A ENGENHARIA DE APLICAÇÕES NO CONTEXTO DA REUTILIZAÇÃO BASEADA EM MODELOS DE DOMÍNIO

Nelson Miler Júnior

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 MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:

_____
Profª. Cláudia Maria Lima Werner, D.Sc.

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

_____
Profª. Itana Maria de Souza Gimenes

RIO DE JANEIRO, RJ – BRASIL
JULHO DE 2000

MILER JUNIOR, NELSON

A Engenharia de Aplicações no Contexto da Reutilização Baseada em Modelos de Domínio [Rio de Janeiro] 2000

VIII, 98 p. 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia de Sistemas e Computação, 2000)

Tese – Universidade Federal do Rio de Janeiro, COPPE

1. Engenharia de Aplicações
2. Reutilização
3. Processos de desenvolvimento de software
4. Infra-estrutura de suporte ao desenvolvimento de software
5. Projeto Odyssey

I. COPPE/UFRJ II. Título (série)

A minha esposa

e a meus pais...

Agradecimentos

À Professora Cláudia Werner por me apresentar a reutilização de software, por me guiar pelos caminhos da vida acadêmica, por orientar minha tese, por participar da banca examinadora da minha tese, por sua amizade e, principalmente, por sua infinita paciência e compreensão com os percalços ocorridos no decorrer deste trabalho.

Aos Professores da área de Engenharia de Software da COPPE, Guilherme Travassos e Ana Regina da Rocha, pela contribuição ao meu aprendizado no decorrer do curso.

À Professora Marta Mattoso por participar da banca examinadora da minha tese.

À Professora Itana Gimenes por participar da banca examinadora da minha tese.

À Regina Braga por sua contribuição na definição dos processos contidos neste trabalho e por sua paciência nos vários momentos em que precisei de sua ajuda.

Aos colegas da COPPE, Marcio Barros, Leonardo Murta, Alexandre Dantas, José Ricardo Xavier, Monica Roseti, Alexandre Correa e Robson Souza pela contribuição no trabalho, além da amizade, incentivo e motivação.

Aos demais colegas da COPPE pelo incentivo e motivação.

Ao CNPQ pelo apoio financeiro.

E, finalmente, mas não menos importante, a Deus por permitir a conclusão de mais esta etapa importante da minha vida.

Meus sinceros agradecimentos.

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

A ENGENHARIA DE APLICAÇÕES NO CONTEXTO DA REUTILIZAÇÃO BASEADA EM MODELOS DE DOMÍNIO

Nelson Miler Júnior

Julho/2000

Orientadora: Cláudia Maria Lima Werner

Programa: Engenharia de Sistemas e Computação

Este trabalho define as atividades básicas de um processo genérico de Engenharia de Aplicações, detalhando as tarefas mais importantes envolvidas num processo de desenvolvimento baseado em Reutilização como este.

Neste contexto, também são abordados as adaptações necessárias ao processo de modelagem de domínios, para que este se torne totalmente compatível com a proposta aqui contida, o relacionamento entre os processos de Engenharia de Aplicações e Engenharia de Domínio e a implementação da infra-estrutura de suporte correspondente.

As propostas desta tese foram realizadas no contexto do Projeto Odyssey, em desenvolvimento na COPPE/UFRJ.

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

APPLICATION ENGINEERING IN THE CONTEXT OF REUSE BASED ON DOMAIN MODELS

Nelson Miler Júnior

July/2000

Advisor: Cláudia Maria Lima Werner

Department: Computer and System Engineering

This work presents the activities involved in a generic Application Engineering process. It details the most important tasks that are present in a reuse-based development process.

In this context, we have also approached the adjustments needed to the domain modeling process, for being totally compatible with the proposal of this work, the relationship between the Application and Domain Engineering processes, and the implementation of the corresponding support structure.

The proposals of this thesis were done in the context of the Odyssey Project, under developed at COPPE/UFRJ.

Índice

Capítulo 1 – Introdução **1**
1.1 – A Engenharia de Aplicações 1
1.2 – Objetivos da Tese 2
1.3 – A Organização da Tese 3

Capítulo 2 – Engenharia de Aplicações **5**
2.1 – Introdução 5
2.2 – Engenharia de Aplicações: Conceitos Básicos 8
2.2.1 – Criação de componentes 8
2.2.2 – Seleção de componentes do domínio 10
2.2.3 – Ligação semântica entre modelos de domínio 11
2.2.4 – Arquiteturas de software 12
2.3 – Uma classificação para as abordagens de EA 13
2.3.1 – Desenvolvimento Baseado em Componentes 13
2.3.2 – Arquiteturas Específicas de Domínio 17
2.3.3 – Linhas de Produção 19
2.3.4 – Abordagens mistas e outras técnicas 21
2.4 – Problemas encontrados nas abordagens de ED/EA atuais 23
2.5 – Exemplos de abordagens existentes de EA 25
2.5.1 – OOSE/RSEB 25
2.5.2 – EDLC/KBRET 26
2.5.3 – MRAM/TRAM 28
2.6 – Conclusão 29

Capítulo 3 – Atividades de um processo genérico de Engenharia de Aplicações **30**
3.1 – Introdução 30
3.2 – Processos adotados no Odyssey 31
3.2.1 – Engenharia de Domínio 32
3.2.2 – Engenharia de Aplicações 35
3.3 – O Modelo de Features Estendido 39
3.3.1 – Taxonomia 41
3.3.1.1 – Features Funcionais/Conceituais 41
3.3.1.2 – Features de Projeto 42
3.3.1.3 – Features Implementacionais 42
3.3.2 – Relacionamentos 43
3.3.3 – Outras Características 44
3.3.4 – Exemplo 45
3.4 – O Processo de Engenharia de Aplicações: Odyssey-EA 46
3.4.1 – Planejamento e Análise de Risco 47
3.4.2 – Análise da aplicação 48
3.4.3 – Projeto da aplicação 54
3.4.4 – Implementação da aplicação 59
3.5 – Infra-estrutura de suporte ao processo de EA 62
3.6 – Conclusão 65

Capítulo 4 – Uma Infra-estrutura de suporte à Engenharia de Aplicações no contexto do Projeto Odyssey **67**
4.1 – Introdução 67

4.2 – O contexto do Projeto Odyssey 68
4.3 – A infra-estrutura de suporte Odyssey 70
4.3.1 – Diagramadores genéricos 71
4.3.2 – Documentação de componentes de domínio 72
4.3.3 – Geração de código 72
4.3.4 – Localização dos componentes de domínio 73
4.3.5 – Definição e instanciação de arquitetura de software 74
4.3.6 – Avaliação de padrões e anti-padrões de projeto 75
4.3.7 – Planejamento e Análise de Riscos 75
4.3.8 – Gerência do processo de desenvolvimento 76
4.4 – A infra-estrutura de suporte ao processo de Engenharia de Aplicações implementada 77
4.4.1 – Extensões para suporte ao Modelo de Features Estendido 77
4.4.2 – Suporte ao Processo de Engenharia de Domínio 78
4.4.3 – Suporte ao Processo de Engenharia de Aplicações 84
4.5 – Conclusão 89

Capítulo 5 – Conclusão **91**
5.1 – Visão Geral 91
5.2 – Contribuições da abordagem 91
5.3 – Deficiências da proposta e tópicos para pesquisa futura 92

Bibliografia **94**

Capítulo 1

Introdução

1.1 A Engenharia de Aplicações

A reutilização de software caracteriza-se pela utilização de produtos de software desenvolvidos, ao longo de um ciclo de vida, em uma situação diferente da qual foram originalmente produzidos (FREEMAN, 1980). Por este enfoque, o conceito de reutilização significa muito mais do que reaproveitar código, pois especificações, projetos, planos de testes, entre outros, também podem ser objetos desta técnica.

A Engenharia de Aplicações (EA) se dedica ao estudo das melhores técnicas, processos e métodos para a produção de aplicações, no contexto do desenvolvimento baseado em reutilização (GRISS *et al.*, 1998). Este processo de reutilização é feito sobre produtos de uma área de conhecimento específica, denominada domínio de aplicação. Um domínio de aplicação é um contexto de desenvolvimento, com escopo bem definido, para o qual serão desenvolvidos componentes reutilizáveis, através de um processo conhecido como Engenharia de Domínio (ED).

Segundo o Army Reuse Center (SIMOS e ANTHONY, 1998), a ED é o processo de busca, análise, manipulação e formalização das informações do domínio relevantes para a implementação de um programa de reutilização, que promova o desenvolvimento de aplicações do domínio a um custo reduzido. Os modelos de domínio são o resultado final de uma atividade de Engenharia de Domínio (BRAGA e WERNER, 1999) e servem de referência para a construção de componentes reutilizáveis e de aplicações a partir destes mesmos componentes. Um modelo de domínio captura características comuns e variáveis dentro de uma família de produtos de software numa área específica de aplicação (GRISS *et al.*, 1998).

Nos últimos anos, atingir a maturidade do processo de desenvolvimento de software é uma meta estabelecida por diversos grupos, após terem tido resultados frustrantes ao longo de duas décadas de aplicação de novas metodologia e tecnologias de software (PAULK *et al.*, 1993). Assim, diversas propostas de processos de desenvolvimento foram apresentadas na literatura, na última década (e.g. COAD e

YOURDON, 1992, ROCHA *et al.*, 1996, JACOBSON *et al.*, 1997, JACOBSON *et al.*, 1999), com relatos variados de resultados em sua aplicação.

Todavia, apesar de existirem atualmente diversas propostas de processos de desenvolvimento, muitas delas já adaptadas para o contexto de reutilização baseada em ED/EA, há na maior parte delas deficiências no aspecto do relacionamento entre construir componentes e reutilizá-los, e uma deficiência no nível de detalhamento do processo de construção da aplicação (MANNION *et al.*, 1999). Assim, é necessário um melhor detalhamento do processo de desenvolvimento de uma aplicação através da reutilização de componentes (EA), bem como um maior destaque aos aspectos que interferem no relacionamento entre os processos de ED e EA, como o tratamento de modelos não integrados, por exemplo.

1.2 Objetivos da Tese

O objetivo principal desta tese é o de identificar e descrever as principais atividades envolvidas em um processo genérico de EA, para utilização no contexto do Projeto Odyssey. Este processo deve ser completamente integrado e compatível com o processo de ED do projeto, previamente definido em (BRAGA e WERNER, 1999).

O Projeto Odyssey teve início em 1998, pelo grupo de Reutilização do Programa de Engenharia de Sistemas e Computação (PESC) da COPPE/UFRJ (BRAGA *et al.*, 1999, WERNER *et al.*, 1999). Este projeto tem como objetivo a construção de uma infra-estrutura de reutilização baseada em modelos de domínio, que permitirá a construção de componentes reutilizáveis para uma determinada família de sistemas e sua posterior utilização no desenvolvimento de aplicações.

Os processos de ED e EA do Odyssey são baseados em conceitos já utilizados na literatura, como orientação a objetos (OO) (BOOCH *et al.*, 1998), *use-cases* (JACOBSON *et al.*, 1992) e *features*[1] (KANG *et al.*, 1990). Várias propostas de processos de desenvolvimento utilizam hoje estes conceitos, porém na maior parte das vezes elas são neutras e, algumas vezes, até hostis à reutilização (GRISS *et al.*, 1994). O que ocorre nestes casos é que o processo de reutilização é encarado como uma simples tarefa adicional, anexada ao processo de desenvolvimento de forma trivial. Tendo estes

[1] *Features* são características essenciais do domínio (KANG *et al.*, 1990), e representam uma forma de captura da informação semântica do mesmo (SIMOS e ANTHONY, 1998).

fatores em vista, os processos definidos para o Odyssey são fortemente focados nos aspectos inerentes à tarefa de reutilizar componentes de software.

Nesta tese, buscamos o detalhamento das atividades identificadas do processo de EA. Este detalhamento inclui aspectos pouco citados na literatura, como: política de seleção de componentes reutilizáveis e o relacionamento entre as diferentes atividades relacionadas do processo de EA e destas com as do processo de ED.

A aplicação efetiva destas propostas depende da adoção de um ferramental adequado. Deste modo, é também objetivo deste trabalho a implementação de uma infra-estrutura de suporte ao processo proposto. Hoje, as ferramentas de suporte são parte integrante e fundamental de qualquer processo de desenvolvimento (JACOBSON *et al.*, 1999), sendo sua disponibilidade fator fundamental na utilização de um determinado método ou processo.

1.3 A Organização da Tese

Esta tese está organizada da seguinte forma: Neste primeiro capítulo, é feita uma breve introdução à Engenharia de Aplicações, identificando algumas motivações e problemas nas atuais abordagens. Além disso, são descritos os objetivos da tese, dentro do contexto do Projeto Odyssey, e a organização deste trabalho.

No capítulo 2, é feita uma revisão da literatura atual em EA, identificando e descrevendo as principais abordagens e propostas de EA hoje existentes. Assim, neste capítulo, descrevemos os conceitos básicos envolvidos nestas abordagens, citando alguns exemplos representativos. Além disso, são mostrados certos problemas identificados na implantação destas propostas.

No capítulo 3, descrevemos uma proposta detalhada das atividades de um processo genérico de Engenharia de Aplicações, supondo sua adoção no contexto de uma infra-estrutura de reutilização baseada em modelos de domínio, como o Odyssey. Além disso, são discutidos neste capítulo as extensões necessárias ao processo de ED correspondente, como o Modelo de *Features* Estendido, e os requisitos que uma infra-estrutura de suporte deve possuir para atender tanto ao processo de EA, quanto às extensões à ED.

No capítulo 4, é feita uma descrição da implementação da infra-estrutura de suporte ao processo proposto no capítulo 3, especialmente com relação às atividades

ligadas a fase de análise da aplicação. Neste capítulo, também mostramos um panorama geral do ferramental atualmente disponível no Odyssey (WERNER *et al.*, 1999), bem como a forma na qual esta nova parte da infra-estrutura foi agregada àquela já existente. Um exemplo de utilização desta ferramenta para a construção de novas aplicações também é mostrado neste capítulo.

Finalmente, no capítulo 5, é feita uma conclusão sobre o trabalho desenvolvido nesta tese. As soluções apresentadas são sintetizadas, identificando as contribuições, os méritos e os tópicos que ainda necessitam de um trabalho adicional. Além disso, também é feita uma discussão sobre o trabalho futuro necessário para a evolução do processo atual de Engenharia de Aplicações e da infra-estrutura de apoio a ele.

Capítulo 2

Engenharia de Aplicações

2.1 Introdução

Mudanças no desenvolvimento de software nos últimos anos, causadas pela necessidade de produtos com funcionalidades mais complexas e de aplicação mais específica, desenvolvidos em curto espaço de tempo e com custo cada vez mais reduzido, têm criado um ambiente de contínuos desafios para a comunidade de Engenharia de Software. Para tratar estes desafios, a tendência tem sido a aceleração do processo de produção de software. Por outro lado, a complexidade do software construído não para de aumentar, afetando adversamente sua confiabilidade e flexibilidade (DIGRE, 1998).

Entre as estratégias para facilitar o alcance de objetivos tão diversos, se destacam algumas abordagens de Reutilização, como a combinação Engenharia de Domínio/Engenharia de Aplicações. Entre os mecanismos utilizados por estas abordagens, para atingir os objetivos mencionados, estão técnicas como o desenvolvimento baseado em componentes (DIGRE, 1998), as arquiteturas de software (ARPA, 1994) e os cenários e casos de uso (JACOBSON *et al.*, 1992).

Segundo KRUEGER (1992), para que uma técnica de reutilização de software possa ser efetiva, ela deve reduzir a distância cognitiva entre o conceito inicial de um sistema e sua implementação final. Sendo assim, no contexto de um processo de desenvolvimento de software baseado em reutilização, o reuso nas fases iniciais do desenvolvimento deve ter como resultado o reuso de componentes na implementação da aplicação, ou seja, deve haver uma relação estreita entre os conceitos de um domínio nas fases iniciais do processo de desenvolvimento e os componentes codificados que serão utilizados na implementação da aplicação (BRAGA e WERNER, 1999). Assim, para que o processo de reutilização seja efetivo, devemos utilizar um processo de ED capaz de criar componentes de qualidade em todas as fases do desenvolvimento e um processo correspondente de EA que seja capaz de utilizá-los.

A ED e a EA têm como principal objetivo a utilização de uma abordagem baseada em modelos de domínio para facilitar a implantação da reutilização e sua

utilização em cadeia no desenvolvimento de software. Assim, a ED deve permitir a criação e posterior evolução de modelos de domínio, apresentando-se, portanto, como um processo intuitivo, interativo e iterativo (SIMOS e ANTHONY, 1998). Neste sentido, as informações do domínio coletadas durante o processo devem ser organizadas, manipuladas e validadas pelos engenheiros e especialistas do domínio durante todo o processo, para assegurar resultados mais confiáveis.

Num processo de ED, é fundamental o papel do Engenheiro do Domínio e do Especialista no domínio de aplicação. O Engenheiro do Domínio e sua equipe são os responsáveis pela modelagem e construção do domínio e de todos os seus componentes. O especialista entende os conceitos e procedimentos relativos da área de conhecimento, fornecendo as informações necessárias para que esta possa ser modelada.

No desenvolvimento de um domínio (ou família de sistemas), é necessário considerar atividades envolvidas no desenvolvimento tradicional de sistemas (como análise de requisitos, projeto e implementação) como atividades do processo de modelagem do domínio. No contexto de ED, estas atividades são conhecidas como Análise do Domínio, Projeto do Domínio e Implementação do Domínio.

O processo de EA atua de forma paralela ao de ED, sendo que é na EA que os componentes do domínio são efetivamente utilizados na construção de aplicações reais. Todavia, o processo de EA (fortemente integrado ao processo de ED) tem sido deixado em segundo plano por boa parte das atuais abordagens encontradas na literatura (MANNION *et al.*, 1999), necessitando hoje de uma maior formalização e detalhamento nas atividades que o compõem. A Tabela 2.1 mostra as principais atividades envolvidas em cada uma das fases dos dois processos.

Num processo de EA, sistemas específicos podem ser gerados a partir do modelo de domínio, instanciando-o de acordo com os requisitos da aplicação. Com um modelo de domínio claro e bem definido, o Engenheiro da Aplicação pode desenvolver a especificação de sua aplicação a partir de componentes definidos previamente, não precisando realizá-la a partir do zero a cada vez que uma nova aplicação num dado domínio precisar ser construída (GOMAA *et al.*, 1992). Logo, o Engenheiro da Aplicação é o responsável pelo desenvolvimento da aplicação a partir dos componentes reutilizáveis construídos pelo processo de ED (SIMOS e ANTHONY, 1998).

Por todos estes fatores, na última década, as áreas industrial e acadêmica têm mostrado interesse na reutilização de domínios de aplicação como telefonia, aviação,

máquinas industriais, governamental, etc. Esta forma de reutilização é conhecida como Reutilização Vertical (CHEONG e JARZABECK, 1999). No contexto industrial, a construção destas famílias de sistemas tem sido denominada Linha de Produção (MAYMIR-DUCHARME, 1997). Abordagens que suportam este enfoque vão desde bibliotecas de componentes reutilizáveis até arquiteturas genéricas de software e geradores de aplicações. Estas abordagens diferem no que se refere à representação semântica do domínio, no processo utilizado para analisar e construir seus componentes e no papel que estes desempenham na construção de aplicações específicas, no contexto do domínio modelado.

	Engenharia de Domínio	**Engenharia de Aplicações**
Análise	Definição do escopo do domínio, Descrição dos componentes do domínio, Modelagem do diagrama de integração dos componentes, Especificação dos modelos estruturais do domínio (OO).	Definição do escopo da aplicação, Seleção dos componentes do domínio, Adequação dos componentes do domínio, Especificação de novos componentes.
Projeto	Definição das arquiteturas suportadas no domínio, Refinamento dos modelos estruturais, Especificação da interface básica entre os componentes.	Definição e Instanciação da arquitetura da aplicação, Refinamento dos modelos estruturais e de interação da aplicação, Adaptação da estrutura dos componentes de domínio, Especificação da interface básica dos novos componentes.
Implementação	Codificação dos componentes, Testes dos componentes, Empacotamento do código legado.	Adaptação dos componentes de domínio, Desenvolvimento de novos componentes, Instanciação da aplicação, Testes da aplicação.

