Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

# SGMOO: SISTEMA GESTOR DE MÉTODOS ORIENTADOS A OBJETOS BASEADO EM CONHECIMENTO

por
**Gilene do Espírito Santo Borges**

Dissertação submetida ao Departamento de Ciência da Computação
como requisito parcial para obtenção do grau de
Mestre em Ciência da Computação

Orientadora
Prof(a). Dr(a). Maria Elenita Menezes Nascimento

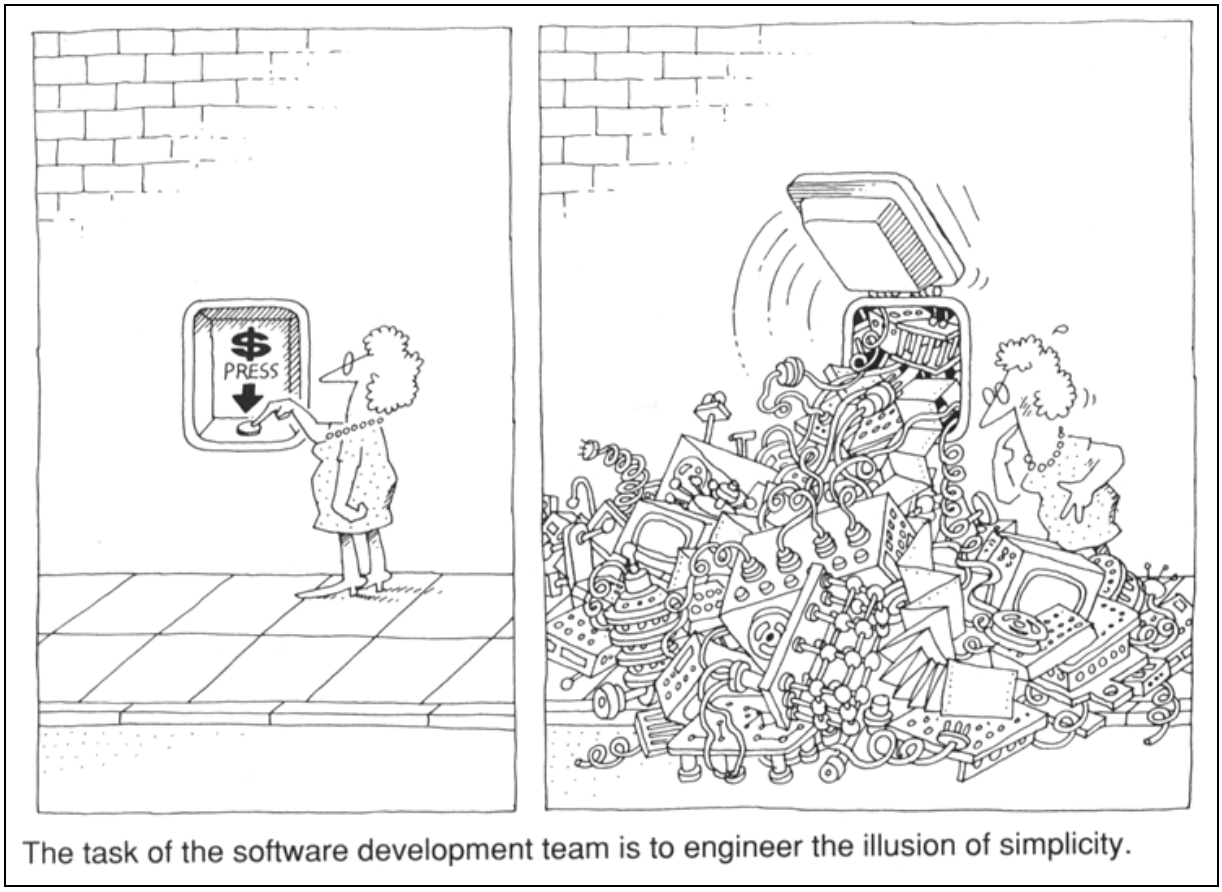
Brasília, Dezembro de 1998.

BORGES, Gilene do Espírito Santo
SGMOO: Sistema Gestor de Métodos Orientados a Objetos Baseado em Conhecimento / Gilene do Espírito Santo Borges: UnB, Ciência da Computação, 1998.
p. 236 : il.
Dissertação (mestrado) - Universidade de Brasília, curso de pós-graduação em Ciência da Computação, Brasília, 1998.
Orientadora: Dra. Maria Elenita Menezes Nascimento

1. Orientação a objetos
2. Métodos de desenvolvimento
3. Funções de crença

Aos meus pais **Gil** e **Leny**,
aos meus irmãos **Cleide** e **Everton**,
ao meu amigo e amado **Alexandre M. Gomes**,
ao meu amigo e tutor **Fábio Bianchi Campos** e
às minhas amigas **Jussely Costa**, **Renata P. Oliveira** e **Viviane Meireles**.

# *Mensagem*




Grady Booch (Booch, 1992)

*A tarefa da equipe de desenvolvimento de software é projetar a ilusão de simplicidade.*

# Agradecimentos

Quero agradecer a **Deus** pela proteção e orientação que me são sempre concedidos.

À minha orientadora, **Dra. Maria Elenita Menezes Nascimento**, pelas críticas, pela orientação e também pelo incentivo que me foram tão preciosos.

Ao colega e amigo **M.Sc. Fábio Bianchi Campos** pela grande paciência e inestimável disponibilidade em poder me esclarecer dúvidas a cerca dos aspectos técnicos de Engenharia de *Software*.

Ao prof. **Dr. Wagner Teixeira da Silva** e à colega **M.Sc. Raquel Regis A. Carvalho** pelo carinho com que sempre sanaram minhas dúvidas sobre a teoria de função de crença.

Aos meus pais **Gil** e **Leny** e irmãos **Cleide** e **Everton** pela ajuda, compreensão e coragem, que mesmo de longe souberam me dar.

Ao meu grande amigo e amado **Alexandre Mesquita Gomes** pelo seu carinho, paciência e apoio que me foram tão preciosos, possibilitando-me concluir este trabalho com tranqüilidade e perseverança.

Aos meus tios **Manoel** e **Luceny Soares** e primos que me acolheram em sua casa, me proporcionando uma tranqüilidade inestimável. Igualmente, aos meus tios **Edilson** e **Lucely Moura** que acolheram em sua casa no início desta batalha.

Aos colegas de mestrado que, de uma forma ou de outra, me auxiliaram na realização deste trabalho, em especial **M.Sc. Renata Peluso de Oliveira**, **José Roberto Valentin**, **M.Sc. Lellis Marçal Mesquita**, **André Luiz Moura**, **M.Sc. Estevam Rafael Hruschka Júnior**, **Ádja de Jesus Rêgo** e **Hélio Berchó Pereira** (in memorian).

Às minhas duas amigas, **Jussely Costa** e **Viviane Meireles**, que sempre estiveram do meu lado me incentivando e me passando a certeza de que tudo daria certo. Jamais esquecerei que foi você, Jussely, quem me encorajou a iniciar essa batalha.

Aos meus professores de graduação, **Joilson dos Reis Brito** e **Ronaldo Del Fiaco**, pelo incentivo através de recomendações.

Aos professores do departamento que muito contribuíram através de críticas e sugestões, especialmente ao **Dr. Francisco de Assis Cartaxo Pinheiro**, **Dra. Alba Cristina de Melo** e **Dr. Antônio Nuno de C. Santa Rosa**.

Aos funcionários da UnB, em especial aos do departamento do CIC: **Rosa**, **Marinalva**, **Márcia**, **Sandra**, **Cida**, **Pablo** e **Celso**.

À empresa ***Hewlett Packard Laboratories***, que gentilmente enviou-me artigos de suma importância para minha pesquisa. Ao **depto. de Ciências Matemáticas e de Computação da USP - São Carlos**, pela inestimável ajuda com artigo e sugestões, que foram de grande valia a minha pesquisa.

Aos meus amigos "virtuais": **Carlos Eduardo de Barros Paes** (PUC - SP), **Dr. Guilhermo Bustos Reinoso** (Chile), **Ismar Frango Silveira** (ITA - SP), **Leandro Pompermaier** (UFRGS - Rio Grande do Sul), **M.Sc. Rejane Moreira da Costa** (USP - São Carlos) e **M.Sc. Ricardo Pereira e Silva** (UFRGS - RS), que foram muito atenciosos; enviando de longe monografias, referências, artigos e *sites* para que eu pudesse compreender melhor o assunto.

A CAPES pela disponibilização de uma bolsa de estudos.

# Resumo

Atualmente estão disponíveis diferentes métodos para desenvolvimento de *software*. No entanto, estratégias apropriadas para seleção dos métodos mais adequados dependem das características da aplicação a ser desenvolvida. Uma estratégia apropriada é aquela que provê um sistema de alta qualidade e com bons resultados a custos mínimos de manutenção. O objetivo dessa pesquisa envolve a seleção de métodos orientados a objetos mais adequados para um projeto de desenvolvimento de *software*. Para que essa seleção seja possível, a partir de um estudo comparativo, será identificado um conjunto de características desses métodos. Uma associação entre essas características e as do sistema será feita, possibilitando a escolha dos métodos mais adequados. O resultado obtido será uma associação bem fundamentada, que poderá ser utilizada como guia em qualquer desenvolvimento de sistema possibilitando uma qualidade e rendimento superiores aos encontrados atualmente, devido ao fato de se utilizar o método mais adequado ao desenvolvimento. Entretanto, esse guia não garante o sucesso do desenvolvimento, o qual deverá ser medido também por outros aspectos, como: qualificação e tamanho da equipe de desenvolvimento, alterações de requisitos, etc. Um protótipo foi desenvolvido a fim de validar o estudo, utilizando a teoria de função de crença para combinar evidências, favorecendo um método ou subconjunto desses. Estas evidências são mapeadas através das características dos sistemas existentes e combinadas com as características dos métodos. Com este trabalho, esperamos suportar os gerentes de desenvolvimento, além de conscientizá-los de que a escolha de um único método para o desenvolvimento de vários tipos de sistemas, não é a solução de todos os problemas nessa área. Cada sistema possui suas próprias características e exigem uma escolha adequada para que o produto final apresente qualidade.

# Abstract

Different methods are currently available for software development. However, appropriate strategies to choice more suitable methods depend upon the characteristics of the application to be developed. A suitable strategy is one, which provides a system with high quality and good results with minimum maintenance costs. The goal of this research covers the object-oriented methods selection more appropriate to a project of software development. For this selection to be possible, from a comparative study, a set of characteristics of methods will be identified. An association among these characteristics and characteristics of systems will be made, making it possible to choose the methods more indicated to this development. The obtained result will be a well based association, which could be used as guide in any system development, allowing a better quality and efficiency than the systems found nowadays, due to use the method more adapted to the development. However, this guide does not guarantee the success of the development, which must be considered others aspects like qualification and proportion of development team, requirements alteration, etc. A prototype was developed in order to validate the study, using the belief functions theory to combine evidences that benefit one method or a subgroup of them. These evidences are mapped through the characteristics of existent systems, combined together with the characteristics of the methods. It is intended of this research to support the managers and developers and to be aware that a fixed method choice or a set of them is not the solution to all kinds of systems. Each system has its own characteristics and demands the right choice of the method in order for the final product to present quality.

# Índice Analítico

# Lista de Figuras

# Lista de Tabelas

# Capítulo 1

# *Introdução*

## *1.1 Motivação da pesquisa*

Quando se iniciaram os primeiros desenvolvimentos de *softwares* não eram utilizados técnicas ou passos determinados, porque os tipos de sistemas a serem automatizados eram bastante simples, devido à capacidade de *hardware* ser baixa. Não poderiam ser desenvolvidos sistemas complexos, visto que não teriam onde ser executados.

Com o passar do tempo, especificamente nas décadas de 60 e 70, o *hardware* teve uma grande evolução, juntamente com a evolução das redes, dos bancos de dados, dos sistemas operacionais e de linguagens de programação, que possibilitaram o desenvolvimento de sistemas mais complexos. Essa evolução trouxe consigo, também, vários problemas na área de engenharia de *software*, conhecida como "crise do *software*". Desses problemas os mais conhecidos são: baixa produtividade, qualidade abaixo do adequado, prazos e custos estabelecidos, mas não cumpridos, falta de tempo na coleta de requisitos, alto custo na fase de manutenção, etc. Essa crise surge porque sistemas mais complexos são desenvolvidos sem padrões, orientação ou gerência.

Desde a identificação da crise do *software*, pesquisadores vêm propondo alternativas de solução com o propósito de minimizar os problemas que afetam o desenvolvimento de *software*. Dentre essas alternativas de solução destacam-se as seguintes: (i) criação de modelos de processo, que ajudam a definir estágios no desenvolvimento e a transição entre esses estágios; (ii) criação de métodos que especificam como fazer o desenvolvimento, quais diagramas utilizar e quais modelos gerar; (iii) desenvolvimento de ferramentas CASE para automatizar a criação de diagramas e modelos, deixando todo o sistema coerente; (iv) criação de linguagens de programação; (v) métricas para melhor gerenciar o processo de desenvolvimento e (vi) criação de ferramentas gráficas que auxiliam na criação de diagramas, dentre outros.

Apesar da existência de inúmeras ferramentas, pode-se notar que a insatisfação e desconfiança dos usuários ainda persistem, possivelmente devido à falta de qualidade, elevado tempo despendido, tanto no desenvolvimento quanto na correção e falta de planejamento.

Apesar de o gerente de desenvolvimento dispor de uma enorme quantidade de métodos, tais como: métodos formais, métodos estruturados e métodos orientados a objetos[1], a crise ainda persiste.

[1] Dentre esses últimos, pode-se notar ainda a combinação das melhores características de dois ou três métodos gerando outros métodos, chamados métodos integradores.

O que se tem verificado é que a escolha de um método não apropriado ao desenvolvimento poderá acarretar conseqüências indesejáveis, tais como: dificuldade na aplicação do método, aumentando o prazo de entrega e custos estimados, e como conseqüência um desenvolvimento que poderá gerar um produto sem qualidade e de difícil manutenção.

A dificuldade em se escolher um método adequado para o desenvolvimento de um dado sistema e os riscos de uma escolha errada, geram os vários problemas citados anteriormente. Diante desses problemas, sentimo-nos motivadas a: (i) estudar alguns dos principais métodos orientados a objetos, pois é um assunto que necessita de pesquisas[2] e (ii) identificar para quais tipos de sistema eles seriam mais indicados.

A escolha de um método para o desenvolvimento de um sistema depende, como já dito, das características do sistema a ser desenvolvido, dentre outros aspectos, tais como: conhecimento/treinamento da equipe de desenvolvimento em relação ao método, tempo disponível para o desenvolvimento. Entretanto, estes aspectos não são abordados neste trabalho, porque ele trata somente da parte técnica da escolha de um método. Afirma-se, portanto, que a parte gerencial é muito importante e influi bastante na escolha do método.

A seguir, serão apresentados o objetivo geral e os objetivos específicos deste trabalho.

## *1.2 Objetivos*

O objetivo geral deste trabalho é auxiliar os desenvolvedores de sistema a escolher o método orientado a objetos mais indicado para o desenvolvimento de um determinado domínio de aplicação.

Os objetivos específicos são:

1) Estudar e comparar os métodos orientados a objetos através da comparação entre os métodos será possível determinar quais são os que melhor modelam determinado aspecto, depois de um estudo aprofundado de cada um dos métodos;
2) Estudar uma técnica de Inteligência Artificial, que possibilite identificar quais são os métodos mais indicados para um determinado domínio de aplicação;
3) Desenvolver um protótipo para implementar a proposta, possibilitando a manutenção da base de dados (fases, métodos, perguntas e funções de crença) e interface entre o protótipo e o engenheiro de *software* para aplicação do questionário;
4) Agregar os resultados desse trabalho em um contexto mais amplo, o SMM, trata-se de um modelo desenvolvido por Nascimento. *O SMM - Software Management Model tem o propósito de integrar aspectos gerenciais e técnicos em um único modelo* (Nascimento, 1992). Os resultados deste trabalho seriam adicionados na subfunção do SMM denominada *Building Strategy* (construindo estratégia), mais especificamente em

[2] A escolha dos métodos orientados a objetos se fortalece pelo fato de que trabalho semelhante já foi desenvolvido para os métodos estruturados (Nascimento, 1990).

*Technical Environment Selection*, onde os métodos são escolhidos. Atualmente, somente alguns métodos estruturados estão disponíveis.

## *1.3 Delimitação do estudo*

Essa pesquisa está centrada no estudo de cinco métodos orientados a objetos. O objetivo é identificar, através da descrição dos métodos e da comparação entre eles, alguns elementos essenciais ao desenvolvimento orientado a objeto, tais como conceitos básicos, tipos de abordagens, foco dos métodos, etc. Essa identificação busca facilitar a inclusão de novos métodos orientados a objetos ao trabalho, pois estes serão estudados tendo como base/guia estes elementos essenciais. É importante ressaltar que a possibilidade de inclusão é um aspecto importante porque não limita a pesquisa realizada, uma vez que os métodos orientados a objetos estão em plena evolução.

Os novos métodos que irão surgir poderão ser bastante completos. Isso poderá levar as equipes que desejam um desenvolvimento rápido e simples a adotar métodos mais simples. Os métodos criados até os dias de hoje podem não cobrir todos os aspectos necessários em um desenvolvimento, mas continuarão a ser utilizados no desenvolvimento dos tipos de sistemas para os quais eles têm sido aplicados, pois esses tipos de sistemas e suas características continuarão a existir. Muitas vezes, não haverá necessidade de métodos completos, e sim daqueles que atendam à modelagem necessária.

A seguir, será apresentada a metodologia adotada para o desenvolvimento do trabalho.

## *1.4 Metodologia*

Nesta seção, é descrita a metodologia adotada para o desenvolvimento deste trabalho.

