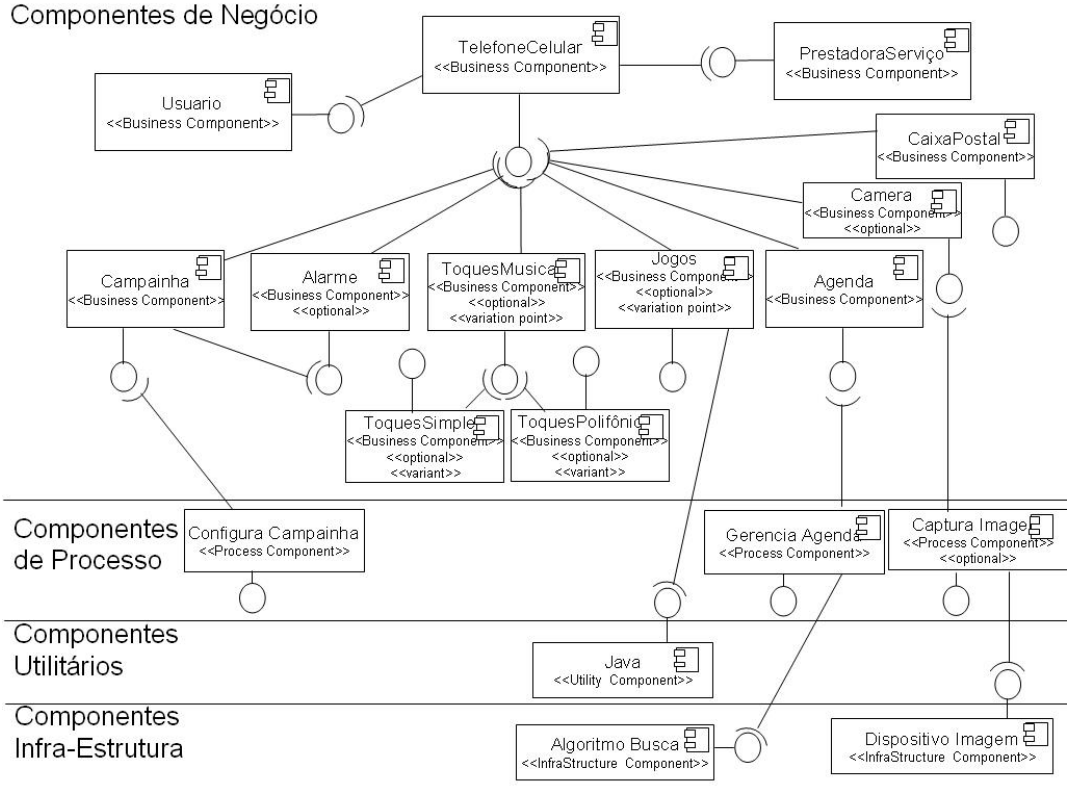


Figura 3.11: Parte de um Modelo de Componentes do Domínio de Telefonia Móvel

A atividade de geração de componentes é concluída quando o Engenheiro de Domínio percebe que todos os componentes do domínio e suas respectivas especificações foram identificados e gerados. Além disso, é possível identificar para quais características do domínio foram ou não criados componentes de negócio, utilitários e de infra-estrutura, com intuito de orientar as decisões do engenheiro.

3.3.2.2 - Atividade de Agrupamento de Componentes do Domínio

As atividades descritas na Seção 3.3.2.1 geram muitos componentes representando conceitos do negócio, processos, e os aspectos tecnológicos do domínio.

Estes componentes podem trocar um grande volume de mensagens e, muitas vezes, esta interação ocorre entre um grupo específico de componentes. Quanto maior a quantidade de mensagens trocadas entre componentes, maior o acoplamento destes componentes. Como conseqüência, mais complexa se torna a arquitetura do domínio, comprometendo a sua compreensão, manutenibilidade e principalmente a sua reusabilidade.

Para melhorar a qualidade esperada de uma arquitetura de componentes, o processo CBD-Arch-DE sugere que componentes que trocam muitas mensagens entre si sejam agrupados num único elemento arquitetural, o qual representa um conjunto de requisitos do domínio que interagem entre si com uma maior incidência quando comparado a outros. Para a geração destes elementos arquiteturais, o CBD-Arch-DE oferece critérios para apoio ao agrupamento de componentes, os quais são detalhados na Seção 3.5.

A utilização por algum agrupamento de componentes sugerido deve estar em consonância com o estilo arquitetural adotado para a arquitetura de referência do domínio, a qual será construída durante a próxima atividade do CBD-Arch-DE, montagem da arquitetura, discutido em mais detalhes na próxima seção.

O Engenheiro de Domínio pode utilizar estes critérios de agrupamento de componentes para obter informações sobre a natureza das interações entre os componentes de um domínio, agrupando os componentes analisados segundo os critérios.

Os elementos arquiteturais assim formados pelo Engenheiro de Domínio devem ser compostos por um ou mais componentes, conforme discutido na Seção 3.5.

Após a identificação destes elementos arquiteturais, o processo sugere que o Engenheiro de Domínio construa uma nova versão da arquitetura de componentes, a qual será composta por componentes que representam elementos arquiteturais. Espera-se que o resultado desta nova versão da arquitetura de componentes do domínio permita uma melhor organização dos componentes do domínio, uma melhor compreensão da arquitetura do domínio, baseado nos requisitos iniciais e, sobretudo, uma melhor reutilização desta arquitetura do ponto de vista de Engenharia de Aplicação.

3.3.2.3 - Atividade de Montagem da Arquitetura de Referência do Domínio

Na montagem da Arquitetura de Referência, o Engenheiro de Domínio deve organizar a arquitetura do domínio, utilizando estilos arquiteturais para a sua

organização. Como resultado é possível que sejam geradas diferentes arquiteturas de referência para um domínio específico.

O CBD-Arch-DE sugere que os Engenheiros de Domínio identifiquem os requisitos de qualidade esperados pela arquitetura do domínio para que então, com base nestes, possa ser indicada a utilização de algum padrão arquitetural. O processo indica a utilização da proposta de Xavier (2001), descrita na Seção 2.4.2, para apoio a criação e instanciação de estilos arquiteturais. Com base na avaliação dos requisitos não-funcionais esperados pelo Engenheiro para um determinado domínio, é gerada uma lista de padrões arquiteturais adequados. A partir deste resultado, o Engenheiro de Domínio deve avaliar quais os componentes do domínio previamente criados nas duas atividades anteriores do processo e posteriormente efetuar as instanciações necessárias. Nesta atividade, um elemento arquitetural do padrão pode ser instanciado através de um ou mais componentes do domínio modelado.

Como exposto nas Seções 2.3.2 e 2.4.2.1, existe uma tendência na geração de arquiteturas de componentes baseado no estilo arquitetural em camadas. Os métodos de DBC sugerem a criação de componentes que representem conceitos do negócio, processos associados a estes conceitos e ainda componentes que possam servir de base para apoio às atividades do negócio da aplicação. Estes componentes estão distribuídos em diferentes camadas da aplicação, relacionados através de suas interfaces. Em função da categorização dos componentes a partir de suas funcionalidades bem definidas, a estruturação de uma arquitetura baseada em componentes na forma de camadas se torna, em grande parte, um processo natural.

No processo CBD-Arch-DE, os componentes do domínio são categorizados em componentes de negócio, processo, utilitário e de infra-estrutura. Esta organização privilegia que a AR de um domínio apresente componentes organizados de acordo com estas categorias. Embora tal organização sugira o uso do estilo arquitetural em camadas, onde em cada camada deveriam ser administrados os componentes das diferentes categorias, a construção destas camadas e a fronteira existente entre as mesmas não foram apoiadas pela DECOM. Para um apoio a contento, é necessário o suporte a construção de elementos que representem estas camadas e propriedades a elas associadas de forma a garantir a consistência intra-camadas e inter-camadas, uma vez que componentes de diferentes camadas trocam mensagens entre sim.

O modelo da Figura 3.11 mostra os componentes de negócio, processo, utilitário, e de infra-estrutura organizados na forma de camadas, seguindo a estrutura proposta por Brown (2000).

3.3.3 - Fase de Implementação do processo CBD-Arch-DE

Nas atividades da fase de projeto, foram gerados artefatos sem qualquer vínculo tecnológico. Porém, na atividade de implementação do domínio é necessário que os artefatos sejam criados a partir de uma tecnologia de componentes específica, e que exista alguma forma de mapeamento dos modelos previamente criados, em outros que estejam vinculados a alguma tecnologia. O CBD-Arch-DE utiliza uma abordagem para transformação de modelos, independentes de tecnologia, em outros modelos, dependentes de tecnologia.

A fase de implementação do domínio é composta de duas atividades: Transformação de Modelos para uma Tecnologia Específica e Geração do Código.

Para a transformação de modelos é utilizada uma abordagem, definida em (MAIA *et al.*, 2005b), denominada Odyssey-MDA, a qual foi desenvolvida com base no *framework* conhecido como *Model Driven Architecture* (MDA) (OMG, 2003). O *framework* MDA provê uma sistemática para especificar um modelo independente de plataforma (PIM – *Platform Independent Model*) e realizar uma transformação, com base numa tecnologia específica, a fim de obter um modelo de uma plataforma em particular (PSM – *Platform Specific Model*).

Com base no *framework* MDA, a abordagem Odyssey-MDA permite ao desenvolvedor realizar a transformação dos PIMs em PSMs. Por exemplo, os componentes de negócio do Domínio de Telefonia Móvel (modelo PIM) podem ser transformados para componentes, classes, interfaces, atributos e operações no modelo PSM.

Depois de obtidos os PSMs esperados, a outra atividade desta fase do CBD-Arch-DE é a geração do código fonte. Nesta atividade, é criada uma representação do código fonte, a partir do modelo PSM, utilizado a linguagem de programação da plataforma escolhida (e.g. Java para a plataforma EJB).

Um maior detalhamento da abordagem Odyssey-MDA é apresentado na Seção 3.6.

3.3.4 – Considerações Finais sobre o CBD-Arch-DE

Concluídas todas as fases do processo CBD-Arch-DE, espera-se que o Engenheiro de domínio tenha:

- Efetuado uma modelagem de domínio consistente com relação aos requisitos do domínio e suas respectivas variabilidades, opcionalidades e relacionamentos;
- Gerado possíveis arquiteturas de referência compostas de elementos arquiteturais do domínio;
- Gerado uma codificação consistente em relação a arquitetura de referência e a tecnologia de componentes escolhida.

Os artefatos gerados pela ED podem ser reutilizados durante a instanciação de uma aplicação.

3.4 - Heurísticas para Apoio à Manutenção de Variabilidades e Opcionalidades de um Domínio

De nada adianta representar variabilidades no modelo de características se as mesmas não forem incorporadas nos artefatos de domínio a partir delas geradas, ao longo de todas as fases da ED.

Neste sentido, foram analisados três domínios de aplicações para se obter indícios sobre um possível mapeamento dos elementos do modelo de características em outros artefatos do domínio (i.e. tipos de negócio, casos de uso e componentes). Os domínios analisados foram: agropecuário; máquina de processo e persistência de dados.

O domínio agropecuário já possuía modelos de características, casos de uso, classes e componentes. Para este domínio, foi feita uma análise dos diferentes modelos e solicitado ao Engenheiro do domínio que indicasse o processo de mapeamento de algumas características do domínio para os demais artefatos de diferentes níveis de abstração.

Para os domínios de persistência de dados e máquina de processo não existia o modelo de características. Neste caso, foi solicitado que os Engenheiros desses domínios criassem um modelo de características correspondente, utilizando a notação Odyssey-FEX. A partir deste modelo de características, foi solicitado, aos Engenheiros, a indicação das classes, casos de uso e componentes correspondentes às características modeladas.

A partir destes mapeamentos foi feita uma análise dos mapeamentos recorrentes nos três domínios distintos.

Em (OLIVEIRA, 2006), foi efetuada uma análise específica do modelo de classes destes domínios com o objetivo de obter indícios de mapeamentos comuns. A partir desta análise, foi proposto um conjunto de heurísticas para o mapeamento de características em classes de domínio.

Nesta Tese, tais análises foram ampliadas com o intuito de mapear variabilidades, opcionalidades, relacionamentos e regras de composição provenientes do modelo de características para todos os artefatos de domínio utilizados no CBD-Arch-DE. Assim, este conjunto de heurísticas auxilia o mapeamento consistente inter-modelos, ou seja, entre diferentes artefatos do domínio (i.e. características, tipos de negócio, casos de uso e componentes), utilizando as notações estendidas para tipos de negócio, casos de uso e componentes, conforme descrito nas Seções 3.2.2.1, 3.2.2.2 e 3.2.2.3. A Tabela 3.8 mostra um resumo das heurísticas a serem discutidas na próximas subseções. A tabela indica as fases do processo CBD-Arch-DE que podem ser utilizadas, quais são os artefatos de origem e os artefatos resultantes deste mapeamento, e a identificação de cada heurística.

Tabela 3.8: Heurísticas Propostas na Abordagem DECOM

Fases	Artefato de Origem	Artefato de Destino	Heurísticas Propostas
Análise	Características	Tipos de Negócio	TN1 à TN9
Análise	Características	Casos de Uso	CUso1 à CUso9
Projeto	Tipos de Negócio	Componentes de Negócio	CN1 à CN6
Projeto	Casos de Uso	Componentes de Processo	CP1 à CP4
Projeto	Características Tecnológicas	Componentes Utilitários e de Infra-estrutura	CUI1 à CUI5

3.4.1 - Variabilidade em Tipos de Negócio (TN)

Dado um modelo de característica do domínio, é necessário relacionar tipos de negócio às características conceituais e de entidades, preservando todas as suas propriedades incluindo as variabilidades, opcionalidades, relacionamentos e regras de composição. Caso tais propriedades não sejam preservadas, os artefatos gerados a partir deste ponto do processo de ED podem vir a ser considerados mandatórios e invariantes, violando a especificação do modelo de características. Estas violações por sua vez serão

propagadas a todos os demais artefatos, gerando inconsistências em relação ao domínio e, conseqüentemente, para as aplicações derivadas do domínio.

3.4.1.1 - Heurística TN1

Esta heurística define quais características modeladas no domínio não devem ser mapeadas para tipos de negócio, isto é, características funcionais e tecnológicas.

Características funcionais e tecnológicas do domínio não devem ser mapeadas para tipos de negócio.

3.4.1.2 - Heurística TN2

A heurística TN2 é considerada uma heurística essencial dentre este conjunto proposto, pois sugere o mapeamento de uma característica conceitual em um tipo de negócio. As heurísticas TN3 à TN9, que tratam do mapeamento de variabilidades, opcionalidades e relacionamentos de características conceituais em tipos de negócio, só podem ser avaliadas se a TN2 for usada previamente pelo Engenheiro de Domínio. Não é possível, por exemplo, avaliar o mapeamento da variabilidade de uma característica conceitual para um tipo de negócio, sem que anteriormente tenha ocorrido o mapeamento dos artefatos propriamente ditos, independente de relacionamentos, variabilidades e regras de composição existentes. Portanto, as heurísticas TN3 à TN9 assumem que tal mapeamento já ocorreu. A Figura 3.12 apresenta a heurística e um exemplo de mapeamento.

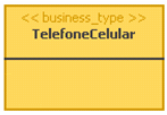
Uma característica conceitual pode ser mapeada para um tipo de negócio.	
Característica Conceitual	**Tipo de Negócio Gerado**
	

Figura 3.12: Mapeamento Sugerido pela Heurística TN2

3.4.1.3 - Heurística TN3

Na notação Odyssey-FEX, é proposto o relacionamento Alternativo para especificar um ponto de variação do domínio. Tendo em vista o conjunto de relacionamentos existentes no modelo de tipos de negócio, se observou que a semântica sugerida pelo relacionamento Alternative no Odyssey-FEX se aproximava da semântica sugerida pelo relacionamento de herança. Para tornar a semântica do relacionamento alternative mais completa em tipos de negócio, são utilizados os estereótipos

<<variation point>> e <<variant>> complementando a semântica deste relacionamento. Assim, a heurística TN3 sugere que características conceituais associadas através do relacionamento Alternative se relacionem através de herança no modelo de tipos de negócio. A identificação de pontos de variação e variantes é feita através de estereótipos no modelo de tipos de negócio. A Figura 3.13 apresenta a heurística TN3 e um exemplo de mapeamento.

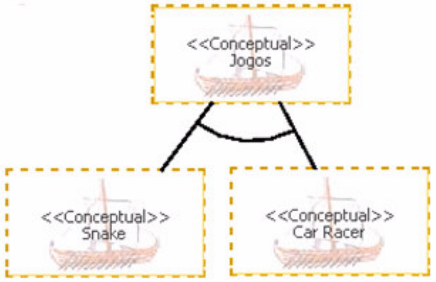
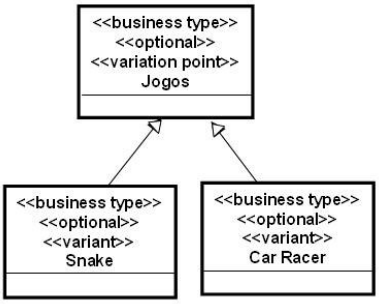
Para cada ponto de variação e suas variantes devem existir tipos de negócio com estereótipos <<variation point>> e <<variant>>, respectivamente, relacionados através de herança. O pai da relação de herança equivale ao ponto de variação, e o filho, uma variante.	
Características Conceituais associadas pelo relacionamento Alternativo	**Tipos de Negócio gerados**
	

Figura 3.13: Mapeamento Sugerido pela Heurística TN3

3.4.1.4 - Heurística TN4

Uma vez que tanto o modelo de características quanto o modelo de tipos de negócio utilizam os relacionamentos de herança, composição, agregação e associação, propostos pela UML, esta heurística sugere que estes relacionamentos sejam mantidos no modelo de tipos de negócio correspondente. A Figura 3.14 mostra a heurística proposta e um exemplo de mapeamento, ilustrando os relacionamentos de composição e agregação.

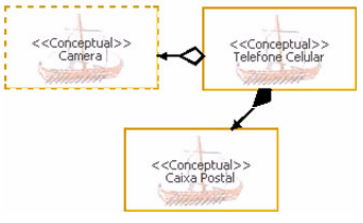
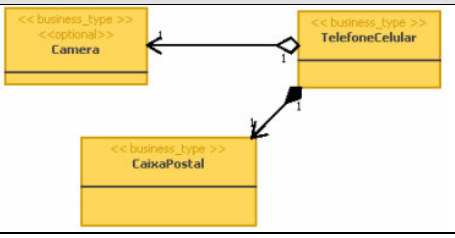
Os mesmos relacionamentos de composição, agregação e herança entre duas características devem existir entre os tipos de negócio correspondentes.	
Características Conceituais usando relacionamentos da UML	**Tipos de Negócio relacionados**
	

Figura 3.14: Mapeamento Sugerido pela Heurística TN4

3.4.1.5 - Heurística TN5

Propõe-se que uma Regra de Composição Inclusiva do Odyssey-FEX seja representada no modelo de tipos de negócio, usando a associação simples. Uma outra forma de representar esta regra é especificando tal dependência através das propriedades do tipo de negócio, indicando quais outros tipos de negócio um determinado tipo é dependente. A Figura 3.15 ilustra a heurística.

3.4.1.6 - Heurística TN6

Como não existe na notação de tipos de negócio um relacionamento que represente a mesma semântica da Regra de Composição Exclusiva, a heurística TN6 sugere que sejam utilizados estereótipos para este fim. Por exemplo, a representação <<XOR Z>> significa que um ou mais tipos de negócio, representado pela letra Z, não devem existir concomitantemente com o tipo de negócio em que este estereótipo foi definido. A Figura 3.16 mostra a heurística e o mapeamento sugerido. A regra de composição Exclui é representada por um X no canto inferior esquerdo de cada característica e no tipo de negócio através do estereótipo <<XOR ...>>.

Para cada regra de composição do tipo "requer", deve ser especificada a dependência no tipo de negócio que corresponde ao antecedente da regra, através das propriedades do próprio tipo. Uma outra opção de mapeamento é utilizando o relacionamento de associação entre os tipos de negócio envolvidos. Se utilizado tal relacionamento, deve ser especificada a sua navegabilidade, onde a sua origem deve ser atribuída ao tipo de negócio que representa o antecedente da regra, e o destino da associação, deve ser o conseqüente.

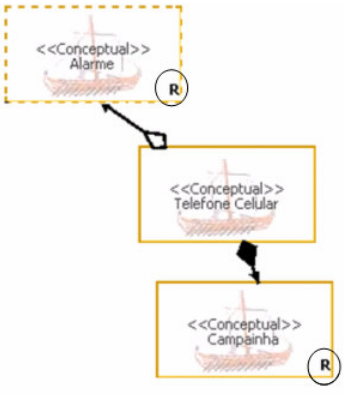
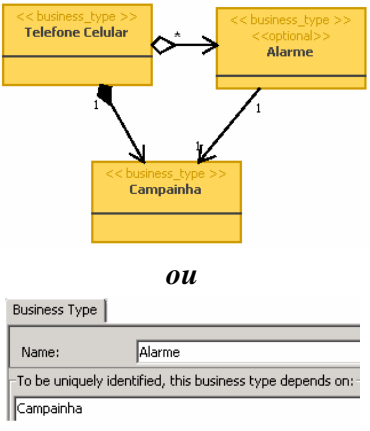
Características Conceituais com uma regra de composição Requer	**Possíveis Resultados do Mapeamento**
	

Figura 3.15: Mapeamento Sugerido pela Heurística TN5

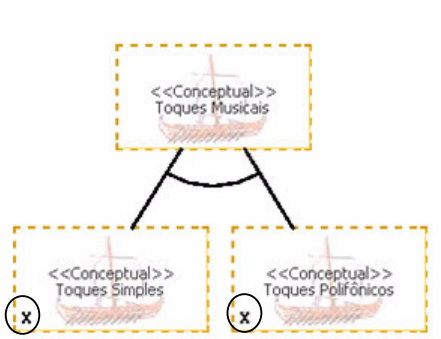
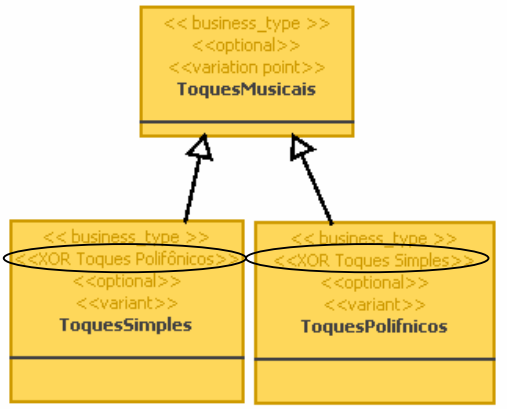
Para cada regra de composição do tipo "exclui", deve existir um estereótipo <<xor>> nos tipos de negócio que correspondem às características envolvidas.	
Características Conceituais com regra de composição Exclui	**Resultados do Mapeamento**
	

Figura 3.16: Mapeamento Sugerido pela Heurística TN6

3.4.1.7 - Heurística TN7

A propriedade de opcionalidade de uma característica conceitual deve ser mantida em seu respectivo tipo de negócio. Assim, para representar esta opcionalidade em um tipo de negócio, será utilizado, nesta tese, o estereótipo <<optional>>. Tipos de negócio que não possuem tal estereótipo são considerados mandatórios. A Figura 3.17 ilustra um exemplo.

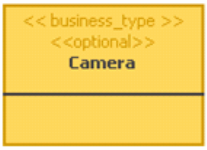
A opcionalidade deve ser transposta para o tipo de negócio ou atributo que represente a característica em questão.	
Característica conceitual opcional	**Resultado do Mapeamento**
	

Figura 3.17: Mapeamento Sugerido pela Heurística TN7

3.4.1.8 - Heurística TN8

Em algumas situações, características de entidades podem também representar informações que futuramente serão persistidas através de um componente. Neste caso, esta heurística permite que o Engenheiro de Domínio tenha a possibilidade de gerar tipos de negócio a partir de características de entidade do domínio. A Figura 3.18 apresenta a descrição da heurística e ilustra um exemplo.

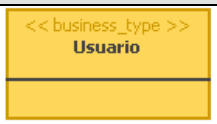
Características classificadas como Entidade só serão mapeadas para tipos de negócio se também representarem um conceito do domínio.	
Característica de entidade	**Resultado do Mapeamento**
	

Figura 3.18: Mapeamento Sugerido pela Heurística TN8

3.4.1.9 - Heurística TN9

Esta heurística só deve ser utilizada se o Engenheiro de Domínio optar por também adotar a heurística TN8. Assim, a relação entre uma característica de entidade e uma característica conceitual deve ser mapeada para um relacionamento de associação entre tipos de negócio. A Figura 3.19 apresenta a descrição da heurística e ilustra um exemplo.

3.4.2 - Variabilidade em Casos de Uso (CUso)

Casos de uso do domínio representam aspectos funcionais de um domínio e descrevem possíveis operações sobre artefatos estáticos, ou seja, sobre os tipos de negócio previamente criados. Neste caso, as características de opcionalidade, variabilidade, regras de composição e relacionamentos existentes entre características funcionais devem, também, ser representadas nos casos de uso do domínio correspondentes. A fim de manter tais consistências, foram definidas heurísticas para apoiar o mapeamento de características funcionais em casos de uso do domínio, as quais estão descritas nas seguintes subseções 3.4.2.1 à 3.4.2.9.

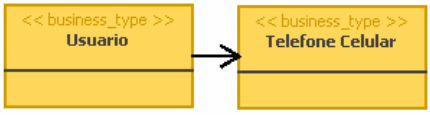
O relacionamento "Communication Link" entre uma característica de entidade e uma característica conceitual é mapeado para uma associação entre os dois tipos de negócio envolvidos. A origem da relação ocorre a partir do tipo de negócio que representa a característica de entidade e o destino da mesma corresponde à característica conceitual.	
Características relacionadas através de Communication Link	**Resultado do Mapeamento**
	

Figura 3.19: Mapeamento Sugerido pela Heurística TN9

Como resultado destas dez heurísticas, podem ser gerados casos de uso que representam aspectos funcionais de um domínio durante a sua fase de análise, porém

num nível de abstração mais baixo do que as características funcionais. São gerados casos de uso mandatórios ou opcionais, que podem representar invariantes, pontos de variação e variantes do domínio, com seus respectivos relacionamentos e restrições (e.g. estereótipo XOR).

3.4.2.1 - Heurística CUso1

Esta heurística define quais características do domínio não devem ser mapeadas para casos de uso do domínio.

Características conceituais e tecnológicas não devem ser mapeadas para casos de uso.

3.4.2.2 - Heurística CUso2

A heurística CUso2 é considerada uma heurística essencial dentre este conjunto proposto, uma vez que ela determina quais características dão origem a casos de uso do domínio. A Figura 3.20 apresenta a heurística e um exemplo de mapeamento.

Assim como a heurística TN2, esta heurística deve ser adotada pelo Engenheiro de Domínio para que as demais (i.e. CUso3 à Cuso9) também possam ser mapeadas. Portanto, as heurísticas CUso3 à Cuso9 assumem que tal mapeamento foi feito. Da mesma forma, se o Engenheiro de Domínio entender que a característica funcional não necessita ser tratada futuramente pelas aplicações derivadas deste domínio, então, esta e as demais heurísticas para caso de uso não deverão ser utilizadas.

Uma característica funcional pode ser mapeada para um caso de uso do domínio.	
Característica Funcional	**Caso de Uso Gerado**
	

Figura 3.20: Mapeamento Sugerido pela Heurística CUso2

3.4.2.3 - Heurística CUso3

A heurística CUso3 trata do mapeamento de características funcionais que representam pontos de variação e variantes em casos de uso de domínio.

Dentre os relacionamentos existentes na notação de casos de uso, o de herança foi o que mais se aproximou da semântica do relacionamento Alternativo da notação

Odyssey-FEX, tendo que ser complementado com os estereótipos <<variation point>> e <<variant>>. A Figura 3.21 apresenta a heurística e um exemplo de mapeamento.

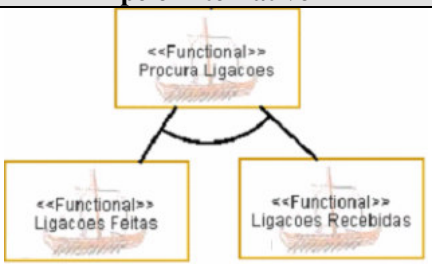
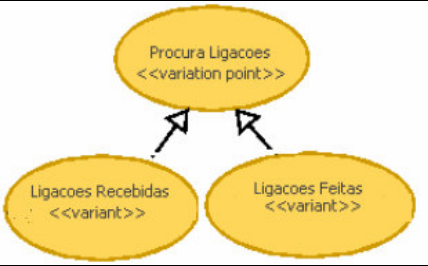
Para cada ponto de variação e suas variantes, devem existir casos de uso com estereótipos <<variation point>> e <<variant>>, respectivamente, relacionados através de herança.	
Características Funcionais relacionadas pelo Alternativo	**Casos de Uso Gerados**
	

Figura 3.21: Mapeamento Sugerido pela Heurística CUso3

3.4.2.4 - Heurística CUso4

A notação Odyssey-FEX utiliza relacionamentos de composição da UML para representar relacionamentos entre características funcionais. Considerando que casos de uso só prevêem os relacionamentos de herança, *include*, *extend* e *communication link*, esta heurística sugere que características funcionais relacionadas por composição sejam mapeadas para casos de uso relacionados por *include*. A Figura 3.22 apresenta a heurística e um exemplo de mapeamento.

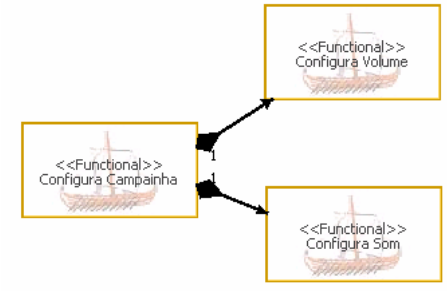
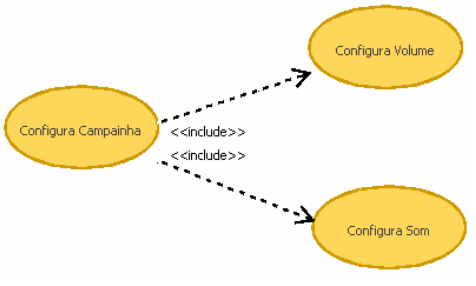
Para cada relacionamento de composição entre características funcionais, deve existir um relacionamento de include entre os casos de uso correspondentes. O caso de uso que origina a relação de include deve representar a característica funcional que representa o todo da relação de composição. As partes da relação de composição devem ser representadas pelo caso de uso destino da relação de include.	
Características Funcionais relacionadas por Composição	**Casos de Uso Gerados**
	

Figura 3.22: Mapeamento Sugerido pela Heurística CUso4

3.4.2.5 - Heurística CUso5

A heurística CUso5 trata do mapeamento de características funcionais relacionadas por herança, sugerindo a manutenção do mesmo relacionamento entre casos de uso que correspondem às características funcionais envolvidas. A Figura 3.23 apresenta a heurística e um exemplo de mapeamento.

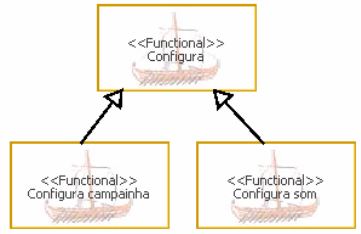
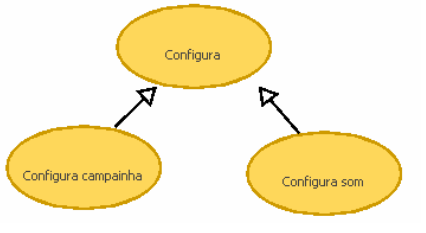
Para características funcionais relacionadas por herança, devem existir casos de uso relacionados da mesma forma.	
Características Funcionais relacionadas por Herança	**Casos de Uso Gerados**
	

Figura 3.23: Mapeamento Sugerido pela Heurística CUso5

3.4.2.6 - Heurística CUso6

A heurística CUso6 trata do mapeamento da regras de composição de inclusão entre características funcionais. Se existe uma ou mais características funcionais que requerem uma ou mais características funcionais, representadas através de uma regra de composição, então, devem existir casos de uso, cada um com sua respectiva descrição de pré-condição. Cada caso de uso deve se relacionar ao seu respectivo ator. A Figura 3.24 apresenta a heurística e um exemplo de mapeamento, salientados pelos círculos.

3.4.2.7 - Heurística CUso7

Esta heurística trata do mapeamento da regras de composição de exclusão entre características funcionais. Para representar esta regra, devem existir casos de uso do domínio estereotipados com a notação <<XOR Y>>, onde Y representa um ou mais casos de uso do domínio que fazem parte desta exclusão. Foi utilizado o recurso de estereótipo, pois não existe um relacionamento na UML que permita expressar esta condição de existência entre casos de uso. A Figura 3.25 apresenta a heurística Cuso7 e um exemplo de mapeamento.

Para cada regra de composição do tipo "requer" envolvendo características funcionais, deve existir uma pré-condição especificada no caso de uso que equivale a característica funcional que faz parte do antecedente da regra. Nesta descrição de pré-condição, deve ser registrado que o caso de uso requer um outro que equivale ao conseqüente da regra de composição original.

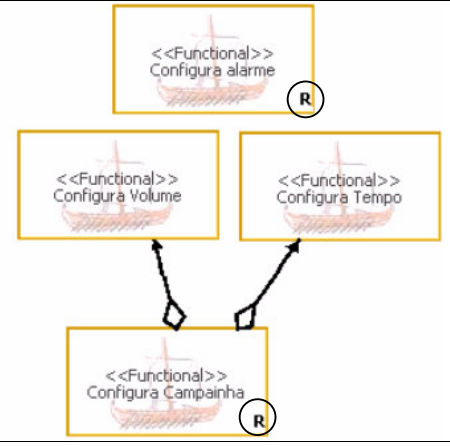
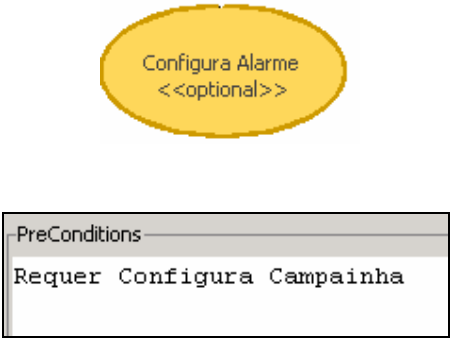
Características Funcionais que possuem regras de composição Requer	Casos de Uso Gerados
	

Figura 3.24: Mapeamento Sugerido pela Heurística Cuso6

Para cada regra de composição do tipo "excluir"envolvendo características funcionais, deve existir um estereótipo <<XOR ... >> nos casos de uso que correspondem às características envolvidas.

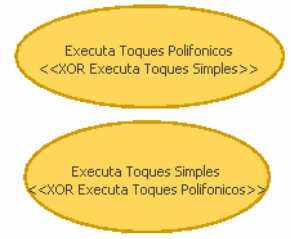
Características Funcionais que possuem regras de composição Exclui	Casos de Uso Gerados
	

Figura 3.25: Mapeamento Sugerido pela Heurística Cuso7

3.4.2.8 - Heurística CUso8

A heurística CUso8 indica que a opcionalidade das características funcionais deve também ser mapeada para os casos de uso. A opcionalidade de um caso de uso é representada através do estereótipo <<optional>>. A Figura 3.26 apresenta a heurística e um exemplo de mapeamento. A ausência deste estereótipo indica a obrigatoriedade do caso de uso.

A opcionalidade deve ser transposta para o caso de uso que represente a característica funcional em questão. O caso de uso receberá o estereótipo <<optional>>.	
Característica Funcional Opcional	**Caso de Uso Gerado**
	

Figura 3.26: Mapeamento Sugerido pela Heurística CUso8

3.4.2.9 - Heurística CUso9

Características de entidade podem ser mapeadas para um ator de caso de uso. A Figura 3.27 apresenta a heurística e um exemplo de mapeamento sugerido.

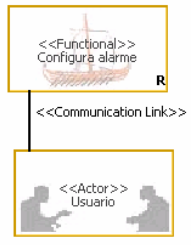
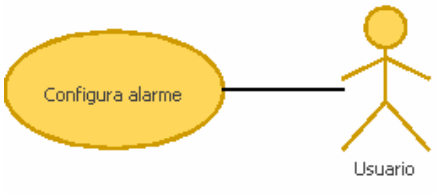
Características de Entidade podem ser mapeadas para atores de casos de uso se tal entidade representa um elemento externo ao domínio. Neste caso, o relacionamento <<communication link>> entre as características de entidade e funcionais, deverá ser mapeado para um relacionamento equivalente entre um ator e um caso de uso.	
Característica de Entidade	**Ator de Casos de Uso**
	

Figura 3.27: Mapeamento Sugerido pela Heurística CUso9

3.4.3 - Variabilidade em Componentes do Domínio

Na DECOM, os artefatos de projeto são representados por meio de componentes. As heurísticas apresentadas nesta Seção lidam com a preservação de todas as propriedades e relacionamentos especificados nos respectivos artefatos de análise, com ênfase na variabilidade, opcionalidade e relacionamentos. As heurísticas CN1 à CN6, se referem ao mapeamento de Tipos de Negócio em Componentes de Negócio. As heurísticas CP1 à CP4, auxiliam o mapeamento de casos de uso em componentes de processo. As heurísticas CUI1 à CUI5 auxiliam o engenheiro de domínio no mapeamento de características de ambiente operacional, tecnologia de domínio e técnica de implementação, em componentes utilitários ou de infra-estrutura.

No tocante aos componentes de negócio, não existe uma heurística específica que considere originalmente a proposta de Teixeira (2003) para a geração destes componentes. Isto ocorre porque nesta tese, além dos relacionamentos avaliados por

Teixeira, são também avaliados, em sua maioria ortogonalmente, as variabilidades, opcionalidades e regras de composição dos tipos de negócio. As heurísticas CN1, CN3, CN4 e CN5 auxiliam, além do mapeamento ou agrupamento de tipos de negócio em função de seus respectivos relacionamentos, também o devido mapeamento das variabilidades e opcionalidades dos tipos envolvidos.

Na abordagem DECOM, somente tipos de negócio podem ser agrupados para gerar um único componente de negócio. Aos demais artefatos (e.g. características tecnológicas e casos de uso) que são usados para gerar componentes de processo, utilitário e de infra-estrutura, não é indicado o agrupamento. Tal justificativa se deve aos seguintes fatores:

- Tipos de negócio representam conceitos do domínio que podem na maioria das vezes ser reutilizados em conjunto por uma mesma aplicação;
- Casos de uso são artefatos do domínio que pode realizar passos referentes a mais de um tipo de negócio. Ao agrupar dois ou mais casos de uso num único componente de processo, este último poderia realizar operações referentes a um número elevado de tipos de negócio, isto é, aos seus respectivos componentes de negócio. Por esta razão, é tomada a decisão de não agrupar casos de uso entre si, mas sim, deixar que tal agrupamento possa ser decidido durante a construção dos elementos arquiteturais do domínio.
- Características tecnológicas por sua vez não serão agrupadas neste contexto por prestarem serviços bem definidos e autocontidos. O agrupamento destas características em componentes utilitários e de infra-estrutura pode limitar o potencial de reutilização dos mesmos.

As próximas seções detalham cada uma das heurísticas propostas.

3.4.3.1 - Heurística CN1

A Heurística CN1 trata do mapeamento de tipos de negócio relacionados por herança em componentes de negócio. Neste caso, se dois os mais tipos estiverem relacionados desta forma, a heurística sugere, conforme descrito na Seção 3.3.2.1.2, o agrupamento destes num único componente de negócio. Este componente de negócio gerado deve receber a variabilidade e opcionalidade do tipo de negócio que representa o

pai da relação de herança. A Figura 3.28 mostra a descrição da heurística e um exemplo de mapeamento.

Para tipos de negócio relacionados por meio de herança, deve ser gerado um componente de negócio com a especificação de variabilidade e opcionalidade do tipo de negócio que representa o pai desta relação. O Componente gerado deve prover uma interface para cada tipo de negócio filho nesta relação.

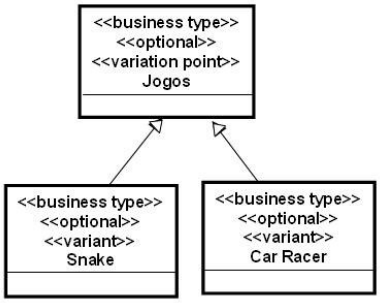
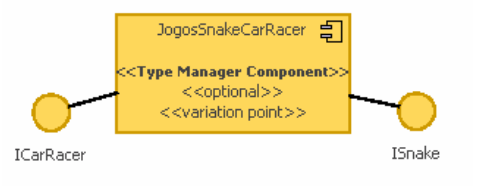
Tipos de Negócio	Componente de Negócio Gerado
	

Figura 3.28: Mapeamento Sugerido pela Heurística CN1

Contudo esta heurística não é aplicável se os tipos de negócio filhos da relação possuem estereótipo <<XOR>>. Tal restrição será tratada em mais detalhes pela heurística CN2, descrita na Seção 3.4.3.2.

3.4.3.2 – Heurística CN2

Esta heurística determina que tipos de negócio com estereótipo XOR não podem ser agrupados uns com os outros. Tipos de negócio que possuem este estereótipo estão relacionados por meio de herança, representando os filhos desta relação. Se os tipos de negócio filhos de um mesmo pai, representando um ponto de variação, são mutuamente excludentes (e.g. toques simples e toques polifônicos), então não faz sentido o seu agrupamento através de um componente de negócio. Tal restrição é, portanto, tratada pela heurística CN2. A Figura 3.29 mostra a descrição da heurística e um exemplo no domínio de Telefonia Móvel. Neste caso, os componentes de negócio que representam o filho da relação devem requerer a interface do componente que representa o pai.

3.4.3.3 - Heurística CN3

A Heurística CN3 avalia o agrupamento de tipos de negócio quando estes estão relacionados por composição. Vale lembrar que tipos de negócio relacionados desta forma devem ter a mesma especificação de opcionalidade (todos mandatórios ou todos

opcionais). Conforme descrito na Seção 3.3.2.1.2, é recomendado que tipos de negócio relacionados por composição sejam agrupados num único componente. Por esta razão, esta heurística apresenta duas opções de mapeamento para tipos de negócio relacionados por composição: a) se os tipos forem agrupados, é gerado um componente de negócio com uma interface para cada tipo agrupado; b) se não forem agrupados, o componente que representa o todo desta relação deve requerer a interface do componente que representa a parte. Em ambos os casos, os componentes mantêm a especificação de opcionalidade dos tipos de negócio correspondentes. A Figura 3.30 mostra a descrição da heurística CN3 e um exemplo ilustrando uma situação onde dois tipos foram agrupados, e outros dois, relacionados da mesma forma, não foram.

Não devem ser agrupados tipos de negócio que possuem estereótipos XOR.

