RAPDIS: UM PROCESSO E UM AMBIENTE MDA PARA O DESENVOLVIMENTO DE SISTEMAS DE INFORMAÇÃO

Gisele Pereira Morgado

Universidade Federal do Rio de Janeiro
Instituto de Matemática
Núcleo de Computação Eletrônica

Mestrado em Informática

Orientador: Eber Assis Schmitz
Co-orientadora: Priscila Machado Vieira Lima

Rio de Janeiro
2007

GISELE PEREIRA MORGADO

RAPDIS: UM PROCESSO E UM AMBIENTE PARA O DESENVOLVIMENTO DE SISTEMAS DE INFORMAÇÂO

UFRJ

V. I

Gisele Pereira Morgado

RAPDIS: UM PROCESSO E UM AMBIENTE MDA PARA O DESENVOLVIMENTO DE SISTEMAS DE INFORMAÇÃO

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Informática, Núcleo de Computação Eletrônica, Instituto de Matemática, Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Informática.

Orientador: Eber Assis Schmitz
Co-orientadora: Priscila Machado Vieira Lima

Rio de Janeiro
2007

MORGADO, Gisele Pereira.

RAPDIS: Um Processo e um Ambiente MDA para o Desenvolvimento de Sistemas de Informação. Rio de Janeiro: NCE / IM / UFRJ, 2007.

Dissertação (Mestrado em Informática) – Núcleo de Computação Eletrônica – NCE / Instituto de Matemática – IM / Universidade Federal do Rio de Janeiro – UFRJ, 2007.

Orientador: Eber Assis Schmitz
Co-orientador: Priscila Machado Vieira Lima

1. MDA. 2. Modelagem de Negócios. 3. Modelagem de Sistemas de Informação. 4. Processo de Desenvolvimento de Sistemas de Informação. I. Universidade Federal do Rio de Janeiro. II. Eber Assis Schmitz. III. Título.

Gisele Pereira Morgado

RAPDIS: UM PROCESSO E UM AMBIENTE MDA PARA O DESENVOLVIMENTO DE SISTEMAS DE INFORMAÇÃO

Rio de Janeiro, 27 de abril de 2007.

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Informática, Núcleo de Computação Eletrônica, Instituto de Matemática, Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Informática.

_____

Professor Eber Assis Schmitz, Ph.D.
NCE/IM/UFRJ (Orientador)

_____

Professora Priscila Machado Vieira Lima, Ph.D.
NCE/IM/UFRJ (Co-orientadora)

_____

Professora Maria Luiza Machado Campos, Ph.D.
NCE/IM/UFRJ

_____

Professor Alexandre Luis Correa, D. Sc.
UCAM

_____

Professor Ivan Mathias Filho, D. Sc.
PUC-Rio

AGRADECIMENTOS

Aos meus **pais Roberto e Juracilda**, pelo suporte emocional e por nunca terem medido esforços para me dar a melhor formação possível.

Ao meu **namorado Bruno**, pelo apoio em todas as minhas escolhas e iniciativas, pelo carinho e pela compreensão durante todos estes anos de graduação e pós-graduação.

Aos **amigos Maylle, Felipe, Bene, Iron, João, Pedro, Hugo e Carlos**, pelo companheirismo e por tornarem estes anos da minha vida mais divertidos.

A todos os **colegas do grupo de pesquisa de Gestão Estratégica de Tecnologia da Informação**, pela atenção e apoio durante este período.

Aos **alunos de graduação e mestrado que utilizaram o RAPDIS** por colaborarem com críticas e sugestões a este trabalho.

Aos **estagiários deste grupo Bruno, Gustavo e Leandro**, pela dedicação e profissionalismo que tornaram o projeto RAPDIS viável.

Ao **aluno de mestrado deste grupo Felipe**, por ter gerenciado este projeto na sua fase de manutenção e por ter assumido a responsabilidade de dar continuidade a este trabalho.

Aos meus **orientadores Eber e Priscila**, pelo conhecimento transmitido, pelo incentivo e, principalmente, por acreditarem em mim.

Aos **professores Ivan Mathias Filho, Alexandre Luis Correa e Maria Luiza Machado Campos**, pela gentileza de aceitar o convite para participar da avaliação deste trabalho.

À **Universidade Federal do Rio de Janeiro,** por oferecer uma formação de excelente qualidade mesmo com todas as dificuldades que esta instituição enfrenta.

Ao **Núcleo de Computação Eletrônica** e ao **Conselho Nacional de Desenvolvimento Científico e Tecnológico**, pelo suporte financeiro e logístico.

E, finalmente, a todos aqueles que direta ou indiretamente contribuíram na elaboração deste trabalho.

RESUMO

MORGADO, Gisele Pereira. **RAPDIS:** Um Processo e um Ambiente MDA para o Desenvolvimento de Sistemas de Informação. Dissertação (Mestrado em Informática) – Núcleo de Computação Eletrônica, Instituto de Matemática, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2007.

O desenvolvimento de sistemas de informação é uma atividade complexa. Para tentar lidar com esta complexidade uma grande variedade de idéias traduzidas em conceitos, métodos e ferramentas foram criadas pela comunidade de profissionais de tecnologia da informação. Apesar de cada uma destas idéias poderem individualmente gerar benefícios, o potencial delas pode ser apenas alcançado através da sua integração. Este trabalho apresenta um processo de desenvolvimento de sistemas (processo RAPDIS) e um ambiente para apoiar as atividades deste processo (ambiente RAPDIS). O processo proposto é resultado da integração dos conceitos da MDA (*Model Driven Architecture*) com alguns trabalhos desenvolvidos pelo grupo GETI (Gestão Estratégica em Tecnologia da Informação) na área de modelagem de negócios e modelagem de sistemas de informação. Sua aplicação não somente agiliza e facilita as tarefas dos profissionais de tecnologia da informação, como também possibilita o desenvolvimento de sistemas que atendam de forma eficiente aos negócios a que se destinam.

Palavras Chaves: Arquitetura Dirigida a Modelos, Modelagem de Negócios, Modelagem de Sistemas de Informação, Processo de Desenvolvimento de Sistemas de Informação.

ABSTRACT

MORGADO, Gisele Pereira. ***RAPDIS:*** *A MDA Process and Environment for the Development of Information Systems. Thesis (Master Degree in Computer Science) – Núcleo de Computação Eletrônica, Instituto de Matemática, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2007.*

The development of Information Systems is a complex activity. To deal with this high complexity, many ideas translated into concepts, methods and tools have been created by the Information Technology community. Although these ideas may individually generate benefits, their potential can only be reached through the integration. This work presents a process for the development of Information Systems (RAPDIS process) and an environment to support the activities of this process (RAPDIS environment). The proposed process is a result of the integration of MDA (Model Driven Achitecture) concepts and some works developed by the group GETI (Gestão Estratégica em Tecnologia da Informação) related to Business Modeling and Information Systems Modeling. The application of this process speeds up the tasks of Information Technology professionals and make possible the development of systems that are more aligned to the business needs.

Keywords: *Model Driven Architecture, Business Modeling, Information Systems Modeling, Information Systems Development Process.*

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE SIGLAS E ABREVIATURAS

AS	*Action Semantics*
CAD	*Computer-Aided Design*
CAP	*Computer-Aided Programming*
CASA	*Computer-Aided Systems Analysis*
CASE	*Computer-Aided Software Engineering*
CIM	*Computation Idependent Model*
CWM	*Common Warehouse Meta-model*
GETI	Gestão Estratégica de Tecnologia da Informação
HAG	Heurísticas para Ajuste Geral
HIA	Heurísticas para Identificação de Atores
HIC	Heurísticas para Identificação de Casos de Uso
HIR	Heurísticas para Identificação de Relações
HPReq	Heurísticas para transformação Processo – Requisitos
MDA	*Model Driven Architecture*
MSI2	Modelagem de Sistemas de Informação 2
MVC	*Model - View - Controller*
OCL	*Object Constraint Language*
OMG	*Object Management Group*
PIM	*Platform Independent Model*
PROLOG	*PROgrammation en LOGique*
PSM	*Platform Specific Model*
QVT	*Query View Transformation*
RAPDIS	*Rules And Process for the Development of Information Systems*
UML	*Unified Modeling Language*
XMI	*XML Metadata Interchange*
XML	*eXtensible Markup Language*

SUMÁRIO

1. INTRODUÇÃO

"Idéias e descobertas tecnológicas
são forças propulsoras do
crescimento econômico."

The Wall Street Journal

1.1 Motivação

Traduzir as necessidades de uma organização em um sistema que apóie efetivamente suas operações não é uma tarefa fácil. Um estudo americano (STANDISH GROUP, 2004) aponta que apenas 28% dos projetos desenvolvidos atingem os resultados planejados, podendo ser considerados projetos de sucesso. Por outro lado, 72% não atingem os seus objetivos, ou seja, são entregues fora do prazo, ultrapassam os seus orçamentos ou não apresentam a qualidade desejada.

Não há uma explicação simples para esse fato. Uma das possíveis explicações para essa alta taxa de fracassos é que o desenvolvimento de software é, e sempre será, uma atividade essencialmente complexa. Segundo (BROOKS, 1986), sistemas de software possuem um número de componentes maior do que qualquer outro tipo de sistema desenvolvido pelo homem. Além disso, software é constituído de bits e não de átomos. Não sendo concreto, não é visualizável, e o seu desenvolvimento depende fortemente da criatividade humana.

Uma das maneiras de enfrentar essa complexidade consiste na utilização de modelos (MELLOR *et al*, 2004). Pensando nisso, o grupo de Gestão Estratégica de Tecnologia da Informação (GETI) vem desenvolvendo várias técnicas, métodos e ferramentas. Os principais trabalhos do grupo são: a ferramenta *Régula* (MORGADO, 2005) de gerência de Regras de Negócio e o seu gerador do modelo de domínio (MARTINS, 2005); a ferramenta *FastCase* (SILVEIRA, 1999) que permite a modelagem de sistemas utilizando UML e seu

o gerador de código (PINTO, 2004); o *HPReq* (CRUZ, 2004) que apresenta heurísticas para a identificação de requisitos a partir de modelos de processo; e o MRDS (SCHMITZ; SILVEIRA, 2000) que consiste em uma metodologia rápida para o desenvolvimento de sistemas.

Paralelamente, o Grupo de Gerenciamento de Objetos[1] (*Object Management Group*, OMG) criou a Arquitetura Dirigida a Modelos (*Model Driven Architecture*, MDA) (MUKERJI; MILLER, 2003). A MDA foi construída baseada na idéia de que os modelos são os artefatos mais importantes dentro do desenvolvimento de sistemas. Eles estão presentes e devem guiar todo o processo de desenvolvimento: desde a fase inicial, onde são capturadas as necessidades do negócio que são atendidas pelo sistema, até as fases finais, onde são elaboradas as especificações técnicas desse sistema. Além disso, na MDA, os modelos são considerados ativos (bens) e não despesas, isto é, os modelos são especificações formais que podem ser lidas por uma máquina (computador) e transformadas até se chegar ao código do sistema.

Todos os trabalhos citados auxiliam, de alguma forma, o desenvolvimento de sistemas de informação. Porém, eles representam projetos isolados, não havendo uma integração adequada nem um processo para guiar a sua utilização. A necessidade de uma forma estruturada e de um ambiente integrado que possibilitem a aplicação destes conceitos constitui a principal motivação deste trabalho.

[1] *Object Management Group*. Grupo para produção e manutenção de especificações para a indústria da computação. Disponível em www.omg.org.

1.2 Objetivo

O objetivo deste trabalho é elaborar um processo de desenvolvimento de sistemas de informação (processo RAPDIS) e criar de um ambiente integrado que viabilize a execução das atividades definidas neste processo (ambiente RAPDIS). Serão aproveitadas a metodologia e as ferramentas já desenvolvidas pelo grupo GETI e, além disso, serão incorporados os conceitos da MDA.

1.3 Evolução do Trabalho

Desde 1999, o grupo de Gestão Estratégica de Tecnologia da Informação (GETI) tem estudado formas de se construir e transformar modelos de sistemas de informação. Os principais trabalhos do grupo relacionados a essa linha de pesquisa são apresentados em ordem cronológica no quadro 1.1 a seguir.

Quadro 1.1 - Trabalhos do grupo GETI

Projeto FASTCASE	
Descrição	O FASTCASE foi criado para auxiliar no desenvolvimento de sistemas Orientados a Objetos. A ferramenta é composta por um desenhador acoplado a um mecanismo de definição e por um gerador de código fonte. O desenhador permite a construção de diagramas da UML. O mecanismo de definição permite a associação de textos a objetos dos diagramas. O gerador de código fonte segue uma arquitetura padrão definida pela Metodologia Rápida de Desenvolvimento de Sistemas (MRDS).
Principais Publicações	• **(SILVEIRA, 1999)** Dissertação de Mestrado sobre a ferramenta FASTCASE. • **(SILVEIRA; SCHMITZ, 1999)** Artigo no Simpósio Brasileiro de Engenharia de Software sobre a ferramenta. Premiada como melhor Ferramenta. • **(SCHMITZ; SILVEIRA, 2000)** Livro Desenvolvimento de Software Orientado a Objetos. • **(PINTO; SILVEIRA; SCHMITZ, 2000)** Artigo no Simpósio Brasileiro de Engenharia de Software sobre a ferramenta. • **(PINTO, 2004)** Dissertação de Mestrado sobre o gerador de código da ferramenta FASTCASE.

Projeto HPREQ	
Descrição	As Heurísticas para transformação de Processo em Requisitos (HPRQ) foram desenvolvidas com o objetivo de acelerar a obtenção dos modelos de requisitos funcionais através da transformação automática do diagrama de processos em diagrama de casos de uso.
Principais Publicações	• **(CRUZ; SILVEIRA; SCHMITZ, 2002)** Artigo na Conferência da Associação Portuguesa de Ambientes de Informação sobre as heurísticas para identificação de requisitos a partir de modelos de processo. • **(CRUZ, 2004)** Dissertação de Mestrado sobre as heurísticas para identificação de requisitos a partir de modelos de processo.
Projeto RÉGULA	
Descrição	O RÉGULA permite a captura, o armazenamento e a edição de parte dos requisitos de software expressos como Regras de Negócio, auxiliando o processo de desenvolvimento de software. A partir dos termos e das Regras de Negócio armazenados em seu repositório, o RÉGULA gera como produtos computacionais um conjunto de regras executáveis e um modelo de informação aderente a estas regras.
Principais Publicações	• **(DIAS *et al*, 2004)** Artigo no Simpósio Brasileiro de Sistemas de Informação sobre a Modelagem Organizacional baseada em Regras de Negócio. • **(MORGADO, 2005)** Projeto Final de Curso sobre a ferramenta Regula. • **(MARTINS, 2005)** Dissertação de Mestrado sobre captura e representação sistemática das definições dos termos das Regras de Negócio. • **(ARAÚJO *et al*, 2006)** Artigo no Simpósio Brasileiro de Engenharia de Software sobre a ferramenta. • **(MORGADO, 2006)** Artigo no Simpósio Brasileiro de Sistemas de Informação resumindo o Projeto Final de Curso. Prêmio de melhor Trabalho de Conclusão de Curso.

A partir de 2005, o grupo GETI começou a trabalhar na idéia de integração desses projetos. Essa integração foi realizada utilizando a MDA, originando o trabalho apresentado nesta dissertação de mestrado. A descrição deste projeto de integração (projeto RAPDIS) e as suas publicações são apresentadas no quadro 1.2.

Quadro 1.2 - Trabalho de Integração do grupo GETI

Projeto RAPDIS	
Descrição	O RAPDIS consiste em um processo e um ambiente MDA para o desenvolvimento de sistemas de informação. Através dele é possível construir modelos com alto nível de abstração e transformá-los automaticamente em modelos com baixa abstração, facilitando e tornando mais rápido o processo de desenvolvimento.
Principais Publicações	• **(DIAS *et al*, 2006)** Artigo no *IX Workshop on Requirements Engineering* sobre a transformação automática do Modelo de Negócio em Modelo de Requisitos através do ambiente RAPDIS. • **(MORGADO *et al*, 2006)** Artigo no Simpósio Brasileiro de Sistemas de Informação sobre o processo RAPDIS que apresenta uma forma estruturada de se utilizar a MDA no desenvolvimento de Sistemas de Informação.

Vale notar que o projeto RAPDIS continua em evolução. A definição da MDA, que serviu de base para este trabalho, é uma proposta recente e há muitos tópicos que ainda estão sendo estudados pelo OMG. Deste modo, a expectativa é que este trabalho sirva como base para outros projetos que serão realizados dentro do grupo GETI.

1.4 Contribuições

Este trabalho oferece como sua principal contribuição a integração, através da MDA, dos diversos trabalhos desenvolvidos pelo grupo GETI. O processo RAPDIS estabelece uma forma estruturada de se aplicar a MDA, abrangendo as diversas etapas do processo de desenvolvimento, desde a Modelagem de Negócios até a codificação do Sistema de Informação. O ambiente RAPDIS viabiliza a utilização do processo proposto, automatizando as atividades de construção e transformação de modelos.

Por automatizar a transformação do Modelo de Negócio em Modelo de Sistema, espera-se que os sistemas construídos com o RAPDIS atendam, de forma mais eficiente, aos negócios a que se destinam. Por automatizar a transformação do Modelo de Sistema em

código, espera-se que o trabalho dos desenvolvedores seja facilitado, eliminado alguns passos repetitivos da fase de codificação. Além disso, uma contribuição indireta deste trabalho é a possibilidade de se utilizar o RAPDIS como uma ferramenta de ensino de Engenharia de Sistemas.

1.5 Organização da Dissertação

Esta dissertação foi estruturada em cinco capítulos. Além deste primeiro capítulo introdutório, o segundo capítulo apresenta os conceitos relacionados à Engenharia de Sistemas que serviram de base para este trabalho. No terceiro capítulo, introduzimos o processo RAPDIS, destacando os seus produtos e atividades. No quarto capítulo, mostramos as funcionalidades e a arquitetura do ambiente RAPDIS. No quinto capítulo, avaliamos o RAPDIS através de entrevistas com os usuários e de uma comparação com outras ferramentas de MDA disponíveis no mercado. E, finalmente, no sexto capítulo, apresentamos as conclusões deste trabalho.

2. ENGENHARIA DE SISTEMAS

"Mais que uma atividade ou um campo de conhecimento, engenharia é uma palavra de ação, uma forma de abordar um problema."

Scott Whitmire

2.1 Introdução

Sistemas de informação têm enorme potencial de trazer benefícios. Mas também podem trazer muitos prejuízos, quando feitos de forma errada (PAULA, 2003). A Engenharia de Sistemas ajuda a traduzir as necessidades de um negócio num Modelo de Sistema (PRESSMAN, 2001). Para isso, são necessários processos, métodos e ferramentas automatizadas. Este capítulo aborda algumas das práticas mais consagradas desta disciplina que foram adotadas no trabalho desenvolvido.

2.2 Processo de Engenharia de Sistemas

Quando se desenvolve um sistema é importante percorrer uma série de passos previsíveis – um roteiro que ajude a criar a tempo um resultado de alta qualidade (LARRUCEA, 2004). Processos de Engenharia de Sistemas especificam a seqüência de passos a serem seguidos durante o desenvolvimento de um sistema de informação. A cada um desses passos, associa-se um conjunto de atividades, seus produtos e as regras de verificação que garantem a passagem para a próxima fase (PRESSMAN, 2001).

2.2.1 Processo MDA de Engenharia de Sistemas

A **MDA** é uma modelo de processo para o desenvolvimento de sistemas criada pelo grupo OMG. O objetivo da MDA é separar as decisões orientadas ao negócio das decisões de plataforma, permitindo assim maior flexibilidade durante as fases de especificação e de

desenvolvimento de sistemas (FRANKEL, 2003). Para atingir esse objetivo, um processo MDA define modelos com diferentes níveis de abstração: o **Modelo Independente de Computação** (*Computation Idependent Model*, CIM), o **Modelo Independente de Plataforma** (*Platform Independent Model*, PIM) e o **Modelo Específico de Plataforma** (*Platform Specific Model*, PSM).

O CIM é o modelo de mais alto nível do sistema, representando as possibilidades e funcionalidades deste que não dependem de computação. Este modelo é importante para o entendimento do problema e serve como fonte do vocabulário a ser usado nos demais modelos. Em (MUKERJI; MILLER, 2003), o modelo independente de computação é caracterizado como sendo um mediador entre as pessoas que são especialistas em relação ao negócio e aquelas que são especialistas no projeto e construção dos artefatos do sistema.

O PIM descreve o sistema, porém não apresenta os detalhes da tecnologia que será usada na implementação. Este modelo oferece especificações formais da estrutura e funcionalidade do sistema, abstraindo-se detalhes técnicos. Por possuir um alto grau de independência em relação à plataforma, um mesmo PIM pode ser utilizado para um conjunto de diferentes plataformas.

O PSM combina a especificação do modelo PIM com detalhes de uma determinada plataforma. Através deste modelo são especificados como os serviços da plataforma escolhida serão utilizados para implementar as funcionalidades do sistema. Este modelo é o de mais baixo nível de abstração e os seus elementos estão prontos para a geração de código (BLANC; GERVAIS; SRIPLAKINCH, 2004).

Entre cada um destes modelos há uma série de transformações que são aplicadas para que se possa chegar até uma plataforma específica. A transformação de modelos é a chave principal da MDA e consiste no processo de converter um modelo em outro modelo

do mesmo sistema (MUKERJI; MILLER, 2003). A idéia principal é construir modelos no seu mais alto nível de abstração e transformá-los em modelos com baixa abstração, de forma automática ou semi-automática, facilitando e tornando mais rápido o processo de desenvolvimento (figura 2.1).

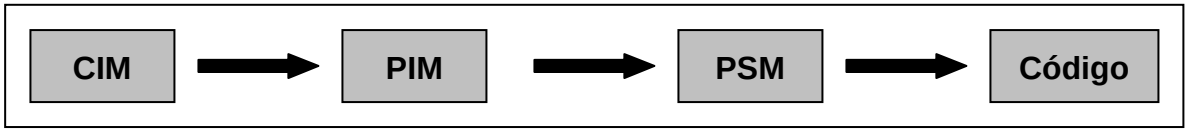


Figura 2.1 - Modelos e Transformações da MDA

2.2.2 Vantagens do Processo MDA

De acordo com (KEPPLE; WARMER; BAST, 2003) e (MELLOR *et al*, 2004), os principais benefícios da utilização de um processo MDA são:

- **Produtividade** - as transformações do CIM para o PIM e do PIM para o PSM precisam ser definidas uma única vez e podem ser aplicadas no desenvolvimento de diversos sistemas. Uma vez definidas as transformações, há uma redução no tempo necessário para a construção de modelos de mais baixo nível.
- **Portabilidade** - dentro da MDA a portabilidade é alcançada através do foco dado ao desenvolvimento do PIM, que é independente de plataforma. Um mesmo PIM pode ser automaticamente transformado em vários PSMs de diferentes plataformas.
- **Interoperabilidade** - diferentes PSMs gerados a partir de um mesmo PIM podem conter ligações entre eles, denominadas em MDA de *pontes*. Uma *ponte* pode ser construída através das especificações técnicas das plataformas referentes aos modelos PSMs e de como os elementos existentes no modelo PIM foram transformados nos modelo PSMs.

- **Manutenção e documentação** - o CIM é utilizado para gerar o PIM, o PIM é utilizado para gerar o PSM e o PSM é convertido em código. Assim, o modelo é a representação exata do código. Quando surge uma solicitação de alteração da especificação do sistema, o modelo deve ser alterado e o código regerado. Isso permite que a documentação do sistema se mantenha consistente com a sua implementação durante todo o processo de desenvolvimento.

2.3 Métodos de Engenharia de Sistemas

Métodos de Engenharia de Sistemas fornecem a técnica de como fazer a construção de um sistema de informação. Esses métodos repousam sobre um conjunto de princípios básicos que regem cada área da tecnologia e incluem atividades de modelagem e outras técnicas descritivas (PRESSMAN, 2001). Neste trabalho, dividimos a tarefa de modelagem em duas etapas (a Modelagem de Negócios e a Modelagem de Sistemas de Informação), e utilizamos técnicas diferentes em cada etapa. Aplicamos também alguns métodos para realizar a transformação entre os modelos, com o objetivo de facilitar o trabalho dos profissionais responsáveis pelo projeto e pela construção dos artefatos do sistema.

2.3.1 Técnicas para a Modelagem de Negócios

Um Modelo do Negócio é uma abstração do funcionamento do próprio negócio. Tal modelo é composto pelos seguintes objetos: Objetivos, Recursos, Processos e Regras (ERIKSSON; PENKER, 2000).

Os **Objetivos** são os propósitos do negócio, ou simplesmente, o resultado que toda a organização deseja atingir. Para que uma atividade do negócio se encaminhe em direção a um dado objetivo, ela deve ser realizada de acordo com uma ou mais metas relacionadas a

esse objetivo. Uma meta é um objetivo quantificado, ou seja, é a obtenção de um valor para uma variável do negócio dentro de um determinado prazo (KOUBARAKIS; PLEXOUSAKIS, 2002).

Os **Processos** constituem um conjunto de atividades estruturadas e medidas para que um produto (bem ou serviço) seja gerado. São, portanto, uma ordenação específica do trabalho no tempo e no espaço, com um começo, meio e fim, entradas e saídas claramente identificados, dispostos segundo a ordem de precedência dessas atividades. São governados por Regras e, quando corretamente executados, satisfazem a um objetivo específico.

Os **Recursos** constituem os objetos utilizados pelos processos, tais como uma pessoa, material, informação ou produto, usado ou produzido no negócio. São organizados em estruturas e possuem relações entre si.

As **Regras** são declarações que restringem, derivam e fornecem condições de existência, representando o conhecimento do negócio. Especificam os estados possíveis do negócio, incluindo os estados dos Recursos. Podem ser estabelecidas quer por regulamentação externa (contexto jurídico), quer por regulamentação interna, de modo que os objetivos do negócio sejam atingidos.

Como pode ser observado pela própria definição apresentada, existem alguns relacionamentos entre cada um dos componentes do negócio. Esses relacionamentos são importantes para a construção de um Modelo de Negócio consistente, e podem ser vistos na figura 2.2 apresentada a seguir.

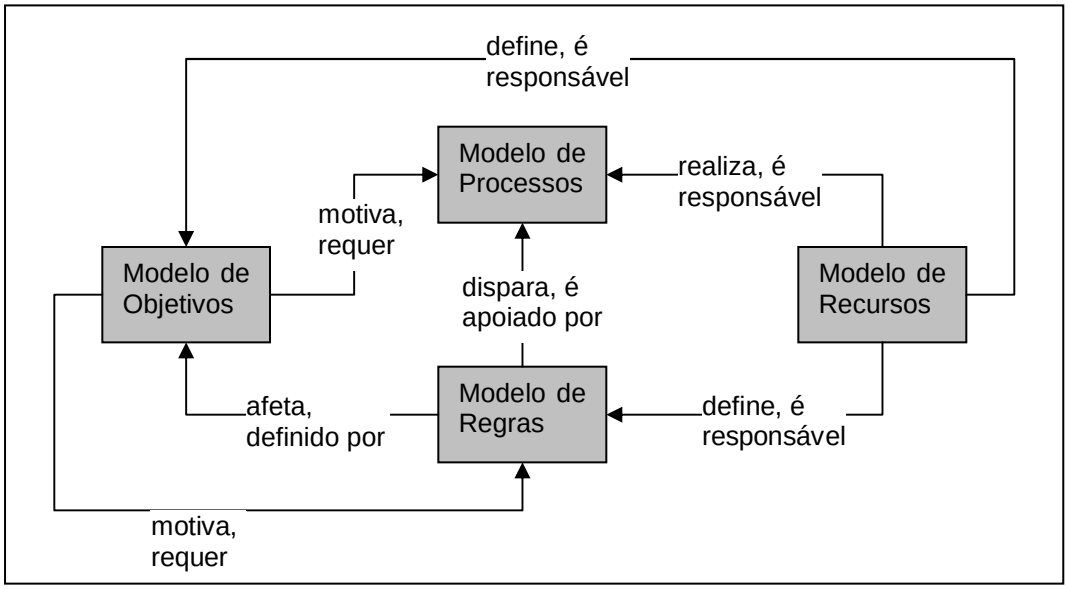


Figura 2.2 - Relacionamentos entre os componentes do negócio (BUBENKO; STIRNA; BRASH, 1998)

Embora todos esses objetos sejam fundamentais para o entendimento do negócio que está sendo modelado, neste trabalho focaremos nos Processos e nas Regras de Negócio. Focamos nesses objetos, porque é a partir deles que é definida a maior parte dos requisitos dos sistemas de informação que apoiarão as operações de um determinado negócio (LARMAN, 2004). As técnicas utilizadas para a modelagem desses dois objetos serão apresentados nas seções 2.3.1.1 e 2.3.1.2.

2.3.1.1 Técnica para a Modelagem de Processos

Segundo (ESHUIS; WIERINGA, 2001), os Processos podem ser representados através do Diagrama de Atividades da Linguagem de Modelagem Unificada (*Unified Modeling Language*, UML). A UML é "uma linguagem para especificação, visualização, construção e documentação de artefatos de sistemas de software, assim como para Modelagem de Negócios e outros tipos de sistemas (OMG, 2005)."

O **Diagrama de Atividades** constitui um elemento de modelagem simples, porém eficaz para descrever o fluxo de trabalho numa organização. Entre as suas características, destaca-se a possibilidade de representação de atividades concorrentes, com execuções em paralelo, diferenciando-o de um mero fluxograma (BOOCH; RUMBAUGH; JACOBSON, 1999).

Neste diagrama, as atividades são situadas em raias de responsabilidade, representativas dos locais onde elas são executadas. Nestes locais existem Recursos que interagem com as atividades durante sua execução. São representados, também, os insumos consumidos, Recursos utilizados e produtos gerados na forma de objetos relacionados às atividades.

2.3.1.2 Técnica para a Modelagem de Regras de Negócio

De acordo com a linha adotada por (BRG, 1997), uma forma de descrever as Regras de Negócio é através de um subconjunto da língua portuguesa, ao qual chamamos de Português Estruturado. Por um lado, esta forma de representação estruturada tem a aparência da linguagem natural e, por outro lado, é uma representação consistente e não ambígua do conhecimento.

A representação do **Português Estruturado** baseia-se no uso de modelos de sentença como forma de viabilizar a representação das Regras de Negócio. Um modelo de sentença é uma seqüência padronizada de termos usada para construir Regras de Negócio (ALENQUER, 2002). Cada categoria de regra deve ter um modelo de sentença característico. Segundo o modelo definido em (BRG, 1997), as Regras podem ser agrupadas em seis categorias: termos, fatos, cálculos, derivações, restrições e habilitações de ação.

Os termos constituem os elementos básicos da linguagem utilizada para expressar as Regras de Negócio, onde a própria definição de um termo é considerada como uma regra. Fatos descrevem a natureza ou estrutura operacional de um negócio, relacionando os termos uns aos outros. Os cálculos e derivações determinam como um conhecimento ou informação pode ser transformado em outro, através de fórmulas ou mudanças de estado. Já as restrições, conforme o nome indica, restringem algum comportamento do negócio, estando relacionadas a decisões sobre quais dados podem ou não ser atualizados. As habilitações de ação podem ser vistas como regras dedutivas, de raciocínio encadeado à frente, representadas através de um par contendo uma condição e respectiva ação. No quadro 2.1 apresentamos os modelos de sentença utilizados neste trabalho separados por categoria[2].

Quadro 2.1 - Modelos de Sentença Separados por Categoria de Regra (MORGADO, 2005).

Categoria	Modelo de sentença
Fato	<termo> É CATEGORIA BÁSICA
	<termo1> É SINÔNIMO DE <termo2>
	<termo1> É SUBTIPO DE <termo2> [QUE <verbo_restrição1> <termo_relacionado_restrição1>, <verbo_restrição2> <termo_relacionado_restrição2> ... E <verbo_restriçãoN> <termo_relacionado_restriçãoN>]
	<termo1> TEM COMO ATRIBUTO <termo2>
	<termo> POSSUI COMO DOMÍNIO <domínio>
	<termo1> TEM COMO PARTE <termo2>
	<termo1> ESTÁ RELACIONADO A <termo2> POR <verbo_associação> [COM GRAU <M>..<N>]
Cálculo	<termo> É CALCULADO COMO <função>
Derivação	SE <condição>, ENTÃO <termo> É CONSIDERADO COMO <estado>
Habilitação de ação	SE <condição>, ENTÃO EXECUTAR <ação>
Permissão	<termo1> TEM PERMISSÃO PARA <verbo> {<prep>} {<artigo>} <termo2>

[2] O conjunto de modelos de sentença aqui utilizado constitui uma variante daquele proposto por (BRG, 1997), levando-se em consideração observações complementares realizadas na literatura da área, mais especificamente em (MORGAN, 2002).

	<termo1> TEM PERMISSÃO PARA <verbo> <comp> <valor> <termo2>
	<termo1> TEM PERMISSÃO PARA <verbo> <termo2> <comp> <valor>
Proibição	<termo1> NÃO TEM PERMISSÃO PARA <verbo> {<prep>} {<artigo>} <termo2>
	<termo1> NÃO TEM PERMISSÃO PARA <verbo> <comp> <valor> <termo2>
	<termo1> NÃO TEM PERMISSÃO PARA <verbo> <termo2> <comp> <valor>
Obrigação	<termo1> DEVE OBRIGATORIAMENTE <verbo> {<prep>} {<artigo>} <termo2>
	<termo1> DEVE OBRIGATORIAMENTE <verbo> <comp> <valor> <termo2>
	<termo1> DEVE OBRIGATORIAMENTE <verbo> <termo2> <comp> <valor>
	<termo1> DEVE OBRIGATORIAMENTE SER <comp> <valor>

2.3.2 Técnicas para a Modelagem de Sistemas de Informação

Segundo (SCHMITZ; SILVEIRA, 2000), a Modelagem de Sistemas de Informação pode ser dividida em duas fases. A primeira é a definição do **Modelo de Requisitos** que tem por objetivo definir quais são as funcionalidades esperadas pelo sistema. Ela é formada pelo Modelo de Casos de Uso, Modelo de Domínio e as Maquetes do sistema. Já a segunda fase, chamada de **Modelo de Análise e Projeto**, tem como objetivo transformar o Modelo de Requisitos na definição dos módulos componentes do software. Esta fase é composta pelo Modelo Estático (classes) e Dinâmico (seqüência e estados). Cada uma dessas fases deve ser realizada de forma iterativa, repetindo os passos várias vezes até que se obtenha um modelo completo e consistente (WIRFS-BROCK; MCKEAN, 2003). A figura 2.3 ilustra estes modelos.

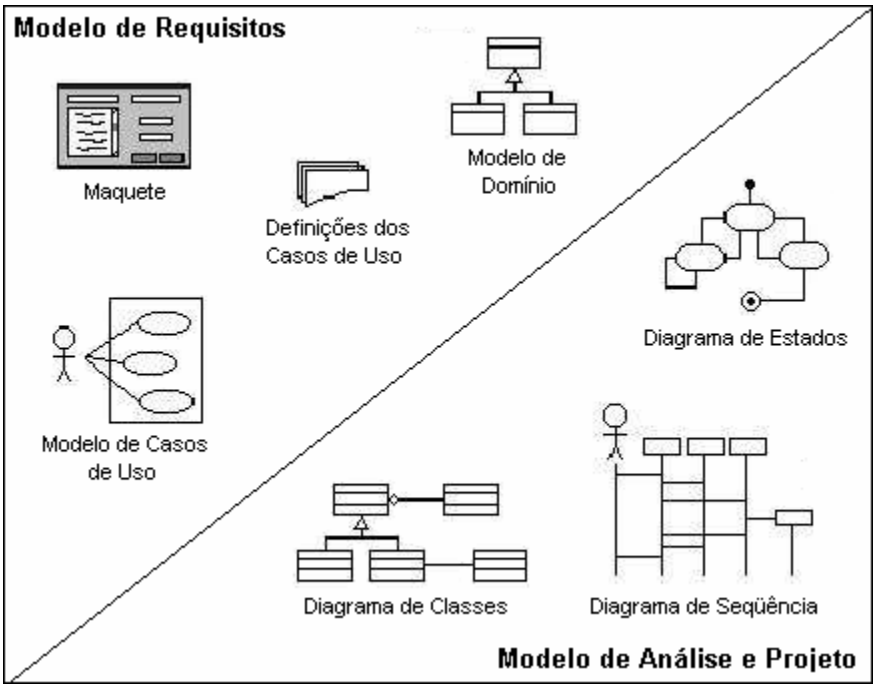


Figura 2.3 Modelos Produzidos na Fase de Requisitos e na Fase de Análise e Projeto (SCHMITZ; SILVEIRA, 2000)

Para descrição dos Modelos de Sistema de Informação será adotada a linguagem UML (OMG, 2005). A UML representa uma coletânea de práticas de análise e projeto orientados a objetos. Os diagramas da UML que serão utilizados neste trabalho são: o Diagrama de Casos de Uso, o Diagrama de Classes, o Diagrama de Seqüência e o Diagrama de Estados.