1) Escolha dos métodos que foi feita a partir de critérios definidos, apresentados com uma descrição na seção 3.1.
2) Estudo preliminar dos métodos foi realizado em uma monografia (Borges, 1997), que serviu como *background* para este trabalho. Nela, foi feito um estudo dos métodos orientados a objetos, resultado do primeiro contato com o assunto, sem fazer um detalhamento sobre eles.
3) Estudo aprofundado dos métodos, descrevendo os diagramas e modelos que o método gera, separado por fase de desenvolvimento, a transição entre as fases, a aplicabilidade de cada método e seus pontos positivos e negativos. A partir deste estudo, pôde-se definir os aspectos modelados pelos métodos.
4) Modelagem de partes de um sistema, que se tornou necessária, pois com isto pôde-se ressaltar as características e deficiências de cada método. São modeladas somente as partes mais interessantes para cada tipo de modelo e não o sistema completo, pois o tempo é limitado e a modelagem deveria ser feita seguindo todos os métodos. O sistema escolhido é a automação de uma biblioteca, detalhes são encontrados na seção 3.2 e apêndices B e C.

5) Comparação entre os métodos, onde se identifica um conjunto de aspectos que os métodos modelam, tais como: identificação de requisitos, modelagem de estados, tratamento de concorrência, etc. Assim, é possível reconhecer para quais aspectos os métodos oferecem melhor modelagem, o que permite identificar qual método é mais indicado para o desenvolvimento de um determinado sistema.
6) Identificação das questões, realizada com base no trabalho de Nascimento (Nascimento, 1990) e com ajuda de engenheiros de *software* experientes. Através das respostas é possível identificar as características do sistema a ser desenvolvido.
7) Estudo da teoria de Dempster e Shafer, ou teoria de função de crença, para compreender a nova área e seus conceitos, e aplicar um algoritmo que possibilita identificar o método mais indicado baseando-se em crenças definidas pelo especialista em métodos, ou seja, alguém que tenha estudado profundamente um método ou vários.
8) Desenvolvimento e análise do protótipo, implementado em *Visual Basic for Application*, por oferecer fácil manuseio e rápido desenvolvimento. A análise será feita a partir de simulação de casos reais aplicados à análise do protótipo.

A seguir será descrita a organização do trabalho, sua divisão em capítulos e apêndices.

## 1.5 Organização do trabalho

Este trabalho está dividido em sete capítulos e quatro apêndices. O primeiro capítulo apresentou a descrição do problema que motivou essa pesquisa, os objetivos do trabalho, a delimitação do estudo e a metodologia a ser utilizada para o seu desenvolvimento.

O capítulo 2 apresenta uma revisão bibliográfica dos principais conceitos sobre modelagem de sistemas e conceitos básicos e avançados de orientação a objetos. Apresenta também um *framework*, que será utilizado para nivelar os métodos e possibilitar uma comparação entre eles.

O capítulo 3 descreve como foi realizada a escolha dos métodos orientados a objetos para estudo, apresenta a contextualização do sistema de biblioteca utilizado para ressaltar os pontos positivos e negativos dos métodos e a descrição dos métodos com base no *framework* definido no capítulo 2.

O capítulo 4 expõe uma comparação entre métodos orientados a objetos com a descrição de várias tabelas e a cobertura que os métodos possuem em cada fase do desenvolvimento de um sistema.

O capítulo 5 apresenta a técnica de Inteligência Artificial utilizada para desenvolver o protótipo, a teoria de função de crença e o algoritmo soma ortogonal de Dempster e Shafer, bem como um questionário.

O capítulo 6 descreve o desenvolvimento do protótipo, sua arquitetura, descrição da base de dados, suas telas e uma análise dos resultados obtidos mediante aplicação de alguns casos.

O capítulo 7 apresenta as conclusões, com considerações finais, contribuições ao conhecimento fornecidas a partir do desenvolvimento da pesquisa e as propostas para trabalhos futuros.

Como dito anteriormente, o trabalho é composto ainda de quatro apêndices.

O apêndice A apresenta o resumo da notação utilizada pelos métodos estudados neste trabalho, facilitando a busca do significado de cada notação utilizada na modelagem do sistema.

O apêndice B descreve os requisitos do sistema de biblioteca utilizado para modelagem seguindo diretrizes dos cinco métodos estudados, possibilitando uma melhor crítica em relação a eles.

O apêndice C apresenta os modelos gerados de acordo com cada método e partes dos modelos são apresentadas no capítulo 3 para melhor explicação do método.

O apêndice D apresenta um resumo sobre a Linguagem de Modelagem Unificada (UML), abordando aspectos relevantes sobre esse novo padrão, definido pela OMG - *Object Management Group*.

# Capítulo 2

## *Revisão bibliográfica*

Este capítulo apresenta os conceitos sobre modelagem de sistemas e métodos orientados a objetos. O objetivo é fornecer a base conceitual do assunto tratado no trabalho. Será apresentado ainda um *framework* que será utilizado para nivelar os métodos possibilitando compará-los.

### *2.1 Conceitos de modelagem de sistemas*

No início da computação não eram necessários modelos de processo nem métodos para o desenvolvimento de *software*, porque os sistemas eram simples devido às várias restrições de *hardware* e *software* existentes àquela época. Com o passar dos anos, nas décadas de 60 e 70, as configurações de *hardware*, a interligação de computadores através de redes, os tipos de sistemas operacionais e banco de dados e os recursos das linguagens tiveram um desenvolvimento significativo, possibilitando a automação de sistemas mais complexos. Assim, surgiu a necessidade da utilização dos métodos e modelos de processo no desenvolvimento de *software*, os quais permitem um melhor controle do processo e da equipe de desenvolvimento, permitindo que o sistema seja desenvolvido com mais qualidade e produtividade (Booch, 1994).

Esta seção objetiva descrever os conceitos sobre modelos de processo, fases de desenvolvimento, métodos, notação, modelo, dimensão e diagramas.

Os **modelos de processo** definem a seqüência das fases associadas ao desenvolvimento e a evolução de um *software* e define, também, as regras para finalizar uma fase e passar para a próxima (Nascimento, 1990). *Um modelo de processo se preocupa com "o que fazer depois?" e "por quanto tempo nós devemos continuar a faze-lo?"* (Nascimento, 1990).

Alguns modelos de processo são: *waterfall*, prototipação, espiral e transformacional. Cada um dos modelos de processo aplica diferentes fases para o desenvolvimento, assim, o gerente de desenvolvimento deve conhecê-los para escolher o mais adequado ao tipo de sistema. Um resumo desses modelos de processo pode ser encontrado em (Nascimento, 1990) e (Borges, 1997).

Os modelos de processo estabelecem a seqüência de utilização das fases do processo de desenvolvimento, e estas **fases** são: i) análise, que define o que o sistema deve fazer e permite compreender o domínio do problema; ii) projeto, que define como será implementada a solução desejada; iii) implementação, também chamada de etapa de codificação, é quando ocorre a tradução do que está definido no projeto em código executável; iv) teste, é quando se verifica se o sistema tem consistência em suas partes e como um todo, encontrando e corrigindo os erros; v) manutenção,

ocorre depois que o sistema já foi entregue, quando é necessário corrigir erros descobertos durante a operação ou melhorar a performance.

Após a escolha do modelo de processo, que orienta o desenvolvedor através das fases de desenvolvimento, são escolhidos um ou mais métodos para este desenvolvimento. O modelo de processo, como dito, estabelece as fases para o desenvolvimento e o método estabelece as técnicas a serem aplicadas e os modelos/diagramas a serem elaborados em cada uma dessas fases. Geralmente, os métodos possuem características que se adaptam melhor a determinados modelos de processo; por isso, a escolha conjunta dos dois se torna necessária.

Um **método** é um processo no qual são produzidos vários modelos, os quais ilustram, através de uma notação, diferentes dimensões de um sistema em desenvolvimento.

Segundo (Rumbaugh, 1994), *uma metodologia de engenharia de software é um processo para a produção organizada de software, com utilização de uma coleção de técnicas predefinidas e convenções notacionais. Uma metodologia costuma ser apresentada como uma série de etapas, com técnicas e notação associada a cada etapa*.

É importante notar que a palavra metodologia muitas vezes é utilizada por alguns autores como tendo o mesmo significado da palavra método, como mostrado acima, embora não seja apropriado, pois **metodologia** é o estudo dos métodos. Entretanto, os termos método e metodologia são largamente utilizados como sinônimos.

Os métodos podem ser classificados sob dois aspectos: i) quanto a **abordagem**, onde se observa a orientação indicada pelo método, o aspecto que o metodologista coloca como mais importante: os dados, o comportamento, etc.; ii) quanto ao **foco**, que determina qual a fase do desenvolvimento que o método fornece maior enfoque, por apresentar mais detalhadamente os modelos, técnicas, diagramas e a ordem de execução, que melhor especificam a fase.

A abordagem de um método pode ser: i) orientada a dados, onde a preocupação está nas estruturas de dados, nas informações armazenadas; ii) orientado a eventos, onde o sistema é modelado tendo em vista os eventos que ocorrem no ambiente e as respostas que ele deve fornecer; ou iii) comportamental, onde se modela a dinâmica e as operações do sistema.

O método pode ter como foco a etapa de identificação dos requisitos, a fase de análise ou a fase de projeto. O que se tem notado é que geralmente, um método tem um foco bastante particular, adotando um dos que foram expostos acima.

Depois de exposto o conceito de método, ainda se observa a necessidade de detalhar alguns pontos de sua definição como: notação, modelos e dimensões.

A **notação** é um sistema de representação que é criada e utilizada pelos autores. Usualmente os autores criam uma notação específica para cada elemento, que é representado pelos modelos de seu método. Por exemplo: uma notação para classe, uma para objeto, etc.; criando assim, a notação dos modelos do método. Somente com uma notação bem definida é que um membro da equipe de desenvolvimento conseguirá expressar todas as soluções criadas em sua mente, possibilitando que outras pessoas entendam e critiquem. Entretanto, em muitos casos não

será utilizada toda a notação criada pelo metodologista, que geralmente é bastante detalhada, porque somente parte dela já será suficiente à modelagem requerida.

Como cada autor cria sua própria notação, às vezes, o estudo ou mesmo uso de vários métodos em conjunto gera uma certa confusão. Por exemplo, o que a notação de uma linha com um círculo preenchido na extremidade significa agregação em um método, pode significar multiplicidade em outro. Sendo assim, uma notação padrão, a UML - *Unified Modeling Language*, foi definida por Grady Booch, Ivar Jacobson e James Rumbaugh e apresentada a OMG - *Object Management Group*, que a qualificou como um padrão de notação em novembro de 1997. Esta notação padrão é descrita com mais detalhes no apêndice D.

Um **modelo** *é considerado uma abstração de certos elementos da realidade e os relacionamentos entre eles, a fim de construir uma representação (usualmente em escala menor) que permite o estudo desta realidade* (Nascimento, 1992). Um modelo enfatiza um aspecto da realidade, sendo assim, um método se utiliza de vários modelos para modelar completamente um sistema, visto que os modelos estão em diferentes níveis de abstrações. O uso desses modelos por um determinado método, não impede que outro método utilize alguns desses mesmos modelos (Silva, 1996).

Os modelos são gerados com o propósito de facilitar a compreensão dos problemas e a comunicação entre os membros de uma equipe de desenvolvimento. Os modelos possibilitam também que o analista compreenda o domínio do problema. Os modelos construídos na fase de análise são chamados de modelos de análise, e na fase de projeto são chamados de modelos de projeto. Os modelos em cada fase possuem diferentes objetivos e características.

Os **modelos de análise** são modelos bastante abstratos, que tem como objetivo retratar os requisitos e as funcionalidades do sistema. Um analista, quando gera modelos de análise, deve compreender o domínio do problema e com a ajuda do usuário, deve modelar a realidade deste domínio, baseando-se nos requisitos do usuário. Estes modelos representam o que o sistema é e o que ele deve fazer.

Os **modelos de projeto** são extensões dos modelos de análise, que incorporam detalhes de níveis inferiores, possibilitando a implementação do sistema. Estes modelos são menos abstratos e tem como objetivo retratar o sistema idealizado na fase de análise em um sistema real, baseando-se em condições reais de configuração de *hardware*, tipo de rede de computadores, banco de dados, sistema operacional e linguagem de programação, que serão utilizados para a concretização do sistema em desenvolvimento.

A transição dos modelos de análise para os de projeto, não é uma tarefa simples, e é muitas vezes confusa. Neste sentido Jacobson (Jacobson, 1992) acrescenta: *A transição do modelo de análise para o modelo de projeto deve ser feita quando as conseqüências do ambiente de implementação começarem a aparecer*.

Cada um dos modelos que são criados abrange uma **dimensão** do sistema e isso é fundamental, pois seria bastante confuso tentar expressar todas as dimensões de um sistema em um único modelo, mesmo que este fosse feito em níveis. A dimensão de um modelo é uma visão

particular de algum aspecto do sistema, por exemplo, o modelo estático modela a dimensão de estrutura do modelo, e assim por diante. Cada modelo, então, modela somente uma dimensão detalhando-a ao máximo.

Os modelos de análise são: o modelo estático ou modelo lógico, o modelo dinâmico, o modelo funcional e o modelo de interface.

O **modelo estático ou modelo lógico** representa as definições de dados e comportamentos existentes em um sistema. Neste modelo também são definidos todos os relacionamentos entre dados e comportamentos, bem como, os tipos de associação - herança e agregação. O modelo apresenta uma visão completa da parte estática de um sistema, que é construído baseando-se principalmente no domínio do problema. Entretanto, o modelo estático pode variar de acordo com cada método, como veremos no capítulo 3.

O **modelo dinâmico** mostra toda a parte dinâmica de um sistema, ou seja, a seqüência de eventos, de operações e de controle (as condições impostas), as ações que o sistema deve executar e seus possíveis estados juntamente com suas transições. Este modelo deixa claro como todos esses elementos se relacionam e suas seqüências.

O **modelo funcional** descreve as transformações de informações e o fluxo destas pelo sistema, sem definir seqüência. É apresentado onde são buscados os dados para transformação e onde são armazenados ou para onde são enviados - para outra transformação ou para fora do sistema, como resposta a um evento.

Os **modelos de interface** representam as diversas formas de interface entre o sistema e o usuário. Estes modelos definem as telas de entrada e saída de dados, as janelas e os relatórios do sistema. O uso do modelo de processo prototipação, para a definição de alguns requisitos ainda não estabelecidos, gera estes modelos antecipadamente para que o usuário possa opinar e criticar.

Um dos modelos de projeto é o **modelo físico**, que mostra a alocação de objetos em módulos e a estrutura destes e de subsistemas. Apresenta também, alocação de tarefas a processadores, dispositivos utilizados e conexões efetuadas. Este modelo descreve o *software* concreto e a composição do *hardware* para a implementação do sistema.

Um modelo pode necessitar de vários diagramas para representar toda a informação que deve estar contida nele. Um **diagrama** seleciona, organiza e mostra informação para realçar as partes que mais interessam sobre um assunto e suprimir as menos importantes. Um diagrama não necessita mostrar todas as partes de uma só vez, e se isso acontecer ele será muito confuso (Rumbaugh, 1994).

Como dito anteriormente, um modelo representa somente uma dimensão do sistema, entretanto, esta dimensão pode ter mais de um tipo de diagrama para representá-la totalmente.

Depois de todas essas definições que formam a base da modelagem de um sistema, são descritos os conceitos de orientação a objetos que também são muito importantes para o contexto do trabalho.

## *2.2 Conceitos de orientação a objetos*

Nesta seção serão descritos os conceitos básicos e os conceitos avançados, utilizados pelos métodos orientados a objetos, de uma maneira bem sintetizada. Maiores detalhes são encontrados nas seguintes referências: (Rumbaugh, 1994), (Jacobson, 1992), (Coad, 1992), (Coad, 1993) e (Booch, 1994).

Vale observar que quando for apresentado um conceito básico ou conceito avançado seguido de uma figura para ilustração, tentaremos apresentar parte de algum modelo ou diagrama gerado a partir da modelagem do sistema de biblioteca. Neste caso, ao final do nome da figura será identificado um asterisco (*), indicando que o modelo completo se encontra no apêndice C.

### *2.2.1 Conceitos básicos - modelagem estática*

Em orientação a objetos identificamos os elementos que compõem o domínio do sistema pelas suas características, os **atributos**, e pelas **operações**, as funções ou transformações que serão aplicadas pelo elemento definido.

Uma **classe** armazena as definições dos atributos e das operações de um elemento. Dá-se o nome de **encapsulamento** a capacidade de uma classe proteger seus atributos, permitindo que somente suas operações alterem seus valores. Cada classe pode ter várias instâncias, chamadas **objetos**.

Uma classe pode ser abstrata ou concreta. Somente a **classe concreta** é uma classe instanciável, sendo que a **classe abstrata** não possui instâncias diretas, mas cujas classes descendentes sim. A classe abstrata é somente usada para a associação de herança.

Uma **associação** *é um relacionamento indicando uma conexão semântica entre duas classes* (Booch, 1994). Uma **ligação** é uma instância de uma associação. Alguns autores aceitam a associação entre mais de duas classes, como veremos na seção 2.2.4.

Uma associação pode possuir um atributo, chamado **atributo de ligação**, que não pode ser nem atributo de uma classe nem de outra isoladamente, sem perda de informação. Figura 2.1 apresenta um exemplo onde o atributo de ligação é permissão de acesso. Este atributo não pode ser colocado na classe Usuário, pois um usuário terá diferentes permissões para diferentes arquivos, e nem pode ser colocado na classe Arquivo, porque um arquivo será acessado de maneira diferente por vários usuários.

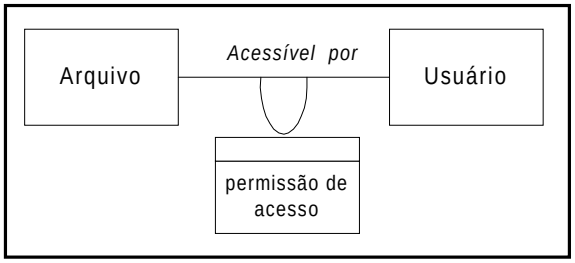



Figura 2.1 - Atributo de ligação (Rumbaugh, 1994)

**Herança** ou **generalização e especialização** é um tipo de associação entre classes, onde todos os atributos e operações definidos na **superclasse** (a classe mais genericamente definida) são compartilhados pela classe mais especializada, a **subclasse**. Na subclasse é possível ser feita a definição de mais atributos e operações e, também, a redefinição das operações que foram herdados pela superclasse.