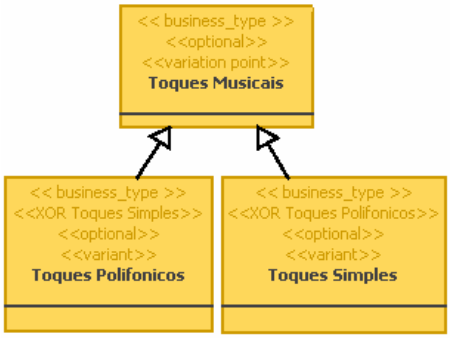
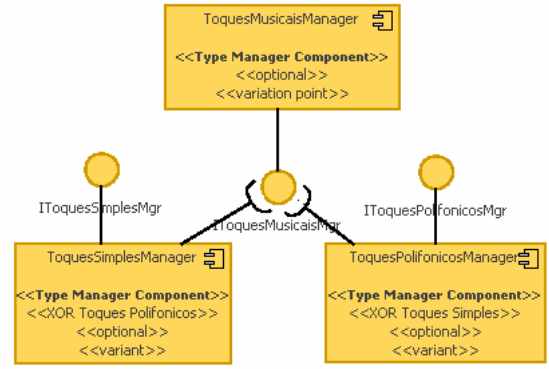
Tipos de Negócio	Componentes de Negócio Gerado
	

Figura 3.29: Mapeamento Sugerido pela Heurística CN2

Para tipos de negócio relacionados por composição, pode ser gerado um único componente com uma interface para cada tipo agrupado. Se não forem agrupados, o componente que representa o todo deve requerer a interface do componente que representa a parte da relação. Em ambos os casos, devem ser mantidos a opcionalidade dos tipos envolvidos.

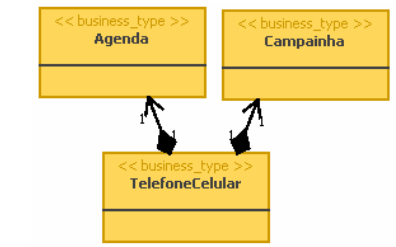
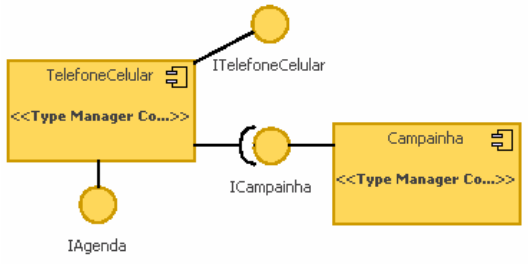
Tipos de Negócio	Componentes de Negócio Gerado
	

Figura 3.30: Mapeamento Sugerido pela Heurística CN3

3.4.3.4 - Heurística CN4

A Heurística CN4 sugere o agrupamento de tipos de negócio quando estão relacionados por agregação. Conforme descrito na Seção 3.3.2.1.2, é recomendado que tipos de negócio relacionados por agregação sejam agrupados num único componente. Diferente da CN3, onde os tipos deveriam ter a mesma especificação de opcionalidade, esta heurística considera que tipos de negócio relacionados por agregação podem ter especificações de opcionalidade diferentes. Assim, caso os tipos sejam agrupados, o componente de negócio receberá a opcionalidade do tipo que representa o todo da relação. Por outro lado, se não forem agrupados, o componente de negócio que representa o todo da relação deve requerer a interface do componente que representa a parte. A Figura 3.31 mostra a descrição da heurística e um exemplo que ilustra um agrupamento e uma situação onde tipos não foram agrupados.

Para tipos de negócio relacionados por agregação, é indicada a geração de um componente, o qual recebe a especificação de opcionalidade do tipo de negócio que representa o todo desta relação. Este componente deve fornecer uma interface para cada tipo de negócio agrupado. Se não forem agrupados, deve haver uma relação de dependência entre os componentes, onde aquele que representa o todo da relação deve requerer a interface do componente que representa a parte.

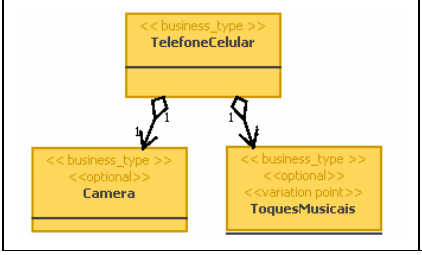
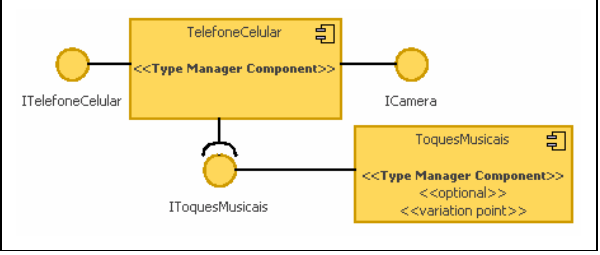
Tipos de Negócio	Componentes de Negócio Gerado
	

Figura 3.31: Mapeamento Sugerido pela Heurística CN4

3.4.3.5 - Heurística CN5

A Heurística CN5 indica como deve ocorrer o agrupamento de tipos de negócio relacionados por associação ou com uma indicação de dependência especificada no tipo de negócio. Neste caso, estes tipos somente serão agrupados se todos tiverem a mesma opcionalidade e, como resultado deste agrupamento será gerado um componente fornecendo uma interface para cada tipo de negócio. Por outro lado, se não forem agrupados, o componente que equivale ao tipo de negócio que origina a relação deve requerer a interface do componente que equivale ao tipo de negócio destino. A Figura 3.32 mostra a descrição da heurística e um exemplo de utilização.

Para tipos de negócio relacionados por associação ou dependência, é indicada a geração de um componente de negócio quando todos os tipos envolvidos são mandatórios ou opcionais. Se não forem agrupados, deve-se especificar a dependência entre os componentes gerados, isto é, o componente que origina a relação deve requerer a interface do componente que representa o destino da relação.

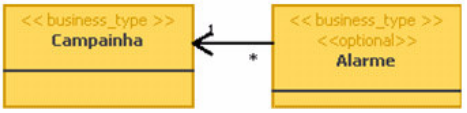
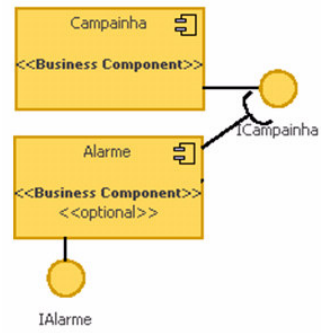
Tipos de Negócio	Componentes de Negócio Gerado
 ***ou*** Tipos de Negócio Alarme *depende* do Tipo de Negócio Campainha	

Figura 3.32: Mapeamento Sugerido pela Heurística CN5

3.4.3.6 -Heurística CN6

Esta heurística determina que, durante um processo de agrupamento de tipos de negócio em componentes, a obrigatoriedade deve prevalecer sobre a opcionalidade, exceto quando estiverem relacionados por agregação. Neste último caso, conforme descrito na heurística CN4, deve prevalecer a opcionalidade do tipo de negócio que representa o pai da relação. Assim, no caso da CN6, se tipos de negócio mandatórios e opcionais forem agrupados, o componente resultante deve ser mandatório. A Figura 3.33 mostra a descrição e um exemplo de aplicação desta heurística.

Deve prevalecer a característica de obrigatoriedade no componente de negócio quando forem agrupados tipos de negócio mandatórios e opcionais.

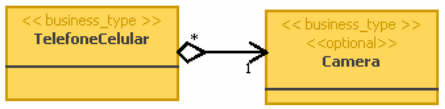
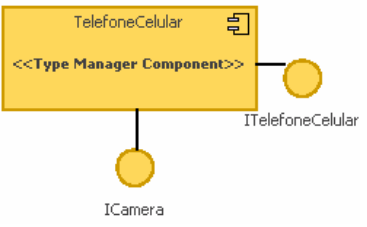
Tipos de Negócio	Componente de Negócio Gerado
	

Figura 3.33: Mapeamento Sugerido pela Heurística CN6

3.4.3.7 - Heurística CP1

Esta heurística é considerada essencial dentre o conjunto proposto para mapeamento de casos de uso em componentes de processo. A CP1 determina quais casos de uso dão origem a componentes de processo. Esta heurística deve ser adotada pelo Engenheiro de Domínio para que as demais (i.e. CP2 à CP4 também possam ser utilizadas). Portanto, as heurísticas CP2 à CP4 assumem que a CP1 foi utilizada.

Cada caso de uso do domínio deve ter seus respectivos componentes de processo, preservando sua variabilidade e opcionalidade. A Figura 3.34 apresenta a heurística e um exemplo de mapeamento.

Para cada caso de uso do domínio, deve ser gerado um componente de processo, com suas respectivas interfaces fornecidas. Todos os estereótipos de variabilidades (i.e. <<variant>>, <<variation point>>), opcionalidades (i.e. <<optional>>) e regras de composição (i.e.<<XOR>>) devem ser igualmente mapeados para os respectivos componentes de processo.

Casos de Uso	Componente de Processo

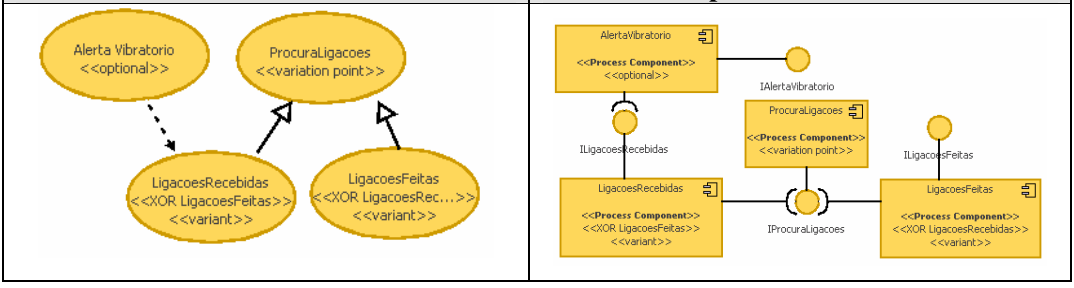


Figura 3.34: Mapeamento Sugerido pela Heurística CP1

3.4.3.8 - Heurística CP2

A heurística CP2 sugere o mapeamento do relacionamento de herança entre casos de uso. Os componentes de processo que representam os filhos da relação de herança devem requerer a interface do componente que representa o pai desta relação. A Figura 3.35 mostra a descrição da heurística e um exemplo de utilização.

Se os casos de uso estão relacionados por meio de herança, os componentes de processo que são derivados dos casos de uso que representam os filhos desta relação devem requerer as interfaces do componente de processo que representa o pai.

Casos de Uso	Componentes de Processo

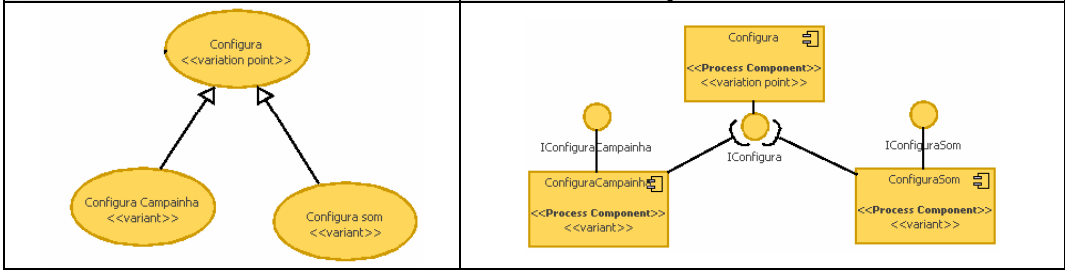


Figura 3.35: Mapeamento Sugerido pela Heurística CP2

3.4.3.9 - Heurística CP3

A especificação de pré-condição de um caso de uso determina que a realização de um envolve também a realização de outro(s). Portanto, no mapeamento para componentes de processo, aquele que representa um caso de uso que contém a especificação de pré-condição deve requerer a interface do componente de processo que representa o caso de uso descrito na pré-condição. Por exemplo, na Figura 3.36, o caso de uso Configura Volume possui a pré-condição de requerer Configura Campainha. Neste caso, durante o mapeamento, o componente de processo Configura Volume deve requerer a interface do componente de processo Configura Campainha.

Se um caso de uso especifica uma pré-condição, então o seu respectivo componente de processo deve requerer a interface do componente de processo que representa o caso de uso descrito na pré-condição.

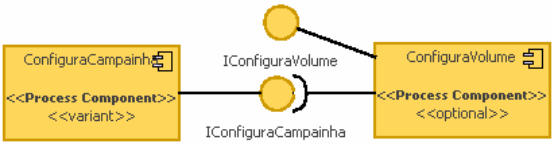
Casos de Uso	Componentes de Processo
 Pré-condição: requer Configura Campainha	

Figura 3.36: Mapeamento Sugerido pela Heurística CP3

3.4.3.10 - Heurística CP4

Relacionamentos de dependência, *include e association* entre casos de uso também devem ser mapeados para relacionamentos entre componentes de processo. No mapeamento para componentes de processo, aquele que representa uma origem de uma destas relações anteriormente mencionadas deve requerer a interface do componente de processo que representa o destino da relação. Por exemplo, na Figura 3.37, o caso de uso Alerta Vibratório depende da realização do caso de uso Ligações Recebidas. Neste caso, durante o mapeamento, o componente de processo Alerta Vibratório deve requerer a interface do componente Ligações Recebidas.

3.4.3.11 - Heurística CUI1

A heurística CUI1 é essencial dentre o conjunto de heurísticas para estas categorias de componentes utilitários e de infra-estrutura, pois determina a criação de um componente para cada característica de ambiente operacional, tecnologia de domínio e técnica de implementação. Nesta heurística, a opcionalidade da característica tecnológica (i.e. mandatória ou opcional) deve ser mantida no seu respectivo

componente utilitário ou de infra-estrutura. A Figura 3.38 mostra a descrição da heurística e um exemplo de mapeamento.

Se os casos de uso estão relacionados por uma dependência, então o componente de processo que representa a origem desta relação deve requerer a interface do componente de processo que representa o destino esta relação.

Casos de Uso	Componentes de Processo

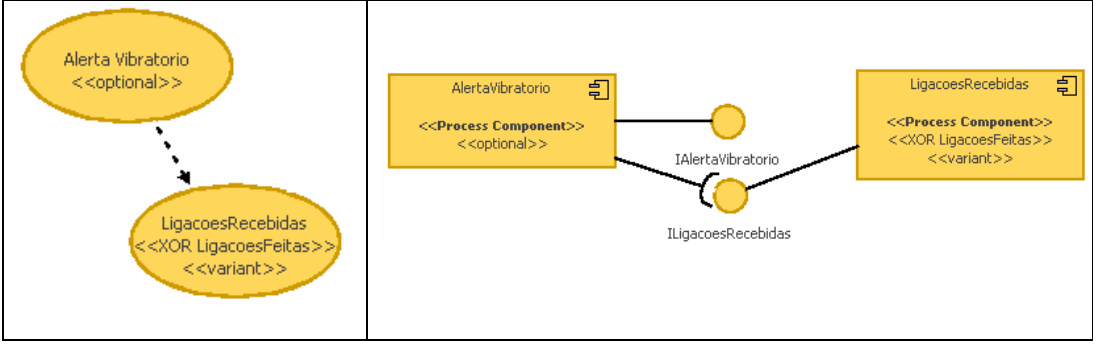


Figura 3.37: Mapeamento Sugerido pela Heurística CP4

Cada característica de ambiente operacional deve ser mapeada para um componente de infra-estrutura. Cada característica de tecnologia de domínio e técnica de implementação deve ser mapeada para um componente utilitário correspondente.

Características Tecnológicas	Componentes Utilitários ou Infra-estrutura

Figura 3.38: Mapeamento Sugerido pela Heurística CUI1

3.4.3.12 - Heurística CUI2

Características de ambiente operacional, tecnologia de domínio e técnica de implementação podem se relacionar através de um <<implemented by>>, conforme previsto pela notação Odyssey-FEX. Quando ocorre o mapeamento de suas respectivas categorias para componentes, o componente que representa a característica que dá origem à relação deve requerer a interface do componente correspondente à característica destino desta relação. A Figura 3.39 mostra a descrição da heurística e um exemplo de mapeamento.

3.4.3.13 - Heurística CUI3

Esta heurística propõe o mapeamento do relacionamento Alternativo, existente entre características de ambiente operacional, tecnologia de domínio e técnica de implementação, em componentes utilitários e de infra-estrutura. Assim, o componente

utilitário e de infra-estrutura que representa uma característica variante, deve requerer a interface do componente que representa o seu ponto de variação. Cada componente variante recebe o estereótipo <<variant>>, e os pontos de variação, o estereótipo <<variation point>>. Além disso, caso as características que representam variantes fazerem parte de uma regra de exclusão, os componentes correspondentes devem receber o estereótipo <<XOR>>. A Figura 3.40 mostra a descrição da heurística e um exemplo de mapeamento.

Para características relacionadas através de uma ligação <<implemented by>>, o componente utilitário ou infra-estrutura que representa a característica que origina esta relação deve requerer a interface do componente que representa a característica que é o destino.

Características Tecnológicas	Componentes Utilitários ou Infra-estrutura

Figura 3.39: Mapeamento Sugerido pela Heurística CUI2

Características tecnológicas relacionadas por "Alternative" devem ser mapeadas para componentes, onde aqueles que representam as variantes, identificadas pelo estereótipo <<variant>>, devem requerer a interface daquele que representa o ponto de variação, representado pelo estereótipo <<variation point>>. Ainda, se as características que foram mapeadas para componentes variantes fazem parte de uma regra de composição exclui, então, deve ser criado, no componente correspondente, o estereótipo <<XOR>>.

Características Tecnológicas	Componentes Utilitários ou Infra-estrutura

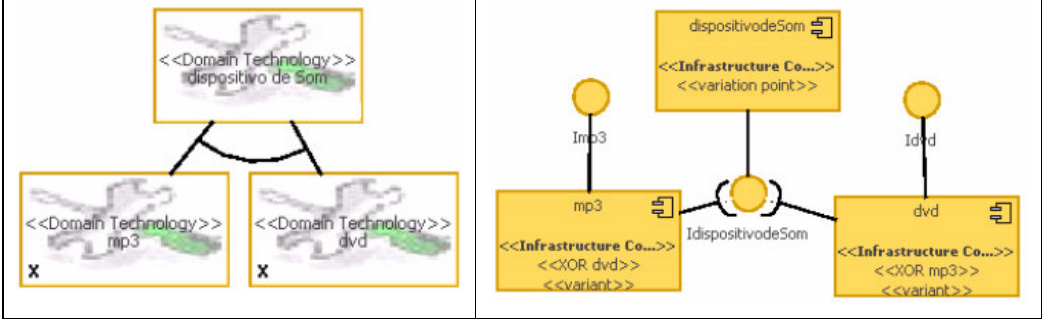


Figura 3.40: Mapeamento Sugerido pela Heurística CUI3

3.4.3.14 - Heurística CUI4

Esta heurística propõe o mapeamento dos relacionamentos de herança, composição, agregação ou associação entre características tecnológicas para os seus respectivos componentes utilitários e de infra-estrutura. Se estiverem relacionadas por herança, as respectivas interfaces dos componentes que representam as características filhas desta relação devem requerer a interface do componente que representa a característica pai. Se as características estiverem relacionadas por associação, composição ou agregação, os componentes que representarem o todo ou origem da relação devem requerer a interface do componente que representa a parte ou destino da mesma relação. A Figura 3.41 mostra a descrição da heurística e um exemplo.

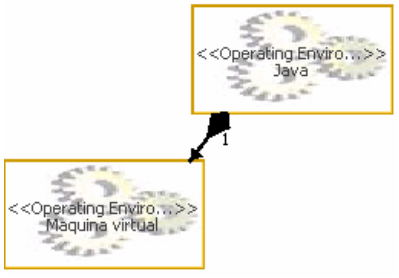
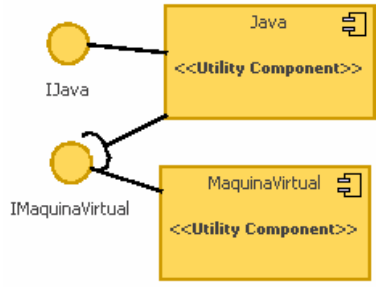
Para características tecnológicas relacionadas por associação, composição e agregação, o componente utilitário ou infra-estrutura que representa o todo ou origem deve requerer a interface do componente que representa a parte ou destino da relação.	
Características Tecnológicas	**Componentes Utilitários ou Infra-estrutura**
	

Figura 3.41: Mapeamento Sugerido pela Heurística CUI4

3.4.3.15 - Heurística CUI5

A última heurística proposta para o mapeamento destas categorias de características trata das regras de composição *include*. Caso tais regras existam, os componentes que foram mapeados a partir das características referentes ao antecedente da regra, devem requerer as interfaces dos componentes que foram mapeados a partir do conseqüente. A Figura 3.42 mostra a descrição e um exemplo da CUI5.

3.4.4 - Considerações Finais sobre as Heurísticas de Mapeamento

A Seção 3.4 apresentou um conjunto de heurísticas para apoio ao mapeamento de diferentes artefatos da análise de domínio para componentes gerados durante a fase

de projeto no CBD-Arch-DE. Este conjunto de heurísticas visa contemplar grande parte dos mapeamentos possíveis entre os artefatos do domínio explorado no CBD-Arch-DE.

Características tecnológicas que fazem parte de uma regra de composição "inclui", o componente utilitário ou infra-estrutura que representa um característica que faz parte do antecedente da regra deve requerer a interface do componente que representa a característica referente ao conseqüente da mesma regra.

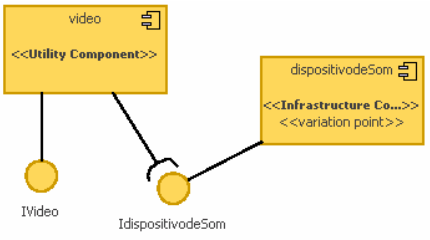
Características Tecnológicas	Componentes Utilitários ou Infra-estrutura
 Vídeo **requer** dispositivo de Som	

Figura 3.42: Mapeamento Sugerido pela Heurística CUI5

As heurísticas propostas tratam do mapeamento inter-modelos, ou seja, dos relacionamentos, variabilidades, opcionalidades e regras de composição entre artefatos de diferentes níveis de abstração. No contexto de um processo de ED, tais mapeamentos representam um grande esforço, influenciando diretamente na construção e na consistência das arquiteturas de referência do domínio.

Embora as heurísticas contemplem os mapeamentos encontrados nos domínios analisados (e.g. agropecuário, máquina de processo e persistência), não necessariamente significa que este conjunto esteja completo. Assim, com o uso destas heurísticas em outros domínios, novas necessidades de mapeamento podem vir a surgir, ampliando este conjunto inicial.

3.5 - Critérios para a Identificação de Elementos Arquiteturais de um Domínio

A geração de componentes a partir de artefatos de análise, tal como proposto no CBD-Arch-DE, tem como resultado um grande número de componentes. A organização destes componentes numa arquitetura de domínio é um aspecto chave para a obtenção de bons resultados. Quanto maior o número de componentes envolvidos durante a troca de mensagens, mais complexa pode se tornar a arquitetura de referência do domínio, comprometendo conseqüentemente a sua compreensão, manutenibilidade e, sobretudo, a sua reutilização.

Para melhorar a qualidade esperada da arquitetura de referência, é aconselhável agrupar componentes que trocam mensagens entre si através de um único artefato, definido nesta tese como um elemento arquitetural que representa um agrupamento de componentes. Estes componentes são agrupados dentro de um novo componente (agrupamento de componentes) junto aos componentes originais. Esta tarefa de agrupamento de componentes (BLOIS *et al.*, 2005b), deve ser executada para que se possa compreender o motivo pelo qual os componentes interagem intensivamente e então organizá-los numa arquitetura de referência, baseado em algum critério adequado. Como resultado desta organização, são geradas arquiteturas de referência de melhor entendimento, influenciado no aumento de reutilização desta arquitetura durante um processo de Engenharia de Aplicação.

Métodos de DBC não apresentam propostas para estruturar componentes numa arquitetura, tampouco algum critério de agrupamento de componentes em particular. Os trabalhos discutidos na Seção 2.4.3 avaliam o grau de acoplamento e coesão dos artefatos para propor agrupamentos dos mesmos. Estes trabalhos estão mais dirigidos às questões de acoplamento num nível de granularidade mais baixo (e.g. classes). Nestes trabalhos, são avaliados os relacionamentos existentes entre classes de uma aplicação para sugerir o acoplamento de classes através de componentes. Por outro lado, componentes, no contexto de uma arquitetura, também possuem um alto acoplamento em função da troca de mensagens através de suas interfaces ou por informações a respeito da semântica do domínio de aplicações.

Por esta razão, são propostos nesta tese critérios que avaliam a interação existente entre componentes, além de sugerir o seu agrupamento através de elemento arquitetural. Estes critérios são complementares à literatura existente, principalmente em relação as métricas propostas por (HOEK *et al.*, 2003). Conforme discutido na Seção 2.4.3, as métricas propostas pelos autores somente avaliam a quantidade de interfaces providas e requeridas para propor o acoplamento de componentes. No entanto, num contexto de ED, outras informações (i.e. categoria dos componentes, a variabilidade, opcionalidade, rastro existente entre artefatos do domínio) devem também ser consideradas para gerar um elemento arquitetural.

Embora os critérios de agrupamento tenham sido propostos num contexto de ED, a maioria deles pode ser usada em qualquer atividade de engenharia de software, para ajudar o projetista a agrupar componentes de software.

No contexto do CBD-Arch-DE são utilizados como elementos de entrada os componentes de negócio, processo, utilitário e de infra-estrutura, resultantes da atividade de Geração de Componentes. A partir destes componentes, os critérios podem ser aplicados, gerando como resultado, componentes de maior granularidade, que representam agrupamentos de componentes que constituem elementos arquiteturais de possíveis AR de um domínio.

A motivação para propor uma atividade de agrupamento de componentes num contexto de ED é que: a) existem componentes que são mutuamente dependentes, i.e., eles fornecem e requerem interfaces uns dos outros; b) existem componentes requerendo muitas interfaces de um número restrito de componentes; c) existem componentes trocando mensagens com outros componentes e todos eles estão associados a um mesmo contexto do domínio. Para identificar problemas de acoplamento de componentes, foi proposto um conjunto inicial de seis critérios de agrupamento de componentes. Através dos resultados obtidos pelos critérios, o Engenheiro de Domínio pode decidir quais componentes devem ser agrupados.

Os resultados esperados com o agrupamento de componentes em elementos arquiteturais são:

1) Uma arquitetura de componentes refinada, composta por elementos arquiteturais;
2) Um modelo de cada agrupamento de componentes que mostra como os componentes que fazem parte do agrupamento trocam mensagens uns com os outros;
3) Um elemento arquitetural que representa um conjunto de componentes que por sua vez possuem suas variabilidades e opcionalidades.

Os critérios propostos nesta tese, e que serão detalhados nas próximas subseções, são:

- Acoplamento por Contextos do Domínio (ACD)
- Acoplamento por Componentes de Processo (ACP)
- Acoplamento por Interfaces Providas (AIP)
- Acoplamento por Interfaces Requeridas (AIR)
- Acoplamento por Componentes Requeridos (ACRd)
- Acoplamento por Componentes requerendo outros componentes (ARnC)

Cada critério avalia os componentes, sob um ponto de vista diferente, gerando informações para que o engenheiro de domínio possa decidir se um conjunto destes componentes pode ser ou não agrupado. Resultados obtidos pelos critérios podem ser combinados, integrando assim, em um único elemento arquitetural, um ou mais agrupamentos de componentes sugeridos.

Embora critérios possam ser usados de forma combinada, as sugestões de agrupamento de um dado critério não eximem que componentes, que já fazem parte de uma sugestão, também sejam avaliados por outros critérios, fazendo parte de outras sugestões, segundo outros critérios.

Como os componentes avaliados possuem suas respectivas variabilidades, opcionalidades e regras de exclusão mútua, é necessário que tais propriedades sejam também consideradas para a geração de um elemento arquitetural. Por esta razão, após o Engenheiro de Domínio escolher alguma sugestão de agrupamento, deverão ser atribuídas ao elemento arquitetural gerado suas respectivas propriedades de variabilidade, opcionalidade e regra de exclusão mútua.

Por esta razão, tais propriedades serão tratadas da seguinte forma:

- se o agrupamento sugerido contiver, pelo menos, um componente mandatório, então o elemento arquitetural gerado deve ser considerado mandatório.

- não é permitido que um elemento arquitetural seja composto por uma variante sem o seu ponto de variação, incluindo variantes que representam regras de composição de mútua exclusão (i.e. <<XOR>>). Tal restrição se deve ao fato de que a variante depende de seu ponto de variação para prestar seus serviços. Como o objetivo de um elemento arquitetural é aumentar o acoplamento dos componentes, então não faz sentido que ponto de variação e suas variantes façam parte de elementos arquiteturais diferentes.

- se o agrupamento sugerido contiver, pelo menos, um componente ponto de variação, então o elemento arquitetural gerado deve ser considerando como um ponto de variação na arquitetura de referência. Este estereótipo relacionado ao elemento arquitetural deve significar que o mesmo representa um ponto de variação do domínio o qual pode ter, dentre outros componentes, pelo menos três que representam um ponto de variação e suas variantes.

Os critérios a seguir apresentados utilizaram como exemplo a arquitetura do domínio de telefonia móvel, apresentada na Figura 3.11.

3.5.1 - Acoplamento por Contextos do Domínio (ACD)

Este critério avalia com quantos e quais contextos de um domínio um componente está relacionado.

Durante a Análise de Domínio, características são organizadas em diferentes contextos, podendo pertencer a um ou mais contextos. Características são utilizadas para derivar tipos de negócio, casos de uso e componentes. De acordo com o critério ACD, se dois componentes forem gerados a partir de características que pertencem ao mesmo contexto, o agrupamento destes componentes é sugerido para o Engenheiro de Domínio. É importante salientar que durante um processo de engenharia de aplicação, o engenheiro deve identificar os contextos do domínio, visando a reutilização dos artefatos a eles relacionados para a construção de sua aplicação específica. Portanto, para serem melhor compreendidas e reusáveis, as arquiteturas de referência de um domínio deveriam agrupar, tanto quanto possível, componentes que estão relacionados a um mesmo contexto, e evitar agrupamentos de componentes que pertencem a contextos diferentes.

No contexto do domínio de telefonia móvel (Figura 3.11), existem os contextos básico e adicionais. Neste caso, o critério sugere os seguintes agrupamentos: 1) agrupar DispositivodeImagem, DispositivodeSom, AlgoritmodeBusca, Java, ToquesMusicaisManager, ToquesSimplesManager, ToquesPolifônicosManager, CâmeraManager, CampainhaManager, ConfiguraCampainha e JogosoSnakeCarRacer, que fazem parte do contexto adicionais, e 2) agrupar ChamadasTelefonicas, AgendaManager, GerenciaAgenda, CaixaPostalManager, TelefoneCelularManager, AlarmeManager, UsuárioManager e PrestadoradeServicoManager, que fazem parte do contexto denominado Básico. Nestes dois casos seriam gerados elementos arquiteturais mandatórios e pontos de variação. A Figura 3.43 mostra o resultado do agrupamento dos componentes da Figura 3.11. O elemento arquitetural Básico somente requer do Elemento arquitetural Adicionais a interface IAlgoritmoBusca. Já o elemento Arquitetural Adicionais requer as interfaces IAlarmeMgr e ITelefoneCelularMgr fornecida pelos componentes agrupados pelo elemento arquitetural Básico.

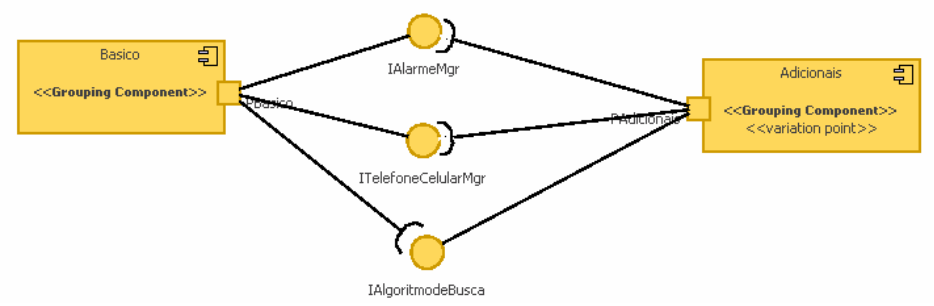


Figura 3.43: Arquitetura do Domínio de Telefonia Móvel com os Elementos Arquiteturais sugeridos

3.5.2 - Acoplamento por Componentes de Processo (ACP)

Este critério avalia quantos e quais componentes de negócio são requeridos por componentes de processo.

Componentes de processo requerem interfaces de componentes de negócio para executar seus serviços. Este critério sugere agrupar componentes de processo com os componentes de negócio dos quais eles requerem interfaces. Através deste agrupamento de componentes, a arquitetura de referência pode se tornar mais simples porque as características de negócio e processo estão encapsuladas num único artefato de domínio. Além disso, os elementos arquiteturais propostos por este critério estão em consonância com a proposta de Brown (2000), o qual mantém componentes de negócio e de processo numa mesma camada da arquitetura de componentes.

Para exemplificar este critério, considere o modelo de componentes do domínio de Telefonia Móvel apresentado na Figura 3.11. Existem 7 componentes de processo requerendo interfaces de seis componentes de negócio (e.g. componente de processo GerenciaAgenda requer componente de negócio AgendaManager; componentes de processo ProcuraLigacoes, LigacoesRecebidas e LigacoesFeitas requere o componente de negócio CaixaPostalManager). Através deste critério e considerando este conjunto de componentes, são obtidos os Elementos Arquiteturais da Figura 3.44 (a). A figura mostra (Figura 3.44 (b)) ainda a arquitetura interna do elemento arquitetural CaixaPostal.

3.5.3 - Acoplamento por Interfaces Providas (AIP)

O critério AIP avalia quantas e quais interfaces são providas por cada componente. Este critério fornece o grau de responsabilidade de cada componente em nível arquitetural. Se um componente fornece duas ou mais interfaces, o agrupamento com os componentes que as requerem é sugerido.

Considerando o exemplo da Figura 3.11, o componente DispositivodeImagem fornece IDispositivodeImagem2D e I DispositivodeImagem3D para o componente de processo CapturaImagem. Segundo este critério, estes componentes devem ser agrupados num único elemento arquitetural. Conforme mostra a Figura 3.45, o elemento arquitetural, denominado Imagem, deve requerer a interface ICameraMgr fornecida por outro elemento arquitetural de uma AR do domínio.

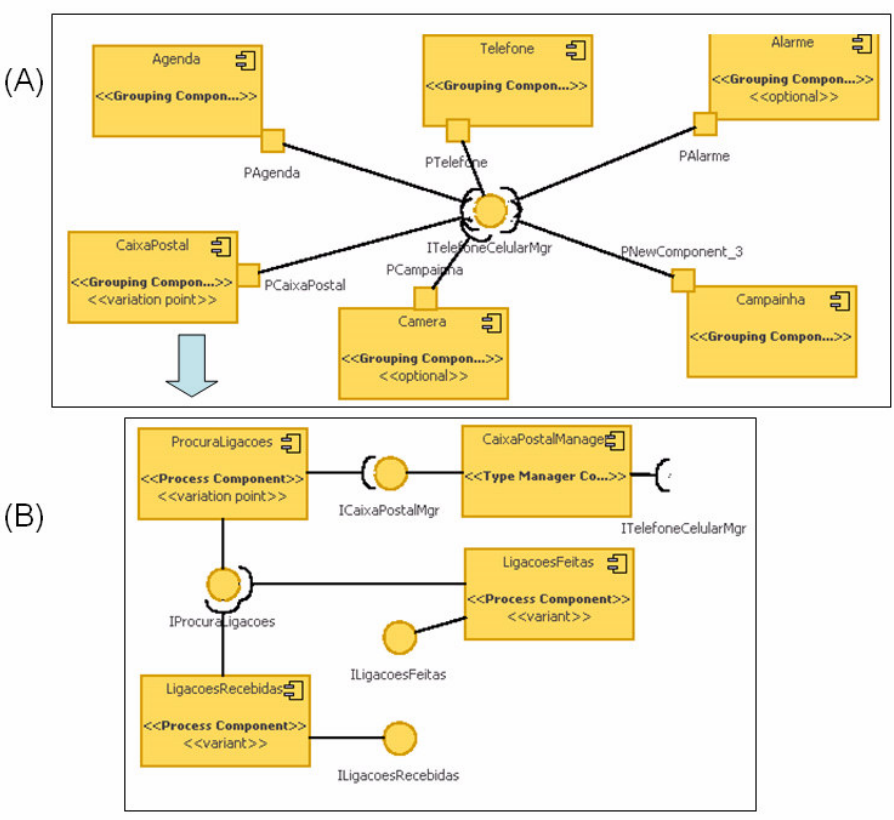


Figura 3.44: Elementos Arquiteturais obtidos pelo ACP e especificação do Elemento Caixa Postal

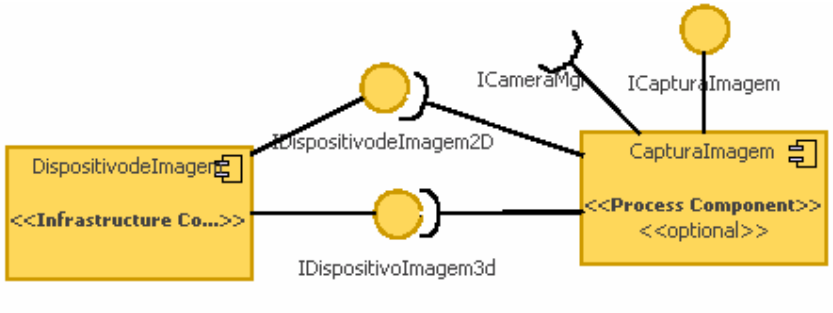


Figura 3.45: Especificação Interna do Elemento Arquitetural Imagem.

3.5.4 - Acoplamento por Interfaces Requeridas (AIR)

O critério AIR avalia quantas e quais interfaces são requeridas por cada componente.

Este critério obtém o grau de dependência de um componente através de suas interfaces. Se um componente requer duas ou mais interfaces, então, pode ser sugerido agrupá-lo com o conjunto de componentes que provêem estas interfaces.

Considerando o modelo de componentes da Figura 3.11, o critério sugere, por exemplo, os seguintes agrupamentos, com as respectivas variabilidades e opcionalidades:

1) Agrupar os componentes JogosSnakeCarRacer, Java e TelefoneCelularManager, gerando um elemento arquitetural mandatório e ponto de variação;
2) Agrupar os componentes CampainhaManager, AlarmeManager, TelefoneCelularManager, DispositivoSom, gerando um elemento arquitetural mandatório e invariante;
3) Agrupar os componentes TelefoneCelularManager, PrestadoradeServicoManager, UsuárioManager, gerando um elemento arquitetural mandatório e invariante;
4) Agrupar os componentes CâmeraManager, DispositivoImagem, TelefoneCelularManager, gerando um elemento arquitetural mandatório e invariante.

3.5.5 - Acoplamento por Componentes Requeridos (ACRd)

O critério ACRd avalia quantos e quais componentes são requeridos por um componente. Este critério obtém o grau de dependência de cada componente, sem considerar suas interfaces. Se um componente requer dois ou mais componentes, é sugerido o agrupamento dos mesmos.

Considerando o modelo de componentes da Figura 3.11, o critério sugere, por exemplo, os seguintes agrupamentos:

1) Agrupar os componentes TelefoneCelularManager, PrestadoradeServiçoManager e UsuarioManager, gerando um elemento arquitetural mandatório e invariante;

2) Agrupar os componentes CapturaImagem, DispositivoImagem e CameraManager, gerando um elemento arquitetural mandatório e invariante;
3) Agrupar os componentes AgendaManager e TelefoneCelularManager, gerando um elemento arquitetural mandatório e invariante.

Pode-se observar que no caso deste modelo de componentes, este critério sugere os mesmos agrupamentos do critério AIR. Isto porque a maioria dos componentes deste exemplo fornece, cada um, somente uma interface. Neste caso, os critérios AIR e ACRD podem fornecer resultados diferentes quando componentes fornecem, cada um, mais de uma interface.

3.5.6 - Acoplamento por Componentes requerendo outros Componentes (ARnC)

O critério ARnC obtém quantos e quais componentes requerem um dado componente, sem considerar suas interfaces. Se um componente é requerido por outros, então, o critério sugere o agrupamento destes componentes.

Considerando o domínio de Telefonia Móvel da Figura 3.11, o critério será, por exemplo, as seguintes sugestões de agrupamento, com suas respectivas variabilidades e opcionalidades:

1) Agrupar os componentes TelefoneCelularManager, CâmeraManager, AlarmeManager, JogosSnakeCarRacer, CampainhaManager, CaixaPostalManager e AgendaManager, gerando um elemento arquitetural mandatório e ponto de variação;
2) Agrupar os componentes PrestadoradeServiço e TelefoneCelularManager, gerando um elemento arquitetural mandatório e invariante;
3) Agrupar os componentes TelefoneCelularManager e UsuárioManager, gerando um elemento arquitetural mandatório e invariante; e
4) Agrupar os componentes AgendaManager e ConfiguraAgenda, gerando um elemento arquitetural mandatório e invariante.

3.5.7 - Considerações gerais sobre os critérios

Os critérios apresentados anteriormente sugerem o agrupamento de componentes baseado em quatro diferentes aspectos: contextos do domínio, os componentes de processo, interfaces dos componentes e os próprios componentes sem as suas interfaces. Os critérios AIP, AIR, ACRd e ARnC usam o mesmo artefato para propor o agrupamento de componentes, mas sob diferentes pontos-de-vista. Quando um componente fornece somente uma interface, os critérios AIP e ACR produzem resultados similares. Entretanto, quando os componentes fornecem mais de uma interface, tais critérios produzem resultados consideravelmente diferentes.

O Engenheiro deve escolher o critério apropriado baseado nas diferentes perspectivas de avaliação sobre as dependências dos componentes (e.g. interfaces requeridas, providas, contextos do domínio, processo e negócio).

Como mencionado no início da Seção 3.5, um ou mais critérios podem ser usados pelo engenheiro para executar uma única tarefa de agrupamento de componentes, resultando em um elemento arquitetural do domínio.

Considerando o exemplo de Telefonia Móvel, se observa que existem sugestões de agrupamento que se repetem em vários critérios. A título de exemplo considere as seguintes sugestões de agrupamento: a) os componentes AgendaManager e GerenciaAgenda; b) os componentes CâmeraManager e TelefoneCelularManager. O agrupamento A é sugerido pelos critérios ACD, ACP e ARnC. O agrupamento B é sugerido pelos critérios AIR e ARnC. Adotando estas sugestões de agrupamento, uma nova versão da arquitetura do domínio deve ser gerada. Esta nova versão terá, portanto, dois elementos arquiteturais, os quais agrupam os componentes citados acima, e outros elementos arquiteturais, um para cada componente do domínio (i.e. negócio, processo, utilitário e de infra-estrutura) criado durante o processo CBD-Arch-DE e não agrupados. Já com estes elementos arquiteturais, é sugerida uma arquitetura do domínio melhor organizada, afetando a compreensão desta arquitetura por parte do engenheiro de domínio. Além disso, estes componentes encapsulados num único elemento arquitetural podem ser reutilizados no contexto de uma Engenharia de Aplicação.