O **Diagrama de Casos de Uso** ilustra um conjunto de casos de uso, atores e suas relações. Seu objetivo é representar o contexto de um sistema em estudo, com ênfase na identificação das fronteiras (atores), e dos requisitos funcionais através do significado das suas funções. Os atores representam as fronteiras de interação do sistema em estudo. Eles podem representar papéis executados por pessoas, sistemas ou organizações, que em algum momento interagem com as funcionalidades do sistema. Um caso de uso é uma seqüência

de ações que um ou mais atores realizam num sistema de modo a obterem um resultado particular (BOOCH; RUMBAUGH; JACOBSON, 1999).

O **Diagrama de Classes** descreve as classes que formam a estrutura do sistema e as suas relações (BOOCH; RUMBAUGH; JACOBSON, 1999). Uma classe define os atributos e as operações de um conjunto de objetos. Todos os objetos desta classe (*instâncias* da classe) compartilham o mesmo comportamento e possuem os mesmos atributos. As relações entre as classes podem ser associações, composições ou heranças.

O **Diagrama de Seqüência** é um tipo de diagrama de interação que exibe o comportamento dinâmico do sistema. Este diagrama representa a interação entre objetos através da troca de mensagens entre eles para a realização de determinada tarefa ou funcionalidade especificada em um caso de uso (BOOCH; RUMBAUGH; JACOBSON, 1999).

O **Diagrama de Estados** é utilizado para representação do comportamento discreto de um sistema através de máquinas de estado finitas, que podem ser ilustradas como dispositivos que recebem um número finito de estímulos como entrada e conseguem processá-las e oferecer respostas (BOOCH; RUMBAUGH; JACOBSON, 1999). Desta forma, é possível modelar o comportamento de vários elementos através dos seus estados e os estímulos (*transições*) que participam da máquina de estados.

2.3.3 Métodos Utilizados na Transformação entre os Modelos

O ponto mais importante do processo MDA é a transformação entre os modelos. As seções a seguir descrevem os métodos utilizados para a transformação do Modelo de Negócio em Modelo de Sistema de Informação e para transformação do Modelo de Sistema de Informação em código.

2.3.3.1 Método para a Transformação da Definição dos Termos em Classes

Uma vez que os termos de um determinado negócio tenham sido definidos, é possível mapear alguns destes termos em classes de domínio (Diagrama de Classes) (MARTINS, 2006). O critério de seleção de termos a serem representados como classes foi o relacionamento de atributo. Segundo (MELLOR; SHLAER, 1990), um objeto deve possuir atributos. Desta forma, o mapeamento dos termos para uma classe de modelo conceitual considera apenas os termos que possuem atributos.

Assim, cada termo que possui relacionamento de atributo com outro termo é mapeado para uma classe. Os termos atributos são mapeados em atributos desta classe. Caso um termo atributo também possua atributos, ele será representado através de uma classe e o relacionamento de atributo será representado através de uma associação com rótulo "possui" entre os termos.

Os relacionamentos de herança, parte e associação são representados, respectivamente, por relacionamentos de generalização, agregação e associação da UML. O quadro 2.2 mostra o mapeamento utilizado para gerar as classes de domínio.

Quadro 2.2 - Mapeamento da Definição dos Termos para Classes de Domínio

Código	Descrição
MPT-1	Cada termo que possui pelo menos um relacionamento de atributo com outro termo (modelo de sentença "TEM COMO ATRIBUTO") será mapeado para uma classe.
MPT-2	Os termos atributos (modelo de sentença "TEM COMO ATRIBUTO") serão mapeados para atributos da classe, onde o domínio utilizado na especificação do relacionamento será mapeado para o tipo do atributo. Caso um termo atributo também possua atributos, ele será representado através de uma classe e o relacionamento de atributo será representado através de uma associação com rótulo "possui" entre os termos.
MPT-3	Os relacionamentos de herança (modelo de sentença "É SUBTIPO DE") serão mapeados para relacionamentos de generalização entre as classes.
MPT-4	Os relacionamentos de parte (modelo de sentença "TEM COMO PARTE") serão mapeados para relacionamentos de agregação entre as classes.
MPT-5	Os relacionamentos de associação (modelo de sentença "ESTÁ RELACIONADO A")

	serão mapeados para relacionamentos de associação entre as classes, onde o verbo e o grau utilizados na definição do relacionamento entre os termos serão mapeados, respectivamente, para o rótulo e o grau da associação entre as classes.

2.3.3.2 Método para a Transformação dos Processos em Casos de Uso

A transformação dos Processos em Casos de Uso consiste na aplicação de quatro conjuntos de heurísticas a um determinado Modelo de Processo (Diagrama de Atividades), para a obtenção do respectivo Modelo de Requisitos (Diagrama de Casos de Uso) (CRUZ, 2004). Os diagramas utilizados são modelos gráficos que possuem a vantagem da facilidade de comunicação, porém não possuem um rigor matemático. Assim, a identificação das heurísticas foi realizada através de experimentos e em conformidade com a literatura especializada de Modelagem de Processos e Requisitos, a exemplo dos seguintes trabalhos: (MARSHALL, 2000), (SANTANDER; CASTRO, 2000) e (SHNIEDER; WINTERS, 2001).

Em um processo de negócio, uma atividade é um conjunto de ações detalhadas para o cumprimento do objetivo do processo. Por outro lado, um caso de uso é um conjunto de passos a serem executados com a finalidade de atingir um objetivo computacional (do sistema). Assim, existe uma correlação entre esses elementos (atividade e caso de uso), sendo esta a base para as heurísticas descritas nesta seção (CRUZ, 2004).

Algumas das heurísticas foram definidas baseando-se na topologia, disposição e interação entre os elementos utilizados nos diagramas. Esta topologia acarreta na adoção de determinados padrões de diagramação do modelo de requisitos funcionais. Uma restrição de topologia imposta para uso deste método é a de representação de apenas uma transição de entrada e uma de saída para as atividades. Esta restrição não caracteriza uma limitação,

tendo em vista que o diagrama de atividades possui recursos para a junção e separação de transições.

A seguir, apresentamos as heurísticas desenvolvidas em (CRUZ, 2004), divididas em quatro grupos: HAG (Heurísticas para Ajuste Geral), HIA (Heurísticas para Identificação de Atores), HIC (Heurísticas para Identificação de Casos de Uso) e HIR (Heurísticas para Identificação de Relações). Cada um dos grupos possui uma finalidade própria, sendo que a seqüência apresentada deve ser respeitada para a obtenção dos resultados.

Quadro 2.3 - Heurísticas para Ajuste Geral (HAG) (CRUZ, 2004).

A finalidade deste grupo de heurísticas é direcionar a aplicação das demais heurísticas, expressando ordenamento e limitações.

Código	Descrição
HAG-1	As heurísticas serão aplicadas seguindo-se o fluxo de trabalho.
HAG-2	As heurísticas serão aplicadas para elementos contidos em cada raia de responsabilidade isoladamente. Ou seja, elementos de raias diferentes não serão analisados em conjunto para aplicação das heurísticas.

Quadro 2.4 - Heurísticas para Identificação dos Atores (HIA) (CRUZ, 2004).

A finalidade deste grupo de heurísticas é identificar atores a serem representados no Diagrama de Casos de Uso, a partir dos elementos existentes no Diagrama de Atividades.

Código	Descrição
HIA-1	Um processo, independente de sua natureza, é concebido para o atendimento de algum cliente externo ao processo. Este cliente deve ser identificado como um ator em potencial.
HIA-2	As raias de responsabilidades caracterizam postos de trabalho e/ou seus responsáveis, representando efetivamente quem executa as atividades nelas contidas. Estes executores devem ser identificados como atores em potencial.
HIA-3	Os artefatos caracterizam recursos, insumos ou produtos. Particularmente os recursos podem representar os executores de uma determinada atividade do processo. Recursos representados poderão ser identificados como responsáveis pela execução das atividades. Estes executores devem ser identificados como atores em potencial, interagindo com o caso de uso derivado da atividade.
HIA-4	Os sinais representados para envio de notificações de eventos para fora da fronteira da aplicação possuem intrinsecamente um receptor. Este receptor deve ser identificado como um ator em potencial, interagindo com o caso de uso derivado da atividade que enviou o sinal. Analogamente, os sinais representados para recepção de eventos externos possuem um emissor que também será um ator em potencial.

HIA-5	Atores identificados que apresentam interação irrelevante dentro do contexto em análise deverão ser desprezados pelo projetista.
HIA-6	Atores identificados com denominações diferentes e com significados semânticos iguais deverão ser unificados pelo projetista.

Quadro 2.5 - Heurísticas para Identificação dos Casos de Uso (HIC) (CRUZ, 2004).

A finalidade deste grupo de heurísticas é identificar as funcionalidades a serem representadas no Diagrama de Casos de Uso, a partir dos elementos existentes no Diagrama de Atividades.

Código	Descrição
HIC-1	Atividades de execução "passiva", sem realização de trabalho e representadas na forma de estado, serão retiradas do diagrama sem a interrupção do fluxo de trabalho. Os elementos relacionados ao estado deverão ser transferidos para a próxima atividade de execução "ativa".
HIC-2	Atividades devem ser representadas na forma de casos de uso distintos, na relação de uma atividade para um caso de uso.
HIC-3	Objetos interagindo com atividades darão origem a novos casos de uso que permitam a consulta de suas informações.
HIC-4	Objetos não interagindo com atividades para atualização de informações darão origem a novos casos de uso que permitam a manutenção de suas informações.

Quadro 2.6 - Heurísticas para Identificação de Relações (HIR) (CRUZ, 2004).

A finalidade deste grupo de heurísticas é identificar as relações a serem representadas no Diagrama de Casos de Uso, a partir dos elementos existentes no Diagrama de Atividades. Estas relações ocorrem tanto entre os atores e casos de uso, como também entre os casos de uso.

Código	Descrição
HIR-1	Os casos de uso originados de atividades seqüenciais em uma mesma raia de responsabilidade serão relacionados na forma "<<include>>" com um novo caso de uso que agrupe suas funcionalidades.
HIR-2	Os casos de uso originados de atividades concorrentes em uma mesma raia de responsabilidade, representadas a partir de um ponto de separação e sincronizadas em um outro ponto de junção dando continuidade ao fluxo de trabalho ou finalizando o processo, serão representados isoladamente.
HIR-3	Os casos de uso originados de atividades alternativas em uma mesma raia de responsabilidade, representadas a partir de um ponto de desvio e unificadas em um outro ponto de fusão dando continuidade ao fluxo de trabalho, serão relacionados na forma "<<extend>>" com um novo caso de uso que agrupe suas funcionalidades.
HIR-4	Os casos de uso originados de atividades alternativas em uma mesma raia de responsabilidade representadas a partir de um ponto de desvio e finalizando o processo serão relacionados na forma "<<extend>>" com um novo caso de uso que agrupe suas funcionalidades. Os casos de uso originados da atividade alternativa, cujo caminho dá continuidade ao fluxo de trabalho, deverão ser representados isoladamente.
HIR-5	Os casos de uso criados pelas heurísticas HIR-1, HIR-3 ou HIR-4 para agrupamento de funcionalidades, que se relacionarem apenas a um caso de uso, deverão ser desprezados.

HIR-6	Os atores identificados pela heurística HIA-1 serão relacionados com todos os casos de uso principais, exceto os identificados nas heurísticas HIR-3 e HIC-4, através de relações do tipo interação.
HIR-7	Os atores identificados pela heurística HIA-2 serão relacionados através de relações do tipo interação com todos os casos de uso principais originados de atividades da raia de responsabilidade em referência.
HIR-8	Os atores identificados pela heurística HIA-3 serão relacionados através de relações do tipo interação com todos os casos de uso originados das atividades, que se relacionam os objetos em referência.
HIR-9	Os atores identificados pela heurística HIA-4 serão relacionados, através de relações do tipo interação, com todos os casos de uso originados das atividades, que se relacionam os sinais em referência.
HIR-10	Quando da existência de relação entre casos de uso do tipo “<<include>>” ou “<<extend>>”, os atores identificados pela heurística HIA-2 terão sua relação do tipo interação com o caso de uso principal.

2.3.3.3 Método para a Transformação do Modelo de Sistema em Código

A idéia central deste método consiste na transformação de um modelo abstrato formal de alto nível em um programa de computador. Em outras palavras, o método trata da tradução do Modelo de Sistema de Informação, representado através da linguagem de modelagem UML, em código-fonte do programa, representado através de uma linguagem de programação padrão (*Pascal*, *C*, *C++*, *Java*, etc.).

Para realizar esta transformação é necessário acrescentar um valor semântico aos modelos da UML (BECK, 2006). Em (PINTO, 2004) é proposto um mapeamento para os diagramas de Casos de Uso, Classes, Estados e Seqüência (quadro 2.7). Neste mapeamento, cada elemento destes diagramas é transformado em uma parte diferente do código, seguindo o padrão arquitetural Modelo - Visão - Controlador (*Model - View – Controller*, MVC), descrita em (BURBECK, 1987).

A arquitetura MVC (GAMMA *et al*, 1995) determina a separação de responsabilidades de uma aplicação em três componentes: o **Modelo** (*Model*) que é formado por entidades de negócio que representam os dados do sistema; a **Visão** (*View*)

que tem o objetivo de apresentar as informações pertencentes ao Modelo e capturar eventos disparados pelos usuário; e o **Controlador** (*Controller*) que realiza o tratamento dos eventos capturados, representando o elo entre os componentes de Visão e os componentes do Modelo (PAIS, 2004).

Quadro 2.7 - Mapeamento do Modelo de Sistema para Código

Código	Descrição
MPC-1	Os casos de uso que possuem pelo menos uma interação com um ator definirão os itens do menu principal do programa.
MPC-2	Cada classe do domínio do problema representa um conjunto de elementos. Assim, uma classe modelada será mapeada para uma tabela do banco de dados. Os atributos da classe serão mapeados em campos da tabela.
MPC-3	Um objeto do domínio do problema será implementado por três classes: *GerenteObjeto*, *TabObjeto* e *UmObjeto*. A classe *GerenteObjeto* atua como gerente do conjunto de objetos, atendendo a serviços relativos ao conjunto de elementos. A classe *TabObjeto* representa a estrutura que faz o acesso aos dados em memória permanente. A classe *UmObjeto* serve para representar um elemento do conjunto.
MPC-4	As classes de interface modeladas serão mapeadas para formulários que permitem a interação dos usuários com o sistema. Cada atributo da classe de interface está relacionado a um campo do formulário.
MPC-5	As classes de controlador modeladas serão mapeadas para classes que apresentam basicamente dois serviços: o que inicia o caso de uso e o que trata as sinalizações provindas da interface. Esses serviços são gerados independentes do modelo.
MPC-6	Os Diagramas de Seqüência estabelecerão a dependência entre as classes, o nome das instâncias das classes e a seqüência das mensagens que são trocadas entre estas instâncias no início do caso de uso.
MPC-7	Os Diagramas de Estados serão mapeados para uma máquina de estados que é embutida dentro do código da classe que está associada ao diagrama de estado modelado.

2.4 Ferramentas Automatizadas de Apoio à Engenharia de Sistemas

As técnicas e métodos apresentados nas seções anteriores necessitam de ferramentas automatizadas para que sua aplicação se torne viável. A utilização do computador para apoiar as atividades ligadas a Engenharia de Sistemas não é um conceito novo. No início da década de 70, a expressão Projeto Apoiado por Computador (*Computer-Aided Design*, CAD) era comum entre engenheiros de projeto. Nos anos 80, surgiram as expressões

Análise de Sistemas Apoiada por Computador (*Computer-Aided Systems Analysis*, CASA) e Programação Apoiada por Computador (*Computer-Aided Programming*, CAP). Atualmente, as duas expressões foram combinadas na expressão **Engenharia de Software Apoiada por Computador** (*Computer-Aided Software Engineering*, CASE).

Segundo (PRESSMAN, 2001), ferramentas CASE fornecem o apoio automatizado ou semi-automatizado para o processo e para os métodos de engenharia de sistemas. Elas automatizam as atividades, gerenciam os produtos de trabalho produzidos ao longo do projeto e apóiam os engenheiros na fase de análise, projeto, codificação e teste.

Nesta concepção, CASE é uma abordagem para o ciclo de vida de sistemas que é baseada na automação. Para realizar essa automação, muitas ferramentas CASE incluem editores gráficos que implementam uma interface com o usuário transformando o desenvolvimento de software num processo visual. Toda a informação do sistema é armazenada num único repositório de dados que pode ser automaticamente gerenciado e distribuído.

Apesar das ferramentas CASE individuais proporcionarem uma grande economia de trabalho, os maiores benefícios do CASE só podem ser alcançados através da integração (FORTE, 1990). Um CASE Integrado, ou simplesmente I-CASE, é um ambiente que cobre todo o processo de desenvolvimento de um sistema. Esse tipo de ambiente se caracteriza pela combinação de várias ferramentas diferentes, pela existência de uma base de dados que contenha informações consistentes de engenharia de sistemas e pela transferência constante de informação de uma ferramenta para outra (CHEN; NORMAN, 1992).

Entre os benefícios do CASE integrado, (FORTE, 1990) destacam-se:

- a transferência harmoniosa de informações (modelos, programas, documentos, dados) de uma ferramenta para outra e de uma etapa de engenharia de sistemas para a seguinte;
- uma redução do esforço exigido para realizar atividades como: gerenciamento de configuração de software, garantia de qualidade e produção de documentação;
- aumento no controle do projeto, que é obtido por meio de um melhor planejamento, monitoração e comunicação;
- melhor coordenação entre os membros da equipe de um projeto de software.

2.5 Conclusão

Este capítulo apresentou os conceitos que constituem os fundamentos sobre os processos, métodos e ferramentas da área de Engenharia de Sistemas. Um mapa contendo esses conceitos e os seus relacionamentos pode ser encontrado no APÊNDICE A deste texto. No próximo capítulo, apresentaremos um processo de desenvolvimento de sistemas que faz uso desses conceitos.

3. O PROCESSO RAPDIS

"O trabalho de software segue com freqüência a primeira lei do ciclismo: não importa onde você vá, é sempre subida e contra o vento."

Autor desconhecido

3.1 Introdução

Estabelecer um Processo de Engenharia de Sistemas significa definir uma estrutura comum, aplicável a todos os projetos de sistema, independente do seu tamanho ou complexidade. O objetivo deste capítulo é apresentar o processo RAPDIS (*Rules And Process for the Development of Information Systems*) mostrando seus principais produtos e atividades.

3.2 Visão Geral do Processo

O documento do OMG que explica a MDA, encontrado em (MUKERJI; MILLER, 2003), não explicita como construir os modelos CIM, PIM e PSM; tampouco define como transformar um modelo de alto nível de abstração em outro de nível mais baixo. Assim, para construirmos o processo RAPDIS nos baseamos não apenas na definição da MDA, como também nas técnicas de modelagem e nos métodos de transformação desenvolvidos pelo grupo de pesquisa GETI. O processo RAPDIS é, portanto, uma forma de integração desses trabalhos. O diagrama principal do processo pode ser visto na figura 3.1.

O processo RAPDIS se inicia com o término da negociação do projeto. Resumidamente, o processo consiste na execução seqüencial das seguintes atividades: "construir modelo de negócio", "construir modelo de sistema de informação" e "implementar sistema". Com o fim do processo RAPDIS, tem início a fase de homologação do sistema.

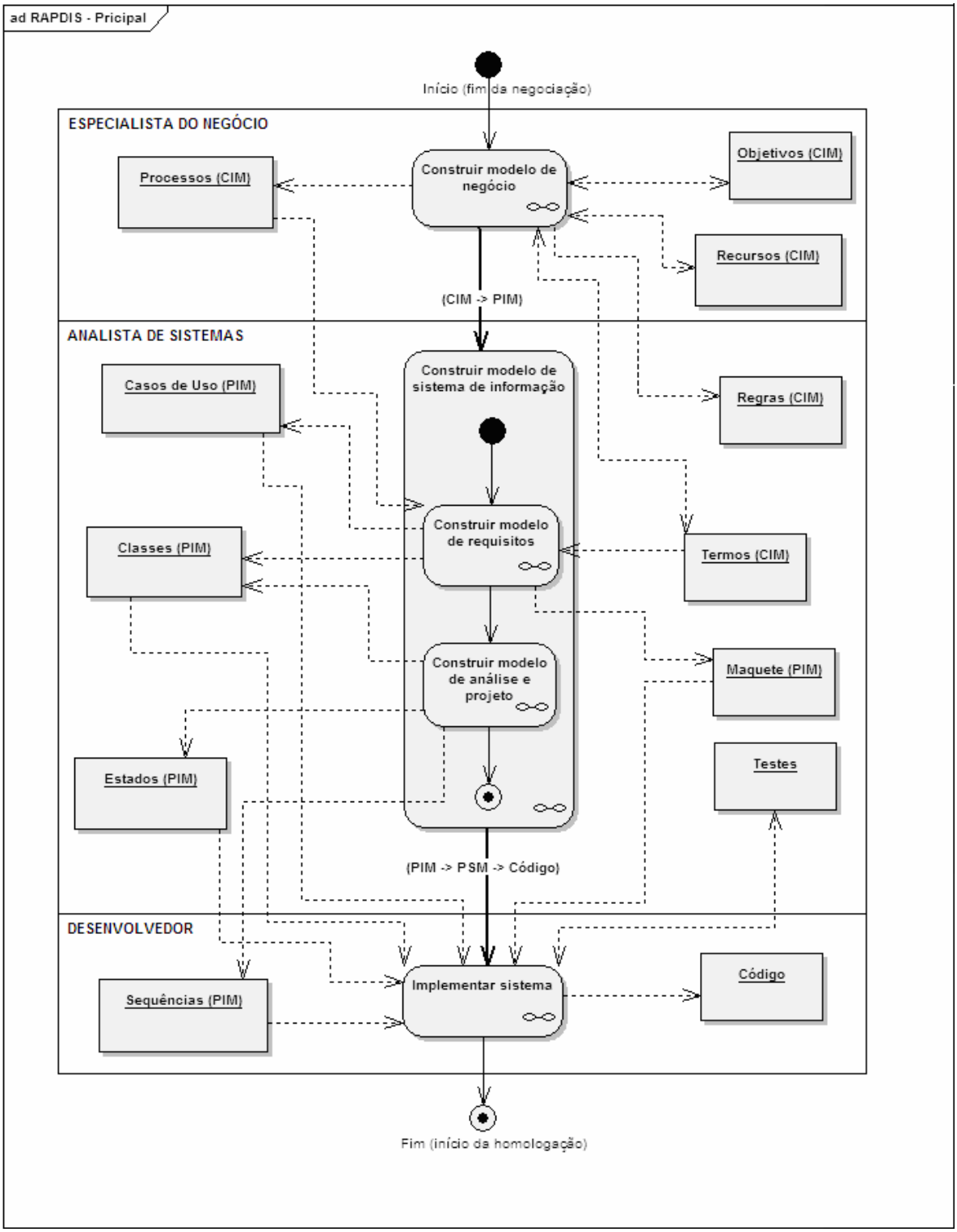


Figura 3.1 - Diagrama principal do processo RAPDIS

Participam deste processo os 3 atores, com diferentes responsabilidades e habilidades. O **Especialista do Negócio** detém o conhecimento sobre o funcionamento do negócio, atuando na identificação das necessidades de informatização; o **Analista de Sistemas** é responsável por construir um modelo de sistema que atenda às necessidades identificadas pelo Especialista do Negócio; e o **Desenvolvedor** implementa o sistema projetado pelo analista, executando atividades de codificação e teste.

Na figura 3.1 utilizamos as etiquetas “CIM”, “PIM” e “PSM” para marcar onde os modelos da MDA estão presentes no processo RAPDIS. Além disso, podemos observar que, durante todo o processo, os documentos/modelos produzidos em uma determinada atividade são utilizados como insumos da atividade seguinte. A atividade seguinte, por sua vez, produz estes insumos novos documentos/modelos. Este fato caracteriza a existência das transformações dos modelos dentro das atividades do processo RAPDIS. Nas seções a seguir detalhamos as principais atividades deste processo.

3.3 Atividade “Construir Modelo de Negócio”

Quem projeta um Sistema de Informação deve analisar o seu funcionamento tanto pela perspectiva do negócio como pela perspectiva técnica. Precisa entender a sua forma de operar, as suas regras e as suas necessidades. Por este motivo, o primeiro modelo a ser especificado dentro do processo RAPDIS é o modelo de negócio através da atividade “construir modelo de negócio”.

No RAPDIS, o modelo de negócio é o resultado da definição dos Objetivos, Recursos, Termos, Regras de Negócio e Processos de Negócio. Os objetivos são os resultados que a organização deseja atingir. Os recursos são objetos utilizados ou produzidos no negócio. Os termos constituem os elementos básicos da linguagem utilizada

no modelo de negócio. As regras são declarações que restringem, derivam e fornecem condições de existência no modelo de negócio. Os Processos constituem um conjunto de atividades estruturadas e medidas para que um produto seja gerado.

A modelagem desses objetos do negócio segue a técnica apresentada em 2.3.1. Como esses objetos estão fortemente relacionados entre si, o processo RAPDIS propõe que as definições sejam construídas em paralelo, como pode ser visto na figura 3.2.

Após a definição desses objetos, é realizada uma revisão por pares (*peer review*). A revisão por pares é uma técnica em que um produto do trabalho é examinado por um grupo de pessoas, com o objetivo de avaliar o conteúdo técnico e qualidade deste produto (IEEE, 1998). A atividade de revisão é aplicada em diversos pontos do processo RAPDIS e funciona como um “filtro” dentro do processo. Os principais produtos do processo devem ser revisados para que os erros possam ser detectados e corrigidos o mais cedo possível durante o desenvolvimento.

Os modelos produzidos nesta atividade são fundamentais para o entendimento do sistema que se pretende desenvolver e correspondem aos insumos da atividade “construir modelo de sistema de informação”. Eles representam o modelo CIM da MDA e servem de base para a obtenção do modelo de requisitos que faz parte do PIM da MDA.

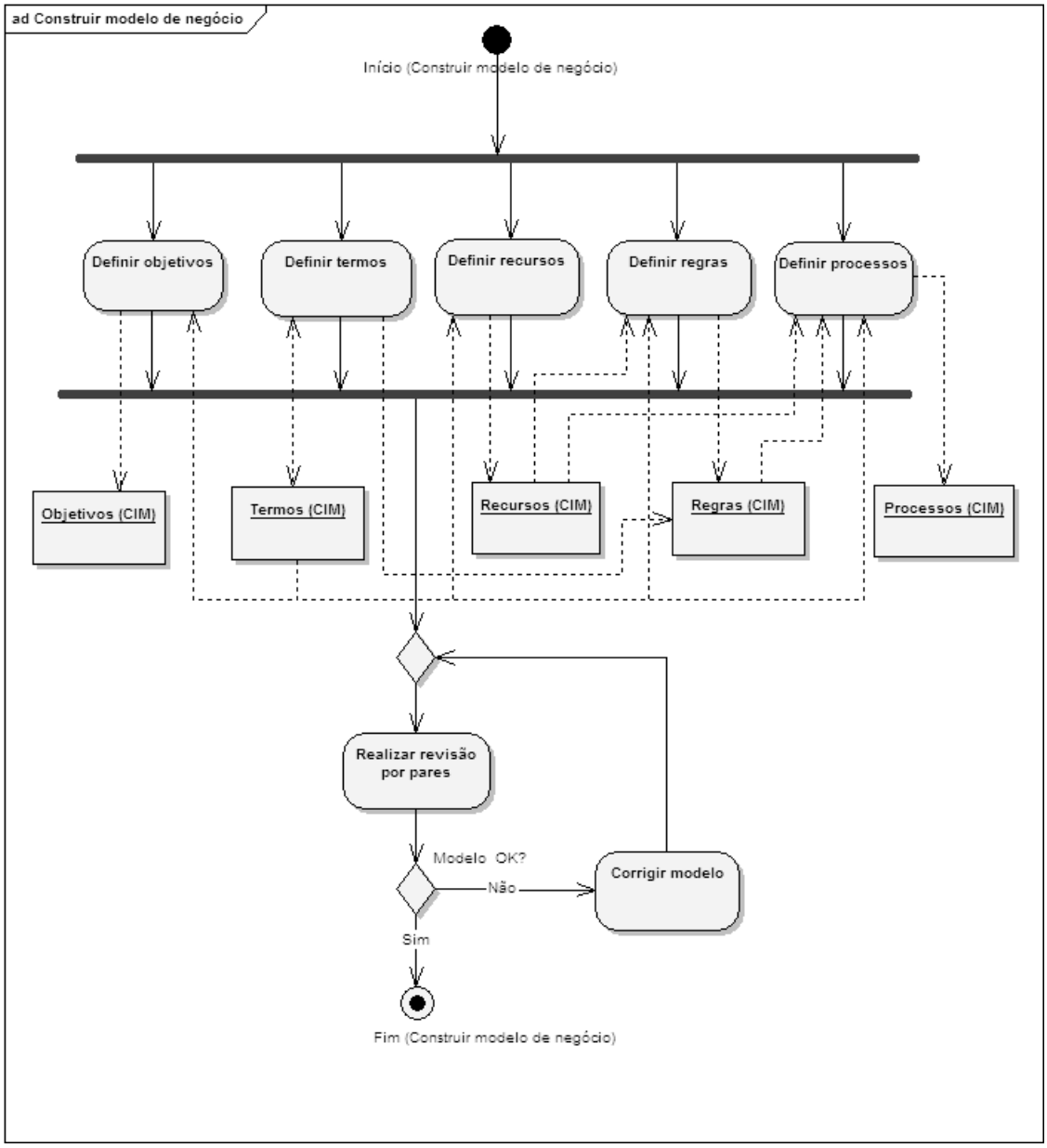


Figura 3.2 - Diagrama da Atividade “Construir Modelo de Negócio”

3.4 Atividade “Construir Modelo Sistema de Informação”

Após a construção do modelo de negócio, inicia-se a atividade “construir modelo de sistema de informação”, que corresponde à construção do PIM da MDA. É nesta atividade que os componentes do sistema serão especificados segundo a técnica apresentada na seção 2.3.2. Ela é composta por duas sub-atividades que devem ser executadas em seqüência: “construir modelo de requisitos” e “construir modelo de análise e projeto”. Estas sub-atividades serão descritas nas seções 3.4.1 e 3.4.2 a seguir.

3.4.1 Sub-atividade “Construir Modelo de Requisitos”

O objetivo desta sub-atividade é definir a funcionalidade esperada pelo usuário do sistema. Num projeto de construção civil, esta atividade corresponderia ao projeto de arquitetura, onde se descrevem as características da casa que são visíveis ao seu morador. O produto desta fase é chamado de Modelo de Requisitos (SCHMITZ; SILVEIRA, 2000). O diagrama desta sub-atividade pode ser visto na figura 3.3.

Para construir este modelo, as funcionalidades do sistema são definidas através dos seus casos de uso, classes de domínio e maquetes. Os casos de uso mostram as diversas situações de utilização do sistema, identificando também os seus verdadeiros usuários, que serão chamados de atores. As classes de domínio representam o vocabulário empregado pelos usuários do sistema. A maquete mostra ao usuário do sistema a forma como os formulários serão implementados e como este usuário poderá interagir com o sistema.

O primeiro passo na construção deste modelo é obter uma versão preliminar das classes de domínio e dos casos de uso a partir do modelo de negócio. Estes modelos são obtidos aplicando os métodos de transformação apresentados nas seções 2.3.3.1 (Transformação da Definição dos Termos do Negócio em Classes de Domínio) e 2.3.3.2

(Transformação do Modelo de Processos em Casos de Uso). Após obter esta versão preliminar, deve-se executar em paralelo os seguintes passos: refinar/completar os casos de uso, refinar as classes de domínio (modelo de domínio) e construir a maquete (projeto de interface).

Esses passos devem ser executados em paralelo, pois, à medida que os casos de uso são detalhados, o modelo de domínio é melhorado e novos componentes no projeto da interface são descobertos. Quando a execução desses passos se conclui, o modelo de requisitos passa por uma revisão por pares, as correções necessárias ao modelo são realizadas e os testes de aceitação são definidos.

Vale notar que nesta parte do processo RAPDIS ocorre a primeira transformação da MDA, a transformação do CIM para o PIM. Essa transformação consiste em dois passos: a transformação da definição dos termos do negócio (CIM) em um diagrama de classes de domínio (PIM) e a transformação dos processos de negócio (CIM) em um diagrama de casos de uso (PIM). A transformação facilita o trabalho de construção do modelo de requisitos, porém os modelos gerados precisam ainda ser refinados pelo analista de sistemas.

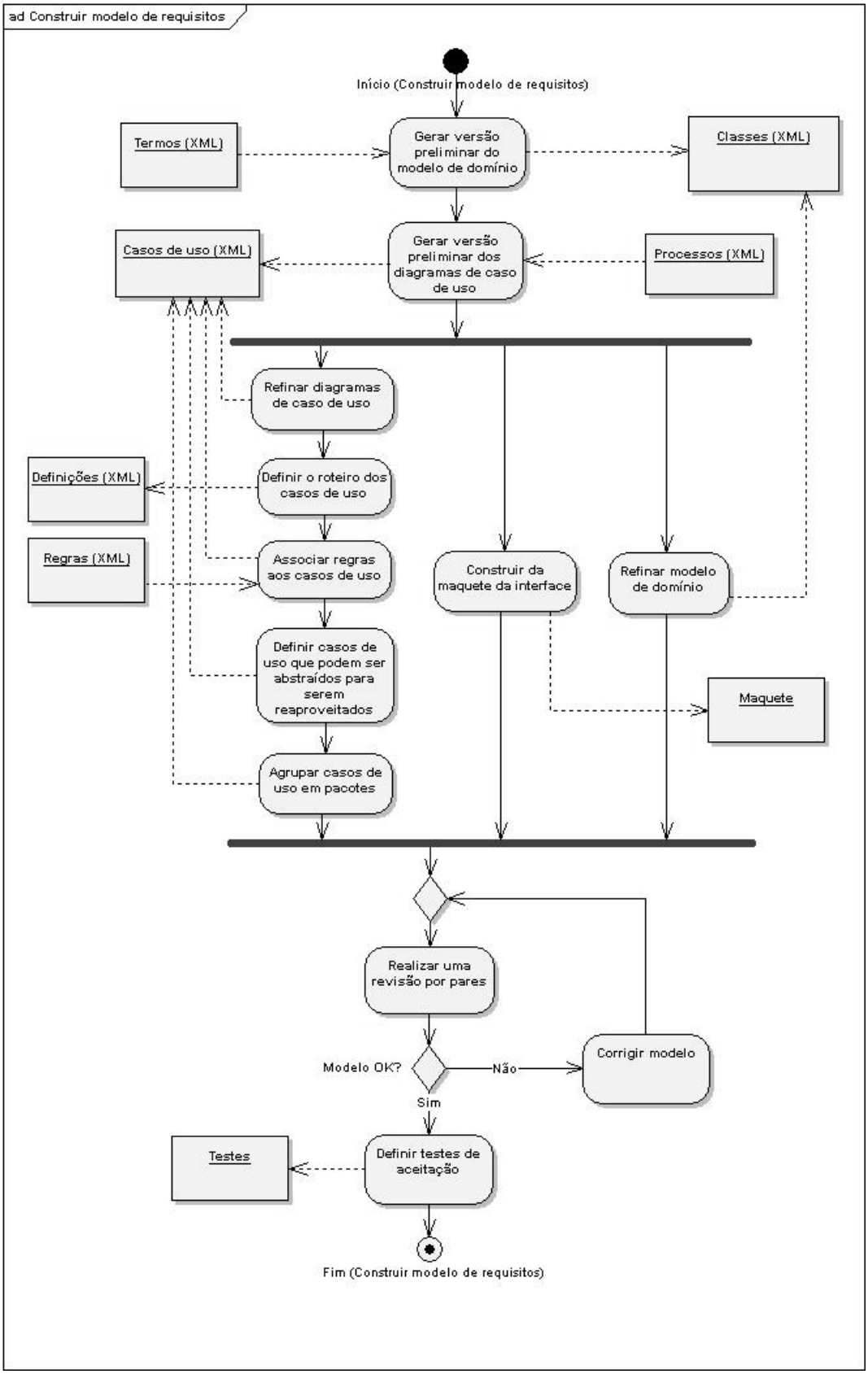


Figura 3.3 - Diagrama “Construir Modelo de Requisitos”

3.4.2 Sub-atividade “Construir Modelo de Análise e Projeto”

O objetivo desta sub-atividade é produzir um plano que permita a construção do sistema. Num projeto de construção civil, esta fase corresponderia às plantas detalhadas da construção da casa. Em termos de software, a planta do sistema é a especificação de suas classes componentes. Esta "planta" é chamada de Modelo de Análise e Projeto (SCHMITZ; SILVEIRA, 2000). O diagrama desta sub-atividade pode ser visto na figura 3.4.