Pode-se ter **herança simples** onde a subclasse herda os atributos e operações de uma única superclasse e a **herança múltipla** onde a subclasse, chamada de **classe de junção**, herda os atributos e operações de duas ou mais superclasses (Rumbaugh, 1994).

Outro tipo de associação é a **agregação** ou **todo-parte**, onde uma classe representa o todo e uma ou mais classes representam as partes deste todo. *Um agregado (o todo) possui partes, que, por sua vez, podem ter partes* (Rumbaugh, 1994). Um exemplo é ilustrado na figura 2.2, onde a classe Obra é composta por Ficha_tombo.

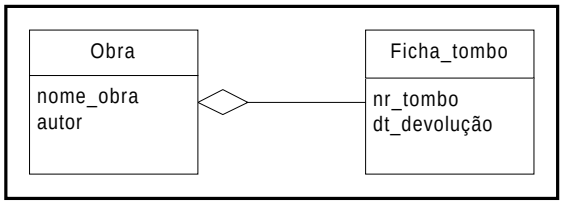



Figura 2.2 - Agregação*

Terminada a base conceitual da modelagem estática, segue a apresentação da modelagem dinâmica.

### *2.2.2 Conceitos básicos - modelagem dinâmica*

Os conceitos básicos da dinâmica dos objetos são: estado, evento, ação, transição e condição, descritos a seguir.

Um **estado** é a situação atual de um objeto definido pelos valores dos atributos deste objeto. Na figura 2.3, os estados do objeto Ficha_controle são: Disponível, Emprestado e Extraviado. O objeto pode assumir todos os estados durante sua existência e somente um de cada vez.

**Eventos** são estímulos internos que acontecem entre objetos, ou externos que são aqueles que vem de fora do sistema, estimulando o sistema a produzir alguma resposta ou mudança de estado. Na figura 2.3, um evento externo é solicita_empréstimo ou devolver_obra e um estímulo interno é cria_ficha ou destrói_ficha.

A reação de um objeto a um evento pode constituir-se de uma ação ou transição. A **ação** é uma operação instantânea associada a um evento. **Transição** é a modificação de estado causada pelo evento que depende do estado corrente e do evento. Na figura 2.3, o evento solicita_empréstimo causa uma transição, uma modificação do estado de Disponível para o de Emprestado. É importante lembrar que um evento pode gerar mais de uma transição, porque a transição depende do estado atual do objeto.

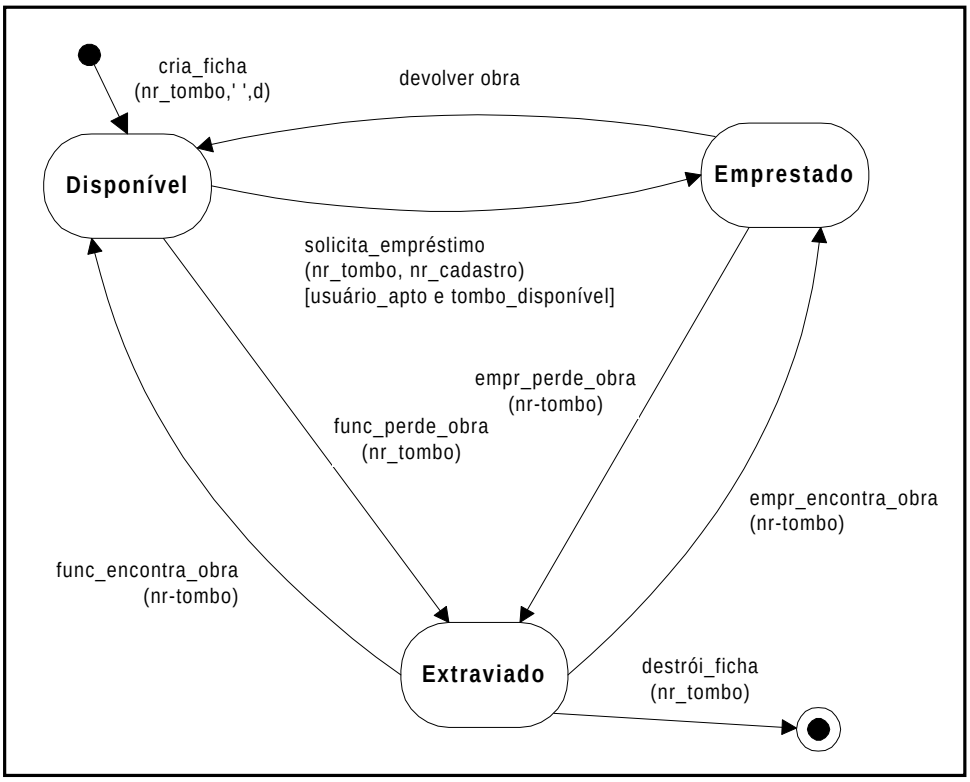



Figura 2.3 - Ilustração dos conceitos básicos de dinâmica*

Uma **condição** *é uma função booleana de valores de objetos válida dentro de um intervalo de tempo*. Somente se a condição for verdadeira, é que a transição poderá ser executada. Um exemplo de condição é Usuário apto e tombo disponível, ilustrado na figura 2.3. Neste caso, as duas condições devem ser satisfeitas.

Se uma condição for usada como guarda nas transições, possibilita que as **transições guardadas** disparem quando seu evento ocorrer, mas somente se a condição de guarda for verdadeira (Rumbaugh, 1994). Observando a figura 2.3, a transição do estado Disponível para Emprestado é uma transição guardada, pois as condições - usuário_apto e tombo_disponível - devem ser verdadeiras para que a transição ocorra.

Estes são os conceitos básicos da modelagem dinâmica, serão descritos a seguir os da modelagem funcional.

### *2.2.3 Conceitos básicos - modelagem funcional*

A modelagem funcional possui conceitos básicos que estão ligados ao diagrama de fluxo de dados, são eles: i) **processos** que transformam os valores de dados e que são as operações das classes; ii) o **fluxo de dados** que interliga a saída de um objeto à entrada de outro, transportando valores de dados; iii) **atores** que são as extremidades dos fluxos de dados, tanto como origens ou como destinos de dados, eles são objetos ativos[3] que consomem ou produzem valores; iv) **depósito**

[3] Um objeto ativo pode executar ações internas sem ter recebido um estímulo externo.

**de dados** é um objeto passivo[4] que armazena dados para o futuro e v) **fluxo de controle** é um valor *booleano* que afeta como um processo é avaliado. A figura 2.4 ilustra a sintática de todos os conceitos abordados acima.

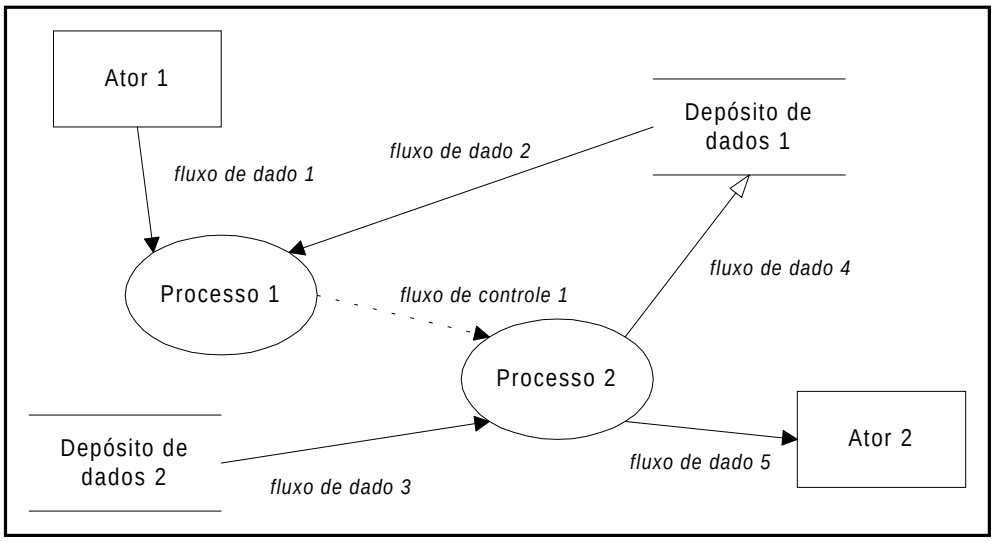



Figura 2.4 - Ilustração dos conceitos básicos de funcionalidade

Nas próximas seções serão descritos os conceitos avançados de orientação a objetos da parte estática e dinâmica, requerendo um completo entendimento das seções anteriores.

### *2.2.4 Conceitos avançados – modelagem estática*

(Booch, 1994) descreve alguns tipos especiais de classe como classe parametrizada e metaclasse. Devido a sua importância, seus conceitos são descritos a seguir.

**Metaclasse** *é uma classe cujas instâncias são classes* (Booch, 1994). Estas instâncias são chamadas de **metaobjetos**, visto que não são objetos do mundo real, suas instâncias é que serão objetos. Uma metaclasse possui seus **atributos de classe** que descrevem um valor comum a toda uma classe de objetos, e suas **operações de classe** que são operações sobre a própria classe, geralmente para criação de novas instâncias de classes.

**Classe parametrizada ou classe genérica** é uma classe utilizada como modelo para a criação de uma família de classes diferenciadas apenas pelos seus parâmetros. A classe parametrizada Pilha - tipo, ilustrada na figura 2.5, deve ser instanciada com um valor para o parâmetro tipo, surgindo assim, a classe Pilha - int e a classe Pilha - string. A partir dessas classes instanciadas é que será possível a instanciação de objetos reais.

Tanto a classe paramétrica como a metaclasse são classes cujas instâncias, também, são classes. A diferença básica é que as instâncias da classe parametrizada possuem os mesmos atributos e operações e são de tipos diferentes. As instâncias da metaclasse compartilham valores de atributos comuns entre si. Todos os dois tipos de classe se preocupam com o reuso, o primeiro ao nível de definições e o segundo ao nível de valores.

[4] Um objeto passivo, ao contrário do objeto ativo, não possui ações, é estático.

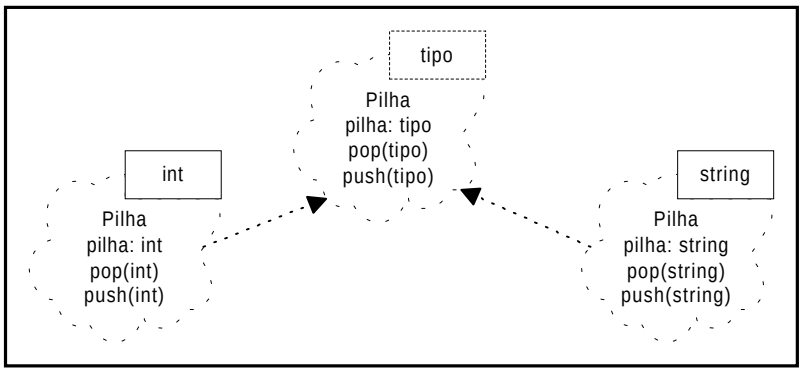



Figura 2.5 - Classe parametrizada

Existem tipos de associações mais especializadas como: associação ternária, associação qualificada, classe agregada, herança por extensão, herança por restrição, agregação fixa, agregação variável, agregação recursiva, agregação física e agregação não física. Estes conceitos não são difundidos em todos os livros de métodos orientados a objetos, mas são descritos por serem de muita importância, especializando os conceitos já apresentados.

As associações podem ser: i) binárias, são as mais comuns; ii) ternárias, são mais difíceis de serem modeladas, mais ainda podem ser encontradas em muitos modelos; ou iii) de ordem mais elevada, que acontecem com menor ou quase nenhuma freqüência. A **associação ternária ou de ordem mais elevada** é uma unidade atômica e não pode ser subdividida em associações binárias sem perder informação (Rumbaugh, 1994). A figura 2.6 apresenta um exemplo de associação ternária, onde o Funcionário cadastra a Ficha_usuário com os dados informados pelo Usuário. A figura 2.7 tenta apresentar a mesma idéia, mas sem a forte semântica apresentada pela figura 2.6.

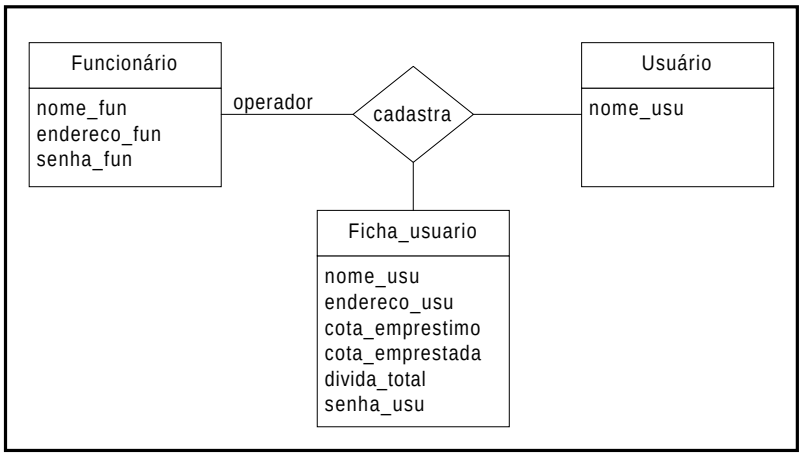



Figura 2.6 - Associação ternária*

Analisando a figura 2.7, não é possível identificar claramente o relacionamento ternário, pois no modelo completo claramente observa-se os relacionamentos binários entre Funcionário e Ficha_usuário e entre esta e Usuário, mas o relacionamento ternário ficaria obscuro.

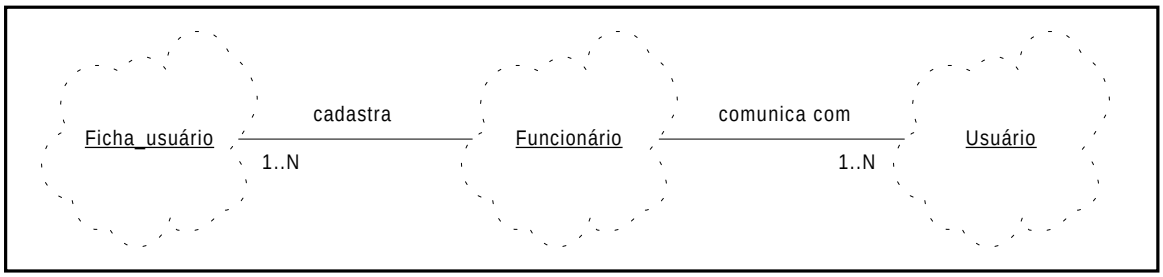



Figura 2.7 - Inexistência de associação ternária*

**Associação qualificada** é estabelecida quando se utiliza de um qualificador para reduzir a multiplicidade de uma associação. Na Figura 2.8 temos que o Cadastro_títulos é uma agregação de muitas Ficha_título, para simplificar é colocado um qualificador, o cod_cadastral, que identifica unicamente as fichas, reduzindo a multiplicidade da associação.

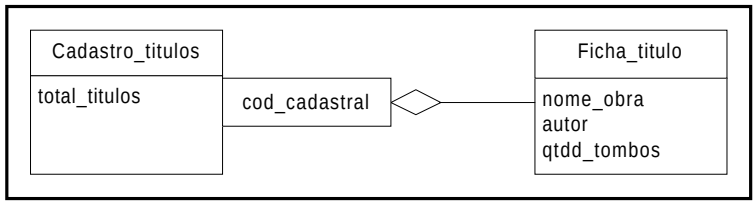



Figura 2.8 - Associação qualificada*

**Classe agregada** é uma classe criada a partir de outras classes, através da herança múltipla, herdando todos seus atributos e operações e não adiciona nenhum atributo ou operação próprios da classe criada.

A **herança por extensão** acontece quando uma subclasse herda todos atributos e operações da superclasse e novos atributos e/ou novas operações são adicionadas à subclasse. Já a **herança por restrição** acontece quando a subclasse herda todos os atributos e operações da superclasse, mas restringe alguns desses, ou atributos e/ou operações. Esta restrição pode acontecer para que a subclasse não possua certos atributos ou para modificar as operações herdadas.

Rumbaugh afirma que a **restrição** (ou cancelamento) das operações pode acontecer devido a quatro razões:

1) **cancelamento para extensão** - a nova operação é igual à operação herdada, mas acrescenta alguns detalhes de comportamento, afetando novos atributos da subclasse.

2) **cancelamento para restrição** - a nova operação pode ter os tipos de argumentos reduzidos, restringindo a interface, o que pode ser necessário para manter a operação herdada dentro da subclasse.

3) **cancelamento para otimização** - a nova operação deve ter a mesma interface e apresentar os mesmos resultados que o antigo, mas sua representação interna e seu algoritmo podem ser completamente diferentes.

4) **cancelamento por conveniência** - as novas operações substituem as operações inconvenientes que foram herdadas utilizando o mesmo nome de operação, mas especificando comportamentos diferentes.

A associação de agregação pode ser classificada em fixa, variável e recursiva. A **agregação fixa** ocorre quando o agregado possui a quantidade das partes fixa, foi apresentado um exemplo na figura 2.2, onde a classe Obra possui uma Ficha_tombo. A **agregação variável** ocorre quando o agregado tem um número de níveis finito, entretanto a quantidade de suas partes é variável.

A **agregação recursiva** ocorre quando o agregado recursivo contém, direta ou indiretamente, uma instância do mesmo tipo do agregado, um exemplo deste conceito é ilustrado pela figura 2.9, onde um Bloco pode ser um Comando simples ou um Comando composto, e a recursão acontece porque o Comando composto pode conter vários Blocos.