Nos exemplos de sugestões de agrupamento apresentados anteriormente, o componente TelefoneCelular aparece em grande parte delas. Tal fato pode levar o Engenheiro de Domínio a refletir sobre a possibilidade de não considerar tal componente para compor nenhum elemento arquitetural com outros componentes da

arquitetura original. Isto porque tal componente provavelmente representa um requisito importante do domínio, o qual presta e/ou requisita muitos serviços de outros componentes. Se o Engenheiro de domínio agrupá-lo num elemento arquitetural, tal elemento terá uma sobrecarga de mensagens elevada, desviando-se do objetivo maior da arquitetura que é prestar serviços bem definidos e autocontidos na AR de um domínio. Ao agrupar um componente como este num elemento arquitetural, tal elemento será sempre reutilizado por aplicações derivadas deste domínio, mesmo que possua internamente componentes que não são necessários para a aplicação instanciada. Exemplos de elementos arquiteturais potencialmente reutilizáveis foram apresentados nas Figuras 3.44, 3.45 e 3.46.

Independente das sugestões escolhidas, a percepção do Engenheiro do Domínio a respeito da arquitetura é soberana. Portanto, se alguma sugestão de agrupamento não atende por completo às necessidades do Engenheiro, a abordagem considera que este poderá decidir por retirar algum componente de uma determinada sugestão, ou ainda, combinar duas ou mais sugestões para a geração de um único elemento arquitetural.

3.6 – Abordagem de Transformação de Modelos – Odyssey-MDA

A abordagem Odyssey-MDA, detalhada em (MAIA, 2006), permite ao desenvolvedor realizar a transformação dos modelos PIM, em modelos PSM e, posteriormente, obter a implementação destes componentes. Embora a Odyssey-MDA tenha sido adotada nesta tese, num contexto de ED, a mesma pode dar apoio tanto à ED como à EA. Basicamente, a abordagem consiste na realização das seguintes etapas:

a) Definição de plataformas e transformações – O desenvolvedor define as plataformas desejadas (e.g. EJB) e as transformações necessárias (e.g. componente de negócio do modelo PIM em um *Entity Bean* do modelo EJB) para que os modelos PIM sejam transformados em modelos específicos (PSM) para essas plataformas.

b) Marcação do modelo interno dos componentes – Os elementos do modelo interno dos componentes (e.g. modelo de classes que especifica um componente) são preparados com um mecanismo de marcação para guiar a realização das transformações posteriores.

d) Escolha da plataforma e execução de transformações – O desenvolvedor escolhe uma plataforma previamente definida. As transformações são aplicadas sobre o modelo previamente marcado para a obtenção do PSM para a plataforma escolhida.

e) Geração da implementação – Uma representação em código fonte do modelo PSM pode ser gerada na linguagem de programação da plataforma.

A Figura 3.46 mostra um cenário típico de utilização da abordagem. O desenvolvedor elabora o seu modelo inicial em uma ferramenta CASE ou ambiente de reutilização de sua preferência.

Através de ambiente escolhido, o desenvolvedor deve realizar marcações sobre os elementos do seu modelo interno de componentes utilizando estereótipos e etiquetas. Após a realização dessas marcações, o desenvolvedor deve exportar o modelo de origem (PIM marcado) em formato XMI (XML Metadata Interchange) (OMG, 2002b) e importá-lo na máquina de transformações Odyssey-MDA, onde transformações podem ser executadas sobre o modelo de origem, resultando em um modelo de saída (PSM). O PSM gerado pode ser exportado também em formato XMI, para posterior importação na ferramenta CASE inicial, ou transferência para algum gerador de código.

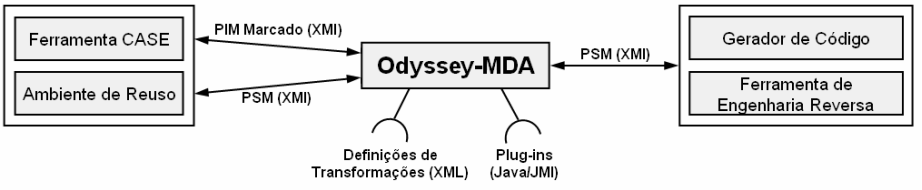


Figura 3.46: Cenário de Utilização da Odyssey-MDA

A Figura 3.47 mostra um exemplo de algumas transformações efetuadas pela Odyssey-MDA de um modelo PIM para um PSM em tecnologia EJB.

A abordagem Odyssey-MDA permite, ainda, a definição e execução de transformações bidirecionais. Neste caso, o desenvolvedor pode gerar um PSM a partir de um PIM (transformação direta), efetuar refinamentos e alterações no modelo PSM e propagar essas alterações de volta ao modelo PIM original (transformação reversa). Essa propagação é feita de forma extensível, onde elementos já existentes no modelo de destino são atualizados pela transformação e não substituídos. Um cenário de transformação reversa é a extração das informações independentes de plataforma de um modelo PSM, ou código fonte pré-existente, para a criação de um PIM. Isso é bastante

útil em casos em que um componente originalmente modelado para uma tecnologia específica tenha que ser migrado para outra tecnologia.

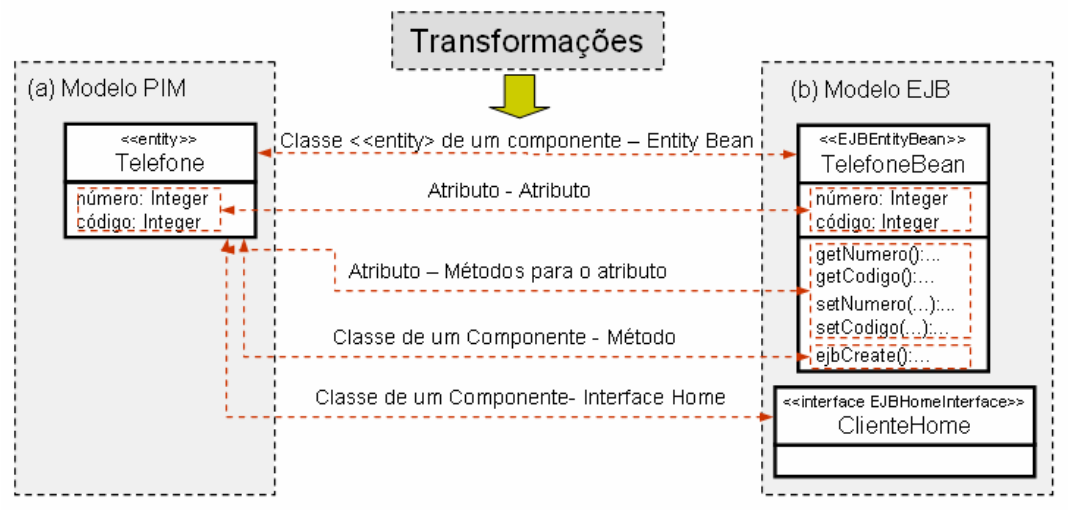


Figura 3.47: Exemplos de Transformações para a tecnologia EJB

Maiores detalhes sobre esta abordagem podem ser obtidos em (MAIA, 2006).

3.7 - Considerações Finais

Neste Capítulo foi apresentada a abordagem DECOM para apoio a um processo de ED com suporte ao DBC. Como pode ser observado nas seções anteriores, o CBS-Arch-DE dá apoio efetivo a todas as fases de um processo de ED. Uma contribuição importante do processo CBD-Arch-DE, quando comparado aos demais processo de ED e DBC, é o apoio ao mapeamento dos requisitos do domínio até os componentes lógicos e implementacionais, considerando suas relações e, sobretudo, as variabilidades e opcionalidades dos artefatos envolvidos.

Foram apresentadas as heurísticas para o mapeamento de características conceituais e funcionais para tipos de negócio e casos de uso, respectivamente. Posteriormente, as heurísticas para o mapeamento de características tecnológicas, tipos de negócio e casos de uso para, respectivamente, componentes utilitários e de infra-estrutura, de processo e de negócio. Através do exemplo explorado no Capítulo, foi possível observar como a notação e as heurísticas para a modelagem de variabilidades, opcionalidades e relacionamentos podem apoiar na manutenção destas propriedades em todos os artefatos do domínio, sendo este um dado relevante para as decisões sobre a geração de elementos arquiteturais e, sobretudo, na AR do domínio.

A DECOM tem a preocupação de efetivamente dar apoio a todas atividades da ED, inclusive na fase de implementação na qual nenhum dos processos de ED e DBC, até o momento, fornecem qualquer tipo de apoio. Além disso, o CBD-Arch-DE adota uma proposta baseada no *framework* MDA, que permite manter os componentes desenvolvidos na etapa de projeto desvinculados de qualquer característica implementacional de uma tecnologia de componentes.

Conforme apresentado na Tabela 3.9, a DECOM contempla todos os critérios para avaliação de métodos de ED e DBC.

No próximo Capítulo é apresentada a infra-estrutura desenvolvida para apoio a esta proposta de Projeto Arquitetural baseado em Componentes, no contexto do Ambiente Odyssey (ODYSSEY, 2005). A existência desta infra-estrutura também é considerada como um diferencial desta proposta em relação aos demais métodos de ED e DBC.

Tabela 3.9: Comparativo do apoio oferecido pelos métodos de ED, DBC e a Abordagem DECOM

Critérios de Avaliação		Abordagens de ED		Abordagens de DBC		Abordagens de ED com suporte ao DBC		
		FODA (1990)	FORM (2002)	Catalysis (1999)	UML Components (2000)	Odyssey-DE (2000)	Kobra (2000)	**DECOM (2006)**
1	Modelagem de variabilidades e opcionalidades em diferentes níveis de abstração	Parcial	Parcial	Não	Não	Parcial	Parcial	Sim
2	Manutenção das ligações e consistência entre os diferentes artefatos do domínio	Não	Não	Não	Não	Parcial	Parcial	Sim
3	Organização dos artefatos para a composição de elementos arquiteturais	Não	Não	Parcial	Parcial	Não	Sim	Sim
4	Uso de princípios de DBC na modelagem do domínio	Não	Parcial	Parcial	Parcial	Parcial	Sim	Sim
5	Necessidade de processos de ED com apoio ao DBC	Sim	Sim	Não	Não	Sim	Sim	Sim
6	Geração de código na linguagem de programação empregada por uma tecnologia de DBC	Não	Não	Não	Não	Parcial	Não	Sim
7	Modelagem e Reutilização de Artefatos e Modelos através de um ferramental apropriado	Não	Parcial	Não	Parcial	Parcial	Não	Sim

Capítulo 4 - Implementação da DECOM: um protótipo

4.1 - Introdução

No capítulo anterior, foi apresentada a Abordagem DECOM. Para viabilizar essa proposta, foi desenvolvida uma implementação que dá apoio a todas as atividades previstas na abordagem.

Esta implementação integra-se à infra-estrutura do ambiente Odyssey (ODYSSEY, 2005), aproveitando e adaptando os recursos já existentes de apoio à ED.

Este capítulo está dividido em mais quatro seções. Na Seção 4.2, apresentamos o ambiente Odyssey e o seu papel no desenvolvimento de software. Na Seção 4.3, são descritas as implementações, incluindo adaptações e novas funcionalidades.

As considerações finais do capítulo são apresentadas na Seção 4.4.

4.2 – O Ambiente Odyssey

O Odyssey é um ambiente baseado em modelos de domínio que permite a geração de artefatos *para* a reutilização (ED), bem como a criação de novas aplicações (EA), *com* a reutilização de artefatos através da *Engenharia de Aplicação*.

O processo de implementação de sua infra-estrutura ocorre desde 1997 (WERNER *et al.*, 1999). Desde então, uma série de ferramentas para apoio a ED e/ou EA tem sido construídas, visando tornar o Odyssey um ambiente de reutilização mais completo. No entanto, com o passar do tempo, o ambiente tornou-se muito grande, gerando problemas de escalabilidade e desempenho para seus usuários. Neste sentido, foi feita uma análise da infra-estrutura como um todo no sentido de identificar as funcionalidades básicas de um ambiente de reutilização e as funcionalidades secundárias, as quais só seriam utilizadas de acordo com as necessidades do usuário do ambiente.

Assim, desde 2003 foi disponibilizada uma versão reestruturada do ambiente, a qual foi denominada Odyssey Light (MURTA *et al.*, 2004b). Conforme apresenta a Figura 4.1, o Odyssey Light possui um conjunto de funcionalidades básicas, disponíveis através do *kernel* do ambiente. Exemplos de ferramentas que fazem parte do *kernel* são: editores de diagramas, gerência de processo (MURTA *et al.*, 2002) e a geração de componentes (TEIXEIRA, 2003). As demais funcionalidades são disponibilizadas sob a

forma de *plug-ins*. Dentre os *plug-ins* disponíveis no Odyssey Light, podemos citar: ferramentas de especificação e instanciação de arquiteturas específicas de domínios (XAVIER, 2001); verificação de consistência de modelos UML (DANTAS *et al.*, 2001), apoio à engenharia reversa (VERONESE *et al.*, 2002), suporte a padrões no projeto de software (DANTAS *et al.*, 2002), versionamento de modelos (OLIVEIRA *et al.*, 2004) e edição concorrente de modelos (LOPES *et al.*, 2004).

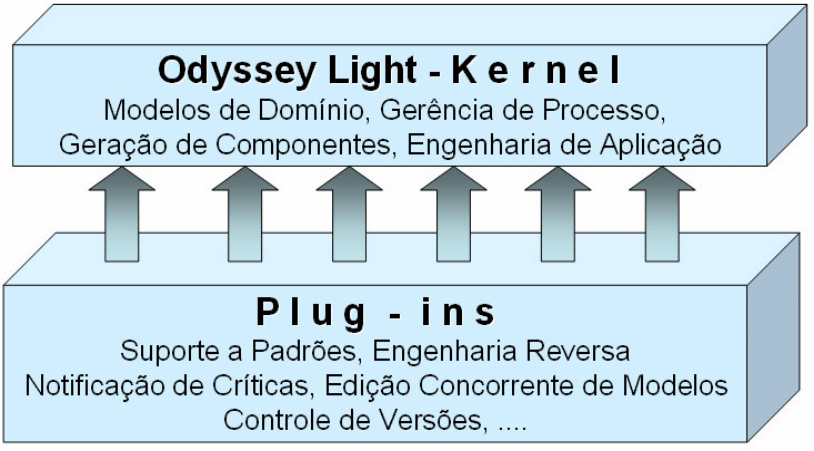


Figura 4.1: A Infra-Estrutura Odyssey

O Odyssey foi implementado através da linguagem JAVA (SUN, 2005b). Durante este capítulo, ao fazermos referência ao ambiente Odyssey, estaremos sempre considerando a versão Odyssey Light.

4.3 - Implementação da Abordagem DECOM

Anteriormente a esta tese, o Odyssey tinha como objetivo apoiar os processos *Odyssey-DE* (BRAGA, 2000) e *Odyssey-AE* (MILER, 2000). No entanto, no que tange ao Odyssey-DE, não existe apoio completo às atividades que compreendem ao projeto arquitetural de domínio baseado em componentes.

Neste sentido, foi feita uma análise das ferramentas existentes para avaliar se as mesmas apoiavam as atividades previstas num novo processo (i.e. CBD-Arch-DE). Como resultado, se observou que algumas funcionalidades do *kernel* ou disponíveis via *plug-ins* deveriam ser estendidas para as necessidades específicas do processo proposto. Além disso, outras funcionalidades e *plug-ins*, até o momento não previstos na infra-estrutura, teriam que ser construídos. Neste sentido, as subseções que se seguem descrevem estas extensões e novos *plug-ins*.

4.3.1 - Estrutura Semântica do Ambiente Odyssey

A Figura 4.2 mostra um diagrama de classes que representa parte da estrutura interna do Odyssey. Foram contempladas neste diagrama o conjunto de classes que apresenta maior relevância para este trabalho, e que fazem parte do *kernel* do ambiente.

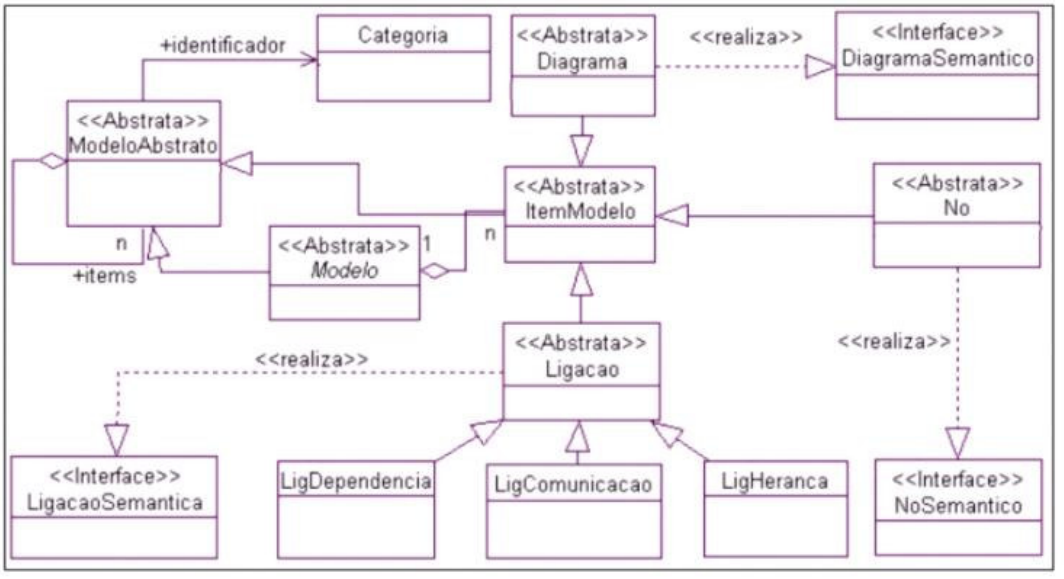


Figura 4.2 - Representação interna do ambiente Odyssey - *kernel*

A representação interna do ambiente Odyssey é organizada hierarquicamente através de uma árvore semântica de objetos. Todo objeto da árvore semântica é chamado de *ModeloAbstrato*, e representa um elemento de modelagem. Neste sentido, a árvore semântica é composta de um ou mais *ItemModelo*, os quais podem ser artefatos (e.g. características, tipos de negócio, contextos, casos de uso, pacotes, componentes, classes, interfaces) e modelos – *Diagrama* (e.g. modelos de características, de tipos de negócio, de componentes). A partir de um destes elementos, é possível percorrer a árvore hierarquicamente através das relações entre os objetos, a fim de obter as relações entre os artefatos do domínio, que por sua vez interfere no recorte de uma aplicação. Durante a EA, se o Engenheiro selecionar uma característica do domínio a ser reutilizada, todos os seus artefatos relacionados, através da árvore semântica, serão também reutilizados.

A árvore semântica é organizada através de categorias de modelos, representados, na figura pelas classes *Modelo* e *Categoria*. São definidas no Odyssey as categorias Contextos, Características, Casos de Uso, Tipos de Negócio, Classes, Interfaces e Componentes. Cada modelo é composto por diversos itens de modelagem, instâncias da classe *ItemModelo*, que representam pacotes, diagramas, nós e ligações

específicas da categoria do modelo. A classe abstrata *No* é uma superclasse para elementos como classes, características, componentes, etc, enquanto a classe *Ligacao* representa as ligações entre os diversos nós, tais como Herança e Associação.

4.3.2 - Implementações no Kernel do Ambiente Odyssey

No contexto desta tese, foram feitas as seguintes implementações no *kernel* do Odyssey: a) a modelagem do processo CBD-Arch-DE para a sua instanciação durante a engenharia de um domínio; b) a extensão das notações dos modelos do domínio para modelagem de variabilidades, opcionalidades e regras de composição e; c) as heurísticas de mapeamento de artefatos do domínio. Estas implementações serão apresentadas, respectivamente, nas Seções 4.3.2.1, 4.3.2.2, 4.3.2.4 e 4.3.2.5.

4.3.2.1 - Processo CBD-Arch-DE

A modelagem do processo CBD-Arch-DE foi desenvolvida a partir da máquina de processo Charon. A Charon permite a modelagem, simulação, execução e acompanhamento de processos de software, e também pertence ao *kernel* do ambiente Odyssey (MURTA *et al.*, 2002). A máquina de processo Charon prevê que as atividades sejam modeladas através de processos primitivos e compostos. Os processos compostos agrupam processos primitivos e estes últimos detalham a atividade que vão desenvolver, o tempo gasto, os artefatos consumidos, os papéis das pessoas que farão parte da atividade e os artefatos produzidos. Cada processo primitivo deve ser devidamente descrito para guiar as pessoas que o executarão, visando o seu desenvolvimento correto.

Embora os processos Odyssey-DE e Odyssey-AE possam ser utilizados a partir da ambiente Odyssey, não existe efetivamente uma modelagem prévia destes processos, através da Charon, para que um Engenheiro de Domínio ou Aplicação possa utilizá-los. Para o uso dos recursos da Charon, estes processos devem ser modelados a cada novo domínio ou aplicação, gerado no ambiente Odyssey. A falta da modelagem destes processos dentro de uma infra-estrutura restringe a utilização dos mesmos para as atividades de ED e EA apoiadas pelo Odyssey. Neste sentido, foi modelado o processo CBD-Arch-DE no Odyssey.

Conforme mostra a Figura 4.3, as classes com o fundo escuro fazem parte do pacote *generator*. Através da classe *Facade,* o processo é disponibilizado no ambiente Odyssey (classe Ambiente), mais especificamente durante a criação de um usuário de administração do domínio (classe *GerenteLogin*). A classe *Facade* serve como interface

entre o Odyssey e a classe *GeneratorControl*. O Odyssey só tem conhecimento dessa classe, que se encarrega de chamar o *GeneratorControl* para criar todas as atividades que o compõem: *CreationProcessGenerator, AnalysisProcessGenerator, DesignProcessGenerator e CBDProcessGenerator*.

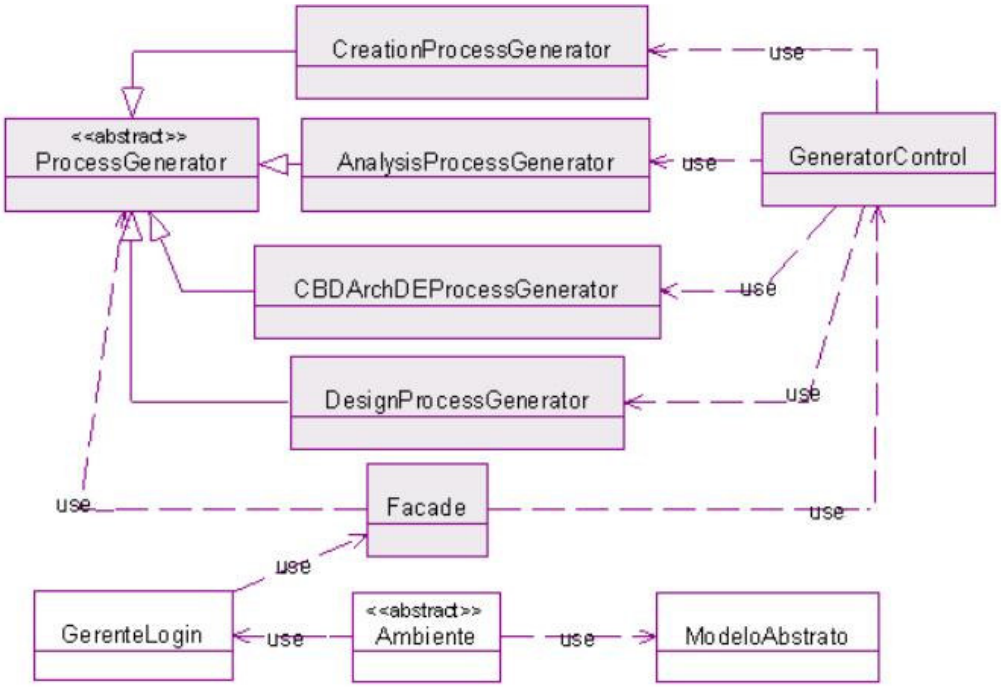


Figura 4.3: Modelo de Classes simplificado do Processo CBD-Arch-DE

Com esta implementação, sempre que o ambiente é iniciado e um novo domínio é criado, o processo CBD-Arch-DE pode ser instanciado. A partir desta instanciação, a máquina Charon controla a execução e o acompanhamento do processo.

4.3.2.2 - Notação para Modelagem de Variabilidades, Opcionalidades e Regras de Composição em Modelos de Características

A outra implementação desenvolvida no *kernel* do Odyssey refere-se as notações para modelagem de variabilidades, opcionalidades e regras de composição. Neste contexto, o maior volume de implementação ocorreu no contexto do trabalho de Oliveira (2006) para a representação da Notação Odyssey-FEX, ou seja, para o modelo de características. Nesta seção são apresentados alguns detalhes de implementação importantes para o contexto desta Tese, referente à notação Odyssey-FEX. Maiores detalhes podem ser obtidos em (OLIVEIRA, 2006). As demais implementações referentes à modelagem de variabilidades, opcionalidades, relacionamentos e regras de

composição de outros artefatos do domínio, desenvolvidos nesta Tese, serão apresentados no final da Seção.

A Figura 4.4 destaca as extensões necessárias no ambiente Odyssey para a implementação da notação Odyssey-FEX. A classe *NoCaracterística* representa uma característica genérica. Esta classe é a superclasse para as outras características existentes e subclasse da classe abstrata *No.* A representação de característica mandatória ou opcional, externa, não definida e organizacional, é feita através de atributos da classe *NoCaracterística*.

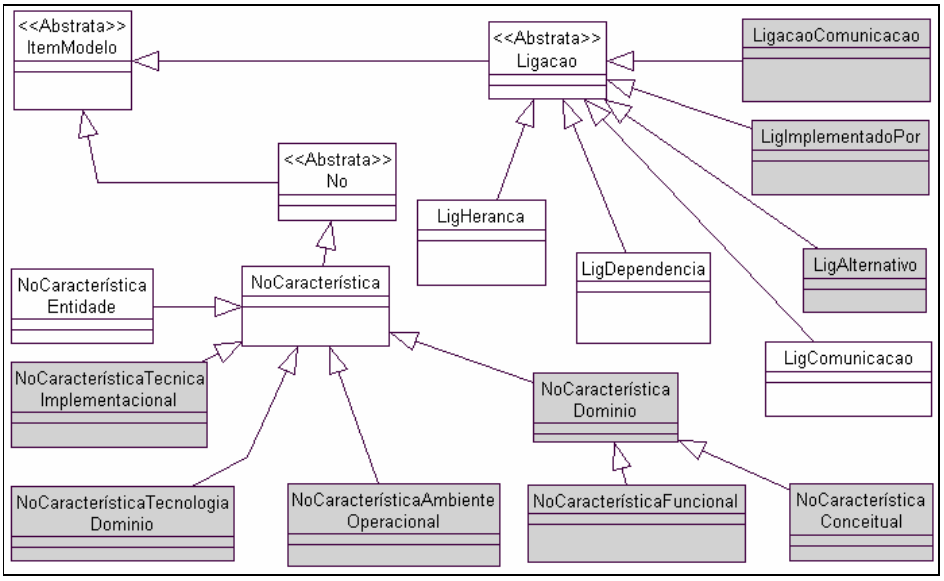


Figura 4.4: Extensão da estrutura interna do Odyssey de acordo com a notação Odyssey-FEX

Outra classe estendida na estrutura interna do Odyssey é a *Ligacao*, que agora representa uma generalização de classes que pode ser especializada em diferentes tipos de ligações (e.g. Alternativo, ImplementadoPor e LigaçãoDeComunicação) de um *ItemModelo*.

A Figura 4.5 mostra a atual janela de especificação de uma característica. Na aba "Feature", é possível visualizar os campos necessários à definição da característica, baseada na notação Odyssey-FEX.

As outras classificações descritas na notação Odyssey-FEX podem ser configuradas de acordo com a necessidade do usuário.

Também é possível estabelecer ligações da característica com os demais artefatos do domínio (e.g. tipos de negócio, casos de uso, componentes, classes). Estas

ligações são importantes para o Engenheiro de Aplicação durante a instanciação de artefatos de um domínio.

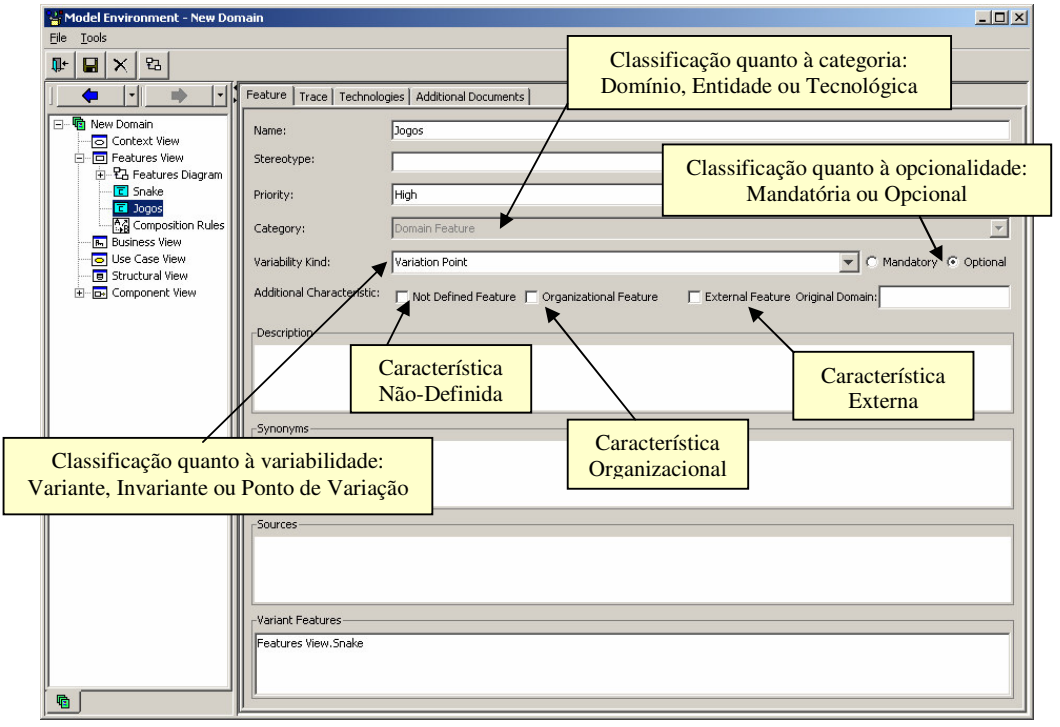


Figura 4.5: Padrão de domínio da características adaptado para a notação Odyssey-FEX

As regras de composição da notação Odyssey-FEX foram acrescentadas à estrutura semântica do Odyssey, através do pacote *RegrasComposicao*.

A Figura 4.6 apresenta a janela de criação de uma regra de composição, criada a partir da *Features View* através da qual é também possível ter acesso a todas as características e modelos de características do domínio. As regras de composição inclusivas e exclusivas são criadas, separadamente, através de botões na barra de ferramentas. Cada regra instanciada se torna parte da árvore semântica do Odyssey, e recebe um nome – **R** para regras inclusivas e **X** para regras exclusivas – incrementado seqüencialmente.

Estas regras de composição podem, também, ser visualizadas através do modelo de características e consideradas na construção dos demais artefatos do domínio e durante o processo de Engenharia de Aplicação.

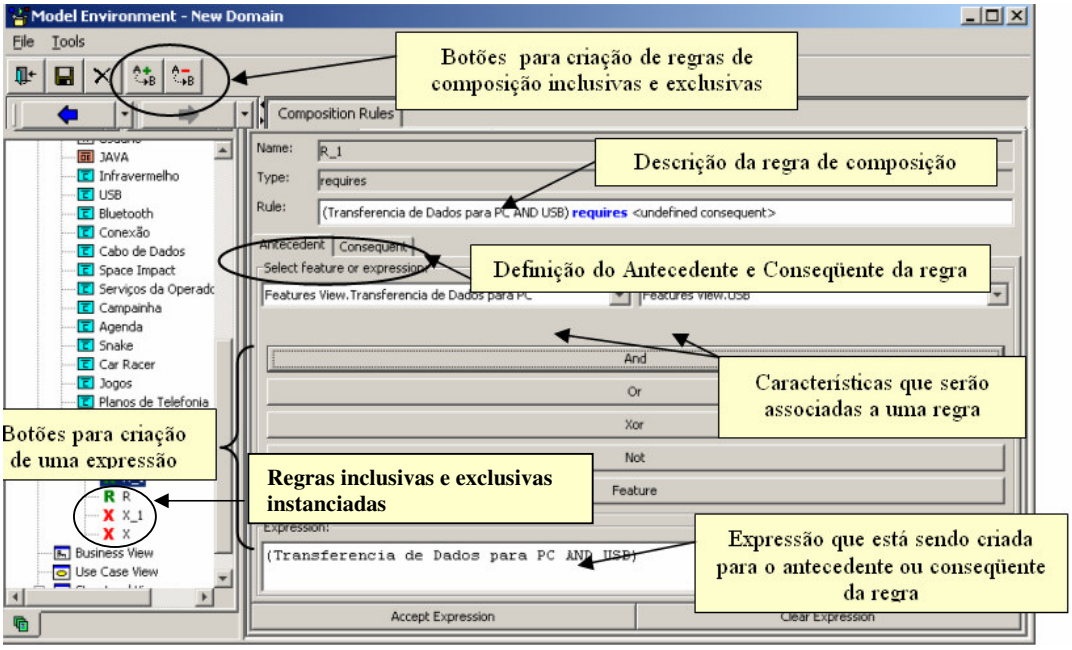


Figura 4.6: Definição dos elementos de uma regra de composição – Antecedente e Conseqüente

Para a representação da estrutura léxica das características, a notação Odyssey-FEX utiliza a estrutura léxica do Odyssey, a qual é formada por todos os diagramas de cada modelo. A Figura 4.7 mostra a janela do diagramador de características gerado a partir da notação Odyssey-FEX.

Na parte superior da janela, existe uma barra de ferramentas que auxilia o usuário nas tarefas de modelagem. É através dela que elementos são inseridos no diagrama.

A árvore de itens semânticos não contém os elementos do tipo relacionamento. Neste sentido, tais itens semânticos só poderão ser acessados através do diagrama de características. Por exemplo, somente através do diagramador, é possível associar ou desassociar um ponto de variação de suas variantes, bem como remover relacionamentos.

Com a utilização da notação Odyssey-FEX, foram necessárias algumas adaptações no processo de reutilização dos artefatos do domínio realizado. Com estas adaptações, o Odyssey passa a validar a seleção de características do domínio, interpretando as regras de composição, detectando eventuais quebras de regras e relacionamentos, bem como inconsistências como a não-seleção de uma característica mandatória.

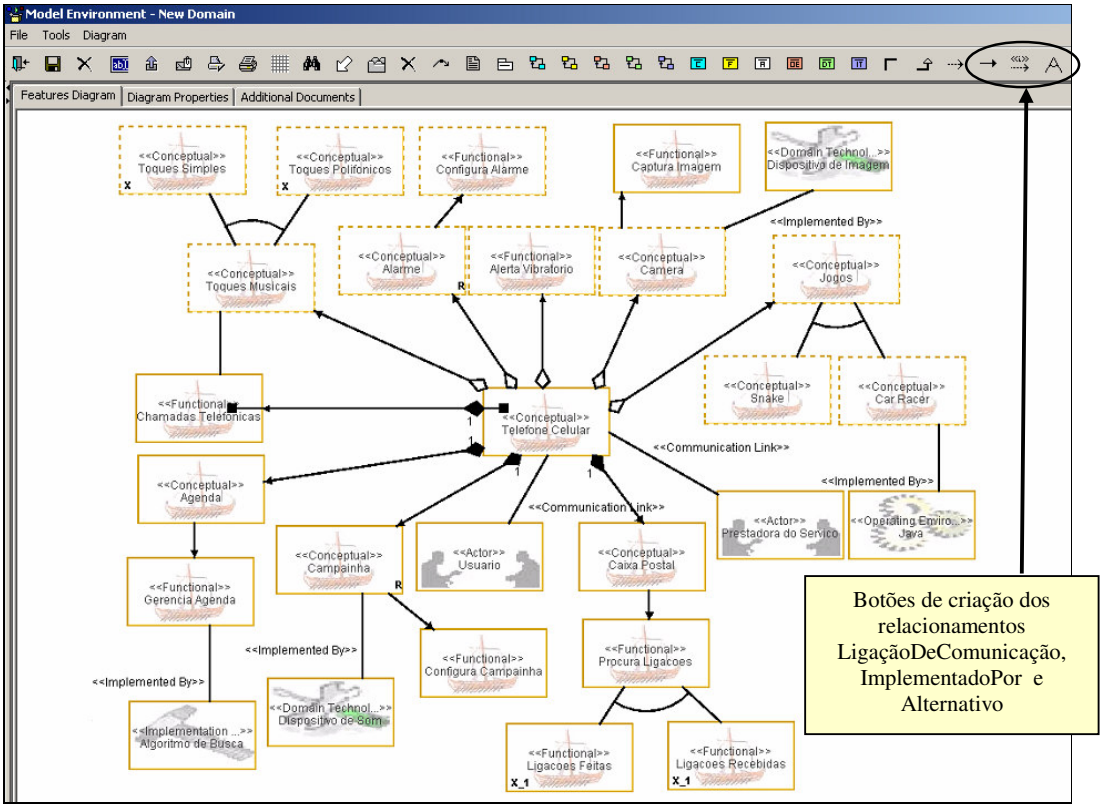


Figura 4.7: Janela de diagramação do ambiente Odyssey

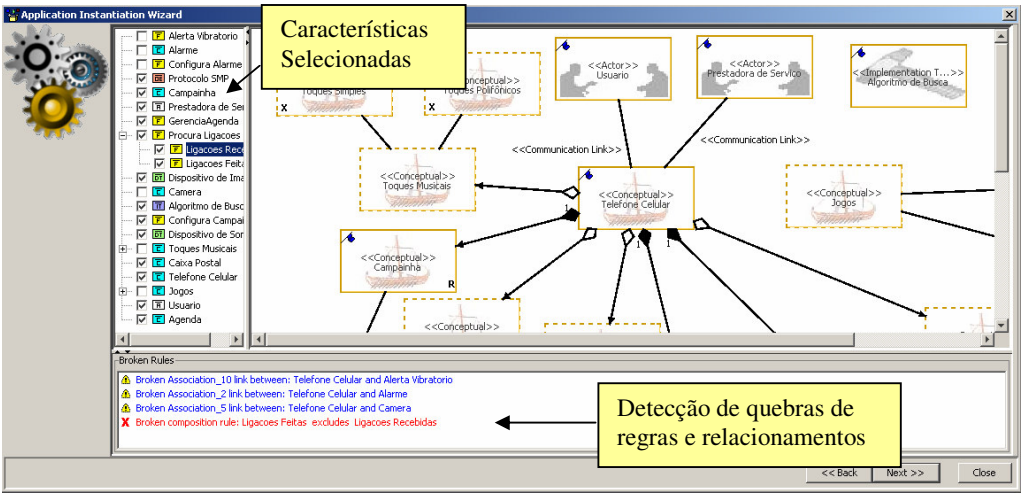


Figura 4.8: Recorte das Características do domínio de Telefonia Móvel na Engenharia de Aplicação

A Figura 4.8 mostra um exemplo de recorte para a Engenharia de uma Aplicação, baseada nas características do domínio de telefonia móvel. Nesta figura é possível observar, à esquerda, uma árvore de características que podem ser selecionadas. A seleção de uma característica da árvore implica inclusão desta - e conseqüentemente, dos artefatos a ela relacionados - na nova aplicação ou produto a ser

desenvolvido. Por convenção, as características que são mandatórias no domínio já se encontram selecionadas. As características não selecionadas na árvore representam as características opcionais no domínio, cabendo ao desenvolvedor selecionar dentre estas as características desejadas. As características que representam um ponto de variação são apresentadas em um nível hierárquico (na figura, característica *Procura Ligacoes*), tendo suas variantes como elementos subordinados a estas na árvore (características *Ligações Recebidas* e *Ligações Feitas*). Assim como nas outras características, pontos de variação ou variantes que sejam mandatórios já se encontram selecionados. As características selecionadas na árvore aparecem, na parte direita da figura, com uma bandeira no canto superior esquerdo.

4.3.2.3 – Extensões nas Notações de Tipos de Negócio, Casos de Uso e Componentes para representação de variabilidades, opcionalidades e regras de composição.

No modelo de casos de uso, foram feitas as seguintes extensões: a) inclusão dos atributos ehMandatório, TipoVariabildade e xor para a especificação de opcionalidades, variabilidades, regras de mútua exclusão, respectivamente; e b) foram criados dois outros tipos de relacionamento de dependência com estereótipos <<agregation>> e <<association>>, para representar relações de agregação e associação, respectivamente. A especificação de pré-condição, esperada para o mapeamento de uma regra de inclusão proveniente do modelo de características, já estava implementada no modelo de casos de uso no ambiente Odyssey.

Para os modelos de tipos de negócio e casos de uso, só foram necessários criar atributos para a especificação de opcionalidade, variabilidade e regra de mútua exclusão. Para os demais mapeamentos necessários, foram utilizados os recursos já existentes em cada modelo.

4.3.2.4 - Heurísticas de Mapeamento de Artefatos da Análise de Domínio

A outra implementação, desenvolvida nesta Tese no *kernel* do Odyssey, está associada às heurísticas para o mapeamento de variabilidades entre os artefatos do domínio. Nesta seção são apresentados como foram implementados os mapeamentos para tipos de negócio e casos de uso que estão descritos nas Seções 3.4.1 e 3.4.2.

Todos os mapeamentos foram implementados sob a forma de assistentes, os quais são compostos de diferentes passos.

A Figura 4.9 mostra um modelo de classes simplificado desta implementação. O *AssistenteMapeamento* é responsável por manipular os painéis de interface com o usuário, neste caso, *PainelIntrodução* e *PainelMapeamento*. A classe *PainelIntrodução* apresenta uma descrição das heurísticas que podem ser aplicadas, enquanto que a classe *PainelMapeamento* permite a seleção das heurísticas desejadas. As heurísticas são exibidas numa árvore de três níveis: uma raiz seguida das características e das heurísticas que podem ser aplicadas a cada característica (lado esquerdo da Figura 4.10). Os nós da árvore são representados através da classe *NoArvore*.

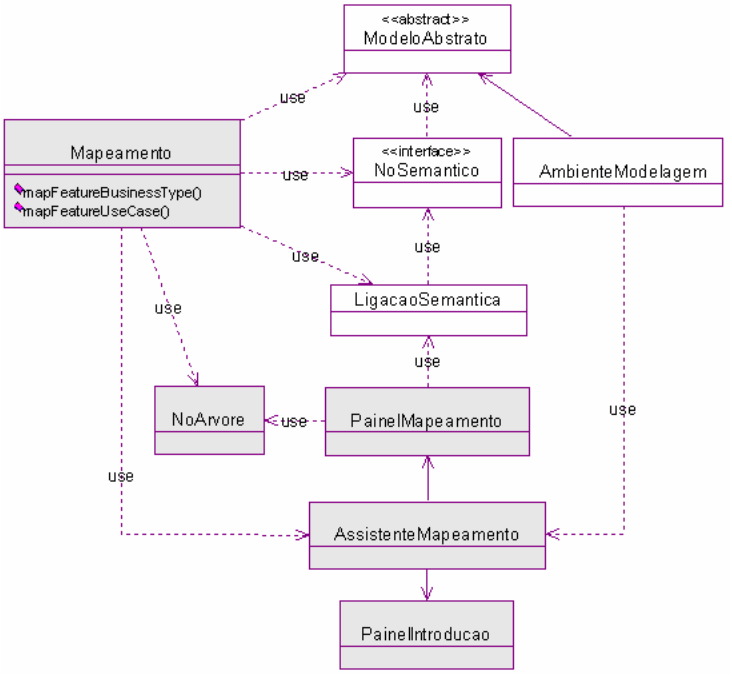


Figura 4.9: Modelo de Classes Simplificado do Pacote de Mapeamento de Artefatos de Análise de Domínio

Na classe *Mapeamento*, são executadas as heurísticas selecionadas anteriormente no *PainelMapeamento*. Ao executar as heurísticas, podem ser gerados relacionamentos no modelo alvo, utilizando, neste caso, a classe *LigacaoSemantica* para representar os mesmos, e a classe *NoSemantico* para tratar os elementos que fazem parte de cada relacionamento. Nesta implementação, a classe *ModeloAbstrato* é utilizada para manipular o modelo fonte e o modelo alvo do mapeamento, onde são lidos e incluídos elementos, respectivamente, e a classe *AmbienteModelagem* representa o ambiente onde o mapeamento será efetuado.