Para construir este modelo, são definidos as demais classes do sistema, os diagramas de estado e os diagramas de seqüência. O modelo de classes é elaborado identificando e definindo os atributos das classes, as associações entre instâncias e as relações de herança entre as classes. O diagrama de estados mostra a seqüência de estados de um objeto ou uma interação durante sua vida em resposta a um estímulo recebido, junto com suas respostas e ações. O diagrama de seqüência permite que o projetista mostre a troca de serviços entre os objetos participantes de cada pacote do sistema.

Primeiramente, para cada pacote de casos de uso definido no Modelo de Requisitos, deve ser definido um diagrama de classes. Esse diagrama de classes deve mostrar os relacionamentos entre as classes de domínio, interface e controle, conforme o modelo MVC apresentado na seção 2.3.3.3. Depois disto, para cada pacote de casos de uso, deve ser definido um diagrama de seqüência. A criação dos diagramas de seqüência é um passo fundamental para a definição da implementação dos casos de uso do sistema. Ao definir os serviços dos objetos participantes do caso, o projetista vai refinando as definições das classes do sistema, previamente, definidos no Modelo de Domínio. Antes de prosseguir, os diagramas de classes e os diagramas de seqüência devem passar por uma revisão por pares e devem ser efetuadas as correções necessárias.

O passo seguinte dentro desta sub-atividade é criar um diagrama de transição de estados para as classes de domínio, interface e controle que tenham ciclos de vida não-triviais. A determinação dos diagramas de estado pode levar à descoberta de novas relações de herança entre as classes. Ao terminar este passo, o Modelo de Análise e Projeto deve ser novamente revisado para eliminar os eventuais erros de consistência.

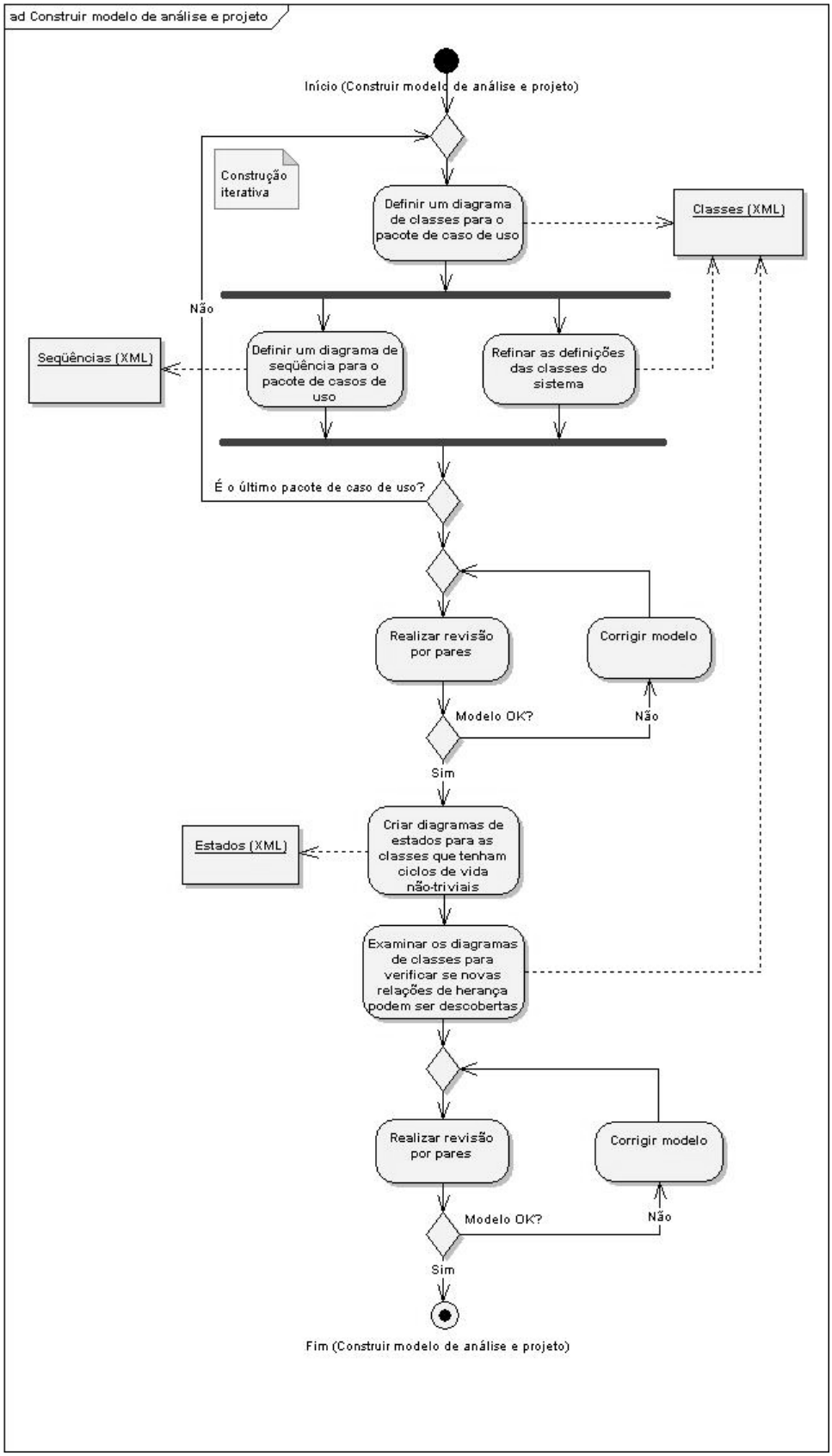


Figura 3.4 - Diagrama da Sub-atividade “Construir Modelo de Análise e Projeto”

3.5 Atividade “Implementar Sistema”

Nesta atividade, o Modelo de Sistema de Informação construído deve ser implementado. Os modelos desenhados devem ser transformados num conjunto de sentenças escritas em uma linguagem de programação para uma determinada plataforma. Depois de compilado, este código transforma-se em software, ou seja, em um programa executável.

Para atingir este objetivo, é preciso, primeiramente, especificar a plataforma alvo (*Delphi*, *Java*, *C#*, etc.), especificar alguns parâmetros necessários para geração de código (tabelas do banco, casos de manutenção, recuperação de modelos antigos, etc.), transformar o modelo em código através do método apresentado em 2.3.3.3 e especificar os testes de integração.

O código gerado, entretanto, é apenas um esqueleto da arquitetura que será utilizada e uma parte do código deve ser completada manualmente pela equipe de desenvolvimento. Assim, para o esqueleto de cada pacote de caso de uso gerado devem ser definidos os testes de unidade, o código deve ser completado e revisado, os testes de unidade devem ser executados e, se necessário, o código deve ser corrigido.

Quando todos os pacotes tiverem sido implementados, os pacotes devem ser integrados, uma nova revisão do código deve ser realizada, os testes de integração e as correções devem ser executadas. Em seguida, os testes de aceitação definidos durante a construção do Modelo de Requisitos devem ser executados. As devidas correções devem ser realizadas até que o resultado dos testes seja considerado aceitável, o que finaliza a execução do processo RAPDIS (figura 3.5 e figura 3.6).

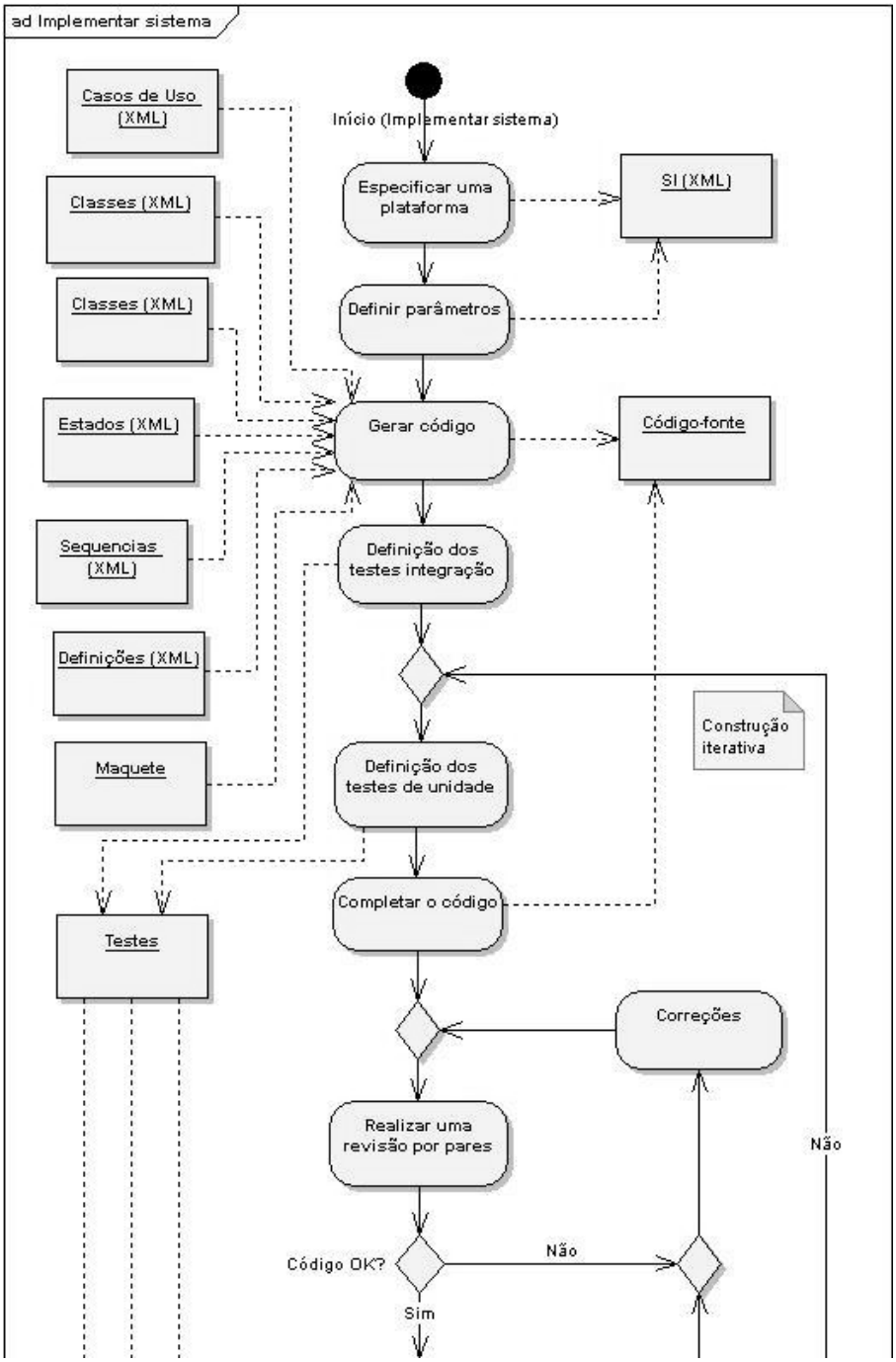


Figura 3.5 - Diagrama “Implementar Sistema” (1/2)

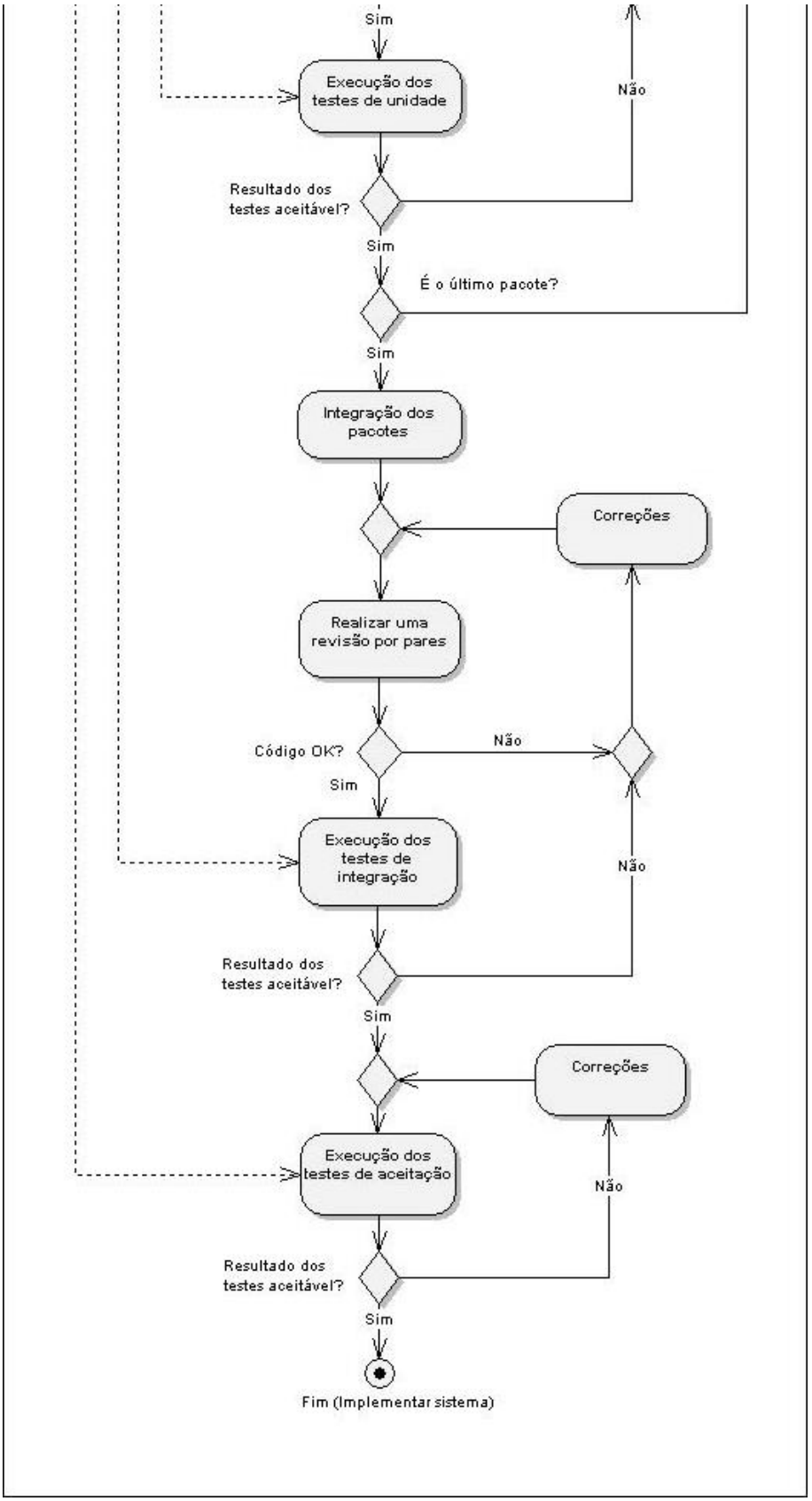


Figura 3.6 - Diagrama “Implementar Sistema” (2/2)

A geração de código no processo RAPDIS representa a transformação do PIM para o código na MDA. Esta é a última transformação dentro do processo e tem como objetivo facilitar o trabalho dos desenvolvedores, eliminado alguns passos repetitivos da fase de codificação.

Entre o PIM e o código existe um outro modelo, o PSM, que apresenta os detalhes da plataforma escolhida. A função deste modelo intermediário é possibilitar a otimização e customização, antes da geração do código (CZARNECKI; HELSEN, 2004). No RAPDIS, adotamos uma simplificação da abordagem da MDA, ocultando o PSM. Posteriormente, esta transformação poderá ser decomposta em duas partes: a transformação do PIM em PSM e a transformação do PSM em código.

3.6 Conclusão

Este capítulo apresentou as atividades, os insumos e os produtos do processo RAPDIS. Um exemplo das transformações deste processo é apresentado no APÊNDICE B deste texto. Uma abordagem mais efetiva exige que este processo seja acompanhado um ferramental que automatize suas principais atividades. O capítulo seguinte mostra o ambiente RAPDIS que é um I-CASE criado para apoiar o processo apresentado.

4. O AMBIENTE RAPDIS

"Os milagres mais comuns da engenharia de software são as transições da análise para o projeto e do projeto para o código."

Richard Dué

4.1 Introdução

Para apoiar o processo RAPDIS, foi criado um ambiente I-CASE que integra ferramentas de modelagem de negócios e modelagem de sistemas de informação. Este tipo de ambiente necessita de representações consistentes da informação, de interfaces padronizadas e de um mecanismo de comunicação eficiente entre o usuário e a ferramenta. Neste capítulo, apresentamos o ambiente RAPDIS em detalhes.

4.2 Requisitos do Ambiente

A partir da definição do processo RAPDIS, foi possível realizar o levantamento dos requisitos do ambiente RAPDIS. O quadro 4.1 a seguir apresenta a listagem dos macro-requisitos deste ambiente organizada por módulo.

Quadro 4.1 - Macro-Requisitos do Ambiente RAPDIS

Código	Descrição
Módulo Modelo de Negócio	
MRQ-1	O ambiente deve permitir a descrição textual do negócio.
MRQ-2	O ambiente deve permitir o cadastramento dos termos do negócio.
MRQ-2.1	O ambiente deve permitir o cadastramento da definição dos termos do negócio (regras do tipo fato).
MRQ-2.2	O ambiente deve gerar o modelo de classes de domínio do sistema de informação a partir da definição dos termos do negócio.
MRQ-3	O ambiente deve permitir o cadastramento dos objetivos do negócio.
MRQ-3.1	O ambiente deve permitir o cadastramento das metas relacionadas a um determinado objetivo do negócio.
MRQ-4	O ambiente deve permitir o cadastramento dos recursos do negócio.

MRQ-4.1	O ambiente deve permitir a definição da estrutura do negócio.
MRQ-5	O ambiente deve permitir o cadastramento das regras do negócio.
MRQ-5.1	O ambiente deve permitir o cadastramento das informações administrativas sobre as regras do negócio.
MRQ-5.2	O ambiente deve permitir a consulta às regras do negócio.
MRQ-5.3	O ambiente deve permitir a exportação das regras do negócio.
MRQ-6	O ambiente deve permitir a construção dos diagramas de atividades para a modelagem dos processos do negócio.
MRQ-6.1	O ambiente deve permitir a definição dos elementos do diagrama de atividades.
MRQ-6.2	O ambiente deve gerar o diagrama de casos de uso a partir do diagrama de atividades.
Módulo Modelo de Sistema de Informação	
MRQ-7	O ambiente deve permitir a descrição textual do sistema.
MRQ-8	O ambiente deve permitir a construção dos diagramas de casos de uso do sistema.
MRQ-8.1	O ambiente deve permitir a definição dos elementos do diagrama de casos de uso.
MRQ-9	O ambiente deve permitir a construção dos diagramas de classes do sistema.
MRQ-9.1	O ambiente deve permitir a definição dos elementos do diagrama de classes.
MRQ-10	O ambiente deve permitir a construção dos diagramas de estados do sistema.
MRQ-10.1	O ambiente deve permitir a definição dos elementos do diagrama de estados.
MRQ-11	O ambiente deve permitir a construção dos diagramas de seqüência do sistema.
MRQ-11.1	O ambiente deve permitir a definição dos elementos do diagrama de seqüência.
MRQ-12	O ambiente deve gerar o código do esqueleto da arquitetura do sistema.
Módulo Gerador de Relatório	
MRQ-13	O ambiente deve gerar um relatório com todos os modelos construídos no RAPDIS.

4.3 Interface do Ambiente

Uma vez definidas as principais funcionalidades do ambiente, é preciso especificar a sua interface. A interface é o meio de interação do usuário com o ambiente e deve satisfazer as suas necessidades de usabilidade, aplicabilidade e comunicação. O diagrama a seguir (figura 4.1) mostra as possíveis formas de interação do usuário com as funcionalidades do RAPDIS através das suas principais telas.

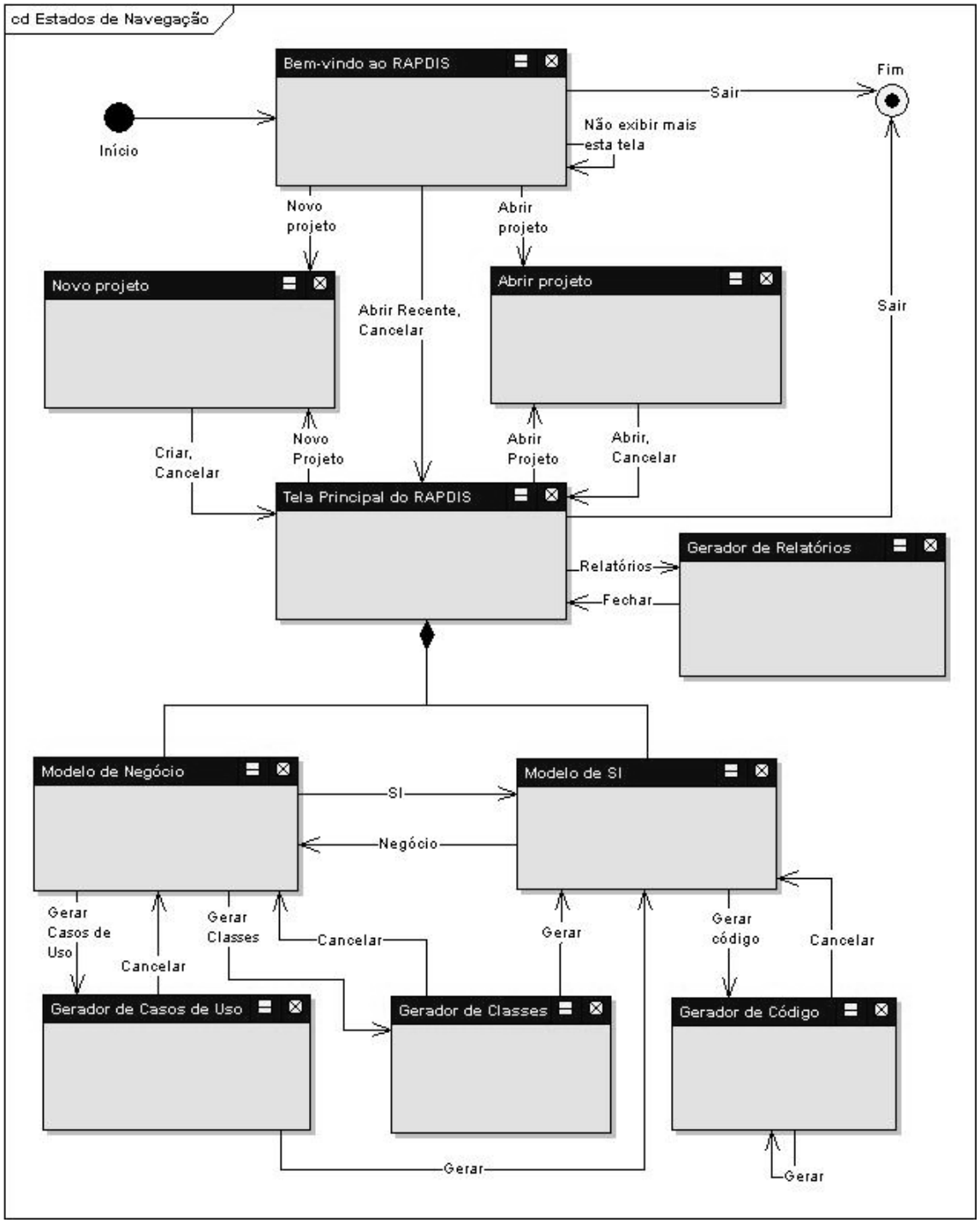


Figura 4.1 - Diagrama de Estados de Navegação do RAPDIS

Na parte de construção de regras o estilo de interface adotado foi o “preenchimento de formulário”. Já os desenhadores dos diagramas da UML possuem o estilo “manipulação

direta". Os diferentes estilos foram adotados para que o fator facilidade de uso tivesse prioridade máxima neste ambiente. A figura 4.2 ilustra a tela principal do RAPDIS.

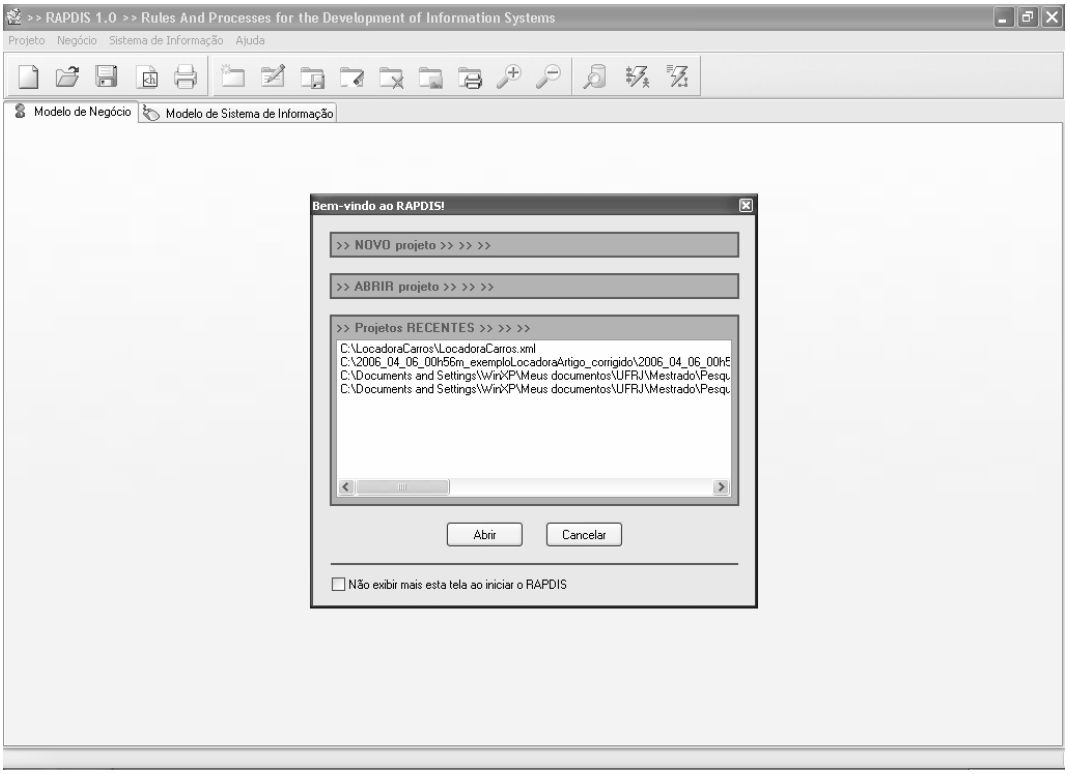


Figura 4.2 - Tela Principal do RAPDIS

4.4 Inventário das Ferramentas do Grupo GETI

Alguns dos requisitos levantados já foram implementados em ferramentas desenvolvidas pelo grupo GETI. Com o intuito de identificar os módulos e funções que poderiam ser reaproveitados, foi realizado um inventário destas ferramentas.

4.4.1 Ferramenta *Régula*

A ferramenta *Régula* visa auxiliar o gerenciamento das Regras de Negócio (MORGADO, 2005; MARTINS, 2005). As principais funcionalidades do *Régula* são: captura da descrição, missão e visão do negócio, captura dos objetivos e metas do negócio; captura dos recursos e da estrutura do negócio; definição dos termos do negócio; transformação da definição dos termos do negócio em modelo de domínio; captura de regras de negócio através de modelos de sentença em Português Estruturado; consulta a permissões, proibições e obrigações; e tradução das regras para o PROLOG (*PROgrammation en LOGique*, PROLOG) (BRATKO, 1990).

Assim, esta ferramenta atende aos macro-requisitos MRQ-1, MRQ-2, MRQ-2.1, MRQ-2.2, MRQ-3, MRQ-3.1, MRQ-4, MRQ-4.1, MRQ-5, MRQ-5.1, MRQ-5.2 e MRQ-5.3. O *Régula* foi desenvolvido em ***Delphi*** e utiliza um banco ***Paradox*** para o armazenamento de dados. Sua interface é apresentada na figura 4.1.

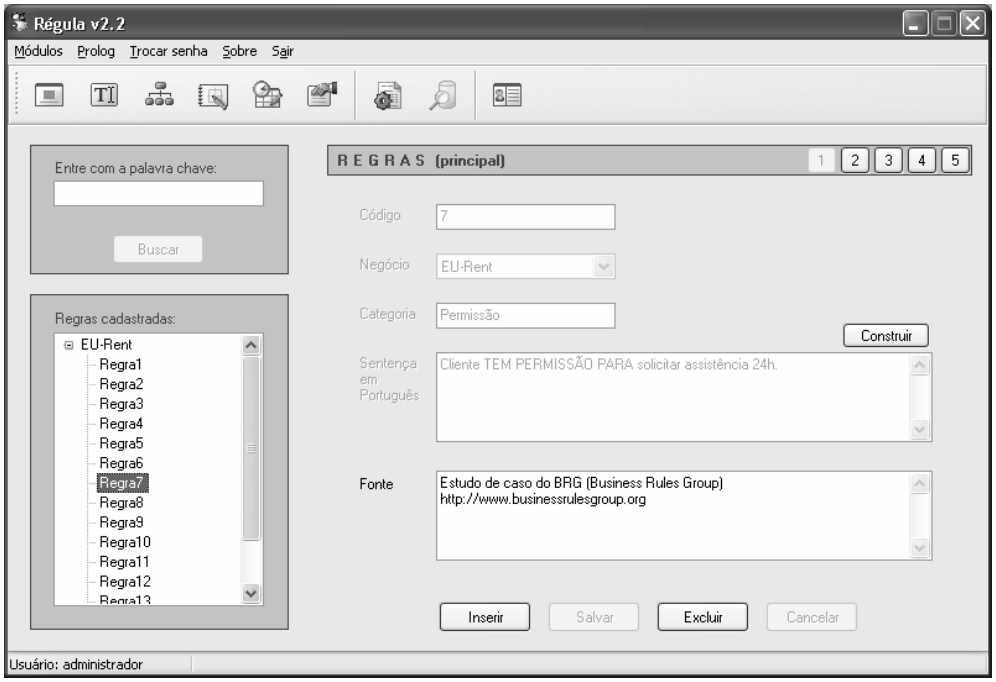


Figura 4.3 - Interface da Ferramenta *Régula*

4.4.2 Ferramenta *FastCase*

O *FastCase* é uma ferramenta utilizada na construção visual de sistemas orientados a objetos. Ele consiste basicamente de um desenhador acoplado a um mecanismo de definição e de um gerador de código fonte (SILVEIRA, 1999; PINTO, 2004). O desenhador constrói diagramas de casos de uso, classes, estados e seqüência. O mecanismo de definição permite a associação de textos em objetos dos diagramas. O gerador de código fonte transforma os desenhos e as definições em código de acordo com a arquitetura MVC.

A ferramenta implementa aos macro-requisitos MRQ-7, MRQ-8, MRQ-8.1, MRQ-9, MRQ-9.1, MRQ-10, MRQ-10.1, MRQ-11, MRQ-11.1, MRQ-12. Ela foi desenvolvida em ***Delphi*** e os seus dados são armazenados através de **arquivos binários**. A interface do desenhador do *FastCase* é apresentada na figura 4.4.

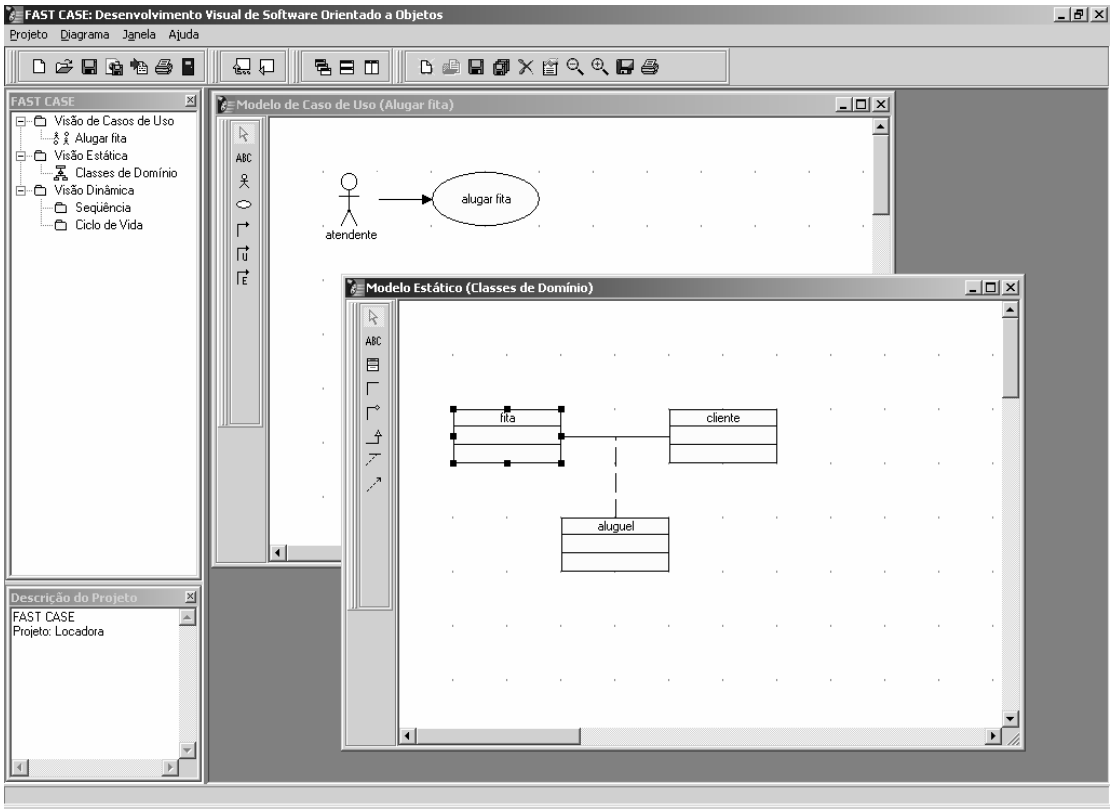


Figura 4.4 - Interface da Ferramenta *FastCase*

4.4.3 Ferramenta *HPReq*

A ferramenta *HPReq* (Heurísticas para transformação Processo –> Requisitos) (CRUZ, 2004) foi desenvolvida com o objetivo de acelerar a obtenção dos modelos de requisitos funcionais através da transformação automática do diagrama de processos em diagrama de casos de uso. O *HPReq* utiliza o recurso de importação e exportação de diagramas da ferramenta *Enterprise Architect* (SPARXSYSTEMS, 2006) para obter e gerar os diagramas. Esta ferramenta implementa os macro-requisitos MRQ-6, MRQ-6.1, MRQ-6.2.

Esta ferramenta foi desenvolvida utilizando a linguagem de programação ***C#***, no ambiente de desenvolvimento ***Dot Net***. Os arquivos de entrada e de saída da ferramenta são escritos na linguagem **XML**, seguindo o *schema* definido pela ferramenta *Enterprise Architect.* A interface do *HPReq* é apresentada na figura 4.5.