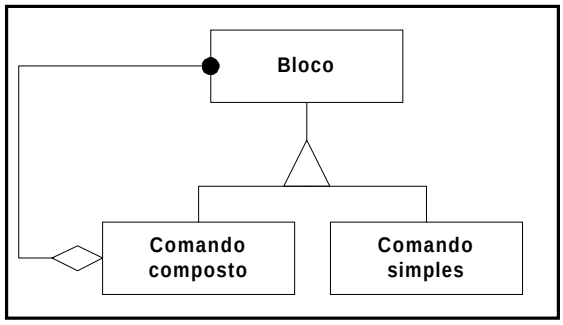



Figura 2.9 - Agregação recursiva (Rumbaugh, 1994)

Uma agregação pode ser física ou não. A **agregação não física** ocorre quando as partes da agregação não existem fisicamente. A figura 2.10 ilustra um exemplo onde o Acionista possui Ações, mas estas ações não são partes físicas do acionista. A **agregação física** indica que as partes da agregação existem fisicamente, como no exemplo da figura 2.10, onde uma Ficha_tombo é uma parte física que está agregado a Obra física (Booch, 1994).

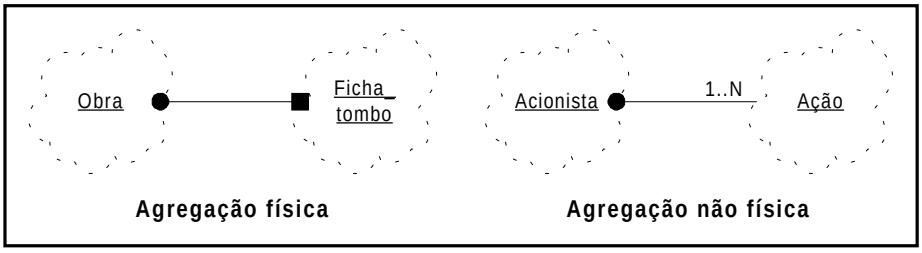



Figura 2.10 - Agregação Física e não Física* /(Booch, 1994)

Uma agregação física pode ser por valor ou por referência. Na **agregação por valor** o tempo de vida do agregado (a classe que representa o todo) e de suas partes estão intimamente conectados. Quando o agregado é instanciado, são instanciadas também as partes deste agregado, e quando a instância do agregado é destruída, as instâncias das partes também são. Na **agregação por**

**referência**, diferentemente da agregação por valor, os tempos de vida do agregado e de suas partes não estão mais ligados, o que permite a criação e destruição das instâncias independente uma das outras (Booch, 1994). A figura 2.10 apresenta um exemplo de agregação por valor, a existência de Ficha_tombo está ligada à existência de Obra.

(Booch, 1994) ainda acrescenta: *agregação por valor não pode ser recursiva (que é, ambos os objetos não podem fisicamente ser partes um do outro), embora agregação por referência possa (cada objeto pode manter um ponteiro para o outro)*.

Um conceito muito importante é o de polimorfismo. Existem dois tipos: polimorfismo único ou polimorfismo e o polimorfismo múltiplo.

**Polimorfismo** *é uma forma de dizer que o mesmo método pode fazer coisas diferentes dependendo da classe onde ele é implementado... Objetos de classes diferentes recebem a mesma mensagem e reagem de formas diferentes. O objeto que enviou a mensagem não sabe a diferença, o receptor interpreta a mensagem e provê o comportamento adequado* (Orfali, 1996).

A figura 2.11 apresenta um exemplo de polimorfismo único. As duas classes derivadas da superclasse Figura, Quadrado e Círculo implementam de forma diferente a mesma operação desenhar. Neste caso, diz-se que esta operação é uma operação polimórfica[5], pois seu comportamento pode ser implementado de forma diferente nas classes Quadrado, Círculo e Figura. Sendo assim, a mesma operação desenhar se comportará de forma distinta preservando, porém, o mesmo nome e parâmetros.

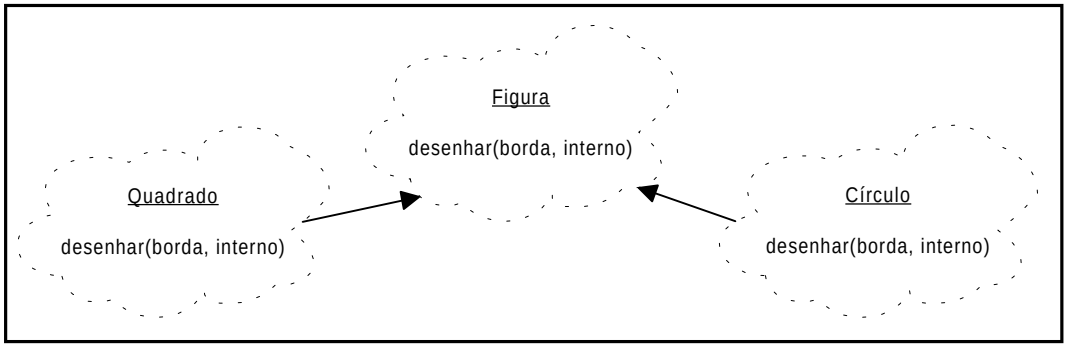



Figura 2.11 - Operação polimórfica

A operação desenhar, ilustrada pela figura 2.12, exemplifica o conceito de **polimorfismo múltiplo**. Além da existência da primeira operação desenhar, ilustrada na figura 2.11, é criado uma nova operação, que difere da anterior pela quantidade e/ou tipo de parâmetros. Todas as duas operações são implementadas pelas subclasses e pela superclasse. Desta forma, a operação desenhar poderá apresentar seis comportamentos diferentes.

Após explanação dos conceitos avançados da modelagem estática, seguem os conceitos avançados da modelagem dinâmica.

[5] Operação polimórfica é o mesmo que método polimórfico, neste trabalho referimos a operação e não a método, por referenciarmos métodos aos métodos de desenvolvimento de sistemas e não para definir o comportamento dos objetos.

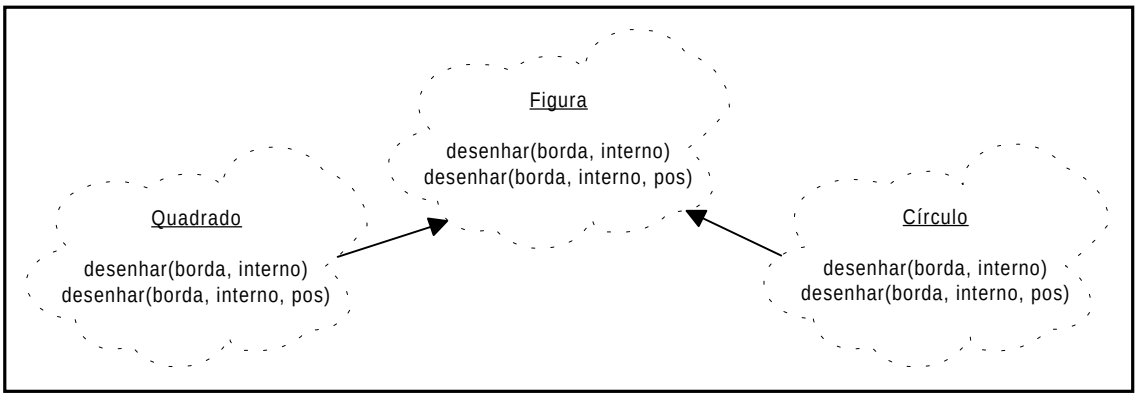



Figura 2.12 - Polimorfismo múltiplo

### *2.2.5 Conceitos avançados - modelagem dinâmica*

Os conceitos básicos da parte dinâmica apresentados na seção 2.2.2 podem ser especializados. Nesta seção são conceituados o superestado e subestado, ações de entrada e de saída, ações internas, atividade e transição automática.

Um estado é chamado de **superestado** quando este possui **subestados**. Os subestados herdam as transições de seus superestados. *Assim, qualquer transição que se aplique a um estado aplica-se a todos os seus subestados, a menos que seja cancelada por uma transição equivalente no subestado* (Rumbaugh, 1994).

As **ações de entrada e de saída** de um objeto são utilizadas geralmente, quando todas as transições em um estado executam a mesma ação, melhorando o aspecto visual, pois as duas notações[6] não possuem diferenças semânticas.

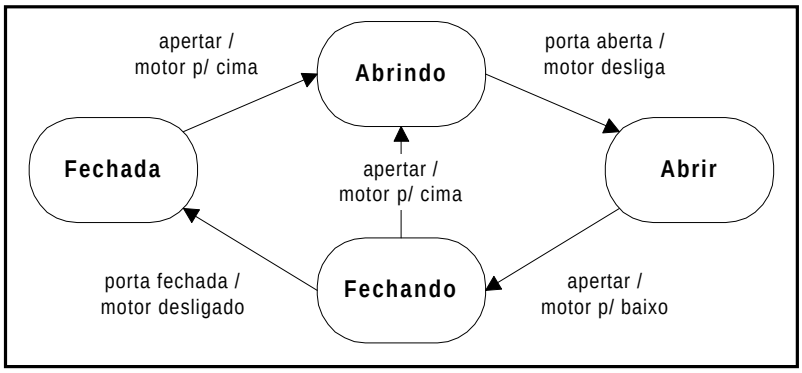



Figura 2.13 - Ações sobre transições (Rumbaugh, 1994)

A figura 2.13 apresenta ações sobre transições, note que o estado Abrindo recebe todas as transições geradas pelo evento apertar. Estas transições têm a ação em comum motor p/ cima, deste modo, pode-se simplificar o modelo, colocando esta ação comum como ação de entrada para o estado Abrindo, como ilustrado pela figura 2.14.

[6] As notações que se refere são as ações sobre transições e ações de entrada e saída representadas dentro do estado.

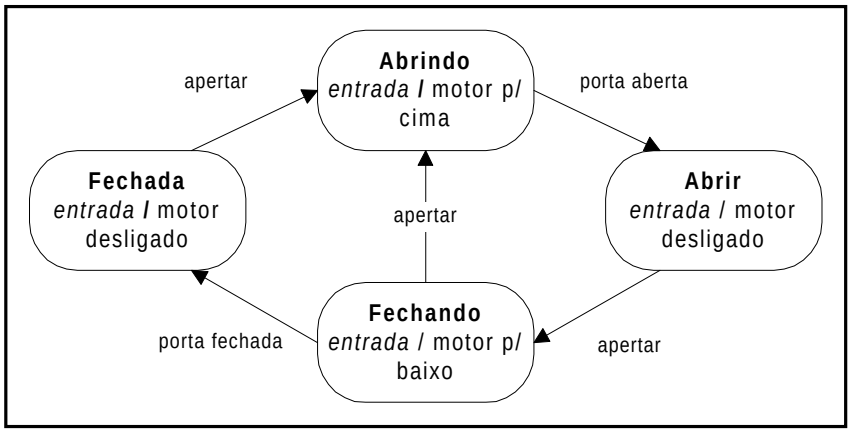



Figura 2.14 - Ações sobre a entrada em estados (Rumbaugh, 1994)

Entretanto, se todas as transições de saída de um estado possuírem a mesma ação, a simplificação do modelo seria criar uma ação de saída para este estado.

As **ações internas** são ações que são executadas sem causar uma mudança de estado. Portanto, as ações internas estão ligadas ao acontecimento de um evento onde as ações de entrada e de saída do estado não são executadas.

**Atividade** é um tipo de ação que é contínua e seqüencial, leva um tempo para executar e está associada a um estado. Quando um objeto mudar para um estado que possui uma atividade, esta inicia-se sendo finalizada após um intervalo de tempo ou quando houver uma transição de estado.

Existe a possibilidade de um estado ter o único propósito de executar uma atividade e quando esta atividade termina, uma transição para outro estado é disparada, a qual recebe o nome de **transição automática**.

(Rumbaugh, 1994) deixa claro a ordem de execução das operações: *se forem especificadas múltiplas operações em um estado, elas serão executadas na seguinte ordem: ações de transição que chega, ações de entrada, atividades faça, ações de saída e ações da transição que sai*.

Outros aspectos a serem levados em conta é o problema da sincronização das mensagens enviadas entre os objetos e a visibilidade entre esses objetos. Estes conceitos são descritos a seguir.

Objetos interagem através do intercâmbio de mensagens. Um objeto pode enviar uma mensagem para um ou mais objetos, os quais podem recebê-lo concorrentemente, então, diferentes operações podem ter diferentes tipos de sincronização, sendo necessário o estudo dos tipos possíveis (Booch, 1994):

i) **sem concorrência** - único fluxo de controle;

ii) **síncrono** - o cliente espera até o servidor aceitar a mensagem, ficando bloqueado enquanto o servidor não aceitar a mensagem;

iii) ***balking*** - igual ao síncrono, exceto pelo fato de que o cliente abandonará a operação se o servidor não estiver imediatamente pronto para receber a mensagem;

iv) ***timeout*** - o cliente abandonará a solicitação, continuando sua seqüência de execução normal se o servidor não atender dentro do tempo estipulado;

v) **assíncrono** - o cliente envia mensagem para a fila e continua sua execução, independente da aceitação ou não da mensagem pelo servidor.

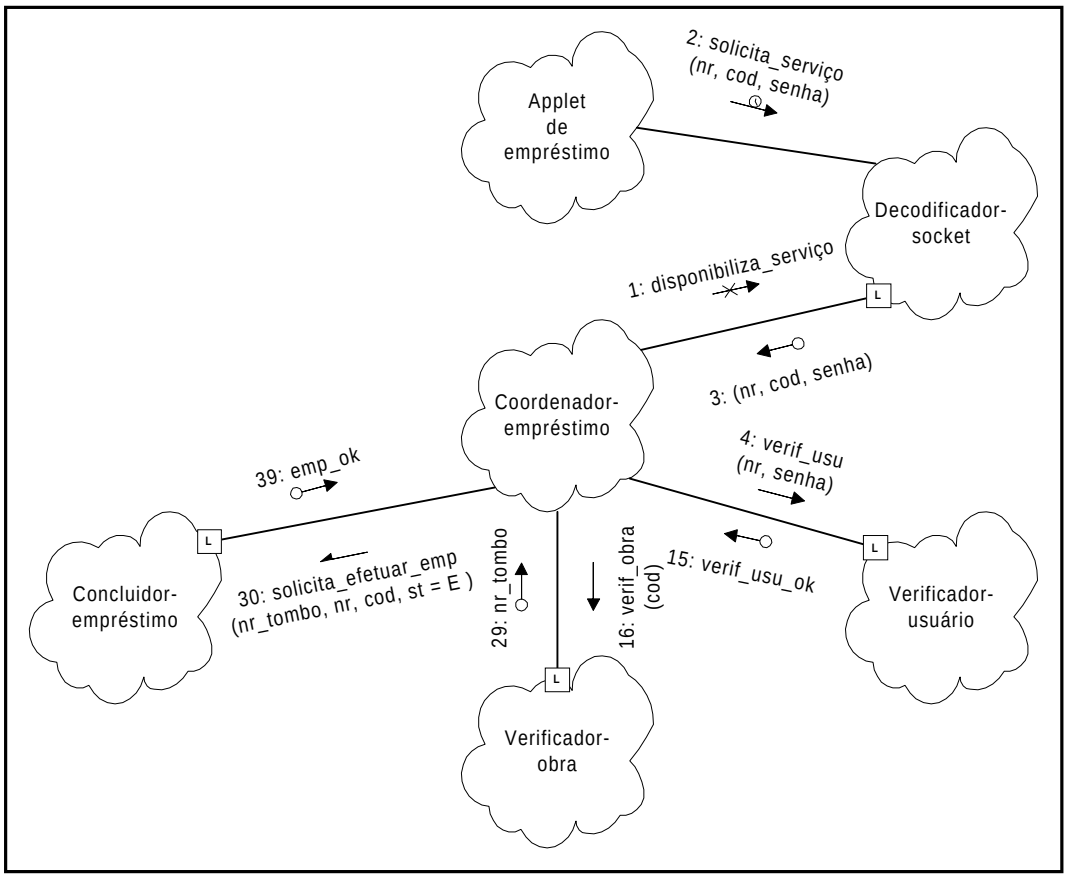



Figura 2.15 - Ilustração dos conceitos de tipos de sincronização*

A figura 2.15 ilustra parte de um modelo de projeto onde é modelada a concorrência. A operação 1: disponibiliza serviço é síncrona, espera até que Decodificador-socket esteja pronto para receber. A operação 2: solicita serviço é do tipo *timeout* e tentará a conexão com Decodificador-socket por um tempo determinado. Através da operação 3: nr, cod, senha são passados valores que serão verificados pelo Coordenador-empréstimo. A operação 4: verif usu é uma operação sem concorrência que possibilita Verificador-usuário validar o usuário. A operação 30: efetua-empréstimo(nr-tombo, nr, cod, st = E) é uma operação assíncrona que é enviada ao servidor independente de sua fila de espera.

Para que os objetos possam enviar mensagens para outros objetos é necessário fazer a definição da visibilidade de cada objeto. Existem quatro maneiras diferentes (Booch, 1994):

i) **global (G)** – quando o objeto servidor é global para o objeto cliente;

ii) **parâmetro (P)** – quando o objeto servidor é um parâmetro para alguma operação do objeto cliente;

iii) **campo (F)** – quando o objeto servidor é uma parte do objeto cliente;

iv) **local (L)** – quando o objeto servidor é um objeto localmente declarado no escopo do diagrama de objeto.

Outro conceito importante neste contexto é o de **cenário**. Segundo (Rumbaugh, 1994), **cenário** *é uma seqüência de eventos que ocorrem durante uma determinada execução de um sistema. A abrangência de um cenário pode variar, ele pode incluir todos os eventos do sistema ou somente aqueles que influenciam ou que são gerados por certos objetos do sistema*. Um exemplo de um cenário com a seqüência de eventos para um empréstimo é ilustrado na figura 2.16.