Para apresentarmos a geração dos tipos de negócio e casos de uso a partir das heurísticas propostas, é utilizado o diagrama de características do domínio de Telefonia Móvel, apresentado na Figura 4.7.

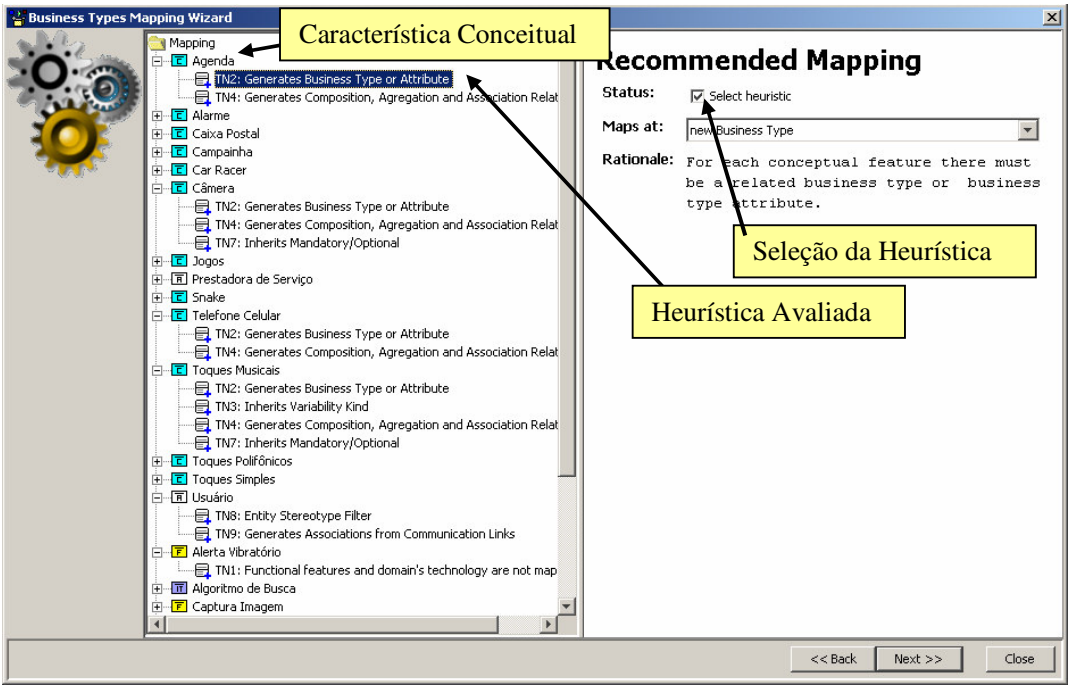


Figura 4.10: Janela de Mapeamento de Características em Tipo de Negócio

4.3.2.4.1 - Mapeamento de Características Conceituais e de Entidade para Tipos de Negócio

Para a utilização das heurísticas, o Engenheiro de Domínio pode pressionar o botão esquerdo do mouse sobre o ícone  na barra de ferramentas, ou selecionado a opção *Map Business Type from Features*, conforme mostra a Figura 4.11.

Ao selecionar uma das opções, é apresentada o assistente de mapeamento ao Engenheiro do Domínio que, dentre outras, apresenta a tela de seleção das heurísticas de mapeamento, conforme mostra a Figura 4.10. Do lado esquerdo, a árvore com as características conceituais e de entidade do domínio. Para cada característica, é listado o conjunto de heurísticas que pode ser utilizado pelo Engenheiro de Domínio. Ao selecionarmos uma determinada heurística, seus detalhes são apresentados do lado direito da janela. Como as heurísticas são apresentadas como sugestões para o Engenheiro de Domínio, utilizamos *checkboxes*, para que ele possa selecioná-las conforme suas necessidades.

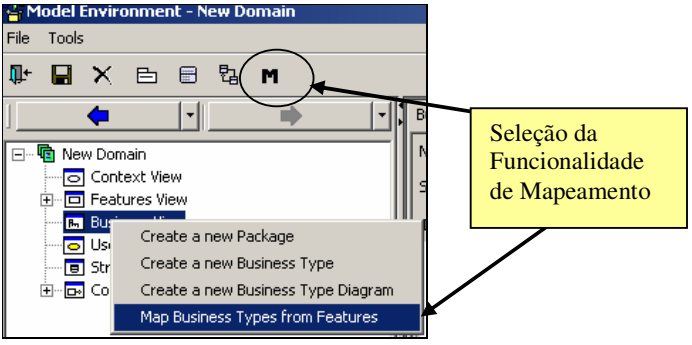


Figura 4.11: Janela de Acesso às Heurísticas de Mapeamento de Características para Tipos de Negócio

Após a análise de todas as heurísticas aplicadas a cada característica, é indicada, numa próxima janela do assistente, a geração de um pacote com os tipos de negócio obtidos pelo mapeamento, bem como o seu modelo de tipos de negócio com os relacionamentos. A Figura 4.12, mostra a árvore semântica dos tipos de negócio gerados, bem como o modelo de tipos de negócio resultante.

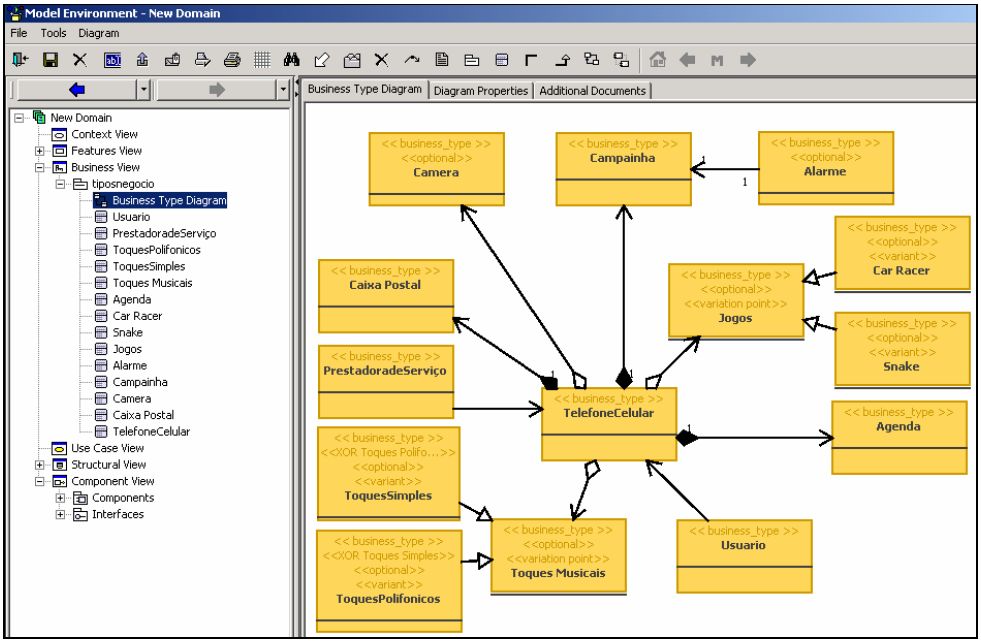


Figura 4.12: Diagrama de Tipos de Negócio gerado

4.3.2.4.2 - Mapeamento de Características Funcionais para Casos de Uso

Semelhante ao apresentado na Seção 4.3.2.4.1, para a utilização das heurísticas de mapeamento de características funcionais em casos de uso, o Engenheiro de Domínio deve pressionar o botão esquerdo do mouse sobre o ícone na barra de ferramentas, ou pela árvore semântica, na *use case view,* escolhendo a opção *Map Use Case and Actors from Features*, conforme mostra a Figura 4.13.

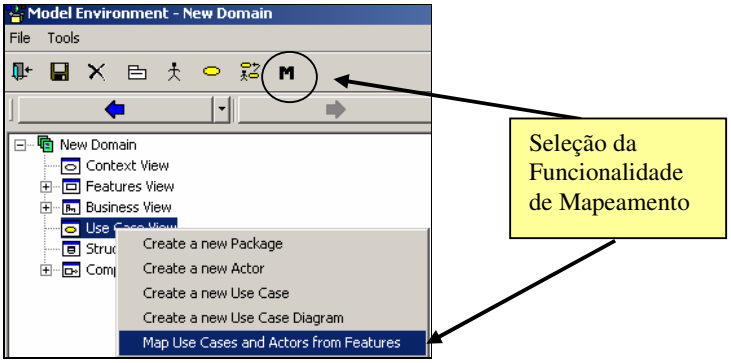


Figura 4.13: Janela de Acesso às Heurísticas de Mapeamento de Características para Casos de Uso

O processo de seleção das heurísticas para a obtenção de casos de uso é semelhante ao utilizado para a obtenção de tipos de negócio. O Engenheiro do Domínio utiliza um assistente que apresenta uma janela de seleção das heurísticas, conforme mostra a Figura 4.14.

Como resultado desta atividade, é gerado um pacote com casos de uso do domínio na árvore semântica do Odyssey, bem como o seu modelo de casos de uso com os relacionamentos, também, obtidos pelo mapeamento, conforme mostra a Figura 4.15.

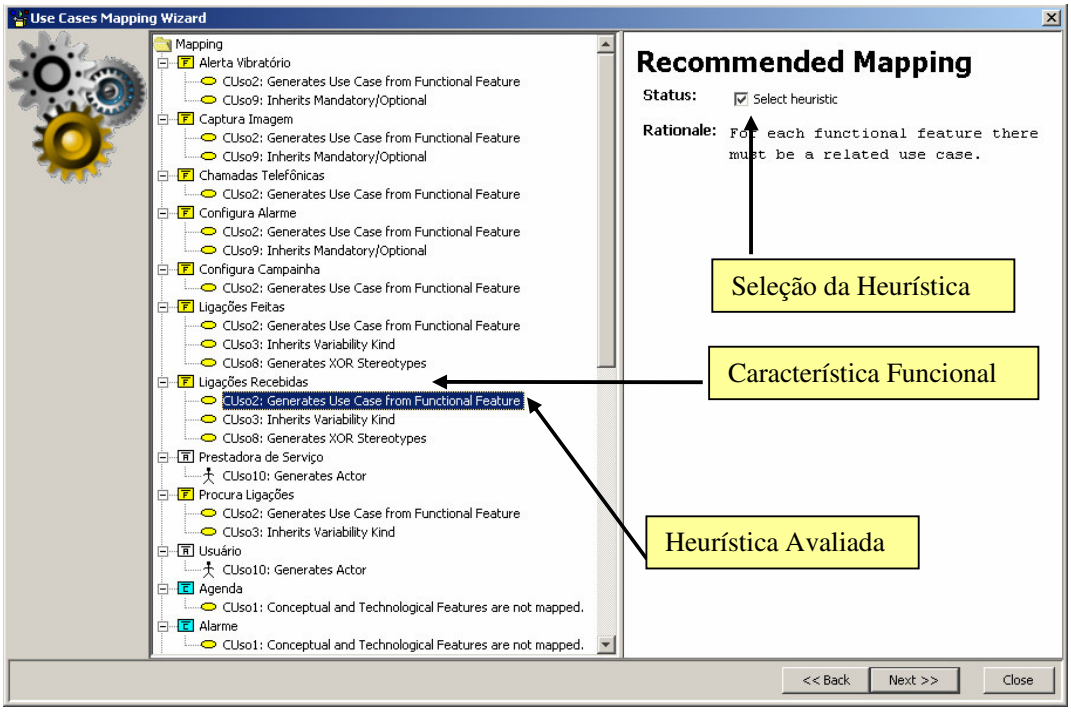


Figura 4.14: Janela de Mapeamento de Características Funcionais em Casos de Uso

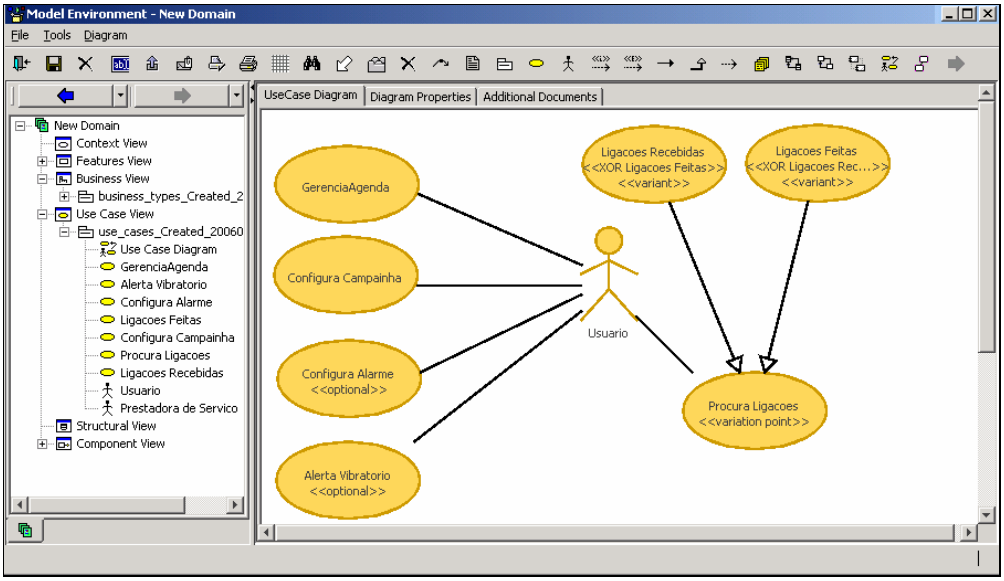


Figura 4.15: Árvore semântica dos Casos de Uso e o Diagrama de Casos de Uso gerado

4.3.2.5 - Heurísticas de Mapeamento de Artefatos de Projeto de Domínio

As heurísticas para o mapeamento de artefatos de análise em componentes do domínio auxiliam a geração de componentes de negócio, processo, utilitário e infra-estrutura. Conforme mostra a Figura 4.16, existe uma classe no pacote arquitetura, denominada *GerenteArquitetura* que é responsável pela geração dos diferentes componentes do domínio. Estes componentes são criados com base em diferentes artefatos do domínio, representados pelas classes *NoUseCase*, *NoTipoNegocio*, *NoComponente* e *NoFeature*, auxiliados por diferentes painéis que auxiliam a tomada de decisões do Engenheiro do Domínio.

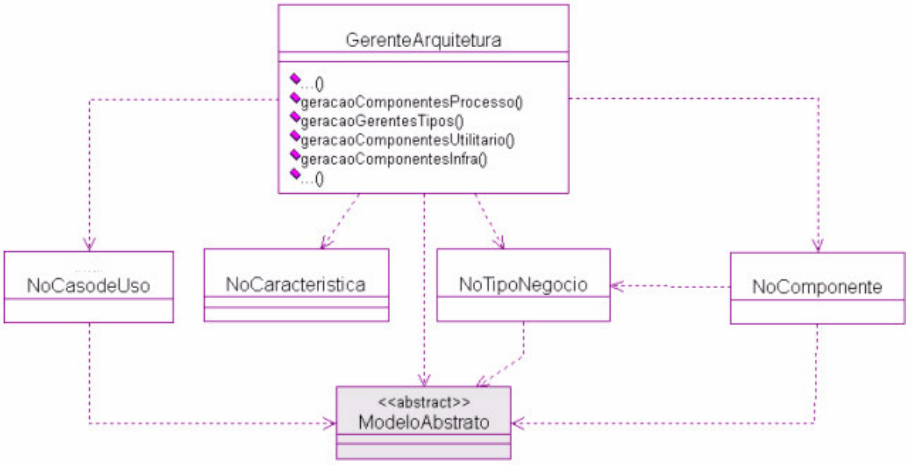


Figura 4.16: Modelo de Classes Simplificado da Geração de Componentes do Domínio

4.3.2.5.1 Geração de Componentes de Negócio

A geração de componentes de negócio no ambiente Odyssey já havia sido implementada no contexto do trabalho de Teixeira (2003). Nesta tese, algumas melhorias foram feitas, dentre elas: a) a identificação das interfaces requeridas pelos componentes, e b) a especificação do componente de negócio, gerando classes e um diagrama de classes para representar a interação dos tipos de negócio gerenciados por um determinado componente. Ainda, foi incluído todo o apoio à representação e mapeamento de variabilidades e opcionalidades dos componentes de negócio gerados, através das heurísticas apresentadas na Seção 3.4.3.

A geração destes componentes é iniciada com o agrupamento dos tipos de negócio, com base nas relações de herança, composição, agregação, associação e dependência.

Para a geração de todos os componentes, incluindo os componentes de negócio, são utilizados assistentes compostos de diferentes janelas. Na geração de componentes de negócio, dentre as várias janelas do assistente, é exibida uma semelhante a Figura 4.17. Do lado esquerdo, temos uma árvore com os grupos pré-formados em função de seus relacionamentos, onde são indicados os candidatos obrigatórios (), recomendados (R) e sugeridos (S) de cada grupo. Ao clicarmos em um dos grupos na árvore, seus detalhes são apresentados do lado direito da janela. Como as heurísticas de geração de componentes de negócio são sugestões para o Engenheiro, utilizamos *checkboxes* para que ele possa selecionar os candidatos que ele preferir.

Quando a *checkbox* é marcada, o candidato passa a fazer parte do grupo que está selecionado, mantendo a consistência da variabilidade e opcionalidade sugerida por cada heurística. Com isso, todos os candidatos que antes faziam parte do grupo do candidato, são trazidos, agora, para o grupo no qual ele foi inserido.

Após concluir a avaliação dos agrupamentos, que estão em consonância com as heurísticas propostas, são gerados os componentes, com suas respectivas interfaces providas e requeridas. A Figura 4.18 mostra um modelo de componentes resultante. Ainda através da mesma figura, se observa que foram mantidas as representações de variabilidades, opcionalidades e regras de exclusão mútua entre os componentes de negócio do domínio.

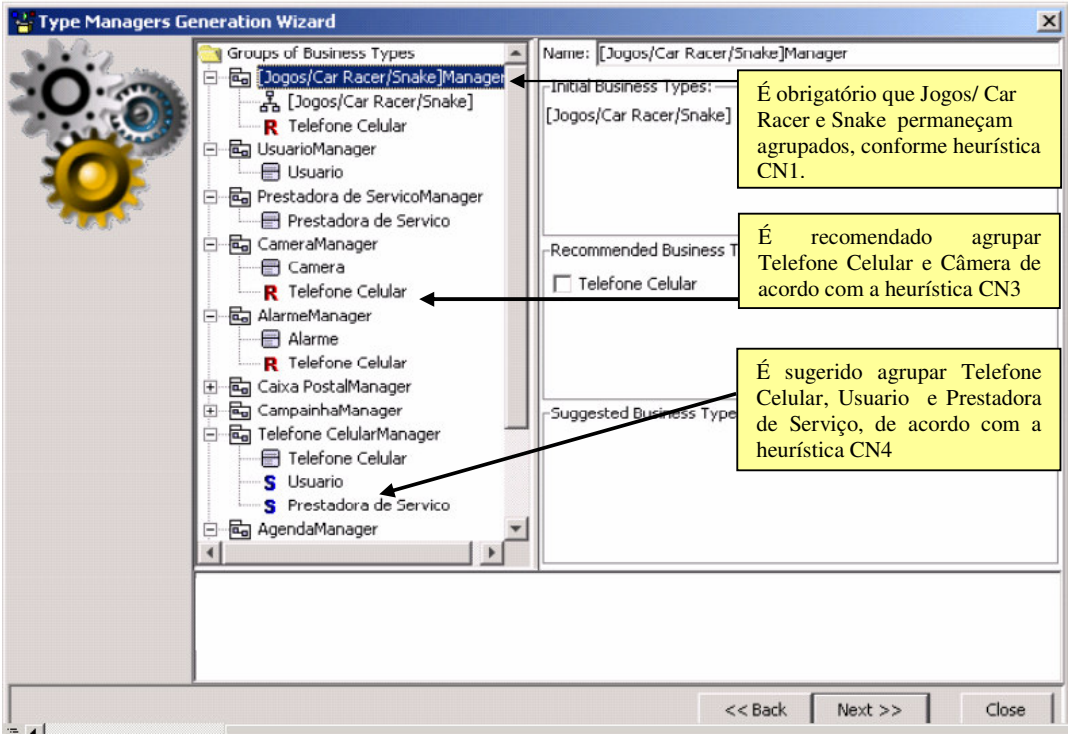


Figura 4.17: Janela de escolha de agrupamentos de tipos de negócio em componentes de negócio

4.3.2.5.2 Geração de Componentes de Processo

Não havia a geração de componentes de processo no ambiente Odyssey. No trabalho de Teixeira (2003), somente era gerado um diagrama de seqüência, que descrevia a interação existente entre interfaces de componentes de negócio, obtida através da descrição dos casos de uso do domínio. No entanto, somente a geração de um diagrama de interação associado a um componente de negócio, não era informação suficiente para indicar que tal modelo representava um componente de processo.

Neste sentido, nesta tese são efetivamente gerados componentes de processo, os quais são criados a partir dos casos de uso do domínio, com base nas heurísticas definidas nas Seções 3.4.3.7 a 3.4.3.10.

Para a geração dos componentes de processo são utilizados assistentes, os quais, dentre outras janelas, exibem aquela apresentada na Figura 4.19. Depois de selecionados os casos de uso utilizados para a geração de componentes de processo, são avaliadas as interfaces dos componentes de negócio requeridas para a criação das operações previstas nestes componentes. Com esta informação, são gerados os componentes de processo, com uma interface fornecida que identifica tal componente, e com tantas interfaces requeridas quantos forem os componentes de negócio dos quais necessita prestar algum serviço. As relações existentes e a especificação de variabilidades e

opcionalidades dos casos de uso selecionados são avaliadas com base nas heurísticas citadas no parágrafo anterior, visando a geração de interfaces requeridas.

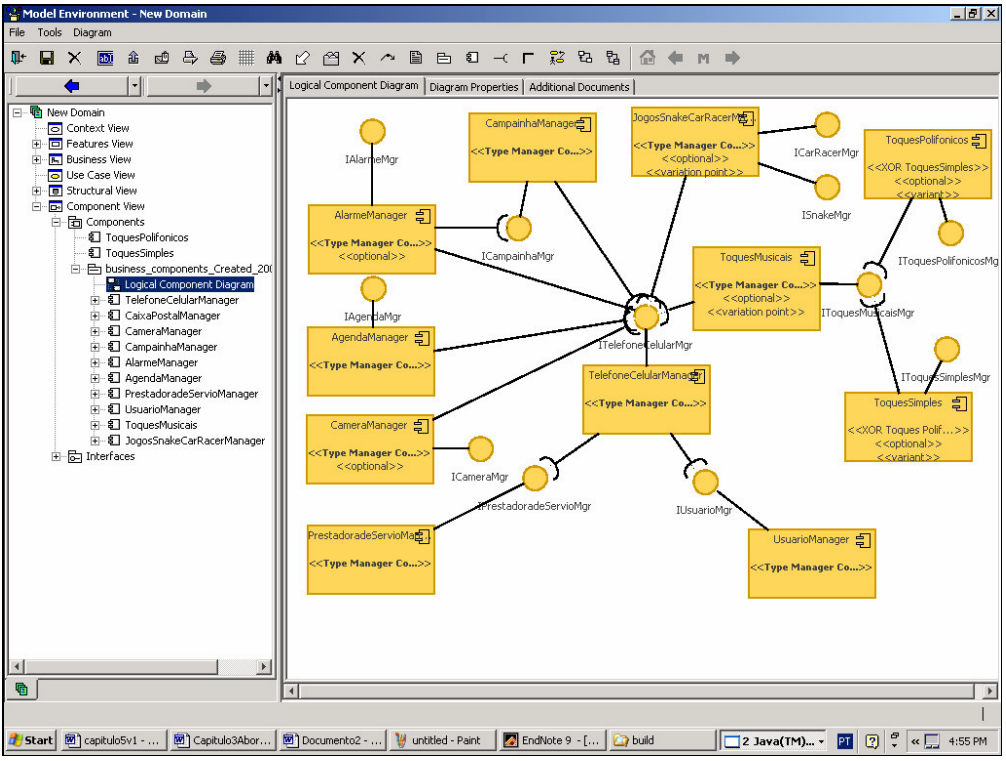


Figura 4.18: Modelo de Componentes de Negócio obtido a partir no modelo de Tipos de Negócio da Figura 4.12.

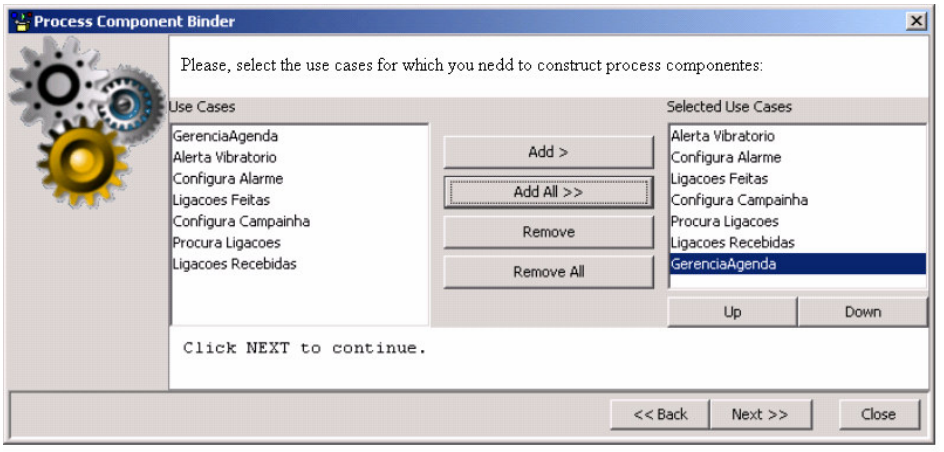


Figura 4.19: Janela de escolha de casos de uso para a geração de componentes de processo

4.3.2.5.3 Geração de Componentes Utilitários e de Infra-Estrutura

Conforme estabelecido no Capítulo 3, estes componentes são gerados a partir das características tecnológicas do domínio, atendendo as heurísticas CUI1 à CUI5,

apresentadas as Seções 3.4.3.11 a 3.4.3.15, respectivamente. Tais componentes não estavam implementados no ambiente Odyssey.

Dentre as telas dos assistentes de mapeamento de componentes utilitários e de infra-estrutura existentes, constam as apresentadas nas Figuras 4.20 e 4.21. Depois de selecionadas as características, são criados os componentes utilitários e de infra-estrutura, um para cada tipo de característica, respeitando as heurísticas definidas para os mesmos. Após o termino de execução destes assistentes, pode ser obtido um modelo de componentes, conforme apresentado na Figura 4.22.

Figura 4.20: Janela de escolha de características de ambiente operacional para a geração de componentes utilitários

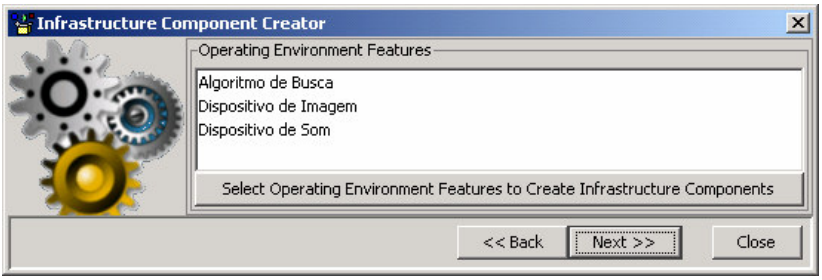


Figura 4.21: Janela de escolha de características de tecnologias do domínio e técnicas de implementação para a geração de componentes de infra-estrutura

Com base neste conjunto de funcionalidades implementando as heurísticas propostas, o Engenheiro de Domínio obtém um conjunto de componentes representativo do domínio, os quais prestam diferentes serviços para o domínio, obtendo como resultado uma arquitetura do domínio preliminar. Esta arquitetura deve ser refinada para que a sua estrutura atenda da melhor forma possível os requisitos não funcionais do domínio, e organize os componentes para que possam ser melhor reutilizados na engenharia de uma aplicação. Neste sentido, é que são propostas as implementações e adaptações descritas nas próximas seções.

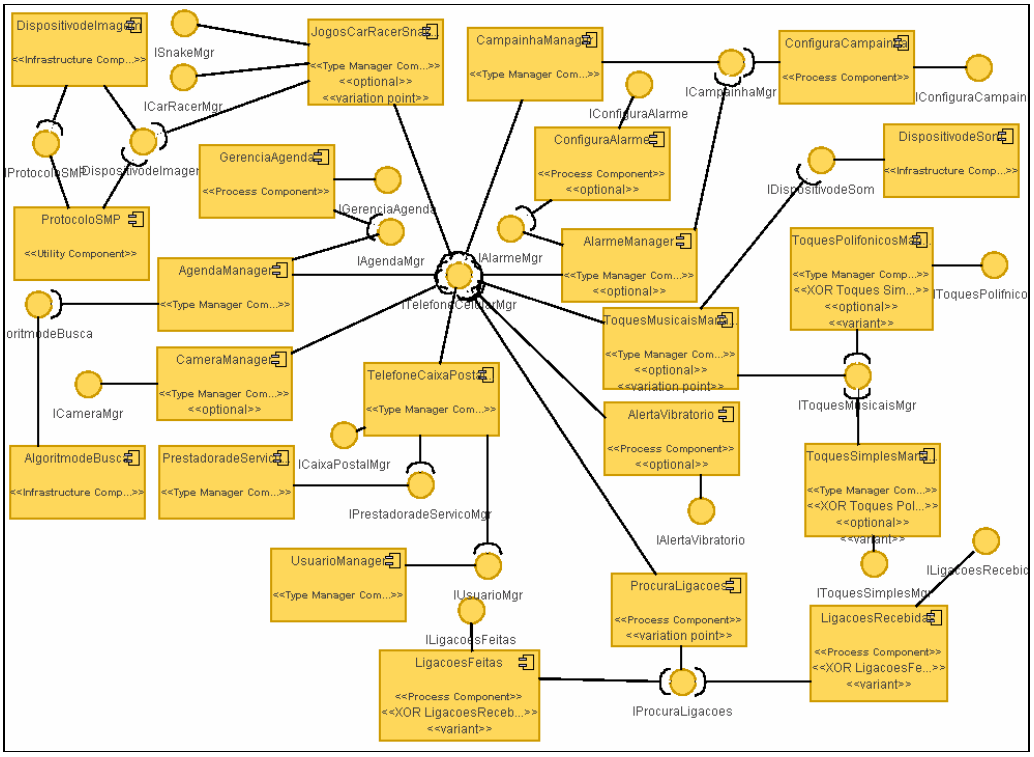


Figura 4.22: Modelo contemplando componentes de negócio, processo, utilitários e de infra-estrutura gerados a partir das heurísticas definidas

4.3.3 - Implementações de *Plug-ins* no Odyssey

Conforme citado no início do capítulo, originalmente o ambiente Odyssey pouco contemplava funcionalidades para apoio à construção de arquiteturas para domínios. O suporte até então oferecido se limitava ao plug-in de suporte aos padrões de projeto (DANTAS *et al.*, 2002) e a geração de componentes de negócio (TEIXEIRA, 2003). Neste sentido, são implementadas as funcionalidades referentes aos critérios para identificação de elementos arquiteturais e a abordagem Odyssey-MDA, sendo esta última desenvolvida em (MAIA, 2006).

4.3.3.1 - GroupCriteria

O *plug-in* GroupCriteria tem por objetivo implementar os critérios de agrupamento de componentes sugeridos na Seção 3.5. A Figura 4.23 mostra um modelo de classes simplificado do *plug-in* GroupCriteria.

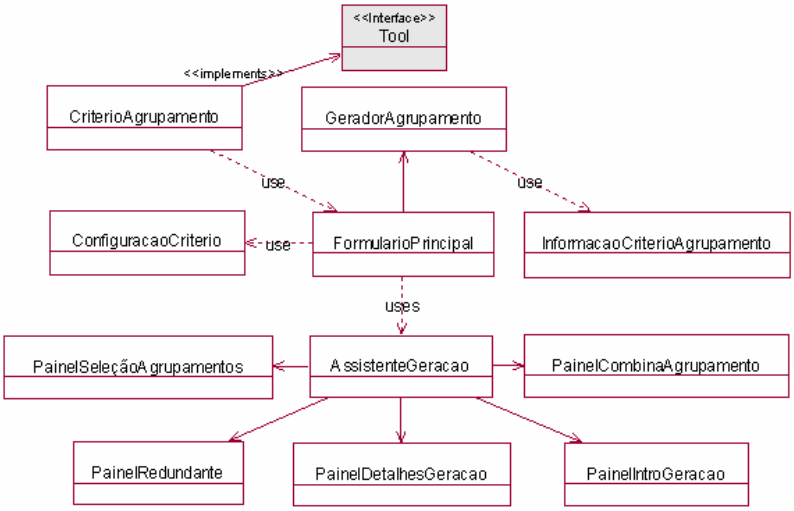


Figura 4.23: Modelo de Classes da Ferramenta de Agrupamento de Componentes

Para a implementação deste *plug-in*, foi estendida a interface *Tool* do Odyssey, a qual é responsável por disponibilizar os *plug-ins* através do ambiente Odyssey. Neste sentido, a classe *CriterioAgrupamento* implementa a interface *Tool*, fazendo a integração entre o *plug-in* e o Odyssey. A classe *FormularioPrincipal* manipula os elementos de interface com o usuário, e usa a classe *ConfiguracaoCriterio* para a seleção dos critérios e componentes do domínio a serem avaliados. As sugestões de agrupamentos são geradas pela classe *GeradorAgrupamentos* a qual utiliza as informações armazenadas em *InformacaoCriterioAgrupamento*. Através da classe *AssistenteGeracao,* os usuários são orientados a selecionar os agrupamentos desejados (*PainelSelecaoAgrupamentos*), combinar agrupamentos selecionados (*PainelCombinaAgrupamento*), identificar redundâncias entre agrupamentos selecionados (*PainelRedundante*), e gerar informações sobre o pacote de componentes de agrupamento gerados pelo *plug-in* (*PainelDetalhesGeracao*).

Para utilização do *plug-in*, o Engenheiro de Domínio deverá acessar o menu *Tools* ► *Components Grouping Criteria*. Ao selecionar esta opção, é aberta a janela (Figura 4.24) onde as funcionalidades principais são disponibilizadas através de botões, ou através do menu *File*.

Conforme mostra a Figura 4.25, através da opção *Configure Criteria*, o Engenheiro de Domínio pode: a) selecionar por meio de *checkboxes* os critérios de agrupamento que deseja avaliar; b) determinar um número máximo de sugestões de acoplamento que pretende obter de cada critério escolhido; c) selecionar os tipos de

componentes que deseja considerar na avaliação dos critérios e, d) indicar a necessidade de atribuição de variabilidade e opcionalidade ao elemento arquitetural gerado.

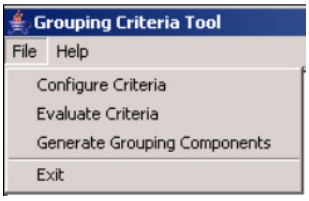


Figura 4.24: Janela Principal do Plug-in GroupCriteria

Depois de executada todas as configurações necessárias, o Engenheiro pode obter informações sobre a execução dos critérios, através da opção *File* ▸*Evaluate Criteria,* apresentado na Figura 4.24.

Ao selecionar esta opção, é mostrada uma janela composta de tabelas, uma para cada critério selecionado. A Figura 4.26 mostra as sugestões de agrupamento, considerando o modelo de componentes da Figura 4.22. Abaixo de cada tabela, são descritas as sugestões de agrupamento de cada critério, respeitando o limite máximo de sugestões definido anteriormente.

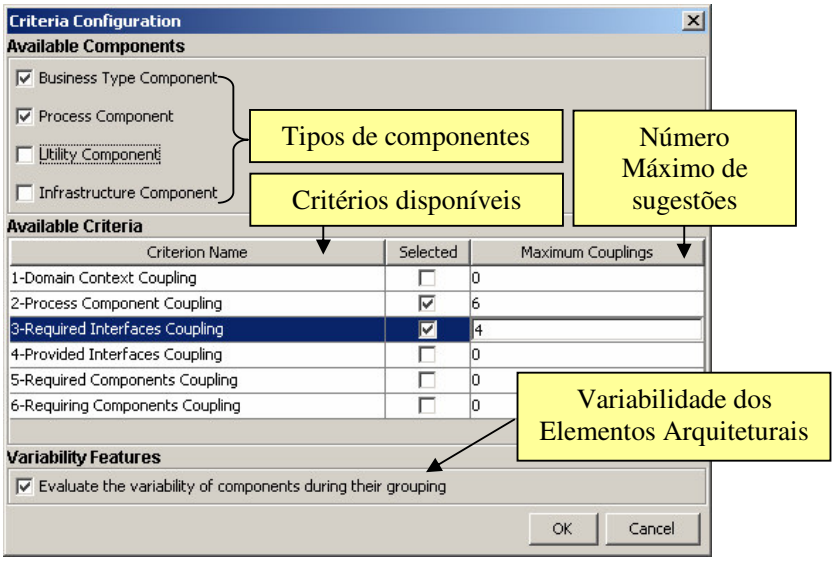


Figura 4.25: Janela de Configuração dos Critérios de Agrupamento

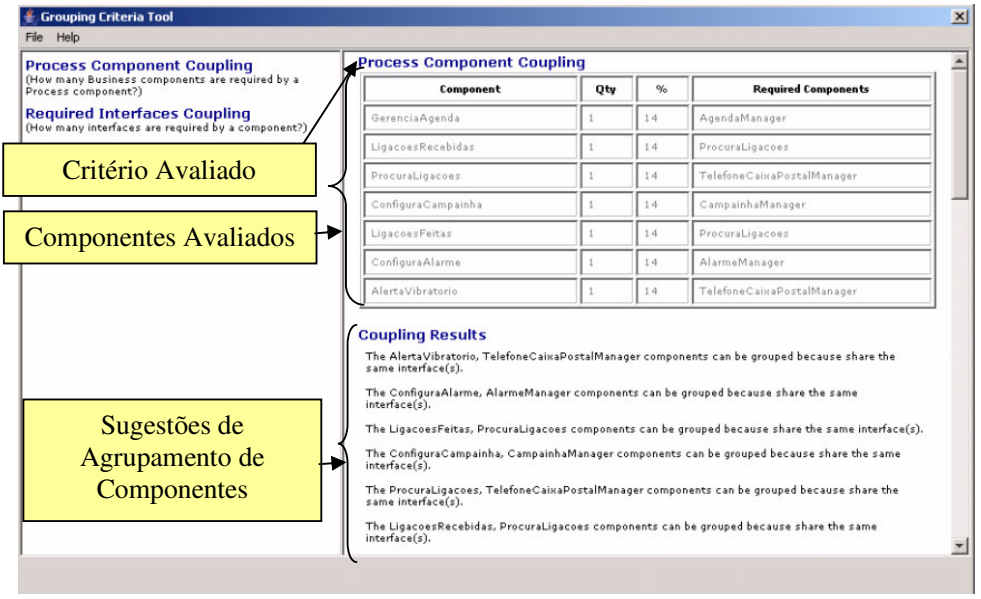


Figura 4.26: Janela de Apresentação dos Resultados dos Critérios Selecionados

Para a geração dos agrupamentos de componentes, o Engenheiro de Domínio deve selecionar o botão *Generate Grouping Components*, conforme apresenta a Figura 4.24.

Através de um assistente de agrupamento, é apresentada, dentre outras, uma janela como mostra a Figura 4.27. Neste exemplo, são listadas todas as sugestões de agrupamento para o critério *Required Interfaces Coupling*. Do conjunto sugerido, o Engenheiro de Domínio selecionou, neste exemplo, as sugestões número 1 e 2. Porém, optou por retirar o componente *PrestadoradeServicoManager* da sugestão número 2.

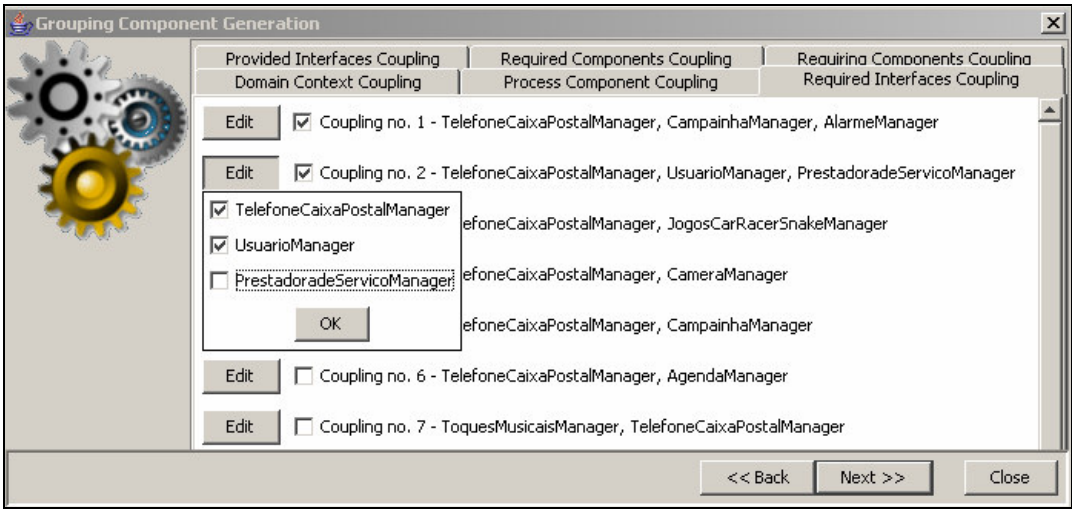


Figura 4.27: Janela de Seleção de Sugestões de Agrupamento

Além de permitir ao Engenheiro de Domínio selecionar quais sugestões deseja aceitar, o Engenheiro de Domínio pode refinar as sugestões, eliminando componentes do agrupamento (Figura 4.27) ou combinando agrupamentos selecionados (Figura 4.28).