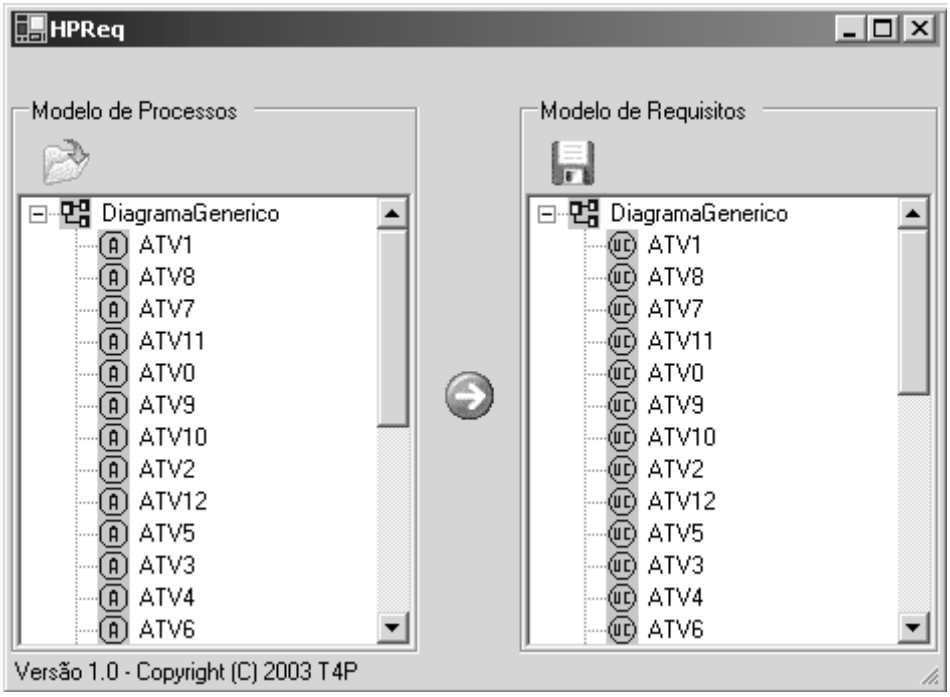


Figura 4.5 - Interface da Ferramenta *HPReq*

4.5 Arquitetura do Ambiente

Com o levantamento dos requisitos, da interface e a realização do inventário das ferramentas do grupo GETI, foi possível estabelecer uma arquitetura para o ambiente. Assim, escolhemos o modelo arquitetural **Repositório** (SHAW; GARLAN, 1996) para desenvolver RAPDIS. Esta arquitetura possui uma estrutura de dados central que representa o repositório e possui um conjunto de componentes independentes que operam neste repositório, como é mostrado na figura 4.6 abaixo.

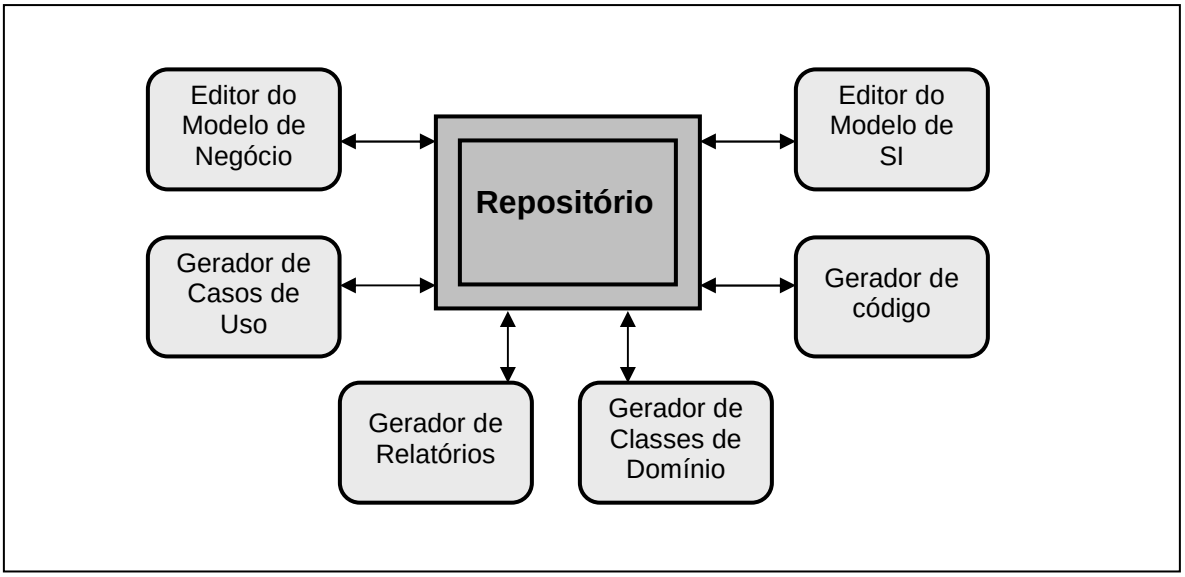


Figura 4.6 - Arquitetura do ambiente

4.5.1 Componentes do Ambiente

O ambiente RAPDIS é constituído por seis componentes: Editor do Modelo de Negócio, Editor do Modelo de Sistema de Informação, Gerador de Casos de Uso, Gerador de Classes de Domínio, Gerador de Código e Gerador de Relatórios. Todos os componentes foram desenvolvidos em *Delphi* e seguem a arquitetura MVC. Apresentamos a seguir a descrição de cada um destes componentes.

4.5.1.1 Editor do Modelo de Negócio

Através deste editor é possível gerenciar as informações que compõem um modelo de negócio (CIM). Seguindo a técnica apresentada em 2.3.1, este modelo é construído no RAPDIS através dos seguintes objetos: termos, objetivos, recursos, processos e regras do negócio.

Os objetivos e recursos são especificados através de um texto livre. Já as definições dos termos e das regras são capturadas utilizando um subconjunto da língua portuguesa, o qual chamamos Português Estruturado. Os processos são modelados através do desenhador do diagrama de atividades.

O editor foi construído aproveitando as funcionalidades para tratamento de regras de negócio disponíveis no Regula. A figura 4.7 mostra a interface para a edição de uma regra do modelo de negócio no RAPDIS.

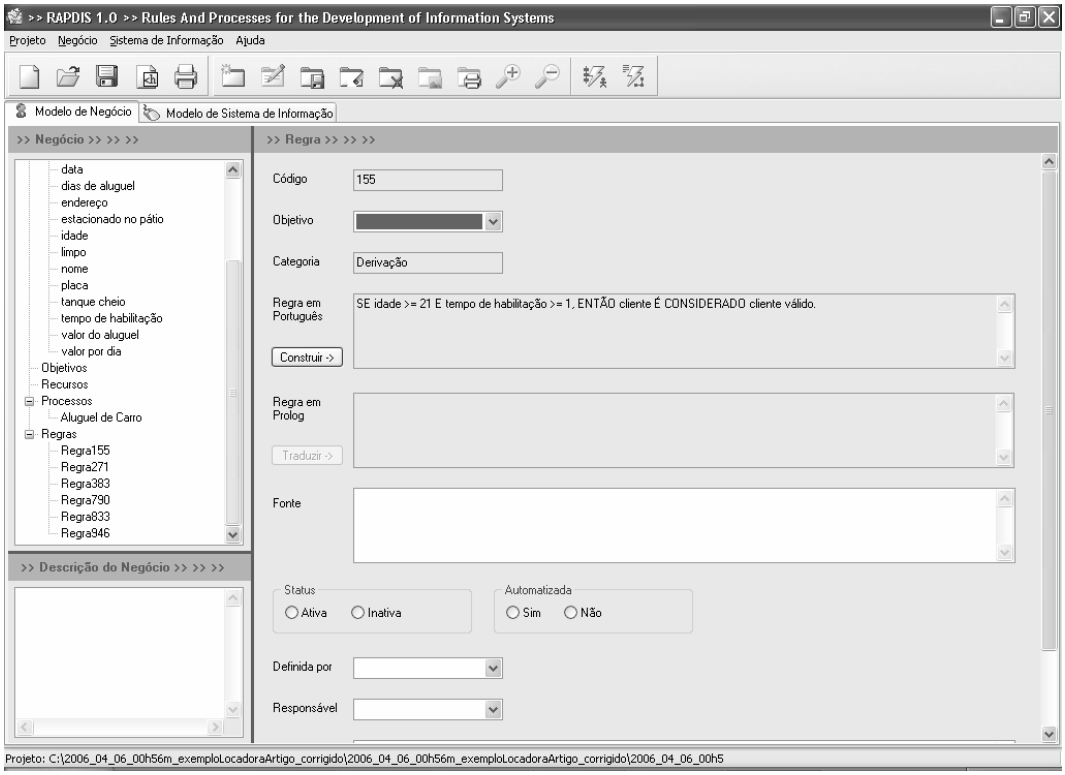


Figura 4.7 – Utilização do Editor do Modelo de Negócio

4.5.1.2 Editor do Modelo de Sistema de Informação

O editor modelo de sistema de informação (PIM) permite a modelagem dos principais diagramas da UML: o diagrama de casos de uso, o diagrama de classes, o diagrama de estados e o diagrama de seqüência. Este editor disponibiliza um desenhador para a construção destes diagramas (figura 4.8) e realiza o armazenamento das definições dos elementos presentes nos diagramas. A ferramenta FastCase foi a base para sua construção.

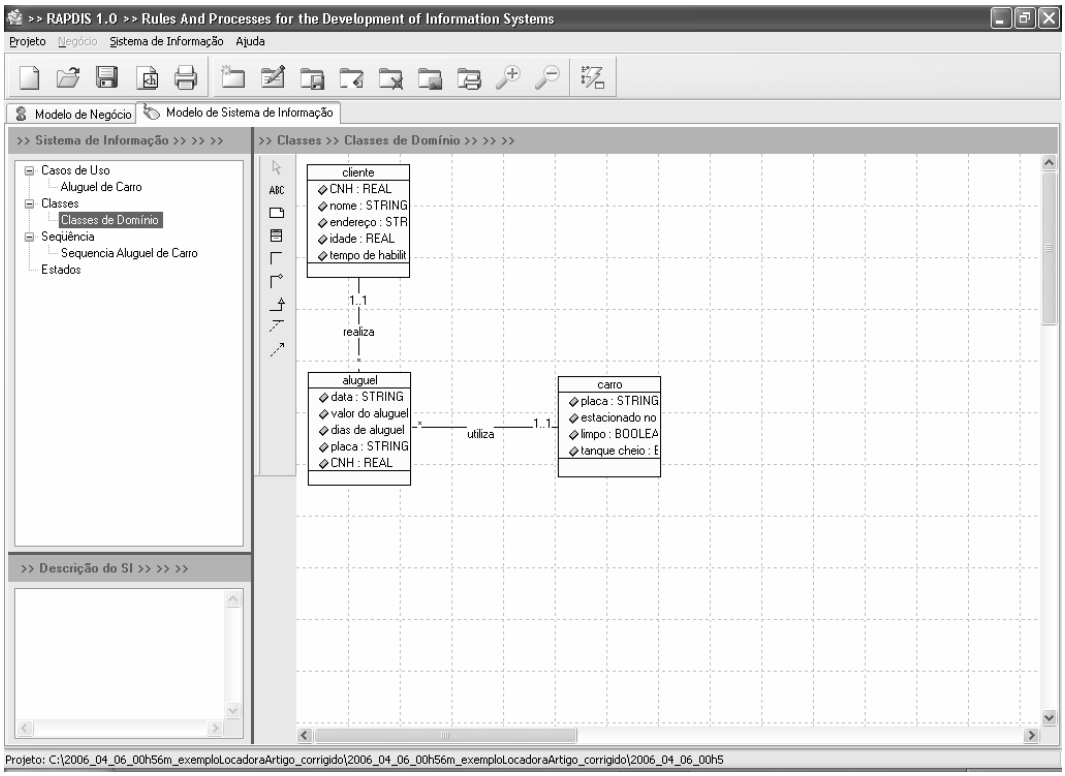


Figura 4.8 – Utilização do Editor do Modelo de Sistema de Informação

4.5.1.3 Gerador de Casos de Uso

O objetivo deste gerador é facilitar a especificação dos requisitos de um sistema de informação a partir da transformação automática dos processos de negócio (CIM → PIM). Assim, este gerador funciona implementando heurísticas que mapeiam os elementos de um diagrama de atividade (utilizado para representar processos de negócio) em elementos do

diagrama de casos de uso. Estas regras seguem o método apresentado na seção 2.3.3.2. A figura 4.9 ilustra a utilização deste gerador.

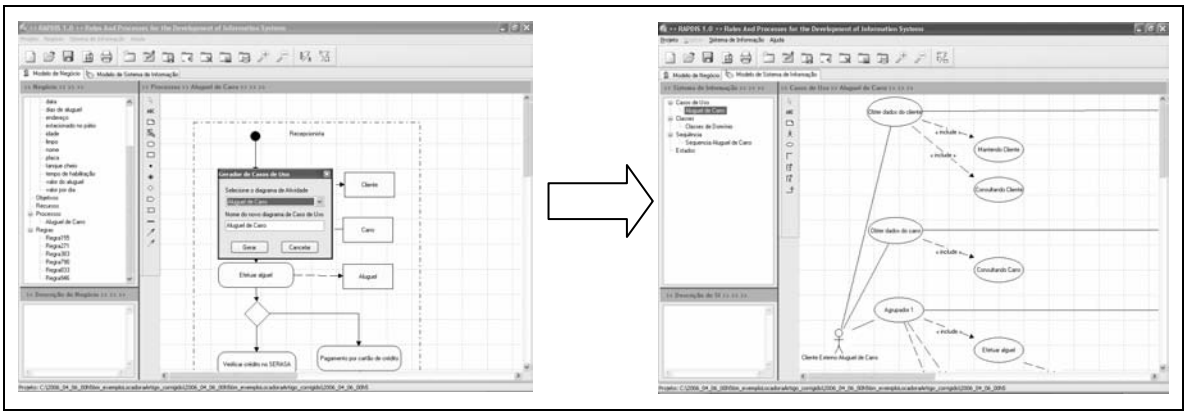


Figura 4.9 - Utilização do Gerador de Casos de Uso

4.5.1.4 Gerador de Classes de Domínio

Uma vez que os termos de um determinado negócio tenham sido definidos, é possível mapear alguns destes termos em classes de domínio (diagrama de classes) (CIM → PIM). O Gerador de Classes de Domínio implementa as regras do método apresentado na seção 2.3.3.1, automatizando esta transformação. A figura 4.10 a seguir apresenta a utilização deste gerador.

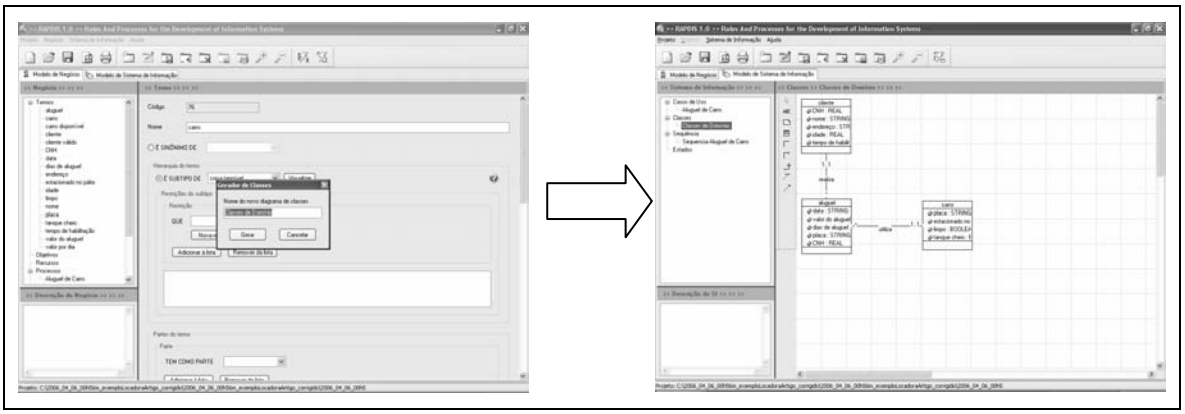


Figura 4.10 - Utilização do Gerador de Classes de Domínio

4.5.1.5 Gerador de Código

A idéia central deste gerador é transformação de um modelo abstrato formal de alto nível em um programa para plataforma específica (PIM → PSM → Código). Ou seja, o gerador realiza a transformação do modelo de sistema de informação, representado através dos diagramas da UML, em código-fonte do programa, representado através de uma linguagem de programação padrão (*Delphi* ou *Java*). Para realizar esta transformação foram implementadas as regras apresentadas na seção 2.3.3.3. A figura 4.11 mostra a configuração deste gerador.

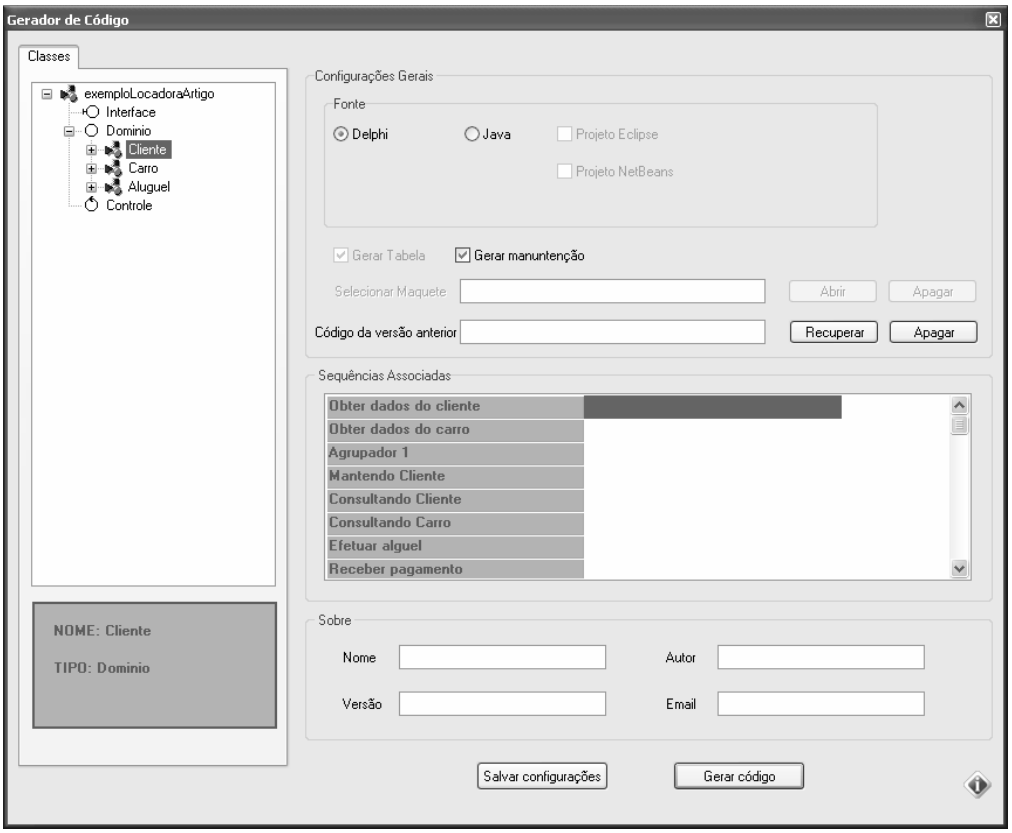


Figura 4.11 - Configuração do Gerador de Código

4.5.1.6 Gerador de Relatórios

Através deste gerador, é possível montar um relatório contendo as principais informações do modelo de negócio e do modelo de sistema de informação. Este relatório é gerado no

formato HTML e tem o intuito de facilitar a divulgação e a impressão das informações sobre um projeto construído no RAPDIS. A figura 4.12 apresenta a tela de configuração do gerador de relatórios.

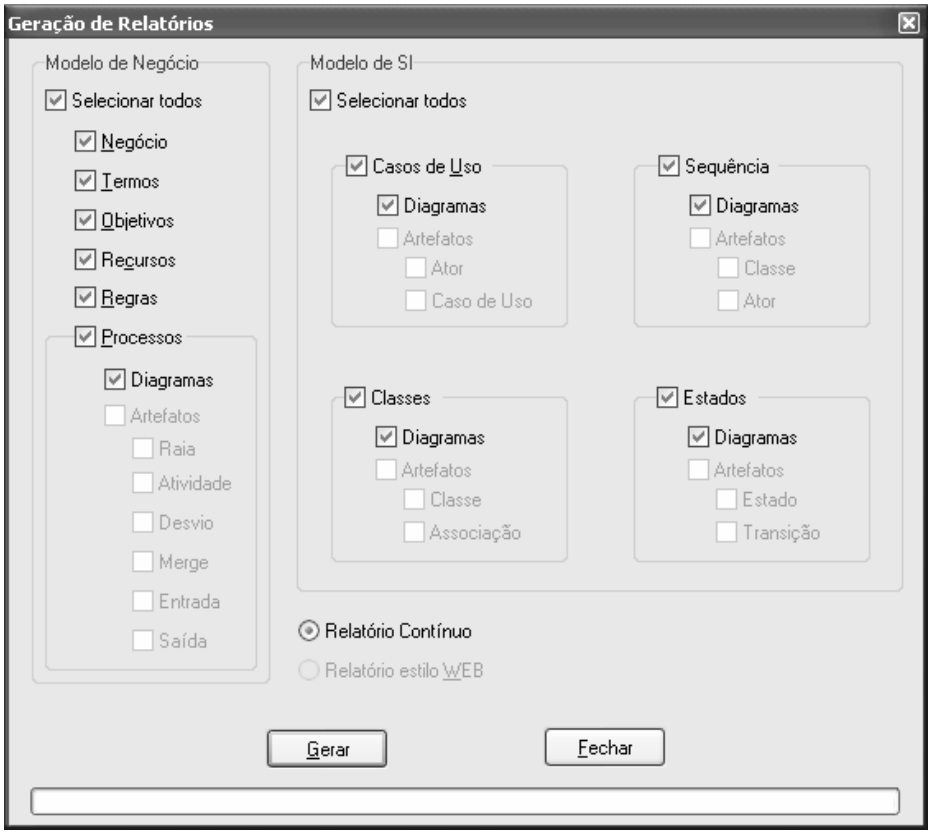


Figura 4.12 - Configuração do Gerador de Relatórios

4.5.2 Repositório do Ambiente

Uma das partes mais importantes de um ambiente I-CASE é o seu repositório. Ele é o centro de acúmulo e armazenamento das informações. De acordo com (PRESSMAN, 2001), o repositório deve fornecer as seguintes funções: integridade dos dados, compartilhamento da informação, integração dados/ferramenta, integração dados/dados, imposição de metodologia e padronização de documentos.

Para realizar estas funções o repositório é definido em termos de um metamodelo. O metamodelo é o gabarito no qual a informação é colocada. O metamodelo do RAPDIS é representado pela figura 4.13.

A raiz deste metamodelo é representada pelo **Projeto** que será desenvolvido no RAPDIS. Este projeto é composto por dois modelos: o **Modelo de Negócio** e o **Modelo de Sistema de Informação**.

O Modelo de Negócio segue a definição apresentada na seção 2.3.1, sendo formado por **Objetivos**, **Recursos**, **Processos**, **Termos** e **Regras**. Os objetivos estão associados a metas. As metas estão associadas a termos. Os recursos estão associados a termos. Os termos relacionam-se entre si e este relacionamento pode ser especializado em: associação, subtipo ou atributo. As regras estão associadas a termos e podem ser especializadas em: cálculo, derivação, habilitadora de ação (habilitadora), proibição, permissão e obrigação (PPO).

O modelo de Sistema de Informação segue a definição da seção 2.3.2, sendo constituído pelos diagramas de **Casos de Uso**, **Classe**, **Estado**, **Seqüência** e pelos parâmetros para geração de código (**Implementação**). Os diagramas estão associados a elementos e os elementos relacionam-se entre si através de links. Os elementos e os links estão associados a uma definição.

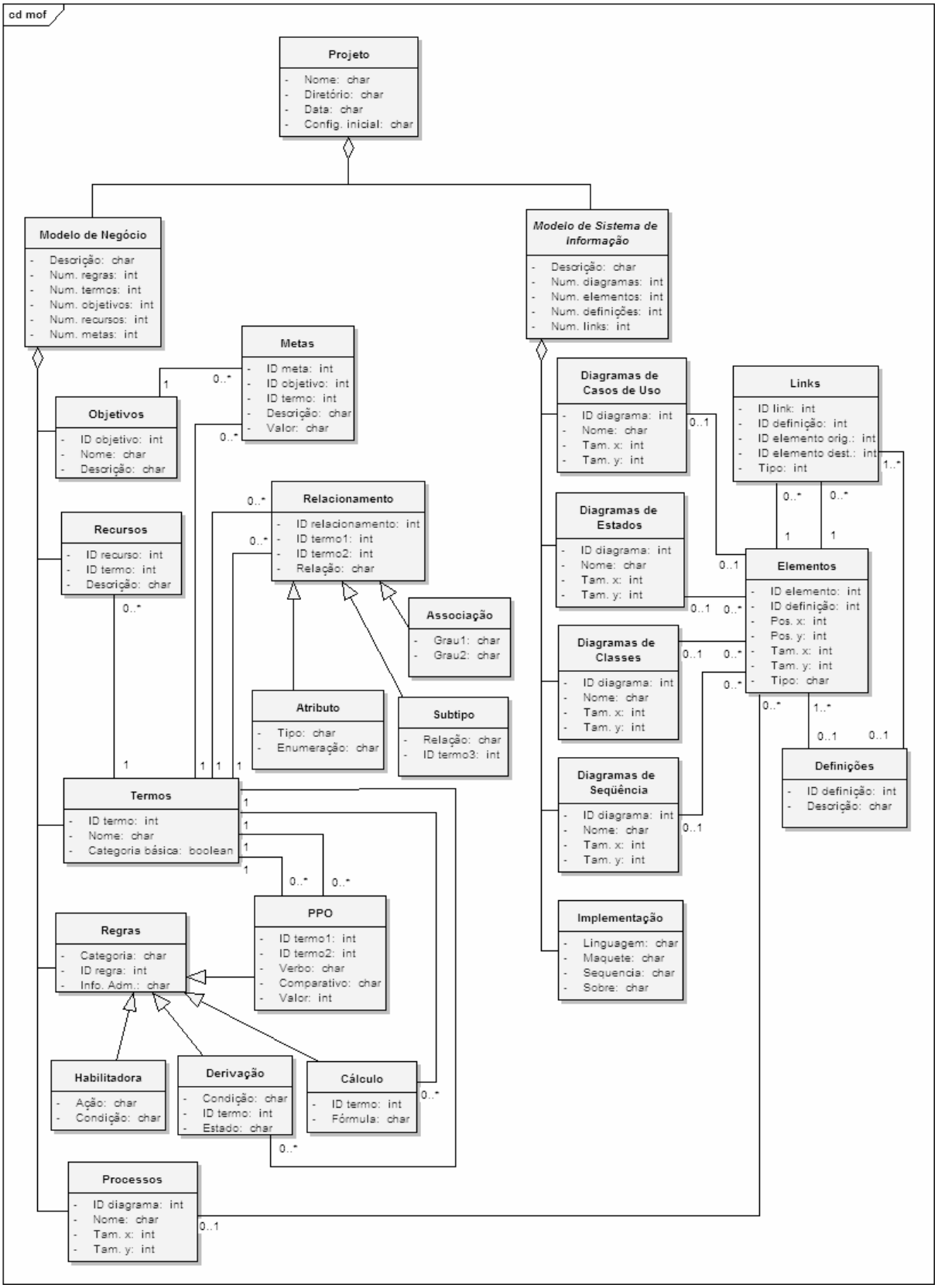


Figura 4.13 - Metamodelo do RAPDIS

Para a implementação do metamodelo do RAPDIS utilizamos a Linguagem de Marcação Extensível (*eXtensible Markup Language*, XML). A XML é uma linguagem padronizada para a construção de documentos eletrônicos com formato simples, textual, estruturado, flexível a mudanças e portável em diversas plataformas tecnológicas. Esta linguagem possui como objetivo principal descrever informações (YERGEAU *et al*, 2004). Estas características da XML e a existência de Interfaces de Programação de Aplicativos (*Application Program Interfaces,* API) para a manipulação de dados em documentos XML motivaram a escolha desta linguagem para implementação do repositório do RAPDIS. O modelo físico do repositório do RAPDIS é apresentado na figura 4.14 a seguir.

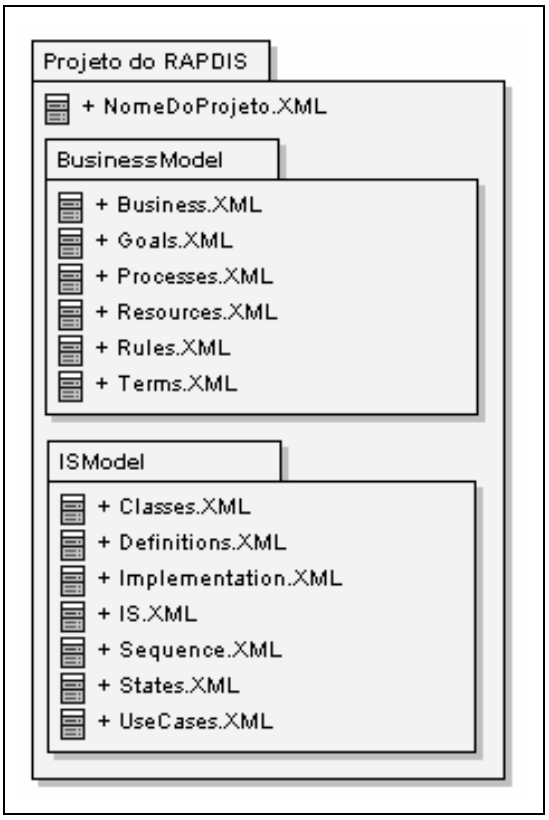


Figura 4.14 – Modelo Físico do Repositório do RAPDIS

A raiz do modelo físico é a pasta com o nome do projeto RAPDIS onde será armazenado o arquivo que identifica o projeto (“NomeDoProjeto.XML”). Dentro desta pasta encontramos duas subpastas: “BusinessModel” e “ISModel”.

Na subpasta “BusinessModel” ficarão todos os arquivos relacionados ao Modelo de Negócio. Estes arquivos são:

- **Business.XML** - armazena a descrição do modelo de Negócio;
- **Terms.XML** – armazena os Termos e as suas definições;
- **Rules.XML** – armazena as Regras separadas por categoria;
- **Goals.XML** – armazena os Objetivos e as metas relacionadas;
- **Resources.XML** – armazena os Recursos e as suas definições; e
- **Process.XML** – armazena os elementos gráficos que compõem o Modelo de Processos desenhado.

Já na subpasta “ISModel” ficarão todos os arquivos relacionados ao Modelo de Sistema de Informação e geração de código. São eles:

- **IS.XML** – armazena a descrição do Modelo de Sistema de Informação.
- **UseCases.XML** – armazena os elementos gráficos que compõem o Diagrama de Casos de Uso desenhado.
- **Classes.XML** – armazena os elementos gráficos que compõem o Diagrama de Classes desenhado.
- **Sequence.XML** - armazena os elementos gráficos que compõem o Diagrama de Seqüência desenhado.
- **States.XML** - armazena os elementos gráficos que compõem o Diagrama de Estados desenhado.

- **Definitions.XML** - armazena todas as definições dos elementos que foram desenhados nos diagramas.
- **Implementation.XML** – armazena a configuração do Gerador de Código.

A utilização de um conjunto de arquivos em XML para armazenamento das informações sobre o projeto facilita a instalação do ambiente e compartilhamento da base de dados. Estas duas características são essenciais para a utilização do RAPDIS no meio acadêmico. Além do RAPDIS, outros ambientes CASE utilizam a linguagem XML e a sua variante para Intercâmbio de Metadados (*XML Metadata Interchange*, XMI) como forma de armazenar e trocar dados. Entretanto, após realizarmos testes com as ferramentas de diferentes fabricantes (*Rational Rose*[3], *Enterprise Architect*[4], *MagicDraw*[5], *Poseidon*[6], *Jude*[7], *Visual Paradigm*[8] e *IdeogramicUML*[9]), constatamos que não existe um padrão de estrutura para os documentos XML gerados por estas ferramentas. Assim, optamos por definir uma estrutura específica para cada documento XML do RAPDIS, elaboradas de acordo com as necessidades deste ambiente. No APÊNDICE C deste texto descrevemos estas estruturas através da linguagem *XML Schema*.

4.6 Conclusão

Este capítulo descreveu as características do ambiente RAPDIS que representa um conjunto de ferramentas que sevem de apoio ao processo RAPDIS. Mais detalhes sobre este

[3] Endereço eletrônico do fabricante: http://www.rational.com
[4] Endereço eletrônico do fabricante: http://www.sparxsystems.com.au
[5] Endereço eletrônico do fabricante: http://www.magicdraw.com
[6] Endereço eletrônico do fabricante: http://www.gentleware.com
[7] Endereço eletrônico do fabricante: http://jude.change-vision.com/jude-web/index.html
[8] Endereço eletrônico do fabricante: http://www.visual-paradigm.com
[9] Endereço eletrônico do fabricante: http://www.ideogramic.com

ambiente podem ser encontrados no APÊNDICE D deste texto (Manual do Usuário) e no endereço eletrônico http://www.geti.dcc.ufrj.br/projetos/rapdis, onde é possível efetuar o *download* de sua última versão e visualizar um vídeo de demonstração. No capítulo seguinte, apresentaremos uma avaliação do RAPDIS.

5. AVALIAÇÃO DO RAPDIS

"Se for desenvolvido sem pensar
e aplicado sem atenção, o processo
pode se tornar a morte do bom senso."

Philip K. Howard

5.1 Introdução

Após a criação do processo e do ambiente RAPDIS, surgiu a necessidade de se verificar a qualidade desta nova proposta. Para isto, realizamos uma avaliação que foi composta por duas etapas: análise da percepção dos usuários do processo e do ambiente RAPDIS e comparação do ambiente RAPDIS com outras ferramentas baseadas em MDA. O método utilizado nestas etapas e os resultados obtidos são discutidos neste capítulo.

5.2 Análise da Percepção dos Usuários do Processo e do Ambiente RAPDIS

A aceitação de uma nova tecnologia está fortemente ligada a fatores humanos (IGBARIA, PARASURAMAN, BAROUDI, 1996). Por este motivo, é fundamental analisar a percepção dos indivíduos que efetivamente lidarão com o novo sistema.

5.2.1 Método Utilizado na Avaliação

Para avaliar a percepção dos usuários do RAPDIS (processo e ambiente), foi realizado um estudo com a turma de Modelagem de Sistemas de Informação 2 (MSI2) do curso de graduação em Ciência da Computação da UFRJ. Esta disciplina apresentava 4 meses de duração, com 4 horas de aula por semana. A disciplina contou com a participação de 28 alunos, que foram divididos em 8 grupos de trabalho (5 grupos de 4 alunos, 2 grupos de 3 alunos e 1 grupo de 2 alunos).

No primeiro dia de aula, cada grupo escolheu um negócio real que serviu de tema para a confecção do trabalho. Os temas escolhidos foram: Clínica Odontológica Dr. Simone Barreto, Aluguel de Máquinas, Farmácia do IPPMG, Pizzarias GatoPardo, Boliche Barra Bowling, Fábrica de Páginas Web, Centro de Informações e Atendimento Toxicológicos - CIAT e Venda de Conteúdo para Celulares - TNL. O trabalho foi realizado de forma incremental, com interações semanais. As interações eram realizadas da seguinte maneira: os alunos recebiam uma aula expositiva sobre uma parte do processo RAPDIS e era solicitado que eles aplicassem os conceitos aprendidos no trabalho. A aplicação dos conceitos se dava através do ambiente RAPDIS. Todas as interações eram acompanhadas através das apresentações dos grupos e das aulas de monitoria. O objetivo do acompanhamento era tirar dúvidas e revisar os trabalhos.

Para a confecção dos trabalhos, os alunos utilizaram todo o processo RAPDIS; desde a construção do modelo de negócio, até a implementação do sistema de informação. Além disso, todas as funcionalidades do ambiente RAPDIS foram utilizadas para apoiar a aplicação deste processo. Naturalmente, alguns defeitos foram encontrados durante a utilização deste ambiente. Estes defeitos eram relatados pelos alunos através da lista de discussão da disciplina e a equipe de desenvolvimento do RAPDIS realizava as correções, disponibilizando na Internet uma nova versão do ambiente para ser utilizada.

Ao final da disciplina, os grupos entregaram os modelos construídos no RAPDIS, o sistema de informação implementado e um relatório descrevendo o trabalho realizado. A entrega dos trabalhos permitiu verificar a viabilidade do processo e do ambiente RAPDIS, porém era necessário ainda avaliar a percepção dos usuários diante desta nova proposta. Para avaliar este aspecto tentamos, inicialmente, fazer com que os alunos respondessem por escrito a um questionário (quadro 5.1). Esta tentativa, entretanto, não foi satisfatória, pois

as perguntas eram muito amplas e as respostas obtidas foram insuficientes para completar uma avaliação qualitativa.

Quadro 5.1 - Questionário

1. Como você avaliaria a disciplina de MSI2?
2. Quanto tempo foi gasto em cada etapa do trabalho?
3. Compare o diagrama de classes gerado pelo RAPDIS e o diagrama final.
4. Compare o diagrama de casos de uso gerado pelo RAPDIS e o diagrama final.
5. Compare o código gerado pelo RAPDIS e o código final.
6. Como você avaliaria o RAPDIS?
7. Você acredita que seja viável aplicar os conceitos aprendidos nesta disciplina no seu ambiente de trabalho? Justifique a sua resposta.