Tabela 2.1: Atividades dos processos de ED e EA

Este capítulo tem como objetivo apresentar e analisar as diferentes propostas para processos de EA, encontradas na literatura atual. Conforme dito anteriormente, este processo tem sido menos detalhado por seus autores, em comparação ao processo de modelagem de domínio, não merecendo muitas vezes processos diferenciados e metodologias detalhadas para suas atividades mais importantes. Isto ocorre, aparentemente, por uma observação de que este processo parece ser muitas vezes intuitivo e de fácil execução, se comparado à modelagem e construção de um domínio.

Neste capítulo, apresentamos inicialmente os conceitos básicos envolvidos na área de EA (seção 2.2). Em seguida, relatamos os diferentes tipos de abordagens utilizadas nos atuais processos de EA (seção 2.3), descrevendo suas características, dificuldades e benefícios. Os problemas e soluções particulares encontrados na aplicação prática da reutilização baseada em ED/EA são discutidos na seção 2.4. Na seção 2.5, são citados alguns exemplos representativos dos conceitos e tipos de abordagens citados neste trabalho. Finalmente, na seção 2.6 é feita uma conclusão sobre o capítulo.

2.2 Engenharia de Aplicações: Conceitos Básicos

Esta seção apresenta alguns aspectos importantes para as atividades de um processo de EA (e ED, consequentemente), identificando os requisitos necessários para que este tipo de abordagem possa ser utilizado num processo efetivo de construção de aplicações com reutilização. São eles: a forma de criação de componentes do domínio, a política de seleção de componentes para a aplicação, a ligação semântica entre os modelos que representam o domínio e a modelagem arquitetural do domínio/aplicação.

2.2.1 Criação de componentes

Segundo FISCHER (1992), nas pesquisas realizadas em Engenharia de Software, para resolver problemas de desenvolvimento de sistemas, existe uma maior preocupação com a transição da especificação para a implementação (chamadas de atividades *downstream*) do que como e de que forma as especificações correspondem realmente ao problema (atividades *upstream*), atividade chave num processo de modelagem de domínio. Estas duas atividades são complementares e precisam ser coordenadas, apesar de envolverem diferentes grupos de profissionais e requererem metodologias e mecanismos de suporte diversos.

Algumas das primeiras propostas para tratar o problema da modelagem e reutilização de domínios foram genericamente denominadas Análise de Domínio (NEIGHBORS, 1981, KANG *et al.*, 1990, PRIETO-DIAZ e ARANGO, 1991). A Análise de Domínio consiste numa metodologia que permite ao desenvolvedor capturar, confiável e sistematicamente, as informações necessárias para maximizar a reutilização no processo de desenvolvimento (PRIETO-DIAZ e ARANGO, 1991). O domínio em

questão pode ser visto como uma coleção de aplicações (ou componentes) que compartilham características comuns. Assim, o resultado do processo de análise de domínio é uma taxonomia de aplicações, que demonstra todas as similaridades e variações entre os sistemas estudados, organizando seus componentes comuns (ARANGO, 1994).

A principal limitação destas primeiras propostas de modelagem de domínio era a forma de representação do resultado, que muitas vezes apresentava-se numa estrutura não diretamente instanciável para o processo de produção de uma aplicação real (GRISS *et al.*, 1998). Assim, mais recentemente outras propostas (JACOBSON *et al.*, 1997, KANG *et al.*, 1998) surgiram, propondo uma correspondência mais direta entre os modelos de domínio e os componentes reutilizáveis, inclusive com relacionamentos entre requisitos abstratos e características arquiteturais e implementacionais. Estas propostas mais abrangentes de modelagem adotaram a nomenclatura: Engenharia de Domínio, proposta por ARANGO (1988).

O principal objetivo de um processo de Engenharia de Domínio é disponibilizar um modelo de domínio, com informações que possam ser reutilizadas durante todo o processo de desenvolvimento de uma aplicação (CIMA, 1996). PRIETO-DIAZ e ARANGO (1991) definem que é importante que o modelo do domínio contenha informações sobre os conceitos que possibilitam a especificação de sistemas, planos descrevendo como mapear estas especificações abstratas em código e a lógica para relacionamento entre os componentes do domínio. Assim, o modelo resultante do processo de ED deve representar as diversas características que compõem um domínio (como funcionalidades, conceitos, arquiteturas, entre outros), nos seus diferentes níveis de abstração.

O processo de modelagem do domínio se apresenta de várias maneiras distintas (CIMA, 1996), mas de uma forma geral podemos dizer que o conhecimento sobre o domínio pode ser visto de duas formas: como regras ou teorias que caracterizam os problemas reais (FISCHER, 1992) ou como modelos que representam uma coleção de componentes de software do domínio (GRISS *et al.*, 1998). Esta última visão, que prevê a modelagem de domínio através da especificação detalhada de seus componentes, funcionais e conceituais, utiliza vários conceitos oriundos de técnicas de desenvolvimento baseado em componentes (DIGRE, 1998). O enfoque baseado em componentes é considerado, hoje, como bastante adequado para propostas que visam

maximizar a reutilização, e já possui algumas propostas descritas na literatura (GRISS *et al.*, 1998, VICI, 1998).

2.2.2 Seleção de componentes de domínio

Um aspecto fundamental no processo de reutilização dos conceitos e funcionalidades de um domínio é a tarefa de seleção dos componentes que irão compor as aplicações construídas no seu contexto. Apesar desta tarefa parecer ser a primeira vista simples, uma série de fatores influencia esta atividade do desenvolvimento, podendo inclusive acarretar problemas em fases posteriores do processo, caso seu resultado seja inconsistente.

A técnica mais utilizada nos processos atuais de Engenharia de Aplicações é a Seleção Livre de Requisitos (MANNION *et al.*, 1999). Seleção livre significa permitir que o Engenheiro da Aplicação explore o modelo do domínio e, simplesmente, copie os componentes de qualquer ponto desta estrutura para a sua aplicação. Este enfoque traz alguns benefícios de curto prazo, como a ausência de um modelo global de componentes (este poderia ser substituído por um tipo de classificação facetada (PRIETO-DIAZ e ARANGO, 1991), diminuindo o número de digramas do domínio) e o conseqüente tempo reduzido de modelagem do relacionamento entre os componentes (poupando algum tempo nas primeiras fases do processo de ED).

Porém, existem outros aspectos que devem ser discutidos na escolha de uma técnica de seleção de componentes de domínio. A escolha de um requisito isolado não é adequada na maior parte das vezes, já que os componentes do domínio possuem uma série de relacionamentos entre si, mesmo que estes não estejam explicitados no modelo. Assim, a reutilização de requisitos será mais efetiva quando puder atingir grupos de características ligadas por estes relacionamentos. Quando se tenta isolar um requisito aleatoriamente, ele perde todas as suas dependências e restrições, e estas características podem representar parte importante de seu papel no domínio.

Além disso, a seleção livre gera uma carga adicional de verificação de consistência considerável após cada seleção (MANNION *et al*., 1999). Componentes organizados hierarquicamente, que possuam relacionamentos como mútua exclusão, requisitam um trabalho de verificação após cada escolha. Esta carga de verificação aumenta com o número de componentes já escolhidos e as possíveis restrições (em qualquer ponto do modelo) que eles possam trazer. Similarmente, existem componentes

obrigatórios do domínio e que, portanto, devem estar sempre presentes. A seleção livre pode implicar na ausência destes elementos, o que representaria um erro.

Assim, é importante garantir que sempre que um componente seja escolhido para reutilização, os componentes relacionados também possam ser selecionados. Do mesmo modo, dois componentes que sejam mutuamente exclusivos devem ser impedidos de serem utilizados conjuntamente. Por isso, políticas de seleção de componentes devem ser implementadas em todos os níveis de abstração, especialmente nos mais altos. Esta política precisa levar em conta os relacionamentos e dependências entre os componentes do domínio, conforme discutido acima. Este tipo de abordagem pode sistematizar o processo de escolha de uma boa parte dos componentes, fator este que é muito importante em domínios com um grande número de requisitos (MANNION *et al.*, 1999).

Outra técnica que pode auxiliar na tarefa de localização de componentes para seleção é o uso de técnicas mais complexas, baseadas em inteligência artificial. Já existem na literatura relatos de pesquisas sobre técnicas mais poderosas para a busca de informações relevantes em um domínios de aplicação (RIG, 1999, SEACORD *et al.*, 1998). Agentes inteligentes e programas com algoritmos *fuzzy* são as abordagens mais usadas nestas tarefas (BRAGA *et al.*, 1999).

2.2.3 Ligação semântica entre modelos de domínio

Existem diversos trabalhos na literatura que visam a modelagem e reutilização de componentes de análise, projeto e implementação de software. GOMAA (1995), por exemplo, utiliza diferentes variações no modelo de análise do domínio, valendo-se de diversas técnicas, como hierarquias de agregação e de generalização e diagramas de dependência entre requisitos e objetos, para representar os diferentes aspectos do domínio. Todavia, entender o conjunto destas visões não é uma tarefa simples, devido a diversidade de componentes envolvidos. Por isso, o caminho do problema à solução é muitas vezes difícil de seguir sem algum tipo de auxílio (MANNION *et al.*, 1999).

A maioria das propostas de ED tem foco no desenvolvimento de estruturas de representação de alto nível, com o intuito de capturar informações sobre todos os requisitos abstratos do domínio. Estes modelos podem ser explorados mais tarde por técnicas de desenvolvimento que suportam seu entendimento, análise e aproveitamento de seus componentes, mesmo que de forma manual (KANG *et al.*, 1998). Em geral,

para a tarefa de reutilização dos componentes do modelo, são desenvolvidas ferramentas que auxiliam a recuperação de componentes baseado nestas especificações. Estas abordagens possuem contribuições, como a definição de uma representação semântica adequada para o domínio, porém outros fatores também precisam ser detalhados, como o processo de seleção dos componentes, a descrição da ligação entre os modelos e o empacotamento dos componentes em padrões reutilizáveis para o desenvolvimento de aplicações (MANNION *et al.*, 1999), entre outros.

Por outro lado, as abordagens que visam o detalhamento do processo de especificação dos requisitos de um domínio e da escolha de componentes (requisitos) de alto nível, como o MRAM (MANNION *et al.*, 1999), muitas vezes pecam pelo isolamento de uma única fase. Ou seja, o processo é bem definido, porém o mapeamento do resultado deste processo para as fases posteriores de um processo de EA, como projeto e implementação, nem sempre é simples e automático (ver detalhes na seção 2.3.1).

Estes casos mostram a necessidade da presença de uma característica muito importante nos processos de ED e EA: a *rastreabilidade*. Esta característica representa a capacidade de ligação entre os diferentes modelos e representações que compõem o domínio, permitindo uma conexão entre os diferentes diagramas e entre os diversos níveis de abstração de um domínio. O excesso de técnicas utilizadas e a presença de ferramentas e/ou métodos desconectados do restante do processo de desenvolvimento podem representar um obstáculo a sua utilização num processo de EA. Por isso, é importante que o desenvolvedor possa explorar qualquer tipo de relacionamento, intra e inter-modelos, para que o entendimento e o aproveitamento dos componentes do domínio possa ser realizado de forma facilitada.

2.2.4 Arquiteturas de software

Arquiteturas de software específicas de domínio são utilizadas em várias propostas de ED e EA, para representar aspectos tecnológicos do modelo. Dependendo da abordagem, as arquiteturas podem ocupar vários papéis, desde o de representar apenas a estrutura de projeto do domínio, até o de modelo principal de representação semântica do mesmo. Existem autores que propõem até mesmo arquiteturas ainda mais genéricas, aplicáveis em diversos domínios (POULIN, 1997).

Uma possibilidade de representação de arquiteturas de software específicas de domínio é o uso de *frameworks*. Um *framework* fornece um ambiente de execução para os componentes arquiteturais de domínio. Um *framework* é uma aplicação semi-completa e reutilizável que pode ser utilizada para a produção de sistemas específicos (JOHNSON e RUSSO, 1991). Podemos dizer que os *frameworks*, no contexto da ED, são um tipo especial de arquitetura de software, descrita (e composta) por classes abstratas que colaboram entre si. Além de classes, este tipo de arquitetura pode conter componentes de mais alta granularidade, como contextos e *features* (XAVIER, 2000).

Existem algumas alternativas ao uso de *frameworks* para a representação de arquiteturas de domínio. Ambientes baseados em geração de software, normalmente, possuem abordagens específicas – e muitas vezes formais – de representação. Uma delas é o uso de *frames* (CHEONG e JARZABECK, 1999). Esta abordagem visa o suporte de famílias de sistemas baseada em padrões formais pré-processados, chamados *frames*. Os *frames* surgem da programação de componentes com a adição de pontos de espera e outros parâmetros, através dos quais é gerado o código do software. Os comandos de *frames* estendem ou trocam os valores destes parâmetros em outros *frames*, preenchendo assim o código nos pontos de espera.

2.3 Uma classificação para as abordagens de EA

Como já foi dito anteriormente, diversas abordagens para processos de ED, e consequentemente para EA, já foram propostas na literatura. Nesta seção, analisamos algumas das mais utilizadas, discutindo suas peculiaridades, benefícios, dificuldades e citando alguns exemplos de cada uma delas.

As propostas aqui analisadas foram escolhidas tendo em vista seus processos de EA, ou seja, como (e se) elas detalham a construção de aplicações baseadas em sua modelagem do domínio. A classificação utilizada foi baseada na proposta por GRISS em (1999a) e serve de base para o enquadramento de abordagens mais recentes.

2.3.1 Desenvolvimento Baseado em Componentes

Uma das abordagens mais utilizadas hoje para processos de ED/EA é a do Desenvolvimento Baseado em Componentes (DBC). Conceitualmente, o DBC unifica conceitos de uma série de tecnologias, tais como programação orientada a objetos

(BOOCH *et al.*, 1998), arquiteturas de software (TRACZ, 1994) e, até mesmo, computação distribuída (PRESSMAN, 1997).

Porém, para que este enfoque de desenvolvimento se torne efetivo, ele precisa estar associado ao planejamento da arquitetura da aplicação. BOSCH (1999), por exemplo, realça que o DBC traz a arquitetura do software para o centro do processo de EA. O papel da arquitetura, neste caso, é separar a infra-estrutura da lógica funcional da aplicação.

Outros pontos importantes também devem ser levados em conta num processo de desenvolvimento baseado nesta tecnologia, como ciclo de vida de desenvolvimento, formas de manutenção, adição de funcionalidades e gerenciamento de complexidade. Além disso, é muito importante que se apresente uma forma de definição padrão dos componentes (como *templates* ou uma Linguagem de Definição de Componentes - CDL, no original) que permita sua padronização e uso.

Processos de ED e EA baseados em conceitos de DBC possuem, na maioria das vezes, um modelo de domínio semântico com alto nível de abstração. Este modelo contém os conceitos do domínio de desenvolvimento, o que permite ao Engenheiro da Aplicação especificar o comportamento coletivo de grupos dos componentes da aplicação desde o começo do processo de EA (BROWN e WALLNAU, 1998).

Os componentes do modelo do domínio podem ser descritos através de contratos. Um contrato configura e estrutura os conceitos (e as funcionalidades) do domínio, não expondo detalhes técnicos ou requisitos de performance do software. Assim, o contrato reforça a separação da semântica do domínio e de sua possível implementação (DIGRE, 1998).

Uma maneira típica de encapsular a tecnologia contida nos elementos de infra-estrutura é através de *frameworks* de domínio, isto permite que o contrato de um componente foque somente na semântica específica do componente, deixando de lado seus aspectos tecnológicos. O papel do *framework* é o de garantir o contrato do domínio em todos os seus componentes, na infra-estrutura técnica e nos sistemas clientes. A tecnologia de *frameworks* suporta, ainda, uma série de características desejáveis ao DBC como introspeção, customização, edição de atributos, criação de instâncias de objetos, coleção de tipos, identificação e redirecionamento, que não devem fazer parte do contrato do componente.

O aumento da produtividade na EA, neste caso, é dada pela diminuição da complexidade do uso dos componentes. O número de elementos que devem ser entendidos e manipulados, para o funcionamento correto no uso dos componentes, é fator fundamental para o sucesso no desenvolvimento da aplicação. Vários fatores influem nesta complexidade (DIGRE, 1998):

- Quantidade de informações visíveis: a complexidade aumenta com o número de informações que são expostas através da interface de composição do software, incluindo elementos de dados, tipos de dados e nomes de funções.
- Dependência seqüencial: a complexidade aumenta sempre que existam requisitos que levam os usuários a ter que fazer operações em uma determinada ordem.
- Ambiente: a complexidade aumenta sempre que o usuário dos componentes é responsável por gerenciar o ciclo de vida, persistência, localização e outros aspectos físicos dos componentes, dentro de um contexto de aplicação muito grande. O suporte de uma ferramenta automática é muito importante neste caso.
- Dependências tecnológicas: a complexidade aumenta com a exposição a cada característica tecnológica - como distribuição e forma de armazenamento - ou de interface.
- Concorrência: a complexidade aumenta quando aspectos da concorrência são expostos ao usuário.

Uma CDL reduz a complexidade através do fornecimento de uma abstração orientada a domínio, o que permite ao usuário focar somente no problema da especificação da aplicação. Assim, o encapsulamento dos *frameworks* reduz a complexidade exposta ao reutilizador, porque fatores como ambiente, escopo de responsabilidades e concorrência ficam fora do domínio a ele mostrado.

Para a EA, é importante que a metáfora da CDL seja capaz de suportar componentes em todos os níveis de granularidade e abstração existentes no processo de desenvolvimento. Os componentes existem em níveis de granularidade que vão de muito alto (representando todo um sub-domínio, por exemplo) ao muito baixo (como as primitivas semânticas do domínio).

Para alcançar o reuso em larga escala dentro de um domínio de componentes, várias abordagens advogam a derivação de componentes reutilizáveis implementados de produtos nos níveis mais altos de abstração, como os requisitos de análise (GRISS *et al.*, 1998, JACOBSON *et al.*, 1997). Assim, aplicações específicas são construídas através da seleção de componentes reutilizáveis de alto nível, sendo a primeira tarefa do desenvolvedor, neste caso, selecionar um subconjunto de requisitos do domínio. Porém, como já foi dito anteriormente, a maioria dos métodos não fornece detalhes sobre esta tarefa, tratando-a como uma atividade menor do processo de EA.

Um dos exemplos de proposta de detalhamento desta atividade chave do processo de reutilização é a abordagem MRAM (*Method for Requirements Authoring and Management*), que descreve como produzir um modelo de requisitos de uma família de aplicações e como gerar um modelo para cada aplicação, derivado do modelo genérico do domínio (MANNION *et al*., 1999).

Neste enfoque, uma família de aplicações é composta por um conjunto de requisitos numerados, atômicos e em linguagem natural – um dicionário de domínios e um conjunto de discriminantes. Um discriminante é alguma característica que diferencia um sistema de outro. FODA (KANG *et al.*, 1990) e FeatuRSEB (VICI, 1998) apresentam uma abordagem similar de modelagem, mas neste caso usam *features* e não requisitos como base.

Embora definir os pontos comuns e variações seja um passo importante na maioria dos métodos e processos de ED, o detalhamento é muitas vezes esparso. Uma abordagem proposta para este tipo de análise é o SCV (*Scope, Commonality and Variability*) que visa tentar tornar mais clara esta lacuna (MANNION *et al.*, 1999). A identificação dos requisitos e discriminantes é o principal objetivo desta abordagem.

A idéia por trás desta técnica é que os requisitos reutilizados trazem com eles projetos reutilizáveis, que por sua vez trazem consigo o código reutilizável, numa típica abordagem baseada em componentes, extrapolada para níveis mais altos de abstração no processo de EA. Para isso, o MRAM tenta, através dos discriminantes, reutilizar os requisitos do domínio, executando uma verificação lógica de suas dependências, através dos relacionamentos definidos no processo de ED. O resultado final é o modelo de requisitos da aplicação, que neste caso ainda carece de maior suporte para integração às demais fases do processo de desenvolvimento.