· Usuário_cadastrado solicita empréstimo ao funcionário.
· Funcionário solicita nr_cadastro ao usuário_cadastrado.
· Usuário_cadastrado informa seu nr_cadastro.
· Funcionário verifica nr_cadastro em ficha_usuário.
· Funcionário verifica se usuário_cadastrado tem dívida em ficha_usuário.
· Funcionário verifica se usuário excedeu cota_empréstimo em ficha_usuário.
· Funcionário verifica se usuário deve tombo em ficha_empréstimo.
· Funcionário solicita cód_cadastral ao usuário_cadastrado.
· Usuário_cadastrado fornece cód_cadastral.
· Funcionário verifica se existe título.
· Funcionário verifica algum tombo disponível.
· Funcionário verifica se tombo está reservado.
· Funcionário verifica se emprestante = reservante.
· Funcionário solicita senha a usuário_cadastrado.
· Usuário_cadastrado fornece senha.
· Funcionário verifica senha em ficha_usuário.
· Funcionário preenche ficha_empréstimo.
· Funcionário atualiza ficha_controle.
· Funcionário atualiza ficha_tombo.
· Funcionário atualiza ficha_usuário com cota_emprestada + 1.
· Funcionário entrega obra ao usuário_cadastrado.

Figura 2.16 - Um cenário para empréstimo*

**Diagrama de interação** é um diagrama utilizado para descrever seqüencialmente a comunicação entre os objetos. É uma maneira gráfica de representar o que está descrito no cenário. A figura 2.17 apresenta parte do diagrama de interação para empréstimo, onde se têm objetos na parte de cima do diagrama, uma descrição de cada mensagem no lado esquerdo e a troca de mensagens entre os objetos em seqüência.

Na próxima seção será descrito o *framework* utilizado para descrever os métodos.

## *2.3 Definição do framework*

Nesta seção será definido o *framework* utilizado no capítulo 3 para descrever os métodos orientados a objetos sob os mesmos aspectos.

O *framework* é composto pela descrição, características, fases, transição entre as fases, aplicabilidade, pontos positivos e negativos, explicados abaixo.

- Descrição - apresenta os autores dos métodos, a evolução do método (se existente) e uma breve descrição do método.
- Características - descreve a abordagem adotada, a fase do desenvolvimento que é dado maior enfoque e alguma característica pertinente ao método, que o destaque dos outros.

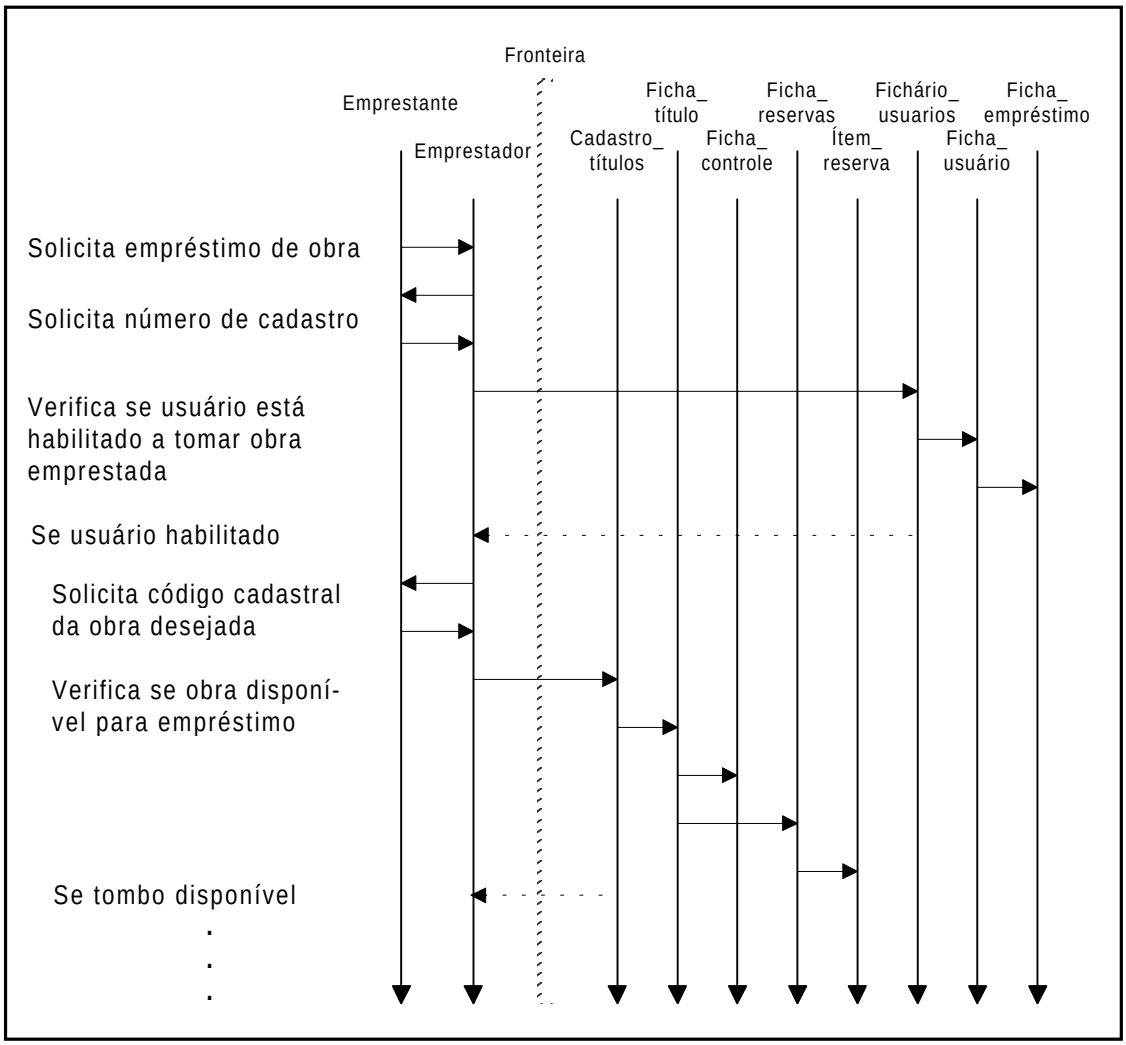



Figura 2.17 - Parte do diagrama de interação para empréstimo*

- Fases - os modelos são descritos em relação a dimensão em cada fase do desenvolvimento[7]: análise (análise de requisitos, também), projeto e outras, se o método eventualmente modelar. Para cada fase são apresentadas as dimensões dos modelos: estática, dinâmica, funcional e de interface. Os modelos e diagramas de cada método são descritos resumidamente, apresentando suas características semânticas e sintáticas e referências a parte dos modelos e diagramas gerados como estudo são fornecidas, os quais estão por completo no apêndice C.
- Transição entre as fases - descreve como se estabelece a evolução entre as fases que o método aborda proporcionando uma melhor compreensão do método.

Posteriormente são descritos a Aplicabilidade, os Pontos positivos e os Pontos negativos de cada método.

O *framework* adotado é ilustrado pela figura 2.18 apresentada a seguir.

[7] O destaque maior está para a análise e projeto, pois todos os métodos modelam essas fases. A sub-fase da análise, a análise de requisitos será abordada separadamente, e comentários sobre outras fases modeladas serão feitos.

1. Descrição
2. Características
3. Fases
   a) Análise de requisitos
   b) Análise
   c) Projeto
   d) Outras
4. Transição entre as fases
5. Aplicabilidade
6. Pontos positivos
7. Pontos negativos

Figura 2.18 - *Framework* adotado para descrição dos métodos

## *2.4 Considerações finais*

Neste capítulo foram apresentados os conceitos de modelagem de sistemas, os conceitos básicos de orientação a objetos, bem como os conceitos avançados, utilizados pelos métodos, visando esclarecer a base conceitual em que está fundamentado este trabalho, facilitando a compreensão do que será abordado nos próximos capítulos. Foi apresentado ainda um *framework* para possibilitar a descrição dos métodos orientados a objetos, realizada no próximo capítulo.

É importante ressaltar que muitas vezes, a diferente nomeação para o mesmo conceito entre os diversos métodos gera confusão e dificuldade para o aprendizado do mesmo. Uma padronização neste sentido facilitaria o estudo e a compreensão.

# Capítulo 3

## *Métodos orientados a objetos*

Neste capítulo pretende-se discorrer sobre aspectos relacionados aos métodos orientados a objetos, tendo como base o *framework* definido no capítulo anterior. Nas seções subsequentes são apresentados: i) os critérios que foram utilizados para se fazer a escolha dos métodos tratados neste trabalho, ii) a contextualização do sistema utilizado como exemplo para modelagem, antes que se inicie a descrição dos métodos, realizada nas cinco seções posteriores e iii) as considerações finais.

### *3.1 Escolha dos métodos*

(Nascimento, 1990) realizou trabalho semelhante a este descrevendo os métodos estruturados. Deste modo, optou-se por pesquisar os métodos orientados a objetos. Além disso, pode-se adicionar um módulo ao trabalho de (Nascimento, 1992) com essa pesquisa.

Inicialmente identificou-se os métodos orientados a objetos mais divulgados e citados em bibliografias e trabalhos comparativos, tais como: (Silva, 1996), (Andersson, 1995), (Brinkkemper, 1995) e (Marques), os quais são listados na primeira coluna da tabela 3.1.

Na primeira linha da tabela 3.1, temos as características:

1) cobertura do método, identificando quais fases do ciclo de desenvolvimento (análise de requisitos (Req.), análise (An.), projeto (Proj.), implementação (Impl.), teste (Teste) e manutenção (Man.)), que cada método aborda. Como não é possível estudar todos os métodos, são escolhidos os que cobrem, principalmente, análise e projeto;
2) ferramenta genérica, seis ferramentas CASE mais divulgadas, em revistas, como: *Object Magazine* - (Patterns, 1997) -, *Byte* - (Business, 1996) -, *Software Development* - (Choosing, 1995) - e *Dr. Dobb's Journal* - (Microkernels, 1994), (Cross, 1995) e (Encryption, 1997) - e no meio profissional (em *sites* da Internet), foram analisadas, obtendo os métodos para os quais havia suporte. Isto significa que o método é bem divulgado e aceito como eficiente. As ferramentas CASE se encontram na tabela 3.2;
3) foi verificado se o método faz parte de algum método integrador, assim verifica-se que o método é aceito por outro metodologista, confirmando sua eficiência. Os métodos integradores observados se encontram na tabela 3.3.

*Tabela 3.1*
*Maturidade dos métodos básicos orientados a objetos*

| **Método**<br>**Autor(es)** | **Cobertura do método** | **Ferramenta genérica** | **Parte de métodos integradores** | **Total** |
|---|---|---|---|---|
| ***Designing Object Oriented Software***<br>*R. Wirfs-Brock et.al.* | Req. Proj. Impl. Teste Man. | - | - | 5 |
| ***Entity Relationship Object Oriented Specification***<br>*Research Group* | An. Proj. Impl. | - | - | 3 |
| ***Methodology for Object-Oriented Software Engineering of Systems***<br>*B. Henderson-Sellers & J. M. Edwards* | An. Proj. Impl. | - | - | 3 |
| ***Object Modelling Technique***<br>*J. Rumbaugh et. al.* | Req+. An. Proj. Impl. | 1 - 2 - 3 - 4 - 5 - 6 | 4 | 13,5 |
| ***Object Oriented Analysis and Design***<br>*G. Booch* | Req. An. Proj. Impl. Man. | 1 - 2 - 3 | 3 | 11 |
| ***Object Oriented Analysis and Design***<br>*J. Martin & J. Odell* | An. Proj. | - | - | 2 |
| ***Object-Oriented Analysis - Object-Oriented Design***<br>*P. Coad & E. Yourdon* | An. Proj. | 1 - 3 - 5 - 6 | - | 6 |
| ***Object Oriented Information Engineering***<br>*J. Martin & J. Odell* | An. Proj. Impl. | 1 | - | 4 |
| ***Object Oriented System Analysis***<br>*S. Shlaer & S. Mellor* | An. | 1 - 3 - 4 | - | 4 |
| ***Object-Oriented System Development***<br>*D. Champeaux et. al.* | Req+. An. Proj. Impl. | - | - | 3,5 |
| ***Object Oriented Software Engineering***<br>*I. Jacobson et. al.* | Req. An. Proj. Impl. Teste | 1 - 3 - 6 | 2 | 10 |

Para cada uma das características foram atribuídos pontos da seguinte maneira:

1) cobertura do método: 1 ponto para cada fase, (Req+[8] , 0,5);
2) ferramenta genérica: 1 ponto para cada ferramenta que aborda o método;
3) parte de métodos integradores: 1 ponto para cada método integrador, para o qual o método faça parte.

De acordo com a pontuação definida anteriormente, somando-se as três colunas internas na tabela 3.1, obtém-se um total de pontos de cada método, identificado na coluna Total, onde pode-se notar que os métodos mais pontuados são:

*(13,5) - Object Modelling Technique* - James Rumbaugh et. al.

*(11) - Object Oriented Analysis and Design* - Grady Booch

*(10) - Object Oriented Software Engineering* - Ivar Jacobson et. al.

*(6) - Object-Oriented Analysis - Object-Oriented Design* - Peter Coad & Edward Yourdon.

É importante ressaltar que esta pontuação não define o método com a maior pontuação como o melhor método. Existem vários aspectos como: modelos criados, notação oferecida, tratamento das fases do desenvolvimento e a transição entre elas; que são os que devem ser levados em consideração na escolha de um método. E este é o estudo que é feito neste trabalho.

A tabela 3.2, como dito anteriormente, apresenta as seis ferramentas CASE mais amplamente divulgadas. Na primeira coluna desta tabela tem-se o nome das ferramentas CASE, na segunda coluna encontram-se os fabricantes de cada uma das ferramentas, seguido pelo endereço na Internet, onde podem ser encontradas informações sobre a ferramenta.

*Tabela 3.2*
*Ferramentas CASE que suportam métodos orientados a objetos*

| Nr. | *Ferramenta CASE* | *Nome do fabricante* | *Endereço Internet* |
|---|---|---|---|
| **1** | ***Paradigm Plus*** | ***Platinum Technology*** | http://www.platinum.com |
| **2** | ***Rational Rose*** | ***Rational Software Corporation*** | http://www.rational.com |
| **3** | ***System Architect*** | ***Popkin Software & Systems, Inc.*** | http://www.popkin.com |
| **4** | ***ObjectTeam*** | ***Cayenne Software, Inc.*** | http://www.cayennesoft.com |
| **5** | ***Together*** | ***Object International, Inc.*** | http://www.oi.com |
| **6** | ***Select Perspective*** | ***SELECT Software Tools*** | http://www.pmp.co.uk |

A tabela 3.3 apresenta os métodos integradores, que são: *Catalysis*, *Fusion*, *Syntropy* e *Objectory/Unified Method*. A primeira coluna dessa tabela mostra o nome do método integrador e seu(s) autor(es), e na segunda coluna os métodos utilizados para compor o método integrador. Por exemplo: o método Fusion agrega os métodos OMT, OOD, métodos formais e a técnica CRC.

O método Fusion que é de segunda geração também foi escolhido, por ser um método já divulgado e bem estabelecido, sendo também considerado um método utilizado para compor outro

[8] A fase que contém um sinal + seguido, significa que o método referencia a fase fornecendo poucas explicações ou detalhamentos de como o gerente do desenvolvimento deve proceder naquela fase.

método: o *Catalysis*. Diferentemente ao método *Rational Objectory Method* - Jacobson, Rumbaugh e Booch, que ainda se encontra em fase de desenvolvimento.

Assim, o método Fusion é acrescentado à lista de métodos anteriormente mencionada, para fechar o grupo de métodos que serão alvo deste estudo.

*Tabela 3.3*
*Métodos integradores*

| **Método integrador**<br>**Autor(es)** | **Métodos utilizados** |
|---|---|
| ***Catalysis***<br>*D. D'Souza & A. Wills* | *OOSE (Jacobson) / OMT (Rumbaugh) / Fusion (Coleman)* |
| ***Fusion***<br>*D. Coleman et.al.* | *OMT (Rumbaugh) / OOD (Booch) / Técnica CRC (Wirfs-Brock) / Métodos Formais* |
| ***Syntropy***<br>*Syntropy User Group* | *OMT (Rumbaugh) / OOAD (Booch)* |
| ***Rational Objectory Method***<br>*I. Jacobson - J. Rumbaugh - G. Booch* | *OOSE (Jacobson) / OMT (Rumbaugh) / OOAD (Booch)* |

Após a exposição dos critérios utilizados para a escolha dos métodos, é apresentada a contextualização tentando esclarecer alguns pontos importantes.

## *3.2 Contextualização*

Nas seções que se seguem, os métodos orientados a objetos serão descritos, tendo como nivelador o *framework* definido no capítulo anterior. Para que seus conceitos e características, bem como suas vantagens e desvantagens sejam identificados com clareza, será utilizado a modelagem de um sistema de biblioteca[9], o qual terá suas partes mais significativas[10] modeladas, para melhor esclarecimento e comparação dos métodos.

Essa modelagem tem o objetivo maior de fornecer aos pesquisadores, uma melhor compreensão dos modelos e diagramas utilizados pelos métodos. Isto permitirá um maior embasamento para que possa ser feita a comparação entre os métodos no capítulo 4.

O levantamento de requisitos do sistema de biblioteca se encontra no apêndice B - Requisitos do sistema de biblioteca. Todos os modelos e diagramas gerados durante o estudo dos métodos se encontram no apêndice C em seções separadas por métodos. Eles são utilizados como exemplos nas próximas seções para ilustrar conceitos e diagramas dos métodos.

[9] O sistema de biblioteca será, também, modelado por duas alunas de graduação deste departamento, sob orientação do prof. Dr. Wagner Teixeira da Silva. Este trabalho de graduação tem como objetivo a modelagem completa do sistema, o qual deverá ser implementado. Portanto, é um trabalho diferente deste, pois aqui serão modeladas partes do sistema utilizando todos os métodos estudados neste trabalho e, não abordará parte de implementação. A única semelhança é a utilização do mesmo sistema, que ainda, possivelmente, conterá requisitos diferentes.