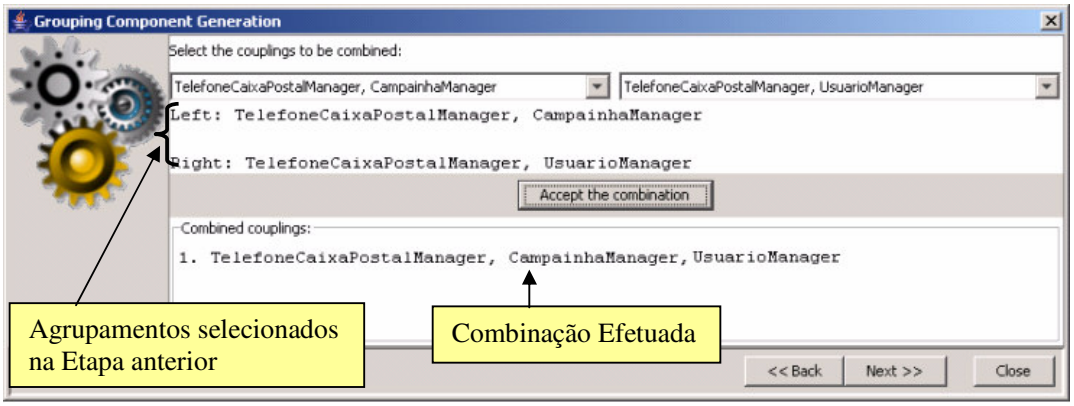


Figura 4.28: Janela de Combinação de Agrupamentos Selecionados

Antes de efetivamente criar os elementos arquiteturais, o *plug-in* detecta se existem agrupamentos redundantes entre as sugestões selecionadas. A Figura 4.29 mostra um exemplo de janela apresentada quando alguma redundância é detectada. Neste caso, o Engenheiro pode escolher somente um dos agrupamentos dentre aqueles listados, ou então, selecionar todos os agrupamentos, mesmo que existam redundâncias entre eles. Neste último caso, não é indicado que o Engenheiro de Domínio utilize estes elementos juntos numa mesma AR.

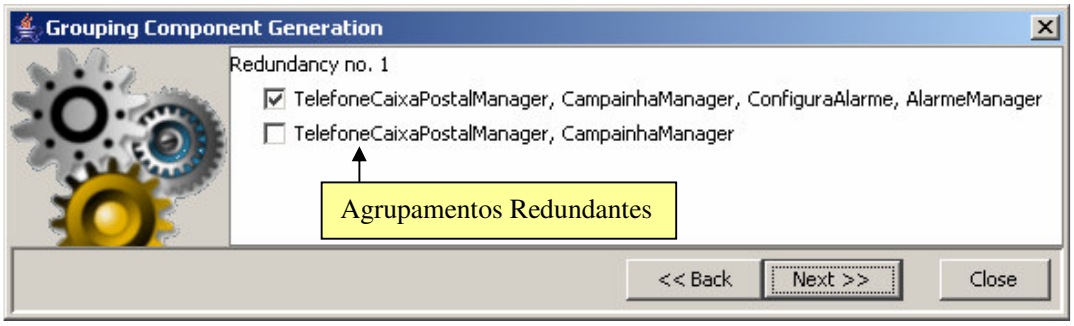


Figura 4.29: Redundâncias entre os Agrupamentos Selecionados

Ao final da atividade, é gerado um pacote com os componentes de agrupamento escolhidos, os quais caracterizam um elemento arquitetural do domínio. Os componentes gerados recebem um estereótipo *<<Grouping Component>> e os estereótipos <<optional>> e <<variant>> quando for necessário*. Para cada

componente, é também gerada uma porta, conforme definido pela UML 2.0. Esta porta gerencia todas as interfaces providas e requeridas pelos componentes que pertencem à este elemento arquitetural. A porta deve fornecer as interfaces dos componentes internos que são requeridas por outros elementos arquiteturais, bem como requerer interfaces providas por outros elementos arquiteturais.

A Figura 4.30 mostra um elemento arquitetural gerado e a sua especificação interna.

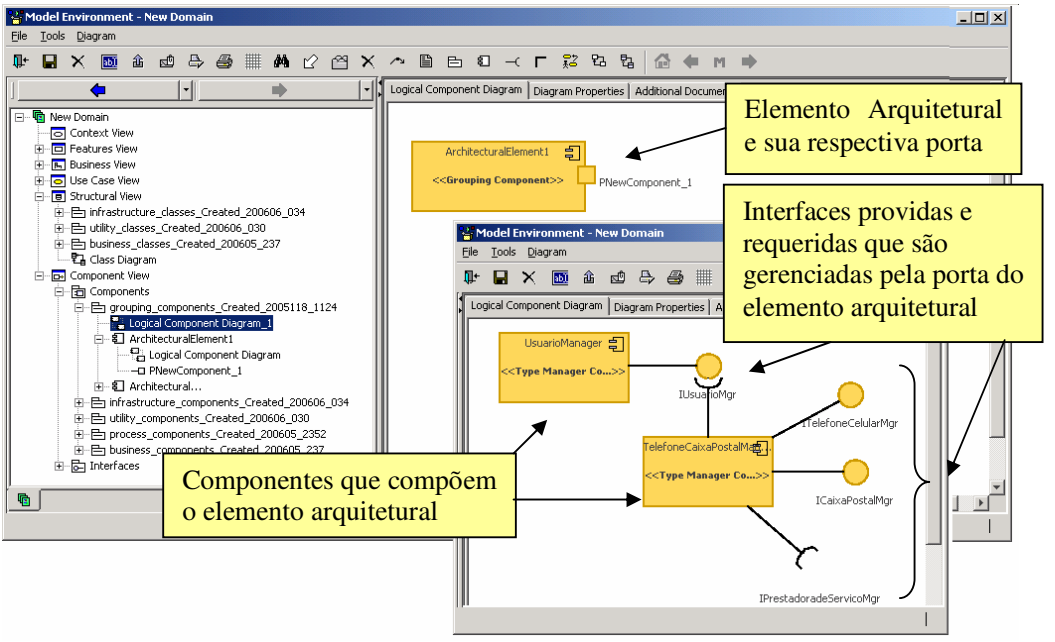


Figura 4.30: Exemplo de Elemento Arquitetural gerados através do *Plug-in*

Neste caso, a figura mostra que o *ArchitecturalElement1* é composto pelos componentes *TelefoneCaixaPostalManager* e *UsuarioManager*. As interfaces providas e requeridas pelos componentes TelefoneCaixaPostalManager e UsuarioManager são disponibilizadas para outros elementos arquiteturais através da porta *PNewComponent_1*. Neste exemplo, a porta *PNewComponent_1* não gerencia a interface requerida *IUsuarioMgr* do Componente *TelefoneCaixaPostalManager*, pois tal interface pertence ao próprio elemento arquitetural, não sendo necessário requerer seus serviços a outros componentes ou elementos arquiteturais da arquitetura do domínio.

Com estes elementos arquiteturais, o Engenheiro pode obter uma nova versão da arquitetura de componentes do domínio, a caminho para obtenção de uma AR.

4.3.3.2 - Odyssey-MDA

O outro *plug-in* desenvolvido no trabalho de Maia (2006), foi o Odyssey-MDA, o qual tem por objetivo, nesta Tese, apoiar a fase de implementação do processo CBD-Arch-DE. A implementação deste *plug-in* atende a um dos objetivos da abordagem DECOM, que é a transformação de modelos independentes de tecnologia, em modelos gerados para uma tecnologia específica, bem como a geração de código fonte.

Com a implementação do *plug-in* Odyssey-MDA, é atingido o objetivo de apoiar a fase de implementação de um domínio através do processo CBD-Arch-DE. Esta fase de implementação, como podemos observar no Capítulo 2, é desprovida de apoio por todas as propostas de ED e DBC analisadas, representando um diferencial importante desta abordagem em relação às demais.

Nesta Seção, são apresentados os detalhes de implementação importantes para o contexto desta Tese. Maiores detalhes podem ser obtidos em (MAIA, 2006).

O Odyssey-MDA é composto de duas ferramentas: ferramenta de transformação de modelos e ferramenta de geração de código. Conforme mostra a Figura 4.31, a comunicação entre o Odyssey e a Odyssey-MDA é estabelecido com o uso do padrão XMI, que permite ao segundo interpretar os modelos UML produzidos no Odyssey.

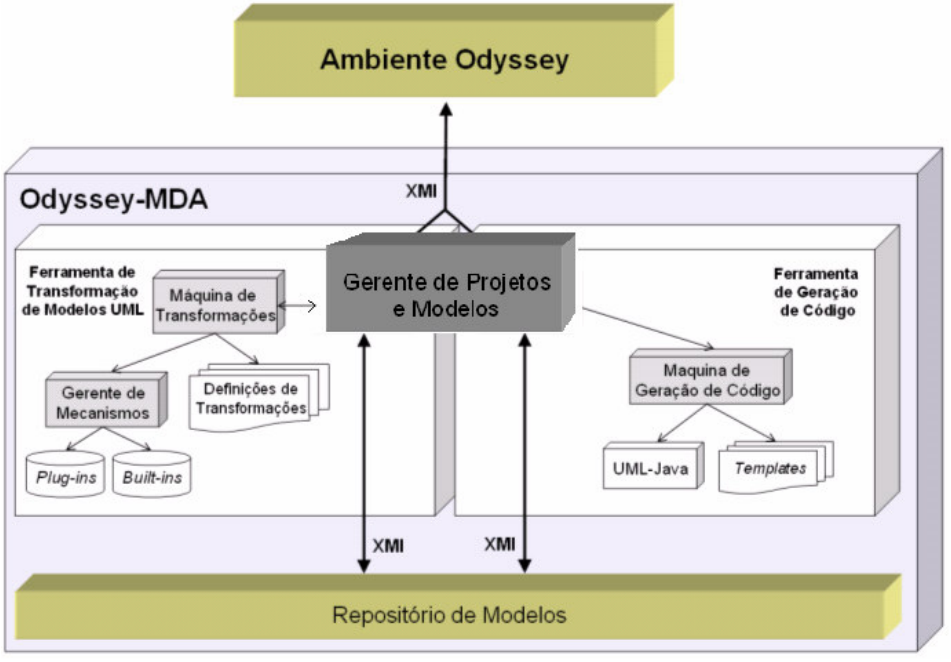


Figura 4.31: Estrutura usada na implementação do protótipo (adaptado de (MAIA, 2006))

Tanto a ferramenta de transformação de modelos quanto a de geração de código utilizam o Gerente de Projetos e Modelos. Este gerente é responsável pela criação, salvamento e carregamento dos projetos de transformação. Através deste gerente o usuário importa e exporta modelos.

Como pode ser observado, na parte esquerda da Figura 4.31, a ferramenta de transformação de modelos PIM em PSM é composta ainda de mais dois elementos principais:

a) a máquina de transformações, que é responsável por ler e validar os arquivos XML das definições das transformações, analisar transformações, instanciar mapeamentos necessários, solicitar a execução de mecanismos de mapeamento e gerar modelos de saída, e,

b) o gerente de mecanismos, que deve instanciar os mecanismos (*built-ins* ou *plug-ins*) solicitados pela Máquina de Transformações.

Na parte direita da figura, a ferramenta de Geração de código realiza as transformações do tipo modelo-código (e.g. modelo UML para Código Java). Esta ferramenta é composta de mais um elemento principal, além do gerente de projeto e modelos, Este elemento é a Máquina de Geração de Código, que é responsável por executar a geração do código fonte.

Para a realização de uma transformação sobre um modelo, é necessário definir os mapeamentos entre os elementos do modelo a ser transformado, denominado modelo de entrada, e os elementos desejados no modelo resultante da transformação, denominado modelo de saída. Neste sentido, a especificação de uma transformação deve ser elaborada em formato *XML*, a qual contém o conjunto de mapeamentos que compõem essa transformação, conforme apresentado na Seção 3.3.3. A Figura 4.32 mostra um trecho da especificação de transformação para a tecnologia EJB - *EJBComponents*. Nas linhas 4 à 17, por exemplo, pode ser visto um trecho da definição do mapeamento entre os componentes do modelo *PIM* e *EJB*.

Com base nesta especificação de transformação *EJBComponents*, o *plug-in* Odyssey-MDA pode executar esta transformação sobre os modelos de componentes gerados no ambiente Odyssey.

O Odyssey executa a exportação e importação dos modelos *XMI*, de forma transparente ao usuário. Neste sentido, cabe ao Engenheiro de Domínio escolher a plataforma e as transformações através de um conjunto de assistentes compostos de diferentes etapas.

```
1   <transformation-mapping name="EJBComponents"
2       implementation-package="br.ufrj.cos.lens.odyssey.tools.mda.transformations.builtin">
3
4       <classifier-map name="Entity-Component-PIM - EntityBean-Component-EJB"
5           implementation="ComponentComponent">
6           <!-- FINDERS -->
7           <finder side="left" criteria="type" value="Component" />
8           <finder side="left" criteria="stereotype" value="entity" />
9           <finder side="right" criteria="type" value="Component" />
10          <finder side="right" criteria="stereotype" value="EntityBeanComponent" />
11
12          <!-- FORWARD PROPERTIES -->
13          <property direction="forward" name="stereotype" value="EntityBeanComponent" />
14              ...
15          </classifier-map>
16          ...
17      </classifier-map>
18  ...
19  </transformation-mapping>
```

Figura 4.32: Especificação XML da transformação EJBComponents

Após a escolha da tecnologia de transformação, devem ser efetuadas as configurações da transformação, como mostra a Figura 4.33. O Engenheiro de Domínio deve selecionar o modelo de entrada e o modelo de saída, informar a direção da transformação (que pode ser de um PIM para um PSM ou vice-versa), bem como informar se deseja visualizar ou não cada mapeamento a ser realizado.

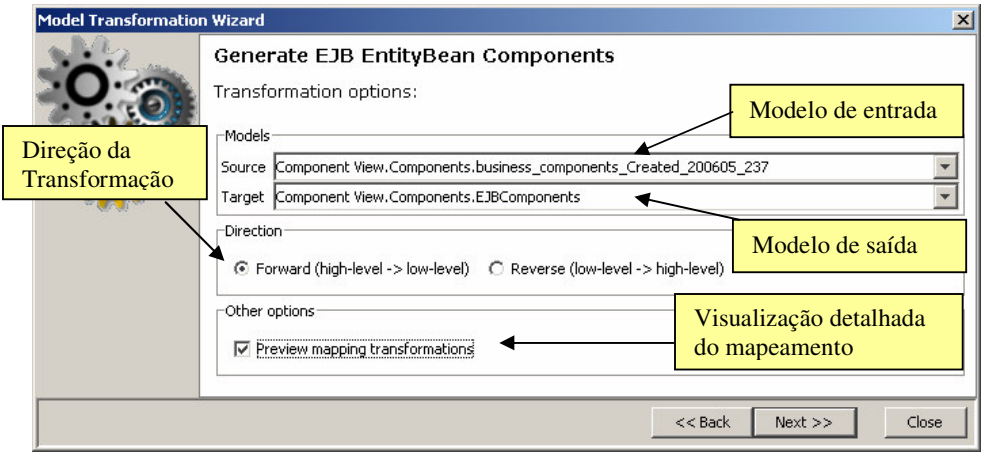


Figura 4.33: Configurações da execução da transformação *EJBComponents*.

Após a configuração, a execução da transformação se inicia, e a máquina de transformação instancia cada mapeamento. A Figura 4.34 mostra as transformações *EJBComponents* que serão realizadas para o componente TelefoneCelular, sendo que o usuário pode desmarcar algum mapeamento indesejado.

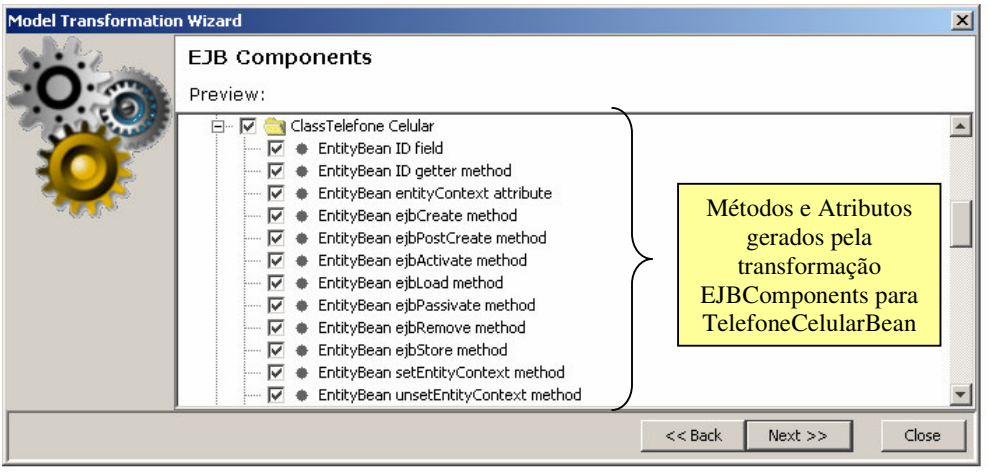


Figura 4.34: Pré-visualização da execução da transformação *EJBComponents*.

Depois da visualização das transformações, o usuário deve pressionar o botão *Next* do assistente de transformação, confirmando a execução dos mapeamentos selecionados. A Figura 4.35 mostra o modelo gerado pelo mapeamento do componente independente de tecnologia *TelefoneCaixaPostalManager* para o componente *TelefoneCelularCaixaPostalManagerBean*. Conforme previsto na transformação *EJBComponents*, cada componente *Entity Bean* é composto por uma classe que representa a sua implementação e possui atributos e métodos padrão, definidos na especificação EJB. Além desta classe, o componente *Entity Bean* possui uma outra classe que encapsula a sua chave primária e um conjunto de interfaces locais e remotas (SUN, 2005a).

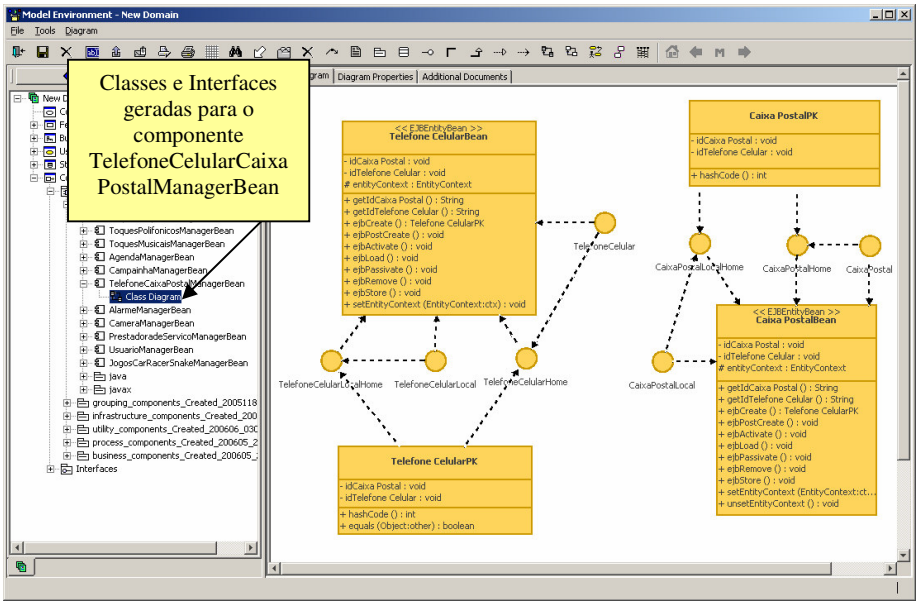


Figura 4.35: Modelo de saída da transformação *EJBComponents*

Tendo como base os modelos gerados com a especificação EJB, a ferramenta para a geração de código pode ser utilizada no ambiente *Odyssey* de forma transparente.

A Figura 4.36 ilustra a utilização do Gerador de Código-fonte no Odyssey-MDA. Uma janela é aberta, solicitando ao Engenheiro a indicação de um diretório onde o código fonte será gerado. Como exemplo, a Figura 4.37 mostra o resultado do código gerado para a classe *TelefoneCelularBean*.

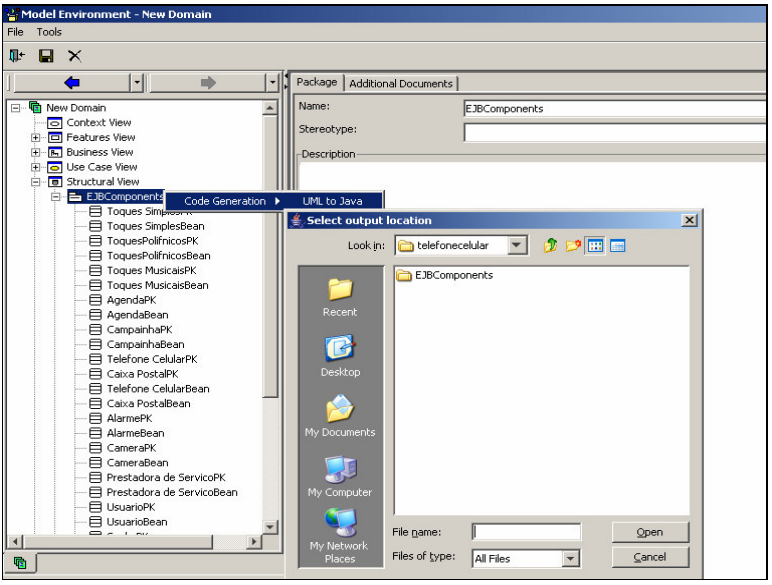


Figura 4.36: Geração de código UML para Java no Ambiente *Odyssey*

```
/* Generated by java_class_template.vm */
package EJBComponents;

public class TelefoneCelularBean {
        /* Attributes */
        private void idCaixaPostal;

        private void idTelefoneCelular;

        protected EntityContext entityContext;

        /* Methods */
        public String getIdCaixaPostal() {
        }

        public String getIdTelefoneCelular() {
        }

        public TelefoneCelularPK ejbCreate() {
        }

        public void ejbPostCreate() {
        }

        public void ejbActivate() {
        }

        public void ejbLoad() {
        }

        public void ejbPassivate() {
        }

        public void ejbRemove() {
        }

        public void ejbStore() {
        }

        public void setEntityContext() {
        }

        public void unsetEntityContext() {
        }
}
```

Figura 4.37: Código gerado para o TelefoneCelularBean

4.3.3.3 - *Plug-ins* Estendidos

Para total apoio ao CBD-Arch-DE, dois outros plug-ins pré-existentes foram estendidos: um no contexto desta Tese (i.e. Odyssey-Patterns (DANTAS *et al.*, 2002)) e outro no contexto do trabalho de Oliveira (2006) (i.e. Oráculo (DANTAS *et al.*, 2001)).

4.3.3.3.1 - Oráculo

O Oráculo consiste, originalmente, de um sistema de críticas para modelos UML, o qual notifica ao usuário inconsistências existentes em, por exemplo, modelos de classes e casos de uso. O Oráculo é uma ferramenta auxiliar à ferramenta de diagramação do Odyssey, que monitora a inclusão, remoção e edição de elementos em modelos, enviando notificações aos desenvolvedores a cada regra de boa formação violada.

Em (OLIVEIRA, 2006) foi desenvolvida uma extensão do Oráculo para verificar a consistência dos modelos de características. É através do modelo de características que todas as opcionalidades e variabilidades do domínio são especificadas, afetando o bom uso das heurísticas de mapeamento de variabilidades nos diferentes artefatos do domínio e, conseqüentemente, no resultado das arquiteturas de referência geradas para o domínio modelado. Maiores detalhes sobre a extensão podem ser obtidos em (OLIVEIRA, 2006).

Embora esta extensão amplie o apoio oferecido pelo Oráculo na manutenção de consistência em modelos do domínio, nem todos os artefatos previstos no CBD-Arch-DE são avaliados por este sistema de críticas. Neste sentido, é necessário apliar a extensão deste sistema de tal sorte que tipos de negócio e componentes possam também ser verificados e consistidos na modelagem do domínio.

4.3.3.3.2 – Odyssey-Patterns

O Odyssey-Patterns, originalmente, é uma ferramenta para apoio a especificação e instanciação de padrões para o projeto de software, incluindo padrões de projeto (GAMMA *et al.*, 1995) e arquiteturais (BUSCHMANN *et al.*, 1996). Conforme proposto em (XAVIER, 2001), a instanciação dos padrões arquiteturais é desenvolvida com base num conjunto de requisitos não-funcionais identificados pela ISO 9126 (1992), visando a avaliação de qualidade de software. Na atual implementação do *plug-in*, um padrão arquitetural pode ser instanciado para um conjunto de classes pré-existentes, ou geradas para exercer o papel do elemento arquitetural instanciado.

Durante a fase de projeto do domínio no processo CBD-Arch-DE, o Engenheiro deve montar possíveis arquiteturas de referência para o domínio. Neste sentido, os *plug-ins* GroupCriteria e Odyssey-Patterns dão apoio, em conjunto, a definição de arquiteturas de referência no contexto do CBD-Arch-DE.

Por esta razão, foi feita uma extensão do *plug-in* Odyssey-Patterns para prover o apoio à instanciação de padrões arquiteturais sobre componentes do domínio. Assim, cada elemento arquitetural do padrão pode ser instanciado para um ou mais componentes do domínio. Os atributos e métodos de cada elemento arquitetural são instanciados para interfaces providas pelos componentes que correspondem ao elemento arquitetural avaliado, sob orientação do Engenheiro de Domínio. Já as relações existentes entre os elementos arquiteturais, são instanciadas em trocas de mensagens através de interfaces providas e requeridas.

4.4 - Considerações Finais

Este Capítulo apresentou o arcabouço de implementação para dar apoio à abordagem DECOM.

Uma vez que a DECOM está contextualizada na área de ED e no grupo de reutilização da COPPE/UFRJ, foi utilizada a infra-estrutura do ambiente Odyssey como base para as implementações específicas desta Tese.

Um dos objetivos desta pesquisa é que, para cada atividade proposta no processo CBD-Arch-DE, existisse um ferramental mínimo para a sua execução. Embora as implementações desenvolvidas devam ser consideradas como protótipos, podemos afirmar que este objetivo foi atingido.

Tais implementações foram desenvolvidas no contexto desta Tese e através de duas Dissertações de Mestrado, as quais envolveram:

- adaptações (notação Odyssey-FEX) e implementações (processo CBD-Arch-DE e heurísticas de mapeamento) no kernel do Odyssey, e;

- extensões (Oráculo e Odyssey-Patterns) e implementações (GroupCriteria e Odyssey-MDA) de *plug-ins* .

Em geral, as implementações exploram o uso dos recursos da própria linguagem Java. Somente o *plug-in* Odyssey-MDA utilizou padrões adicionais (e.g. JMI (DIRCZE, 2002), XMI (OMG, 2002b), MOF (OMG, 2002a) e Velocity (APACHE, 2005)).

A especificação oficial da documentação sobre a super-estrutura da UML 2.0 (OMG, 2005) foi disponibilizada há alguns poucos meses pela OMG. Desde, então, foi

feito um esforço para que os aspectos importantes desta nova especificação pudessem estar contemplados nesta Tese, em especial no que diz respeito a especificação de componentes. A implementação utilizada nesta Tese contempla os principais construtores (i.e. componentes, interfaces e portas) desta notação, no que diz respeito à modelagem de componentes.

Contudo, como trabalhos futuros pretende-se estender o apoio oferecido à UML 2.0, em especial ao restante da especificação e notação referente a componentes, mas também extensões que envolvam casos de uso e tipos de negócio, sendo este último através da especificação de classes, atribuída pela UML 2.0.

No que se refere a especificação de componentes, as extensões previstas são com relação as diferentes representações de componentes e a especificação do conector de delegação, conforme apresentados no Anexo 2. Como o elemento arquitetural desta proposta de tese é especificado sob a forma de componente estereotipado (i.e. << components grouping>>), o papel deste conector é o de delegar à sua respectiva porta, a função de prover os serviços prestados pela interface provida por um determinado componente que internamente o compõe. A inclusão deste conector de delegação dará maior semântica ao elemento arquitetural proposto, influenciando na consistência da AR.

Do ponto de vista do ambiente Odyssey, as implementações descritas neste capítulo dão apoio a um ciclo de vida para um processo de ED, o qual não existia de forma completa e integrada anteriormente.

Uma outra contribuição importante desta pesquisa é que nenhum dos métodos de ED e DBC, discutidos no Capítulo 2, possuem infra-estrutura de reutilização que apóie as suas atividades, como proposto pelo Odyssey. Neste sentido, este ambiente e, sobretudo, as contribuições desta tese neste ambiente, permitem que a Engenharia de um Domínio possa de fato ser utilizada, com base num ambiente de reutilização real.

As implementações e, sobretudo, esta pesquisa, representam uma contribuição importante para o grupo de reutilização onde está inserida e, sobretudo, para a comunidade de desenvolvimento de software, que é carente de abordagens e ferramentas que efetivamente apóiem o desenvolvimento *com* e *para* reutilização, baseado em componentes de software.

Finalmente, algumas das funcionalidades implementadas no *kernel e/ou plug-ins* foram avaliadas através de estudos de observação. Estes estudos são discutidos no Capítulo 5.

Capítulo 5 - Estudos de Observação

5.1 - Introdução

O desenvolvimento de atividades de experimentação sobre um método, processo, técnica ou ferramenta, tem sido uma das formas cientificamente aceita para obter indícios sobre a viabilidade, eficiência, enfim, os benefícios (ou não) esperados de alguma destas propostas.

Em (TRAVASSOS *et al.*, 2002), os autores destacam o crescimento no número de trabalhos científicos com atividades de experimentação em ES e, como conseqüência, a oportunidade dos pesquisadores se concentrarem em abordagens promissoras e rejeitarem as idéias duvidosas.

Segundo Wohlin *et al.* (2000), existem três principais estratégias de experimentação:

- Pesquisa de campo: é uma investigação executada em retrospectiva e conduzido para avaliar algumas técnicas ou ferramentas já utilizadas. Geralmente, são utilizados entrevistas e questionários para efetuar a coleta de dados.
- Estudos de caso: são usados para monitorar projetos, atividades ou atribuições. Através destes estudos, é possível observar um atributo ou estabelecer relacionamentos entre diferentes atributos. Estudos de caso podem ser organizados para comparar resultados de projetos similares, um utilizando uma tecnologia atual e outro uma nova tecnologia. Também podem ser usados para aplicar um método aleatoriamente atribuído a um projeto específico de uma empresa, e não atribuí-lo a outros da mesma empresa como forma de comparação.
- Experimentos: é uma atividade geralmente efetuada em laboratório, com o propósito de descobrir algo desconhecido ou de testar uma hipótese. Experimentos são apropriados para confirmar teorias, avaliar a predição dos modelos ou validar as medidas. As principais características dos experimentos estão no controle total sobre o processo e variáveis, e na possibilidade de ser repetido.

Wohlin *et al.* (2000), ainda, propõem um conjunto de etapas para um processo de experimentação.

A primeira etapa compreende a **definição do estudo experimental**. Nesta etapa, os autores sugerem o uso da estrutura proposta por (BRIAND *et al.*, 1996), a qual consiste dos seguintes elementos:

Analisar <<objeto de estudo>>
Com o propósito de <<qual é a intenção>>
Com respeito a <<qual o efeito a ser estudado>>
Do ponto de vista <<quem participa do estudo>>
No contexto de <<onde o estudo será conduzido>>

Na segunda etapa, deve ser feito o **planejamento do estudo experimental**, que é composto pelas seguintes atividades:

- Seleção do contexto em que o estudo será desenvolvido;
- Formulação das hipóteses do estudo experimental;
- Seleção das variáveis dependentes (servem como entrada para o experimento) e independentes (a(s) resposta(s) do experimento);
- Seleção dos executores;
- Projeto do Experimento, contendo o treinamento, planejamento das atividades, e as diferentes formas de participação (e.g. usando uma tecnologia, *ad-hoc*);
- Instrumentação, ou seja, o que será utilizado para avaliar o objeto de estudo;
- Avaliação da validade do estudo.

Após o planejamento, ocorrem as etapas de execução do estudo experimental, análise e interpretação dos resultados obtidos e, apresentação e empacotamento do estudo.

Neste capítulo serão apresentados três estudos de observação, que são considerados um tipo de estudo de caso. Foi escolhida esta técnica de experimentação, pois o objetivo destes estudos é obter indícios de que o objeto de estudo possa ser adequado ao propósito para o qual foi desenvolvido.

Por esta razão, a Seção 5.2 descreve os estudos de observação. A Seção 5.3 apresenta uma análise de fatores que interferem no resultado dos estudos apresentados na Seção 5.2, os quais são considerados como lições aprendidas. Por último, na Seção 5.4 são apresentadas as considerações finais deste Capítulo.

5.2 - Estudos de Observação

Nesta Seção, são apresentados os estudos de observação para a avaliação da DECOM, utilizando os protótipos desenvolvidos no ambiente Odyssey Light.

A proposta inicial do estudo de observação era avaliar todas atividades que compreendem o processo CBD-Arch-DE. No entanto, logo se percebeu que efetuar um estudo de observação para o processo como um todo, exigiria um tempo considerável para o seu planejamento, treinamento, execução, repetições necessárias e avaliação de resultados, que transcendiam o contexto desta Tese. Partiu-se, então, para a identificação de atividades estratégicas do CBD-Arch-DE.

A Tabela 5.1 recapitula as principais tarefas do CBD-Arch-DE, sendo que as escolhidas para os estudos de observação estão formatadas em itálico.

Foram escolhidas atividades de identificação de tipos de negócio, de criação de componentes de negócio e de geração de elementos arquiteturais. Tais atividades foram escolhidas pois, do ponto de vista de uma AR baseada em componentes, representam atividades estratégicas no contexto do CBD-Arch-DE. No entanto, para avaliar a viabilidade do processo como um todo, todas as atividades devem ser avaliadas.

A atividade de *Identificação de tipos de negócio* foi escolhida, pois através dela podemos avaliar o apoio oferecido por um conjunto de heurísticas para o mapeamento de características do domínio em outros artefatos referentes à análise do domínio. Através das heurísticas exploradas nesta atividade, o Engenheiro de Domínio pode especificar tipos de negócio com suas respectivas variabilidades, opcionalidades e relacionamentos, os quais por sua vez são considerados para a geração de componentes de negócio do domínio, durante a fase de projeto. Esta atividade é avaliada através do 1º. Estudo de Observação (E/OBS-1), descrito na Seção 5.2.1.

A outra atividade escolhida para avaliação foi a de *Criação de Componentes de Negócio*, a qual refere-se ao 2º. Estudo de Observação (E/OBS-2) descrito nesta tese. Esta escolha se deve, principalmente, pela importância da atividade no contexto de um processo de ED, pois são gerados componentes que por sua vez, poderão fazer parte de elementos arquiteturais de uma possível AR. Adicionalmente, se justifica esta escolha

pelo fato de representar uma atividade da fase de projeto do CBD-Arch-DE e, também, por permitir a avaliação o se o conjunto de heurísticas de mapeamento de tipos de negócio em componentes de negócio atende às necessidades de mapeamento dos Engenheiros de Domínio. Esta atividade é avaliada através do 2º. Estudo de Observação (E/OBS-2), descrito na Seção 5.2.2.

Por último, a atividade escolhida para avaliação foi a de *Agrupamento de Componentes*. Esta escolha se justifica pelo fato desta atividade representar uma contribuição importante para a geração de arquiteturas de referência para o domínio, considerando a semântica dos artefatos do domínio para a criação de elementos arquiteturais. Tal estudo, identificado como 3º. Estudo de Observação (E/OBS-3), é descrito na Seção 5.2.3.

Tabela 5.1: Atividades do processo CBD-Arch-DE

Fases do Processo	Atividades Principais
Fase de Análise	Identificação de Contextos
	Identificação de Características
	Identificação de Tipos de Negócio (E/OBS-1)
	Identificação de Casos de Uso
Fase de Projeto	*Criação de Componentes de Negócio*, Processo, Utilitário e de Infra-estrutura *(E/OBS-2)*
	Agrupamento de Componentes (E/OBS-3)
	Montagem da Arquitetura
Fase de Implementação	Transformação de Modelos
	Geração do Código

5.2.1 - Estudo de Observação 1 (E/OBS-1)

5.2.1.1 - Objetivo Global

O objetivo geral deste estudo é avaliar a utilização das heurísticas propostas nesta tese para o mapeamento de características do domínio em tipos de negócio.

Neste estudo, pretende-se observar se os tipos de negócio obtidos através das heurísticas satisfazem os requisitos especificados no modelo de características do domínio. Assim, pode ser observado se estas heurísticas de mapeamento são úteis para atender as necessidades de mapeamento possíveis destes diferentes artefatos da análise do domínio.

5.2.1.2 - Objetivo do E/OBS1

Analisar as heurísticas de mapeamento de características do domínio em tipos de negócio
Com o propósito de caracterizar
Com respeito a utilização das heurísticas propostas
Do ponto de vista do Engenheiro de Domínio
No contexto de modelagem de tipos de negócio do domínio de Ensino Colaborativo apoiado por Computador

5.2.1.3 - Planejamento do E/OBS1

5.2.1.3.1 - Seleção do Contexto

Dois requisitos nortearam a seleção do contexto desta pesquisa: a) a possibilidade de viés da população; b) a seleção de profissionais com perfis de Engenheiro de Domínio. Para evitar o viés no estudo de observação, foram convidadas a participar, através de e-mail, pessoas que não estão ligadas a esta pesquisa, e tampouco, ao grupo de reutilização de software que está relacionada. Neste sentido, o primeiro requisito foi atendido. Por outro lado, foi complexo encontrar uma população sem viés e, ao mesmo tempo, engajadas em atividades de engenharia no domínio de Ensino Colaborativo apoiado por Computador. A população do estudo de observação obteve algum tipo de contato com este domínio, mas não necessariamente em atividades de engenharia.

Foi enviada uma mensagem eletrônica aos Professores de Engenharia de Software do PPGCC-FACIN-PUCRS (Anexo 3). O conteúdo desta mensagem era composto de: a) uma breve descrição dos objetivos desta pesquisa de doutorado; b) uma motivação descrevendo a necessidade de efetuar estudos de observação para algumas das tarefas do processo CBD-Arch-DE; c) uma justificativa para a necessidade de pessoas para participar deste estudo; d) a indicação de que tais executores poderiam ser alunos que desenvolvem pesquisas em ED.

Para a execução do estudo, foi utilizado o domínio de Ensino Colaborativo apoiado por Computador, no contexto de Instituições de Ensino Superior, com ênfase aos requisitos que correspondem a estrutura acadêmica de cursos de extensão, graduação e pós-graduação. Os alunos deveriam utilizar o ambiente Odyssey, mais

especificamente o modelo de características baseado na Notação Odyssey-FEX, e as funcionalidades do protótipo que auxiliam na utilização das heurísticas a serem observadas.

5.2.1.3.2 - Seleção dos Indivíduos

Como resultado da solicitação enviada aos professores, foi reunida uma população de 6 alunos de Pós-Graduação. Estes alunos foram graduados em diferentes instituições de ensino superior. Atualmente, são alunos ligados à área de ES e fizeram a disciplina de reutilização de software, na qual tiveram contato conceitual com as áreas de ED e DBC.

A partir deste resultado, foi feito um convite, também através de mensagem eletrônica, individualmente, a cada aluno indicado (Anexo 3). Os alunos responderam à mensagem enviada, concordando em participar do estudo e informando que se adequavam ao perfil desejado.

Assim, a população deste estudo é composta por alunos de ES do PPGCC-FACIN-PUCRS, que fizeram graduação em diferentes Instituições e nunca desenvolveram algum trabalho com o observador e pesquisador que efetuou este plano de observação.

A partir desta confirmação de participação, foram trocadas mensagens, num período de duas semanas, individualmente, com cada executor, para marcar data e horário de execução do estudo, que ocorreu nas dependências do PPGCC-FACIN-PUCRS, em dois dias consecutivos.

5.2.1.3.3 – Seleção das Variáveis

As variáveis independentes deste estudo são:

- experiência dos executores sobre atividades de análise de domínio;
- experiência sobre o domínio de Ensino Colaborativo apoiado por Computador;
- o modelo de características apresentado aos executores;
- o resultado do treinamento da notação Odyssey-FEX.

A variável dependente deste estudo é o modelo de tipos de negócio resultante.

5.2.1.4 - Operação do E/OBS1

5.2.1.4.1 - Preparação

Nesta etapa, os executores fizeram o treinamento da Notação Odyssey-FEX, à distância, antes do dia marcado para o estudo de observação. No dia do estudo, os executores preencheram o Formulário de Consentimento (Anexo 4) e o Formulário de Caracterização (formação acadêmica e experiência profissional) (Anexo 5).

5.2.1.4.2 - Treinamento da Notação e Heurísticas

Foi efetuado um treinamento prévio, à distância, da Notação Odyssey-FEX. Este treinamento foi desenvolvido para que os executores pudessem compreender a semântica do modelo de característica do domínio do estudo de observação. Foi enviada uma mensagem a cada executor, com um documento em anexo (Anexo 6). Este documento continha uma breve descrição sobre reutilização, ED e modelo de características, e notação Odyssey-FEX. Para auxiliar a compreensão da notação, o documento também continha, a título de exemplo, parte de um modelo de características do domínio de Telefonia Móvel. Os alunos que tivessem dúvidas quanto ao treinamento poderiam enviar mensagem eletrônica questionando sobre as mesmas. Além disso, antes da execução do estudo, foi apresentado o conjunto de heurísticas para mapeamento de características conceituais em tipos de negócio, explorando as funcionalidades do protótipo, apresentadas na Seção 4.3.2.4.1.

5.2.1.4.3 - Utilização das Heurísticas

No dia do estudo de observação, foi apresentado aos executores um modelo de características do domínio de Ensino Colaborativo apoiado por Computador (Anexo 7), desenvolvido no ambiente de reutilização Odyssey, baseado na notação Odyssey-FEX. Este modelo foi desenvolvido com base num conjunto de requisitos para este domínio, definidos em (BLOIS, 2003). O modelo contempla características conceituais, tecnológicas, funcionais, cada qual com suas opcionalidades, variabilidades, relacionamentos e, em alguns casos, regras de composição, conforme apresenta o Anexo 7.

Os executores analisaram o modelo de características do domínio, e posteriormente, utilizaram as heurísticas para mapeamento de características conceituais do domínio em tipos de negócio, apresentadas na Seção 3.4.1.

Cada executor, neste momento, fez a análise de cada característica conceitual do domínio, visando avaliar se as heurísticas sugeridas poderiam ser aplicadas ou não.

O resultado desta atividade foi a geração de um pacote com os tipos de negócio, baseado nas heurísticas selecionadas pelo executor do E/OBS-1. Ao final da atividade de mapeamento, os executores preencheram o formulário de avaliação apresentado no Anexo 8. Para cada questão do formulário, os usuários responderam o item objetivo (e.g. sim, não, parcialmente) e o item subjetivo (e.g. justificativa das respostas). O resultado do mapeamento e das respostas do formulário de avaliação é que efetivamente foram considerados para avaliar a viabilidade das heurísticas.

O observador do estudo conduziu o treinamento dos participantes, seja com relação a notação Odyssey-FEX, seja com relação as heurísticas a serem observadas. Além disso, o observador acompanhou o desenvolvimento das atividades dos executores, registrando o tempo gasto por participante para cada atividade (e.g. preenchimento de formulários, execução do estudo) e dúvidas com relação ao entedimento das atividades e preenchimento de formulários.

5.2.1.5 - Avaliação do E/OBS1

Nesta seção, serão discutidos os resultados das participações dos executores do estudo, preenchidos através do Anexo 7. O estudo de observação foi executado em dois dias consecutivos. No primeiro dia foi executado por 3 participantes e no segundo dia por outros 3..

Inicialmente, alguns executores tiveram dificuldades para compreender a proposta das heurísticas. Tais dificuldades são derivadas principalmente pelo entendimento equivocado dos conceitos de variabilidade e opcionalidade, e pelo relacionamento *implemented by,* propostos na notação Odyssey-FEX, e que não faz parte da UML. No entanto, tais dificuldades foram solucionadas anteriormente, com o auxílio do observador, a atividade de mapeamento.