2.3.2 Arquiteturas Específicas de Domínio

Na abordagem baseada em arquiteturas específicas de domínio, tenta-se construir uma arquitetura que seja representativa de uma família de aplicações. Este processo é feito geralmente de forma *top-down*, baseado em informações obtidas durante o processo de Análise de Domínio. Uma das propostas mais conhecidas para suportar esta abordagem é conhecida como *Domain Specific Software Architecture* (DSSA) (ARPA, 1994).

O processo de EA desta abordagem se inicia através da especificação de requisitos para um sistema específico. Em geral, isto é feito através do refinamento dos requisitos genéricos do domínio. O modelo, gerado nesta atividade, e a arquitetura precisam estar integrados (seja automaticamente ou através de uma forma manual de relacionamento) para que o fluxo de informação necessário ao desenvolvimento da aplicação possa ser estabelecido.

A visão predominante neste enfoque é a de que a arquitetura de domínio é a conexão entre a arquitetura da aplicação e a análise de domínio (CZARNECKI, 1997). Assim, a arquitetura de um domínio consiste de um modelo do domínio, incluindo uma descrição dos requisitos genéricos, e de uma arquitetura de referência.

Um exemplo de proposta baseada neste tipo de enfoque pode ser encontrado no projeto TINA (Telecommunications Information Networking Architecture), descrito em (CZARNECKI, 1997). O seu processo de EA é chamado Metodologia de Desenvolvimento de Serviços. As fases deste processo incluem: definir objetivos empresariais, definir requisitos da aplicação, configuração da aplicação (com desenvolvimento dos novos componentes necessários) e depuração da aplicação. O processo de EA pode também ser implementado usando ainda características de *workflow* e repositórios de componentes distribuídos, presentes no ambiente de suporte. A arquitetura do domínio é representada por um DSSA, e é ela que guia o processo de desenvolvimento da aplicação.

Um outro exemplo representativo do uso de arquiteturas de software é dado pelo processo EDLC (Evolutionary Domain Life Cycle). Neste caso, a arquitetura é definida por uma Linguagem para Descrição de Arquiteturas (ADL, *Architecture Definition Language*, no original) e é organizada em níveis: independente do domínio, específica do domínio e específica da aplicação. A arquitetura é composta de padrões de arquitetura de domínio reutilizáveis (do tipo caixa preta) e de padrões caixa-branca

extensíveis. A implementação desta arquitetura consiste num *framework* que pode evoluir e ser configurado para permitir que novas funcionalidades sejam agregadas à arquitetura (GOMAA e FARRUKH, 1999).

Conforme dito anteriormente, uma alternativa ao uso de *frameworks* como base da arquitetura, pode ser a utilização de *frames* configuráveis. Durante a EA, seleciona-se antecipadamente os requisitos das aplicações do domínio e especificam-se os requisitos não esperados num *frame* de especificação que formará uma raiz (ou hierarquia) de *frames*. Um estudo qualitativo (CHEONG e JARZABECK, 1999) mostrou que esta tecnologia pode levar à redução do tempo de produção (70%) e dos custos do projeto (84%), além de aumentar a efetividade do processo de reuso.

Assim, durante o processo de ED ocorrem as atividades de construção da biblioteca de reuso como: especificação da família de sistemas, planejamento da construção da arquitetura reutilizável, e implementação do *framework* (ou estrutura análoga) que será reutilizado posteriormente.

Durante a EA, as atividades que ocorrem são as seguintes:

* Especificação da aplicação. Dados os requisitos de uma aplicação específica (contida no domínio), a especificação da aplicação é gerada através de um corte na especificação reutilizável do domínio.
* Composição da arquitetura da aplicação. Feita a especificação e com a ajuda da arquitetura reutilizável do domínio, a arquitetura da aplicação é gerada a partir desta última.
* Configuração da aplicação. Baseado na arquitetura, os tipos de componentes que devem ser incluídos na aplicação são selecionados da biblioteca de reuso. Dados os parâmetros da aplicação, uma primeira versão é composta através da instanciação dos componentes escolhidos, sua interconexão e seu mapeamento para a configuração de hardware.

O que diferencia esta abordagem das demais aqui citadas é o uso da arquitetura como estrutura chave no processo de geração da aplicação, desde os primeiros momentos do processo de EA. Outras abordagens utilizam as arquiteturas apenas como uma técnica de definição da estrutura do aplicação, na fase de projeto. Assim, as arquiteturas específicas de domínio levam a arquitetura para um nível de abstração mais alto. O principal benefício é o de tratar assuntos relativos às restrições arquiteturais num

momento anterior do desenvolvimento. Por outro lado, muitas vezes esta abordagem adiciona uma complexidade maior ao primeiro contato do desenvolvedor com o domínio.

Um último enfoque do processo baseado em arquiteturas é dado pelo método OODE (*Object-Oriented Domain Engineering*). Esta abordagem insere mais uma fase no processo de ED: a Engenharia de Distribuição (CHAN e LAMMERS, 1998). O ciclo de vida do OODE é dividido em três fases, num modelo espiral: Análise de Domínio (OODA, no original), Projeto de Domínio (OODD) e Engenharia de Distribuição do Domínio (OODED).

O intuito da fase de distribuição do domínio é preparar a arquitetura do domínio (*framework* orientado a objetos) para que ela possa ser instanciada como uma aplicação pronta para ser utilizada, completa e com suas respectivas ferramentas, técnicas de treinamento e suporte, e não somente uma ligação entre um conjunto de componentes de software.

A abordagem de arquiteturas específicas de domínio é uma técnica que pode ser combinada com outras, como DBC (GRISS, 1999b) e geração formal de software (KANG *et al.*, 1998). A maioria das propostas deste tipo utiliza de fato conceitos de outras técnicas para facilitar atividades específicas do processo de EA.

2.3.3 Linhas de Produção

Uma das abordagens de ED e EA mais estudadas recentemente tem sido a de Linhas de Produção (*Product Lines*, no original). Esta técnica privilegia, como o próprio nome diz, a produção em série de aplicações de um determinado domínio, visando obter um alto grau de reutilização.

Apesar disso, um entendimento sobre a relação e as diferenças entre os conceitos de domínios, linhas de produção e arquiteturas de software aparentemente não existe. Uma parte desta dúvida vem do nível de abstração que cada um destes componentes representa. Na maior parte das vezes, “Linhas de Produção” refere-se ao mesmo nível que “Domínio” e os dois termos existem apenas para comunicar a mesma idéia para duas audiências distintas: Acadêmica e Empresarial (POULIN, 1997). Estes conceitos são bastante antigos, PARNAS se referiu a uma coleção de sistemas que compartilha características comuns como uma família de sistemas em (1972).

A definição de linha de produção também provoca controvérsia. No painel sobre linhas de produção de um simpósio de reuso (GRISS, 1999b) foi perguntado aos participantes a diferença entre linhas de produção e domínios. Todos os participantes deram diferentes respostas. A mais aceita foi: “Não há diferença entre os conceitos de linha de produção e domínio, exceto pelo fato de que a primeira idéia tem a conotação de mercado, enquanto a segunda não”.

As organizações que constróem sistemas altamente complexos, ou extremamente específicos, precisam de uma arquitetura do modelo de domínio e procedimentos para implementá-la como uma forma de reuso de componentes em larga escala. Esta definição é dada por GRISS (1999b) para a necessidade de utilização de linhas de produção. Porém, ela dá ênfase a um objetivo geral de qualquer processo de reutilização, que é o de maximizar a taxa de reaproveitamento dos produtos de software.

Uma outra definição de Linha de Produção é dada em (CLEMENS, 1997): “É um grupo de produtos similares dentro de um segmento de mercado. Já uma família de produtos é um conjunto de produtos que compartilham seus projetos e padrões. Uma linha de produção, construída tendo como base uma família de produtos bem definida, maximiza a taxa de reuso, produzindo uma economia no processo de produção de software”.

Uma linha de produção tenta simplificar o processo de reuso. Para isso, ela tenta inserir no processo uma grande variedade de produtos, incluindo requisitos, a análise e o projeto (arquitetura) do software, sistemas prontos com seus componentes, documentação, estratégias de testes e outros dados relevantes. Além disso, o conhecimento e a perícia do pessoal do projeto, os processos, métodos, ferramentas usados para desenvolver um software também devem poder ser reutilizados.

A estrutura de uma linha de produção de software é, geralmente, uma arquitetura em níveis clássica. Um exemplo é a arquitetura da CesciusTech para a linha de produção SS2000 (CLEMENS, 1997). No nível básico, encontra-se uma máquina virtual que encapsula a plataforma de hardware utilizada, sistema operacional e a comunicação entre processos. Serviços genéricos de suporte ao domínio, como distribuição, são fornecidos no nível acima. Aplicações típicas, que compõem a maioria dos produtos desenvolvidos no domínio, formam o próximo nível e as características específicas de cada aplicação formam o nível mais alto (Figura 2.1). Os níveis e seus

respectivos conteúdos devem ser consistentes e fazer uso da abstração e encapsulamento, promovendo assim um ganho adicional na facilidade de reutilização.

Uma idéia polêmica por trás de várias propostas de linhas de produção é a união dos desenvolvimentos *para* e *com* reutilização. O objetivo é reutilizar automaticamente todo software desenvolvido pela empresa. Além disso, segundo os autores, há um ganho com a utilização de uma só equipe de desenvolvimento, abandonando a necessidade de profissionais dedicados somente à construção de componentes reutilizáveis.

Porém, este tipo de abordagem já foi questionada, há algum tempo, por diversos pesquisadores na área de reutilização, com argumentos como o de que o desenvolvimento de componentes confiáveis exclusivamente *para* reutilização evita a incidência de erros inseridos no processo de modelagem de necessidades específicas de uma aplicação e de que nem todos os componentes desenvolvidos tendo em vista as necessidades de uma aplicação específica podem ser adequados para um processo genérico de reutilização (WERNER, 1995, JACOBSON *et al.*, 1997).

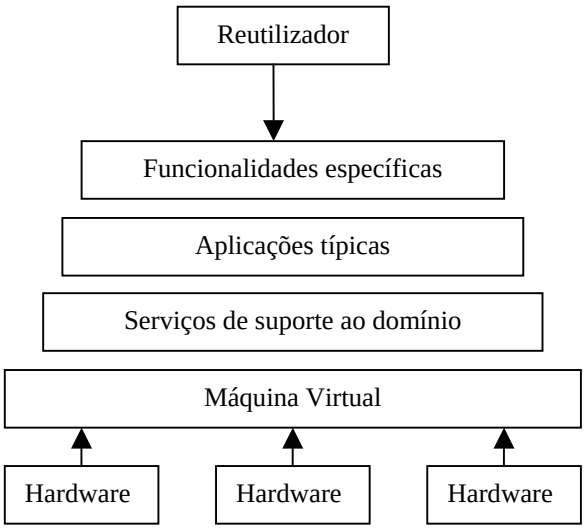

Figura 2.1: Estrutura da Linha de Produção SS2000

2.3.4 Abordagens mistas e outras técnicas

Na maior parte das vezes, os processos, técnicas e métodos dedicados à ED, em geral - e à EA, em particular – utilizam abordagens mistas para a solução dos problemas envolvidos no processo de desenvolvimento. Este tipo de enfoque visa utilizar o que há de melhor nas abordagens mais conhecidas, utilizando por exemplo arquiteturas para a

fase de Projeto da aplicação, onde se mostram mais adequadas, e conceitos de componentização para os conceitos relativos as fases de nível mais alto, como a análise.

Um exemplo típico é o EDLC (GOMAA, 1999). Ele utiliza fortemente arquiteturas para representar o projeto do domínio. Porém, na fase de análise e especificação dos requisitos do domínio, utiliza os diagramas de componentes como sugestão (Agregação, Especificação, Diagrama de Objetos, Diagrama de Transição de Estados e *Features*) para a modelagem do domínio, sofrendo influência de técnicas de DBC.

Parte do seu processo de EA é baseada em técnicas de prototipação. A infra-estrutura que suporta esta atividade consiste de uma série de ferramentas que visam utilizar componentes de prateleira, em conjunto com componentes desenvolvidos sob medida para o processo de desenvolvimento da aplicação.

Todas as visões do domínio são armazenadas num repositório genérico e estão disponíveis para uso. As *features*, por exemplo, representam os componentes de alta granularidade. Neste modelo, existem dependências entre as *features* e entre elas e os objetos do modelo. O relacionamento entre *features* denota, por exemplo, a dependência de dois componentes, mútua inclusão ou exclusão. O relacionamento com os objetos define quais são necessários ao uso de uma *feature* em particular.

Uma vez que o processo de ED tenha sido realizado, a geração de aplicações baseadas em seu modelo pode ser executada. O suporte a esta atividade é dado por um sistema de aquisição de requisitos, denominado KBRET (*Knowledge-Based Requirements Elicitation Tool*), que pretende automatizar o processo de geração de especificações para aplicações do domínio. Através de perguntas ao Engenheiro da Aplicação, esta ferramenta gera um modelo da aplicação, que permite sua prototipação no ambiente de desenvolvimento.

Mais uma vez, o conhecimento dependente e independente do domínio ficam localizados em modelos separados, o que permite que o desenvolvedor se encarregue da escolha dos componentes que julgar necessário, enquanto a inteligência contida no KBRET se encarrega de tarefas como: Verificação de dependência, Eliminação de características desnecessárias e Geração do protótipo do software. A geração é feita com base nas escolhas feitas pelo desenvolvedor e na arquitetura armazenada no ambiente. O ambiente se encarrega de verificar as variações e opções escolhidas e,

através dos relacionamentos entre as *features* e os objetos, deriva o protótipo da aplicação e as visões do modelo relativas a ela.

Métodos formais de geração de software também têm sido usados em algumas propostas recentes. Um exemplo é a proposta de KANG *et al.* (1998), que utiliza *features* para representar o modelo genérico do domínio e a partir deste ponto faz uso de um método baseado em regras para a geração das aplicações baseadas nas características escolhidas.

Outras técnicas, como "*scripting*" e linguagens de *workflow*, também têm sido objetos de pesquisa na área de EA, para solução de problemas específicos. Segundo alguns autores (GRISS, 1999c), estas tecnologias, em conjunto com o uso de componentes, podem mudar sensivelmente a maneira como é hoje feita a reutilização, apesar das mesmas não serem, de forma alguma, revolucionárias. A possibilidade marcação das colaborações entre componentes, por exemplo, com a obtenção de uma flexibilidade semântica maior do que a de um simples diagrama estático, garante lugar às linguagens de *workflow* como uma técnica de futuro para o ramo de reutilização de componentes de domínio (GRISS, 1999c).

2.4 Problemas encontrados nas abordagens de ED/EA atuais

A reutilização estratégica de software, e a EA em particular, promete sempre ganhos no tempo de produção, na qualidade e no custo do desenvolvimento de software. Porém, tecnologias promissoras como Ferramentas CASE, OO, RAD ou Java, não resolveram sozinhas os problemas relativos a esta abordagem de desenvolvimento. Mesmo que a OO, a tecnologia de componentes e o uso de arquiteturas tragam contribuições significativas, um processo de reuso de sucesso deve considerar muitos outros aspectos como os organizacionais e culturais, além dos tecnológicos (GRISS, 1999b).

Domínios, arquiteturas, padrões[2], processos de desenvolvimento e aspectos organizacionais estão fortemente relacionados neste contexto, e devem ser tratados por uma política comum, seguindo um planejamento incremental de transição. Um esforço

[2] Padrões são descrições detalhadas de um determinado componente de software ou problema (GAMMA *et al.*, 1994).

arquitetado para a implantação do reuso, baseado em ED e EA, em larga escala deve ser (GRISS, 1999b):

- Orientado ao negócio – A organização tem uma necessidade articulada e forçada de ganhos dramáticos em matéria de tempo, custo, produtividade e agilidade. Elas produzem aplicações de software e sistemas específicos que muitas vezes formam claramente um domínio, linha de produção e/ou família de aplicações.
- Arquitetado – Aplicações, subsistemas e componentes são projetados para se encaixarem e para preencher as necessidades de uma família ou domínio. Uma estrutura em níveis e modular pode facilitar o processo de reutilização.
- Orientado a processo – Processos distintos para desenvolvimento de software orientado a reutilização e manutenção de arquiteturas, componentes e aplicações são necessários. Eles expressam as opções do domínio, projetam e empacotam os componentes para reutilização.
- Organizado – Compromisso de longo prazo das lideranças e gerências mostra que a organização está estruturada, treinada e compromissada para seguir os processos e padrões. Vários grupos distintos podem criar, reutilizar e manter os componentes reutilizáveis.

Porém, os problemas relacionados à implantação de um processo real de EA não são somente organizacionais. Um exemplo é dado pelas falhas existentes em algumas técnicas, no que diz respeito ao reuso em altos níveis de abstração. Estas falhas vão desde limitações na capacidade de representação de alguns métodos - não suportando representações em grafos complexos e a descrição de casos de uso, por exemplo - até o excesso de possibilidades de outros, o que acaba possibilitando a geração de inconsistências (MANNION *et al.*, 1999).

Além disso, é essencial o suporte de ferramentas computacionais em algumas tarefas, como as de seleção de requisitos e exploração do modelo. Mais uma vez, nestes casos, nem sempre as abordagens fornecem tecnologias adequadas ao reutilizador. A sistematização da tarefa de adição e eliminação de requisitos, por exemplo, deve mostrar suas conseqüências sobre a construção da aplicação, o que nem sempre ocorre.

Outro grande problema encontrado na EA e na ED é a falta de uma padronização mínima entre as propostas de autores distintos. Apesar da existência de

tecnologias, métodos e padrões promissores nesta área, não existe um consenso na comunidade de reuso de quais destas técnicas funcionam melhor, ou nem mesmo de que tarefas devem ser executadas num processo padrão de ED e EA (GRISS, 1999b).

Na literatura já se apresenta a necessidade de um consenso sobre um pequeno padrão de processo de ED, que claramente articule e diferencie o que são as áreas básicas e avançadas para pesquisa. Isto provavelmente ajudaria na pesquisa de técnicas realmente úteis que servissem para agregar um maior valor ao processo de construção de domínios e de aplicações (GRISS, 1999c).

2.5 Exemplos de abordagens existentes de EA

Alguns processos e abordagens de EA exemplificam bem os conceitos e problemas citados neste capítulo. Estes trabalhos expõem também características destas abordagens, mostrando algumas de suas vantagens e desvantagens. Nesta seção, detalhamos algumas propostas, que foram selecionadas devido a sua representatividade com relação aos conceitos e a classificação mostrados nas seções 2.2 e 2.3.

2.5.1 OOSE/RSEB

A combinação das experiência de Jacobson e Griss, nas áreas de OO e reutilização respectivamente, resultou em uma abordagem para reuso denominada RSEB (*Reuse-driven Software Engineering Business*) (JACOBSON *et al.*, 1997). Para suportar a reutilização de software, é utilizado um processo chamado OOSE (Object Oriented Software Engineering), que integra as técnicas de OO e reuso necessárias, diversas delas na área de ED e EA.

O OOSE sugere um processo de desenvolvimento incremental/iterativo orientado a modelos de reutilização. No processo também é utilizada a técnica de *OO Business Engineering* para modelar processos do domínio, objetos do domínio, processos de reutilização e projetos empresariais. Os modelos que representam o domínio definem precisamente os detalhes do processo, como os componentes interagem e o papel dos vários atores presentes (GRISS, 1999a).

Cada aplicação ou componente é expressa como um conjunto conectado de modelos OO, usando OOSE e a notação UML. Componentes são elementos públicos de vários modelos, incluindo requisitos, análise, interfaces, classes de projeto e

implementação (código). Componentes não são utilizados sozinhos na construção da aplicação, mas em grupos de alta granularidade, chamados de sistemas de componentes. Estes sistemas são construídos através de um processo de Engenharia de sistemas de componentes e as aplicações são construídas através do processo de Engenharia de sistemas de aplicações.

De acordo com o OOSE, formar as aplicações e garantir altos níveis de reutilização é tarefa da arquitetura das aplicações e componentes reutilizáveis. Estas aplicações e componentes interagem entre si através de uma arquitetura modular e em níveis. Padrões e *frameworks* garantem consistência das estruturas e dos mecanismos de reuso.