[10] Partes mais significativas, quer dizer, por exemplo, que quando o modelo dinâmico do modelo OMT (Rumbaugh) tiver que ser feito, será escolhido somente a parte mais dinâmica do sistema. Nos grafos de transição de estado do método OOSE (Jacobson), será modelado o objeto com maior mudança de estados.

As seções que se seguem descrevem os métodos orientados a objetos abordados neste trabalho, tendo como base o *framework* definido na seção 2.4 do capítulo anterior. Ele apresenta os seguintes itens: descrição, características, fases (análise de requisitos, análise, projeto e outras), transição entre as fases, aplicabilidade, pontos positivos e negativos.

## *3.3 OOSE - Object Oriented Software Engineering*

### *3.3.1 Descrição*

OOSE - *Object Oriented Software Engineering* é um método orientado a objeto desenvolvido por Ivar Jacobson, Magnus Christerson, Patrik Jonsson e Gunnar Övergaard em 1992.

O método OOSE é o mais abrangente dentre os que são abordados neste trabalho, pois fornece todo um conjunto de modelos e diagramas, que permitem a modelagem do sistema durante as seguintes fases: análise de requisitos, análise, projeto, implementação e teste. As duas primeiras fases são agrupadas e tratadas pelo subprocesso Análise, as duas fases seguintes estão sob controle do subprocesso Construção e a última fase, sob o controle do subprocesso Teste, como mostra a figura 3.1.

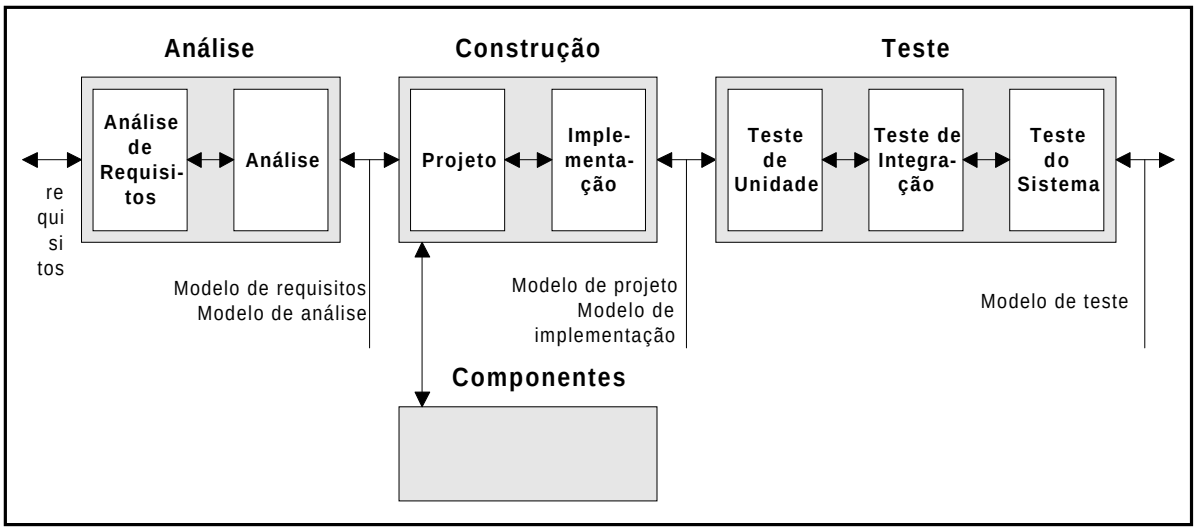



Figura 3.1 - Fases do método OOSE

Estes subprocessos geram cinco modelos diferentes: modelo de requisitos, modelo de análise, modelo de projeto, modelo de implementação e modelo de teste. Os requisitos do sistema servem como base para o primeiro subprocesso análise. Ao final do processo, obter-se-á um sistema desenvolvido e testado.

O subprocesso Construção utiliza um repositório chamado Componentes, para o reuso de códigos fontes de partes de projetos já desenvolvidos. Um componente é uma unidade padrão construída em uma organização, que é usada para desenvolver aplicações (Jacobson, 1992). É necessário que ele seja bem testado, eficiente e bem documentado, além de possuir uma interface bem projetada. Um componente pode ser um *white-box* (caixa branca) ou um *black-box* (caixa

preta), os quais estarão prontos para serem reutilizados. Entretanto, a diferença básica entre *white-box* e *black-box* se encontra no fato de que o código fonte da primeira pode ser modificado, aumentado ou diminuído para adaptar ao propósito de uma parte do projeto em questão. E o código fonte do segundo não pode ser de forma alguma alterado, devendo ser reutilizado como foi projetado, mas os parâmetros passados ao componente podem representar o conteúdo necessário para satisfazer uma determinada implementação.

### *3.3.2 Características*

A abordagem desse método é orientada a comportamento, tendo como característica marcante a identificação das funcionalidades do sistema, na forma de *use cases* (casos de uso). Eles conduzem todo o desenvolvimento do sistema, o que permite uma coerência entre os modelos gerados nas diversas fases, pois todos estarão centrados em refiná-los.

Apesar do método possuir uma larga abrangência das fases do ciclo do desenvolvimento, fornecendo um vasto conjunto de modelos e diagramas, seu foco está na fase de análise, especificamente na análise de requisitos, quando são gerados vários modelos: modelo de objetos do domínio, modelo *use case* e modelo de interfaces, que serão descritos com mais detalhes na seção 3.4.3.

### *3.3.3 Fases*

**a) Análise de Requisitos**

Durante a análise de requisitos desenvolvem-se três modelos: i) modelo *use case* e as descrições de cada *use case*; ii) o modelo de objetos do domínio; e iii) o modelo de interface.

O modelo *use case* é gerado para especificar as funcionalidades do sistema a partir da perspectiva do usuário. O modelo é desenvolvido em duas fases: o primeiro modelo desenvolvido retrata o curso básico, a seqüência ideal sem ocorrência de erros. O segundo modelo, que representa a evolução do primeiro, descreve os cursos alternativos, as variações do curso básico e a ocorrência de erros.

A figura 3.2 apresenta uma parte do modelo *use case* básico, onde contém três *use case* (Cadastrar usuário, Emprestar obra e Cancelar reserva) e os atores que interagem com eles (Usuário, Cadastrador, Emprestador e Emprestante).

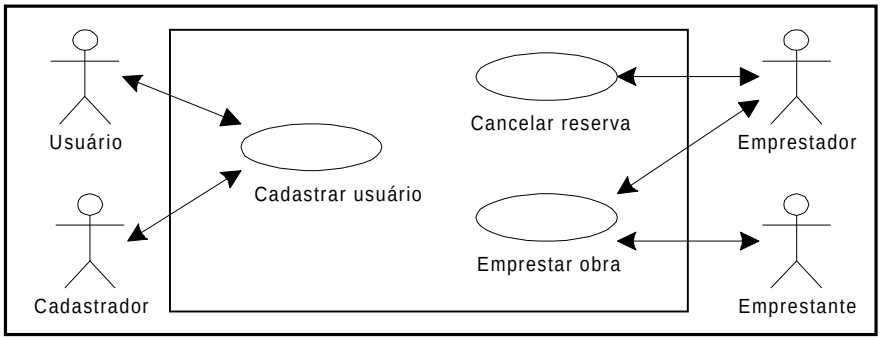



Figura 3.2 - Parte do modelo *use case*

Para cada *use case* é necessário que se faça uma descrição sobre suas entradas, restrições, validações e saídas. A figura 3.3 apresenta a descrição do *use case* Emprestar obra.

> O funcionário (emprestador) solicita empréstimo de obra ao sistema. O sistema solicita ao emprestador o número de cadastro do emprestante, para verificar se o mesmo está apto a fazer empréstimo. Esta verificação é feita observando se o emprestante já excedeu o limite máximo para empréstimo, se possui alguma dívida pendente e se está com alguma obra emprestada cuja data de devolução já esteja vencida.
>
> Se o usuário estiver apto, o sistema solicita a obra e o usuário fornece o seu código cadastral. O sistema verifica se o código cadastral é válido, e é verificado, também, se existe algum tombo daquela obra disponível, se existir, verifica se existe reserva para o tombo. Se existir reserva, observar se o reservante é o mesmo usuário que o emprestante, se for, o tombo é liberado, se não for, o tombo não pode ser liberado e toda verificação é feita para outro tombo. Se algum tombo for liberado, é solicitado ao emprestante sua senha, que é verificada na ficha do usuário.
>
> Se a senha estiver correta, o sistema preenche a ficha de empréstimo e atualiza a ficha de controle do tombo, a ficha de usuário e a ficha de tombo. O emprestador libera a obra ao solicitante.

Figura 3.3 - Descrição do *use case* - emprestar obra

A figura 3.4 apresenta dois tipos de relacionamentos entre *use cases*: a) relação de extensão e b) relação de uso.

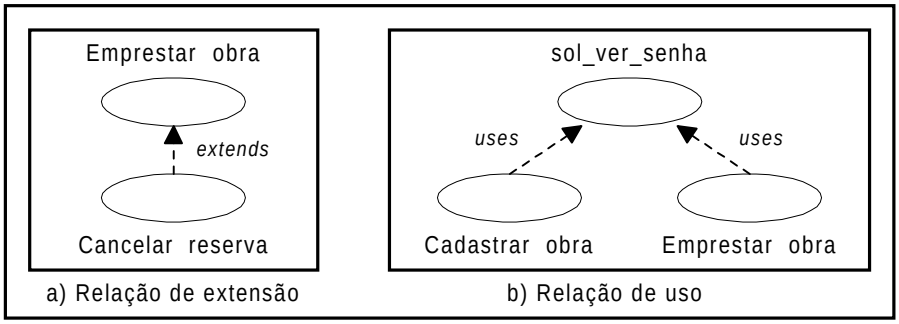



Figura 3.4 - Tipos de relacionamentos entre *use cases*

A figura 3.4.a) modela o *use case* Emprestar obra, que é inteiramente independente. Em alguns casos, o *use case* Cancelar reserva é ativado estendendo o *use case* Emprestar obra, isto modela o caso onde o emprestante possui uma reserva. A relação de extensão é utilizada para modelar: i) partes opcionais de *use cases*, ii) cursos complexos e alternativos, iii) sub-cursos separados, que são executados somente em certos casos, como no caso da figura 3.4.a) e iv) situação onde vários *use cases* podem ser inseridos em um *use case* especial.

A figura 3.4.b) modela os use cases Cadastrar obra e Emprestar obra, tendo uma relação de uso com sol_ver_senha (solicita e verifica senha). O objetivo é obter procedimentos comuns existentes entre dois ou mais *use cases* para criar um *use case* comum, que será utilizado pelos outros *use cases*.

O modelo de objetos do domínio serve como base comum de entendimento entre todas as pessoas envolvidas no desenvolvimento - analistas, projetistas, especialistas do domínio e usuários - . Este modelo engloba os objetos do domínio e os relacionamentos entre eles.

Objetos do domínio são aqueles objetos identificados no domínio da aplicação, não fazendo parte os objetos de implementação.

A figura 3.5 apresenta parte do modelo de objetos do domínio do sistema de biblioteca. Os objetos do domínio são: Acervo_obras, Obra, Ficha_controle, Cadastro_títulos, Ficha_título, Ficha_tombo e Ficha_empréstimo. Os relacionamentos são unidirecionais e os que são nomeados consiste-de são um tipo especial chamado de agregação. Pode-se verificar que todos os objetos pertencem ao domínio de uma biblioteca.

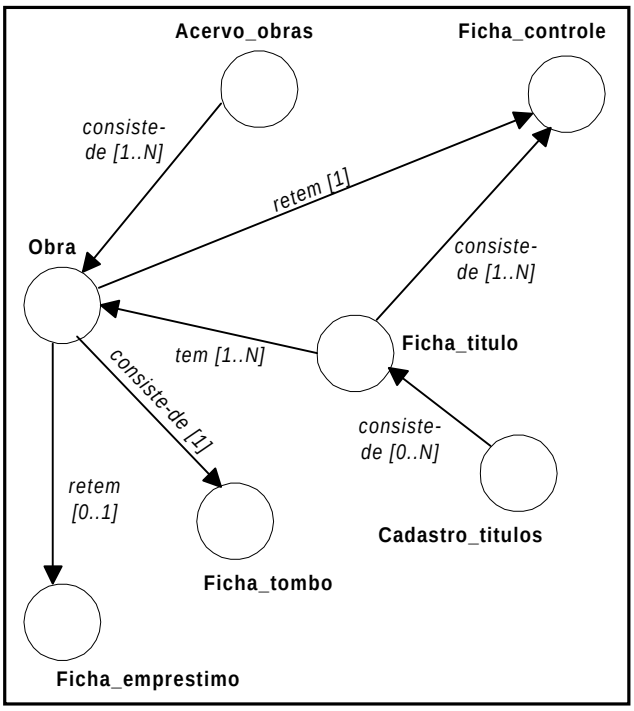



Figura 3.5 - Parte do modelo de objetos do domínio

O próximo passo é desenvolver o modelo de interface, que é criado com as descrições de interface, que conterão especificações detalhadas da interface do sistema. Entretanto, o método não apresenta um diagrama mais formal, mas deixa claro que as descrições devem estar consistentes, retratando a mesma idéia que o desenvolvedor e o usuário têm do sistema. Como sugere os exemplos do livro (Jacobson, 1992), estas descrições mostram as telas referentes a cada *use case*.

A figura 3.6 apresenta um exemplo de descrição de interface durante a realização de um empréstimo, a qual deve acompanhar o *use case* Emprestar obra.

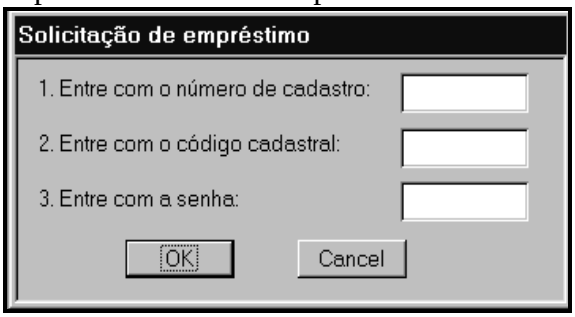



Figura 3.6 - Descrição de interface

**b) Análise**

Na análise desenvolve-se um modelo estático, o modelo de análise, que é baseado no modelo de requisitos. O modelo de análise descreve o sistema usando três tipos de objetos: objeto entidade, objeto de controle e objeto de interface, que garantem a robustez do sistema. O objetivo é distribuir o comportamento especificado nas descrições de *use case* entre os três tipos de objetos do modelo de análise, sendo que um *use case* é trabalhado por vez. Os objetos entidade são objetos do domínio, os objetos de interface e de controle são uma invenção de objetos que são definidos pelos outros métodos na fase de projeto. Deste modo, há uma antecipação na definição destes objetos.

O método classifica os objetos em três tipos: objeto entidade onde se armazena toda informação do sistema por um longo tempo, objeto de interface que modela os procedimentos de interface entre o sistema e o usuário e objeto de controle que modela as várias operações necessárias (recuperação, atualização, inclusão), realizadas sobre os objetos entidades, retornando resultados aos objetos de interface. A figura 3.7 ilustra os três tipos de objetos mencionados.

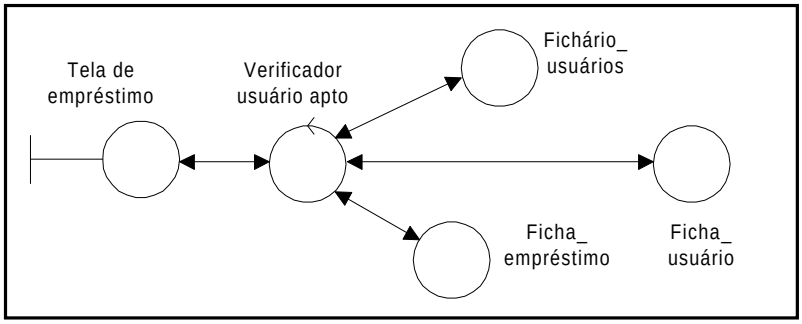



Figura 3.7 - Parte do modelo de análise para o *use case* - emprestar obra

O objeto Tela de empréstimo é um objeto de interface, obtendo dados do usuário e apresentando a ele os resultados (as mensagens) das verificações necessárias para se fazer o empréstimo.

O objeto Verificador usuário apto é um objeto de controle que acessa vários objetos entidade para concluir sua tarefa de verificar se o usuário está apto a fazer empréstimo. Os objetos entidade acessados por este objeto de controle são: fichário_usuários, ficha_usuário e ficha_empréstimo, que armazenam os dados.

Os objetos entidade são exatamente os objetos do domínio identificados durante o desenvolvimento do modelo de objetos do domínio na sub-fase de análise de requisitos. Assim, os objetos de controle e objetos de interface são considerados extra-domínio, ou seja, são inventados pelo analista, o objetivo é facilitar futuras evoluções e manutenções no sistema, pois assim o sistema se torna robusto e estável.

(Jacobson, 1992) argumenta, que as mudanças mais comuns em um sistema são de sua funcionalidade ou de sua interface. As modificações de interface afetam os objetos de interface, entretanto as modificações de funcionalidade podem afetar os três tipos de objeto: o objeto de interface quando as funcionalidades estão relacionadas a interface, o objeto entidade quando a

funcionalidade está encapsulada no objeto, ou o objeto de controle quando a funcionalidade é entre os objetos. Entretanto, segundo (Jacobson, 1992), a maioria das modificações não afetam os objetos entidade.

Juntamente com a identificação dos objetos pode ser feita a identificação dos atributos dos objetos entidades, e modelar como mostra a figura 3.8. Abaixo da representação para os atributos coloca-se qual é o seu tipo, e no relacionamento entre o objeto e o atributo coloca-se o nome e a cardinalidade do atributo.