Após analisarmos as falhas dessa primeira tentativa, optamos por aplicar a técnica da entrevista como forma de complementar as respostas obtidas. Optamos por esta técnica, pois ela estabelece uma interação entre o entrevistado e o entrevistador que permite colher uma gama maior de informações, aprofundar os dados fornecidos, e realizar correções sobre dados levantados, ouvindo direta e imediatamente da fonte informante (LAKATOS; MARCONI, 2006).

Para guiar a entrevista e obter resultados uniformes com o grupo de entrevistados, foi elaborado um roteiro (quadro 5.2). Este roteiro de entrevista é divido em três partes: Perfil do Entrevistado, Conhecimento sobre as Tecnologias Relacionadas ao RAPDIS e Opinião sobre o RAPDIS. A primeira parte (Perfil do Entrevistado) visa descobrir informações sobre o histórico do aluno e a sua experiência profissional, uma vez que estes fatores influenciam na percepção do usuário. A segunda parte (Conhecimento sobre as Tecnologias Relacionadas ao RAPDIS) tem como objetivo analisar o RAPDIS como uma ferramenta de ensino, verificando se o aprendizado de Engenharia de Sistemas foi facilitado por esta ferramenta. E, finalmente, a terceira parte (Opinião sobre o RAPDIS) visa

descobrir os pontos fortes e fracos do processo e do ambiente, apresentando questões sobre usabilidade, aplicabilidade, agilidade e facilidade de manutenção.

Quadro 5.2 - Roteiro da Entrevista

1. Perfil do Entrevistado

- Qual é o seu período na faculdade?
- Qual é a sua experiência no desenvolvimento de sistemas?

2. Conhecimento sobre as Tecnologias Relacionadas ao RAPDIS

- Antes de cursar esta disciplina você tinha conhecimentos sobre Processos de Negócio, Regras de Negócio, UML, MVC e MDA?
- Após a disciplina, você se sente em condições de utilizar estas tecnologias?
- Como o ambiente RAPDIS influenciou o aprendizado destas tecnologias?
- Dê uma nota de 0 a 10 para a importância desta disciplina na sua formação.

3. Opinião sobre o RAPDIS

- Quais são os pontos fortes e fracos do RAPDIS (processo e ambiente)?
- Você possui experiência com outros processos de desenvolvimento como o RUP ou o XP? Como você compararia o RAPDIS a estes outros processos?
- Você possui experiência com outras ferramentas CASE como o Rose ou Together? Como você compararia o ambiente RAPDIS a estas outras ferramentas?
- Você possui experiência com outras ferramentas MDA como o AndroMDA ou OptimalJ? Como você compararia o ambiente RAPDIS a estas outras ferramentas?
- Você acha que o ambiente RAPDIS é fácil de usar?
- Qual é a função mais importante do ambiente RAPDIS?
- Se você fosse desenvolvedor do ambiente RAPDIS, que funcionalidade você implementaria imediatamente?
- Você acha que a utilização do processo e do ambiente RAPDIS torna o desenvolvimento de sistemas mais ágil?
- Você acredita que a estrutura utilizada facilita a manutenção?
- Você gastou mais tempo na modelagem do que o de costume? Você acha que isso facilitou o desenvolvimento?
- Se eu pedisse para você trocar para a tecnologia *Java*, utilizando o gerador do RAPDIS, você acha que seria necessário muito esforço?
- Você acha que é viável utilizar o RAPDIS na instituição onde você trabalha?
- Se você fosse contratado para fazer um sistema hoje, você utilizaria o RAPDIS para desenvolver este sistema?
- Dê uma nota de 0 a 10 para o RAPDIS.

Embora existisse um roteiro, a entrevista foi realizada de forma semi-estruturada, ou seja; foi construído um rol de perguntas básicas, mas havia a possibilidade de torná-la mais flexível. Antes de realizar a entrevista, foram estabelecidos alguns requisitos:

- **As entrevistas devem durar no máximo 30 minutos**. Este tempo limite foi estabelecido para evitar que a entrevista se tornasse cansativa.
- **As entrevistas devem ser gravadas**. Esta gravação tem como objetivo facilitar o processo de coleta de dados. Este requisito foi atendido com a permissão dos entrevistados.
- **As entrevistas devem ser realizadas após a divulgação das notas**. Este requisito foi estabelecido para garantir um clima de confiança e permitir a livre expressão dos alunos.

Foi realizada uma entrevista por grupo. Todos os requisitos acima foram atendidos e todos os grupos foram entrevistados. Na sessão a seguir apresentamos os resultados obtidos com este método.

5.2.2 Resultado da Avaliação

Como se trata de uma pesquisa qualitativa, a análise dos resultados se deu através da construção de um conjunto de categorias do tipo descritivas. Buscamos no registro da fala dos entrevistados os momentos em que estas categorias aparecem, e apresentamos a seguir os resultados, preservando a identidade dos entrevistados.

O perfil identificado nas entrevistas é de alunos que estão concluindo o curso de graduação e que possuem experiência no desenvolvimento de sistemas (trabalham ou já trabalharam nesta área). A maioria dos alunos, entretanto, não conhecia o conceito de MDA

antes cursar a disciplina de MSI2. Apenas dois alunos relataram já conhecer o conceito, embora nunca tenham utilizado a MDA na prática.

A parte prática possibilitada pelo ambiente RAPDIS foi considerada o ponto fonte desta disciplina. Todos os alunos expressaram que o RAPDIS auxiliou o ensino de Engenharia de Sistemas e manifestaram que o ambiente deve continuar a ser utilizada no curso de MSI2.

A geração de código foi a funcionalidade do RAPDIS que mais impressionou os usuários. Muitos alunos disseram que nunca haviam utilizado um gerador de código e que antes de cursar a disciplina não acreditavam no poder deste tipo de ferramenta. A experiência com o RAPDIS mudou a opinião destes usuários, que hoje conseguem enxergar as vantagens deste tipo de abordagem. O fato do RAPDIS ser uma ferramenta bastante abrangente, oferecendo suporte à modelagem de negócio e à transformação deste modelo, também foi apontado como um fator positivo da ferramenta.

Os alunos apresentaram também algumas sugestões melhorias que precisam ser feitas no RAPDIS. Estas melhorias estão relacionadas ao sincronismo entre os modelos (modelo origem e modelo destino) após a realização das transformações, rastreabilidade das transformações, iteração no design e/ou desenvolvimento de modelos (*Round Trip Engineering*), suporte ao Modelo Específico de Plataforma (PSM), suporte a diversas plataformas alvo e controle de versão.

Além disso, os alunos relataram as suas experiências com outras ferramentas CASE como o *Jude*[10], *ArgoUML*[11], *Poseidon*[12], *MS Visio*[13], *Rational Rose*[14] e *Enterprise*

[10] Endereço eletrônico do fabricante: http://jude.change-vision.com/jude-web/index.html
[11] Endereço eletrônico do fabricante: http://argouml.tigris.org/
[12] Endereço eletrônico do fabricante: http://www.gentleware.com
[13] Endereço eletrônico do fabricante: http://office.microsoft.com/pt-br/visio/default.aspx

Architect[15]. Comparando o RAPDIS com estas ferramentas, foram apontadas as seguintes diferenças: as outras ferramentas oferecem uma cobertura maior a UML (mais diagramas e mais elementos) e apresentam mais facilidades em seu desenhador (recursos como seleção múltipla, alinhamento das linhas, comando desfazer, copiar e colar). Não foram relatadas experiências com outras ferramentas de MDA.

Na avaliação final dos usuários, o RAPDIS é um ambiente fácil de usar (possui uma interface amigável) e que permite a construção de sistemas de forma estruturada, facilitando o seu desenvolvimento e a sua manutenção. Quando foram perguntados sobre a agilidade proporcionada pelo RAPDIS, os alunos responderam que consideram este processo de desenvolvimento mais rápido devido às transformações automáticas entre os modelos. Porém, foi levantado também, que o RAPDIS envolve muitos conceitos complexos e que existe uma curva de aprendizado acentuada neste novo processo de desenvolvimento.

Alguns usuários manifestaram o interesse em continuar usando o RAPDIS para fins acadêmicos e até profissionais. Além disso, todos os alunos esperam aplicar pelo menos uma parte do processo aprendido em seu ambiente de trabalho. Estas respostas e as notas que os alunos atribuíram ao RAPDIS (todas acima de 6) são indícios de que o RAPDIS (processo e ambiente) possui uma forte aceitação dentro deste grupo de usuários.

5.3 Comparação do ambiente RAPDIS com outras Ferramentas MDA

Existem atualmente mais de 40 ferramentas (incluindo as comerciais, gratuitas e *open source*) que oferecem suporte a uma ou mais características da MDA (OMG, 2007). Deste

[14] Endereço eletrônico do fabricante: http://www.rational.com
[15] Endereço eletrônico do fabricante: http://www.sparxsystems.com.au

modo, uma outra forma de avaliar o ambiente RAPDIS consiste em compará-lo com estas ferramentas.

5.3.1 Método Utilizado na Avaliação

Para efetuar esta comparação, utilizamos o método conhecido como *benchmarking*. Este método consiste em executar testes com várias ferramentas e comparar as características que elas apresentam (KITCHENHAM, 1996). A lista de características e de ferramentas de MDA utilizada neste *benchmarking* tem como base uma pesquisa realizada para a obtenção do grau de mestre em Engenharia e Gerenciamento de Sistemas de Informação do Departamento de Ciência da Computação e Sistemas, Royal Institute of Tecnology, Karonlinska University Hospital, Estocolmo, Suécia (TARIQ; AKHTER, 2005; TARIQ *et al*, 2005).

Esta lista de características foi extraída do documento da OMG de especificação da MDA (MUKERJI; MILLER, 2003) e da UML (OMG, 2005). Cada característica da lista apresenta um peso. Para estabelecer este peso, foi distribuído um questionário para 33 pessoas de 22 organizações diferentes (instituições de ensino e pesquisa, empresas de consultoria e fabricantes de software). 14 pessoas responderam ao questionário, ordenando as características de acordo com o seu grau de importância. A versão final desta lista de características é apresentada no quadro 5.3.

Quadro 5.3 - Lista de Características

ID	Nome	Peso
1	Suporte ao *Computational Independent Model* (CIM).	2,6
2	Suporte ao *Platform Independent Model* (PIM).	4,9
3	Suporte ao *Platform Specific Model* (PSM).	4,5
4	Suporte à criação, customização e extensão do *Meta Object Facility* (MOF).	2,4
5	Modelos desenvolvidos devem ser compatíveis ao MOF.	3,8

6	Suporte à UML 2.0.	3,5
7	Suporte à UML 1.5.	2,5
8	Suporte à UML 1.4.	1,5
9	Suporte à UML 1.3.	1,2
10	Suporte ao Diagrama de Casos de Uso.	2,8
11	Suporte ao Diagrama de Classes.	4,3
12	Suporte ao Diagrama de Seqüência.	3,0
13	Suporte ao Diagrama de Colaboração.	2,3
14	Suporte ao Diagrama de Atividades.	3,1
15	Suporte ao Diagrama de Estados.	3,5
16	Suporte ao Diagrama de Componentes.	3,1
17	Suporte ao Diagrama de Objetos.	2,8
18	Suporte ao Diagrama de Distribuição.	3,0
19	Suporte ao Diagrama de Tempo.	2,0
20	Suporte ao Diagrama de Visão Geral de Interação.	2,1
21	Suporte ao Diagrama de Estrutura de Composição.	2,5
22	Suporte ao *UML Action Semantics* (AS).	2,3
23	Suporte aos padrões de *UML Profiles* existentes na indústria de ferramentas de MDA.	2,6
24	Suporte à criação, customização e extensão de *UML Profiles.*	4,0
25	Suporte ao *XML Metadata Interchange* (XMI) e conseqüentemente a importação e exportação de UML e MOF utilizando o formato padrão XMI.	4,9
26	Suporte ao *Common Warehouse Meta-model* (CWM).	1,7
27	Suporte ao *Mapping Language Portability.*	1,8
28	Suporte ao *Query View Transformation* (QVT).	3,7
29	Suporte ao *Object Constraint Language* (OCL).	3,1
30	Transformação CIM → PIM	3,9
31	Transformação CIM ← PIM	1,3
32	Transformação PIM → PSM → Código.	3,9
33	Transformação PIM ← PSM ← Código.	1,3
34	Transformação direta PIM → Código.	2,5
35	Transformação direta PIM ← Código.	0,5
36	Transformação PSM → PSM *Bridge* → PSM. Por exemplo, PSM *Java* para PSM *Dot Net*.	1,2
37	Transformação PSM ← PSM *Bridge* ← PSM. Por exemplo, PSM *Dot Net* para PSM *Java*.	0,8
38	Transformação Código → Código *Bridge* → Código.	0,5

	Por exemplo, código *Java* para código *Dot Net*.	
39	Transformação Código ← Código *Bridge* ← Código. Por exemplo, código *Dot Net* para código *Java*.	0,5
40	Suporte à iteração no design e/ou desenvolvimento de modelos. As alterações feitas manualmente no modelo/código gerado devem permanecer na próxima iteração.	2,9
41	Transformações baseadas em marcações.	2,8
42	Transformações baseadas em metamodelos.	4,0
43	Transformações baseadas em modelos.	4,0
44	Transformações baseadas em *patterns.*	3,7
45	Registro das transformações. Todas as transformações são registadas/salvas num formato legível como o XML.	2,6
46	Rastreabilidade das transformações. Por exemplo, a capacidade de encontrar que elemento do PIM foi transformado em outro modelo.	4,0
47	Suporte à mesclagem modelos. Mesclagem de 2 ou mais CIMs/PIMs/PSMs.	1,2
48	Importar e/ou exportar de serviços de domínios existentes como saúde, transporte, etc.	2,9
49	Prover templates para serviços de domínios existentes como saúde, transporte, etc.	1,1
50	Prover suporte à extensões para estes *templates* de serviços de domínio e/ou modelos importados.	2,1
51	Suporte à *Pervasive Services* como *Directory* e *Naming Services*, *Event Handling/Notification Services*, etc.	1,2
52	Código gerado é baseado nos padrões de projeto desenvolvidos pelo *Gang Of Four* (GOF) ou pela indústria.	3,5
53	Suporte para uma ou mais plataformas alvo, tais como *Java*, *C++*, *Delphi*, etc.	4,1
54	Suporte a um ou mais sistemas alvo, tais como sistemas de informação, sistemas complexos, sistemas embutidos, etc.	3,0
55	A ferramenta de MDA pode ser executada em diferentes sistemas operacionais.	3,6
56	Suporte a geração automática de uma documentação do ciclo de vida projeto. Por exemplo, documentos de análise, *design*, teste e distribuição.	3,2
57	Suporte à integração com sistemas de controle de versão como CVS, VSS, etc.	4,2
58	Suporte a testes. Geração de *scripts* de teste como JUnit para *Java*, etc.	3,2
59	Suporte a *templates* de código, para que o usuário possa definir as suas próprias transformações.	4,2

A lista de ferramentas de MDA utilizadas nesta comparação foi escolhida aleatoriamente. Todas as ferramentas foram testadas com o mesmo exemplo: um sistema para o gerenciamento de aparelhos biomédicos que serão utilizados no departamento de

medicina ou na casa dos pacientes. As ferramentas receberam uma nota por característica. Esta nota variava de 0 a 5, onde 0 significava que a característica não atendida e 5 significava que a característica completamente atendida.

O trabalho que serviu de base para esta comparação, entretanto, não apresenta em detalhes o significado das notas atribuídas para cada característica da lista. A ausência desta informação dificultou a comparação do RAPDIS com as demais ferramentas apresentadas neste trabalho. Assim, para evitar uma avaliação subjetiva, optamos por simplificar a comparação, apresentando apenas as notas 0 ou 1, que significavam, respectivamente, característica não atendida e característica atendida. O quadro 5.4 apresenta a lista de ferramentas utilizadas nesta comparação.

Quadro 5.4 - Lista de Ferramentas

Sigla	Nome	Fabricante	Versão	Preço
OJ	Optimal J[16]	Computware	3.3	$18,000
AS	Arcstyler[17]	Interactive Objects	5.0	A partir de $12,000
CT	Constructor[18]	Dot Net Builders	1.0	n/a
CA	Codagen Architect[19]	Codagen	3.2	n/a
OG	Objecteering[20]	Objecteering Software	5.3.0	$2,280
AM	Ameos[21]	Aonix	9.1.5	$4,395
TA	Together Architect[22]	Borland	1.0	A partir de $5,000
MS	MDE Studio[23]	M1 Global	3.1.3	Gratuita para fins não comerciais
XM	XMF Mosaic[24]	Xactium	1.0	A partir de $5,000
RP	RAPDIS	NCE/UFRJ	1.0	Gratuita

[16] Endereço eletrônico do fabricante: http://www.compuware.com
[17] Endereço eletrônico do fabricante: http://www.arcstyler.com
[18] Endereço eletrônico do fabricante: http://www.dotnetbuilders.com
[19] Endereço eletrônico do fabricante: http://www.manyeta.com
[20] Endereço eletrônico do fabricante: http://www.objecteering.com
[21] Endereço eletrônico do fabricante: http://www.aonix.de
[22] Endereço eletrônico do fabricante: http://www.borland.com
[23] Endereço eletrônico do fabricante: http://www.m1global.com
[24] Endereço eletrônico do fabricante: http://www.xactium.com

Para completar esta comparação, foi atribuída uma nota para cada ferramenta. Esta nota final foi calculada utilizando a seguinte fórmula:

$$\text{Nota Final} = \sum_{i=1}^{59} (\text{Nota da Característica i} \times \text{Peso da Característica i})$$

5.3.2 Resultado da Avaliação

Após aplicar o método apresentado na seção anterior, obtivemos o quadro 5.5 com os resultados. As linhas representam as 59 características desejáveis e as colunas representam as 10 ferramentas avaliadas. A última linha do quadro apresenta a nota final de cada ferramenta.

Quadro 5.5 - Resultado da Comparação

	OJ	AS	CT	CA	OG	AM	TA	MS	XM	RP
1	1	1	0	0	1	1	1	1	1	1
2	1	1	1	1	1	1	1	1	1	1
3	1	1	1	1	0	0	1	1	1	0
4	1	1	0	0	0	0	0	1	1	0
5	1	1	1	1	1	1	0	1	1	1
6	0	0	0	0	0	0	0	0	1	1
7	0	0	0	0	0	0	0	0	0	0
8	1	1	0	0	1	0	1	1	0	0
9	1	1	1	1	1	1	1	1	0	0
10	1	1	0	0	1	1	1	1	0	1
11	1	1	1	0	1	1	1	1	1	1
12	1	1	0	0	1	1	1	0	0	1
13	1	1	0	0	0	1	1	0	0	0
14	1	1	0	0	0	1	1	0	0	1
15	1	1	0	0	0	1	0	0	1	1
16	1	1	0	0	1	1	1	1	1	0
17	1	0	0	0	1	0	0	1	1	0
18	1	1	0	0	1	1	1	1	1	0
19	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	1	0	0	0

21	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	1	1	1	0
23	0	0	0	0	0	0	0	0	1	1
24	1	1	0	1	1	1	0	1	1	0
25	1	1	0	1	1	0	1	1	1	0
26	0	0	0	0	0	0	0	0	0	0
27	0	0	0	0	0	0	1	1	1	0
28	0	0	0	0	0	0	0	0	1	0
29	0	1	0	0	0	0	0	0	1	0
30	0	0	0	0	0	0	0	0	0	1
31	0	0	0	0	0	0	0	0	0	0
32	1	1	1	1	0	0	1	1	1	0
33	1	1	1	0	0	0	1	1	0	0
34	1	1	1	1	1	1	1	1	1	1
35	1	1	0	0	0	0	1	0	0	0
36	0	0	0	0	0	0	1	0	1	0
37	0	0	0	0	0	0	1	0	1	0
38	0	0	0	0	0	0	0	0	0	0
39	0	0	0	0	0	0	0	0	0	0
40	1	1	0	0	0	0	1	1	1	1
41	1	1	0	0	0	0	1	1	1	1
42	1	1	0	0	0	1	1	1	1	1
43	1	1	1	0	0	0	1	1	1	0
44	1	1	0	0	0	1	1	1	1	0
45	1	1	0	0	0	0	1	1	1	0
46	1	1	1	1	1	1	1	0	0	0
47	1	1	0	0	1	0	1	0	1	0
48	0	0	0	0	0	1	1	1	0	0
49	0	0	0	0	0	0	1	0	0	0
50	0	0	0	0	0	0	1	0	0	0
51	0	0	0	0	0	0	0	0	0	0
52	1	1	0	0	0	0	1	0	0	1
53	1	1	1	1	1	1	1	1	1	1
54	1	1	1	1	1	1	1	1	1	1
55	1	1	1	1	1	1	1	1	1	0
56	1	1	1	1	1	1	1	1	1	1

57	1	1	0	0	1	1	1	1	0	0
58	1	1	0	0	0	0	1	1	1	0
59	1	1	1	1	1	1	1	1	1	0
NOTA	120,1	120,4	52,5	51,8	71,9	81	117,9	106	111,7	64

Analisando este quadro, podemos concluir que o RAPDIS está bem posicionado dentro do mercado de ferramentas de MDA. Embora a nota final do RAPDIS não tenha sido a maior, este ambiente mostrou não estar muito distante dos líderes do mercado.

O maior destaque do RAPDIS está no suporte oferecido à construção do CIM e transformação CIM → PIM. Nenhuma outra ferramenta analisada apresentou suporte a esta transformação. Esta característica é, portanto, um importante diferencial deste ambiente (ZHANG *et al*, 2005).

Em compensação, a ausência de suporte a *templates* de código mostrou ser a principal deficiência do RAPDIS. Todas as demais ferramentas analisadas permitem que o usuário defina as suas próprias transformações, enquanto que no RAPDIS, estas transformações são fixas.

A comparação nos permite concluir também que há várias características que fazem parte da especificação da MDA, mas que ainda não foram implementadas por nenhuma das ferramentas analisadas. São elas: suporte ao Diagrama de Tempo, suporte ao Diagrama de Estrutura de Composição, suporte ao Repositório Comum de Metadados (*Common Warehouse Meta-model*, CWM), transformação CIM ← PIM, transformação Código → Código *Bridge* → Código, transformação Código ← Código *Bridge* ← Código e suporte a *Pervasive Services*.

Além disso, observamos que algumas características aparecem com uma freqüência muito baixa (duas ocorrências no máximo). São elas: suporte ao Diagrama de Visão Geral

de Interação, suporte a Linguagem de Restrições de Objetos (*Object Constraint Language*, OCL), suporte a Consulta Visão Transformação (*Query View Transformation*, QVT), transformação CIM → PIM, transformação PSM → PSM *Bridge* → PSM, transformação PSM ← PSM *Bridge* ← PSM, suporte a *templates* para serviços de domínios existentes, suporte a extensões para estes *templates* e suporte aos padrões de *UML Profiles* existentes na indústria de ferramentas de MDA.

A ausência/baixa freqüência destas características pode ser explicada pelo fato destas características estarem relacionadas a conceitos e padrões novos para a indústria de software. A QVT, por exemplo, é um padrão que ainda está em fase de desenvolvimento (OMG, 2002). Como conseqüência, não temos hoje no mercado ferramentas que ofereçam uma cobertura completa a todas as características da MDA. Podemos dizer, portanto, que este é um mercado que ainda está em fase de amadurecimento.

Por fim, vale lembrar que as ferramentas avaliadas neste estudo possuem propósitos diferentes (quadro 5.4). Arcstyler e a OptimalJ, que foram consideradas as melhores ferramentas de MDA, são ferramentas comerciais e foram produzidas por grandes fabricantes de software (*Interactive Objects* e *Computware*). Já o RAPDIS é um produto acadêmico e possui distribuição gratuita.

5.4 Conclusão

Este capítulo discutiu a avaliação do RAPDIS. Os resultados desta avaliação apresentam indícios de que a maioria dos usuários do RAPDIS está satisfeita com o processo e o ambiente, por considerarem que eles facilitam o desenvolvimento de sistemas e auxiliam o aprendizado da disciplina de MSI2. Além disso, a comparação do ambiente RAPDIS com outras ferramentas baseadas em MDA nos permitiu identificar os pontos fortes e fracos

deste ambiente. Os trabalhos produzidos na disciplina MSI2 podem ser encontrados no endereço eletrônico http://www.geti.dcc.ufrj.br/projetos/rapdis. No próximo capítulo, apresentaremos a conclusão deste trabalho e utilizaremos os resultados desta avaliação para apontar as contribuições e as oportunidades de trabalhos futuros.

6. CONCLUSÃO

"Encarar o amanhã pensando em usar os métodos de ontem é imaginar a vida estagnada."

James Bell

6.1 Considerações Finais e Contribuições

A abstração através de modelos é um poderoso instrumento para o entendimento e desenvolvimento de Sistemas de Informação. Neste sentido, diversos trabalhos têm sido realizados pela comunidade de profissionais de Tecnologia da Informação para tentar tirar o máximo de proveito das atividades de modelagem. Esta dissertação apresentou um processo MDA (processo RAPDIS) para o desenvolvimento de sistemas de informação e o ambiente I-CASE (ambiente RAPDIS) que serve de apoio para este processo. Esta abordagem permitiu a integração dos métodos, técnicas e ferramentas desenvolvidas pelo grupo GETI com a proposta da MDA especificada pelo OMG.

Uma das principais contribuições do processo RAPDIS é a transformação de modelos. A transformação do Modelo de Negócio em Modelo de Sistema promove o desenvolvimento de sistemas de informação cujas funcionalidades estejam mais aderentes ao comportamento do negócio. Adicionalmente, a abordagem promove a uniformização da nomenclatura utilizada nos modelos e uma maior consistência e padronização dos modelos de requisitos gerados. Já a transformação do Modelo de Sistema de Informação em Código possibilita a redução no tempo gasto em atividades mecânicas de programação.

O ambiente RAPDIS fornece o suporte computacional necessário para aplicação do processo, automatizando suas atividades. Estudos realizados com um grupo de alunos da graduação mostraram que os usuários apresentam uma percepção positiva do processo e o

ambiente RAPDIS. Na visão dos alunos, o RAPDIS não só auxiliou o desenvolvimento de sistemas de informação, como também contribuiu como uma ferramenta de ensino.

Além disso, a análise comparativa do ambiente RAPDIS com outras ferramentas baseadas em MDA tornou evidente um dos pontos fortes do RAPDIS: o suporte oferecido à construção do CIM e a transformação de CIM para PIM. Esta característica, que não foi coberta por nenhuma outra ferramenta analisada, permitiu avançar o estado-da-arte no assunto desta dissertação.

6.2 Limitações e Trabalhos Futuros

O estudo realizado com os alunos de graduação e a comparação do RAPDIS com as outras ferramentas mostrou também as principais limitações desta abordagem. Assim, as principais características que precisam ser desenvolvidas são: o suporte a *templates* de código que permitam ao usuário a edição das regras de transformação; a construção das transformações no sentido inverso (Código → PSM → PIM → CIM) para permitir a engenharia reversa; o suporte aos demais diagramas da UML (apenas quatro diagramas estão disponíveis no RAPDIS); o suporte à importação e exportação de modelos UML utilizando o padrão XMI; e a integração do RAPDIS com um sistema de controle de versão como CVS e VSS.

Além disso, algumas das limitações encontradas já estão sendo tratadas em outros trabalhos. Neste sentido, destacamos os seguintes trabalhos do grupo GETI que estão em andamento: a revisão e ampliação das regras de transformação do CIM para o PIM, visando contemplar todos os elementos que compõem a Modelagem de Negócio; o estudo sobre as formas de consulta e implementação das Regras de Negócio; o suporte ao PSM, o que permitirá a geração de código através de duas etapas de transformação (PIM → PSM e

PSM → Código); a geração de código para outras plataformas, como *Java* e *Dot Net*; e a implementação de melhorias no mecanismo de *Round Trip* do gerador de código.

Outro trabalho relacionado ao RAPDIS que se encontra em desenvolvimento consiste em um estudo de caso sobre o Sistema Integrado de Gestão Acadêmica da UFRJ (SIGA). O SIGA agrega todos os sistemas de registro acadêmico da UFRJ, viabilizando o acesso às informações para toda comunidade acadêmica. Através deste estudo será possível testar o processo e o ambiente RAPDIS em um projeto real.

Como trabalhos futuros, apontamos a implementação de um construtor de maquetes dentro do ambiente RAPDIS; a implementação de um mecanismo que permita o rastrear as transformações efetuadas; o suporte ao OCL e a geração de código a partir das restrições expressas nesta linguagem; a adaptação do RAPDIS para o desenvolvimento de diferentes tipos de sistemas, como sistemas especialistas e sistemas complexos; o suporte a pontes (*Bridge*) para facilitar a interoperabilidade entre os sistemas; a geração automática dos casos de teste a partir do modelo (*Model Driven Test*); e a complementação dos modelos através de linguagens que permitam a especificação de ações (LEAL *et al*, 2006)

Por fim, vale notar que embora a MDA seja baseada em idéias antigas, como a geração automática de código, a formalização e a padronização destas idéias é um processo recente. Na definição da MDA há muitos tópicos que ainda estão em desenvolvimento, como a linguagem QVT e Ações Semânticas (*Action Semantics*). Vale ainda notar que, com a divulgação de novas versões da MDA e com o desenvolvimento de padrões relacionados a esta arquitetura, o RAPDIS deverá ser revisto para se adequar a estes novos padrões.

REFERÊNCIAS BIBLIOGRÁFICAS

ALENQUER, P. L. - **Regras de Negócio para Análise em Ambientes OLAP**. Dissertação (Mestrado em Informática), Núcleo de Computação Eletrônica, Instituto de Matemática, Universidade Federal do Rio de Janeiro, Rio de Janeiro, julho de 2002.

ARAÚJO, B.; DIAS, F.; MORGADO, G. MARTINS, A. SILVEIRA, D.; MANSO, F.; LIMA, P.; SCHMITZ, E. - **RÉGULA**: Uma Ferramenta para a Captura de Requisitos de Software através de Regras de Negócio. XX Simpósio Brasileiro de Engenharia de Software, Sessão Ferramentas, Florianópolis, SC, Brasil, outubro de 2006.

BECK, K - **Implementation Patterns**. Addison-Wesley, 2006.

BLANC, X.; GERVAIS, M.; SRIPLAKINCH, P. - **Model Bus**: Towards the interoperability of modeling tools. Proceedings of Model-Driven Architecture: Fundations and Applications, 2004.

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I.- **Unified Modeling Language User Guide**. Addison-Wesley. 1999.

BRATKO, I. **Prolog Programming for Artificial Intelligence**. 2ª edição, Ed. Addison-Wesley, 1990.

BROOKS, F. P. - **No Silver Bullet:** Essence and Accident in Software Engineering. Proceedings of the IFIP Tenth World Computing Conference, pp. 1069-1076, 1986.

BRG Business Rules Group - **GUIDE Business Rules Project**: Final Report. Revisão 1.2, 1997. Disponível em: http://www.businessrulesgroup.org Acesso em: 20 jul. 2006, 12:10:00.

BUBENKO, J.; STIRNA, J.; BRASH, D. - **EKD user guide.** Departament of computer and systems sciences, Stockholm: Royal Institute of Technology, 1998.

BURBECK, S. - **Applications Programming in Smalltalk-80**: How to use Model-View-Controller (MVC). Segunda versão, UIUC Smalltalk Archive, Softsmarts Inc., 1987. Disponível em: "http://st-www.cs.uiuc.edu/users/smarch/st-docs/mvc.html". Acesso em: 23 nov. 2005, 18:18:00.

CHEN, M.; NORMAN, R. J. - **A Framework for Integrating Case**. IEEE Software, pages 18--22, 1992.

CRUZ, P. O. S. - **Heurísticas para Identificação de Requisitos de Ambientes de Informações a partir de Modelos de Processos**. Dissertação (Mestrado em Informática), Núcleo de Computação Eletrônica, Instituto de Matemática, Universidade Federal do Rio de Janeiro, Rio de Janeiro, fevereiro de 2004.

CRUZ, P. O. S., SILVEIRA, D. S., SCHMITZ, E. A. - **Heurísticas para Extração dos Casos de Uso de Negócio a Partir da Modelagem de Processos**. III Conferência da Associação Portuguesa de Ambientes de Informação, Coimbra, Portugal, 2002.

CZARNECKI, K.; HELSEN, S. - **Classification of Model Transformation Approaches**. Online Proceedings of the 2nd OOPSLA - Workshop on Generative Techniques in the Context of MDA, Anaheim, California, Outubro 2003.

DIAS, F. G.; MORGADO, G. P.; MARTINS, A. E.; SEABRA, C. M.; SILVEIRA, D. S.; ALENCAR, A. J.; LIMA, P. M. V.; SCHMITZ, E. A. - **Um Ambiente para Modelagem Organizacional Baseado em Regras de Negócio.** I Simpósio Brasileiro de Ambientes de Informação, Porto Alegre, RS, Brasil, Outubro, 2004.

DIAS, F.; MORGADO, G.; OSCAR, P.; SILVEIRA, D.; ALENCAR, A. J.; LIMA, P.; SCHMITZ, E. **Uma Abordagem para a Transformação Automática do Modelo de Negócio em Modelo de Requisitos**. IX Workshop on Requirements Engineering, Rio de Janeiro, RJ, Brasil, Julho, 2006.

ERIKSSON, H.; PENKER, M. - **Business Modeling with UML**: Business Patterns at Work. John Wiley & Sons, 2000.

ESHUIS R., WIERINGA R. - **A Formal Semantics for UML Activity Diagrams**: Formalising Workflow Models. CTIT Technical Report 01-04, University of Twente, Fevereiro, 2001.

FORTE, G. - **Integrated CASE**: a definition. 3º Annual Team-Workers Intl. User Group Conference, Cadre Technologies, 1990.

FRANKEL, D. S. F. - **Model Driven Architecture**: Applying MDA to Enterprise Computing. Wiley, 2003.

GAMMA, E.; JOHNSON, R.; HELM, R.; VLISSIDES, J. - **Design Patterns**: Elements of Reusable Object-Oriented Software. Addison Wesley, 1995.

IEEE - **IEEE Std 1028-1997**: IEEE Standard for Software Reviews. Software Engineerong Standards Committee of the IEEE Computer Society, USA, Março 1998.

IGBARIA, M.; PARASURAMAN, S.; BAROUDI, J. J. - **A motivational model of microcomputer usage**. Journal of Management Information Systems, v. 13, n. 1, p. 127-143, 1996.

KLEPPE, A.; WARMER, J.; BAST, W. - **MDA Explained**: The Model Driven Architecture: Practice and Promise, Addison-Wesley, 2003.

KITCHENHAM, B. **- Evaluating software engineering methods and tool part 1**: The evaluation context and evaluation methods. ACM SIGSOFT Software Engineering Notes, v. 21 , n. 1 , p. 11 – 14, ACM Press, 1996.

KOUBARAKIS, M.; PLEXOUSAKIS, D. - **A Formal Framework for Business Process Modeling and Design**, Information Systems, 27 299-319, Elsevier, 2002.

LAKATOS, E.; MARCONI, M. **- Técnicas de Pesquisa:** Planejamento e execução de pesquisas; Amostragens e técnicas de pesquisa; Elaboração, análise e interpretação de dados. Atlas, 2006.

LARMAN, C. - **Applying UML and Patterns**: An Introduction to Object-Oriented Analysis and Design and Iterative Development. Hardcover, 2004.

LARRUCEA, X., DIEZ, A. B. G., MANSELL, J. X. **Practical Model Driven Development Process**, II European Workshop on Model Driven Architecture (EWMDA), Canterbury, Inglaterra, Setembro, 2004.

LEAL, L. N., PIRES, P. F., CAMPOS, M. L. M., DELICATO, F. C. - **Natural MDA**: Controlled Natural Language for Action Specifications on Model Driven Development, XIV International Conference on Cooperative Information System (CoopIS), França, Novembro, 2006.

MARSHALL, C. - **Enterprise Modeling with UML**. Addison-Wesley, 2000.

MARTINS, A. E. - **Em direção à captura e representação sistemática das definições dos termos das Regras de Negócio**. Dissertação (Mestrado em Informática), Núcleo de Computação Eletrônica, Instituto de Matemática, Universidade Federal do Rio de Janeiro, Rio de Janeiro, Março, 2006.

MELLOR, S.; SCOTT, K.; UHL, A.; WEISE, D. - **MDA Distilled**: Principles of Model Driven Architecture, Addison-Wesley, 2004.

MELLOR, S., SHLAER, S. - **Análise de Ambientes Orientada para Objetos**. São Paulo: MacGraw-Hill, 1990.

MORGADO, G. P. - **RÉGULA:** Uma Ferramenta para o Gerenciamento de Regras de Negócio. Projeto Final de Curso (Bacharelado em Ciência da Computação). Departamento de Ciência da Computação, Instituto de Matemática, Universidade Federal do Rio de Janeiro, Rio de Janeiro, Março, 2005.