Apesar de ser um dos métodos com maior número de relatos de utilização da literatura, o RSEB possui alguns problemas relativos ao detalhamento das atividades (principalmente nas ligadas à EA, como a seleção de componentes). A representatividade de seus modelos originais tem sido objeto de questionamento, tendo alguns de seus autores inclusive utilizado em trabalhos recentes, extensões para permitir uma maior representatividade deste modelo. Um exemplo é a proposta FeatuRSEB (GRISS *et al.*, 1998)

2.5.2 EDLC/KBRET

O *Evolutionary Domain Life Cycle* (EDLC) (GOMAA, 1999) é um modelo de ciclo de vida que tenta eliminar a distinção entre desenvolvimento e manutenção de software. Ao contrário disso, os sistemas evoluem através de diversas iterações. Assim, os sistemas desenvolvidos usando esta abordagem devem ser capazes de se adaptar a mudanças de requisitos em cada iteração. O EDLC é um ciclo de vida orientado a reuso que foi proposto com o objetivo de prover uma perspectiva de ED (GOMAA *et al.*, 1992), permitindo assim o desenvolvimento e a reutilização de famílias de sistemas.

O domínio de aplicação é modelado de acordo com as seguintes visões:

- Hierarquia de agregação: É usada para decompor complexas agregações entre tipos de objetos em objetos menos complexos, eventualmente ligando as características mais abstratas até níveis mais baixos da hierarquia.
- Diagramas de comunicação entre objetos: Objetos do domínio são modelados como processos concorrentes, que se comunicam entre si usando mensagens. O diagrama de comunicação entre objetos, que também é

estruturado hierarquicamente, mostra como os objetos se comunicam entre si.

- Diagramas de transição de estados: Como cada objeto ativo é modelado como um processo seqüencial, é preciso que se defina de que forma é feita a transição entre seus estados. Este diagrama seqüencial executa esta tarefa.
- Hierarquias de Generalização/Especificação: Como os requisitos de um determinado objeto são trocados para satisfazer as necessidades de uma determinada aplicação, este pode ser especializado através da adição, modificação ou eliminação de operações. As variações de um determinado objeto do domínio são armazenadas aqui.
- *Features* / Dependência entre objetos: Esta visão mostra para cada *feature* (que representam os requisitos do domínio) os objetos necessários para suportá-la.

O desenvolvimento é feito com o auxílio de técnicas de prototipação. O ambiente de suporte à EA consiste de um conjunto integrado de ferramentas que suportam a geração de sistemas específicos através da escolha de requisitos do domínio e sua posterior prototipação. O ambiente utiliza componentes de prateleira para determinadas tarefas e usa a ferramenta CASE *Software through Pictures* (StP), que ajuda a representar as múltiplas visões do domínio. Em cada processo de construção de uma aplicação, a informação das múltiplas visões é extraída, tem sua consistência verificada e é mapeada no repositório de objetos.

Este processo ainda é ajudado pelo suporte de uma ferramenta de Aquisição de Requisitos Baseada no Conhecimento do Domínio (KBRET, *Knowledge-Based Requirement Elicitacion Tool*, no original). Esta funcionalidade foi imaginada para automatizar a geração da especificação da aplicação à partir do modelo do domínio. Esta especificação é derivada do modelo através da escolha de um subconjunto de requisitos específicos da aplicação.

O processo de geração consiste em juntar os requisitos nos termos das *features* do domínio, recuperando desta visão os componentes correspondentes a estas *features* e as relações de dependência entre as *features* para verificação da consistência das escolhas. O KBRET facilita este processo de escolha e geração do protótipo da

aplicação através de funções como navegação no modelo, selecionador de *features*, verificador de dependências e o próprio gerador do sistema.

O selecionador e o verificador de consistência atuam de acordo com os relacionamentos do modelo de *features*, estabelecidos anteriormente no processo de ED. Estes relacionamentos podem denotar incompatibilidade entre *features* ou a necessidade da presença de algumas delas.

O alto número de diagramas e modelos utilizados por esta metodologia tem sido relatado como uma das dificuldades em sua utilização. Outro problema têm sido a falta de suporte para atividades ligadas a especificação da aplicação, nem sempre totalmente fornecida pelo modelo de domínio.

2.5.3 MRAM/TRAM

O MRAM (*Method for Requirements Authoring and Management*) foi proposto tendo em vista o problema da escassa explicação dada à atividade de especificação de requisitos de uma família de aplicações e seu posterior aproveitamento, através da atividade de seleção, no processo de EA. O método é uma proposta de sistematização deste processo, não só através da metodologia proposta, como através do suporte automático ao processo, feito pela ferramenta TRAM (*Tool for Requirements Authoring and Management*) (MANNION *et al.*, 1999).

Definir ligações entre os requisitos é importante. Esta tarefa é a base da primeira parte da metodologia, a Engenharia de Requisitos. Mudanças em um requisito reutilizável podem ter impacto direto em outros deles. Por sua vez, a disciplina em reutilizar os requisitos ajuda os analistas de domínio a consolidar seu entendimento do domínio, o que pode levar a uma melhora numa próxima especificação dos requisitos do domínio. Por isso, a interface entre a fase de EA e a ED é importante nas duas vias.

No MRAM, os requisitos do domínio têm vários atributos. São eles: identificador único, nível de estabilidade, verificabilidade, informações adicionais (que suportam descrições textuais), custo, complexidade, conhecimento pela equipe de desenvolvimento e tecnologia utilizada em sua construção. Os relacionamentos entre os requisitos são do tipo pai-filho, e são gerenciados por uma associação. Assim os requisitos podem ser modelados como uma treliça (MANNION *et al.*, 1999).

Na EA, permitir ao desenvolvedor navegar entre os requisitos do domínio e do sistema, vendo os impactos das mudanças nos pontos de variação, ajuda a focar no que

é essencial ao sistema. A maneira mais comum de seleção é chamada Seleção Livre (conforme discutido na seção 2.2), onde os requisitos que serão utilizados na aplicação são escolhidos aleatoriamente. Porém, no MRAM, a tarefa de construir o modelo da aplicação consiste na escolha dos requisitos numa ordem pré-estabelecida, respeitando sempre relacionamentos como mútua exclusão, mútua inclusão, opcionalidade e obrigatoriedade. A ferramenta que auxilia o processo de escolha (TRAM) ordena as hierarquias que contêm os discriminantes. Neste caso, discriminantes são requisitos que diferenciam uma funcionalidade do domínio de outra.

Apesar de representar um grande auxílio na definição de um processo de modelagem e escolha de requisitos de domínio reutilizáveis, o resultado final do MRAM é uma forma textual de documento, ligada apenas superficialmente ao dicionário do domínio. Assim, é necessário um trabalho manual de adequação dos requisitos escolhidos à arquitetura que permitirá sua implementação.

2.6 Conclusão

A EA é um dos pilares dos processos de reuso baseados em modelagem de domínios. Este enfoque já apresenta algumas pesquisas em ramos anteriormente pouco explorados, como a definição de arquiteturas e a reutilização de componentes de alto nível. Porém, muito trabalho ainda deve ser feito para que outros pontos, como a relação com o processo de ED, a integração entre os diferentes itens dos modelos e a padronização das técnicas utilizadas neste processo, possam ter soluções satisfatórias.

Neste capítulo, fizemos uma revisão da literatura atual sobre técnicas de EA, analisando algumas das abordagens mais representativas na atualidade. Também discutimos alguns requisitos e conceitos desta abordagem, relacionamos problemas encontrados e mostramos exemplos de abordagens e processos que implementam este enfoque.

Capítulo 3

Atividades de um processo genérico de Engenharia de Aplicações

3.1 Introdução

A EA se dedica ao estudo das melhores técnicas, processos e métodos para a produção de aplicações, no contexto do desenvolvimento baseado em reutilização (GRISS *et al.*, 1998). Assim sendo, a EA atua de forma paralela à ED, sendo a última responsável pela construção dos componentes reutilizáveis que representam o domínio (processo conhecido como desenvolvimento *para* reuso). Na EA, os componentes construídos na ED são utilizados para o desenvolvimento de um produto de software (desenvolvimento *com* reuso).

Este capítulo descreve as atividades de um processo genérico de EA, mostrando também algumas adaptações necessárias na modelagem do domínio para facilitar o seu uso. Estas atividades devem ser apoiadas por uma infra-estrutura de reutilização baseada em modelos de domínio. Neste trabalho, consideramos a infra-estrutura desenvolvida no contexto do Projeto Odyssey. O objetivo é obter uma taxa de reuso mais significativa através da sistematização das atividades necessárias para a construção de uma aplicação neste contexto.

Processos de ED e EA possuem uma série de necessidades específicas, devendo levar em conta, dentre outros aspectos, o propósito dos projetos, a tecnologia adotada, a organização da empresa, o grupo de desenvolvimento e o nível de conhecimento dos futuros usuários dos produtos gerados pelo mesmo (BRAGA e WERNER, 1999). Existem na literatura algumas propostas que abordam alguns destes aspectos (JACOBSON *et al.*, 1997, GRISS *et al.*, 1998), mas muitas vezes elas possuem um foco somente no processo de modelagem do domínio, sem mencionar uma descrição detalhada de sua utilização para o desenvolvimento de aplicações e dos aspectos da integração entre estes dois processos.

Um processo de modelagem e utilização de um domínio pode ser compilado a partir de conceitos consagrados na área de reutilização, encontrados nos métodos de ED mais conhecidos da literatura, como FODA (KANG *et al.*, 1990) e ODM (SIMOS,

1996), e suas variações como FODACom (VICI, 1998), FORM (KANG *et al.*, 1998) e OODE (CHAN e LAMMERS, 1998). Além disso, ao adotarmos o paradigma OO como base deste trabalho, há ainda a necessidade de agregar conceitos que adaptem suas características àquelas necessárias ao desenvolvimento baseado em modelos de domínios. Sendo assim, é preciso agregar conceitos como os apresentados em abordagens como o RBSE/OOSE (JACOBSON *et al.*, 1997) e o método Catalysis (D'SOUZA e WILLS, 1998), tais como *use-cases*, *variation points*, refinamentos sucessivos, *packages*, entre outros. Outra característica importante é a adoção de uma notação padronizada. Neste caso, a escolhida é a UML – *Unified Modeling Language* (BOOCH *et al.*, 1998).

O uso de alguns recursos característicos do desenvolvimento visual também pode auxiliar a modelagem do domínio e seu entendimento no decorrer do processo de EA. Neste trabalho, estes recursos são utilizados para promover uma maior semântica na representação dos componentes de domínio, nas diversas fases do desenvolvimento, facilitando assim sua utilização pelo desenvolvedor.

Este capítulo está dividido em cinco seções (incluindo esta introdução). Na seção 3.2, apresentamos uma visão geral sobre o processo de Engenharia de Domínio, bem como o processo genérico de Engenharia de Aplicações correspondente, ambos apoiados pela infra-estrutura do Odyssey. Na seção 3.3, é apresentada a proposta do Modelo de *Features* Estendido, utilizado como uma representação padronizada dos componentes reutilizáveis do domínio. Na seção 3.4, são detalhadas as atividades do processo de EA proposto, definindo as tarefas executadas e os resultados produzidos em cada uma delas. Na seção 3.5, são mostrados os detalhes de suporte necessários ao processo de EA da infra-estrutura Odyssey. Finalmente, na seção 3.6, é feita uma conclusão sobre o capítulo.

3.2 Processos adotados no Odyssey

Nesta seção, descrevemos brevemente o processo de ED utilizado no Projeto Odyssey, denominado Odyssey-ED, conforme apresentado em (BRAGA e WERNER,

Capítulo 4

Uma infra-estrutura de suporte à Engenharia de Aplicações no contexto do Projeto Odyssey

4.1 Introdução

Este capítulo apresenta os principais aspectos de uma infra-estrutura de suporte que auxilia o desenvolvimento de software realizado de acordo com o processo de Engenharia de Aplicações proposto no capítulo anterior: o Odyssey-EA. É também detalhado o suporte às principais atividades do processo, e relatadas as modificações necessárias na infra-estrutura de suporte à Engenharia de Domínio, no que diz respeito ao apoio dado ao Modelo de *Features* Estendido.

A infra-estrutura de suporte à EA foi implementada no contexto do Projeto Odyssey (BRAGA e WERNER, 1999), em desenvolvimento na COPPE/UFRJ. Este projeto tem como objetivo apoiar o desenvolvimento de software baseado em modelos de domínio. A infra-estrutura do Odyssey fornecerá, assim, subsídios para que a reutilização de software possa ser maximizada através de processos da ED e EA, definidos especialmente para sua realidade, utilizando conceitos consagrados pela comunidade de reutilização – como *features* (KANG *et al.*, 1990), *use-cases* (JACOBSON *et al.*, 1997) e o padrão UML (BOOCH *et al.*, 1998).

Conforme dito no capítulo anterior, o processo de EA do Odyssey é fortemente influenciado pela existência de uma infra-estrutura de suporte. As ferramentas desta infra-estrutura são responsáveis por funções tais como: automatizar tarefas repetitivas, manter a estruturação do modelo e gerenciar a base de dados do desenvolvimento. Sem a ajuda de ferramentas específicas, a execução do processo implicaria numa grande quantidade de trabalho manual, seja na manutenção da ligação entre os diversos componentes do domínio, seja na verificação do resultado final das atividades do processo de EA. Por outro lado, obviamente são os processos que especificam as funcionalidades das ferramentas do Odyssey, na medida em que representam a principal razão de sua existência (JACOBSON *et al.*, 1999).

Este capítulo está dividido em cinco seções (incluindo esta introdução). Na seção 4.2, é apresentado o contexto no qual o trabalho foi desenvolvido (i.e. o Projeto

Odyssey). Na seção 4.3, é mostrado um panorama geral da infra-estrutura Odyssey em seu estágio atual, detalhando algumas ferramentas já disponíveis e identificando outras ainda em desenvolvimento. Na seção 4.4, é descrita a infra-estrutura construída para suporte ao processo de EA do Odyssey, bem como seu modo de utilização e as mudanças necessárias na parte relativa a ED. Finalmente, na seção 4.5, é feita uma conclusão sobre a implementação e a utilização da infra-estrutura.

4.2 O contexto do Projeto Odyssey

Sendo o Odyssey uma infra-estrutura de reutilização de software baseada em modelos de domínio (BRAGA e WERNER, 1999), todas as características do processo de desenvolvimento (ciclo de vida, fases, atividades, gerência do projeto e controle de qualidade) são definidas baseadas na premissa de enfatizar a reutilização de software na construção de aplicações.

A proposta das técnicas utilizadas no Odyssey é a de adotar, sempre que possível, alguns padrões de desenvolvimento maduros na comunidade de Engenharia de Software como: desenvolvimento OO, padrão UML, *use-cases* e *features* (para representação de conceitos do domínio) e a utilização de uma linguagem de programação multiplataforma e largamente conhecida na implementação, no caso Java (SUN, 2000). A linguagem Java foi adotada visando facilitar a portabilidade da infra-estrutura, o seu uso fora do contexto da equipe de desenvolvimento e o aprendizado de sua estrutura por novos membros do projeto.

A Figura 4.1 mostra a arquitetura do modelo com as classes genéricas, que servem de base para os componentes que implementam a maior parte dos modelos da infra-estrutura de suporte do Odyssey. O modelo apresentado na Figura 4.1 foi construído no próprio diagramador da infra-estrutura (detalhes na seção 4.3.1).

Neste modelo genérico, a classe *ModeloAbstrato* representa uma abstração de alto nível de um modelo ou elemento do domínio, definindo um protocolo padrão de atributos e serviços que deve ser seguido por todos os modelos e itens que compõem os elementos da infra-estrutura. São especializações desta classe os seguintes conceitos: *ItemModelo, ConjuntoModelos* e *Modelo*.

A classe *ItemModelo* representa os elementos dos diferentes modelos, podendo ser visto como folha da árvore de modelos de uma aplicação/domínio. Entretanto, por ser um modelo abstrato, o item também pode ter outros itens de modelo como

subclasses. Uma aplicação ou domínio, no Odyssey, é representada por um conjunto de modelos, sendo *ConjuntoModelos* é a classe que representa esta estrutura. A classe *Modelo* representa o segundo nível da árvore de Modelos de uma aplicação/domínio, sendo o primeiro nível composto pela aplicação/domínio (*ConjuntoModelos*) e os demais níveis compostos por itens de modelo (*ItemModelo*). O modelo funciona, assim, como um conjunto de itens de modelos.

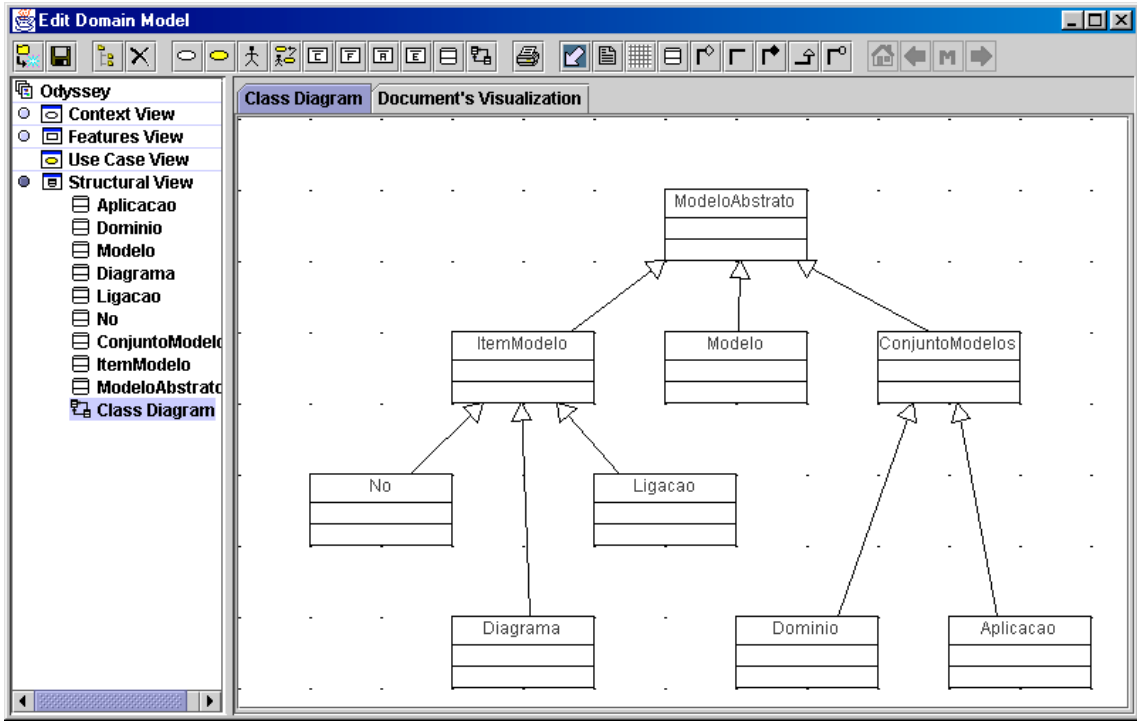

Figura 4.1: Modelo de classes genérico do Odyssey

A classe *Dominio* representa um domínio de aplicações. Um domínio pode estar relacionado com diversas aplicações. Cada aplicação desenvolvida dentro da infra-estrutura se encontra no contexto de um determinado Domínio. A classe *Aplicacao* representa, por sua vez, as aplicações do domínio. As classes *Diagrama*, *No* e *Ligacao* são as especializações dos itens de modelo que compõem um modelo. Assim, todos os modelos que representam o domínio na infra-estrutura especializam a classe *Modelo*, os diagramas, a classe *Diagrama*, os nós, a classe *No* e as ligações entre os nós, a classe *Ligacao*.