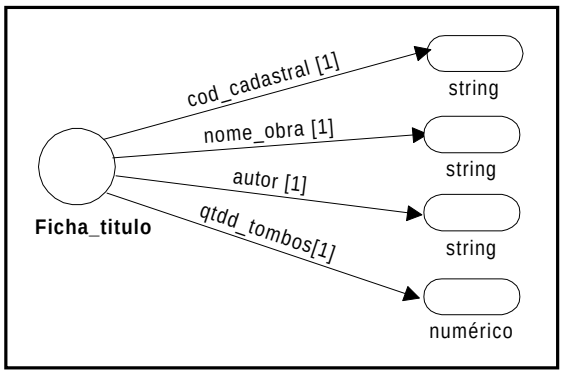



Figura 3.8 - Modelagem dos atributos do objeto entidade - ficha_título

Após a identificação de todos os objetos no modelo de análise, são feitas as descrições de cada um deles. O método não apresenta um procedimento a ser seguido. A figura 3.9 apresenta a descrição de um objeto de controle, o Concluidor de empréstimo.

***Objeto de controle: Concluidor de empréstimo***

1o. Adiciona 1 a cota_emprestada no objeto ficha_usuário.
2o. Altera o status_obra no objeto ficha_controle.
3o. Altera objeto ficha_empréstimo.
4o. Altera dt_devolução no objeto ficha_tombo.

Figura 3.9 - Descrição do objeto de controle - concluidor de empréstimo

As descrições contêm o que cada objeto faz de acordo com o seguinte:

i) objeto entidade: quais dados armazena e quais métodos oferece;
ii) objeto de controle: acessar outros objetos fazendo verificações ou buscando dados (objetos entidade) ou devolvendo o controle (objetos de interface);
iii) objeto de interface: solicitar verificações e mostrar dados aos usuários.

Depois que todos os modelos estiverem prontos e todos os objetos definidos é necessário a identificação de subsistemas, para que se possa reduzir a complexidade. Um subsistema pode ser composto por outro subsistema, como a idéia de recursão, até que se obtenha unidades atômicas. O objetivo é ter um forte acoplamento funcional dentro do subsistema e um fraco acoplamento entre os subsistemas, caracterizando que a identificação do subsistema está correta.

**c) Projeto**

Na fase de projeto é desenvolvido um modelo estático, denominado modelo de projeto. Este modelo refina o modelo de análise, definindo a interface dos objetos e também a semântica das operações e decidindo sobre aspectos do ambiente de implementação atual. Como exemplo pode-se citar: banco de dados a ser utilizado, verificar se a linguagem de programação suporta tudo que foi modelado na análise (especialmente associação de herança), dividir os blocos para distribuí-los no computador.

A figura 3.10 apresenta parte do modelo de projeto referente a parte do modelo de análise apresentado na figura 3.7, onde foi feita somente a passagem da análise para o projeto sem considerações do ambiente atual.

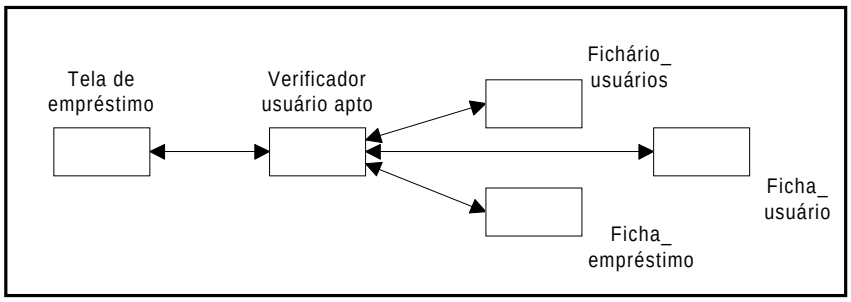



Figura 3.10 - Parte do modelo de projeto

Durante o projeto, a modelagem dinâmica representa a comunicação entre os objetos - envio de mensagens - que pertencem a determinado *use case*, gerando o diagrama de interação. Portanto, tem-se um diagrama de interação para cada *use case* identificado.

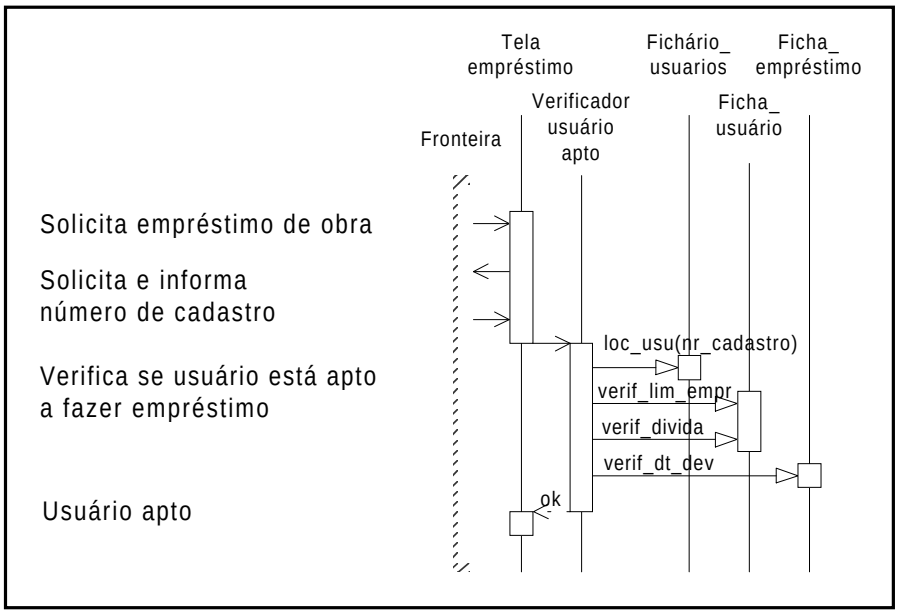



Figura 3.11 - Parte do diagrama de interação para o *use case* - emprestar obra

Através da figura 3.11 pode-se observar a descrição textual das mensagens e sinais do lado esquerdo e os objetos (de controle, de interface e entidade) na parte superior do diagrama. Os sinais, que ocorrem entre processos, estão representados por tudo que acontece entre a fronteira do sistema e o objeto de interface Tela de empréstimo, e entre esse e o objeto de controle Verificador de usuário apto. O restante são as mensagens entre o objeto de controle e os objetos entidade, que ocorrem dentro do processo. Os nomes que vão acima das mensagens são as operações a serem ativadas e entre parênteses seus parâmetros.

Após a criação dos diagramas de interação para todos os *use cases*, é possível identificar as operações de cada objeto através desses diagramas. Por exemplo, para o diagrama da figura 3.11, o objeto Ficha_usuário tem as operações verif_lim_empr e verif_dívida.

Além do diagrama de interação, ainda na fase de projeto, são criados os grafos de transição de estado. Eles fornecem uma descrição simplificada dos blocos, que é menos dependente da linguagem de programação escolhida, o que aumenta o entendimento do bloco, porque não é necessário descer ao nível do código. (Jacobson, 1992) sugere quatro formas diferentes para se desenvolver os grafos: duas formas gráficas simples e comuns (utilizadas por outros autores), uma forma descritiva e uma forma gráfica detalhada, que é adotada para exemplos do livro. Um exemplo é apresentado pela figura 3.12.

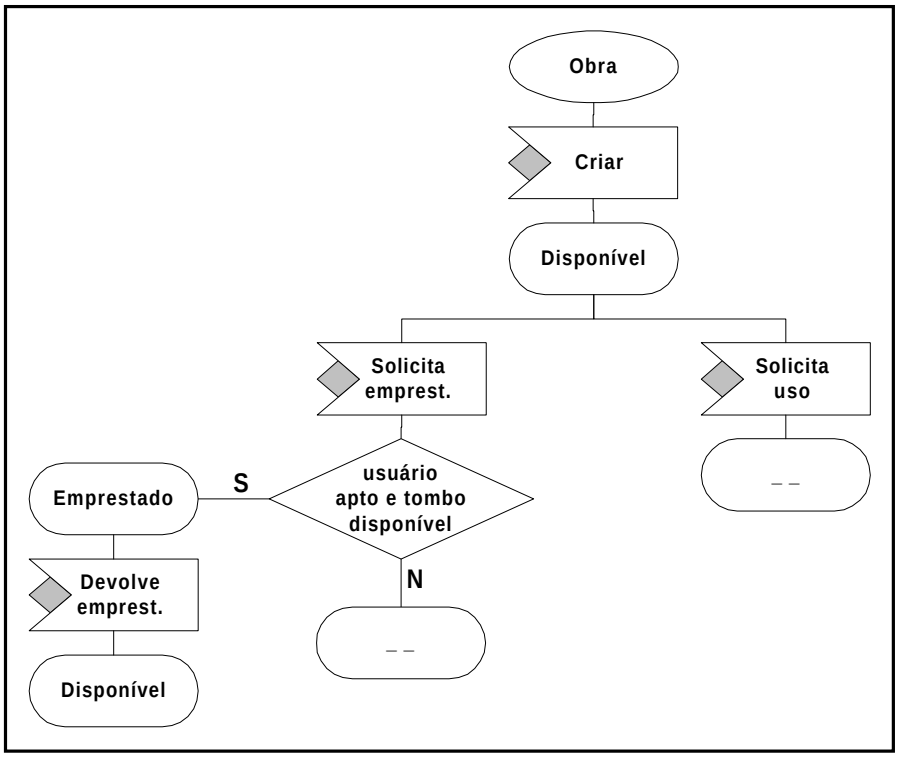



Figura 3.12 - Parte do grafo de transição de estado do objeto: Obra

O objeto Obra é criado e passa para o estado Disponível. Se algum usuário solicitar o uso, então a obra volta ao estado Disponível. Se um usuário solicita empréstimo, é feita verificação de usuário e tombo. Se o usuário não estiver apto ou o tombo não estiver disponível, então a Obra

continua no estado Disponível. Se o usuário estiver apto e o tombo estiver disponível, então a Obra passa ao estado Emprestado. Quando a Obra é devolvida, o objeto passa ao estado Disponível.

**d) Outras**

Este método, como já mencionado, aborda as fases de implementação e teste, que não serão levadas em consideração para o estudo nem comparação por não ser comum em todos os métodos. A fase de implementação é a transformação do modelo de projeto em linguagem de programação (códigos fontes) e durante a fase de teste, tem-se três etapas: o teste de unidades para verificar se as funções e procedimentos estão corretos, o teste de integração para testar se as diferentes unidades estão trabalhando juntas com sucesso e o teste de sistema para verificar se o sistema funciona como um todo.

### *3.3.4 Transição entre as fases*

(Jacobson, 1992) argumenta que: *A transição do modelo de análise para o modelo de projeto deve ser feita quando as conseqüências do ambiente de implementação começarem a aparecer*. Embora essa transição não seja clara, muitas vezes, o analista deve observar quando os aspectos do ambiente de implementação começam a influenciar a análise, tais como: "a linguagem não implementa este recurso!", ou ainda, "o *hardware* não suporta essa implementação!"; neste caso deve-se iniciar o projeto.

É importante notar que o método OOSE sugere que os objetos da análise se tornem objetos de projetos (blocos) e, consequentemente, de código fonte. Assim, existirá a rastreabilidade entre os modelos facilitando uma futura manutenção.

### *3.3.5 Aplicabilidade*

O método pode ser aplicado para o desenvolvimento de sistemas técnicos e administrativos, reutilizando descrições e componentes já definidos. Ele também apresenta características favoráveis a sistemas que necessitem da participação de usuários e especialistas do domínio no detalhamento dos requisitos, devido a utilização de *use cases*, atores e diagramas de interação.

O método é indicado para sistemas com o enfoque principal em suas funcionalidades, onde o modelo *use case* será de grande utilidade, por ter todo o processo baseado nessas funcionalidades e possuir sintática e semântica rica para modelagem.

Além disso, o método pode ser utilizado no desenvolvimento de sistemas reais de grande porte, por isso é indicado para ser usado por uma equipe de desenvolvimento e não por um único desenvolvedor. É um método abrangente com sintática e semântica bem específica em cada fase do desenvolvimento, sendo que as equipes devem ter treinamento em análise de requisitos, análise, projeto e programação.

### *3.3.6 Pontos positivos*

A utilização de três tipos de objetos facilita evoluções e modificações futuras no sistema, pois o sistema se torna robusto e estável tendo poucas modificações nos objetos do domínio.

Outro ponto que vale destacar, refere-se ao diagrama de interação, que possui descrições textuais no lado esquerdo ao gráfico, facilitando sua compreensão. Existe também, a distinção entre uma mensagem, enviada dentro de um processo, e um sinal enviado entre dois processos.

O interessante do grafo de transição de estado é que, além de se perceber os estados possíveis do objeto, também pode-se ver claramente as ações executadas nas transições de estados e as condições para as transições.

### *3.3.7 Pontos negativos*

A notação de atributos utilizada pelo método é, sem dúvida, algo que aumenta a complexidade do modelo. Não sendo viável a visualização de todos os atributos dos objetos em um mesmo modelo, porque torna-se difícil a visualização do modelo como um todo.

O método OOSE é muito extenso, englobando análise (incluindo análise de requisitos), projeto, implementação e teste. Isso na verdade, pode se tornar um ponto negativo ou positivo, dependendo do desenvolvimento. Será um ponto negativo, quando se tem: um sistema muito simples, uma equipe de desenvolvimento inexperiente ou um único desenvolvedor. Será um ponto positivo, quando se tem um sistema muito grande e complexo, necessitando ser bem especificado e documentado.

O método não é prescritivo, ou seja, não fornece passos a serem seguidos para facilitar o seu uso. Além disso, é necessário muito estudo para se compreender os conceitos adotados e os objetivos de cada modelo.

Um ponto não muito claro, diz respeito a utilização da palavra objeto para discriminar tanto o conceito de objeto como o de classe. Existem partes da descrição do método em que é difícil perceber quando o autor se refere a classe, a uma instância (um objeto) ou as instâncias da classe (referindo a todos os objetos instanciados de uma classe).

## *3.4 OMT - Object Modeling Technique*

### *3.4.1 Descrição*

OMT - *Object Modeling Technique* é um método orientado a objeto, desenvolvido por James Rumbaugh, Michael Blaha, Willian Premerlani, Frederick Eddy e Willian Lorensen em 1991. *As bases para a notação do OMT foram desenvolvidas a mais de 10 anos (antes de 1988) com Mary Loomis e Ashwin Shah da Corporação Calma* (Rational, 1997).

Esse método aborda as fases de análise, projeto e implementação, como mostra a figura 3.13. A fase de análise produz três modelos: modelo de objetos, modelo dinâmico e modelo

funcional. A fase de projeto é dividida em projeto de objeto e projeto do sistema, que estende os modelos de análise com detalhes de projeto.

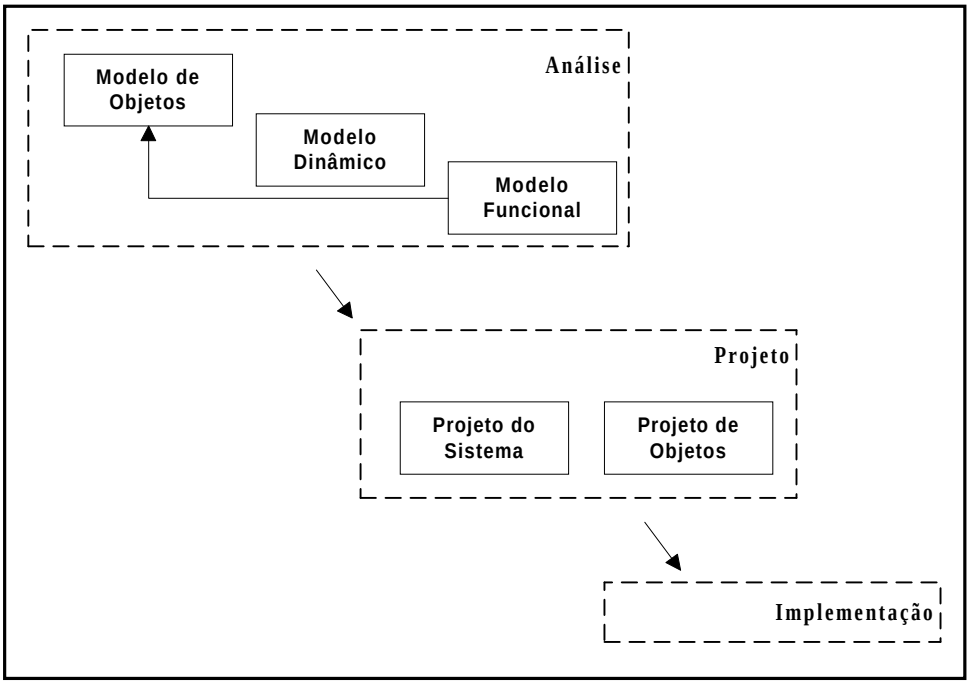



Figura 3.13 - OMT - *Object Modeling Technique* (Coleman, 1996)

### *3.4.2 Características*

A abordagem utilizada pelo método é a orientada a dados, pois no início do desenvolvimento o método se preocupa, principalmente, com a identificação dos atributos das classes, só se preocupando com as operações na fase de projeto.

O foco desse método está na fase de análise, onde se obtém um grande conjunto de modelos e diagramas e uma seqüência de passos a serem seguidos, que facilitam a modelagem nesta fase.

(Rumbaugh, 1994) apresenta os conceitos de orientação a objetos com ótimas explicações e bastante exemplificações, o que facilita o estudo dos leigos na área de orientação a objetos.

### *3.4.3 Fases*

**a) Análise de requisitos**

Este método não possui modelos e/ou diagramas para a modelagem da análise de requisitos, devido ao fato de que os requisitos devem ser informados pelo usuário sem assistência do método.

**b) Análise**

A modelagem estática na análise é feita através do modelo de objetos, que é representado graficamente pelo diagrama de objetos. Este diagrama apresenta a notação gráfica para modelagem

de objetos e os relacionamentos entre eles. O diagrama de classes e o diagrama de instâncias são tipos de modelo de objetos. O primeiro modela as classes do domínio do problema e o segundo descreve como os objetos se relacionam. Portanto, *um dado diagrama de classes pode gerar um conjunto infinito de diagramas de instâncias* (Rumbaugh, 1994), o que depende da quantidade de instâncias de cada classe.