MORGADO, G. - **RÉGULA:** Uma Ferramenta para o Gerenciamento de Regras de Negócio. Concurso de Monografias em Sistemas de Informação do III SBSI, Curitiba, Novembro, 2006.

MORGADO, G.; SILVEIRA, D.; ALENCAR, A. J.; LIMA, P.; SCHMITZ, E. - **RAPDIS**: Um Processo MDA para o Desenvolvimento de Sistemas de Informação. III Simpósio Brasileiro de Sistemas de Informação, Curitiba, Novembro, 2006.

MORGAN, T. - **Business Rules and Information Systems**. Addison-Wesley, 2002.

MUKERJI, J.; MILLER, J. - **MDA Guide**. Versão 1.0.1, OMG, Junho de 2003. Disponível em: "http://www.omg.org/docs/omg/03-06-01.pdf". Acesso em: 22 nov. 2005 18:11:00.

OMG - **Request for Proposal**: MOF 2.0 Query / Views / Transformations RFP. Abril de 2002. Disponível em: http://www.omg.org/docs/ad/02-04-10.pdf. Acesso em: 24 ago. 2006, 12:07:00.

OMG - **Unified Modeling Language**: Superstructure. Versão 2.0, agosto de 2005. Disponível em: "http://www.omg.org/docs/formal/05-07-04.pdf". Acesso em: 20 nov. 2005, 16:11:00.

OMG - **Committed Companies and Their Products**. OMG, 2007. Disponível em: "http://www.omg.org/mda/committed-products.htm". Acesso em: 17 jan. 2007, 13:55:00.

PAIS, A. O. V. - **Arquitetura de controle Hércules**: a base para a geração automática de sistemas de informação com ênfase na camada de controle, Dissertação (Mestrado em Informática), Núcleo de Computação Eletrônica, Instituto de Matemática, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2004.

PAULA W. P. - **Engenharia de Software**: fundamentos, métodos e padrões. LTC, segunda edição, 2003

PINTO, M. A. O.; SILVEIRA, D. S.; SCHMITZ, E. A. - **O Gerador Automático de Programas da Ferramenta FAST CASE**. XIV Simpósio Brasileiro de Engenharia de Software, Fortaleza, 2000.

PINTO, M. A. O. - **O Gerador Automático de Programas do FastCase**. Dissertação (Mestrado em Informática), Núcleo de Computação Eletrônica, Instituto de Matemática, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2004.

PRESSMAN, R. S. - **Software Engineering**: A Practitioner's Approach. McGraw-Hill, Quinta edição, 2001.

SANTANDER, V. F. A., CASTRO, J. F. B. - **Desenvolvendo Use Case a partir de Modelagem Organizacional**. II Workshop on Requirements Engineering, Rio de Janeiro, RJ, Brasil, 2000.

SCHMITZ, E. A.; SILVEIRA, D. S. - **Desenvolvimento de Software Orientado a Objetos**. Rio de Janeiro: Brasport, 2000.

SHAW, M.; GARLAN, D. - **Software Architecture**: Perspectives on an Emerging Discipline. New Jersey: Prentice Hall, 1996.

SHNIEDER, G., WINTERS, J. P. - **Applying Use-Case**: A practical Guide. Addison-Wesley, Segunda edição, 2001.

SILVEIRA, D. S. - **FAST CASE**: Uma Ferramenta para o Desenvolvimento Visual de Ambientes Orientados a Objetos. Dissertação (Mestrado em Informática), Núcleo de Computação Eletrônica, Instituto de Matemática, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 1999.

SILVEIRA, D.S.; SCHMITZ, E.A. - **Fast Case**: Uma Ferramenta Case para o Desenvolvimento Visual de Software Orientado a Objetos. XIII Simpósio Brasileiro em Engenharia de Software, Santa Catarina, SC, Outubro 1999.

SPARXSYSTEMS - **Ferramenta Enterprise Architect**. Disponível em: "http://www.sparxsystems.com.au". Acesso em: 5 jun. 2006, 18:52:00.

STANDISH GROUP - **CHAOS Report**. 2004. Disponí-vel em: http://www.standishgroup.com/. Acesso em: 10 jan. 2007, 16:33:00.

TARIQ, N.; AKHTER, N. - **Comparison of Model Driven Architecture (MDA) Based Tools**. Dissertação (Mestrado em Engenharia e Gerenciamento de Sistemas de Informação), Departamento de Ciência da Computação e Sistemas, Royal Institute of Tecnology, Estocolmo, Karonlinska University Hospital, Suécia, 2005.

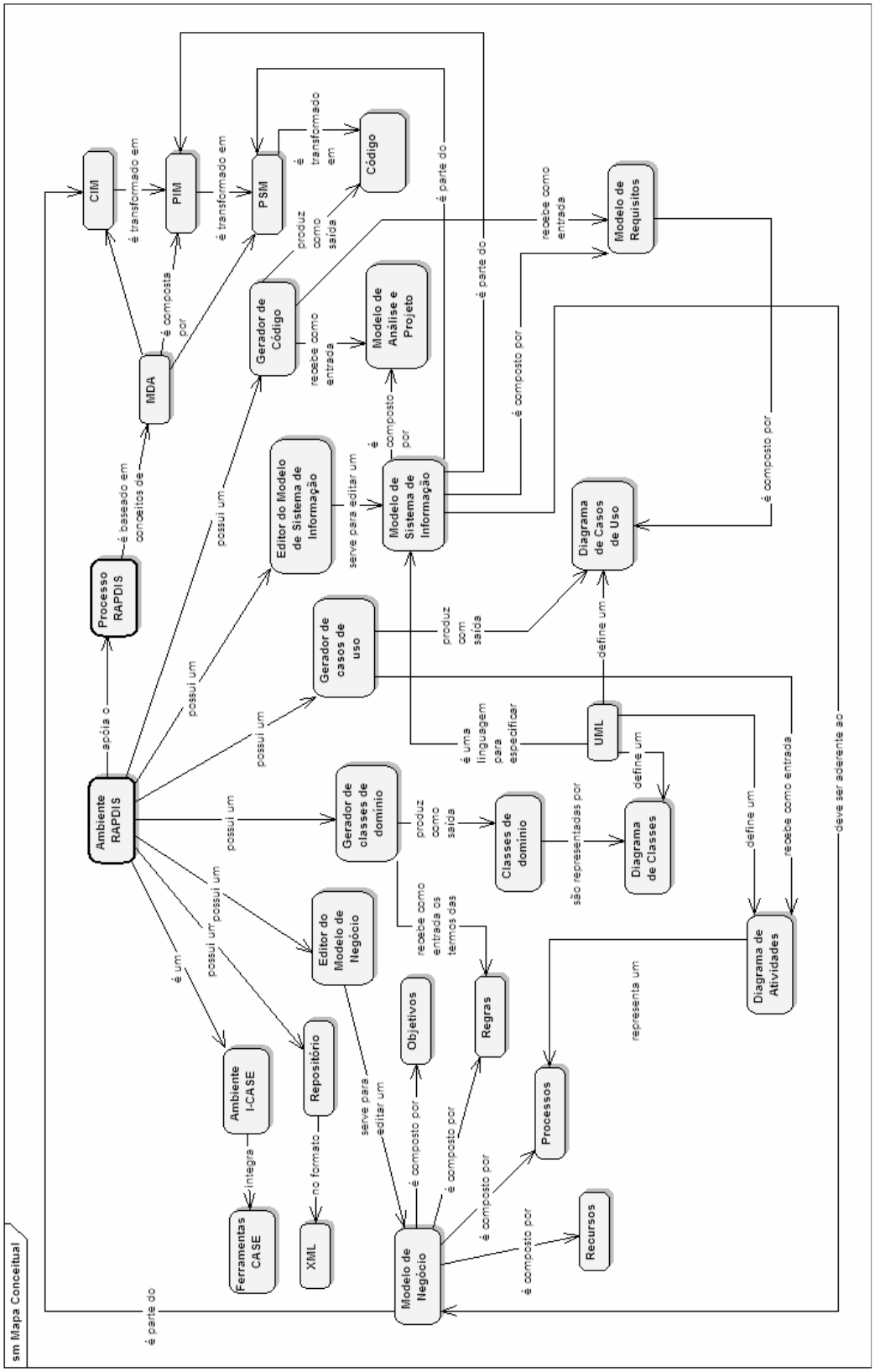
TARIQ, N.; JONSSON, S.; NABIEV, R.; TERIÖ, H.; ANDERSSON, D. - **Comparison of Model Driven Architecture (MDA) Based Tools Using Telemedicine Healthcare System**. 13th Nordic Baltic Conference (NBC), Umeå, Suécia, Junho, 2005.

WIRFS-BROCK, R.; MCKEAN, A. - **Object Design:** Roles, Responsibilities, and Collaborations, Addison-Wesley, 2003.

YERGEAU, F.; BRAY, T.; PAOLI, J.; SPERBERG-MCQUEEN, C. M.; MALER, E. - **Extensible Markup Language (XML)**. Versão 1.0, terceira edição, W3C**,** 2004. Disponível em: "http://www.w3.org/TR/REC-xml/". Acesso em: 23 nov. 2005, 18:52:00.

ZHANG, W.; MEI, H.; ZHAO, H.; YANG, J. - **Transformation from CIM to PIM**: A feature-oriented component-based approach. MoDELS: Model Driven Engineering Languages and Systems. International conference N^{o}8, Montego Bay, Jamaica, Outubro, 2005.

APÊNDICE A – MAPA CONCEITUAL



APÊNDICE B – EXEMPLO LOCADORA DE CARROS

B.1 Exemplo Transformação da Definição dos Termos em Classes

Como exemplo de transformação, apresentaremos o mapeamento dos termos de uma locadora de carros. Este exemplo é baseado no estudo de caso do BRG (*Business Rules Group*). A tabela B.1 seguir apresenta a definição dos termos em Português Estruturado. A figura B.1 mostra o diagrama de classes obtido a partir desta definição.

Tabela B.1 - Definição dos Principais Termos de uma Locadora de Carros. A definição foi construída utilizando os modelos de sentença definidos para representação das regras de negócio.

Tipo	Definição
Atributos	Cliente TEM COMO ATRIBUTO CNH, nome, endereço, idade e tempo de habilitação. Aluguel TEM COMO ATRIBUTO data início, data fim, valor da diária, CNH e placa. Carro TEM COMO ATRIBUTO placa, estacionado no pátio, limpo e tanque cheio.
Domínios dos atributos	Nome, endereço e placa POSSUEM COMO DOMÍNIO texto. CNH, idade e tempo de habilitação POSSUEM COMO DOMÍNIO número inteiro. Data início e data fim POSSUEM COMO DOMÍNIO data. Valor da diária POSSUI COMO DOMÍNIO número. Estacionado no pátio, limpo e tanque cheio POSSUEM COMO DOMÍNIO sim/não.
Associações	Cliente ESTÁ RELACIONADO A aluguel POR realiza COM GRAU um PARA muitos. Carro ESTÁ RELACIONADO A aluguel POR utiliza COM GRAU um PARA muitos.

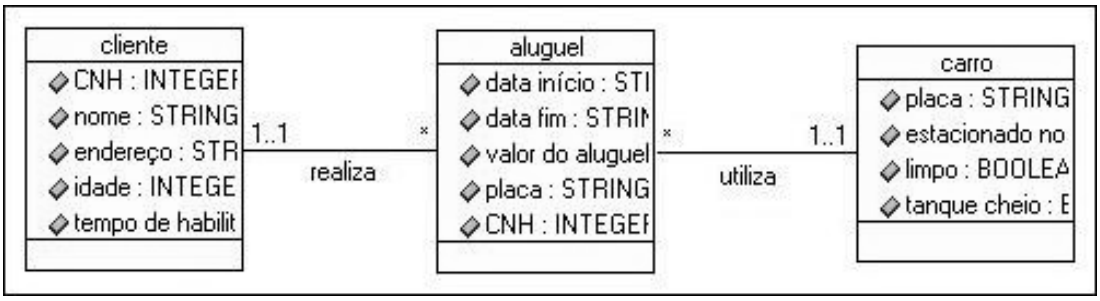


Figura B.1 - Diagrama de Classes Gerado Automaticamente a Partir da Definição dos Termos. Os termos cliente, aluguel e carro foram mapeados como classes. Os demais termos foram mapeados como atributos destas classes.

B.2 Exemplo de Transformação dos Processos em Casos de Uso

Dando continuidade ao exemplo da locadora de carros apresentado na seção anterior, mostraremos a aplicação destas heurísticas em um processo de aluguel de carros. Este processo é representado através do diagrama de atividades da figura B.2. Após a aplicação das heurísticas o processo é transformado automaticamente no diagrama de casos de uso da figura B.3.

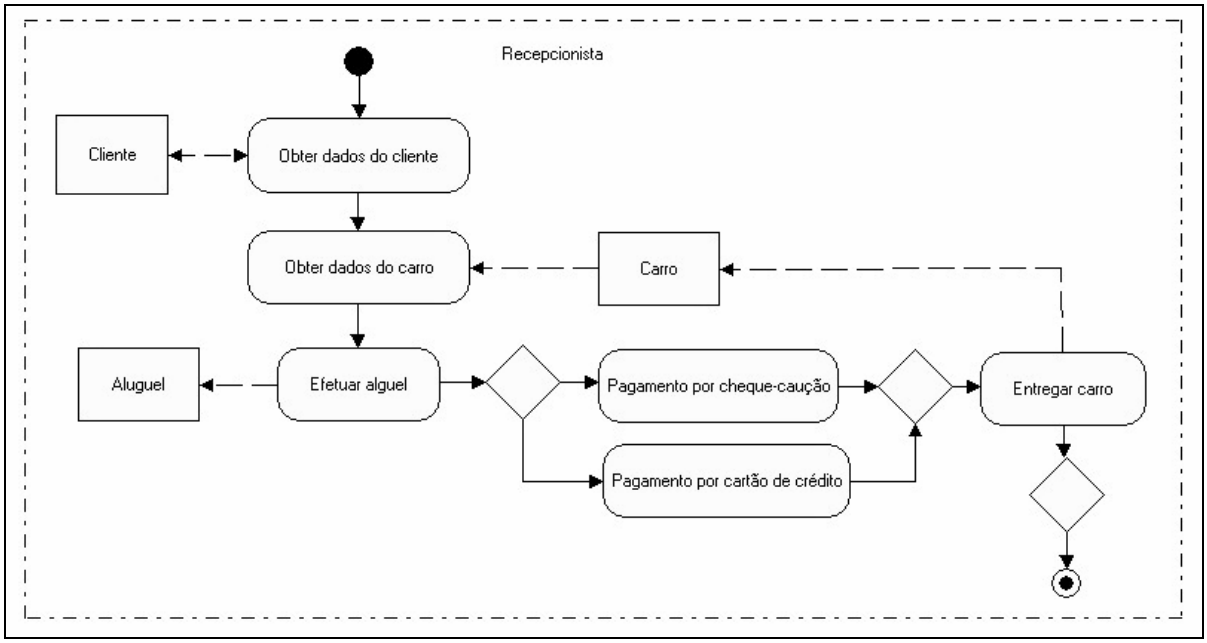


Figura B.2 - Processo de Aluguel de Carros. A representação do processo foi feita através de um diagrama de atividades.

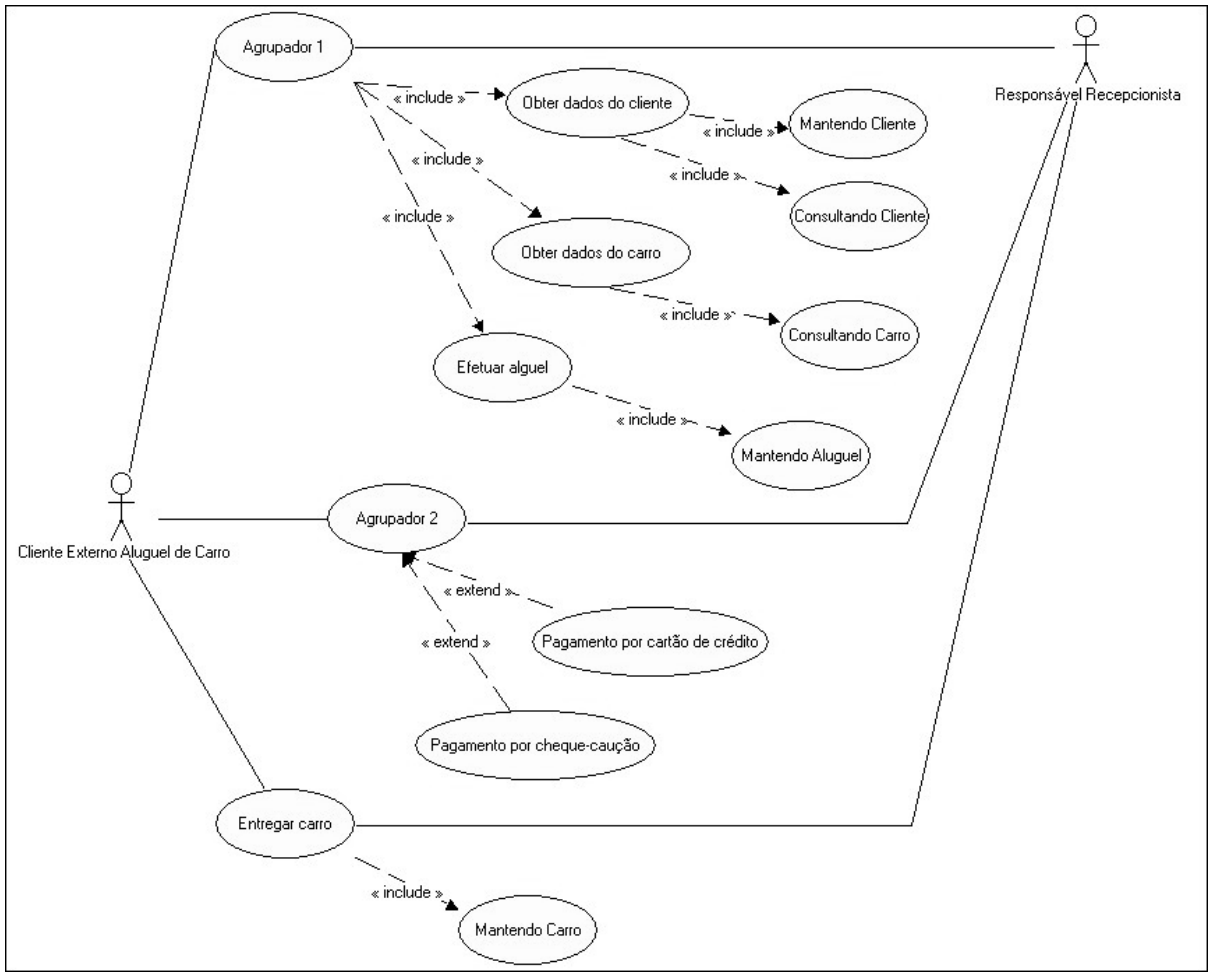


Figura B.3 - Diagrama de Casos de Uso Gerado Automaticamente a Partir do Processo. As atividades do processo foram transformadas em casos de uso.

APÊNDICE C – ESQUEMA DO REPOSITÓRIO DO RAPDIS

C.1 NomeDoProjeto.XSD

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
  <xs:element name="project">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="name"/>
        <xs:element ref="tool"/>
        <xs:element ref="version"/>
        <xs:element ref="date"/>
        <xs:element ref="time"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="name" type="xs:NCName"/>
  <xs:element name="tool" type="xs:NCName"/>
  <xs:element name="version" type="xs:decimal"/>
  <xs:element name="date" type="xs:string"/>
  <xs:element name="time" type="xs:NMTOKEN"/>
</xs:schema>
```

C.2 Business.XSD

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
  <xs:element name="business">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="business.head"/>
        <xs:element ref="business.content"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="business.head">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="project"/>
        <xs:element ref="tool"/>
        <xs:element ref="version"/>
        <xs:element ref="date"/>
        <xs:element ref="time"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="project" type="xs:NCName"/>
  <xs:element name="tool" type="xs:NCName"/>
  <xs:element name="version" type="xs:decimal"/>
  <xs:element name="date" type="xs:string"/>
  <xs:element name="time" type="xs:NMTOKEN"/>
```

```
  <xs:element name="business.content">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="name"/>
        <xs:element ref="description"/>
        <xs:element ref="qtyGoals"/>
        <xs:element ref="qtyQuantifiedgoals"/>
        <xs:element ref="qtyResources"/>
        <xs:element ref="qtyRules"/>
        <xs:element ref="qtyTerms"/>
        <xs:element ref="qtyFacts"/>
        <xs:element ref="qtyProcesses"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="name" type="xs:NCName"/>
  <xs:element name="description">
    <xs:complexType/>
  <xs:element name="qtyGoals" type="xs:integer"/>
  <xs:element name="qtyQuantifiedgoals" type="xs:integer"/>
  <xs:element name="qtyResources" type="xs:integer"/>
  <xs:element name="qtyRules" type="xs:integer"/>
  <xs:element name="qtyTerms" type="xs:integer"/>
  <xs:element name="qtyFacts" type="xs:integer"/>
  <xs:element name="qtyProcesses" type="xs:integer"/>
</xs:schema>
```

C.3 Processes.XSD

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
  <xs:element name="processes">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="processes.head"/>
        <xs:element ref="processes.content"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="processes.head">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="project"/>
        <xs:element ref="tool"/>
        <xs:element ref="version"/>
        <xs:element ref="date"/>
        <xs:element ref="time"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="project" type="xs:string"/>
  <xs:element name="tool" type="xs:NCName"/>
  <xs:element name="version" type="xs:decimal"/>
  <xs:element name="date" type="xs:string"/>
```

```
<xs:element name="time" type="xs:NMTOKEN"/>
<xs:element name="processes.content">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="diagrams"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="diagrams">
  <xs:complexType>
    <xs:sequence>
      <xs:element maxOccurs="unbounded" ref="diagram"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="diagram">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="name"/>
      <xs:element ref="elements"/>
      <xs:element ref="links"/>
    </xs:sequence>
    <xs:attribute name="id" use="required" type="xs:integer"/>
  </xs:complexType>
</xs:element>
<xs:element name="links">
  <xs:complexType>
    <xs:sequence>
      <xs:element maxOccurs="unbounded" ref="link"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="link">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="name"/>
      <xs:element ref="breaks"/>
    </xs:sequence>
    <xs:attribute name="id" use="required" type="xs:integer"/>
    <xs:attribute name="idSource" use="required" type="xs:integer"/>
    <xs:attribute name="idTarget" use="required" type="xs:integer"/>
    <xs:attribute name="qtyBreaks" use="required" type="xs:integer"/>
    <xs:attribute name="removed" use="required" type="xs:boolean"/>
    <xs:attribute name="type" use="required" type="xs:NCName"/>
    <xs:attribute name="xText" use="required" type="xs:integer"/>
    <xs:attribute name="yText" use="required" type="xs:integer"/>
  </xs:complexType>
</xs:element>
<xs:element name="breaks">
  <xs:complexType>
    <xs:sequence>
      <xs:element maxOccurs="unbounded" ref="break"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="break">
  <xs:complexType>
```

```
      <xs:attribute name="x" use="required" type="xs:integer"/>
      <xs:attribute name="y" use="required" type="xs:integer"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="name" type="xs:string"/>
  <xs:element name="elements">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="0" maxOccurs="unbounded" ref="element"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="element">
    <xs:complexType>
      <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element ref="elements"/>
        <xs:element ref="name"/>
      </xs:choice>
      <xs:attribute name="idExternal" use="required" type="xs:integer"/>
      <xs:attribute name="idInternal" use="required" type="xs:integer"/>
      <xs:attribute name="objectType"/>
      <xs:attribute name="removed" use="required" type="xs:boolean"/>
      <xs:attribute name="type" use="required" type="xs:NCName"/>
      <xs:attribute name="xBeginning" use="required" type="xs:integer"/>
      <xs:attribute name="xEnd" use="required" type="xs:integer"/>
      <xs:attribute name="yBeginning" use="required" type="xs:integer"/>
      <xs:attribute name="yEnd" use="required" type="xs:integer"/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

C.4 Goals.XSD

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
  <xs:element name="goals">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="goals.head"/>
        <xs:element ref="goals.content"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="goals.head">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="project"/>
        <xs:element ref="tool"/>
        <xs:element ref="version"/>
        <xs:element ref="date"/>
        <xs:element ref="time"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
```

```
  <xs:element name="project" type="xs:NCName"/>
  <xs:element name="tool" type="xs:NCName"/>
  <xs:element name="version" type="xs:decimal"/>
  <xs:element name="time" type="xs:NMTOKEN"/>
  <xs:element name="goals.content">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="goal"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="goal">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="name"/>
        <xs:element ref="description"/>
        <xs:element ref="quantifiedgoals"/>
      </xs:sequence>
      <xs:attribute name="id" use="required" type="xs:integer"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="quantifiedgoals">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="quantifiedgoal"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="quantifiedgoal">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="name"/>
        <xs:element ref="description"/>
        <xs:element ref="date"/>
      </xs:sequence>
      <xs:attribute name="term" use="required" type="xs:integer"/>
      <xs:attribute name="value" use="required" type="xs:integer"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="date" type="xs:string"/>
  <xs:element name="name" type="xs:string"/>
  <xs:element name="description" type="xs:string"/>
</xs:schema>
```

C.5 Resources.XSD

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
  <xs:element name="resources">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="resources.head"/>
        <xs:element ref="resources.content"/>
      </xs:sequence>
```

```
    </xs:complexType>
  </xs:element>
  <xs:element name="resources.head">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="project"/>
        <xs:element ref="tool"/>
        <xs:element ref="version"/>
        <xs:element ref="date"/>
        <xs:element ref="time"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="project" type="xs:NCName"/>
  <xs:element name="tool" type="xs:NCName"/>
  <xs:element name="version" type="xs:decimal"/>
  <xs:element name="date" type="xs:string"/>
  <xs:element name="time" type="xs:NMTOKEN"/>
  <xs:element name="resources.content">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="resource"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="resource">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="description"/>
      </xs:sequence>
      <xs:attribute name="id" use="required" type="xs:integer"/>
      <xs:attribute name="superior" use="required" type="xs:integer"/>
      <xs:attribute name="term" use="required" type="xs:integer"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="description" type="xs:string"/>
</xs:schema>
```

C.6 Terms.XSD

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
  <xs:element name="terms">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="terms.head"/>
        <xs:element ref="terms.content"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="terms.head">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="project"/>
```

```
      <xs:element ref="tool"/>
      <xs:element ref="version"/>
      <xs:element ref="date"/>
      <xs:element ref="time"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="project" type="xs:NCName"/>
<xs:element name="tool" type="xs:NCName"/>
<xs:element name="version" type="xs:decimal"/>
<xs:element name="date" type="xs:string"/>
<xs:element name="time" type="xs:NMTOKEN"/>
<xs:element name="terms.content">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="substantives"/>
      <xs:element ref="verbs"/>
      <xs:element ref="synonyms"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="substantives">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="bases"/>
      <xs:element ref="types"/>
      <xs:element ref="others"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="bases">
  <xs:complexType>
    <xs:sequence>
      <xs:element maxOccurs="unbounded" ref="term"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="types">
  <xs:complexType>
    <xs:sequence>
      <xs:element maxOccurs="unbounded" ref="term"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="others">
  <xs:complexType>
    <xs:sequence>
      <xs:element maxOccurs="unbounded" ref="term"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="verbs">
  <xs:complexType>
    <xs:sequence>
      <xs:element maxOccurs="unbounded" ref="term"/>
    </xs:sequence>
  </xs:complexType>
```

```
  </xs:element>
  <xs:element name="synonyms">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="term"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="term">
    <xs:complexType mixed="true">
      <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element ref="name"/>
        <xs:element ref="values"/>
      </xs:choice>
      <xs:attribute name="id" use="required" type="xs:integer"/>
      <xs:attribute name="subtype" type="xs:integer"/>
      <xs:attribute name="term" type="xs:integer"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="name" type="xs:string"/>
  <xs:element name="values">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="0" maxOccurs="unbounded" ref="value"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="value" type="xs:string"/>
</xs:schema>
```

C.7 Rules.XSD

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
  <xs:element name="rules">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="rules.head"/>
        <xs:element ref="rules.content"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="rules.head">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="project"/>
        <xs:element ref="tool"/>
        <xs:element ref="version"/>
        <xs:element ref="date"/>
        <xs:element ref="time"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="project" type="xs:NCName"/>
```

```
<xs:element name="tool" type="xs:NCName"/>
<xs:element name="version" type="xs:decimal"/>
<xs:element name="date" type="xs:string"/>
<xs:element name="time" type="xs:NMTOKEN"/>
<xs:element name="rules.content">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="facts"/>
      <xs:element ref="calculus"/>
      <xs:element ref="derivation"/>
      <xs:element ref="actionassertions"/>
      <xs:element ref="permissions"/>
      <xs:element ref="obligations"/>
      <xs:element ref="prohibitions"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="facts">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="supertypes"/>
      <xs:element ref="parts"/>
      <xs:element ref="attributes"/>
      <xs:element ref="associations"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="supertypes">
  <xs:complexType>
    <xs:sequence>
      <xs:element maxOccurs="unbounded" ref="rule"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="parts">
  <xs:complexType>
    <xs:sequence>
      <xs:element maxOccurs="unbounded" ref="rule"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="attributes">
  <xs:complexType>
    <xs:sequence>
      <xs:element maxOccurs="unbounded" ref="rule"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="associations">
  <xs:complexType>
    <xs:sequence>
      <xs:element maxOccurs="unbounded" ref="rule"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="calculus">
  <xs:complexType>
```

```
    <xs:sequence>
      <xs:element maxOccurs="unbounded" ref="rule"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="derivation">
  <xs:complexType>
    <xs:sequence>
      <xs:element maxOccurs="unbounded" ref="rule"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="actionassertions">
  <xs:complexType/>
</xs:element>
<xs:element name="permissions">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="simple"/>
      <xs:element ref="limited"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="obligations">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="simple"/>
      <xs:element ref="limited"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="prohibitions">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="simple"/>
      <xs:element ref="limited"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="rule">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="sentence"/>
      <xs:sequence minOccurs="0">
        <xs:element ref="source"/>
        <xs:element ref="notes"/>
      </xs:sequence>
    </xs:sequence>
    <xs:attribute name="active"/>
    <xs:attribute name="automatized"/>
    <xs:attribute name="definedby"/>
    <xs:attribute name="goal"/>
    <xs:attribute name="id" use="required" type="xs:integer"/>
    <xs:attribute name="responsible"/>
  </xs:complexType>
</xs:element>
<xs:element name="sentence">
```

```
  <xs:complexType>
    <xs:choice minOccurs="0">
      <xs:element ref="function"/>
      <xs:element ref="restrictions"/>
      <xs:sequence>
        <xs:element ref="conditions"/>
        <xs:element ref="state"/>
      </xs:sequence>
    </xs:choice>
    <xs:attribute name="article"/>
    <xs:attribute name="basic" type="xs:NCName"/>
    <xs:attribute name="cardinality1" type="xs:NCName"/>
    <xs:attribute name="cardinality2" type="xs:NCName"/>
    <xs:attribute name="comparative"/>
    <xs:attribute name="identificator" type="xs:NCName"/>
    <xs:attribute name="order" type="xs:integer"/>
    <xs:attribute name="preferential" type="xs:NCName"/>
    <xs:attribute name="preposition"/>
    <xs:attribute name="term" type="xs:integer"/>
    <xs:attribute name="term1" type="xs:integer"/>
    <xs:attribute name="term2" type="xs:integer"/>
    <xs:attribute name="type" type="xs:integer"/>
    <xs:attribute name="value" type="xs:integer"/>
    <xs:attribute name="verb" type="xs:integer"/>
  </xs:complexType>
</xs:element>
<xs:element name="function">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="name"/>
      <xs:element ref="parameters"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="name" type="xs:NCName"/>
<xs:element name="parameters">
  <xs:complexType>
    <xs:sequence>
      <xs:element maxOccurs="unbounded" ref="parameter"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="parameter">
  <xs:complexType>
    <xs:attribute name="term" use="required" type="xs:integer"/>
  </xs:complexType>
</xs:element>
<xs:element name="restrictions">
  <xs:complexType>
    <xs:sequence>
      <xs:element minOccurs="0" ref="restriction"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="restriction">
  <xs:complexType>
    <xs:attribute name="term" use="required" type="xs:integer"/>
```

```
      <xs:attribute name="verb" use="required" type="xs:integer"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="conditions">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="condition"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="condition">
    <xs:complexType>
      <xs:attribute name="operator" use="required"/>
      <xs:attribute name="order" use="required" type="xs:integer"/>
      <xs:attribute name="term1" use="required" type="xs:integer"/>
      <xs:attribute name="type" use="required"/>
      <xs:attribute name="value" use="required" type="xs:NMTOKEN"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="state" type="xs:NCName"/>
  <xs:element name="source">
    <xs:complexType/>
  </xs:element>
  <xs:element name="notes" type="xs:string"/>
  <xs:element name="simple">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="rule"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="limited">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="0" ref="rule"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

C.8 IS.XSD

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
  <xs:element name="is">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="is.head"/>
        <xs:element ref="is.content"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="is.head">
    <xs:complexType>
```

```
      <xs:sequence>
        <xs:element ref="project"/>
        <xs:element ref="tool"/>
        <xs:element ref="version"/>
        <xs:element ref="date"/>
        <xs:element ref="time"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="project" type="xs:string"/>
  <xs:element name="tool" type="xs:NCName"/>
  <xs:element name="version" type="xs:decimal"/>
  <xs:element name="date" type="xs:string"/>
  <xs:element name="time" type="xs:NMTOKEN"/>
  <xs:element name="is.content">
    <xs:complexType>
      <xs:complexContent>
        <xs:extension base="name">
          <xs:sequence>
            <xs:element ref="scope"/>
            <xs:element ref="qtyUseCases"/>
            <xs:element ref="qtyClasses"/>
            <xs:element ref="qtySequences"/>
            <xs:element ref="qtyStates"/>
            <xs:element ref="qtyProcesses"/>
            <xs:element ref="qtyElements"/>
            <xs:element ref="artifacts"/>
            <xs:element ref="diagrams"/>
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="scope">
    <xs:complexType/>
  </xs:element>
  <xs:element name="qtyUseCases" type="xs:integer"/>
  <xs:element name="qtyClasses" type="xs:integer"/>
  <xs:element name="qtySequences" type="xs:integer"/>
  <xs:element name="qtyStates" type="xs:integer"/>
  <xs:element name="qtyProcesses" type="xs:integer"/>
  <xs:element name="qtyElements" type="xs:integer"/>
  <xs:element name="artifacts">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="artifact"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="artifact">
    <xs:complexType>
      <xs:complexContent>
        <xs:extension base="name">
          <xs:attribute name="assocObj" use="required"
type="xs:integer"/>
          <xs:attribute name="class" use="required" type="xs:NCName"/>
```

```
          <xs:attribute name="idInternal" use="required"
type="xs:integer"/>
          <xs:attribute name="qty" use="required" type="xs:integer"/>
          <xs:attribute name="qtyAssoc" use="required"
type="xs:integer"/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="diagrams">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="usecases"/>
        <xs:element ref="classes"/>
        <xs:element ref="sequences"/>
        <xs:element ref="states"/>
        <xs:element ref="processes"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="usecases">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="diagram"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="classes">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="diagram"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="sequences">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="diagram"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="states">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="diagram"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="processes">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="diagram"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:complexType name="name">
    <xs:sequence>
```

```
      <xs:element ref="name"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="name" type="xs:string"/>
  <xs:element name="diagram">
    <xs:complexType>
      <xs:complexContent>
        <xs:extension base="name">
          <xs:sequence>
            <xs:element minOccurs="0" ref="classAssociation"/>
          </xs:sequence>
          <xs:attribute name="id" use="required" type="xs:integer"/>
          <xs:attribute name="idAssociation" type="xs:integer"/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="classAssociation" type="xs:NCName"/>
</xs:schema>
```

C.9 UseCases.XSD

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
  <xs:element name="usecases">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="usecases.head"/>
        <xs:element ref="usecases.content"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="usecases.head">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="project"/>
        <xs:element ref="tool"/>
        <xs:element ref="version"/>
        <xs:element ref="date"/>
        <xs:element ref="time"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="project" type="xs:string"/>
  <xs:element name="tool" type="xs:NCName"/>
  <xs:element name="version" type="xs:decimal"/>
  <xs:element name="date" type="xs:string"/>
  <xs:element name="time" type="xs:NMTOKEN"/>
  <xs:element name="usecases.content">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="diagrams"/>
      </xs:sequence>
    </xs:complexType>
```