As classes do Odyssey estão divididas em pacotes Java. As principais classes, detalhadas na figura acima, estão no pacote *Modelo*. Cada pacote é dividido ainda em sub-pacotes, que especializam ainda mais suas funções. As classes *Diagrama*, *No* e

Ligacao estão em uma subdivisão do pacote modelo, chamado *Modelo.Generico*. O pacote *Modelo* possui também outro sub-pacote chamado *Modelo.Features*, onde ficam os componentes do modelo de *features*. Este pacote, que é a base para a implementação do Modelo de *Features* Estendido (detalhado na seção 3.3), é apresentado na Figura 4.2.

Além do pacote *Modelo*, existem diversos outros na infra-estrutura, como *Ferramentas*, *Lexico* e *PaineisRepresentacao*, por exemplo. Estes pacotes contém as classes que compõem as ferramentas não totalmente integradas à infra-estrutura, as estruturas léxicas de representação e os painéis de interface com o usuário, respectivamente.

4.3 A infra-estrutura de suporte Odyssey

A infra-estrutura de suporte Odyssey tem como objetivo apoiar a execução efetiva de processos de Engenharia de Domínio (seção 3.2.1) e Engenharia de Aplicações (3.4). Para isso, ela disponibiliza uma série de ferramentas genéricas, que dão apoio e/ou automatizam atividades específicas destes processos, sejam elas de desenvolvimento, planejamento, acompanhamento gerencial ou controle de qualidade. A implementação atual da infra-estrutura possui uma série de ferramentas de apoio aos processos de ED e EA, que são detalhadas nesta seção.

Os componentes do Odyssey foram implementados em Java, seguindo uma padronização, tanto estrutural como de documentação, descrita em (BARROS *et al.*, 1999b). A atual versão da infra-estrutura armazena seus dados de duas formas distintas: Através da serialização, nativa do Java, e do gerente de objetos GOA++, desenvolvido pela linha de Banco de Dados da COPPE/UFRJ (MAURO *et al.*, 1997). O relacionamento Java/GOA++ é executado através de um pacote específico do Odyssey (*Goa*) que gerencia todos os aspectos desta ligação, como mapeamento de características, tradução de formatos e tratamento de erros.

Existem hoje diversas ferramentas específicas no Odyssey, atuando de forma integrada com a infra-estrutura de armazenamento de componentes reutilizáveis. Podemos citar como exemplos a ferramenta de documentação de componentes (MURTA, 1999), os diagramadores genéricos, o gerador de código executável (SOUZA e WERNER, 2000), a ferramenta de aquisição de conhecimento (ROSETI, 1999) e o navegador inteligente (BRAGA *et al.*, 1999).

A seguir, analisamos detalhadamente algumas ferramentas, utilizadas diretamente em atividades do Odyssey-EA.

4.3.1 Diagramadores genéricos

Os diagramadores genéricos do Odyssey foram construídos tendo como principal objetivo o suporte à criação dos modelos definidos pelo processo de ED (BRAGA e WERNER, 1999). A notação utilizada é, basicamente, a descrita no padrão UML (BOOCH *et al.*, 1998), com algumas construções adicionais para suportar a representação de componentes reutilizáveis (BRAGA e WERNER, 1999) (ver seção 3.3 e 3.5). Deste modo, o Odyssey possui uma série de diagramadores genéricos disponíveis, para permitir a confecção dos diferentes diagramas utilizados no modelo de domínio: Diagramas de Classes, *Use-cases*, *Features*, Contexto, Diagramas de Seqüência, e de Transição de Estados.

Os diversos modelos, representados por estes diagramas, são implementados através de extensões da estrutura básica mostrada na Figura 4.1. Assim, as classes que compõem o modelo de *use-cases*, por exemplo, são especializações das classes principais (*Modelo*, *Diagrama*, *No* e *Ligacao*) e estão organizadas num pacote chamado *Modelo.UseCase*.

Todas as ferramentas projetadas e construídas no contexto do Odyssey utilizam e implementam o conceito de rastreabilidade (BRAGA e WERNER, 1999). Ou seja, os elementos dos diversos diagramas são conectados semanticamente, e estas ligações são utilizadas sempre que necessário. Um exemplo deste tipo de conexão pode ser encontrado entre os diagramas de seqüência de *use-cases* e os diagramas de classes.

O Odyssey-EA utiliza fortemente esta visão de diversos diagramas integrados, seja para atividades típicas da modelagem da aplicação (seção 3.4.2 a 3.4.4), seja para a exploração do modelo de domínio pelo Engenheiro da Aplicação. A rastreabilidade do modelo de domínio, por exemplo, é fundamental para o processo de instanciação dos componentes do domínio para o contexto da aplicação.

4.3.2 Documentação de componentes de domínio

A documentação dos componentes de domínio do Odyssey é feita com o auxílio de uma ferramenta chamada FrameDoc (MURTA, 1999). O FrameDoc permite a criação de *templates* de documentação específicos para cada tipo de componente (classe, *use-cases*, diagrama de estados, etc.). Os tipos de campos fornecidos para a

construção do *template* são: texto, *memo*, *checkbox*, *radiobutton*, som, imagem, vídeo e HTML.

Algumas das características do FrameDoc são:

* Exportação da documentação utilizando o padrão HTML;
* Utilização de hiperligações no campo HTML para outros documentos ou páginas na Internet;
* Criação de padrões de documentação, representados pelos *templates*;
* Geração de documentação modular;
* Utilização de multimídia na documentação;
* Visualização da documentação a ser gerada;
* Geração da documentação implícita ao componente (ex. métodos e atributos de classes).

Esta ferramenta já se encontra implementada e integrada à infra-estrutura. Os componentes das aplicações desenvolvidas no contexto do Odyssey, mantém a estrutura de documentação dos componentes originais do domínio. Todavia, seus campos são preenchidos de acordo com as características específicas da realidade da aplicação.

4.3.3 Geração de código

A ferramenta de geração de código fonte do Odyssey foi definida e implementada por SOUZA e WERNER (2000), de acordo com as características necessárias à implementação de componentes de domínio do Odyssey, detalhadas em (BRAGA e WERNER, 1999). Ela permite a geração do código fonte das classes, na fase de implementação dos componentes do domínio, baseada em suas definições de atributos e serviços. Até mesmo a geração do corpo dos serviços descritos no modelo estrutural, pode ser executada através da descrição do seu conteúdo. A definição destes algoritmos internos dos métodos é feita numa meta-linguagem, semelhante ao Pascal.

O Odyssey-EA utiliza esta ferramenta para a geração do código final dos componentes da aplicação (seção 3.4.4). Assim, os componentes do domínio, reutilizados sem alterações, tem seu código totalmente gerado de forma automática. Já os novos componentes, e os que tiveram alterações, terão seu esqueleto de serviços e atributos construídos, sendo responsabilidade do desenvolvedor fazer as alterações necessárias.

As adaptações dos componentes modificados podem ser feitas com o auxílio da ferramenta de geração (que oferece campos apropriados para digitação do corpo dos serviços, por exemplo), ou de forma mais direta, na ferramenta utilizada para compilação do código.

A tarefa de geração é executada tomando como base a escolha da linguagem de desenvolvimento, feita pelo Engenheiro da Aplicação na atividade de Especificação das diretrizes de implementação (seção 3.4.3). Esta geração, executada através de um *parser*, pode ser feita, atualmente, em três linguagens diferentes: Java, Delphi e C++.

4.3.4 Localização dos componentes de domínio

A funcionalidade principal desta ferramenta é a de localizar os componentes do domínio, de acordo não só com a funcionalidade requerida, como a partir de uma série de variáveis, que vão desde os interesses do usuário da infra-estrutura, até seu grau de conhecimento do domínio e da ferramenta de desenvolvimento (BRAGA *et al.*, 1999). A tarefa de localização pode ser realizada dentro da própria base de componentes da infra-estrutura de suporte, ou de outras bases via *internet*.

A versão atual da ferramenta provê um agente de interface (AI), agentes de filtragem (AFs) e agentes de recuperação (ARs) que auxiliam na busca inteligente de informações. O AI é responsável pela adaptação da interface com o usuário, apresentando os componentes adequados para reutilização e seus detalhes, de acordo com o perfil do usuário. Os AFs são responsáveis pelo modelo do usuário e sua evolução de acordo com o comportamento de navegação do usuário, utilizando-o para filtragem das informações mais importantes do domínio a serem mostradas pela interface adaptativa. Os ARs são responsáveis pela recuperação de informações dos repositórios de dados a partir de informações fornecidas pelos AFs, lidando de forma transparente ao usuário as questões como distribuição e heterogeneidade (BRAGA *et al.*, 2000).

Esta ferramenta já se encontra implementada e integrada à infra-estrutura. No Odyssey-EA, ela pode ser utilizada na localização de componentes, com intuito de selecioná-los para uma dada aplicação (seção 3.4.2). Mesmo componentes de nível de abstração mais baixo, como *use-cases* e classes, podem ser localizados e vir a ser adicionados na aplicação para atender as necessidades específicas nos momentos adequados do processo (3.4.2 a 3.4.4).

4.3.5 Definição e instanciação de arquiteturas de software

Esta ferramenta tem como objetivo permitir ao desenvolvedor a definição de arquiteturas específicas de domínio (DSSAs), o detalhamento de quais são os casos de uso mais adequados a cada uma delas (baseado nos requisitos da aplicação) e a instanciação destas arquiteturas para aplicações específicas (XAVIER, 2000).

Uma ferramenta de suporte ao uso de DSSAs tem como principal propósito a construção e reutilização de artefatos relacionados ao projeto de um software. Cada uma destas arquiteturas de referência apresenta duas naturezas que se relacionam intimamente: uma natureza orientada ao desenvolvimento *para o reuso*, de artefatos genéricos relacionados a um domínio de aplicação, e uma natureza orientada ao desenvolvimento *com reuso*, objetivando sua utilização na produção de aplicações reais no contexto deste domínio. Ao buscar a produção de aplicações em determinado domínio, um DSSA cria um elo de ligação entre a ED e a EA, enfocando explicitamente a fase de projeto arquitetural através da estruturação dos componentes identificados durante o estudo do domínio e da disponibilização de um arquitetura que sirva como referência para a construção de aplicações neste domínio.

Assim, o Odyssey-EA utiliza os componentes definidos num DSSA para instanciar a arquitetura, integrando-a ao modelo de projeto da aplicação (seção 3.4.3). A sugestão da arquitetura é feita tendo como base os requisitos necessários à aplicação. Para isto, o processo prevê uma atividade anterior de definição dos requisitos de referência (Diretrizes de Projeto) (seção 3.4.2).

Este trabalho se encontra atualmente em desenvolvimento na COPPE/UFRJ.

4.3.6 Avaliação de padrões e anti-padrões de projeto

Esta ferramenta visa analisar, através de um série de heurísticas nela implementadas, a detecção de padrões estruturais de desenvolvimento (GAMMA *et al.*, 1994). A ferramenta detecta também a presença de anti-padrões, que são estruturas não recomendáveis de modelagem (CORREA *et al.*, 2000). A avaliação feita pela ferramenta identifica ainda circunstâncias onde determinados padrões poderiam ser substituídos por outros, com melhores resultados.

No Odyssey-EA, esta ferramenta pode ser aplicada na verificação de modelos estruturais, em uma série de atividades do desenvolvimento. O melhor retorno pode ser obtido na fase de projeto (seção 3.4.3), onde os modelos estruturais já se encontram

totalmente definidos e, uma vez detectadas estruturas não recomendáveis, é possível que o Engenheiro da Aplicação as corrija com maior facilidade.

Esta ferramenta existe atualmente na forma de um protótipo inicial, desconectado da infra-estrutura Odyssey. Entretanto, já existe uma proposta para torná-la integrada à infra-estrutura, em desenvolvimento neste momento.

4.3.7 Planejamento e Análise de Riscos

Esta ferramenta tem como objetivo suportar os processos de gerenciamento de riscos definidos em (BARROS *et al.*, 1999a). O gerenciamento de riscos na infra-estrutura Odyssey se realiza através de dois processos: o processo de identificação e documentação de riscos e o processo de gerenciamento de riscos para o desenvolvimento de aplicações.

O primeiro processo identifica e documenta os riscos associados a elementos de projeto, tais como o domínio da aplicação, as tecnologias utilizadas em seu desenvolvimento, os artefatos produzidos, os papéis desempenhados pela equipe, entre outros. No contexto do projeto Odyssey, o domínio da aplicação é um elemento de projeto fundamental, onde são focalizados os esforços de identificação de riscos. Assim, o primeiro processo de identificação de riscos é ativado num momento oportuno do processo Odyssey-ED, ou seja, sempre que um novo domínio de aplicação é analisado.

Os riscos associados aos elementos de projeto são descritos na forma de arquétipos de risco (BARROS, 2000b). Um arquétipo de risco é uma estrutura padrão para documentação de problemas recorrentes ao longo de diversas aplicações. Eles agregam informações para identificação do problema recorrente, o contexto de sua ocorrência e modelos dinâmicos de projeto para avaliação de impacto, soluções de contenção e contingência (BARROS *et al.*, 2000a).

O Odyssey-EA possui uma fase relativa às tarefas de Planejamento e Análise de Riscos (seção 3.4.1), que corresponde ao segundo processo de gerenciamento de riscos previsto para o Odyssey. Este processo reutiliza os arquétipos de risco produzidos pelo processo anterior, auxiliando o gerente de projetos na identificação, avaliação de impacto e planejamento dos riscos de sua aplicação.

Este trabalho se encontra atualmente em desenvolvimento na COPPE/UFRJ.

4.3.8 Gerência do processo de desenvolvimento

O intuito desta ferramenta é o de possibilitar ao Engenheiro do Domínio/Aplicação a definição e acompanhamento de diferentes processos de desenvolvimento, baseados em especializações dos processos padrão de ED e EA. Ela tem ainda como objetivo a customização do grau de flexibilidade desejado no acompanhamento e a especificação da forma de uso das ferramentas disponíveis na infra-estrutura em atividades específicas do processo (MURTA, 2000).

O acompanhamento do processo se baseia no uso de agentes inteligentes, que devem ser capazes de verificar o estágio de cada uma das atividades e/ou tarefas de um processo instanciado e verificar se seu respectivo andamento está de acordo com o que foi definido. Esta característica permite uma gerência muito mais efetiva de qualquer processo executado no contexto do Odyssey, sendo que até mesmo a tolerância com atrasos e falhas no encadeamento das atividades pode ser definida.

Assim, o Engenheiro da Aplicação poderá avaliar uma determinada necessidade de projeto específica, verificando se ela cria algum novo requisito ou se dispensa alguma atividade prevista no processo padrão. A ferramenta permitirá ainda o acompanhamento detalhado deste processo instanciado pelo gerente de projeto, garantindo uma maior efetividade no controle do processo de desenvolvimento e o cumprimento das fases e atividades previstas.

Este trabalho se encontra atualmente em desenvolvimento na COPPE/UFRJ.

4.4 A infra-estrutura de suporte ao processo de Engenharia de Aplicações implementada

A infra-estrutura de suporte ao processo de EA faz parte, e está totalmente integrada, a infra-estrutura genérica de suporte à modelagem e utilização de domínios do Odyssey. Assim, ferramentas como os diagramadores genéricos e as estruturas léxicas e semânticas são aproveitados na implementação das ferramentas específicas de suporte às atividades do processo de EA.

Porém, algumas adaptações na abordagem original de ED (BRAGA e WERNER, 1999), e obviamente em sua infra-estrutura de suporte, foram necessárias para viabilizar a implementação de algumas das propostas deste trabalho (principalmente, o que diz respeito ao Modelo de *Features* Estendido) e conseguir uma

maior integração entre os processos de ED e EA. Nas próximas seções, descrevemos as adaptações executadas no modelo original do domínio, o modo de utilização da infra-estrutura de ED e EA e a construção das ferramentas de apoio às atividades do Odyssey-EA .

4.4.1 Extensões para suporte ao Modelo de Features Estendido

Os diagramadores genéricos são implementados baseados no modelo estrutural genérico do Odyssey, mostrado na Figura 4.1. Na Figura 4.2, mostramos o pacote *Modelo.Features*, que contém as especializações necessárias ao diagramador de *features*, com as modificações necessárias para implementar as funcionalidades adicionais propostas para o Odyssey (seção 3.3).

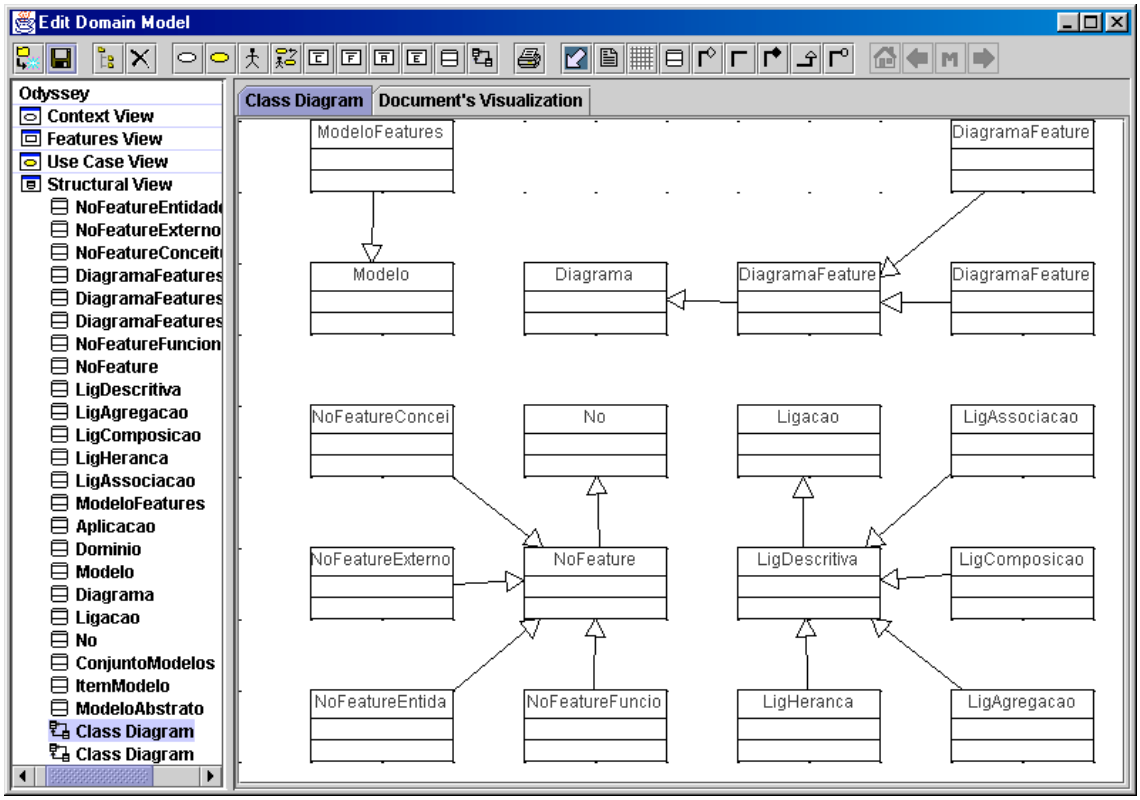

Figura 4.2: Modelo de classes: Pacote Modelo.Features

Como foi dito anteriormente, as classes deste pacote são especializações das classes genéricas mostradas na Figura 4.1. A classe *Modelo.Features* especializa *Modelo*, *DiagramaFeatures* herda de *Diagrama*, *NoFeature* é uma especialização de *No* e as classes de Ligação herdam de *LigDescritiva*. A classe *LigDescritiva* é uma

especialização da classe genérica *Ligacao* e também se encontra no pacote *Modelo.Generico*.

Duas classes são especializações de *DiagramaFeatures*: *DiagramaFeaturesConceitual* e *DiagramaFeaturesFuncional*. Estas classes representam as duas visões do modelo de *features* existentes na fase de análise: Funcional e Conceitual. *NoFeature* também possui algumas especializações: *NoFeatureFuncional*, *NoFeatureConceitual*, *NoFeatureEntidade* e *NoFeatureExterno*. Elas existem devido às diferentes informações existentes no padrão de domínio de cada uma das *features* da taxonomia (seção 3.3). As ligações funcionam da mesma forma, com quatro classes diferentes (*LigAgregacaoFeature*, *LigComposicaoFeature*, *LigAssociacaoFeature* e *LigHerancaFeature*), representando os relacionamentos previstos no modelo.