Os resultados obtidos neste estudo estão organizados nas Tabelas 5.2 e 5.3. Com base nos resultados da Tabela 5.2, se observa que os executores em geral compreenderam o mapeamento sugerido pelas heurísticas.

Os resultados obtidos para cada questão da Tabela 5.2 são:

Questão 1: O executor que respondeu “parcialmente” na questão 1 justificou que não é intuitiva a diferença entre características do domínio e tipos de negócio, devido ao

alto nível de abstração. A maioria dos executores julgou que as heurísticas foram mapeadas com facilidade para o modelo de tipos de negócio;

Tabela 5.2: Avaliação das questões referentes às heurísticas propostas

Questão	Sim	Não	Parcialmente
1) As heurísticas de mapeamento de características do domínio em tipos de negócio em componentes são de fácil entendimento?	5	0	1
2) As heurísticas propostas foram facilmente mapeadas para o modelo de tipos de negócio disponibilizado?	4	0	2
3) Em algum momento você desejou criar algum tipo de negócio e a sua criação não era sugerida por nenhuma das heurísticas disponibilizadas?	0	5	1
4) Em algum momento você não aplicou uma heurística, pois julgou que a sua aplicação não seria vantajosa para o modelo de tipos de negócio?	3	3	0
5) Você identificou alguma outra heurística de mapeamento de características conceituais do domínio em tipos de negócio?	2	4	0
6) Os mapeamentos a respeito da opcionalidade (mandatório ou opcional) e variabilidade (ponto de variação, variante e invariante) sugeridos pelas heurísticas são adequados?	5	0	1
7) Os tipos de negócio gerados a partir das heurísticas são úteis numa possível geração de componentes de negócio?	5	1	0
8) Os tipos de negócio gerados a partir das heurísticas são reutilizáveis?	4	0	2

Questão 2: O mesmo executor que respondeu "parcialmente" para a questão 1, respondeu, também, "parcialmente" na questão 2, justificando a dificuldade em entender algumas heurísticas. O outro executor que respondeu "parcialmente" na questão 2, justificou que não previu o modelo de tipos de negócio de maneira adequada. Como esta resposta é vaga, nenhuma análise mais criteriosa pode ser feita com relação a mesma;

Questão 3: A maioria dos executores não necessitou criar algum outro tipo de negócio que não tivesse sido sugerido durante o mapeamento. O executor que respondeu "parcialmente" a esta questão, justificou a necessidade de heurísticas para mapear requisitos funcionais para casos de uso, que não faz parte do objetivo deste estudo de observação, embora tenham sido propostas na Seção 3.4.2;

Questão 4: A metade dos executores em algum momento não utilizou alguma das heurísticas. As justificativas foram variadas. Dois executores julgaram que o domínio não precisaria da característica conceitual, que seria mapeada para um tipo de negócio. Um outro executor justificou uma confusão quanto aos conceitos de variabilidade e opcionalidade, mas não detalhou o problema ocorrido;

Questão 5: A maioria dos executores não identificou outras heurísticas para mapear características conceituais do domínio em tipos de negócio do domínio. As heurísticas sugeridas pelos executores na questão 5 são referentes ao mapeamento de características funcionais em outros artefatos do domínio, as quais não fazem parte deste estudo. No entanto, no início do estudo foi informado que as heurísticas de mapeamento não tratavam das características funcionais, e que havia um outro conjunto de heurísticas específicas para estes tipos de requisitos;

Questão 6: Um dos objetivos das heurísticas é manter as características de opcionalidade e variabilidade das características conceituais do domínio através dos tipos de negócio. Com os resultados obtidos nas questões 6 e 7 se observou a adequação das heurísticas propostas para este fim. O executor que respondeu "parcialmente" na questão 6 sugeriu que o modelo de características gerado através da notação Odyssey-FEX é complexo, e que a notação proposta em (CZARNECKI *et al.*, 2004) é mais simples e intuitiva. No entanto, neste estudo não foi considerada a avaliação da notação Odyssey-FEX. Tal notação foi somente utilizada como ponto de partida para os mapeamentos em avaliação.

Questão 7: O executor que respondeu "parcialmente" à questão 7 da Tabela 5.3 justificou que os relacionamentos não foram suficientes. No entanto, não fica claro se estes relacionamentos citados são referentes ao modelo de tipos de negócio gerado, ou se são referentes aos relacionamentos previstos na notação Odyssey-FEX;

Questão 8: A maioria dos executores considerou que os tipos de negócio gerados são úteis para uma possível reutilização. Dois executores responderam "parcialmente". Um deles registrou que o modelo de tipos que ele gerou foi incompleto, pois não utilizou alguma heurística (e.g. mapeamento de relações de composição, agregação) durante o processo de mapeamento, afetando, portanto, no resultado do modelo de tipos de negócio e, conseqüentemente, na sua reutilização. O outro justificou que naquele momento ele não teria condições de julgar a capacidade de reutilização dos tipos de negócio gerados.

A maioria dos modelos de tipos de negócio obtido pelos executores foi semelhante ao modelo de tipos de negócio esperado (Anexo 8). Alguns executores não utilizaram a heurística para mapeamento do relacionamento alternative para o relacionamento de herança entre os tipos de negócio Curso, Graduação, Pós-Graduação e Extensão. Outras diferenças foram no uso das heurísticas que sugerem os relacionamentos de agregação e composição. A Tabela 5.3 mostra os resultados obtidos

com relação ao apoio computacional oferecido pelo Odyssey para a utilização das heurísticas.

Tabela 5.3: Avaliação do Apoio Computacional para utilização das heurísticas

Questão	Sim	Não	Parcialmente
1) O apoio computacional existente contemplou adequadamente a utilização das heurísticas de mapeamento de características conceituais do domínio em tipos de negócio?	6	0	0
2) Durante a construção dos tipos de negócio foi fácil identificar qual heurística de mapeamento estava sendo empregada?	6	0	0
3) Você utilizou duas ou mais heurísticas para a geração de um único tipo de negócio?	4	2	0
4) As funcionalidades para mapeamento de características conceituais do domínio em tipos de negócio são de fácil utilização?	5	0	1

Neste estudo se observa que o apoio oferecido atendeu às necessidades da grande maioria dos executores do E-OBS-1. Os executores que não utilizaram duas ou mais heurísticas responderam que foi uma estratégia de projeto, sem mais justificativas. O executor que julgou na questão 4 que as funcionalidades eram "parcialmente" fáceis de utilizar justificou a necessidade de conhecimento mais profundo sobre ED para a utilização de tais funcionalidades.

Tendo em vista os resultados registrados nas Tabelas 5.2 e 5.3, se observa que as heurísticas propostas podem ser úteis para o propósito para as quais foram criadas. As heurísticas podem auxiliar na manutenção da consistência das variabilidades, opcionalidades e relacionamentos existentes entre requisitos conceituais de um domínio, através de seus respectivos tipos de negócio.

5.2.2 – Estudo de Observação 2 (E/OBS-2)

5.2.2.1 - Objetivo Global

O objetivo geral deste estudo (E/OBS-2) é avaliar a viabilidade das heurísticas propostas nesta tese para o mapeamento de tipos de negócio em componentes de negócio.

Neste estudo, pretende-se observar se os componentes de negócio obtidos através das heurísticas satisfazem os requisitos especificados no modelo de tipos de negócio do domínio, ou seja, se estas heurísticas de mapeamento são adequadas para atender a

todas as necessidades de mapeamento possíveis de um artefato de análise para um artefato de projeto do domínio.

5.2.2.2 - Objetivo do E/OBS-2

Analisar as heurísticas de mapeamento de tipos de negócio do domínio em componentes de negócio
Com o propósito de caracterizar
Com respeito a utilização das heurísticas propostas
Do ponto de vista do Engenheiro de Software
No contexto de modelagem de componentes de negócio do domínio de Ensino Colaborativo apoiado por Computador

5.2.2.3 - Planejamento do E/OBS-2

5.2.2.3.1 - Seleção do Contexto

O contexto deste estudo de observação é semelhante ao utilizado no E/OBS-1. Como mostra a Seção 5.2.1.3.1, os executores são alunos de ES do PPGCC-FACIN-PUCRS, pré-selecionados por Professores de ES deste Programa.

Também é utilizado o domínio de ensino colaborativo no contexto de Instituições de Ensino Superior. A diferença do E/OBS-1 para o E/OBS-2 é que o segundo (E/OBS-2) utiliza como artefato de análise o modelo de tipos de negócio do domínio, que é esperado como resultado do primeiro (E/OBS-1). Assim, os executores devem analisar o modelo de tipos de negócio do Anexo 9, através do ambiente Odyssey, e as funcionalidades do protótipo que auxiliam na utilização das heurísticas a serem observadas.

5.2.2.3.2 - Seleção dos Indivíduos

Os indivíduos utilizados neste estudo são os mesmos utilizados no E/OBS-1. Além disso, para cada particopante, o E/OBS-2 foi efetuado logo após a conclusão do E/OBS-1. Esta decisão de ordem de execução e escolha de indivíduos para a execução do E/OBS-2 se deve a:

- dificuldade de encontrar uma quantidade de indivíduos que participassem somente de um único estudo de observação;

- conhecimento adquirido sobre a modelagem de variabilidades, através do E/OBS-1, que pode ser aproveitado para a execução deste estudo de observação;

- conhecimento adquirido sobre os requisitos do domínio, uma vez que é utilizado o mesmo domínio em ambos os estudos.

5.2.2.3.3 – Seleção das Variáveis

As variáveis independentes deste estudo são:

- a experiência dos executores sobre DBC;
- o resultado da sua participação no E/OBS-1;
- o modelo de tipos de negócio apresentado.

A variável dependente deste estudo é o modelo de componentes de negócio resultante.

5.2.2.4 - Operação do E/OBS-2

5.2.2.4.1 - Preparação

Nesta etapa, os executores preenchem o Formulário de Consentimento (Anexo 10), e o executor do estudo faz uma breve explanação sobre a notação do modelo de tipos de negócio (Anexo 9).

5.2.2.4.2 - Treinamento das Heurísticas

Foi feita uma explanação pelo observador a respeito das heurísticas a serem observadas neste estudo. Cada heurística foi apresentada, dando um exemplo de utilização, explorando o domínio de telefonia móvel, utilizado no treinamento da notação Odyssey-FEX.

5.2.2.4.3 - Utilização das Heurísticas

Foi apresentado aos executores do estudo, o modelo de tipos de negócio do Anexo 8, desenvolvido no ambiente Odyssey. Tendo como base este modelo, os executores do estudo deviam utilizar as funcionalidades que dão apoio ao mapeamento de tipos de negócio em componentes de negócio, apresentadas na Seção 3.4.3. Ao selecionar esta opção, o executor teve acesso a um conjunto de painéis que o conduziu na utilização das heurísticas e, conseqüentemente, na criação dos componentes de negócio do domínio. Os executores, neste momento, fizeram a análise das necessidades de agrupamento dos tipos de negócio em componentes de negócio que deviam gerenciar estes tipos.

O resultado desta atividade foi a geração de um pacote de componentes de negócio, com base nas heurísticas avaliadas neste estudo de observação. Ao final da atividade de agrupamento de tipos em componentes, os executores preencheram o formulário de avaliação apresentado no Anexo 10, do qual se espera o registro de respostas objetivas (e.g. sim, não, parcialmente) e subjetivas (e.g. justificativas das respostas). O resultado do agrupamento, o qual foi esperado um modelo semelhante ao Anexo 11, e das respostas do formulário de avaliação é que efetivamente são considerados para avaliar a viabilidade das heurísticas.

5.2.2.5 - Avaliação do E/OBS-2

A avaliação do E/OBS-2 foi feita logo após os usuários terem concluída a tarefa de utilização das heurísticas a serem observadas.

Tendo em vista que os executores já tinham desenvolvido as atividades do E/OBS-1, não houve dificuldades para a compreensão das heurísticas e para o agrupamento de componentes de negócio.

Os resultados obtidos neste estudo estão organizados nas Tabelas 5.4 à 5.7 (Anexo 10). Com os resultados da Tabela 5.4, se observa que a maioria dos executores não teve dificuldade no entendimento e utilização das heurísticas para o mapeamento de tipos de negócio em componentes de negócio.

Os resultados obtidos para cada questão da Tabela 5.4 são:

Questão 1: O executor que respondeu "parcialmente" na questão 1 justificou que, no relacionamento de agregação entre tipos de negócio, não havia compreendido a opcionalidade atribuída ao mesmo;

Tabela 5.4: Avaliação geral das heurísticas

Questão	Sim	Não	Parcialmente
1) As heurísticas de agrupamento de tipos de negócio em componentes são de fácil entendimento?	5	0	1
2) As heurísticas de agrupamento de tipos de negócio em componentes foram adequadamente mapeadas sobre o modelo de tipos de negócio disponibilizado?	5	0	1
3) Em algum momento você desejou agrupar um conjunto de tipos de negócio e esta forma de agrupamento não era sugerida por nenhuma das heurísticas disponibilizadas?	2	4	0
4) Em algum momento você não aplicou uma heurística, pois julgou que a sua aplicação não seria vantajosa para o modelo de componentes gerado?	2	4	0
5) Você identifica alguma outra heurística para agrupamento de tipos de negócio em componentes?	0	6	0

Questões 2, 3 e 4: O executor que respondeu "parcialmente" na questão 2 justificou que o modelo de componentes gerado não foi o ideal, mas não sabe bem o motivo em função das escolhas efetuadas. Tal resposta não acrescenta muito neste estudo de observação, pois até esta questão do formulário não estava em avaliação o modelo de componentes gerado, mas sim o processo de reconhecimento das heurísticas. Pôde-se avaliar que este executor não utilizou as heurísticas que avaliam os relacionamentos de agregação e composição para a maioria dos tipos de negócio, podendo ser esta uma justificativa para a insatisfação do executor com relação ao modelo de componentes de negócio gerado;

Questão 5: Nenhum dos executores identificou uma outra heurística para este objetivo, além das heurísticas propostas neste estudo, o que pode vir a ser um aspecto positivo, em relação à completude das heurísticas propostas para mapeamento de tipos em componentes, num contexto de engenharia de domínio.

Através da Tabela 5.5, se observa que as características de obrigatoriedade e opcionalidade dos componentes resultantes, em geral, satisfazem às mesmas características originalmente especificadas durante a análise de requisitos.

Os resultados obtidos para cada questão da Tabela 5.5 são:

Questões 1, 2 e 3: A maioria dos executores concordou com as decisões a respeito de variabilidades e opcionalidades para os tipos de negócio agrupados. O executor que respondeu "parcialmente" é o mesmo que, na questão 2 da Tabela 5.4, justificou a falta de entendimento da heurística que tratava a questão de agregação entre tipos de negócio. O executor justifica que não é claro para ele que ao agrupar tipos de negócio relacionado por agregação, onde o tipo que representa o todo é mandatório e os que representam partes são opcionais, vai resultar num componente mandatório. Este mesmo executor respondeu "parcialmente" na questão 3 da Tabela 5.4, justificando a mesma coisa.

Questão 4: A maioria dos executores concordou pela decisão de não agrupar tipos de negócio com estereótipo <<XOR>>. A justificativa dada pelo executor que respondeu "parcialmente"esta questõa é que a heurística que restringe tal agrupamento é conflitante com aquela que obriga tipos de negócio relacionados por herança serem agrupados. Esta observação vista pelo executor com um aspecto negativo é considerado neste estudo um aspecto positivo, ou seja, o agrupamento de tipos relacionados por

herança só poderá acontecer se os tipos que representam os filhos desta relação não apresentarem estereótipo <<XOR>>.

Tabela 5.5: Avaliação das Características de Opcionalidade e Variabilidade

Questão	Sim	Não	Parcialmente
1) As decisões a respeito da opcionalidade (mandatório ou opcional) e variabilidade (ponto de variação, variante e invariante) sugeridas pelas heurísticas são adequadas?	5	0	1
2) O agrupamento de tipos de negócio mandatórios e opcionais resultarem num componente mandatório foi adequado durante este estudo?	5	0	1
3) A obrigatoriedade de agrupar tipos de negócio que representam pontos de variação e variantes, num componente como um ponto de variação, foi adequado neste estudo?	6	0	0
4) A proibição de agrupamento de tipos de negócio que representam uma exclusão no domínio (XOR) foi adequado durante este estudo?	5	0	1
5) A restrição de agrupar tipos de negócio relacionados por dependência ou associação, se e somente se os tipos de negócio tiverem a mesma característica de opcionalidade (mandatório ou opcional) foi adequado neste estudo?	5	0	1
6) Atribuir ao componente a variabilidade do tipo de negócio que representa o todo numa relação de agregação foi adequado neste estudo?	5	0	1
7) É adequado que tipos de negócio relacionados por composição tenham a mesma característica de opcionalidade (opcional ou mandatório)?	6	0	0

Questão 5: O executor que respondeu "parcialmente" justificou que seria importante a possibilidade de agrupar tipos de negócio relacionados por associação ou dependência e que fossem opcionais. Neste caso, se acredita que houve falta de entendimento do executor em relação à pergunta efetuada ou quanto à funcionalidade proposta.

Questão 6: O executor que respondeu "parcialmente" na questão 6 justifica que deveria ser possível tornar opcional um componente que fosse resultado de tipos de negócio mandatórios. No entanto, se observa que tal solicitação pode infringir a consistência dos requisitos mandatórios e opcionais especificados no modelo de características do domínio. Para o relacionamento de composição foi unânime a informação de que o componente de negócio, gerado a partir de tipos relacionados desta forma, recebesse a característica de obrigatoriedade de seus tipos envolvidos.

A Tabela 5.6 mostra a satisfação dos executores do experimento a respeito dos componentes obtidos pelo uso das heurísticas, bem como o potencial de reutilização destes componentes para a construção de uma aplicação derivada deste domínio.

Os resultados obtidos para cada questão da Tabela 5.6 são:

Questão 1: O executor que respondeu "parcialmente" na questão 1 justificou que a granularidade dos componentes ainda é bastante alta para reutilização, caso o negócio demande um comportamento específico.

Questão 2: O executor que respondeu "não" na questão 2 justificou que, para linha de produtos, é necessário definir melhor o comportamento para a sua reutilização em aplicações específicas. O executor que respondeu "parcialmente" não se sentiu capaz de julgar a possível reutilização destes componentes.

Tabela 5.6: Componentes gerados a partir das heurísticas

Questão	Sim	Não	Parcialmente
1) Os componentes gerados a partir das heurísticas são úteis numa possível arquitetura de componentes?	5	0	1
2) Os componentes gerados a partir das heurísticas são reutilizáveis no contexto de uma arquitetura de referência para um domínio?	4	1	1

Conforme a Tabela 5.7, o ferramental desenvolvido para apoio ao agrupamento de componentes em tipos de negócio é satisfatório, com funcionalidades que facilitam a identificação das heurísticas que estão em análise e, sobretudo, a utilização destas funcionalidades para a geração dos componentes de negócio.

O único executor que respondeu "parcialmente" na questão 2 justificou que não havia percebido que ao sugerir o agrupamento de um tipo de negócio a outro, através do painel de apoio a criação dos componentes, o primeiro tipo de negócio estaria associado ao segundo (e.x: se agrupar o componente Avaliação com o componente Disciplina, Avaliação não será mais criado independentemente de disciplina). Dentre as justificativas para esta questão são: a) a falta de clareza da interface com relação a esta mudança de estado da árvore de tipos de negócio no assistente de agrupamento, ou b) a falta de um treinamento mais específico para estas questões.

Tabela 5.7: Apoio computacional para as heurísticas

Questão	Sim	Não	Parcialmente
1) O apoio computacional existente contemplou adequadamente as heurísticas de agrupamento dos tipos de negócio em componentes?	6	0	0
2) Durante a construção dos componentes foi fácil identificar qual heurística de agrupamento estava sendo empregada?	5	0	1
3) As funcionalidades para agrupamento de tipos de negócio em componentes são de fácil utilização?	6	0	0

Tendo em vista os resultados registrados nas Tabelas 5.4, 5.5, 5.6 e 5.7, se observa que estas heurísticas, assim como as avaliadas no E/OBS-1, também podem ser adequadas para o propósito para as quais foram criadas. As heurísticas podem auxiliar o Engenheiro de Domínio na manutenção da consistência das variabilidades, opcionalidades e relacionamentos, provenientes de seus respectivos tipos de negócio. Assim, pode ser viável a manutenção destas características nas futuras versões de uma AR do domínio, juntamente com outros tipos de componentes (e.g. processo, utilitários e de infra-estrutura).

5.2.3 – Estudo de Observação 3 (E/OBS-3)

5.2.3.1 - Objetivo Global

O objetivo geral deste estudo (E/OBS-3) é avaliar a viabilidade dos critérios de agrupamento de componentes do domínio em elementos arquiteturais, visando estruturar tais componentes numa AR do domínio.

5.2.3.2 - Objetivo do E/OBS-3

Analisar os critérios de agrupamento de componentes do domínio
Com o propósito de caracterizar
Com respeito à utilização dos critérios propostos
Do ponto de vista do Engenheiro de Software
No contexto de alunos do PPGCC-FACIN-PUCRS
No contexto de geração de elementos arquiteturais do domínio de Ensino Colaborativo apoiado por Computador

5.2.3.3 - Planejamento do E/OBS-3

5.2.3.3.1 - Seleção do Contexto

O contexto deste estudo de observação é semelhante ao utilizado nos estudos E/OBS-1 e E/OBS-2. A justificativa deste contexto é a mesma apresentada para o E/OBS-2, na Seção 5.2.2.3.1.

5.2.3.3.2 - Seleção dos Indivíduos

Os indivíduos utilizados neste estudo de observação são os mesmos utilizados no E/OBS-1 e E/OBS-2. Além disso, o E/OBS-3 foi efetuado logo após a conclusão do

E/OBS-2. Os motivos que levaram a decisão dos indivíduos e do momento de execução deste estudo são os mesmos apresentados para o E/OBS-2, na Seção 5.2.2.3.2.

5.2.3.3.3 – Seleção das Variáveis

As variáveis independentes deste estudo são:

- a experiência dos executores sobre DBC;
- o modelo de componentes do domínio apresentado;

A variável dependente deste estudo é o modelo de componentes do domínio resultante.

5.2.3.4 - Operação do E/OBS-3

5.2.3.4.1 - Preparação

Nesta etapa, os executores preenchem o Formulário de Consentimento (Anexo 14), e o executor do estudo faz uma breve explanação sobre a notação do modelo de componentes do domínio (Anexo 13) e sobre o funcionamento da ferramenta GroupCriteria que será utilizada no E/OBS-3.

5.2.3.4.2 –Treinamento dos Critérios

Durante a apresentação da ferramenta, foi discutido o propósito de cada critério sugerido. Neste contexto, foi apresentado, passo a passo, as atividades que os executores deveriam executar, no que diz respeito às configurações dos dados, a obtenção das informações sobre agrupamentos sugeridos e a como eles poderiam selecionar agrupamentos, gerar combinações de agrupamentos e eliminar redundâncias de agrupamentos sugeridos.

5.2.3.4.3 - Utilização dos Critérios

O processo de Aplicação dos Critérios envolveu as seguintes atividades:

a) conhecimento do modelo de contextos e de componentes do domínio;

b) acesso à ferramenta de agrupamento de componentes;

c) seleção dos critérios para a obtenção de sugestões de agrupamento. Neste estudo, foi indicada pelo observador do estudo a seleção de todos os critérios;

d) determinação de limite de sugestões de agrupamento para cada critério escolhido. A quantidade foi estabelecida pelos executores, individualmente;

e) seleção dos tipos de componentes que devem ser levados em consideração pelos critérios de agrupamento escolhidos (negócio, processo, utilitário e de infra-estrutura).

Em seguida, foi iniciada a atividade de análise dos agrupamentos sugeridos. Nesta atividade, o executor escolheu dentre os agrupamentos sugeridos por cada critério de agrupamento e solicitou e criação do componente para este fim.

O resultado desta atividade é que efetivamente foi considerado para avaliar a utilização dos critérios para agrupamento de componentes, que é o propósito deste estudo. Para auxiliar tal observação, os executores deverão preencher o formulário de avaliação apresentado no Anexo 15.

5.2.3.5 - Avaliação do E/OBS-3

Pelo fato dos executores terem passado pelos estudos E/OBS-1 e E/OBS-2, não se percebeu muitas dificuldades na compreensão da notação do modelo de componentes, bem como da proposta de cada critério de agrupamento de componentes. Os resultados obtidos neste estudo estão organizados nas Tabelas 5.8, 5.9 e 5.10.

Com os resultados da Tabela 5.8, se observa que os executores tiveram facilidade para entender a proposta de cada critério de agrupamento e a forma de configuração dos critérios para a análise das sugestões de agrupamentos de componentes.

Tabela 5.8: Avaliação das configurações da ferramenta

Questão	Sim	Não	Parcialmente
1) Você já utilizou alguma ferramenta de agrupamento de componentes?	0	6	0
2) As configurações previamente necessárias para a execução dos critérios de agrupamentos são de fácil entendimento?	6	0	0
3) Os critérios disponibilizados na ferramenta refletem as necessidades de agrupamento de componentes?	6	0	0
4) Você identifica outros critérios de agrupamento de componentes que não foram sugeridos pela ferramenta?	0	6	0
5) Os limites de número máximo de agrupamento para cada critério são adequados?	6	0	0
6) As sugestões de agrupamento de cada critério são úteis?	5	0	1

Em geral, os executores selecionaram todos os critérios de agrupamento e todos os tipos de componentes para serem avaliados. O executor que considerou as sugestões de agrupamento "parcialmente" úteis registrou que o acoplamento de componentes por

contextos do domínio é demasiadamente abrangente, não favorecendo a construção de uma arquitetura para reutilização.

Através da Tabela 5.9, se observa que houve um empate com relação à compreensão de cada critério, caso não houvesse o auxílio da ferramenta. Dois executores registraram que seria difícil e outros dois julgaram "parcialmente" difícil encontrar as sugestões de agrupamento diretamente no modelo de componentes apresentado no Anexo 12. Os que registraram "parcialmente" ou "sim" relatam em geral que a ferramenta auxilia, mas que não é difícil localizar as sugestões de acoplamento.

Um dos executores registrou que a ferramenta pode vir a ser útil para modelos de componentes complexos e maiores do que o modelo utilizado no estudo. Os recursos para manipulação, escolha e consistência dos agrupamentos escolhidos foram plenamente aceitos pelos executores.

Embora os executores não tenham se motivado a reconstruir toda a arquitetura de componentes original, utilizando os componentes de agrupamento, pelos ensaios efetuados, eles puderam perceber que o modelo de componentes poderia ser mais compreensível se comparado ao modelo original.

Tabela 5.9: Aplicação dos Critérios de Agrupamento de Componentes através de Apoio Computacional

Questão	Sim	Não	Parcialmente
1) As sugestões de agrupamento de cada critério seriam facilmente identificadas no modelo de componentes do domínio caso não houvesse o auxílio da ferramenta?	2	2	2
2) Você identifica a necessidade de outros agrupamentos de componentes e tais agrupamentos não foram sugeridos pelos critérios disponíveis?	0	5	1
3) A forma de apresentação dos resultados dos critérios e dos agrupamentos identificados por critério é adequada?	6	0	0
4) Durante o processo de geração dos agrupamentos, foi adequado o recurso de abas para identificação dos grupos de componentes separados por critério?	6	0	0
5) O aviso de agrupamento de componentes redundantes, apresentado durante o processo de geração dos agrupamentos, facilitou a criação de agrupamentos mais adequados?	6	0	0
6) A forma de apresentação dos agrupamentos de componentes redundantes é adequada?	6	0	0

Em geral, como mostra a Tabela 5.10, os executores julgaram que os componentes agrupados poderiam ser reutilizados em aplicações derivadas no domínio exemplificado.

Tabela 5.10: Resultado da aplicação dos Critérios de Agrupamento de Componentes

Questão	Sim	Não	Parcialmente
1) O modelo de componentes gerado é mais compreensível quando comparado ao modelo de componentes recebido neste estudo de caso?	5	0	1
2) Os agrupamentos sugeridos podem ser reutilizados em aplicações derivadas deste domínio?	5	0	1
3) Você sentiu necessidade de modificar alguma sugestão de agrupamento de componentes (incluir ou remover algum componente)?	1	4	1
4) Você sentiu necessidade de combinar duas ou mais sugestões de agrupamento de componentes para que os componentes formassem um único agrupamento?	0	5	1
5) A ferramenta é de fácil utilização?	5	0	1

Os resultados obtidos para cada questão da Tabela 5.10 são:

Questão 1: O executor que respondeu "parcialmente" na questão 1 registrou que o modelo pode ser mais compreensível, porém não se garante a sua completude. Neste caso, tal relato não é uma preocupação desta avaliação, pois não é o objetivo dos critérios tornar a arquitetura mais completa, e sim, melhor estruturada.

Questão 2: O executor que respondeu "parcialmente" na questão 2 justificou que a reutilização destes agrupamentos vai depender dos tipos de agrupamento, e da forma como se pretende criar novas arquiteturas de aplicação a partir deste domínio.

Questão 3: Na maioria das vezes, os agrupamentos sugeridos não precisavam ser modificados, conforme se observa na questão 3. Os executores que responderam sim e "parcialmente", em geral, não julgaram que, ao retirar um componente do agrupamento escolhido, poderia melhorar o resultado do agrupamento em termos de acoplamento do componente e de sua futura reutilização.

Questão 4: Embora um executor tenha registrado que sentiu necessidade de combinar duas sugestões de agrupamento, ele não utilizou tal recurso na ferramenta, nem justificou o motivo da combinação esperada.

Questão 5: O único executor que registrou "parcialmente" na questão 5 alegou que existiam novos conceitos de abstração que dificultaram a utilização da ferramenta. No entanto, somente com esta breve justificativa, é difícil avaliar as reais dificuldades deste executor com relação à utilização da ferramenta. Durante a execução da atividade, não foi diagnosticado pelo observador alguma dificuldade aparente.

Com relação à avaliação dos elementos arquiteturais gerados, se observou que todos eles foram considerados corretos, pois os seus respectivos componentes estavam

relacionados através de suas interfaces ou pertenciam a um mesmo contexto do domínio. Elementos arquiteturais só seriam considerados incorretos se agrupassem componentes que não pertencessem a um mesmo contexto, ou que não tivessem interfaces providas ou requeridas entre si.

Alguns executores tiveram a preocupação de não agrupar componentes que são requeridos por muitos outros componentes num único elemento arquitetural. Dessa forma, este componente bastante requerido, compõe por si só um elemento arquitetural do domínio, o qual é requerido por muitos outros elementos arquiteturais. Outros executores que não tiveram este cuidado, perceberam tal sobrecarga, tomando a iniciativa de reestrututração de seus elementos arquiteturais.

Tendo em vista os resultados registrados nas tabelas anteriores se observa que estes critérios de acoplamento de componentes são adequados para o propósito para os quais foram criados. Tais critérios avaliam a interação existente entre os componentes e outras características referentes à semântica do domínio que sugerem agrupamentos que fazem sentido para a arquitetura do domínio em construção.

5.3 – Análise dos dados dos Estudos de Observação

As respostas objetivas (e.g. sim, não, parcialmente) e as interpretações atribuídas a cada avaliação dos estudos de observação indicam indícios da utilidade das heurísticas. No entanto, uma série de fatores ocorridos durante os mesmos permite-nos afirmar que estes resultados não são totalmente confiantes. Neste sentido, os estudos podem apresentar algum indicativo de viabilidade, mas não confirmam a sua existência. Os fatores que limitam a validade destes estudos são:

1) Caracterização da População: Estudos de observação num contexto de ED exigem que os executores, além de terem formação em Engenharia de Software e Engenharia de Domínio, que também sejam especialistas na Engenharia daquele domínio específico. Nestes estudos de observação, apesar da maioria dos executores já terem contato com o domínio de observação (CSCL), eles não têm a experiência de Engenheiros deste domínio.

2) Repetição dos Executores nos três Estudos: O fato dos executores serem os mesmos para os três estudos de observação, pode ter gerado resultados tendenciosos nos estudos de observação E/OBS-2 e E/OBS-3. Um executor que, por exemplo, ficou insatisfeito com o resultado obtido no estudo anterior, pode ter efetuado as tarefas dos demais estudos de forma pouco comprometida e dependente dos resultados anteriores.

Neste sentido, os modelos gerados em cada estudo podem não refletir a realidade do entendimento das heurísticas e critérios avaliados.

3) Treinamentos dos Critérios e Heurísticas: Os executores deveriam ter sido treinados a utilizar as heurísticas e critérios antes de dar início ao estudo de observação, e não como uma apresentação prévia ao estudo.

Conforme descrito anteriormente, no E/OBS-1, os executores receberam, previamente, por e-mail, uma documentação sobre a notação Odysse-FEX. No entanto, não se pode garantir a dedicação de cada executor na leitura do material e, sobretudo, o entendimento real dos conceitos desta notação para a execução das tarefas do E/OBS-1. Todos relataram não terem dificuldades na leitura e compreensão do material, mas nenhuma atividade foi feita de maneira a se ter garantias deste entendimento.

Em todos os estudos de observação, o observador reservou 10 minutos para a explicação de cada heurística a ser avaliada e sobre o funcionamento da ferramenta utilizada.

Da forma como ocorreu, não se pode garantir que os executores tenham entendido a contento as heurísticas e critérios a serem observados. Como os estudos foram efetuados em seqüência, pelas mesmas pessoas, a compreensão das heurísticas e critérios pode ter ficado ainda mais comprometido, agravado pelo acúmulo de informações e atividades (execução da atividade de observação, preenchimento de formulários) num período de, aproximadamente, 3 horas de duração.

4) Comparativo dos resultados dos Estudos de Observação: Ainda que todos os problemas identificados acima não tivessem ocorrido nestes estudos, não haveria uma forma de comparar o resultado dos mapeamentos sugeridos através das heurísticas e critérios avaliados. Neste sentido, os executores de cada estudo de observação deveriam ser separados em dois grupos. Um grupo, utilizando um processo *ad-hoc* para gerar o modelo esperado pelos estudos, em um segundo grupo, utilizando as heurísticas/critérios disponíveis no ambiente Odyssey. Os resultados obtidos seriam comparados para avaliar o quanto que os executores, com e sem o uso das heurísticas/critérios, chegaram próximos aos modelos esperados pelo observador.

5) Tempo de Execução dos Estudos: Embora não tenham sido objetos de avaliação destes estudos de observação, foram computados os tempos gastos por cada executor durante a execução da atividade. Estes dados, aliados as lições aprendidas até então descritas, são reflexo das limitações destes estudos de observação.

As Tabelas que compõem a Tabela 5.11, mostram os tempos, em minutos, que cada executor levou para executar cada estudo, sem contar com o treinamento e com o preenchimento de formulários de consentimento, caracterização e avaliação.

Através destas tabelas, se observa que houve uma diferença muito grande no tempo de execução das atividades entre executores de cada estudo, gerando médias (E-OBS-1 = 28, E-OBS-2 = 17 e E-OBS-3 = 19) que não refletem resultados satisfatórios devido ao valor elevado do desvio padrão em cada estudo (E-OBS-1 = 9.7, E-OBS-2 = 5.8 e E-OBS-3 = 7.0).

Tabela 5.11: Tabelas com os Tempos de Execução dos Estudos de Observação

E-OBS-1	
Executor	Tempo
1	35
2	43
3	20
4	25
5	30
6	17
Média	28
Desvio Padrão	9.7

E-OBS-2	
Executor	Tempo
1	17
2	14
3	25
4	9
5	15
6	22
Média	17
Desvio Padrão	5.8

E-OBS-3	
Executor	Tempo
1	32
2	20
3	20
4	15
5	12
6	16
Média	19
Desvio Padrão	7.0

Graficamente, ao aplicar uma técnica de dispersão, pode-se visualizar, através das Figuras 5.1, 5.2 e 5.3 as diferenças nos tempos de execução, de cada executor, em cada estudo. Se as médias refletissem uma realidade dos tempos de execução, os gráficos deveriam ter montado algo aproximado a uma reta, próximo ao valor destas médias. Ao contrário, se observa que os gráficos montaram curvas que denotam a dispersão entre os tempos de execução de cada executor nos diferentes estudos.

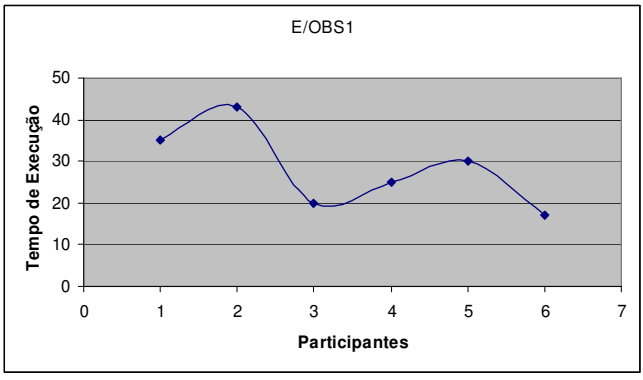


Figura 5.1: Tempo de Execução dos Executores no E/OBS-1

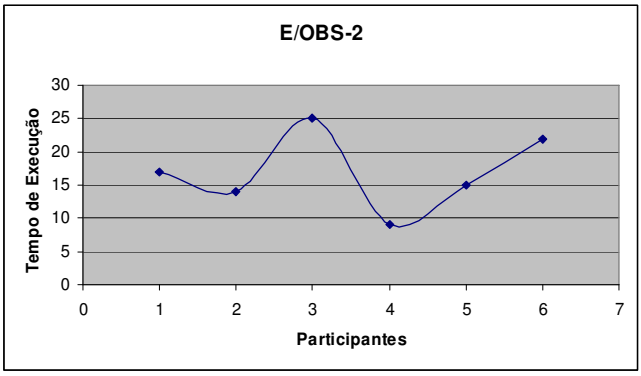


Figura 5.2: Tempo de Execução dos Executores no E/OBS-2

Embora o E/OBS-2 tenham apresentado um desvio padrão menor do que os demais, o gráfico resultante não denota indícios de aproximação entre os tempos de execução dos executores. Já o gráfico referente ao E/OBS-3 mostra uma curva não tão sinuosa, embora ainda exista uma distância considerável entre os tempos de execução dos estudos (maior – 32 minutos e menor – 12 minutos).

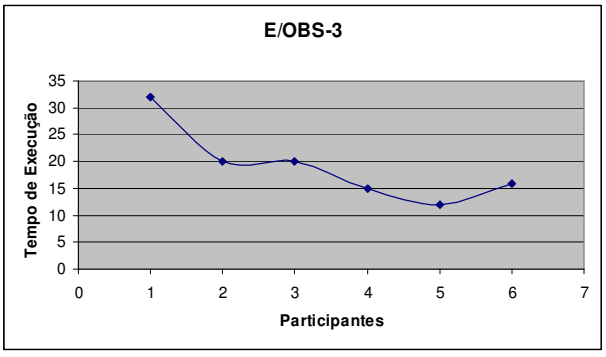


Figura 5.3: Tempo de Execução dos Executores no E/OBS-3

Outro fato a ser observado na Tabela 5.11, é que um mesmo executor ora está dentro da média de tempo de execução de um estudo, ora está fora. Este dado pode confirmar a necessidade de executores diferentes para os estudos de observação, visando aumentar o comprometimento do indivíduo para um estudo específico e diminuir a fadiga em função do grau elevado de informações e atividades.

As informações da Tabela 5.12 confirmam esta interpretação. Por exemplo, o executor 2 obteve um desvio padrão muito elevado, pois levou muito tempo no E/OBS-1 e para os demais estudos diminuiu consideravelmente o seu tempo de execução. Pelas justificativas dadas por este executor no formulário de avaliação, esta diminuição de

tempo não foi em função da facilidade de compreensão dos demais estudos, mas provavelmente pela falta de entendimento ou comprometimento com estes.

Por outro lado, o executor 3 apresenta uma constância na média de tempo de execução que corresponde aos três estudos de observação (E-OBS-1 =20, E-OBS-2 = 25 e E-OBS-3 = 20), embora nem sempre corresponda as médias de cada estudo (E-OBS-1 = 28, E-OBS-2 = 17 e E-OBS-3 = 19).

Tendo em vista estes fatores limitadores dos estudos de observação, é indicada a reestruturação destes estudos de forma que:

- possa ser avaliada a utilização de todas as atividades do processo CBD-Arch-DE;

- o estudo possa ser comparado com um processo *ad-hoc*, onde os mesmos executores utilizam o processo CBD-Arch-DE e *ad-hoc*;

- o estudo seja executado por Engenheiros de Domínio especialistas em um domínio específico e,

- para cada atividade do processo CBD-Arch-DE, possa ser feito um treinamento com os executores.

Tabela 5.12: Média de Tempo e Desvio Padrão de cada Executor

Executor	Média de Tempo	Desvio Padrão
1	28	9.6
2	26	15.3
3	22	2.9
4	16	8.1
5	19	9.6
6	18	3.2

5.4 – Considerações Finais

Este capítulo fez uma breve introdução à atividade de experimentação em Engenharia de Software, destacando alguns benefícios desta área para a avaliação de produtos, processos e técnicas propostas.

Em seguida, foram apresentados três estudos de observação definidos e planejados para caracterizar a viabilidade de heurísticas para mapeamento de artefatos do domínio e critérios de agrupamentos de componentes.

Embora as avaliações dos estudos apresentem resultados que indiquem a viabilidade das heurísticas e critérios avaliados, os fatores limitadores destacados na Seção 5.3 sugerem que os estudos apresentam, apenas, indícios de viabilidade.

Ao final da Seção 5.3, é sugerido uma estratégia a ser refletida para a elaboração do planejamento e execução de novos estudos de observação. Nesta estratégia, é indicada a necessidade de avaliação de todas as atividades do processo CBD-Arch-DE, e não somente as atividades escolhidas para os estudos de observação desenvolvidos, considerado uma situação real.

Capítulo 6 – Conclusão

6.1 – Considerações Gerais e Contribuições

Conforme discutido nos Capítulos 1 e 2, existem algumas iniciativas nas áreas de ED e DBC com o objetivo de promover a reutilização de artefatos de software. Embora métodos de ED e DBC apresentem contribuições importantes para apoio à reutilização, o apoio oferecido até o momento é ainda limitado a algumas atividades existentes num ciclo de vida de um software ou domínio de aplicações. Tais restrições influenciam negativamente na reutilização dos modelos produzidos.

Esta Tese representa mais um passo na especificação de aspectos que promovem a reutilização, através da abordagem DECOM.

Conforme analisado no Capítulo 2, nenhum dos métodos de ED e DBC (KANG *et al.*, 1990; D'SOUZA & WILLS, 1999; BRAGA, 2000; CHEESMAN & DANIELS, 2000; ATKINSON *et al.*, 2002; KANG *et al.*, 2002) apresenta uma proposta integrada, com apoio semelhante ao oferecido pela abordagem DECOM, descrita nesta Tese, e que atenda a todas as fases do ciclo de vida.

Dentre as abordagens analisadas, Odyssey-DE é a que mais atende aos requisitos esperados por um processo de ED. No entanto, Odyssey-DE possui as seguintes limitações: a) falta de apoio à modelagem de variabilidades e opcionalidades em outros artefatos do domínio, além do modelo de características; b) falta de apoio ao mapeamento das propriedades existentes no modelo de características em outros modelos de domínio; c) falta de apoio a organização dos componentes modelados no domínio na construção de arquiteturas de referência; d) falta de apoio à fase de implementação dos artefatos do domínio.