```
  </xs:element>
  <xs:element name="diagrams">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="diagram"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="diagram">
    <xs:complexType>
      <xs:complexContent>
        <xs:extension base="name">
          <xs:sequence>
            <xs:element ref="elements"/>
            <xs:element ref="links"/>
          </xs:sequence>
          <xs:attribute name="id" use="required" type="xs:integer"/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="elements">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="element"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="element">
    <xs:complexType>
      <xs:complexContent>
        <xs:extension base="name">
          <xs:attribute name="idExternal" use="required"
type="xs:integer"/>
          <xs:attribute name="idInternal" use="required"
type="xs:integer"/>
          <xs:attribute name="removed" use="required" type="xs:boolean"/>
          <xs:attribute name="type" use="required" type="xs:NCName"/>
          <xs:attribute name="xBeginning" use="required"
type="xs:integer"/>
          <xs:attribute name="xEnd" use="required" type="xs:integer"/>
          <xs:attribute name="yBeginning" use="required"
type="xs:integer"/>
          <xs:attribute name="yEnd" use="required" type="xs:integer"/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="links">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="link"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="link">
    <xs:complexType>
```

```
      <xs:complexContent>
        <xs:extension base="name">
          <xs:sequence>
            <xs:element ref="breaks"/>
          </xs:sequence>
          <xs:attribute name="id" use="required" type="xs:integer"/>
          <xs:attribute name="idSource" use="required"
type="xs:integer"/>
          <xs:attribute name="idTarget" use="required"
type="xs:integer"/>
          <xs:attribute name="qtyBreaks" use="required"
type="xs:integer"/>
          <xs:attribute name="removed" use="required" type="xs:boolean"/>
          <xs:attribute name="type" use="required" type="xs:NCName"/>
          <xs:attribute name="xText" use="required" type="xs:integer"/>
          <xs:attribute name="yText" use="required" type="xs:integer"/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="breaks">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="break"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="break">
    <xs:complexType>
      <xs:attribute name="x" use="required" type="xs:integer"/>
      <xs:attribute name="y" use="required" type="xs:integer"/>
    </xs:complexType>
  </xs:element>
  <xs:complexType name="name">
    <xs:sequence>
      <xs:element ref="name"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="name" type="xs:string"/>
</xs:schema>
```

C.10 Classes.XSD

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
  <xs:element name="classes">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="domainclasses.head"/>
        <xs:element ref="classes.content"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="domainclasses.head">
```

```
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="project"/>
        <xs:element ref="tool"/>
        <xs:element ref="version"/>
        <xs:element ref="date"/>
        <xs:element ref="time"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="project" type="xs:string"/>
  <xs:element name="tool" type="xs:NCName"/>
  <xs:element name="version" type="xs:decimal"/>
  <xs:element name="date" type="xs:string"/>
  <xs:element name="time" type="xs:NMTOKEN"/>
  <xs:element name="classes.content">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="diagrams"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="diagrams">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="diagram"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="diagram">
    <xs:complexType>
      <xs:complexContent>
        <xs:extension base="name">
          <xs:sequence>
            <xs:element ref="elements"/>
            <xs:element ref="links"/>
          </xs:sequence>
          <xs:attribute name="id" use="required" type="xs:integer"/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="elements">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="element"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="element">
    <xs:complexType>
      <xs:complexContent>
        <xs:extension base="name">
          <xs:attribute name="MostraAtributo" use="required"
type="xs:integer"/>
          <xs:attribute name="MostraOperacao" use="required"
type="xs:integer"/>
```

```
          <xs:attribute name="idExternal" use="required"
type="xs:integer"/>
          <xs:attribute name="idInternal" use="required"
type="xs:integer"/>
          <xs:attribute name="removed" use="required" type="xs:boolean"/>
          <xs:attribute name="type" use="required" type="xs:NCName"/>
          <xs:attribute name="xBeginning" use="required"
type="xs:integer"/>
          <xs:attribute name="xEnd" use="required" type="xs:integer"/>
          <xs:attribute name="yBeginning" use="required"
type="xs:integer"/>
          <xs:attribute name="yEnd" use="required" type="xs:integer"/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="links">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="0" maxOccurs="unbounded" ref="link"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="link">
    <xs:complexType>
      <xs:complexContent>
        <xs:extension base="name">
          <xs:sequence>
            <xs:element ref="breaks"/>
          </xs:sequence>
          <xs:attribute name="cardDestination" use="required"/>
          <xs:attribute name="cardOrigin" use="required"/>
          <xs:attribute name="id" use="required" type="xs:integer"/>
          <xs:attribute name="idSource" use="required"
type="xs:integer"/>
          <xs:attribute name="idTarget" use="required"
type="xs:integer"/>
          <xs:attribute name="qtyBreaks" use="required"
type="xs:integer"/>
          <xs:attribute name="removed" use="required" type="xs:boolean"/>
          <xs:attribute name="type" use="required" type="xs:NCName"/>
          <xs:attribute name="xCardDestination" use="required"
type="xs:integer"/>
          <xs:attribute name="xCardOrigin" use="required"
type="xs:integer"/>
          <xs:attribute name="xText" use="required" type="xs:integer"/>
          <xs:attribute name="yCardDestination" use="required"
type="xs:integer"/>
          <xs:attribute name="yCardOrigin" use="required"
type="xs:integer"/>
          <xs:attribute name="yText" use="required" type="xs:integer"/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="breaks">
    <xs:complexType>
```

```
      <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="break"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="break">
    <xs:complexType>
      <xs:attribute name="x" use="required" type="xs:integer"/>
      <xs:attribute name="y" use="required" type="xs:integer"/>
    </xs:complexType>
  </xs:element>
  <xs:complexType name="name">
    <xs:sequence>
      <xs:element ref="name"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="name" type="xs:string"/>
</xs:schema>
```

C.11 Sequences.XSD

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
  <xs:element name="sequences">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="sequences.head"/>
        <xs:element ref="sequences.content"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="sequences.head">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="project"/>
        <xs:element ref="tool"/>
        <xs:element ref="version"/>
        <xs:element ref="date"/>
        <xs:element ref="time"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="project" type="xs:string"/>
  <xs:element name="tool" type="xs:NCName"/>
  <xs:element name="version" type="xs:decimal"/>
  <xs:element name="date" type="xs:string"/>
  <xs:element name="time" type="xs:NMTOKEN"/>
  <xs:element name="sequences.content">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="diagrams"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
```

```
  <xs:element name="diagrams">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="diagram"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="diagram">
    <xs:complexType>
      <xs:complexContent>
        <xs:extension base="name">
          <xs:sequence>
            <xs:element ref="elements"/>
            <xs:element ref="links"/>
          </xs:sequence>
          <xs:attribute name="id" use="required" type="xs:integer"/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="elements">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="element"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="element">
    <xs:complexType>
      <xs:complexContent>
        <xs:extension base="name">
          <xs:attribute name="idExternal" use="required"
type="xs:integer"/>
          <xs:attribute name="idInternal" use="required"
type="xs:integer"/>
          <xs:attribute name="removed" use="required" type="xs:boolean"/>
          <xs:attribute name="type" use="required" type="xs:NCName"/>
          <xs:attribute name="xBeginning" use="required"
type="xs:integer"/>
          <xs:attribute name="xEnd" use="required" type="xs:integer"/>
          <xs:attribute name="xLifeLine" use="required"
type="xs:integer"/>
          <xs:attribute name="yBeginning" use="required"
type="xs:integer"/>
          <xs:attribute name="yEnd" use="required" type="xs:integer"/>
          <xs:attribute name="yLifeLine" use="required"
type="xs:integer"/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="links">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="link"/>
      </xs:sequence>
    </xs:complexType>
```

```
  </xs:element>
  <xs:element name="link">
    <xs:complexType>
      <xs:complexContent>
        <xs:extension base="name">
          <xs:sequence>
            <xs:element ref="breaks"/>
          </xs:sequence>
          <xs:attribute name="id" use="required" type="xs:integer"/>
          <xs:attribute name="idSource" use="required"
type="xs:integer"/>
          <xs:attribute name="idTarget" use="required"
type="xs:integer"/>
          <xs:attribute name="linkType" use="required"
type="xs:integer"/>
          <xs:attribute name="qtyBreaks" use="required"
type="xs:integer"/>
          <xs:attribute name="removed" use="required" type="xs:boolean"/>
          <xs:attribute name="type" use="required" type="xs:NCName"/>
          <xs:attribute name="xText" use="required" type="xs:integer"/>
          <xs:attribute name="yText" use="required" type="xs:integer"/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="breaks">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="break"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="break">
    <xs:complexType>
      <xs:attribute name="x" use="required" type="xs:integer"/>
      <xs:attribute name="y" use="required" type="xs:integer"/>
    </xs:complexType>
  </xs:element>
  <xs:complexType name="name">
    <xs:sequence>
      <xs:element ref="name"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="name" type="xs:string"/>
</xs:schema>
```

C.12 States.XSD

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
  <xs:element name="states">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="states.head"/>
```

```
      <xs:element ref="states.content"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="states.head">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="project"/>
      <xs:element ref="tool"/>
      <xs:element ref="version"/>
      <xs:element ref="date"/>
      <xs:element ref="time"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="project" type="xs:string"/>
<xs:element name="tool" type="xs:NCName"/>
<xs:element name="version" type="xs:decimal"/>
<xs:element name="date" type="xs:string"/>
<xs:element name="time" type="xs:NMTOKEN"/>
<xs:element name="states.content">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="diagrams"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="diagrams">
  <xs:complexType>
    <xs:sequence>
      <xs:element maxOccurs="unbounded" ref="diagram"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="diagram">
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base="name">
        <xs:sequence>
          <xs:element ref="elements"/>
          <xs:element ref="links"/>
        </xs:sequence>
        <xs:attribute name="id" use="required" type="xs:integer"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
<xs:element name="elements">
  <xs:complexType>
    <xs:sequence>
      <xs:element maxOccurs="unbounded" ref="element"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="element">
  <xs:complexType>
    <xs:complexContent>
```

```
        <xs:extension base="name">
          <xs:attribute name="idExternal" use="required"
type="xs:integer"/>
          <xs:attribute name="idInternal" use="required"
type="xs:integer"/>
          <xs:attribute name="removed" use="required" type="xs:boolean"/>
          <xs:attribute name="type" use="required" type="xs:NCName"/>
          <xs:attribute name="xBeginning" use="required"
type="xs:integer"/>
          <xs:attribute name="xEnd" use="required" type="xs:integer"/>
          <xs:attribute name="yBeginning" use="required"
type="xs:integer"/>
          <xs:attribute name="yEnd" use="required" type="xs:integer"/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="links">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="link"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="link">
    <xs:complexType>
      <xs:complexContent>
        <xs:extension base="name">
          <xs:sequence>
            <xs:element ref="breaks"/>
          </xs:sequence>
          <xs:attribute name="id" use="required" type="xs:integer"/>
          <xs:attribute name="idSource" use="required"
type="xs:integer"/>
          <xs:attribute name="idTarget" use="required"
type="xs:integer"/>
          <xs:attribute name="qtyBreaks" use="required"
type="xs:integer"/>
          <xs:attribute name="removed" use="required" type="xs:boolean"/>
          <xs:attribute name="type" use="required" type="xs:NCName"/>
          <xs:attribute name="xText" use="required" type="xs:integer"/>
          <xs:attribute name="yText" use="required" type="xs:integer"/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="breaks">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="break"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="break">
    <xs:complexType>
      <xs:attribute name="x" use="required" type="xs:integer"/>
      <xs:attribute name="y" use="required" type="xs:integer"/>
```

```
    </xs:complexType>
  </xs:element>
  <xs:complexType name="name">
    <xs:sequence>
      <xs:element ref="name"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="name" type="xs:string"/>
</xs:schema>
```

C.13 Definitions.XSD

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
  <xs:element name="definitions">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="definitions.head"/>
        <xs:element ref="definitions.content"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="definitions.head">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="project"/>
        <xs:element ref="tool"/>
        <xs:element ref="version"/>
        <xs:element ref="date"/>
        <xs:element ref="time"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="project" type="xs:string"/>
  <xs:element name="tool" type="xs:NCName"/>
  <xs:element name="version" type="xs:decimal"/>
  <xs:element name="date" type="xs:string"/>
  <xs:element name="time" type="xs:NMTOKEN"/>
  <xs:element name="definitions.content">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="activities"/>
        <xs:element ref="signals"/>
        <xs:element ref="objects"/>
        <xs:element ref="actors"/>
        <xs:element ref="usecases"/>
        <xs:element ref="classes"/>
        <xs:element ref="associations"/>
        <xs:element ref="states"/>
        <xs:element ref="transitions"/>
        <xs:element ref="swimlanes"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
```

```
<xs:element name="activities">
  <xs:complexType>
    <xs:sequence>
      <xs:element maxOccurs="unbounded" ref="activity"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="activity">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="description"/>
      <xs:sequence minOccurs="0">
        <xs:element ref="precondition"/>
        <xs:element ref="postcondition"/>
      </xs:sequence>
    </xs:sequence>
    <xs:attribute name="idElement" use="required" type="xs:integer"/>
  </xs:complexType>
</xs:element>
<xs:element name="postcondition">
  <xs:complexType/>
</xs:element>
<xs:element name="signals">
  <xs:complexType/>
</xs:element>
<xs:element name="objects">
  <xs:complexType>
    <xs:sequence>
      <xs:element maxOccurs="unbounded" ref="object"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="object">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="description"/>
    </xs:sequence>
    <xs:attribute name="idElement" use="required" type="xs:integer"/>
  </xs:complexType>
</xs:element>
<xs:element name="usecases">
  <xs:complexType>
    <xs:sequence>
      <xs:element maxOccurs="unbounded" ref="usecase"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="usecase">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="actors"/>
      <xs:element ref="scenario"/>
      <xs:element ref="precondition"/>
      <xs:element ref="poscondition"/>
      <xs:element ref="exception"/>
      <xs:element ref="extension"/>
    </xs:sequence>
```

```
      <xs:attribute name="idElement" use="required" type="xs:integer"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="scenario" type="xs:string"/>
  <xs:element name="poscondition" type="xs:string"/>
  <xs:element name="exception" type="xs:string"/>
  <xs:element name="extension">
    <xs:complexType/>
  </xs:element>
  <xs:element name="classes">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="class"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="class">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="heranca"/>
        <xs:element ref="classtype"/>
        <xs:element ref="description"/>
        <xs:element ref="atributes"/>
        <xs:element ref="methods"/>
      </xs:sequence>
      <xs:attribute name="idElement" use="required" type="xs:integer"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="heranca" type="xs:NCName"/>
  <xs:element name="classtype" type="xs:integer"/>
  <xs:element name="atributes">
    <xs:complexType>
      <xs:choice>
        <xs:element maxOccurs="unbounded" ref="atribute"/>
        <xs:element maxOccurs="unbounded" ref="attribute"/>
      </xs:choice>
    </xs:complexType>
  </xs:element>
  <xs:element name="atribute">
    <xs:complexType>
      <xs:complexContent>
        <xs:extension base="name">
          <xs:attribute name="type" use="required" type="xs:NCName"/>
          <xs:attribute name="visibility" use="required"
type="xs:integer"/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="attribute">
    <xs:complexType>
      <xs:complexContent>
        <xs:extension base="name">
          <xs:attribute name="type" use="required" type="xs:NCName"/>
          <xs:attribute name="visibility" use="required"
type="xs:integer"/>
        </xs:extension>
```

```
      </xs:complexContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="methods">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="0" maxOccurs="unbounded" ref="method"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="method">
    <xs:complexType>
      <xs:complexContent>
        <xs:extension base="name">
          <xs:sequence>
            <xs:element ref="description"/>
            <xs:element ref="parameters"/>
          </xs:sequence>
          <xs:attribute name="visibility" use="required"
type="xs:integer"/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="parameters">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="0" maxOccurs="unbounded" ref="parameter"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="parameter">
    <xs:complexType>
      <xs:complexContent>
        <xs:extension base="name">
          <xs:attribute name="mode" use="required" type="xs:NCName"/>
          <xs:attribute name="type" use="required" type="xs:NCName"/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="associations">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="association"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="association">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="role"/>
        <xs:element ref="roleClass1"/>
        <xs:element ref="cardinalityClass1"/>
        <xs:element ref="roleClass2"/>
        <xs:element ref="cardinalityClass2"/>
        <xs:element ref="notes"/>
```