Outros pacotes da infra-estrutura do Odyssey também foram importantes na implementação deste diagrama. Podemos citar como exemplo: *Lexico.Features*, responsável pelas características de representação das *features* (ícone, tamanho, etc.), *Modelo.Contexto*, que teve que ser modificado para armazenar as *features* relacionadas, e *PaineisRepresentacao*, que contém as janelas utilizadas pela ferramenta.

4.4.2 Suporte ao Processo de Engenharia de Domínio

O processo de Engenharia de Domínio do Odyssey foi definido baseado em uma série de conceitos largamente utilizados na literatura de reutilização (BRAGA e WERNER, 1999). Assim, uma infra-estrutura de suporte a este processo foi implementada tendo como bases estes conceitos (seção 3.2.1).

Porém, como dito na seção anterior, para viabilizar a implementação de pontos específicos da proposta descrita nesta tese, algumas adaptações foram necessárias na infra-estrutura original. A maior parte delas está relacionada a modificação do diagramador genérico de *features* para o suporte às características do modelo estendido (ver seção anterior e 3.3). Nesta seção, mostramos brevemente a infra-estrutura de suporte ao processo de ED.

A Figura 4.3 mostra a janela principal do Odyssey. A partir desta tela, o Engenheiro do Domínio pode criar novos modelos de domínio (e aplicações baseadas neste modelo), modificá-los, utilizar as ferramentas de suporte a funcionalidades específicas do processo e executar tarefas de configuração do domínio e da infra-estrutura (como coleta de lixo, por exemplo).

Na figura, podemos visualizar os botões *Model* e *Create*, que permitem o acesso aos modelos existentes e a criação de novos modelos. Além disso, podemos ver também as opções de menu *Applications*, *Tools* e *Config*, que dão acesso às funcionalidades de controle de aplicações, ferramentas adicionais (não totalmente integradas à infra-estrutura do Odyssey) e características de configuração, respectivamente. Ao pressionar o botão *Model* da figura, o Engenheiro do Domínio abre a janela que permite a visualização e a edição do modelo de domínio *Telefonia*, por exemplo, mostrado na *combo-box* ao seu lado.

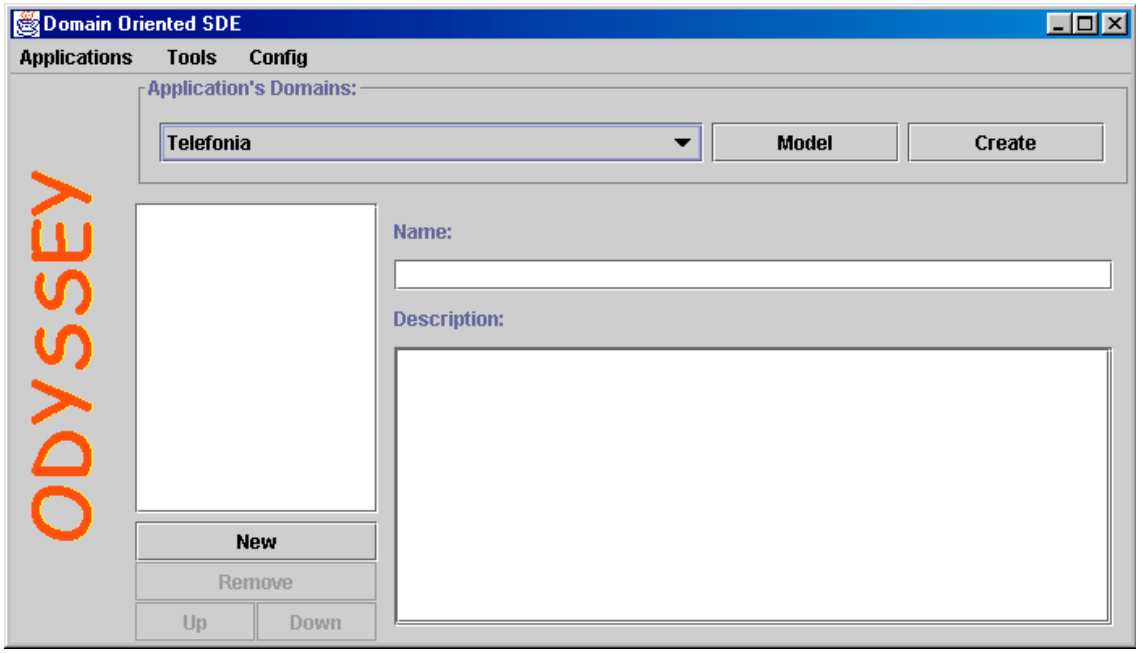

Figura 4.3: Janela Principal do Odyssey

Além dos elementos já citados, visualiza-se na Figura 4.3 duas caixas de Texto: *Name* e *Description*, além de quatro botões: *New*, *Remove*, *Up* e *Down* e uma campo em branco (à esquerda). Todos estes componentes se referem as aplicações do domínio e são sensíveis ao contexto. Assim, o botão *New* permite que se crie uma nova aplicação, que aparecerá automaticamente na janela acima dele. Os outros botões servem para remover aplicações (*Remove*) e trocar a ordem de visualização no campo de aplicações (*Up* e *Down*). As caixas de texto *Name* e *Description* se referem, respectivamente, ao Nome e à Descrição da aplicação selecionada na Janela.

O modelo para o domínio de *Sistemas de Telefonia*, utilizado como exemplo neste capítulo, foi mostrado em detalhe no capítulo anterior (ver seção 3.3.4). Este

modelo de domínio representa os componentes genéricos que refletem uma visão do domínio com os diferentes aspectos da área de telefonia. Ele está dividido em 3 contextos: *Financeiro* (aspectos relativos a parte financeira, como *Forma de Cobrança*), *Operacional* (aspectos relativos a operação do sistema, como *Tipo de Uso* e *Modo de Discagem*) e *Técnico* (aspectos relativos às características técnicas, como *Estação* e *Qualidade da Linha*). O Diagrama de Contexto do domínio pode ser visto na Figura 4.4. Para obtermos uma melhor visualização, o modelo aqui mostrado é uma versão simplificada, omitindo algumas informações, presentes no modelo completo da Figura 3.6.

Nesta figura podemos visualizar ainda a barra de tarefas do Odyssey, que também é sensível ao contexto, ou seja, suas opções são relativas ao diagramador que se encontra em uso. Nesta barra de tarefas, encontram-se as opções de uso do diagrama (como escolha do modo de edição e inserção de elementos diversos) e de funções de suporte da infra-estrutura (como gravar e imprimir).

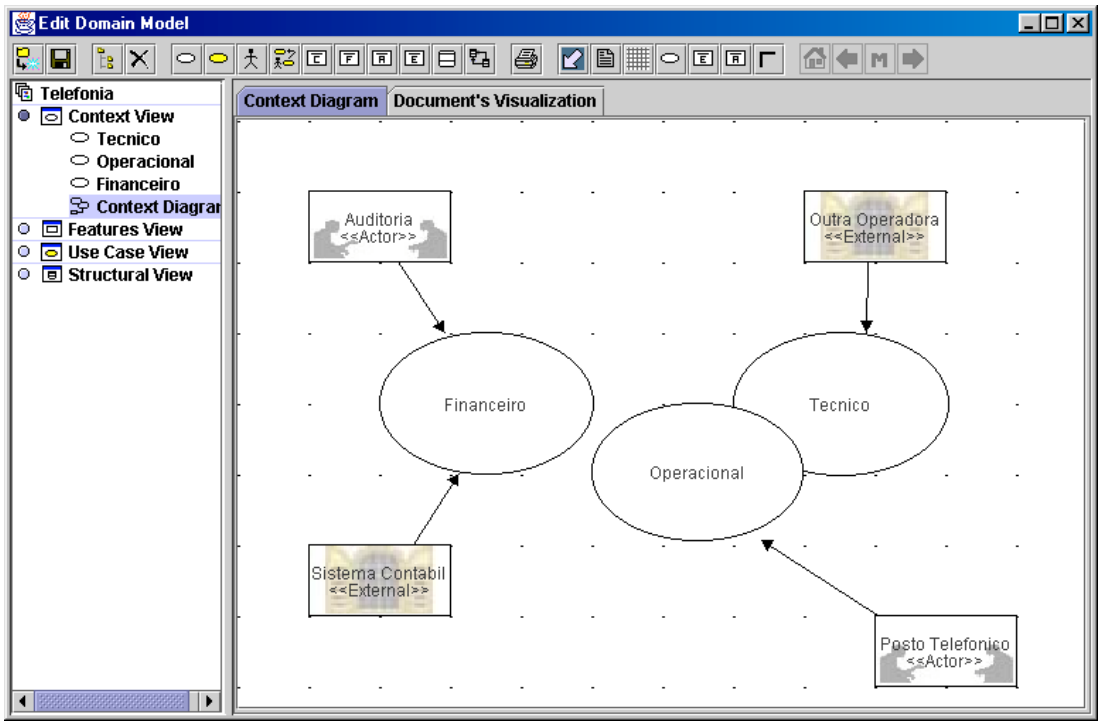

Figura 4.4: Diagrama de Contexto

Na Figura 4.5, vemos a janela com o modelo de domínio. Nele, podemos visualizar as *features* do modelo estendido proposto para o Odyssey. Assim, pode-se notar os diferentes tipos de componentes (Organizacional e Essencial), que são

representados por ícones distintos (seção 3.3). Pode-se ver também um componente opcional (tracejado) e uma restrição entre duas *features* (representado por um X). Notam-se ainda os relacionamentos de composição e herança entre os componentes, dois dos tipos previstos para este diagrama (seção 3.3).

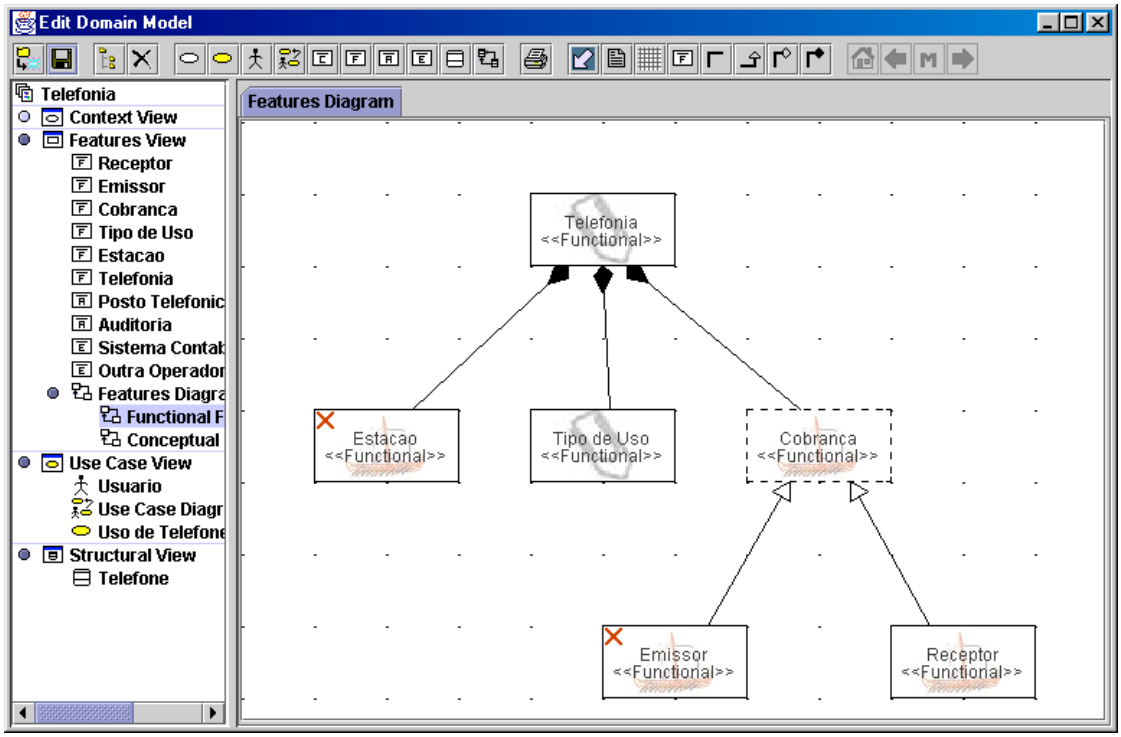

Figura 4.5: Modelo de Features Estendido do Odyssey

Na parte esquerda da figura, encontram-se as diferentes visões do modelo de domínio: Contexto, *Features*, *Use-case* e Estrutural (classes), bem como seus respectivos componentes. No diagrama de *features*, podemos notar a existência de duas visões: *Funcional* e *Conceitual* (seção 3.4.1). O diagramador, do lado direito da tela, é sensível ao contexto, mostrando o diagrama referente à visão que estiver selecionada, ou ainda, o padrão de domínio de um componente, caso um deles seja selecionado (ver Figura 4.6).

Os padrões de domínio das *features* foram implementados para serem utilizados de forma integrada aos componentes que a representam. Na Figura 4.6, podemos visualizar o padrão de domínio de um componente essencial. As informações do componente são mostradas neste padrão: Nome, Descrição, Sinônimos, Tipo, Fonte, Prioridade e Opcionalidade, além das Restrições Exclusivas e Inclusivas e dos componentes relacionados (*use-cases*, conceitos e atores), que dependem do tipo de

feature (Entidade) e da visão envolvida (Funcional ou Conceitual). Existe também uma aba *Document´s Visualization* com a documentação do componente de domínio, gerada pela ferramenta FrameDoc.

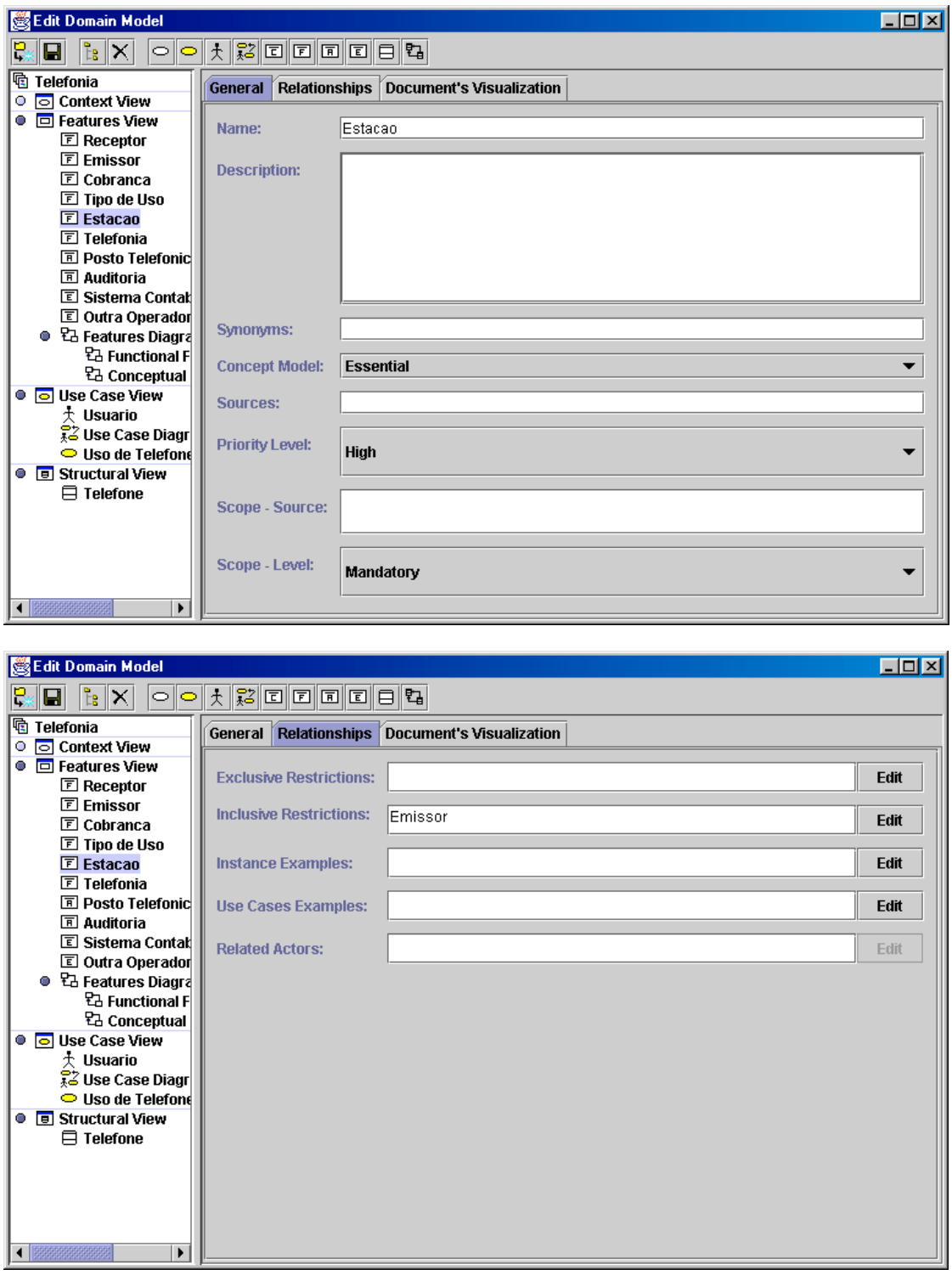

Figura 4.6: Padrão de Domínio de um componente

Para uma maior concisão e entendimento, algumas vezes ocorre a divisão de janelas da infra-estrutura de suporte em diferentes abas, mostradas no canto superior esquerdo da tela. Assim, as informações podem permanecer em uma só janela, mas sem excesso de conteúdo mostrado simultanemente. Um exemplo pode ser visto na janela da Figura 4.6 que possui três abas (*General*, *Relationships* e *Documents´s Visualization*)

É no padrão de domínio que estão explicitados os relacionamentos entre componentes, que implementam o conceito de rastreabilidade. Os componentes de todos os modelos do Odyssey podem ser ligados por este conceito, sendo estas ligações definidas durante o processo de ED e se encontram descritas nos diferentes padrões dos diferentes componentes do modelo.

Na Figura 4.7, é mostrado um padrão de Relacionamento, acessado através de um duplo clique sobre sua representação no diagrama. Neste padrão, podemos verificar as informações básicas do relacionamento (Nome e Descrição), além de informações de apresentação no diagrama (largura da ligação, ângulo, etc.), encontradas na aba *Visualization*. O duplo clique também pode ser utilizado sobre uma *feature* do diagrama para acessar seu padrão de domínio.

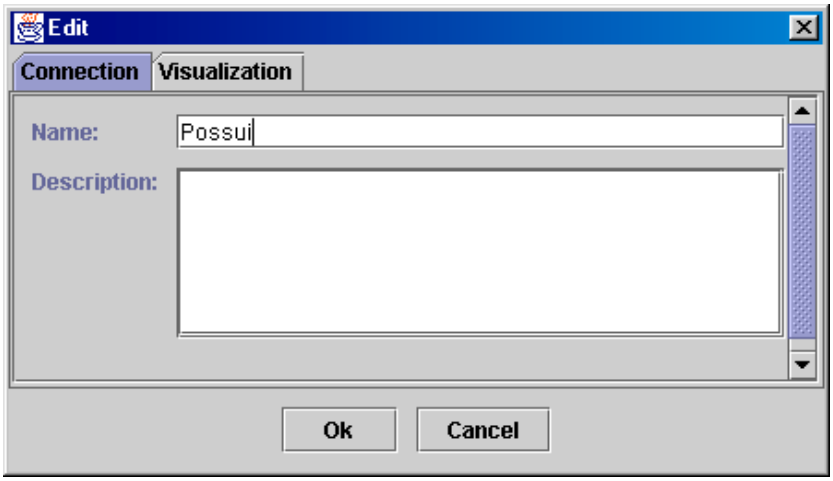

Figura 4.7: Padrão de Relacionamento

Cada componente, em todas as visões do modelo de domínio, possui seu próprio padrão. Nas classes eles detalham seus atributos e métodos, já nos *use-cases* são mostradas as condições de sucesso e de falha, por exemplo. O padrão de domínio do componente é sempre acessível com apenas um clique sobre o componente no lado esquerdo da janela, fornecendo suas informações e assumindo o papel de detalhá-lo de uma forma padronizada dentro do modelo.