O objetivo desta Tese é aproveitar as vantagens providas pelas áreas de ED e DBC e minimizar as limitações apresentadas no capítulo 2, obtendo como resultado: a modelagem de variabilidades e opcionalidades em todo o ciclo de vida do domínio; a utilização de critérios para a identificação de elementos arquiteturais; a especificação de AR para domínios de aplicações; e a possibilidade de geração de modelos para uma tecnologia de componentes específica, a partir de modelos de componentes do domínio.

As contribuições desta pesquisa, detalhadas no Capítulo 3, são:

a) *A Especificação de um Processo de ED - CBD-Arch-DE:* Tal processo foi criado visando atender, através de suas fases, aos requisitos que serviram de base para avaliação dos métodos de ED e DBC, conforme apresentado nas Seções 2.5 e 2.6. Embora aparentemente o processo tenha fases idênticas a outros processos de ED (e.g. Odyssey-DE), o seu detalhamento através de atividades tem foco no apoio necessário para atender aos requisitos definidos.

b) *A Extensão das Notações de Casos de Uso, Tipos de Negócio e Componentes:* Através destas extensões é possível uma representação mais consistente das variabilidades, opcionalidades e relacionamentos obtidos a partir do modelo de características do domínio;

c) *Heurísticas para apoiar o mapeamento de variabilidades, opcionalidades, relacionamentos e regras de composição entre artefatos de análise (e.g. características conceituais e tipos de negócio) e de projeto (e.g. tipos de negócio e componentes de negócio).* A representação de variabilidades em um modelo de domínio deve ser coerente com os requisitos do domínio que ele está representando, bem como entre modelos do domínio que representam um mesmo conjunto de requisitos, porém em diferentes níveis de abstração. Estas heurísticas visam auxiliar o Engenheiro de Domínio na manutenção de tais coerências entre artefatos de análise e projeto do domínio. Com base nestas heurísticas e utilizando a notação estendida de tipos de negócio, casos de uso e componentes, o Engenheiro pode obter melhores resultados na geração dos elementos arquiteturais, na utilização destes elementos na construção das arquiteturas de referência e, na reutilização destes elementos durante a Engenharia de Aplicação.

d) *A definição dos critérios de agrupamento de componentes para a identificação de elementos arquiteturais do domínio*. Estes critérios de agrupamento permitem que elementos arquiteturais do domínio sejam identificados para minimizar a complexidade de estruturação de componentes que pertencem a uma mesma AR. Estes elementos arquiteturais minimizam o número de trocas de mensagens existentes entre componentes de uma arquitetura, uma vez que as mensagens trocadas, através de suas interfaces providas e requeridas, serão veiculadas através do próprio elemento arquitetural. Somente as interfaces requeridas ou fornecidas de um outro elemento arquitetural serão disponibilizadas externamente ao elemento arquitetural, através de sua(s) porta(s).

Embora não tenham sido desenvolvidas nesta Tese, outras contribuições relacionadas a esta pesquisa, são:

c) *a Notação Odyssey-FEX*, especificada na Dissertação de Mestrado de Oliveira (2006). Odyssey-FEX contempla uma semântica rica para a representação dos requisitos do domínio. Através desta notação é possível representar, graficamente ou não, os conceitos de variabilidades e opcionalidades existentes nos requisitos do domínio. As regras para representar dependência e exclusão mútua dão mais expressividade ao modelo de características. A semântica gerada através desta notação facilita a atividade de recorte destas características para a geração de aplicações derivadas do domínio. A partir da notação Odyssey-FEX, esta tese estende as notações de tipos de negócio, casos de uso e componentes para também representar a semântica necessária na modelagem de variabilidades, opcionalidades, relacionamento e regras de composição inclusiva e de mútua exclusão, provenientes do modelo de características do domínio;

e) a *abordagem Odyssey-MDA*, especificada na Dissertação de Mestrado de Maia (2006). O projeto baseado em componentes no CBD-Arch-DE orienta o Engenheiro de Domínio na geração de modelos de componentes sem qualquer vínculo com uma tecnologia de componentes específica. Tal independência permite aumentar o potencial de reutilização destes modelos para diferentes aplicações. Quando há a necessidade de implementação destes componentes, a abordagem Odyssey-MDA pode auxiliar no mapeamento destes componentes para uma tecnologia específica. Assim, os componentes no nível de projeto continuam independentes de tecnologia específica, ficando sob responsabilidade da abordagem Odyssey-MDA, na fase de implementação do domínio, as transformações necessárias de componentes implementacionais para uma tecnologia em particular. Além disso, a abordagem permite que o implementador crie as suas próprias transformações, para a sua tecnologia específica, através de um arquivo XML.

Conforme ressaltado no Capítulo 1, esta Tese faz parte do conjunto de pesquisas relacionadas ao Projeto Integrado do CNPq denominado REUSE: Técnicas e Ferramentas de apoio à Reutilização de Software (WERNER, 2005). Neste projeto, o ambiente Odyssey é utilizado como uma ferramenta integradora, visando o apoio à reutilização baseada em modelos de domínio. Neste sentido, uma das preocupações desta Tese é que para todos os aspectos propostos houvesse um apoio minimamente automatizado através de protótipos. Para tal, algumas ferramentas foram desenvolvidas na forma de *plug-ins* do ambiente Odyssey e outras no *kernel* deste ambiente,

disponíveis através de funcionalidades. As funcionalidades desenvolvidas no kernel do Odyssey são:

- a modelagem do processo CBD-Arch-DE através da máquina de processo Charon (MURTA *et al.*, 2002);
- as heurísticas para o mapeamento de características modeladas durante a ED em tipos de negócio e casos de uso e;
- as heurísticas para a geração de componentes de negócio, processo, utilitários e infra-estrutura, com base nos artefatos de análise de domínio.

Adicionalmente, nesta Tese, foi desenvolvida a ferramenta Group Criteria, para a identificação e geração de elementos arquiteturais de um domínio.

Dentre as ferramentas já existentes, foi estendida a Odyssey-Patterns para dar apoio a instanciação de padrões a partir de componentes especificados no domínio.

Além das funcionalidades e produtos mencionados, os estudos de observação, embora com as restrições mencionadas no Capítulo 5, fazem parte das contribuições esta Tese.

Nestes estudos de observação foram exploradas algumas das contribuições desta Tese, utilizando o ambiente Odyssey. No primeiro estudo de observação, foram utilizadas a Notação Odyssey-FEX e as heurísticas para apoio ao mapeamento de características em tipos de negócio. No segundo estudo de observação, foram utilizadas as funcionalidades para apoio a geração de componentes de negócio, explorando as heurísticas para apoio ao mapeamento de tipos de negócio em componentes de negócio. Por fim, no terceiro estudo de observação, foi utilizado o plug-in Group Criteria para apoio à identificação e geração de elementos arquiteturais do domínio.

6.2 - Limitações e Trabalhos Futuros

Algumas limitações e possíveis extensões puderam ser detectadas ao longo dessa pesquisa, as quais esperamos tratar em trabalhos futuros.

6.2.1 Mapeamento de Relacionamento entre Características de diferentes Categorias

A primeira limitação é que o conjunto de heurísticas propostas não considera, no modelo de características, o mapeamento de uma categoria para outra. Por exemplo, se uma característica funcional está relacionada a uma característica conceitual, tal relação não é mapeada para quaisquer das heurísticas propostas nesta Tese. Um outro exemplo,

se uma característica funcional possui um relacionamento de "implementado por" com uma característica tecnológica, tal relacionamento não é mapeado pelas heurísticas propostas. Por esta razão, este conjunto de heurísticas existente deve ser estendido para atender o mapeamento das relações entre categorias de características distintas. Através destas novas heurísticas, a detecção das relações existentes entre componentes de negócio, processo, utilitários e de infra-estrutura se torna mais consistente e completa.

6.2.2 - Critérios para Identificação de Elementos Arquiteturais

Também podem ser identificadas limitações quanto ao conjunto de critérios de agrupamentos de componentes para a identificação de elementos arquiteturais. Em princípio, estes critérios estão considerando principalmente as trocas de mensagens entre componentes do domínio. No entanto, apesar da expressividade dos critérios já existentes, outras informações sobre a semântica do domínio poderiam ser utilizadas como critério para o agrupamento de componentes. Um exemplo de informação semântica é a ligação existente entre os artefatos de um domínio, o qual é gerado durante a sua modelagem. Se dois ou mais componentes de um domínio estão relacionados a uma mesma característica do domínio, então esta relação pode ser um indicador para agrupar tais componentes num único elemento arquitetural.

6.2.3 - Arquiteturas de Referência: Estilos Arquiteturais, ADLs e Avaliação

Embora o processo oriente a atividade de construção de arquiteturas de referência, muitas contribuições ainda poderiam ser dadas a esta área de pesquisa.

Deve ser feita uma análise mais profunda sobre os estilos arquiteturais geralmente utilizados, os quais são relatados em (SHAW & GARLAN, 1996). Tal análise visa observar suas possíveis contribuições para arquiteturas baseadas em componentes. Embora seja um consenso que aplicações e arquiteturas de referência baseada em componentes adotem o estilo arquitetural em camadas, acredita-se que tal informação não deva ser tratada como um padrão da área. Uma das restrições do estilo em camadas é quanto à baixa escalabilidade das aplicações assim estruturadas. Neste caso, se a alta escalabilidade for um requisito a ser atendido pela arquitetura de uma dada aplicação ou domínio de aplicações, então outras formas de estruturação de componentes devem ser analisadas.

A avaliação de requisitos não-funcionais está geralmente inserida no contexto da linha de pesquisa denominada análise de arquiteturas de software. A literatura apresenta

métodos para a análise de arquiteturas de software, com destaque ao SAAM (*Software Architecture Analysis Method*) (KAZMAN *et al.*, 1994), ATAM (*Architecture Tradeoff Analysis Method*) (KAZMAN *et al.*, 1999) e ARID (*Active Reviews for Intermediate Designs*) (CLEMENTS *et al.*, 2002b). Tais métodos devem ser estudados com o intuito de avaliar a sua adequação para a avaliação de arquiteturas de referência em ED.

Uma outra possibilidade de contribuição na área de arquitetura de software, e que não foi explorada nesta Tese, são as ADLs. No Capítulo 2, foram indicadas ADLs que se propunham a formalizar a descrição arquitetural de aplicações baseadas em componentes, utilizando um estilo arquitetural específico (e.g. xADL utilizando o estilo arquitetural C2). Através da análise desta literatura, se observou que o uso de ADLs pode fortalecer, e sobretudo formalizar, a semântica de uma AR. Neste sentido, deve ser retomado o estudo de ADLs para o contexto de DBC, com o intuito de avaliar a sua adequação à formalização de arquiteturas de referência baseadas em componentes. ADLs podem ser consideradas como uma forma de documentação de arquiteturas de software, através da qual podem também ser feitos estudos para a avaliação e comparação de arquiteturas (CLEMENTS *et al.*, 2002a).

6.2.4 - Adaptação e Composição de Componentes

Através do CBD-Arch-DE foram gerados diferentes tipos de componentes. os quais proviam e requeriam interfaces, sem quaisquer problemas de incompatibilidades (e.g. número de e nome de métodos, número, nome e retorno de seus parâmetros) detectados. Durante a especificação do CBD-Arch-DE, se analisou a possibilidade de utilizar uma técnica para a identificação de incompatibilidades entre componentes, propondo uma forma de adaptá-los e compô-los através de um terceiro componente adaptador.

Foi incorporado ao Odyssey, sob a forma de *plug-in*, uma ferramenta de adaptação e composição de componentes, denominada Adapt-COM. Esta ferramenta foi desenvolvida através de uma Dissertação no PPGCC-FACIN-PUCRS (SPAGNOLI, 2004), durante um projeto de Cooperação Acadêmica FACIN-PUCRS e COPPE/Sistemas-UFRJ (ROCHA, 2004).

Embora esta ferramenta tenha sido desenvolvida no contexto de uma aplicação específica, se observa que problemas de incompatibilidades também acontecem durante a ED. Estas incompatibilidades podem ocorrer em situações tais como:

- O Engenheiro de Domínio não utilizou a sistemática sugerida pelo processo para a geração de componentes de negócio;
- Durante o refinamento da ED, novos componentes são criados e estes componentes requerem serviços que não são prestados por interfaces de outros componentes exatamente da forma esperada.

Por estas razões, pretende-se estender a atividade de geração dos elementos arquiteturais no processo CBD-Arch-DE para contemplar uma atividade de adaptação e composição de componentes originalmente incompatíveis. Esta extensão deve ser analisada no sentido de observar não só a sua incorporação ao processo, mas também o impacto que esta atividade terá sobre as demais atividades da ED. Inicialmente, com a inclusão desta atividade, se prevê a inclusão de mais um critério e/ou restrição durante a atividade de agrupamento de componentes, no sentido de que componentes adaptados não podem pertencer a um elemento arquitetural sem o seu componente adaptador. Além desta questão de acoplamento, este componente adaptador deve ser também analisado sob o ponto de vista da variabilidade e opcionalidade, em função dos componentes envolvidos na sua adaptação.

6.2.5 - Atividades de Experimentação

Uma outra limitação e possibilidade de trabalho futuro é a continuidade dos estudos de observação para o processo CBD-Arch-DE. Deve ser planejado um novo estudo de observação, contemplando todas as atividades do CBD-Arch-DE para avaliar a viabilidade das mesmas. Os resultados obtidos neste estudo de observação devem ser considerados como atividades de pré-experimentação para que então o CBD-Arch-DE possam ser posteriormente analisado numa situação real.

Com base nos resultados obtidos, é previsto o planejamento, a longo prazo, de um estudo experimental. Para este estudo deve ser escolhida uma população que desenvolva atividades de ED e com experiência de desenvolvimento de software baseado em componentes. Parte desta população deve utilizar um processo ad-hoc de ED visando a geração da AR e, a outra parte, o processo CBD-Arch-DE, visando a obtenção da mesma arquitetura. O planejamento desta atividade de experimentação gera uma série de questões a serem analisadas. Exemplos destas questões são: a) a especificação das variabilidades e opcionalidades nos modelos de domínio está consistente em relação as suas respectivas características do domínio? b) o apoio

oferecido para a geração de elementos arquiteturais foi adequado para a geração de arquiteturas de referência?

6.2.6 - Projeto Arquitetural de Componentes para o domínio de Ensino Colaborativo Apoiado por Computador

Esta Tese, apresentada ao longo deste documento, tem como base um conjunto de pesquisas desenvolvidas desde 2000 junto ao PPGCC-FACIN-PUCRS. Neste contexto, foram estudados ambientes da área de Ensino Colaborativo Apoiado por Computador (do inglês *Computer Supported Cooperative Learning* - CSCL), baseados em tecnologias de reutilização como *frameworks*, padrões e componentes. Tais ambientes foram estudados com o intuito de avaliar o apoio oferecido na criação e reutilização de aplicações por eles geradas. Foram estudados alguns *frameworks* e ambientes baseados em componentes para o domínio de CSCL (BLOIS & BECKER, 2001). Neste estudo, se observou dentre outras coisas, que as propostas de CSCL não eram flexíveis do ponto de vista de reutilização. Seus usuários tinham que se adaptar a proposta de estrutura e serviços disponíveis pelo ambiente, não permitindo que os profissionais responsáveis pela construção da aplicação gerada pudessem adaptá-la às necessidades do seu público alvo.

Com base nestas atividades de pesquisa, foi proposta uma abordagem para apoio ao projeto arquitetural, baseado em componentes, no domínio de CSCL (BLOIS & BECKER, 2002), com o objetivo de tornar o processo de projeto mais flexível, adaptado às necessidades dos projetistas de uma aplicação deste domínio. No exame de qualificação deste doutoramento (BLOIS, 2004), esta proposta foi refinada e ampliada para apoiar o projeto arquitetural de componentes, mas num contexto de ED. Como resultado deste refinamento, percebeu-se que este problema não era específico ao domínio de CSCL, mas para quaisquer domínios de aplicações, indicando a necessidade do desenvovimento da presente Tese.

Assim, como forma de resgatar esta motivação inicial e de aplicar a pesquisa inicialmente desenvolvida, pretende-se retomar a proposta de projeto arquitetural de componentes para o domínio de CSCL. Deve ser utilizado processo CBD-Arch-DE para orientar as atividades que norteiam a construção de arquiteturas de referência para este domínio, possivelmente avaliando os ganhos e dificuldades da utilização deste processo e seu ferramental correspondente.

Referências Bibliográficas

APACHE, 2005, "Velocity Java-based Template Engine". In: http://jakarta.apache.org/velocity, accessed in 29/11/2005.

ARANGO, G., 1994, "Domain Analysis Methods". In: SCHÃFER, W., PRIETO-DIÁZ, R., MATSUMOTO, M. (eds), *Software Reusability*, New York, Ellis Horwood.

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

BLOIS, A., 2003, *Framework de Requisitos para CSCL*, Relatório da Disciplina de Tópicos Especiais em Engenharia de Software 1- COPPE-Sistemas, UFRJ.

BLOIS, A.P., BECKER, K., 2001, "Considerations on the application of object-oriented reuse technology to the Computer-Supported Cooperative Learning Domain". In: *7th International Workshop on Groupware - CRIWG 2001*, pp. 164-169, Darmstadt.

BLOIS, A.P., BECKER, K., 2002, "A Component-based Architecture to Support Collaborative Application Design." *Lecture Notes in Computer Science. Berlin : Springer Verlag*, v. 2440, pp. 134-144. , La Serena - Chile.

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

BLOIS, A.P., WERNER, C., BECKER, K., 2005b, "Towards a Components Grouping Technique within a Domain Engineering Process". In: *EUROMICRO*, Porto, September.

BLOIS, A.P.T.B., 2004, *Uma proposta de projeto arquitetural baseado em componentes no contexto de Engenharia de Domínio*, Exame de Qualificação, COPPE, UFRJ, Rio de Janeiro, RJ, Brasil.

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

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

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

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

BRIAND, L.C., EMAM, K.E., MORASCA, S., 1996, "On the Application of Measurement Theory in Software Engineering", *Journal of Empirical Software Engineering*, v. 1, n. 1, pp. 61-88.

BROWN, A., 2000, *Large-Scale Component-Base Development*, Prentice Hall.

BROWN, A., WALLNAU, K., 1998, "The Current State of CBSE", *IEEE Software* (setembro/outubro), pp. 37-46.

BUSCHMANN, F., MEUNIER, R., ROHNERT, H., 1996, *Pattern-Oriented Software Architecture, A System of Patterns*, 1 ed., John Wiley & Sons.

CHEESMAN, J., DANIELS, J., 2000, *UML components: a Simple Process for Specifying Component-based Software*, Addison-Wesley Longman Publishing Co., Inc.

CHEN, F., WANG, Q., MEI, H.*, et al.*, 2002, "An architecture-based approach for component-oriented development". In: *26th Annual International Computer Software and Applications Conference (COMPSAC 2002)*, pp. 450-455 Oxford, England, August.

CHO, E.S., KIM, C.J., KIM, D.D.*, et al.*, 1998, "Static and dynamic metrics for effective object clustering". In: *Asia Pacific Software Engineering Conference*, pp. 78 - 85, Taipei - Taiwan, December.

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

CLEMENTS, P., 1995, "Understanding Architectural Influences and Decisions in Large System Projects". In: *1st. International Workshop om Architectures for Software Systems*, pp. 31-43, Seattle, EUA, April.

CLEMENTS, P., BACHMANN, F., BASS, L.*, et al.*, 2002a, *Documenting Software Architectures: Views and Beyond*, New York, Addison-Wesley

CLEMENTS, P., KAZMAN, R., KLEIN, M., 2002b, *Evaluating Software Architectures: Methods and Case Studies*, New York, Addison-Wesley.

COLLINS-COPE, M., MATTHEWS, H., 2000, "A Reference Architecture for Component-Based Development". In: *6th International Conference on Object Oriented Information Systems (OOIS'2000)*, pp. 225-237, London, UK, 18-20 December

CRISPEN, R., FREEMON, B., KING, K.*, et al.*, 1993, "DARTS: a domain architecture for reuse in training systems". In: *15th Interservice/Industry Training Systems and Education Conference*, Huntsville, Alabama.

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

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

D'SOUZA, D.F., WILLS, A.C., 1999, *Objects, components, and frameworks with UML: the catalysis approach*, Addison-Wesley Longman Publishing Co., Inc.

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

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

DASHOFY, E., HOEK, A.V.D., TAYLOR, R., 2002, "An Infrastructure for the Rapid Development of XML-based Architecture Description Languages". In: *24rd International Conference on Software Engineering (ICSE 2002)*, pp. 266 - 276, May.

DIRCZE, R., 2002, "Java Metadata Interface (JMI) Specification Version 1.0", *Unisys Corporation and Sun Microsystems*, v. 2002.

EMMERICH, W., ELLMER, E., FIEGLEIN, H., 2001, "TIGRA – An architectural style for enterprise application integration". In: *23rd International Conference on Software Engineering (ICSE, 2001)*, pp. 567-576, Toronto, Canada, May.

FAYAD, M., 1997, "Object-oriented application frameworks", *Communication of the ACM*, v. 40, n. 10 (October), pp. 32-38.

FERRÉ, X., VEGAS, S., 1999, "An Evaluation of Domain Analysis Methods". In: *4th CAiSE/IFIP8.1 International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design (EMMSAD'99)*, pp. 14-15., Heidelberg, Germany.

FIELDING, R., 2000, *Architectural Styles and the Design of Network-based software architecture*, University of California, Irvine, USA.

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

FRAKES, W., PRIETRO-DIAZ, R., FOX, C., 1998, "DARE: Domain analysis and Reuse Environment", *Annals of Software Engineering* v. 5, pp. 125-141.

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

GARLAN, D., 2000, "Software Architecture: a Roadmap". In: *International Conference on Software engineering: Future of SE Track*, pp. 91-101, Limerick, Ireland.

GENTLEWARE, 2005, "Poseidon for UML". In: http://gentleware.com, accessed in 25/11/2005.

GIMENES, I.M.D.S., HUZITA, E.H.M., 2005, *Desenvolvimento Baseado em Componentes*, Rio de Janeiro, Ciência Moderna.

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

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

GROSSKURTH, A., GODFREY, M.W., 2005, "A Reference Architecture for Web Browsers". In: *21st IEEE International Conference on Software Maintenance (ICSM'05)*, pp. 661 - 664

HERZUM, P., SIMS, O., 2000, *Business Component Factory: A Comprehensive Overview of Component-Based Development for the Enterprise*, OMG Press.

HOEK, A., DINCEL, E., MEDMIDOVIC, N., 2003, "Using Service Utilization Metrics to assess and Improve Produc Line Architectures". In: *9th IEEE International Software Metrics Symposium (METRICS'2003)*, pp. 298-308, Sydney, Australia.

HOFMEISTER, C., NORD, R.L., SONI, D., 1999, "Describing Software Architecture with UML". In: *3th Working IFIP Conference on Software Architecture*, pp. 145-160, Santo Antonio, Texas, USA.

JACOBSON, I., GRISS, M., JOHNSON, P., 1997, *Software Reuse*, Addison-Wesley.

JOHNSON, R., 1997, "Frameworks = (Components + Patterns)", *Communications of the ACM*, v. 40, n. 10 (October), pp. 39-43.

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

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

KAZMAN, R., ABOWD, G., BASS, L.*, et al.*, 1994, "SAAM: A Method for Analyzing and Properties of Software Architecture". In: *16th International Conference on Software Engineering*, pp. 1981-90, Sorrento, Italy, May.

KAZMAN, R., BARBACCI, M., KLEIN, M., *et al.*, 1999, "Experience with Performing Architecture Tradeoff Analysis". In: *21st Internacional Conference on Software Engineering*, pp. 54-63, Los Angeles, USA, May.

KLEPPE, 2003, *MDA Explained: The Model Drive Architecture – Practice and Promise*, Boston, Addison-Wesley.

KRUCHTEN, P., 1995, "The 4+1 View Model of Architecture", *IEEE Software*, v. 12, n. 6 (November), pp. 42-50.

LEE, J.K., JUNG, S.J., KIM, S.D., *et al.*, 2001, "Component identification method with coupling and cohesion". In: *Eighth Asia-Pacific Software Engineering Conference (APSEC 2001)* pp. 79 - 86, Macau, China, December.

LEE, K., KANG, K.C., LEE, J., 2002, "Concepts and Guidelines of Feature Modeling for Product Line Software Engineering". In: *Software Reuse: Methods, Techniques, and Tools : 7th International Conference, ICSR-7, Proceedings* pp. 62 - 77, Austin, TX, USA, April, 2002.

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

MAIA, N., 2006, *Odyssey-MDA: Uma abordagem para transformação de modelos*, Dissertação de M.Sc., COPPE, UFRJ, Rio de Janeiro. (em conclusão).

MAIA, N., BLOIS, A.P., WERNER, C., 2005a, "Odyssey-MDA: Uma abordagem para transformação de modelos de componentes". In: *V Workshop de Desenvolvimento Baseado em Componentes*, Juiz de Fora, Brasil.

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

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

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

MEDVIDOVIC, N., TAYLOR, R., 2000, "A classification and comparation framework for software architecture description languages", *IEEE Transactions on Software Engineering*, v. 26, n. 1 (January), pp. 70-93.

MENCL, V., 1998, *Component Definition Language*, Facultas Mathematica Physicaque, Universitas Carolina Pragensis, Pragensis.

MENDES, A., 2002, *Arquitetura de Software: desenvolvimento orientado a arquitetura*, Rio de Janeiro, Editora Campus.

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

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

MURTA, L.G.P., OLIVEIRA, H., DANTAS, C.R., *et al.*, 2004a, "Towards Component-based Software Maintenance via Software Configuration Management

Techniques". In: *XVIII Brazilian Symposium on Software Engineering , Workshop on Modern Software Maintenance*, Brasília, DF, Brazil, outubro.

MURTA, L.G.P., VASCONCELOS, A.P.V., BLOIS, A.P.T.B., *et al.*, 2004b, "Run-time Variability through Component Dynamic Loading". In: *XVIII Simpósio Brasileiro de Engenharia de Software. Caderno de Ferramentas*, pp. 67-72, Brasília, DF, Brazil, outubro.

ODYSSEY, 2005, "Odyssey SDE". In: http://reuse.cos.ufrj.br/odyssey, accessed in 25/11/2005.

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

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

OLIVEIRA, R.F., BLOIS, A.P.T.B., VASCONCELOS, A., *et al.*, 2005, *Metamodelo de Características da Notação Odyssey-FEX - Descrição das Classes*, COPPE/UFRJ, Projeto Reuse - Relatório Técnico 2/2005.

OMG, 2002a, "MetaObjectFacility (MOF) Specification - Version 1.4". In: http://www.omg.org/technology/documents/formal/mof.htm, accessed in 08/09/2005.

OMG, 2002b, "OMG - XML Metadata Interchange (XMI) Specification Version 1.2". In: http://www.omg.org/cgi-bin/doc?formal/02-01-01, accessed in 08/09/2005.

OMG, 2003, "MDA Guide Version 1.0.1". In: http://www.omg.org/docs/omg/03-06-01.pdf, accessed in 08/09/2005.

OMG, 2005, "Unified Modeling Language Specification Version 2.0". In: http://www.omg.org/cgi-bin/doc?formal/05-07-04, accessed in 04/10/2005.

PFLEEGER, S.L., 1998, "Design the System", *Software Engineering: Theory and Practice*, 1 ed., Prentice-Hall.

PREE, W., 1995, *Design Patterns for Object-Oriented Software Development*, Addison-Wesley.

PRESSMAN, R., 2000, *Software Engineering: A Practitioner's Approach*, 5 ed., McGraw-Hill.

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

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

ROCHA, A.R.C.D., 2004, *Projeto de Cooperação Acadêmica - PROCAD: FACIN-PUCRS e COPPE/Lens-UFRJ*, CAPES.

ROSSAK, W., KIROVA, V., JOLOLIAN, L., *et al.*, 1997, "A Generic Model for Software Architectures", *IEEE Software* (July/August), pp. 84-92.

SAMETINGER, J., 1997, *Software Engineering with Reusable Components*, Springer-Verlag New York, Inc.

SCHMIDT, D., 2000, *Pattern-Oriented Software Architecture Volume 2*, John Wiley & Sons.

SEI/CMU, 2005, "A Framework for Software Product Line Practice - Version 4.2". In: http://www.sei.cmu.edu/productlines/framework.html, accessed in 25/08/05.

SHAW, M., GARLAN, D., 1996, *Software Architecture: Perspectives on an Emerging Discipline*, New Jersey, Prentice-Hall.

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

SPAGNOLI, L., 2004, *Funcionalidades para Adaptação e Composição de Componentes*, PPGCC-FACIN, PUCRS, Porto Alegre.

SUN, 2005a, "Enterprise JavaBeans Technology". In: http://java.sun.com/products/ejb.

SUN, 2005b, "Java Technology". In: http://www.java.sun.com, accessed in 29/09/2005.

SVAHNBERG, M., GURP, J.V., BOSCH, J., 2002, *A Taxonomy of Variability Realization Techniques*, Blekinge Institute of Technology.

SZYPERSKI, C., 1998, *Component Software: Beyond Object-Oriented Programming*, ACM Press / Addison Wesley Longman.

TEIXEIRA, H., BRAGA, R., WERNER, C.M.L., 2004, "Model-based generation of business component architectures". In: *30th Euromicro Conference*, pp. 176-183, Rennes - France, September.

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

TRACZ, W., 1995, "DSSA (Domain-Specific Software Architecture) Pedagogical Example", *ACM SIGSOFT Software Engineering Notes*, v. 20, n. 4 (July), pp. 49-62.

TRAVASSOS, G.H., GUROV, D., AMARAL, E.A.G.G., 2002, *Introdução à Engenharia de Software Experimental*, Relatório Técnico ES-590/02-Abril. Programa de Engenharia de Sistemas e Computação, COPPE/UFRJ. .

VASCONCELOS, A.P.V., WERNER, C.M.L., 2004, "Software Architecture Recovery based on Dynamic Analysis". In: *XVIII Brazilian Symposium on Software Engineering, Workshop on Modern Software Maintenance*, Brasília, DF, Brasil, outubro.

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

VICI, A.D., ARGENTIERI, N., 1998, "FODAcom: An Experience with Domain Analysis in the Italian Telecom Industry". In: *Fifth International Conference on Software Reuse (ICSR5)* pp. 166-175, April.

VITHARANA, P., ZAHEDI, F., JAIN, H., 2003, "Design, Retrieval, and Assembly in Component-based Software Development", *Communication of the ACM*, v. 46 N 11 (November).

WERNER, C., MATTOSO, M., BRAGA, R.*, et al.*, 1999, "Odyssey: Infra-estrutura de reutilização Baseado em Modelos de Domínios". In: *Caderno de Ferramentas do XIII Simpósio Brasileiro de Engenharia de Software*, pp. 17-20, Florianópolis, outubro.

WERNER, C.M.L., 2005, *Projeto Reuse: Técnicas e Ferramentas de apoio à Reutilização de Software*, Projeto Integrado CNPq.

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, Ciência Moderna.

WOHLIN, C., RUNESON, P., HÖST, M.*, et al.*, 2000, *Experimentation in Software Engineering: an introduction*, USA, Kluwer Academic Publishers.

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

XIA, C.L., M. R.; WONG, K., 2000, "Component-Based Software Engineering: Technologies, Quality Assurance Schemes, and Risk Analysis Tools". In: *7th Asia-Pacific Software Engineering Conference (APSEC'2000)*, pp. 372 - 379 Singapura, December.

Anexos

Anexo 1- Descrição do Domínio de Telefonia Móvel

Para auxiliar o leitor no entendimento da abordagem DECOM, proposta nesta Tese, será utilizado como exemplo o domínio de telefonia móvel. Embora seja um domínio bastante conhecido, este anexo apresenta uma breve descrição dos conceitos referentes ao mesmo.

O domínio de Telefonia Móvel abrange conceitos e funcionalidades que são comuns a todos os sistemas desenvolvidos para diferentes marcas de aparelhos, embora alguns deles telefones apresentem funcionalidades específicas de uma marca.

Os elementos encontrados em todos os aparelhos são:

- Campainha, que representa o toque do telefone;
- Agenda, onde são armazenados números de telefone, nome do proprietário, dentre outras informações;
- Chamada telefônica, que representa um conjunto de informações de uma ligação recebida ou efetuada;
- Caixa Postal, onde são armazenadas mensagens de texto e/ou voz enviadas para o usuário, e;
- Visor, que pode ser colorido ou monocromático.

Embora os elementos acima estejam contemplados em todos os telefones celulares, em alguns casos eles podem apresentar diferentes alternativas de apresentação destas funcionalidades. Tais alternativas é que devem ser consideradas pelo desenvolvedor de software para um aparelho específico.

Outros elementos, também presentes, mas não em todos os aparelhos são: Câmera fotográfica, Alerta Vibratório, Alarme, Acesso à Internet, Envio de mensagens de texto, como e-mails e recados, Jogos e Recebimento de toques musicais.

Toques musicais representam uma forma diferenciada de campainha, ou seja, toques especiais, os quais podem ser do tipo monofônicos, polifônicos ou MP3, e que geralmente são oferecidos pela operadora de telefonia móvel. Toques monofônicos são toques musicais que a maioria dos aparelhos celulares pode reproduzir. O celular só precisa executar uma nota por vez para reproduzir um toque monofônico. Toques polifônicos são toques musicais que podem consistir de várias notas de uma vez, reproduzidas por meio de um alto-falante em vez de uma campainha. São toques mais elaborados, com som muito próximo das músicas reais. Toques do tipo MP3 são toques

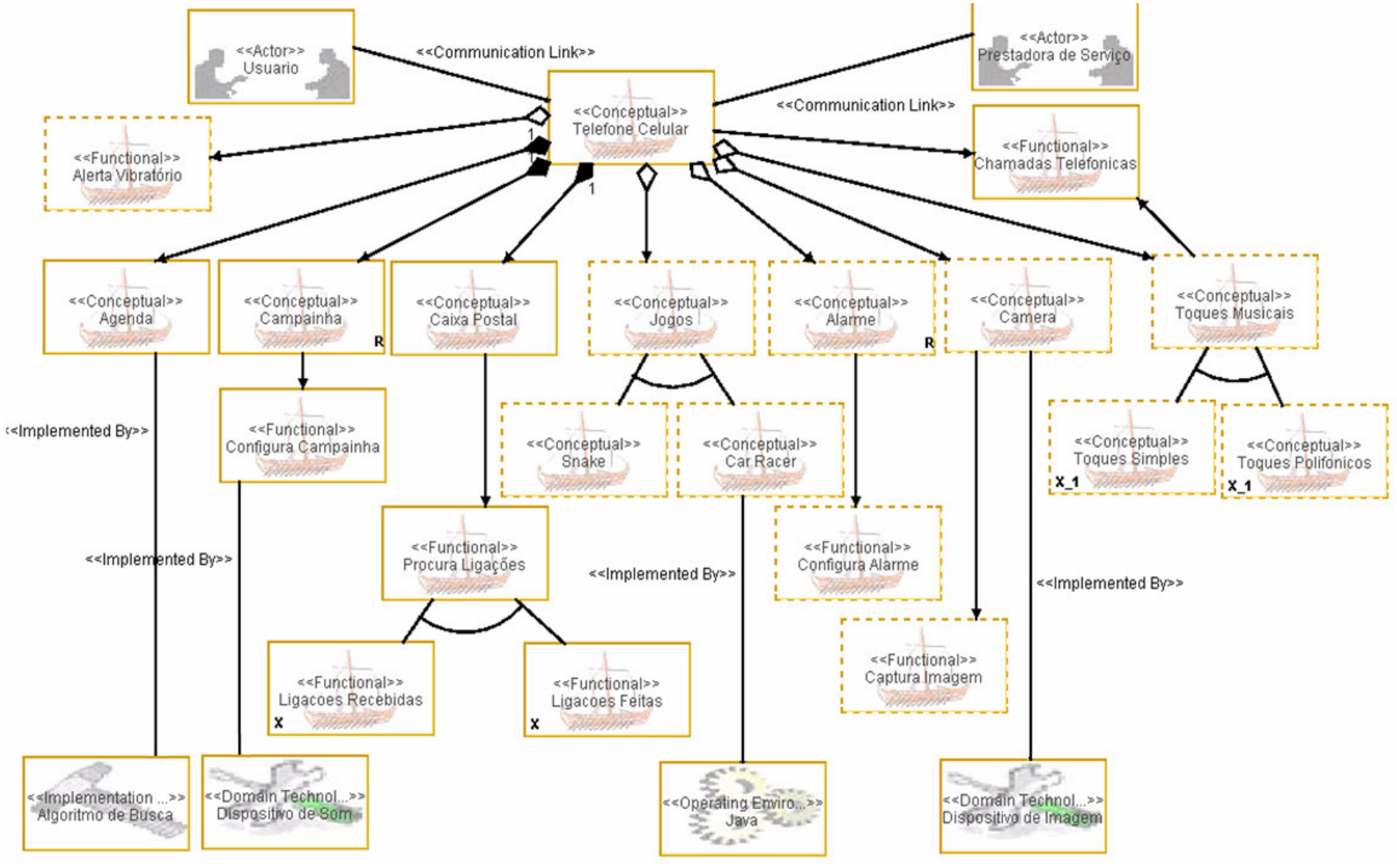
que utilizam tecnologia e formato padrão para compactar uma seqüência de sons em um arquivo extremamente pequeno, preservando o nível original da qualidade do som ao ser reproduzido, i. e., músicas reais podem ser utilizadas como campainha de um celular (NOKIA, 2005).

Jogo é um outro exemplo de elemento opcional num aparelho de telefone celular. Existe uma variedade de jogos que pode estar presente em um ou outro aparelho. Exemplos de jogos são: corrida, campo minado, tênis, etc.

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

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

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


<<Actor>>
Usuario
<<Communication Link>>
<<Actor>>
Prestadora de Serviço
<<Communication Link>>
<<Conceptual>>
Telefone Celular
<<Functional>>
Alerta Vibratório
<<Functional>>
Chamadas Telefonicas
1
1
<<Conceptual>>
Agenda
<<Conceptual>>
Campainha
R
<<Conceptual>>
Caixa Postal
<<Conceptual>>
Jogos
<<Conceptual>>
Alarme
R
<<Conceptual>>
Camera
<<Conceptual>>
Toques Musicais
<<Implemented By>>
<<Functional>>
Configura Campainha
<<Conceptual>>
Snake
<<Conceptual>>
Car Racer
<<Conceptual>>
Toques Simples
X_1
<<Conceptual>>
Toques Polifonicos
X_1
<<Implemented By>>
<<Functional>>
Procura Ligações
<<Implemented By>>
<<Functional>>
Configura Alarme
<<Implemented By>>
<<Functional>>
Ligacoes Recebidas
X
<<Functional>>
Ligacoes Feitas
X
<<Functional>>
Captura Imagem
<<Implementation ...>>
Algoritmo de Busca
<<Domain Technol...>>
Dispositivo de Som
<<Operating Enviro...>>
Java
<<Domain Technol...>>
Dispositivo de Imagem

Anexo 2 – Especificação de Componentes na UML 2.0

A especificação de uma arquitetura de componentes é uma atividade de modelagem que requer o uso de sistemáticas que sugiram formas adequadas para tal especificação. Neste sentido, a UML (Unified Modelling Language) representa uma alternativa que tem sido utilizada para a modelagem de sistemas, incluindo aqueles baseados no paradigma de DBC (OMG, 2005). A partir da UML 2.0 tornou-se possível a modelagem de sistemas baseadas em componentes, desde a sua especificação até a sua implementação, sendo esta última já disponível nas versões anteriores da UML. Dentre os elementos que estão diretamente ligados à tecnologia de DBC na UML 2.0, são brevemente discutidos os seguintes artefatos: componente, interface, porta e conector.

Na UML 2.0, um componente possui uma especificação lógica (e.g., componentes de negócio, componentes de processo) e física (e.g., componentes EJB, CORBA). Um componente é considerado como uma classe especializada que tem uma especificação externa (interfaces providas e requeridas) e interna (classificadores que realizam seu comportamento). A Figura 1 mostra um exemplo das diferentes representações de componentes e interfaces na UML 2.0. A parte (a) da figura mostra a representação de componentes com interfaces providas – *ball e* requerida – *socket*. A parte (b) mostra o mesmo componente, informando as interfaces providas e requeridas dentro do próprio componente. A representação da parte (b) é conhecida como *black-box*. A terceira representação do mesmo componente pode ser observada na parte (c) da figura, onde as interfaces são representadas por classes com estereótipo <<interface>>. A interface fornecida é relacionada ao componente por meio de realização e a requerida através de dependência. A última representação de componentes (d), também conhecida com representação *white-box*, lista os objetos que realizam o componente internamente, bem como os artefatos que representam a implementação do componente.

Além dos conceitos de componente e interfaces, o pacote da UML 2.0 referente a componentes introduz também o conceito de estruturas compostas: Portas e Conectores. Uma porta é uma propriedade de um classificador (componente) que especifica um ponto de interação entre o classificador e seu ambiente ou entre o classificador e suas partes internas. Conectores são a forma de ligação entre interfaces providas e requeridas. Existem dois tipos de conectores. O conector de delegação liga um contrato externo de um componente (baseada na especificação de suas portas) a

uma realização interna de um componente. O conector de montagem conecta duas instâncias de componentes. A Figura 2 mostra exemplos de utilização do conector de delegação e de montagem.