```
      </xs:sequence>
      <xs:attribute name="idLink" use="required" type="xs:integer"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="role" type="xs:NCName"/>
  <xs:element name="roleClass1">
    <xs:complexType/>
  </xs:element>
  <xs:element name="cardinalityClass1" type="xs:NMTOKEN"/>
  <xs:element name="roleClass2">
    <xs:complexType/>
  </xs:element>
  <xs:element name="cardinalityClass2" type="xs:NMTOKEN"/>
  <xs:element name="notes">
    <xs:complexType/>
  </xs:element>
  <xs:element name="states">
    <xs:complexType/>
  </xs:element>
  <xs:element name="transitions">
    <xs:complexType/>
  </xs:element>
  <xs:element name="swimlanes">
    <xs:complexType/>
  </xs:element>
  <xs:element name="description" type="xs:string"/>
  <xs:element name="precondition" type="xs:string"/>
  <xs:element name="actors" type="xs:string"/>
  <xs:complexType name="name">
    <xs:sequence>
      <xs:element ref="name"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="name" type="xs:string"/>
</xs:schema>
```

APÊNDICE D – MANUAL DO USUÁRIO

D.1 Introdução

O **RAPDIS** é um ambiente de Engenharia de Software Apoiada por Computador (CASE, *Computer Aided Software Engineering*) que possibilita a aplicação da **MDA**. Este ambiente tem como objetivo apoiar a construção dos modelos e a automatizar as transformações apresentadas ao longo deste livro. A figura D.1 a seguir mostra a tela de abertura do **RAPDIS**.

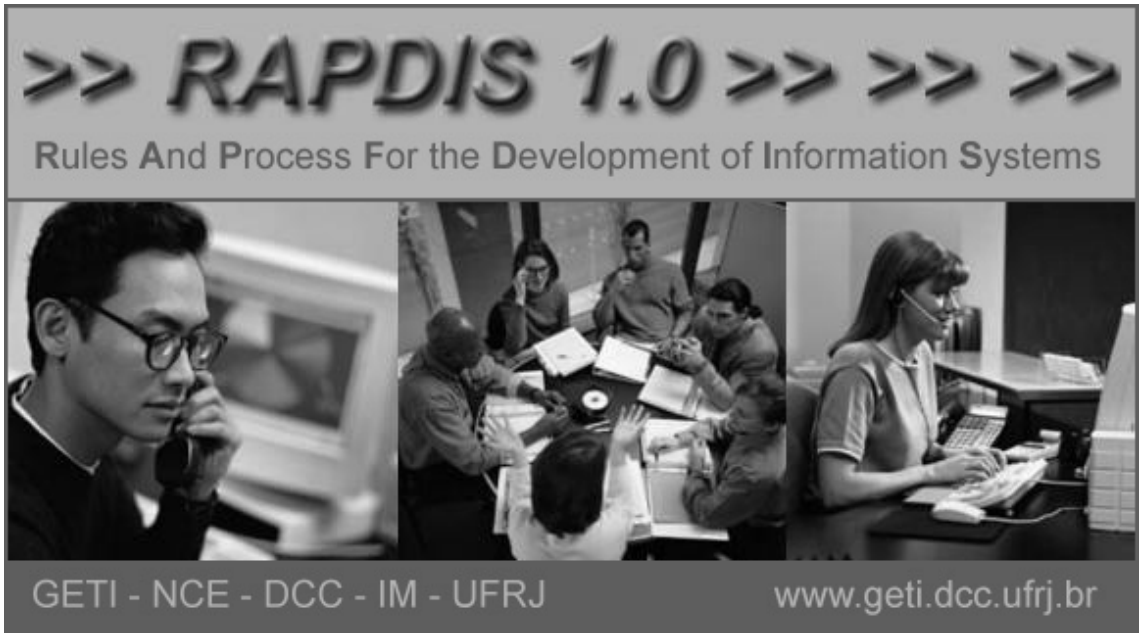


Figura D.1 – Tela de Abertura do **RAPDIS**

D.2 Instalação

Para instalar o **RAPDIS**, execute o programa **INSTALL_RAPDIS.EXE** disponível no endereço eletrônico www.geti.dcc.ufrj.br/projetos/rapdis. Siga os passos indicados pelo instalador até o final.

Além disso, para que o gerador de código do **RAPDIS** funcione corretamente, é necessário ter o BDE (*Borland Database Engine*) instalado. Caso o BDE não esteja instalado, execute também o programa **INSTALL_BDE.EXE** disponível no mesmo endereço eletrônico. Siga os passos indicados pelo instalador até o final.

Após concluir a instalação do **RAPDIS** e do BDE, o ambiente está pronto para ser utilizado. Execute o **RAPDIS** através do atalho criado na Área de Desenho ou do atalho do menu iniciar do Windows.

D.3 Bem-vindo ao RAPDIS

Ao entrar no **RAPDIS**, é exibida uma tela (Figura D.2) com três opções para se iniciar o uso do ambiente: novo projeto, abrir projeto e abrir projetos recentes. Para criar um projeto, clique em NOVO; para abrir um projeto que já existe, clique em ABRIR; e para abrir um projeto que foi recentemente utilizado, selecione um projeto da lista RECENTES e clique no botão ABRIR. O objetivo desta tela de boas-vindas é promover um acesso mais fácil às opções do projeto. A exibição desta tela pode ser cancelada marcando a opção NÃO EXIBIR MAIS ESTA TELA AO INICIAR O RAPDIS.

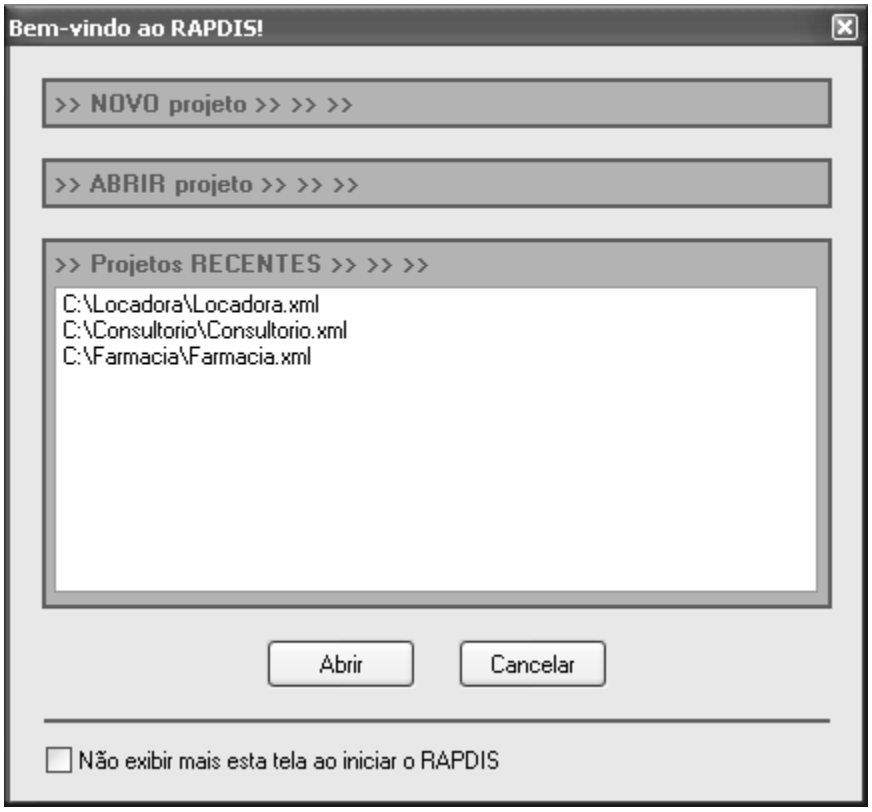


Figura D.2 – Tela Bem-vindo ao **RAPDIS**

D.4 Tela Principal do RAPDIS

Ao fundo da tela de boas-vindas encontramos a Tela Principal do **RAPDIS** (figura D.3). Esta tela apresenta os seguintes componentes: o Menu Principal, a Barra de Ferramentas e as Abas Principais.

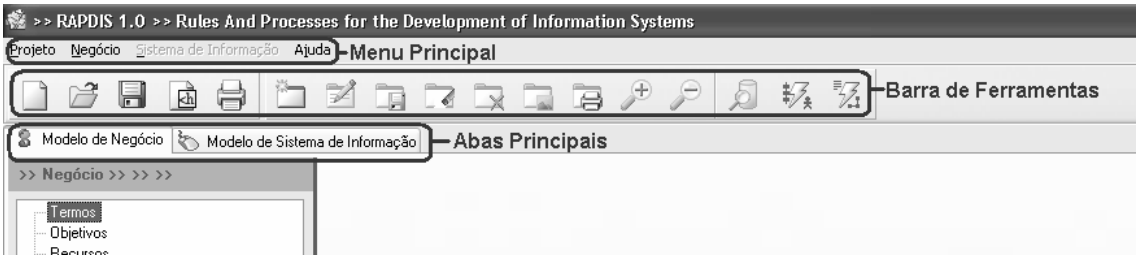


Figura D.3 – Tela Principal do **RAPDIS**

D.5 Trabalhando com Projetos

No **RAPDIS**, as informações sobre os projetos estão armazenadas em diretórios. As funcionalidades disponibilizadas para se trabalhar com um projeto são apresentadas nos itens a seguir.

D.5.1 Criando Projeto

Bem-vindo ao RAPDIS ⇒ NOVO Projeto
Menu Principal ⇒ Projeto ⇒ Novo...
Barra de Ferramentas ⇒ Botão Novo Projeto

Para criar um novo projeto, selecione o local onde será criado o diretório de projeto. Preencha o nome do projeto e o nome do diretório que será criado, como é ilustrado na figura D.4. Clique no botão CRIAR para concluir a operação.

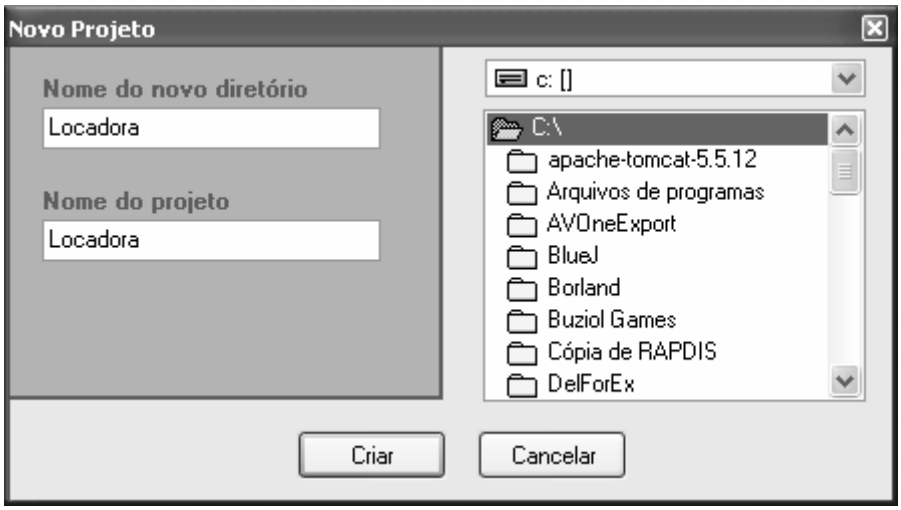


Figura D.4 – Criando Novo Projeto

D.5.2 Abrindo Projeto

Bem-vindo ao RAPDIS ⇒ ABRIR Projeto
Menu Principal ⇒ Projeto ⇒ Abrir...
Barra de Ferramentas ⇒ Botão Abrir Projeto

Para abrir um projeto existente, selecione o diretório do projeto e clique no arquivo do projeto (extensão .xml). Ao selecionar este arquivo, as informações NOME DO PROJETO e DATA serão mostradas (figura D.5). Clique no botão ABRIR para concluir a operação.

Figura D.5 – Abrindo Projeto

D.5.3 Salvando Projeto

Menu Principal ⇒ Projeto ⇒ Salvar Barra de Ferramentas ⇒ Botão Salvar Projeto

Para guardar em disco as alterações feitas no projeto em edição, selecione a opção salvar projeto e confirme a operação.

D.5.4 Gerando Relatório do Projeto

Menu Principal ⇒ Projeto ⇒ Gerar Relatório... Barra de Ferramentas ⇒ Gerar Relatório do Projeto

Uma vez que os modelos tenham sido construídos, é possível gerar um relatório contendo as definições de todos os componentes do projeto. Para isto, selecione a opção gerar relatório. Em seguida, configure o relatório (figura D.6), selecionando os componentes desejados e o formato do arquivo de saída (HTML ou Doc). Clique em gerar para concluir a operação. Com este relatório é possível disponibilizar os modelos na Web ou efetuar a sua impressão.

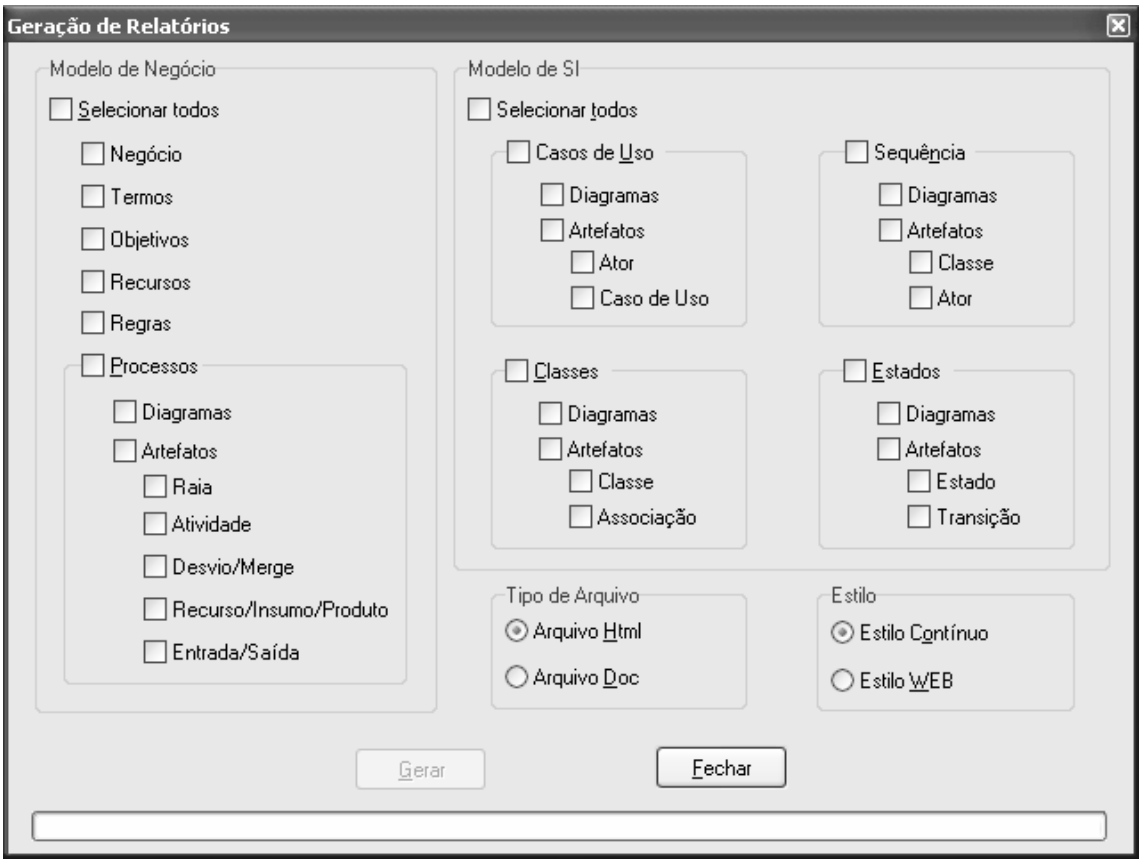


Figura D.6 – Gerando Relatório

D.5.5 Finalizando Projeto

Menu Principal ⇒ Projeto ⇒ Sair

Para finalizar um projeto, selecione a opção sair e confirme a operação. Antes de sair, informe se deseja salvar as últimas alterações feitas no projeto.

D.6 Construindo Modelos

Dentro de um projeto do **RAPDIS** é possível construir dois tipos de modelos: o **Modelo de Negócio** e o **Modelo de Sistema de Informação**. As abas principais permitem a navegação entre estes dois modelos.

D.6.1 Construindo o Modelo de Negócio

Abas Principais ⇒ Modelo de Negócio

O **Modelo de Negócio** (figura D.7) é composto por objetos de cinco tipos: **Termos**, **Objetivos**, **Recursos**, **Processos** e **Regras**. Estes objetos estão dispostos numa estrutura de árvore (Árvore do Negócio) e organizados de acordo com o seu tipo. Abaixo da Árvore do Negócio, encontramos um campo extra que pode ser preenchido com a descrição do negócio. As funcionalidades para a construção deste modelo são apresentadas nos itens a seguir.

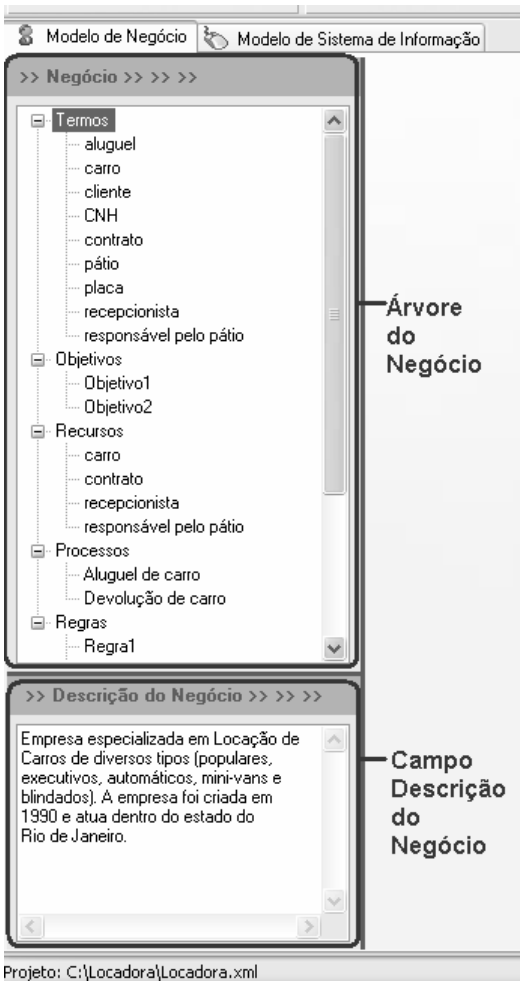


Figura D.7 – Modelo de Negócio

D.6.1.1 Criando Objeto do Modelo de Negócio

Menu Principal ⇒ Negócio ⇒ Novo Objeto ...
Barra de Ferramentas ⇒ Botão Novo Objeto do Negócio

Para criar um novo objeto do Negócio, selecione o tipo do objeto e informe o seu nome no campo correspondente (figura D.8). Caso o tipo selecionado seja um recurso, o seu nome deve ser um termo previamente cadastrado no modelo de negócio. Após preencher os campos, clique no botão CRIAR. O novo objeto será adicionado na árvore do negócio e será exibida a tela para a sua edição.

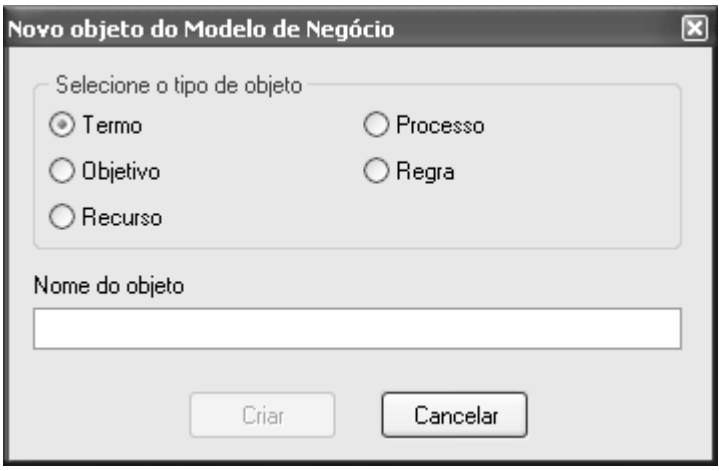


Figura D.8 – Criando Objeto de Negócio

D.6.1.2 Editando Objeto do Modelo de Negócio

Árvore do Negócio ⇒ Duplo clique no nome do objeto

Para editar um objeto do negócio, selecione o seu nome na Árvore do Negócio. As informações sobre este objeto serão exibidas na janela à direita da árvore. Estas informações variam de acordo com o tipo do objeto e estão prontas para serem editadas (figura D.9).

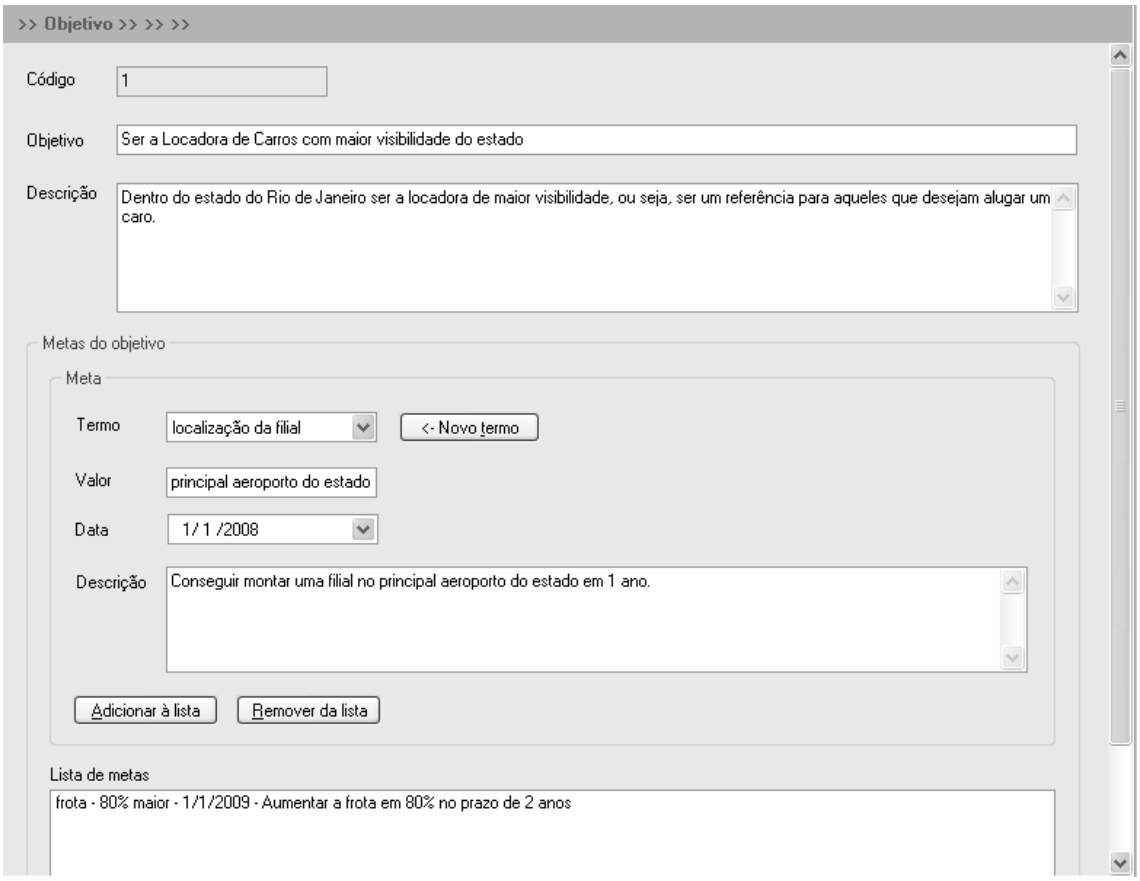


Figura D.9 – Editando Objetivos

D.6.1.3 Salvando Objeto do Modelo de Negócio

Menu Principal ⇒ Negócio ⇒ Salvar Objeto
Barra de Ferramentas ⇒ Botão Salvar Objeto

Para guardar as alterações feitas na definição de um objeto em edição, selecione a opção salvar objeto e confirme a operação.

D.6.1.4 Apagando Todo o Objeto do Modelo de Negócio

Menu Principal ⇒ Negócio ⇒ Apagar Todo o Objeto
Barra de Ferramentas ⇒ Botão Apagar Todo o Objeto

Para apagar todo o conteúdo da definição do objeto em edição, sem excluir este objeto, selecione a opção apagar objeto e confirme a operação.

D.6.1.5 Excluindo Objeto do Modelo de Negócio

Menu Principal ⇒ Negócio ⇒ Excluir Objeto
Barra de Ferramentas ⇒ Botão Excluir Objeto

Para excluir o objeto em edição do Modelo de Negócio, selecione a opção excluir objeto e confirme a operação.

D.6.1.6 Consultando Regras

Menu Principal ⇒ Negócio ⇒ Consultar Regras
Barra de Ferramentas ⇒ Consultar Regras

Com o objetivo de facilitar o processo de definição das regras, o **RAPDIS** dispõe também de funcionalidades para realizar consultas em Prolog. Para isto, efetue a conexão com o Prolog (botão CONECTAR). Escolha um modelo de sentença e construa uma pergunta. Clique no botão CONSULTAR. A pergunta é traduzida para o Prolog e processada. O resultado obtido é traduzido de volta ao usuário no formato textual padrão (figura D.10).

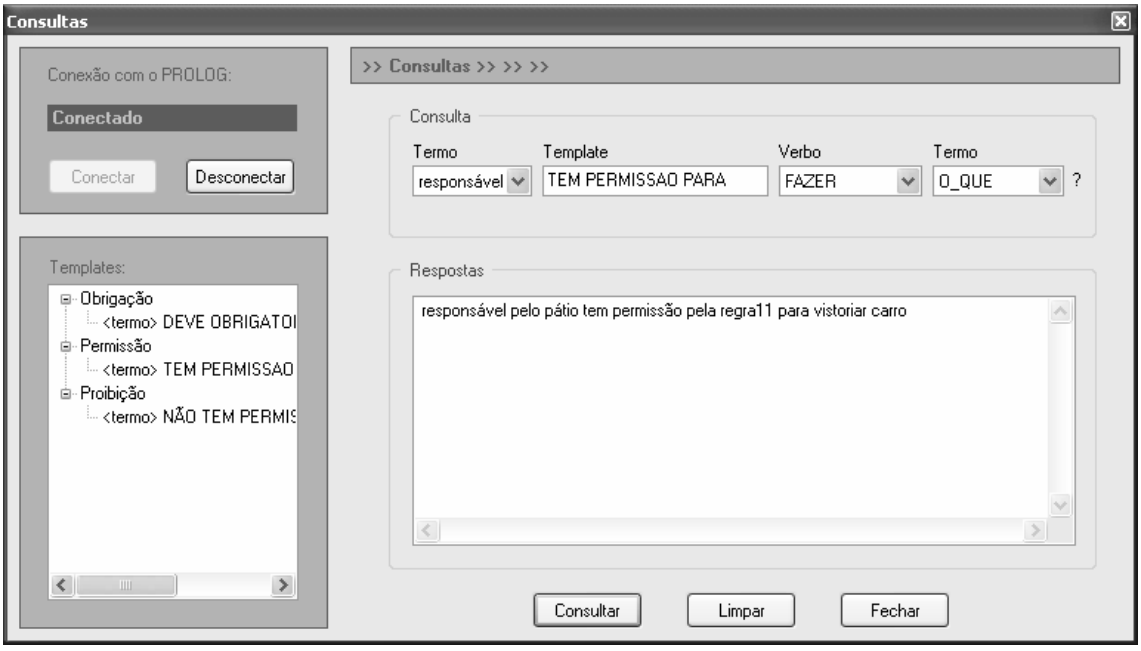


Figura D.10 – Consultando Regras

D.6.2 Construindo o Modelo de Sistema de Informação

Abas da Tela Principal ⇒ Modelo de Sistema de Informação

O **Modelo de Sistema de Informação** (figura D.11) é composto por diagramas de quatro tipos: **Diagrama de Casos de Uso**, **Diagrama de Classes**, **Diagrama de Seqüência** e **Diagrama de Estados**. Estes diagramas estão dispostos numa estrutura de árvore (Árvore do Sistema de Informação) e organizados de acordo com o seu tipo. Abaixo da Árvore do Sistema de Informação, encontramos um campo extra que pode ser preenchido com a descrição do sistema. As funcionalidades para a construção deste modelo são apresentadas nos itens a seguir.

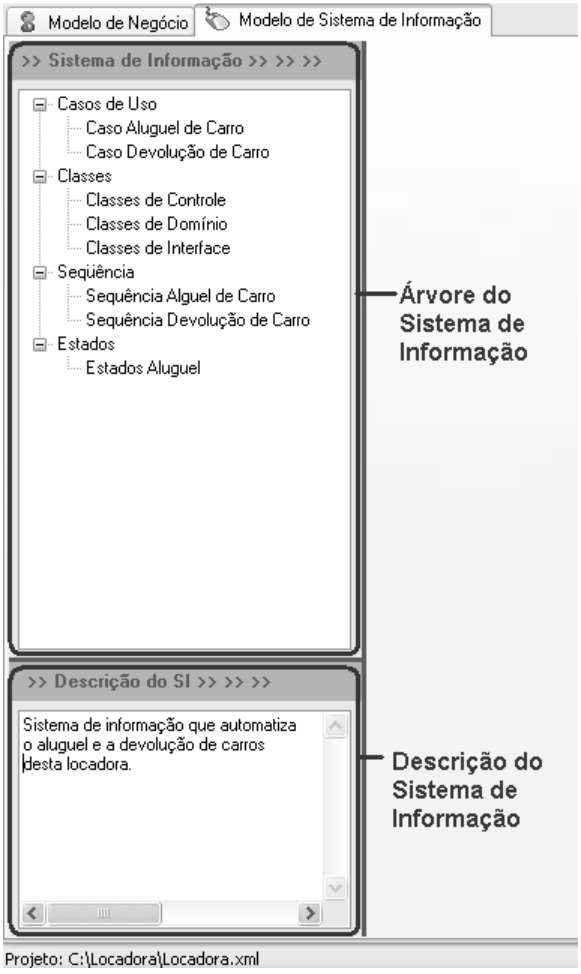


Figura D.11 – Modelo de Sistema de Informação

D.6.2.1 Criando Diagrama do Modelo de Sistema de Informação

Menu Principal ⇒ Sistema de Informação ⇒ Novo Diagrama ...
Barra de Ferramentas ⇒ Botão Novo Diagrama

Para criar um novo diagrama do Modelo de Sistema de Informação, selecione o tipo do diagrama e informe o seu nome no campo correspondente (figura D.12). Após preencher os campos clique no botão CRIAR. O novo diagrama será adicionado na Árvore do Sistema de Informação e será exibida a tela para a sua edição.

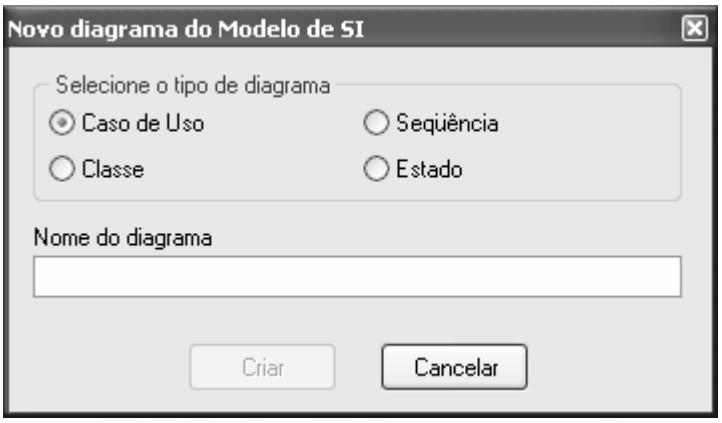


Figura D.12 – Criando Diagrama de Sistema de Informação

D.6.2.2 Editando Diagrama do Modelo de Sistema de Informação

Árvore do Sistema de Informação ⇒ Duplo clique no nome do diagrama

Para editar um diagrama, selecione o seu nome na Árvore do Sistema de Informação. A tela para edição deste diagrama será exibida na janela à direita da árvore (figura D.13). Esta tela apresenta os seguintes componentes: a Barra de Ferramentas do Diagrama e a Área de Desenho. A Barra de Ferramentas do Diagrama possui um conjunto de elementos e ligações que varia de acordo com o tipo do diagrama. A Área de Desenho é a parte da tela utilizada para a edição do diagrama.

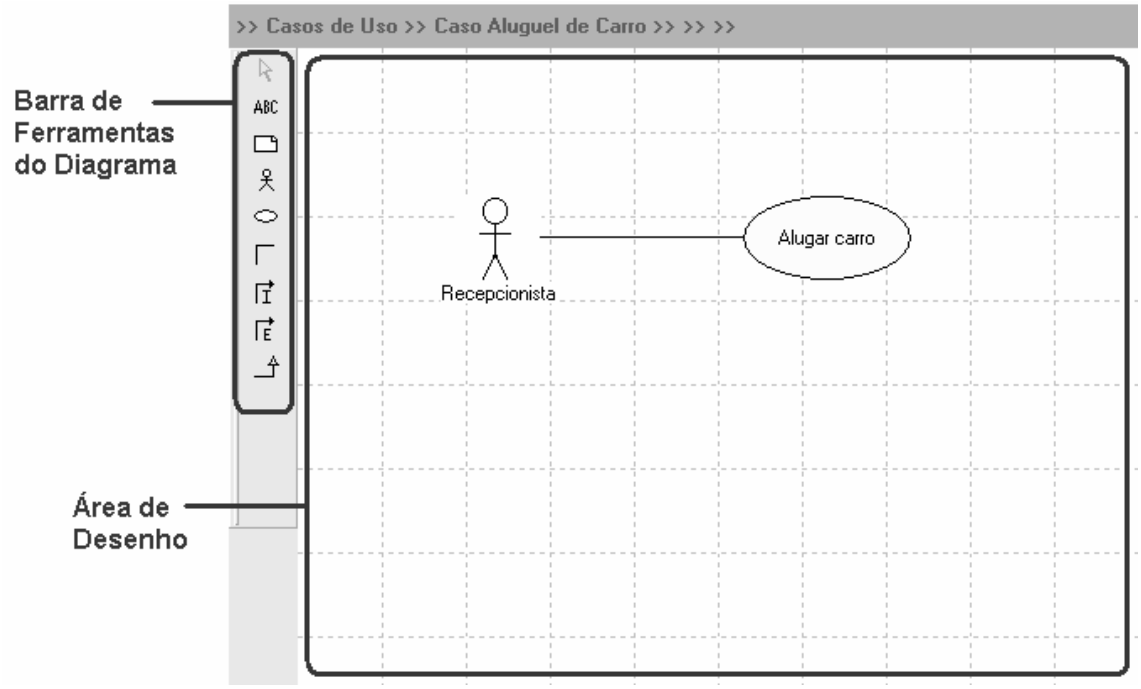


Figura D.13 – Editando Diagrama de Casos de Uso

D.6.2.3 Salvando Diagrama do Modelo de Sistema de Informação

Menu Principal ⇒ Sistema de Informação ⇒ Salvar Diagrama
Barra de Ferramentas ⇒ Botão Salvar Diagrama

Para guardar as alterações feitas em um diagrama em edição, selecione a opção salvar objeto e confirme a operação.

6.2.4 Apagando Todo o Diagrama do Modelo de Sistema de Informação

Menu Principal ⇒ Sistema de Informação ⇒ Apagar Todo o Diagrama
Barra de Ferramentas ⇒ Botão Apagar Todo o Diagrama

Para apagar todo o conteúdo do diagrama em edição, sem excluir este diagrama, selecione a opção apagar objeto e confirme a operação.

D.6.2.5 Excluindo Diagrama do Modelo de Sistema de Informação

Menu Principal ⇒ Sistema de Informação ⇒ Excluir Diagrama Barra de Ferramentas ⇒ Botão Excluir Diagrama

Para excluir o diagrama em edição do Modelo de Sistema de Informação, selecione a opção excluir objeto e confirme a operação.

D.7. Trabalhando com Diagrama

Para construir os diagramas, o **RAPDIS** dispõe de um conjunto extra de funcionalidades. Estas funcionalidades são apresentadas nos itens a seguir.

D.7.1 Criando Elemento

Barra de Ferramentas do Diagrama ⇒ Selecionar um Elemento

Para criar um elemento, selecione-o na Barra de Ferramentas do Diagrama, mova o cursor para um local onde não exista nenhum elemento desenhado (espaço em branco) e dê um clique. Dependendo do tipo de elemento que está sendo criado, será necessário informar o seu nome (figura D.14). Preencha o nome do elemento e clique no botão OK.

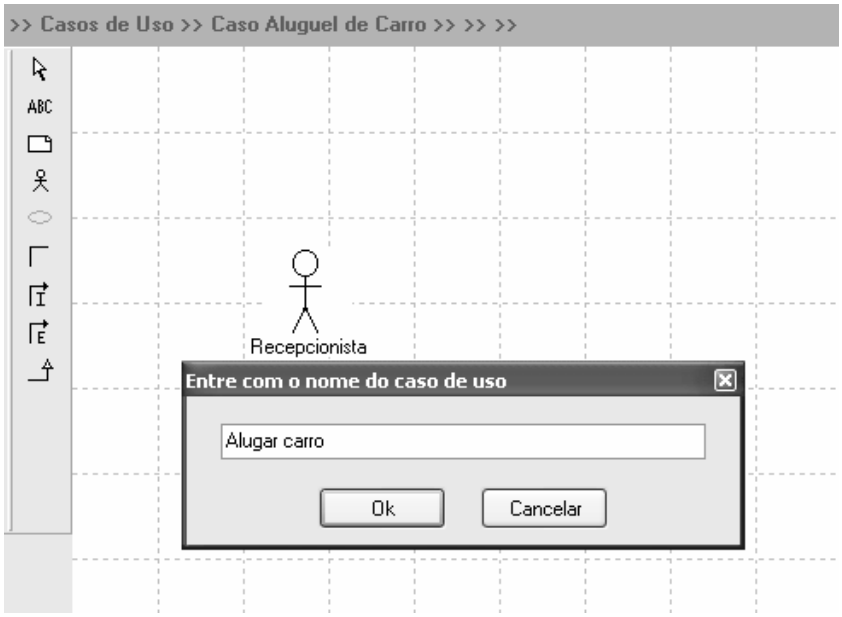


Figura D.14 - Criando Elemento (Caso de Uso)

D.7.2 Criando Ligação

Barra de Ferramentas do Diagrama ⇒ Selecionar uma Ligação

Para criar uma ligação, selecione-a na Barra de Ferramentas do Diagrama, mova o cursor para um elemento (origem) e dê um clique. A partir deste momento, o diagrama irá traçar uma ligação do ponto de origem até a posição do mouse. Para completar a ligação, dê um sobre o segundo elemento (destino).

D.7.3 Adicionando Ponto de Quebra nas Ligações

Botão Direito do Mouse ⇒ Adicionar Ponto de Quebra

Os pontos de quebra permitem que se desenhe uma ligação não-retilínea. Há duas formas se criar um ponto de quebra: (1) durante o desenho da ligação, mova o cursor para um local onde não exista nenhum objeto (espaço em branco), clique com o botão direito do mouse e selecione a opção ponto de quebra; ou (2) na ligação já desenhada, clique com o botão direito do mouse e selecione a opção adicionar ponto de quebra (figura D.15).

Figura D.15 – Adicionando Ponto de Quebra

D.7.4 Movendo Elementos

Selecionar Elemento ⇒ Arrastar

Para mudar um elemento já desenhado de lugar, mova o cursor para o elemento e pressione o botão mouse sobre ele. Este comando "prende" o elemento ao cursor, de maneira que, quando movemos o cursor, este elemento também se movimenta pela tela. É bom ressaltar que o elemento ficará "preso" ao cursor do mouse somente enquanto estiver pressionado o botão do mouse.

D.7.5 Removendo Elemento/Ligação

Botão Direito do Mouse ⇒ Remover Tecla de Atalho Delete

Para remover um elemento, clique com o botão direito do mouse sobre o elemento/ligação e selecione a opção remover. O elemento/ligação será marcado como removido (colorido de vermelho). Em seguida, clique novamente com o botão direito e selecione a opção confirmar remoção (figura D.16). Uma outra forma de realizar a remoção é utilizando a tecla de atalho Delete.

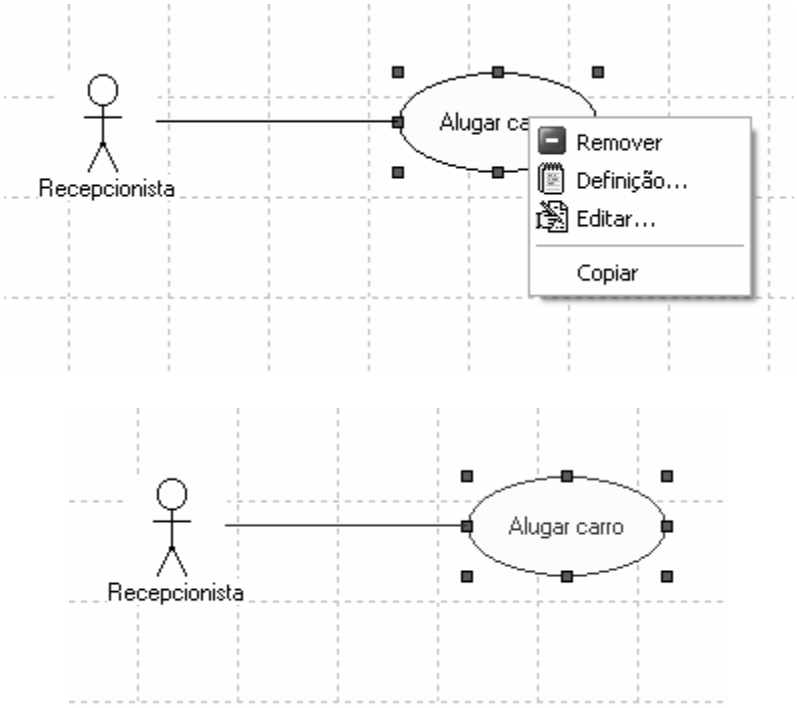


Figura D.16 – Removendo Elementos/Ligações

D.7.6 Editando Elemento

Botão Direito do Mouse ⇒ Editar

Para trocar o seu nome de um elemento, clique com o botão direito do mouse sobre o elemento e selecione a opção editar. Preencha o campo com o novo nome e clique no botão OK (figura D.17).

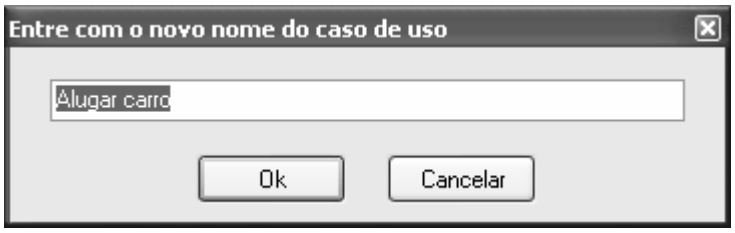


Figura D.17 – Editando Elemento

D.7 Definindo Elemento/Ligação

Botão Direito do Mouse ⇒ Definir

Alguns elementos/ligações do diagrama possuem uma definição. Para editar esta definição, clique com o botão direito do mouse e selecione a opção definir. Preencha os campos da definição do elemento e clique no botão OK (figura D.18).

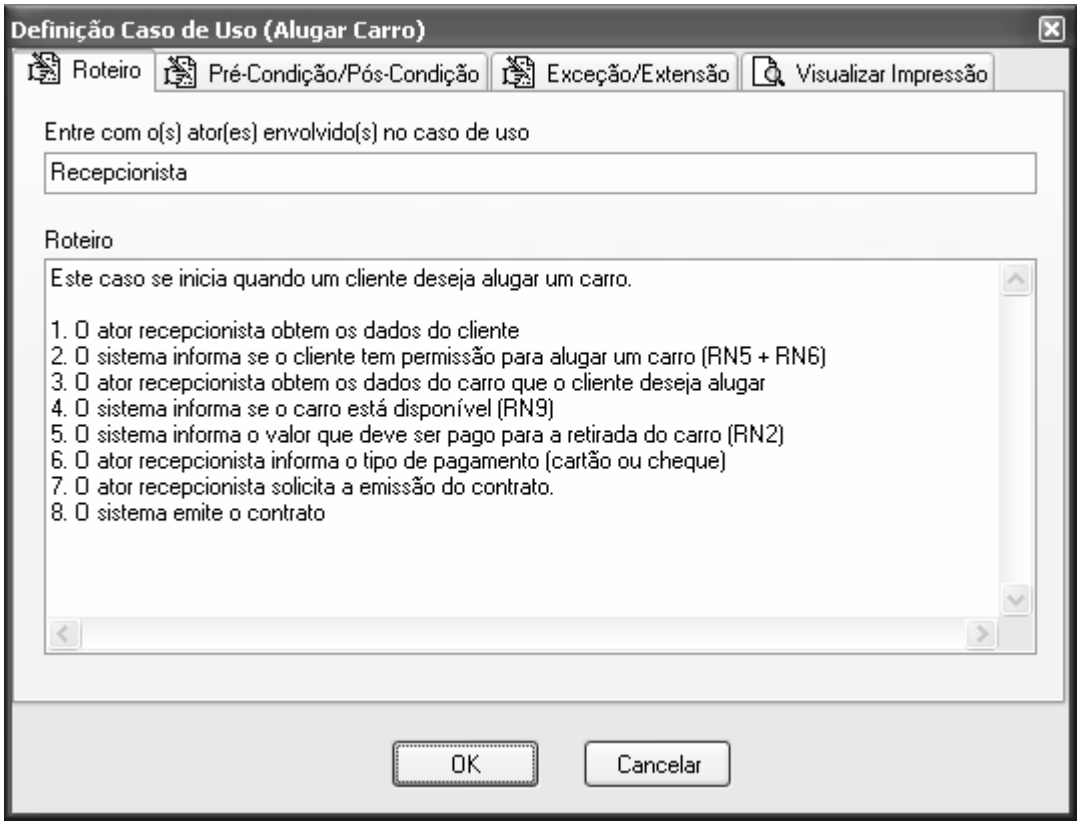


Figura D.18 – Definindo Elemento/Ligação

D.7.8 Copiando/Colando Elemento

Botão Direito do Mouse ⇒ Copiar/Colar
Tecla de Atalho Ctrl+C/Ctrl+V

Para copiar um elemento para a área de transferência, clique com o botão direito sobre o elemento e selecione a opção copiar. Em seguida, clique com o botão direito do mouse sobre uma área vazia e selecione a opção colar. Informe se deseja criar um novo elemento ou criar um ponteiro para o elemento que já existe. Caso esteja sendo criado um novo elemento, preencha o novo nome e clique no botão OK. A funcionalidade colar só funciona se o elemento que está na área de transferência faz parte da linguagem do diagrama destino. Uma outra forma de realizar esta operação é utilizando as teclas de atalho Ctrl+C e Ctrl+V.

D.7.9 Configurando Diagrama

Menu Principal ⇒ Negócio/Sistema de Informação ⇒ Configurar Objeto/Diagrama...
Barra de Ferramentas ⇒ Configurar Objeto/Diagrama
Botão Direito do Mouse ⇒ Configurar

A configuração permite alteração das propriedades do diagrama em edição. Estas propriedades são: o nome do diagrama e o modo de diagramação (figura D.19). O modo de diagramação está relacionado ao funcionamento da grade da Área de Desenho. Para configurar o diagrama, preencha o nome e o modo de diagramação e clique no botão CONFIGURAR.

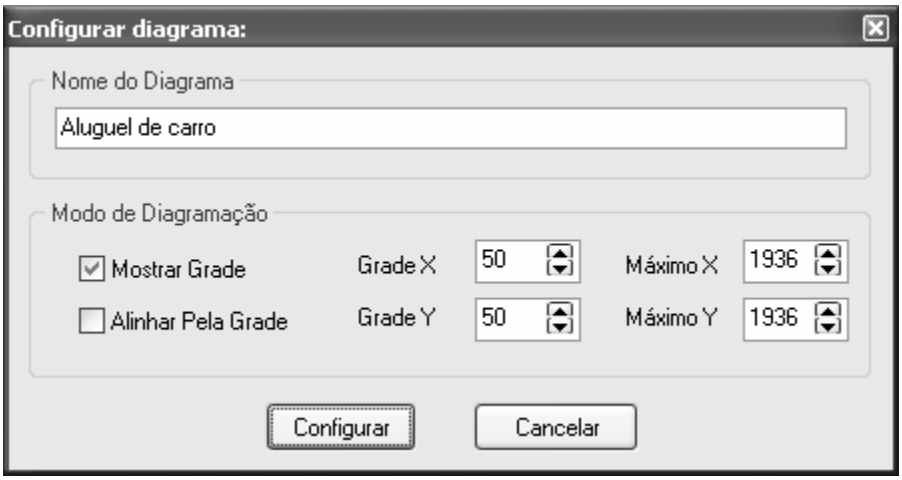


Figura D.19 - Configurando Diagrama

D.7.10 Efetuando Zoom In/Out

Menu Principal ⇒ Negócio/Sistema de Informação ⇒ Zoom In/Out
Barra de Ferramentas ⇒ Zoom In/Out
Botão Direito do Mouse ⇒ Zoom In/Out

O Zoom permite aumentar/diminuir a Área de Desenho do diagrama em edição. Para isto, clique sobre a operação zoom e clique Área de Desenho.

D.7.11 Gerando JPG do Diagrama

Menu Principal ⇒ Negócio/Sistema de Informação ⇒ Gerar JPG do Objeto/Diagrama...
Barra de Ferramentas ⇒ Gerar JPG do Objeto/Diagrama
Botão Direito do Mouse ⇒ Gerar JPG

Este gerador permite exportar o desenho do diagrama em edição através de uma imagem no formato JPG. Para isto, clique na opção gerar JPG e selecione um caminho de saída para o arquivo JPG.

D.8. Transformando Modelos

O **RAPDIS** dispõe permite a aplicação de algumas transformações entre os modelos. Estas transformações são automatizadas a partir dos geradores apresentados nos itens a seguir.

D.8.1 Gerando Casos de Uso

Menu Principal ⇒ Negócio ⇒ Gerar Casos de Uso
Barra de Ferramentas ⇒ Gerar Casos de Uso

Uma vez que os processos do negócio tenham sido definidos, é possível transformar estes processos em casos de uso. Para isto, clique na opção gerar casos de uso e confirme a

operação salvar projeto. Selecione o processo de origem e dê um nome para o diagrama de casos de uso destino (figura D.20). Clique em gerar. O novo diagrama será exibido na aba correspondente ao Modelo de Sistema de Informação.

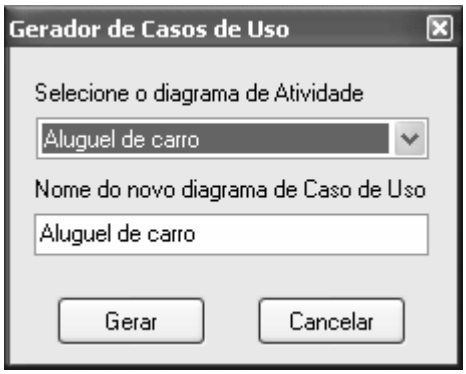


Figura D.20 – Gerando Casos de Uso

D.8.2 Gerando Classes

Menu Principal ⇒ Negócio ⇒ Gerar Classes
Barra de Ferramentas ⇒ Gerar Classes

Uma vez que os termos do negócio tenham sido definidos, é possível transformar esta definição em classes de domínio. Para isto, clique na opção gerar classes e dê um nome para o diagrama de classes destino (figura D.21). Clique em gerar. O novo diagrama será exibido na aba correspondente ao Modelo de Sistema de Informação.

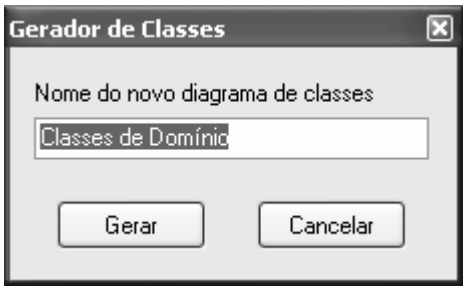


Figura D.21 - Gerando Classes

D.8.3 Gerando Código

Menu Principal ⇒ Sistema de Informação ⇒ Gerar Código
Barra de Ferramentas ⇒ Gerar Código

Uma vez que o Modelo de Sistema de Informação tenha sido definido, é possível transformar este modelo em código. Para isto, clique na opção gerar código e confirme a operação salvar projeto. Em seguida, configure o gerador de código (figura D.22). A configuração do gerador é composta pela escolha da plataforma, geração da tabela e dos casos de manutenção para as classes de domínio, especificação da maquete para classes de interface, recuperação do código da versão anterior, associação dos casos de uso com os diagramas de seqüência e preenchimento das informações do sobre. Após efetuar a configuração, clique no botão GERAR. Confirme a operação de salvar configuração do gerador de código. Especifique um caminho para código que será gerado. Os arquivos serão criados no caminho especificado e o gerador exibirá uma mensagem informado o sucesso desta operação.

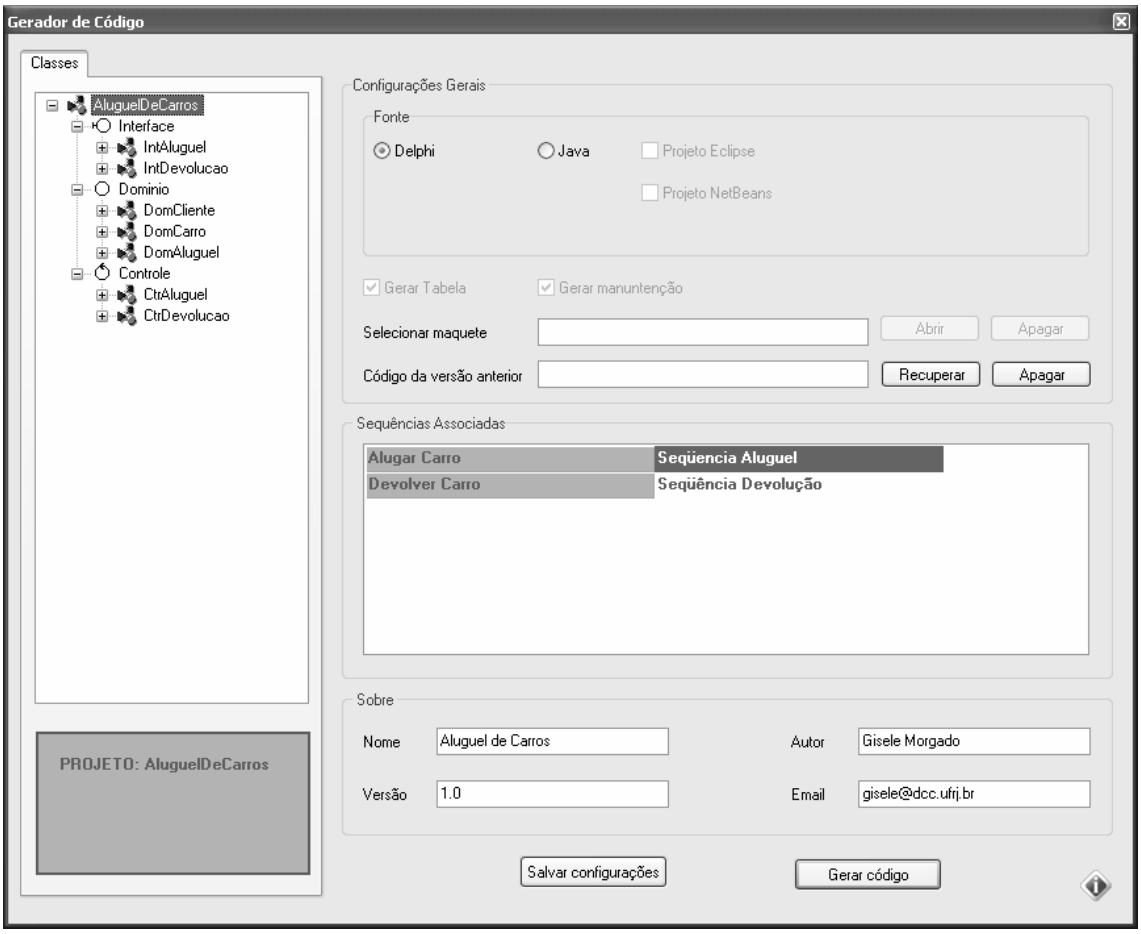


Figura D.22 – Gerando Código

D.9 Atualizações, Licença e Suporte

Para ter acesso às atualizações do **RAPDIS**, utilize a página do grupo GETI www.geti.dcc.ufrj.br/projetos/rapdis. A versão que está sendo distribuída neste endereço possui fins acadêmicos e apresenta uma limitação de 50 classes por projeto. Para obter uma versão sem esta limitação ou para enviar algum comentário sobre este ambiente, entre em contato através do e-mail geti@dcc.ufrj.br.