Assim, o modelo de domínio é representado pelos diferentes diagramas, que implementam as diferentes visões do domínio, e por seus componentes, que possuem padrões detalhados com as informações específicas de cada um. O processo de Engenharia de Domínio do Odyssey é abordado na seção 3.2.1, e é descrito detalhadamente em (BRAGA e WERNER, 1999).

4.4.3 Suporte ao Processo de Engenharia de Aplicações

O suporte ao processo de EA implementado neste trabalho possui foco no apoio às atividades da fase de Análise (seção 3.4.2). A fase de análise é fundamental no desenvolvimento de uma aplicação, sendo as especificações de requisitos funcionais e conceituais de alto nível um passo muito importante no desenvolvimento de um aplicação correta e funcional, sendo seus resultados (componentes instanciados, por exemplo) utilizados nas demais fases do desenvolvimento.

Deste modo, as funcionalidades disponibilizadas pela infra-estrutura implementada são suficientes para a criação de um modelo de análise completo da aplicação, extraído diretamente do modelo de domínio, através das características de rastreabilidade do modelo, contendo inclusive grande parte dos conceitos necessários à sua implementação (como seus conceitos estruturais e sua codificação).

A criação de uma nova aplicação na infra-estrutura do Odyssey começa pela escolha de um domínio de aplicação na Janela Principal (Figura 4.3). Para criar uma nova aplicação, o desenvolvedor deve pressionar o botão *New* da Janela Principal. Neste momento, é dada a opção de escolha dos componentes do domínio (Figura 4.8). Obviamente, o modelo de domínio pode ser explorado livremente antes de executar esta tarefa.

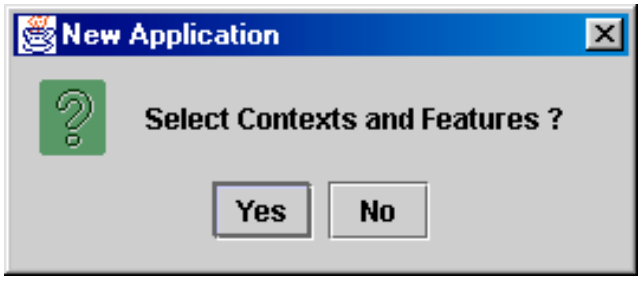

Figura 4.8: Seleção de novos componentes do domínio

Se o Engenheiro da Aplicação escolhe a opção negativa, ele começa o desenvolvimento de sua aplicação sem nenhum componente reutilizável do domínio. Porém, se ele escolhe a primeira opção, o que é desejável, ele será levado para uma janela de escolha de contextos e componentes da aplicação (Figura 4.9), de acordo com o previsto no Odyssey-EA (seção 3.4.2).

Podemos visualizar, na Figura 4.9, a janela de escolha dos contextos da aplicação. Na primeira aba da janela (*Application*), se define o nome e a descrição genérica da aplicação (que aparecerão em campos da Janela Principal, ver Figura 4.3). Na aba mostrada na figura (*Context*), escolhe-se os contextos nos quais a aplicação a ser desenvolvida se encaixa, através dos botões *Add*, *Add All*, *Remove* e *Remove All* que incluem ou retiram os contextos do Domínio (campo do lado esquerdo) na Aplicação (lado direito). A escolha destes contextos pré-seleciona, automaticamente, os componentes que serão sugeridos para o Engenheiro da Aplicação. Ou seja, a infra-estrutura só irá mostrar os componentes relativos aos contextos escolhidos pelo Engenheiro da Aplicação.

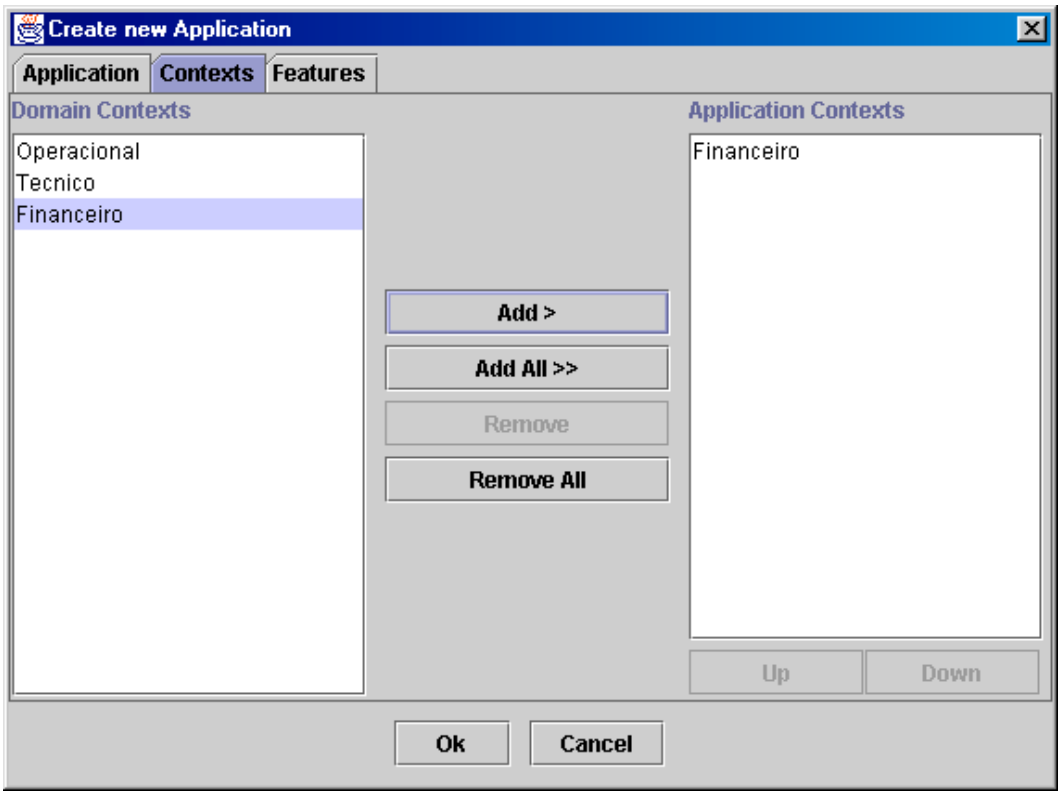

Figura 4.9: Escolha dos contextos do domínio

Na Figura 4.10, vemos os componentes (representados no diagrama de *features*) já com o corte inicial proporcionado pela definição do escopo feita na tarefa anterior. Podemos visualizar, na figura, as *features* funcionais do contexto *Financeiro*, incluindo as interfaces externas. Vemos também os diferentes tipos de *features*, sua natureza opcional ou mandatória, os relacionamentos e as restrições. A seleção é executada através do botão direito do mouse ou através do botão *Add*. O botão Remove retira a *feature* do campo de selecionadas (lado esquerdo).

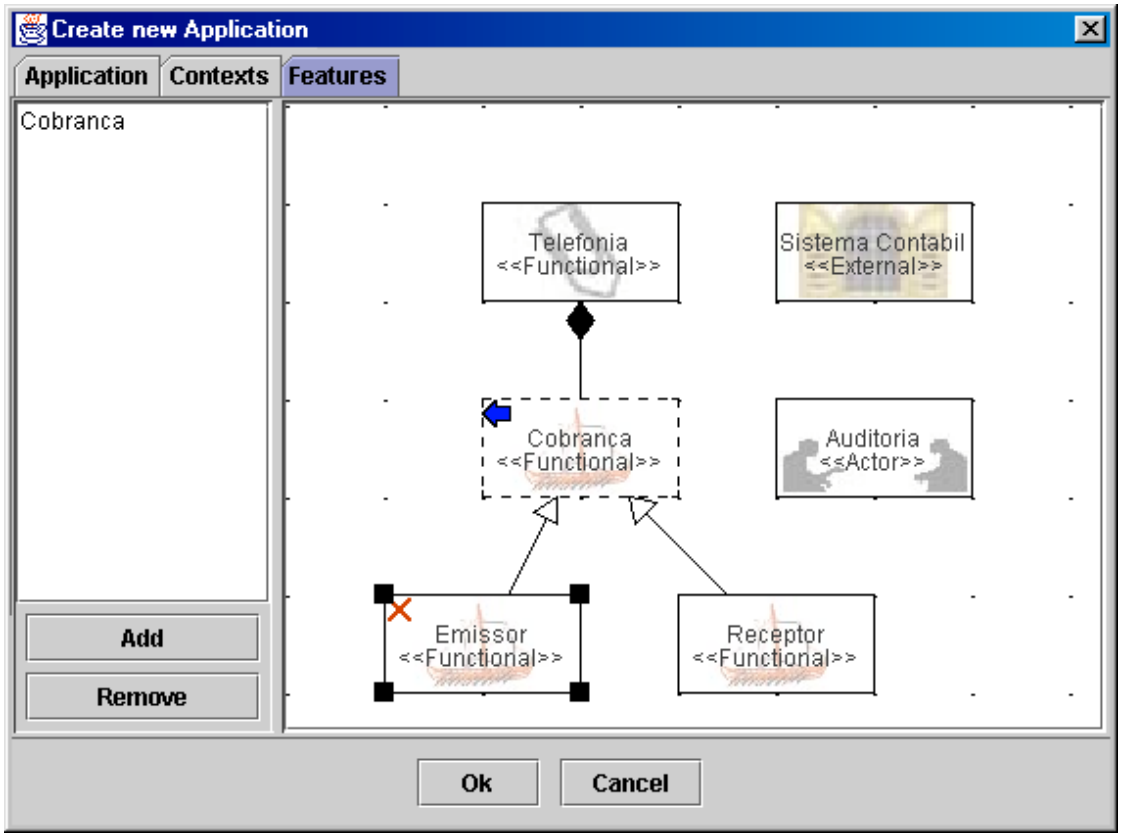

Figura 4.10: Seleção dos componentes de domínio, após definição do escopo

É neste momento do processo de EA que atua a política de seleção de componentes. Na escolha de cada *feature*, o Engenheiro da Aplicação recebe mensagens (Figura 4.11) mostrando os diferentes componentes relacionados, seja através de relacionamentos ou de restrições (seção 3.4.2). O tipo de cada *feature* detectada e a natureza do relacionamento entre ambas também são informados neste momento.

Na Figura 4.10, ao escolher a *feature Cobrança* são sugeridas ao Engenheiro da Aplicação as *features Emissor* e *Receptor*, ligadas a *feature* escolhida em primeiro lugar

por um relacionamento de agregação. Caso alguma destas *features* fosse opcional, este fato também seria informado neste momento.

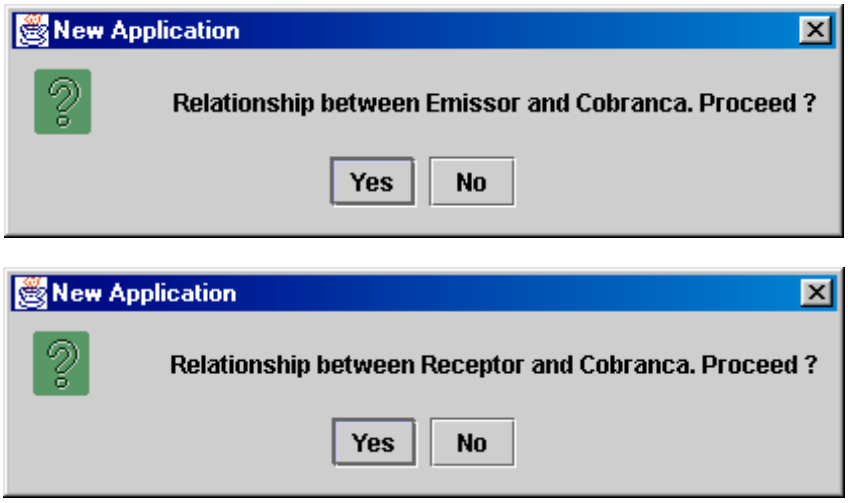

Figura 4.11: Política de Seleção de Componentes

Para localização dos componentes do domínio, que serão utilizados na aplicação a ser desenvolvida, podem ser utilizadas ferramentas adicionais. Um exemplo, é a ferramenta definida em (BRAGA *et al.*, 1999), que auxilia na localização de componentes, tendo como base variáveis como o perfil e os interesses do usuário.

Ao término deste processo, ocorre automaticamente uma verificação do processo de escolha dos componentes da aplicação (seção 3.4.2). Esta verificação mostra os relacionamentos mandatórios ignorados pelo desenvolvedor e/ou restrições não satisfeitas. Os relacionamentos fora do contexto da aplicação também são informados (como a restrição da *feature Emissor*). Assim, cabe ao Engenheiro da Aplicação avaliar a relevância de cada informação relatada e a solução para os problemas criados a cada passo da seleção. O resultado de uma verificação pode ser visto na Figura 4.12 e o formato final do modelo da aplicação, após a atividade de verificação, na Figura 4.13.

Após a verificação da seleção, os componentes do domínio são instanciados para o contexto da aplicação. A partir deste momento, eles se comportarão como componentes distintos, ligados aos componentes de domínio originais apenas através de um relacionamento de associação. Sendo assim, as modificações necessárias para adaptar os componentes de domínio, à realidade da aplicação, não se refletirão no modelo de domínio. Caso estas adaptações sejam relevantes ao contexto do domínio,

estas devem ser sugeridas a partir de um relatório de Avaliação do Processo de EA (seção 3.4.4).

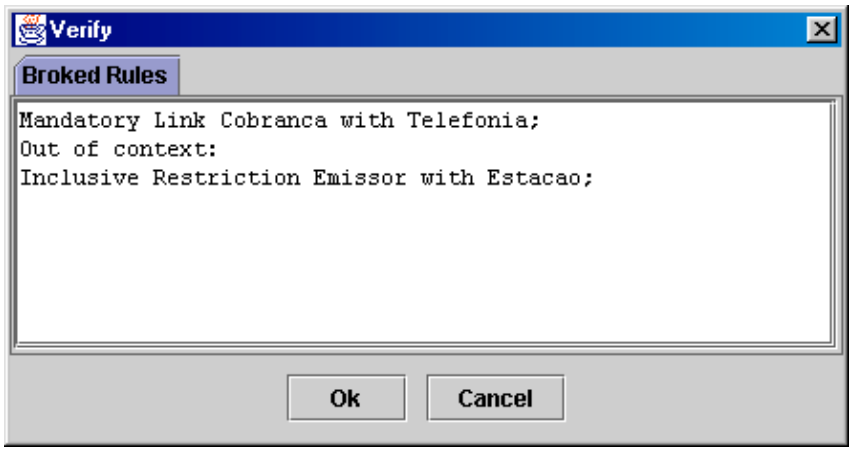

Figura 4.12: Verificação da Seleção de Componentes

A importância da infra-estrutura é muito grande para as atividades do Odyssey-EA aqui descritas. A escolha do contexto (com corte automático dos componentes), a política de seleção, a diagramação com características visuais e a instanciação automática dos componentes do domínio para a aplicação são alguns exemplos de como uma infra-estrutura de suporte pode auxiliar, de forma efetiva, o processo de desenvolvimento.

Todo o processo de instanciação de uma nova aplicação, a partir dos componentes do domínio é controlado por uma classe chamada *GerenteAplicacoes*. É esta estrutura que gerencia a criação dos diagramas de contexto e *features* baseados no modelo de domínio, a execução do corte nos componentes tendo como base o contexto, e a cópia dos componentes do domínio para o contexto da aplicação.

Após a adaptação dos componentes de domínio e a especificação de novos componentes, ambos feitos através dos diagramadores genéricos, é terminada a fase de análise. Neste momento, o Engenheiro da Aplicação deve especificar as Diretrizes de Projeto, que serão úteis para a instanciação da arquitetura da aplicação. Esta atividade, entretanto, faz parte de outro trabalho de tese ainda em desenvolvimento (XAVIER, 2000).

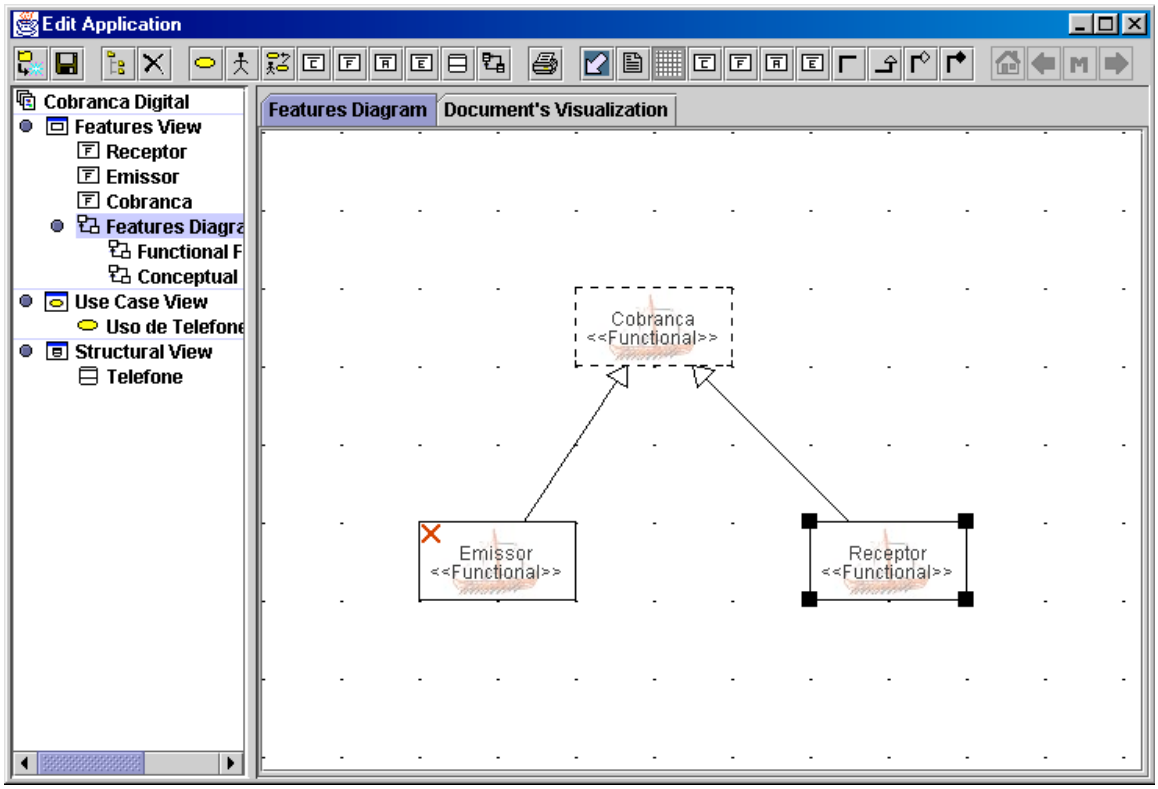

Figura 4.13: Aplicação instanciada

4.5 Conclusão

A implementação descrita neste capítulo visa possibilitar e facilitar a utilização do processo de Engenharia de Aplicações proposto nesta tese: o Odyssey-EA. Assim, tanto as modificações necessárias no suporte oferecido ao processo de ED, quanto as funcionalidades desenvolvidas para facilitar as atividades de EA, foram projetadas tendo em vista a semi-automatização destes processos.

Uma vez que a implementação da infra-estrutura de suporte a EA e as extensões do suporte ao processo de ED, descritas neste capítulo, foram realizadas no contexto do Projeto Odyssey, foram aproveitadas uma série de ferramentas já implementadas no contexto deste projeto, de acordo com a definição de BRAGA e WERNER (1999). Do mesmo modo, as funcionalidades aqui implementadas ficam disponíveis para o uso futuro em outros trabalhos.

A infra-estrutura de suporte, aqui descrita, para apoiar todo o processo de EA descrito no capítulo 3, necessita ainda de construção de ferramentas relativas as atividades específicas da fase de projeto e implementação, como a instanciação da

arquitetura (XAVIER, 2000). Porém, a implementação já realizada é suficiente para suportar toda a fase de análise e também a instanciação dos componentes relacionados às fases de projeto e implementação. Sendo a fase de análise uma etapa fundamental no desenvolvimento de uma aplicação, onde as especificações de requisitos funcionais e conceituais representam um passo muito importante no desenvolvimento de um aplicação funcional, propusemos dar ênfase a esta fase neste trabalho.