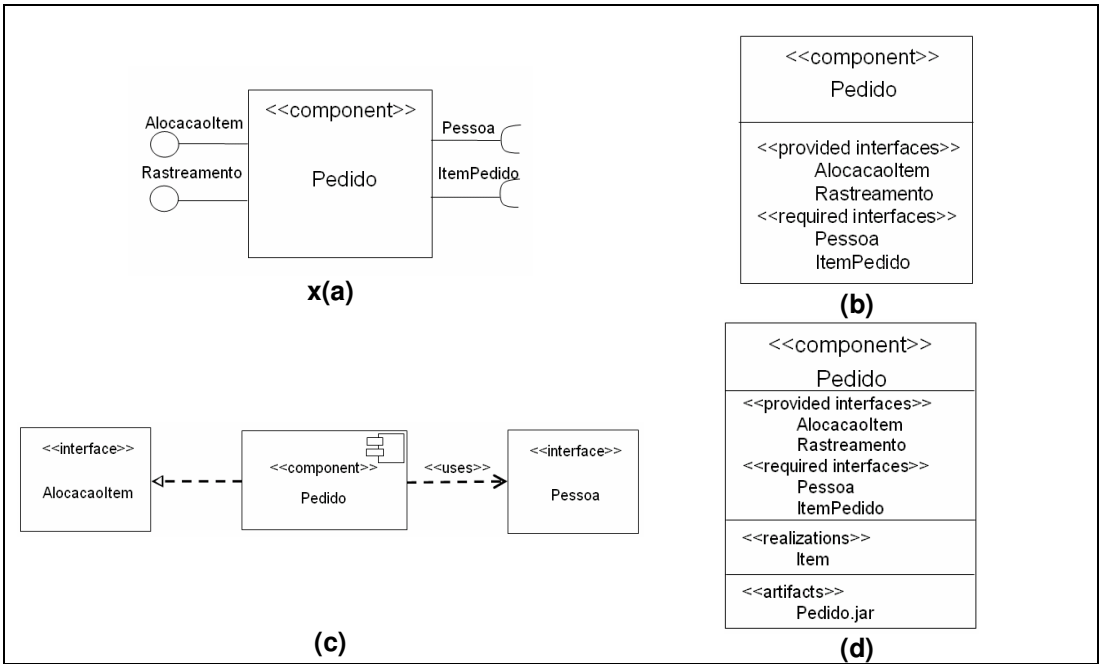


Figura 1: Diferentes representações de Componentes na UML 2.0

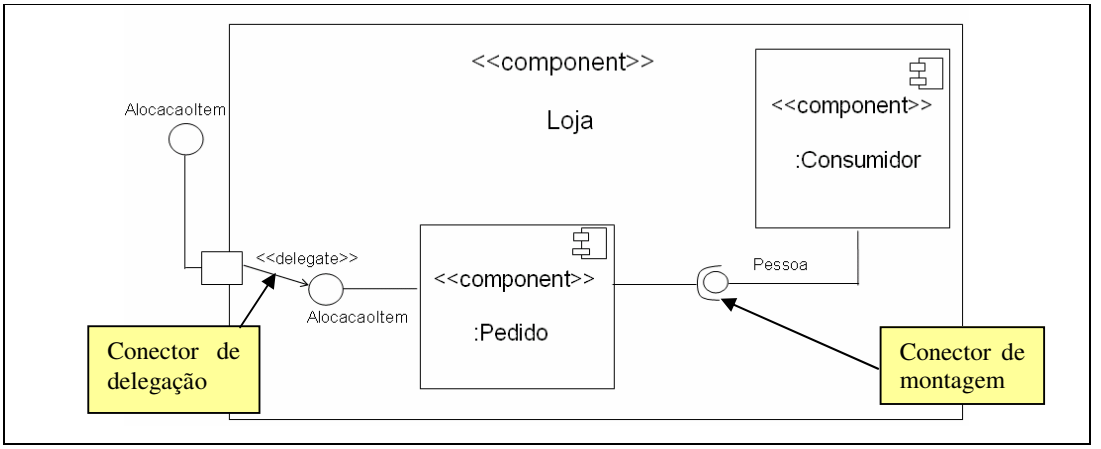


Figura 2: Conectores de montagem e de delegação na UML 2.0

Embora nem todos os classificadores da UML para especificação de componentes tenham sido descritos, já é possível obter uma idéia do potencial de especificação incorporado a esta versão da UML, aumentando a capacidade de especificação dos métodos de DBC que utilizam a UML como linguagem para a modelagem de artefatos. Poseidon é uma das ferramentas case que disponibiliza algumas das especificações para componentes (GENTLEWARE, 2005).

A proposta de projeto arquitetural baseado em componentes desta tese utiliza parte da notação apresentada acima, incluindo a representação de componente da parte (a) da Figura 1, portas e conectores. Esta notação é explorada nos capítulos 3, 4 e 5.

Anexo 3 – Mensagens para Professores e Participantes dos Estudos de Observação

E-mail enviado para os professores da FACIN

Pessoal,

O projeto de Engenharia de Domínio apoiado pelo desenvolvimento baseado em componentes pressupõe a geração de componentes e o agrupamento destes em elementos arquiteturais. Como parte de meu trabalho de doutorado, foi proposto um processo de Engenharia de domínio, e dentro deste contexto:

- um conjunto de heurísticas para apoiar o mapeamento dos tipos de negócio em componentes de negócio. Estas heurísticas levam em conta as características de obrigatoriedade e variabilidade e os relacionamentos existentes entre os tipos de negócio (e.g. composição, agregação, associação, dependência) para a geração dos respectivos componentes de negócio.

- critérios de acoplamento de componentes, visando reduzir troca de mensagens entre componentes, e maior acoplamento dos elementos arquiteturais.

Objetivo:

Realizar estudos de observação, visando avaliar a utilização de cada uma destas propostas.

E-mail enviado para os Participantes do Estudo

Prezado Participante,

Estou precisando fazer três estudos de observação sobre a minha tese.

O primeiro estudo eu quero observar o uso de um conjunto de heurísticas que foram propostas para o mapeamento de características do domínio de Aprendizagem Cooperativa apoiada por computador para tipos de negócio deste domínio.

O segundo estudo eu quero observar o uso de um conjunto de heurísticas que foram propostas para o mapeamento de tipos de negócio do mesmo domínio em componentes de negócio.

O terceiro estudo diz respeito ao uso de um conjunto de critérios para agrupamento de componentes de um domínio de aplicação para a geração de elementos arquiteturais.

Só preciso que você tenha uma experiência mínima de projeto de software.

Vou precisar em torno de 2 horas para efetuar os três estudos. Qual horário e dia ficam melhores para você? Tenho mais algumas pessoas para efetuar a mesma tarefa e queria organizar este calendário de execução do experimento.

Desde já agradeço a atenção e disponibilidade.

Um abraço, Ana Paula

Anexo 4 - Formulário de Consentimento do 1º. Estudo de Observação

GERAÇÃO DE TIPOS DE NEGÓCIO DO DOMÍNIO

Eu declaro ter mais de 18 anos de idade e que concordo em participar de um estudo conduzido por Ana Paula Terra Bacelo Blois, como parte das atividades de doutoramento no Programa de Engenharia de Sistemas e Computação da COPPE/UFRJ. Este estudo visa compreender a viabilidade de aplicação de heurísticas para a geração de tipos de negócio do domínio.

PROCEDIMENTO

Este estudo acontecerá em duas etapas. Primeiro é apresentado o ferramental necessário para a geração dos tipos de negócio e o modelo de características do domínio que será utilizado neste estudo. Na segunda etapa os participantes deverão utilizar o ferramental apresentado previamente para a identificação e criação dos tipos de negócio. Eu entendo que, uma vez o experimento tenha terminado, os trabalhos que desenvolvi, serão estudados visando entender a eficiência dos procedimentos e as técnicas que me foram ensinadas.

CONFIDENCIALIDADE

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

BENEFÍCIOS, LIBERDADE DE DESISTÊNCIA

Eu entendo que os benefícios que receberei deste estudo são limitados ao aprendizado do material que é distribuído e ensinado, independente de participar ou não deste estudo, mas que os pesquisadores esperam aprender mais sobre quão eficiente é a forma de agrupamento de tipos de negócio ao qual o experimento se propõe a conduzir.

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

EQUIPE:

PESQUISADOR RESPONSÁVEL

MSC. Ana Paula Terra Bacelo Blois - Programa de Engenharia de Sistemas e Computação - COPPE/UFRJ

PROFESSORES RESPONSÁVEIS

Profa. Cláudia M.L. Werner - Programa de Engenharia de Sistemas e Computação - COPPE/UFRJ

Profa. Karin Becker - PPGCC – FACIN / PUCRS

Nome (em letra de forma):_____

Assinatura: _____ Data:_____

Anexo 5 - Formulário de Caracterização

CARACTERIZAÇÃO DO DESENVOLVEDOR

Nome : _____

Formação Acadêmica:

□ Doutorado □ Mestrado □ Especialização □ Graduação
□ Técnico □ Outra: _____

Formação Geral:

Qual é sua experiência anterior com desenvolvimento de software com o foco em Engenharia de Domínio ou Linha de Produtos na prática ?
(marque aqueles itens que melhor se aplicam)
( ) nunca desenvolvi software baseado em reutilização.
( ) tenho desenvolvido software com foco em artefatos reutilizáveis para uso próprio.
( ) tenho desenvolvido software com foco em artefatos reutilizáveis como parte de uma equipe, relacionado a um curso.
( ) tenho desenvolvido software com foco em artefatos reutilizáveis como parte de uma equipe, na indústria.

Por favor, explique sua resposta. Inclua o número de semestres ou número de anos de experiência relevante em desenvolvimento. (E.g. "Eu trabalhei por 10 anos como programador na indústria")
_____
_____

Você já utilizou ou utiliza alguma técnica para apoio ao desenvolvimento com foco em reutilização?
( ) Sim ( )Não Indique a(s) técnica(s)?
_____
_____

Por favor, indique o grau de sua experiência nesta seção seguindo a escala de 5 pontos abaixo:
1 = nenhum 2 = estudei em aula ou em livro
3 = pratiquei em 1 projeto em sala de aula 4 = usei em 1 projeto na indústria
5 = usei em vários projetos na indústria

Experiência em projeto de sistemas	1	2	3	4	5
Experiência em desenvolver projetos a partir de tipos de negócio	1	2	3	4	5
Domínio de Conceitos de reutilização de software	1	2	3	4	5
Experiência com Unified Modeling Language (UML)	1	2	3	4	5

Experiência em Contextos Diferentes:

Esta seção será utilizada para compreender quão familiar você está 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:
1 = Eu não tenho familiriadade com a área. Eu nunca fiz isto.
3 = Eu utilizo isto algumas vezes, mas não sou um especialista.
5 = Eu sou muito familiar com esta área. Eu me sentiria confortável fazendo isto.

Área: Ensino Colaborativo: _____

Anexo 6 – Treinamento da Notação Odyssey-FEX

LEITURA INTRODUTÓRIA

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

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

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

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

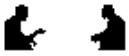
As características na notação Odyssey-FEX possuem a seguinte classificação quanto a sua variabilidade:

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

A classificação das características quanto a opcionalidade indica justamente a obrigatoriedade ou não-obrigatoriedade da presença de um determinado elemento nas aplicações ou produtos a serem desenvolvidos. Vale ressaltar que a opcionalidade é referente ao domínio como um todo. Características que são opcionais em relação ao

domínio, mas que por ventura venham a ser mandatórias em relação a outras características a serem selecionadas, deverão expressar essa informação através de Regras de Composição. Características opcionais são evidenciadas no modelo por um contorno pontilhado.

Tabela 1 – Tipos de Características na notação Odyssey-FEX

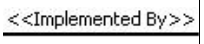
Ícone	Tipo de Característica	
	**Características de Domínio** – Características intimamente ligadas à essência do domínio. Representam as funcionalidades e/ou os conceitos do modelo e correspondem a casos de uso e componentes estruturais concretos.	**Características de Análise**
	**Características de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

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

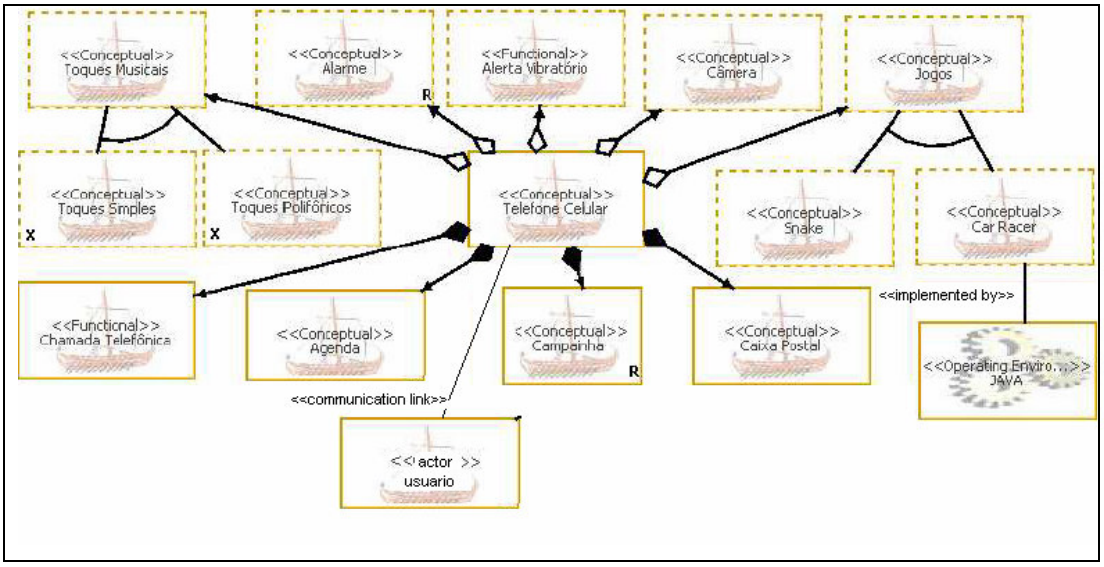
Um dos requisitos para uma notação que represente variabilidade é a necessidade de representação de dependência e exclusividade entre os elementos do modelo.

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

Tabela 2: Relacionamentos da notação Odyssey-FEX

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

Exemplo de Uso da Notação – Domínio de Telefonia Celular



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

As características conceituais, funcionais, de entidade e tecnológicas citadas na Tabela 1 estão identificadas pelos estereótipos <<conceptual>>, <<functional>>, <<actor>> e <<operating environment>>, respectivamente.

As características com retângulos tracejados representam características opcionais e as demais, obrigatórias para qualquer aplicação derivada deste domínio.

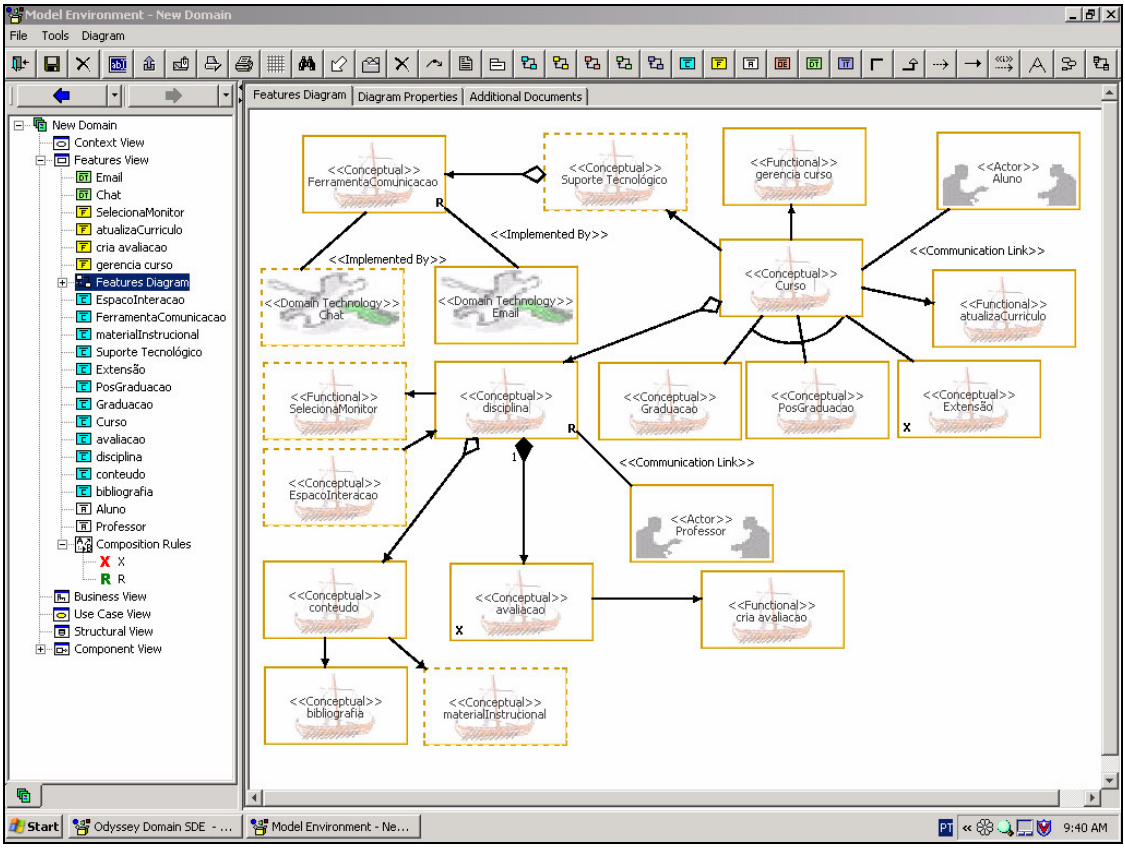
A característica Jogos representa um ponto de variação opcional, e as características Car Racer e Snake representam suas variantes opcionais. Para representar esta relação, é utilizado o relacionamento alternativo. Toques Musicais, Toques Simples e Toques Polifônicos também estão ligados através do relacionamento alternativo, pois representam um ponto de variação e suas variantes.

A característica conceitual Telefone Celular está relacionada a característica de entidade Usuário através de um <<communication link>>. A característica conceitual Car Racer está relacionada a característica de ambiente operacional Java através de um <<implemented by>>.

Telefone Celular está relacionado com as características Chamadas Telefônicas, Agenda, Campanhia e Caixa Postal através de composição. As características Alarme, Alerta Vibratório, Câmera, Jogos e Toques Musicais estão relacionados a Telefone Celular por agregação.

A relação de exclusividade é expressa no modelo através da notação X (e.g. Toques Simples e Toques Polifônicos) e a relação de dependência através da notação R (Alarme e Campainha).

Anexo 7 - Modelo de Características – Estudo de Observação 1



Anexo 8 – Formulário de Avaliação do Mapeamento de Características do Domínio em Tipos de Negócio

O propósito único deste questionário é apoiar o pesquisador no levantamento de dados para melhor entendimento do estudo experimental em questão. Não será utilizado para avaliar o participante.

Avaliação Geral das Heurísticas

1) As heurísticas de mapeamento de características do domínio em tipos de negócio em componentes são de fácil entendimento? Justifique sua resposta caso a opção escolhida seja "não" ou "parcialmente": _____ _____	☐ Sim ☐ Não ☐ Parcialmente
2) As heurísticas propostas foram facilmente mapeadas para o modelo de tipos de negócio disponibilizado? Justifique sua resposta caso a opção escolhida seja "não" ou "parcialmente": _____ _____	☐ Sim ☐ Não ☐ Parcialmente
3) Em algum momento você desejou criar algum tipo de negócio e a sua criação não era sugerida por nenhuma das heurísticas disponibilizadas? Justifique sua resposta caso a opção escolhida seja "sim" ou "parcialmente": _____ _____	☐ Sim ☐ Não ☐ Parcialmente
4) Em algum momento você não aplicou uma heurística, pois julgou que a sua aplicação não seria vantajosa para o modelo de tipos de negócio? Justifique sua resposta caso a opção escolhida seja "sim" ou "parcialmente": _____ _____	☐ Sim ☐ Não ☐ Parcialmente
5) Você identificou alguma outra heurística de mapeamento de características conceituais do domínio em tipos de negócio? Justifique sua resposta caso a opção escolhida seja "sim" ou "parcialmente": _____ _____	☐ Sim ☐ Não ☐ Parcialmente

Avaliação das características de opcionalidade e variabilidades sugeridas pelas heurísticas

6) Os mapeamentos a respeito da opcionalidade (mandatório ou opcional) e variabilidade (ponto de variação, variante e invariante) sugeridos pelas heurísticas são adequados? Justifique sua resposta caso a opção escolhida seja "não" ou "parcialmente": _____ _____	☐ Sim ☐ Não ☐ Parcialmente

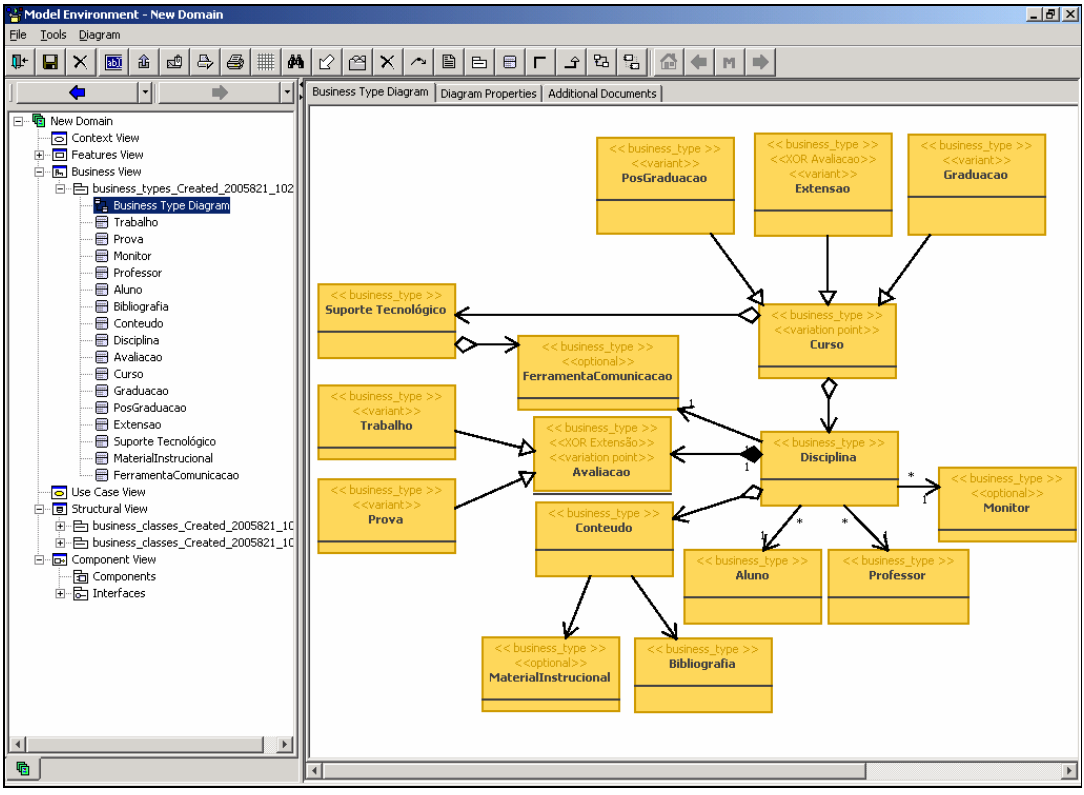
Tipos de Negócio gerados a partir das heurísticas

7) Os tipos de negócio gerados a partir das heurísticas são úteis numa possível geração de componentes de negócio? ☐ Sim ☐ Não ☐ Parcialmente Justifique sua resposta caso a opção escolhida seja "não" ou "parcialmente": _____ _____
8) Os tipos de negócio gerados a partir das heurísticas são reutilizáveis no contexto de uma linha de produtos? ☐ Sim ☐ Não ☐ Parcialmente Justifique sua resposta caso a opção escolhida seja "não" ou "parcialmente": _____ _____

Suporte Computacional para as Heurísticas

9) O suporte computacional existente contemplou adequadamente a utilização das heurísticas de mapeamento de características conceituais do domínio em tipos de negócio? ☐ Sim ☐ Não ☐ Parcialmente Justifique sua resposta caso a opção escolhida seja "não" ou "parcialmente": _____ _____
10) Durante a construção dos tipos de negócio foi fácil identificar qual heurística de mapeamento estava sendo empregada? ☐ Sim ☐ Não ☐ Parcialmente Justifique sua resposta caso a opção escolhida seja "não" ou "parcialmente": _____ _____
11) Você utilizou duas ou mais heurísticas para a geração de um único tipo de negócio? ☐ Sim ☐ Não ☐ Parcialmente Especifique a sua resposta caso a opção escolhida seja "sim" ou "parcialmente": _____ _____
12) As funcionalidades para mapeamento de características conceituais do domínio em tipos de negócio são de fácil utilização? ☐ Sim ☐ Não ☐ Parcialmente Especifique a sua resposta caso a opção escolhida seja "não" ou "parcialmente": _____ _____

Anexo 9 - Modelo de Tipos de Negócio do Domínio



Anexo 10 - Formulário de Consentimento

AGRUPAMENTO DE TIPOS DE NEGÓCIO EM COMPONENTES DE NEGÓCIO DO DOMÍNIO

Eu declaro ter mais de 18 anos de idade e que concordo em participar de um estudo conduzido por Ana Paula Terra Bacelo Blois, como parte das atividades de doutoramento no Programa de Engenharia de Sistemas e Computação da COPPE/UFRJ. Este estudo visa compreender a viabilidade de aplicação de heurísticas para a geração de componentes de negócio do domínio.

PROCEDIMENTO

Este estudo acontecerá em duas etapas. Primeiro é apresentado o ferramental necessário para a geração dos componentes de negócio e o modelo de tipos de negócio do domínio que será utilizado neste estudo. Na segunda etapa os participantes deverão utilizar o ferramental apresentado previamente para a identificação e criação dos componentes de negócio. Eu entendo que, uma vez o experimento tenha terminado, os trabalhos que desenvolvi, serão estudados visando entender a eficiência dos procedimentos e as técnicas que me foram ensinadas.

CONFIDENCIALIDADE

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

BENEFÍCIOS, LIBERDADE DE DESISTÊNCIA

Eu entendo que os benefícios que receberei deste estudo são limitados ao aprendizado do material que é distribuído e ensinado, independente de participar ou não deste estudo, mas que os pesquisadores esperam aprender mais sobre quão eficiente é a forma de agrupamento de tipos de negócio ao qual o experimento se propõe a conduzir.

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

EQUIPE:

PESQUISADOR RESPONSÁVEL

MSC. Ana Paula Terra Bacelo Blois

Programa de Engenharia de Sistemas e Computação - COPPE/UFRJ

PROFESSORES RESPONSÁVEIS

Profa. Cláudia M.L. Werner - Programa de Engenharia de Sistemas e Computação - COPPE/UFRJ

Profa. Karin Becker - PPGCC – FACIN / PUCRS

Nome (em letra de forma):_____

Assinatura: _____ Data:_____

Anexo 11 - Formulário de Avaliação das Heurísticas de Mapeamento de Tipos de Negócio em Componentes de Negócio

O propósito único deste questionário é apoiar o pesquisador no levantamento de dados para melhor entendimento do estudo experimental em questão. Não será utilizado para avaliar o participante.

Avaliação Geral das Heurísticas

1) As heurísticas de agrupamento de tipos de negócio em componentes são de fácil entendimento? Justifique sua resposta caso a opção escolhida seja "não" ou "parcialmente": _____ _____	☐ Sim ☐ Não ☐ Parcialmente
2) As heurísticas de agrupamento de tipos de negócio em componentes foram facilmente mapeadas para o modelo de tipos de negócio disponibilizado? Justifique sua resposta caso a opção escolhida seja "não" ou "parcialmente": _____ _____	☐ Sim ☐ Não ☐ Parcialmente
3) Em algum momento você desejou agrupar um conjunto de tipos de negócio e esta forma de agrupamento não era sugerida por nenhuma das heurísticas disponibilizadas? Justifique sua resposta caso a opção escolhida seja "sim" ou "parcialmente": _____ _____	☐ Sim ☐ Não ☐ Parcialmente
4) Em algum momento você não aplicou uma heurística, pois julgou que a sua aplicação não seria vantajosa para o modelo de componentes gerado? Justifique sua resposta caso a opção escolhida seja "sim" ou "parcialmente": _____ _____	☐ Sim ☐ Não ☐ Parcialmente
5) Você identifica alguma outra heurística para agrupamento de tipos de negócio em componentes? Justifique sua resposta caso a opção escolhida seja "sim" ou "parcialmente": _____ _____	☐ Sim ☐ Não ☐ Parcialmente

Avaliação das características de opcionalidade e variabilidades sugeridas pelas heurísticas

6) As decisões a respeito da opcionalidade (mandatório ou opcional) e variabilidade (ponto de variação, variante e invariante) sugeridas pelas heurísticas são adequadas?	☐ Sim ☐ Não ☐ Parcialmente

Justifique sua resposta caso a opção escolhida seja "não" ou "parcialmente":
_____
_____

7) O agrupamento de tipos de negócio mandatórios e opcionais resultarem num componente mandatório foi adequado durante este experimento?
☐ Sim ☐ Não ☐ Parcialmente

Justifique sua resposta caso a opção escolhida seja "não" ou "parcialmente":
_____
_____

8) A obrigatoriedade de agrupar tipos de negócio que representam pontos de variação e variantes, num componente como um ponto de variação, foi adequado neste experimento?
☐ Sim ☐ Não ☐ Parcialmente

Justifique sua resposta caso a opção escolhida seja "não" ou "parcialmente":
_____
_____

9) A proibição de agrupamento de tipos de negócio que representam uma exclusão no domínio (XOR) foi adequado durante este experimento?
☐ Sim ☐ Não ☐ Parcialmente

Justifique sua resposta caso a opção escolhida seja "não" ou "parcialmente":
_____
_____

10) A restrição de agrupar tipos de negócio relacionados por dependência ou associação, se e somente se os tipos de negócio tiverem a mesma característica de opcionalidade (mandatório ou opcional) foi adequado neste experimento?
☐ Sim ☐ Não ☐ Parcialmente

Justifique sua resposta caso a opção escolhida seja "não" ou "parcialmente":
_____
_____

11) Atribuir ao componente a variabilidade do tipo de negócio que representa o todo numa relação de agregação foi adequado neste experimento?
☐ Sim ☐ Não ☐ Parcialmente

Justifique sua resposta caso a opção escolhida seja "não" ou "parcialmente":
_____
_____

12) É adequado que tipos de negócio relacionados por composição tenham a mesma característica de opcionalidade (opcional ou mandatório)?
☐ Sim ☐ Não ☐ Parcialmente

Justifique sua resposta caso a opção escolhida seja "não" ou "parcialmente":
_____
_____

Componentes gerados a partir das heurísticas

13) Os componentes gerados a partir das heurísticas são úteis numa possível arquitetura de componentes? ☐ Sim ☐ Não ☐ Parcialmente

Justifique sua resposta caso a opção escolhida seja "não" ou "parcialmente":

_____

14) Os componentes gerados a partir das heurísticas são reutilizáveis no contexto de uma arquitetura de domínio ou de uma linha de produtos? ☐ Sim ☐ Não ☐ Parcialmente

Justifique sua resposta caso a opção escolhida seja "não" ou "parcialmente":

_____

_____

Suporte Computacional para as Heurísticas

15) O suporte computacional existente contemplou adequadamente as heurísticas de agrupamento dos tipos de negócio em componentes? ☐ Sim ☐ Não ☐ Parcialmente

Justifique sua resposta caso a opção escolhida seja "não" ou "parcialmente":

_____

_____

16) Durante a construção dos componentes foi fácil identificar qual heurística de agrupamento estava sendo empregada? ☐ Sim ☐ Não ☐ Parcialmente

Justifique sua resposta caso a opção escolhida seja "não" ou "parcialmente":

_____

_____

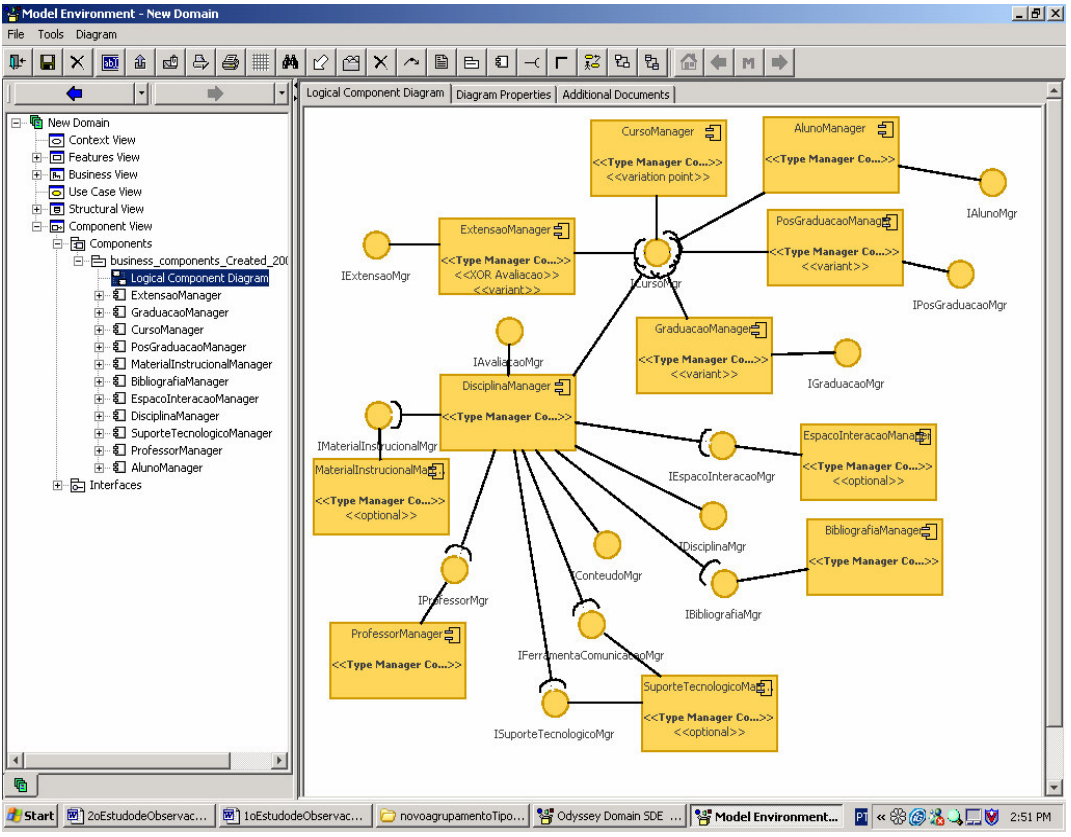
17) As funcionalidades para agrupamento de tipos de negócio em componentes são de fácil utilização? ☐ Sim ☐ Não ☐ Parcialmente

Especifique a sua resposta caso a opção escolhida seja "não" ou "parcialmente":

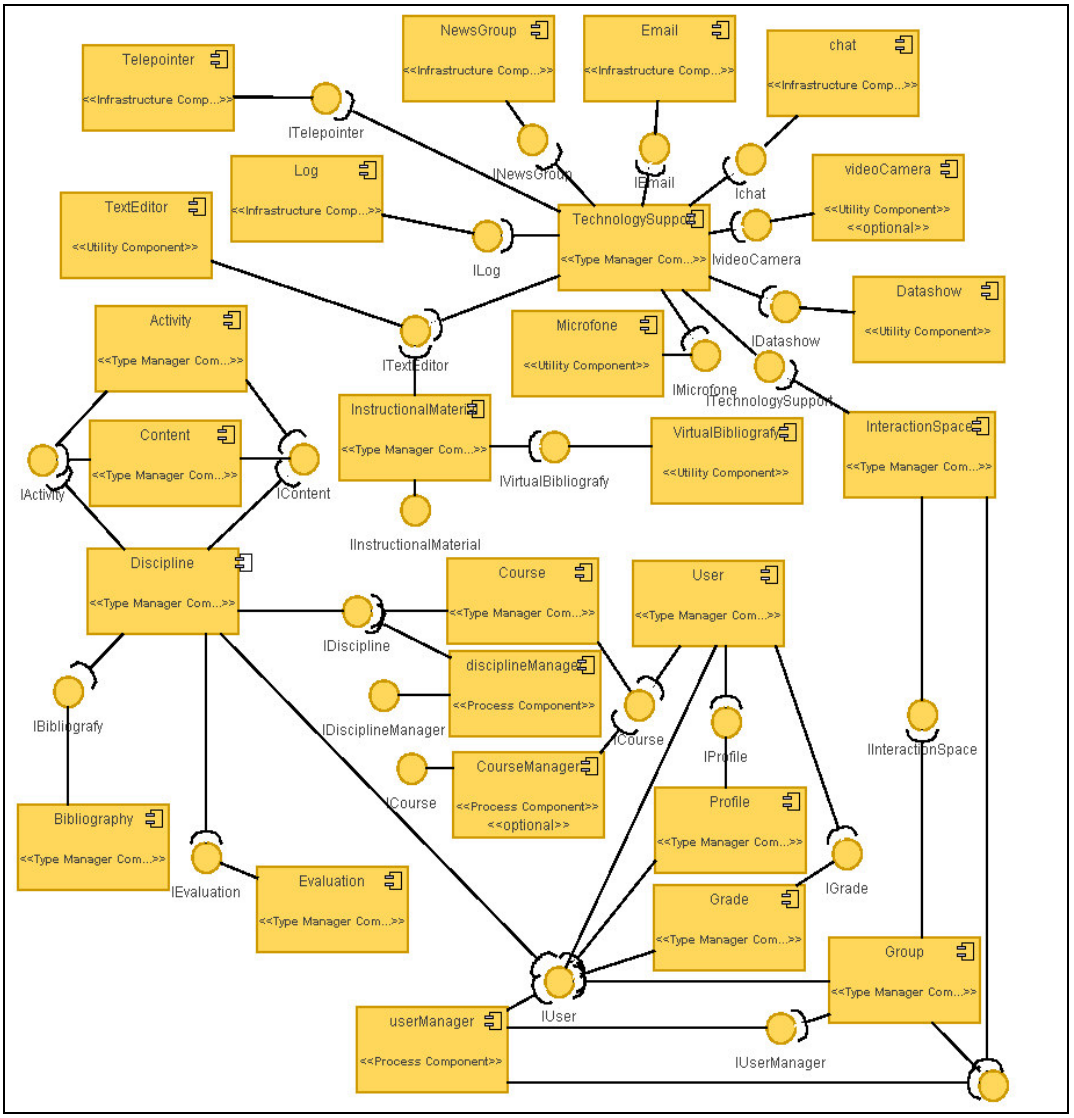
_____

_____

Anexo 12 - Modelo de Componentes de Negócio esperado como resultado do E/OBS-2



Anexo 13 - Modelo de Componentes do Domínio – E/OBS-3



Anexo 14 - Formulário de Consentimento – E/OBS-3

Geração de Elementos Arquiteturais

Eu declaro ter mais de 18 anos de idade e que concordo em participar de um estudo conduzido por Ana Paula Terra Bacelo Blois, como parte das atividades de doutoramento no Programa de Engenharia de Sistemas e Computação da COPPE/UFRJ. Este estudo visa compreender a viabilidade de aplicação de critérios para a geração de elementos arquiteturais do domínio.

PROCEDIMENTO

Este estudo acontecerá em duas etapas. Primeiro é apresentado o ferramental necessário para a geração dos elementos arquiteturais e o modelo de componentes de negócio do domínio que será utilizado neste estudo. Na segunda etapa os participantes deverão utilizar o ferramental apresentado previamente para a identificação e criação dos elementos arquiteturais. Eu entendo que, uma vez o experimento tenha terminado, os trabalhos que desenvolvi, serão estudados visando entender a eficiência dos procedimentos e as técnicas que me foram ensinadas.

CONFIDENCIALIDADE

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

BENEFÍCIOS, LIBERDADE DE DESISTÊNCIA

Eu entendo que os benefícios que receberei deste estudo são limitados ao aprendizado do material que é distribuído e ensinado, independente de participar ou não deste estudo, mas que os pesquisadores esperam aprender mais sobre quão eficiente é a forma de agrupamento de componentes de negócio ao qual o experimento se propõe a conduzir.

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

EQUIPE:

PESQUISADOR RESPONSÁVEL

MSC. Ana Paula Terra Bacelo Blois

Programa de Engenharia de Sistemas e Computação - COPPE/UFRJ

PROFESSORES RESPONSÁVEIS

Profa. Cláudia M.L. Werner - Programa de Engenharia de Sistemas e Computação - COPPE/UFRJ

Profa. Karin Becker - PPGCC – FACIN / PUCRS

Nome (em letra de forma):_____

Assinatura: _____ Data:_____

Anexo 15 - Formulário de Avaliação da Ferramenta de Agrupamento de Componentes –E/OBS-3

Informações Gerais

1) Você já utilizou alguma ferramenta de agrupamento de componentes?	☐ Sim ☐ Não
2) Caso tenha utilizado, qual o nome da ferramenta?	

Configurações da Ferramenta

3) As configurações previamente necessárias para a execução dos critérios de agrupamentos são de fácil entendimento?	☐ Sim ☐ Não ☐ Parcialmente
Especifique caso sua resposta seja "não" ou "parcialmente": _____ _____	

Critérios de Agrupamento de Componentes

4) Os critérios disponibilizados na ferramenta refletem as necessidades de agrupamento de componentes?	☐ Sim ☐ Não ☐ Parcialmente
Especifique caso sua resposta seja "não" ou "parcialmente": _____ _____ _____	
5) Você identifica outros critérios de agrupamento de componentes que não foram sugeridos pela ferramenta?	☐ Sim ☐ Não ☐ Parcialmente
Especifique caso sua resposta seja "não" ou "parcialmente": _____ _____ _____	
6) Os limites de número máximo de agrupamento para cada critério são adequados?	☐ Sim ☐ Não ☐ Parcialmente
Especifique caso sua resposta seja "não" ou "parcialmente": _____ _____ _____	
7) As sugestões de agrupamento de cada critério são úteis?	☐ Sim ☐ Não ☐ Parcialmente
Especifique caso sua resposta seja "não" ou "parcialmente": _____ _____	

Aplicação dos Critérios de Agrupamento de Componentes

8) As sugestões de agrupamento de cada critério seriam facilmente identificadas no modelo de componentes do domínio caso não houvesse o auxílio da ferramenta? Especifique caso sua resposta seja "não" ou "parcialmente": _____ _____ _____	☐ Sim ☐ Não ☐ Parcialmente
9) Você identifica a necessidade de outros agrupamentos de componentes que não foi sugerido pelos critérios existentes na ferramenta? Especifique caso sua resposta seja "não" ou "parcialmente": _____ _____ _____	☐ Sim ☐ Não ☐ Parcialmente
10) A forma de apresentação dos resultados dos critérios e dos agrupamentos identificados por critério é adequada? Especifique caso sua resposta seja "não" ou "parcialmente": _____ _____ _____	☐ Sim ☐ Não ☐ Parcialmente
11) Durante o processo de geração dos agrupamentos, foi adequado o recurso de abas para identificação dos grupos de componentes separados por critério? Especifique caso sua resposta seja "não" ou "parcialmente": _____ _____ _____	☐ Sim ☐ Não ☐ Parcialmente
12) O aviso de agrupamento de componentes redundantes, apresentado durante o processo de geração dos agrupamentos, facilitou a criação de agrupamentos mais adequados? Especifique caso sua resposta seja "não" ou "parcialmente": _____ _____ _____	☐ Sim ☐ Não ☐ Parcialmente
13) A forma de apresentação dos agrupamentos de componentes redundantes é adequada? Especifique caso sua resposta seja "não" ou "parcialmente": _____ _____	☐ Sim ☐ Não ☐ Parcialmente

Resultado da Aplicação dos Critérios de Agrupamento de Componentes

14) O modelo de componentes gerado com os componentes originais e agrupados é mais compreensível quando comparado ao modelo de componentes recebido neste estudo de caso?	☐ Sim ☐ Não ☐ Parcialmente
Especifique caso sua resposta seja "não" ou "parcialmente": _____ _____ _____	
15) Os agrupamentos sugeridos podem ser reutilizados em aplicações derivadas deste domínio?	☐ Sim ☐ Não ☐ Parcialmente
Especifique caso sua resposta seja "não" ou "parcialmente": _____ _____ _____	
16) Você sentiu necessidade de modificar alguma sugestão de agrupamento de componentes (incluir ou remover algum componente)?	☐ 1 vez ☐ Algumas vezes ☐ Muitas vezes ☐ Nunca
Especifique caso sua resposta não seja "nunca": _____ _____ _____	
17) Você sentiu necessidade de combinar duas ou mais sugestões de agrupamento de componentes para que os componentes formassem um único agrupamento?	☐ 1 vez ☐ Algumas vezes ☐ Muitas vezes ☐ Nunca
Especifique caso sua resposta não seja "nunca": _____ _____ _____	
18) A ferramenta é de fácil utilização?	☐ Sim ☐ Não ☐ Parcialmente
Especifique caso sua resposta seja "não" ou "parcialmente": _____ _____	