Capítulo 5

Conclusão

5.1 Visão geral

Segundo FREEMAN (1980), a reutilização de software caracteriza-se pela utilização de produtos de software desenvolvidos, ao longo de um ciclo de vida, em uma situação diferente da qual foram originalmente produzidos. A Engenharia de Aplicações é a área onde são feitos estudos sobre as melhores técnicas, processos e métodos para a produção de aplicações, no contexto do desenvolvimento baseado em reutilização (GRISS *et al.*, 1998).

Assim, um processo de EA visa construir uma aplicação maximizando o reuso. Isto é feito através da definição das atividades deste processo, da descrição das tarefas envolvidas na construção da aplicação, da eliminação de barreiras entre a modelagem dos componentes reutilizáveis e a construção da aplicação, e da implementação de uma infra-estrutura que suporte todos os aspectos peculiares deste tipo de desenvolvimento.

Este trabalho foi pautado por todos estes aspectos, sempre com o intuito de facilitar a utilização dos componentes do modelo de domínio pelo Engenheiro da Aplicação. Assim, acreditamos que, uma vez definido este modelo, seus componentes serão efetivamente utilizados na construção de aplicações reais, o que deve ser o objetivo principal de qualquer processo de reutilização.

5.2 Contribuições da abordagem

Nesta tese, identificamos todas as atividades e tarefas básicas envolvidas num processo genérico de EA. Estas atividades foram detalhadas de modo que possam ser bem compreendidas pelo Engenheiro da Aplicação, não delegando a ele a tarefa de definir e formatar o processo de reutilização.

A proposta aqui definida adotou pontos considerados de consenso na literatura, como a modelagem de classes e objetos feita através do padrão UML, mas detalhou outros pouco discutidos na literatura. Podemos citar como exemplos, a definição de uma política de seleção de componentes do domínio (através do modelo de *features*) e a

formalização da relação contexto/componentes, que simplifica o processo de escolha das funcionalidades da aplicação.

A sistemática do relacionamento entre os processos de ED e EA, que é fundamental no desenvolvimento baseado em reutilização, também foi enfatizada neste trabalho. Deste modo, foram sugeridas adaptações no modelo de domínio para que o processo de desenvolvimento baseado em reutilização como um todo pudesse se tornar mais efetivo, através da proposta do Modelo de *Features* Estendido.

Também foi implementada, no contexto deste trabalho, uma infra-estrutura de suporte que apoia parte do processo de EA proposto. Assim, tarefas repetitivas como a seleção de componentes relacionados e a atualização dos modelos da aplicação (baseados no domínio) são apoiadas pela infra-estrutura. Também através desta infra-estrutura, pode-se utilizar o conceito de rastreabilidade, definido no modelo (BRAGA e WERNER, 1999), para tarefas como: cópia para a aplicação de classes e *use-cases* relacionados às funcionalidades do domínio e cortes nos componentes disponíveis para a aplicação baseado no contexto escolhido.

5.3 Deficiências da proposta e tópicos para pesquisa futura

O processo proposto nesta tese é considerado suficiente para o desenvolvimento de uma aplicação no contexto de uma infra-estrutura como o Odyssey. Porém, obviamente, outras funcionalidades podem auxiliar sua execução, planejamento e acompanhamento.

Um possível auxílio para o processo proposto, é uma infra-estrutura de apoio à definição formal e ao acompanhamento do processo de desenvolvimento escolhido. Através desta infra-estrutura, poderiam ainda ser escolhidas as funcionalidades da disponíveis e analisados os possíveis impactos dos riscos presentes no desenvolvimento, caso eles ocorram. O próprio processo de EA poderia ser customizado de acordo com características específicas do projeto, diminuindo o número de atividades ou aumentando os pontos de controle. Uma infra-estrutura deste tipo será disponibilizada a partir dos trabalhos de (MURTA, 2000) e (BARROS, 2000b), ambos ainda em desenvolvimento no contexto do Projeto Odyssey.

Neste contexto, a agregação de outras funcionalidades à infra-estrutura de suporte Odyssey ao processo de EA também poderia ser feita mais facilmente, através

da definição das atividades onde elas se encaixam. Podemos citar como exemplo as ferramentas de geração de código (SOUZA e WERNER, 2000), localização de componentes (BRAGA *et al.*, 1999), análise de padrões e anti-padrões de modelagem (CORREA *et al.*, 2000) e a análise de regras estática de modelos conceituais (DANTAS, 2000).

Outro ponto do processo que deve ser auxiliado por uma ferramenta específica é a instanciação da arquitetura da aplicação. Assim, arquiteturas definidas no contexto do domínio devem ser utilizadas para o projeto da aplicação. Uma ferramenta deste tipo também está em desenvolvimento no contexto do Odyssey (XAVIER, 2000).

O uso de padrões de análise, projeto e implementação (GAMMA *et al.*, 1994) ainda carece de maior suporte automatizado no Odyssey. Também seria desejável para infra-estrutura a possibilidade de consultas simples e automáticas a estes recursos (previstas nos processos de EA e ED).

Entre os trabalhos futuros sugeridos no intuito de dar continuidade a este trabalho, podemos citar uma sistemática para avaliação do processo de EA no sentido de evolução do modelo de domínio, uma técnica para análise da dinâmica dos componentes na verificação dos modelos e a possibilidade da utilização de ferramentas externas de modelagem e codificação integradas à infra-estrutura.

Obviamente, há a necessidade de aplicação das propostas contidas neste trabalho em casos reais de desenvolvimento, para avaliar a efetividade das atividades detalhadas e a usabilidade da infra-estrutura de apoio. Está prevista a aplicação do processo Odyssey-EA no Domínio Legislativo, em um projeto de colaboração entre o grupo de reutilização do Programa de Engenharia de Sistemas e Computação da COPPE/UFRJ e uma casa legislativa no Rio de Janeiro.

Bibliografia

ARANGO, G., 1988, *Domain Engineering for Software Reuse*, Technical Report UCI-ICS 88-27, Universidade da California.

ARANGO, G., 1994, "Domain Analysis Methods", In: *Software Reusability*, Ellis Horwood Inc.

ARAUJO, M., 1998, *Automatização do Processo de Desenvolvimento de Software nos Ambientes Instanciados na Estação TABA*, Tese de M.Sc., COPPE/UFRJ, Rio de Janeiro, Brasil.

ARPA, 1994, *Architecture-Based Acquisition and Development of Software Guidelines and Recommendations from the ARPA Domain-Specific Software Architecture (DSSA) Program*, Technical Report, Teknowledge Federal Systems.

BARROS, M., WERNER, C., TRAVASSOS, G., 1999a, "Risk Analysis: a Key Success Factor for Complex System Development", *Proceedings of the 12th International Conference on Software & Systems Engineering and their Applications (ICSSEA'99)*, v. 5, paper 19-1, Paris, França.

BARROS, M., MURTA, L., DANTAS, A., 1999b, "Princípios de Desenvolvimento no Odyssey", Documento de trabalho interno, http://www.cos.ufrj.br/~odyssey.

BARROS, M., WERNER, C., TRAVASSOS, G., 2000a, "Applying System Dynamics to Scenario Based Software Risk Management", *2000 International System Dynamics Conference*, Bergen, Noruega, Agosto (aceito para publicação).

BARROS, M., 2000b, *Reutilização de Conhecimento no Gerenciamento de Projetos Baseado em Cenários*, In: Engenharia de Domínio e Desenvolvimento Baseado em Componentes, Relatório Técnico do Projeto Odyssey 10/2000, COPPE/UFRJ, Rio de Janeiro, Brasil.

BOOCH, G., RUMBAUGH, J., JACOBSON, I., 1998, *The Unified Modeling Language User Guide*, 1 ed., Nova York, Addison-Wesley Longman.

BOSCH, J., 1999, "Product Line Architectures in Industry: A Case Study", In: *Proceedings of the 21st International Conference of Software Engineering (ICSE'99)*, pp. 544-554, Los Angeles, EUA, Maio.

BRAGA, R., WERNER, C., 1999, "Odyssey-DE: Um Processo para Desenvolvimento de Componentes Reutilizáveis", *X Conferência Internacional em Tecnologia de Software (X CITS)*, Curitiba, Brasil, Maio.

BRAGA, R., WERNER, C., MATTOSO, M., 1999, "Odyssey: A Reuse Environment Based on Domain Models", *2nd IEEE Symposium on Application-Specific System and Software Engineering Technology (ASSET'99)*, Richardson, EUA, Março.

BRAGA, R., WERNER, C., MATTOSO, M., 2000, "Using Ontologies for Domain Retrieval", *11th International Conference and Workshop on Database and Expert Systems Applications (DEXA'00) - Workshop on Domain Engineering*, Greenwich, Inglaterra, Setembro (aceito para publicação).

BROWN, A., WALLNAU, K., 1998, "The Current State of CBSE", *IEEE Software*, Setembro/Outubro, pp. 37-46.

CHAN, S., LAMMERS, T., 1998, "OODE", *5th International Conference on Software Reuse (ICSR-5)*, Victoria, Canadá, Junho.

CHEONG, Y., JARZABEK, S., 1999, "Frame-based Method for Customizing Generic Software Architectures", *Proceedings of the Symposium on Software Reusability (SSR´99)*, pp. 103-112, Los Angeles, EUA, Maio.

CIMA, A., 1996, *Desenvolvimento de Software com Reutilização Baseada em "Frameworks" Orientados a Domínio*, Tese de M.Sc., COPPE/UFRJ, Rio de Janeiro, Brasil.

CLEMENS, P., 1997, "Successful Product Lines Requires More Than Reuse", *8º Workshop on Institutionalizing Software Reuse (WISR8)*, Columbus, EUA, Março.

COAD, P., YOURDON, M., 1992, *Análise Orientada a Objetos*, 2 ed., Rio de Janeiro, Editora McGraw-Hill.

CORREA, A., WERNER, C., ZAVERUCHA, G., 2000, "Object Oriented Design Expertise Reuse: an Approach based on Heuristics, Design Patterns and Anti-Patterns", *6th International Conference on Software Reuse (ICSR-6)*, Viena, Áustria, Junho.

CZARNECKI, K., 1997, "Leveraging Reuse Through Domain-Specific Software Architectures, *8th Workshop on Institutionalizing Software Reuse (WISR8)*, Columbus, EUA, Março.

DANTAS, A., 2000, *Agentes para Verificação de Consistência e Críticas de Modelos OO*, Projeto Final de Curso, COPPE/IM/UFRJ, Rio de Janeiro, Brasil (em andamento).

DIGRE, T., 1998, "Business Object Component Architecture", *IEEE Software*, Setembro/Outubro, pp. 60-69.

D´SOUZA, D., WILLS, A., 1998, *Objects, Components and Frameworks with UML: The Catalysis Approach*, 1 ed., Massachusetts, Addison-Wesley Longman..

FISCHER, M., 1992, "Domain-Oriented Design Environments", *IEEE Transactions on Software Engineering*, v. 18, n. 16 (Junho), pp. 511-522.

FREEMAN, M., 1980, *Reusable Software Engineering: A statement of long-range research objectives*, Technical Report TR-159, Universidade da California.

GAMMA, E., HELM, R., JOHNSON, R., *et al.*, 1994, *Design Patterns: Reuse of Object Oriented Design*, 1 ed., Nova York, Addison-Wesley Longman.

GOMAA, H., KERSCHBERG, L., SUGUMARAN, V., 1992, "A Knowledge-Based Approach to Generating Target System Specifications from a Domain Model", *IFIP World Computer Congress*, Madri, Espanha, Setembro.

GOMAA, H., 1995, "Reusable Software Requirements and Architectures for Families of Systems", *Journal of System Software*, Novembro, pp. 189-202.

GOMAA, H., FARRUKH, G., 1999, "A Reusable Architecture for Federated Client/Server Systems", *Proceedings of the Symposium on Software Reusability (SSR´99)*, pp. 113-121, Los Angeles, EUA, Maio.

GRISS, M., JETTE, C., KOZACZYNSKY, W., *et al.*, 1994, "Object-Oriented Reuse", *Panel on the 3rd International Conference on Software Reuse (ICSR-3)*, Rio de Janeiro, Brasil, Novembro.

GRISS, M., FAVARO, J., D'ALESSANDRO, M., 1998, "Integrating Feature Modeling with RSEB", *5th International Conference on Software Reuse (ICSR-5)*, ACM/IEEE, Victoria, Canadá, Junho.

GRISS, M., 1999a, "Domain Engineering and Reuse", *IEEE Computer*, Maio.

GRISS, M., 1999b, "Architecting Large Scale Systematic Component Reuse", *21st International Conference of Software Engineering (ICSE'99)*, Los Angeles, EUA, Maio.

GRISS, M., 1999c, "Reuse 2001-2004, What next, now that we have solved all reuse problems ?", *9th Workshop on Institutionalizing Software Reuse (WISR9)*, Austin, EUA, Janeiro.

GUARINO, N., 1998, "Formal Ontology and Information Systems", *1st International Conference on Formal Ontology in Information Systems*, IOS Press, Trento, Itália, Junho.

JACOBSON, I., CHRISTERSON, M., JONSSON, P., *et al.*, 1992, *Object-Oriented Software Engineering – A Use-case Driven Approach*, 1 ed., Massachusetts, Addison-Wesley Longman.

JACOBSON, I., GRISS, M., JONSSON, P., 1997, *Software Reuse. Architecture, Process and Organization for Business Success*, 1 ed., Nova York, Addison-Wesley Longman.

JACOBSON, I., BOOCH, G., RUMBAUGH, J., 1999, *The Unified Software Development Process*, 1 ed., Massachusetts, Addison-Wesley Longman.

JOHNSON, R., RUSSO, V., 1991, Reusing Object Oriented Designs, Technical Report UIUCDCS 91-1696, University of Illinois.

JOHNSON, R., 1992, "Documenting Frameworks Using Patterns", *Proceedings of the Conference on Object-Oriented Programming System, Languages and Interfaces (OOPSLA'92)*, pp. 63-76, Vancouver, Canadá, Outubro.

KANG, K., COHEN, S., HESS, J. *et al.*, 1990, *Feature-Oriented Domain Analysis (FODA) - Feasibility Study*, SEI Technical Report CMU/SEI-90-TR-21.

KANG, K., KIM, S., LEE, J., *et al.*, 1998, *FORM: A Feature-Oriented Reuse Method with Domain-Specific Reference Architectures*, SEI Technical Report.

KNIGHT, J., MYERS, E., 1993, "Improved Inspection Technique", *Communications of the ACM*, Novembro.

KRUEGER, C., 1992, "Software Reuse", In: *ACM Computing Surveys*, v. 24, n. 2, pp. 131-183.

LORD, H., 1994, "Visual Programming for Visual Applications", *Object Magazine*, Julho-Agosto.

MANNION, M., KAINDL, H., WHEADON, J., *et al.*, 1999, "Reusing Single System Requirements from Application Family Requirements", *Proceedings of the 21st International Conference of Software Engineering (ICSE'99)*, pp. 453-461, Los Angeles, EUA, Maio.

MAURO, R., ZIMBRÃO, Z., BRÜGGER, T., *et al.*, 1997, "GOA++: Tecnologia, implementação e extensões aos serviços de gerência de objetos", *Anais do 12º Simpósio Brasileiro de Banco de Dados (XII SBBD)*, pp. 272-286, Fortaleza, Brasil, Outubro.

MAYMIR-DUCHARME, F., 1997, "The Product Line Business Model", *8th Workshop on Institutionalizing Software Reuse (WISR8)*, Columbus, EUA, Março.

MILER, N., 1996, *Programação Visual*, In: Reutilização de Software: Tecnologias Emergentes, pp. 82-94, Relatório Técnico ES-377/96, COPPE/UFRJ, Rio de Janeiro, Brasil.

MURTA, L., 1999, *FRAMEDOC: Um FrameWork para a Documentação de componentes Reutilizáveis*, Projeto Final de Curso, DCC/IM/UFRJ, Novembro.

MURTA, L., 2000, *Uma Máquina de Processo de Desenvolvimento Baseado em Agentes Inteligentes*, Tese de M.Sc., COPPE/UFRJ, Rio de Janeiro, Brasil (em andamento).

NEIGHBORS, J., 1981, *Software Construction Using Components*, Tese de D.Sc., Universidade da California, Irvine, EUA.

PARNAS, 1972, "On the Criteria to Be Used in Decomposition Systems into Modules", *Communications of the ACM*, vol. 15, Dezembro, pp. 1053-1058.

PAULK, M., CURTIS, B., CHRISSIS, M., 1993, "Capability Maturity Model, Version 1.1", *IEEE Software*, Julho, pp. 18-27.

POULIN, J., 1997, "Software Architectures, Product Lines and DSSAs: Choosing the Appropriate Level of Abstraction", *8th Workshop on Institutionalizing Software Reuse (WISR8)*, Columbus, EUA, Março.

PRESSMAN, R., 1997, *Software Engineering – A practitioner's approach*, 3 ed., Cingapura, McGraw-Hill Editions.

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

RIG, 1999, "Reusability Library Interoperability Group", http://www.asset.com/rig.

ROCHA, A., WERNER, C., TRAVASSOS, G., *et al.*, 1996, "Processo de Desenvolvimento de Software Baseado em Reutilização", *VII Conferência Internacional de Tecnologia de Software (VII CITS)*, Curitiba, Brasil, Junho.

ROSETI, M., 1999, *Uma Proposta de Sistemática para Aquisição de Conhecimento no contexto de Análise de Domínio*, Tese de M.Sc., COPPE/UFRJ, Rio de Janeiro, Brasil.

SEACORD, R., HISSAN, S., WALLNAU, K., 1998, *Agora: A Search Engine for Software Components*, SEI Technical Report CMU/SEI-98-TR-011.

SILVA, M., WERNER, C., 1996, "Packing Reusable Components Using Patterns and Hypermedia", *4th International Conference on Software Reuse*, Orlando, EUA, Maio.

SIMOS, M., 1996, "Organization Domain Modeling (ODM): Domain Engineering as a Co-Methodology to Object-Oriented Techniques", In: *Fusion Newsletter*, v. 4, Hewlett-Packard Laboratories, pp. 13-16.

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

SOUZA, R., WERNER, C., 2000, *Um Gerador de Código para o Ambiente Odyssey*, Relatório Técnico do Projeto Odyssey 11/2000, COPPE/UFRJ, Rio de Janeiro, Brasil.

SUN, 2000, Página de referência do Java, http://www.java.sun.com.

TRACZ, W., 1994, *DSSA (Domain-Specific Software Architecture) – Pedagogical Example*, Technical Report ADAGE-IBM-94-13 - Review Version 2.0.

VASCONCELOS, F., 1997, *Reutilização de Processos de Desenvolvimento de Software Baseada em Padrões*, Tese de M.Sc., COPPE/UFRJ, Rio de Janeiro, Brasil.

VICI, A., 1998, "FODACom: An Experience with Domain Analysis in the Italian Telecom Industry", *5th International Conference on Software Reuse (ICSR-5)*, Victoria, Canadá, Junho.

VIEIRA, M., TRAVASSOS, G., 1998, "Apoio automatizado ao Teste Baseado no Comportamento de Sistemas Orientados a Objetos", *IX Conferência Internacional em Tecnologia de Software (IX CITS)*, Curitiba, Brasil, Junho.

WERNER, C., 1995, *Relatório Técnico sobre Reutilização*, COPPE/UFRJ, Rio de Janeiro, Brasil.

WERNER, C., MATTOSO, M., BRAGA, R., *et al.*, 1999, "Odyssey: Infra-estrutura de Reutilização baseada em Modelos de Domínio", *Caderno de Ferramentas do XIV Simpósio Brasileiro de Engenharia de Software (XIV SBES)*, pp.17-20, Florianópolis, Brasil, Outubro.

XAVIER, J., 2000, *Uma Ferramenta de Apoio à Definição e Instanciação de Arquiteturas Específicas de Domínio*, Tese de M.Sc., COPPE/UFRJ, Rio de Janeiro, Brasil (em andamento).