A figura 3.14 ilustra um exemplo de parte do diagrama de classes desenvolvido para o sistema de biblioteca. Este diagrama descreve as classes (ex.: Ficha_título), os relacionamentos simples (ex.: tem), agregação, herança, cardinalidade, os atributos, restrições e ordenações.

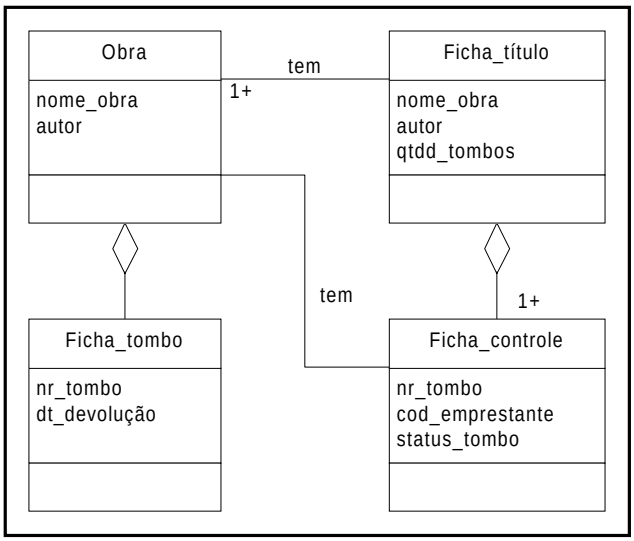



Figura 3.14 - Parte do diagrama de classes

A figura 3.15 apresenta o diagrama de instâncias referente ao diagrama de classes, ilustrado pela figura 3.14. No diagrama de instâncias é modelado além dos valores dos objetos, a quantidade de objetos de um relacionamento com cardinalidade 1+ (expresso no diagrama de classes). Por exemplo, várias Obras iguais possuem uma Ficha_título.

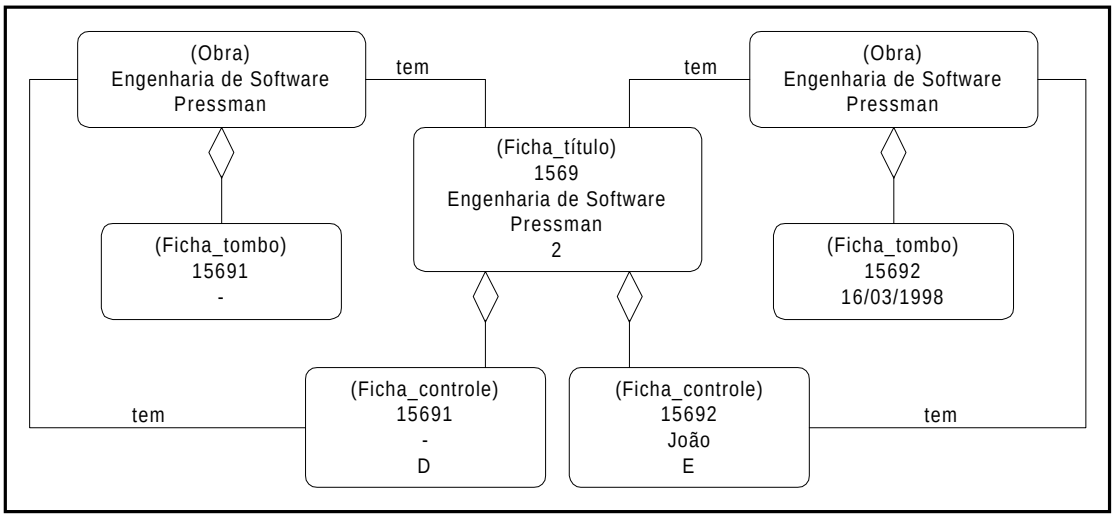



Figura 3.15 - Parte do diagrama de instâncias

No diagrama de instâncias são colocados todos os relacionamentos existentes entre os objetos, ou seja, entre as instâncias das classes que foram definidas no diagrama de classes. Pode-se obter um diagrama bastante complexo dependendo da estrutura das classes, pois uma classe, geralmente, possui várias instâncias.

Ainda durante a análise é necessário a criação de um dicionário de dados, onde todas as classes são descritas quanto a sua abrangência no problema, suas restrições e seu uso. Além disso, as associações, os atributos e as operações também são descritos, entretanto, o método não apresenta um modelo a ser seguido.

Durante a fase de análise são criados quatro diagramas para se fazer a modelagem dinâmica, são eles: cenários, diagramas de eventos, diagrama de fluxo de eventos e diagrama de estado.

Primeiro, são desenvolvidos os cenários contendo a seqüência de eventos, que são intercambiados entre o sistema e um agente externo, descrita de forma textual. Os primeiros cenários são desenvolvidos abrangendo os casos normais, depois há um refinamento, acrescentando, gradualmente, omissão e repetição de dados, e depois os erros e valores inválidos.

A figura 3.16.a) apresenta um cenário relatando uma seqüência normal para o empréstimo de obras, e a figura 3.16.b) ilustra a seqüência do empréstimo com um certo refinamento, apresentando valores inválidos e mensagens de erro.

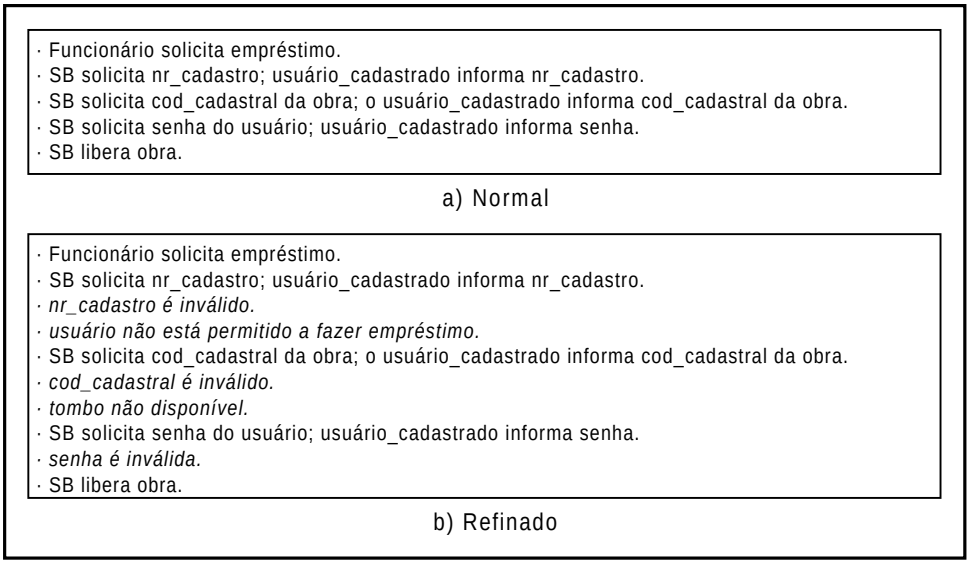



Figura 3.16 - Cenários - a) Normal e b) Refinado

Os cenários são representados graficamente através dos diagramas de eventos, onde os objetos são representados por uma linha vertical e os eventos por setas que interligam esses objetos do emissor para o receptor. Este diagrama, assim como o cenário, preocupa-se com a seqüência dos eventos. A figura 3.17 ilustra parte do diagrama de eventos do sistema de biblioteca, que refere-se a figura 3.16.

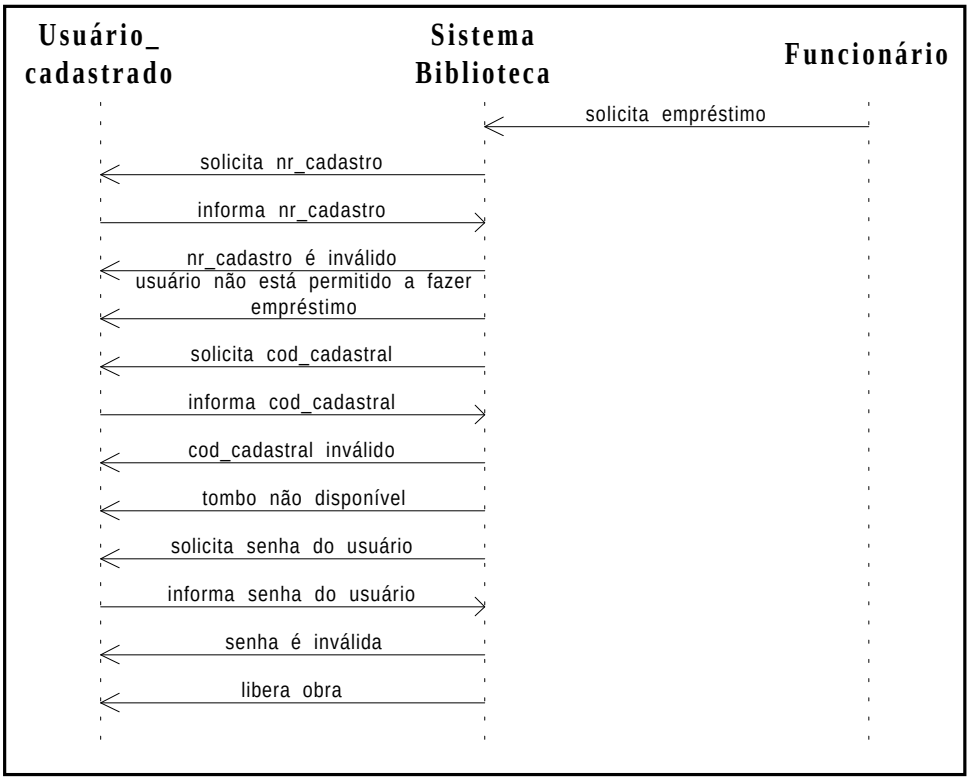



Figura 3.17 - Parte do diagrama de eventos

Todos os eventos de todos os diagramas de eventos são agrupados em um diagrama chamado diagrama de fluxo de eventos, ilustrado pela figura 3.18, o qual não se preocupa com a seqüência. Seu objetivo é visualizar todos os eventos existentes entre os sistema e os agentes externos.

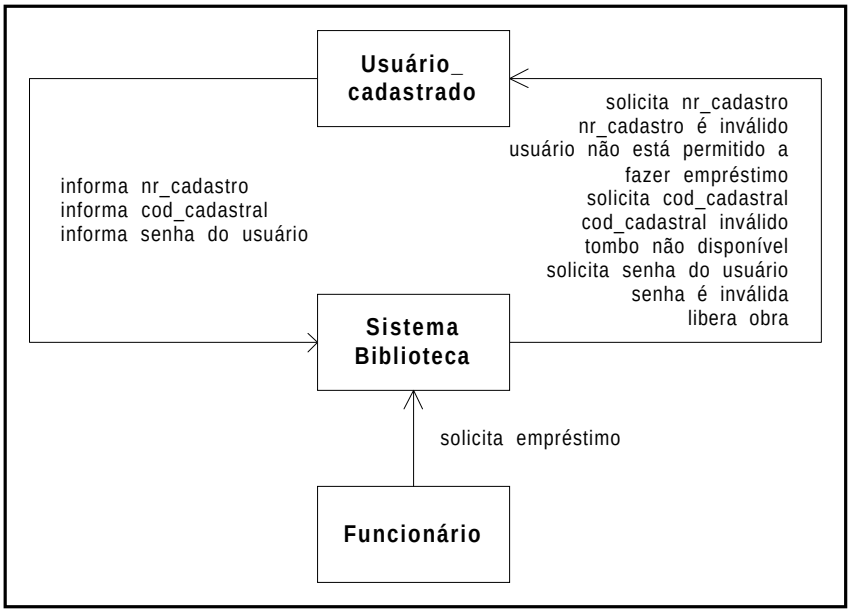



Figura 3.18 - Parte do diagrama de fluxo de eventos

Em seguida, para cada classe que possuir um comportamento dinâmico significativo, será construído um diagrama de estados, que modela os estados possíveis que os objetos de uma classe pode ter e os eventos, condições, ações e atividades que se relacionam ao estado.

A figura 3.19 apresenta parte do diagrama de estado para a classe Obra. É um diagrama bastante completo em comparação aos outros métodos. Ele apresenta, além dos estados, os seguintes itens: i) transições, eventos e condições, ii) as ações nas transições, os atributos de cada evento e as ações internas de entrada, de saída e os eventos que ocorrem dentro do estado sem mudança de estado, iii) as atividades também podem ser expressas dentro do estado.

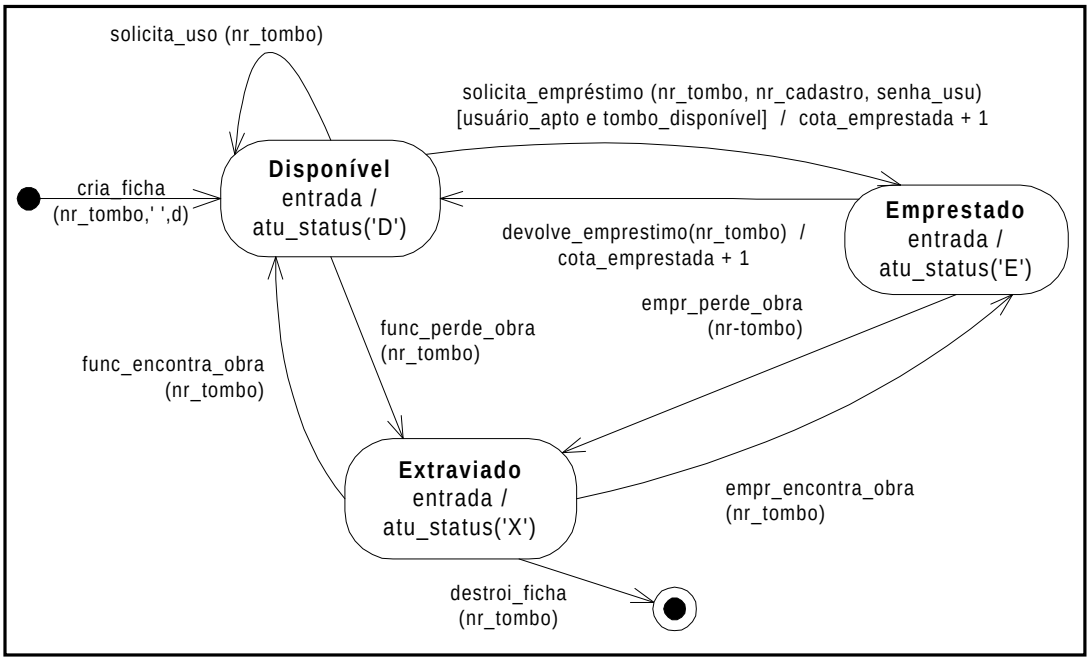



Figura 3.19 - Parte do diagrama de estado para a classe obra

Durante a análise, no que tange a modelagem funcional são utilizados um diagrama para mostrar os valores de entrada e de saída do sistema, um diagrama de fluxo de dados (DFD) e as descrições destas funções.

O primeiro DFD mostra os valores de entrada e de saída do sistema, os quais são parâmetros dos eventos existentes entre o sistema e o mundo externo. É um diagrama simples, que parece até ser redundante se for observado o diagrama de eventos, já que os eventos neles identificados, são seguidos por seus parâmetros. Posteriormente, são construídos os DFD's em níveis, sendo que o DFD de nível mais alto é o mais genérico. O objetivo é mostrar a origem dos valores de dados, sua transformação pelos processos e seus destinos (os objetos). Este diagrama não mostra decisão ou seqüência de operações.

Com os DFD's prontos, é necessária a descrição de cada uma das funções. (Rumbaugh,1994) sugere várias maneiras para descrever as funções: linguagem natural, equações matemáticas, pseudocódigo, tabela de decisões ou alguma outra forma.

Vale destacar que houve tentativas de modelagem de um diagrama de fluxo de dados (DFD) para parte do sistema de biblioteca, não havendo sucesso, pois o sistema possui pouca transformação de dados. Entretanto, através da figura 3.20 pode-se ter uma idéia de um DFD.

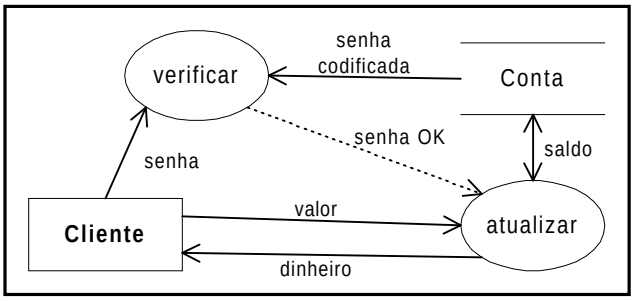



Figura 3.20 - DFD para uma retirada bancária (Rumbaugh, 1994)

A figura 3.20 descreve a retirada de dinheiro de um banco, mediante a verificação de senha, informada pelo cliente e a atualização do saldo decrementado de acordo com o valor solicitado pelo cliente. Este processo só será executado se a senha estiver correta.

Somente durante a análise há modelagem de interface, que possui duas partes: interface com o usuário e lógica da aplicação. A primeira parte modela as telas de interface com o usuário sem se preocupar com a lógica. O objetivo é verificar que nada importante ficou esquecido. A segunda parte modela a dinâmica, a seqüência de interação entre o usuário e as telas e a seqüência entre as telas.

Os três modelos são elaborados abrangendo três visões do sistema. Nem sempre haverá necessidade ou condições de se obter os três modelos, pois isso dependerá do sistema que se está modelando. No caso do exemplo da biblioteca, foi útil a criação dos modelos de objetos e dinâmico, já não sendo possível a criação do modelo funcional.

**c) Projeto**

Durante o projeto do sistema que possui oito etapas, a modelagem funcional está presente na primeira etapa do projeto. A modelagem dinâmica é feita durante a segunda, a terceira, a quinta, a sexta e a sétima etapas, e a modelagem estática é feita pela quarta e oitava etapas, devendo ser todas documentadas. Entretanto, o método não fornece diretrizes para se fazer essa documentação. O método também não oferece modelos gráficos nem textuais para se modelar o projeto do sistema.

O projeto do sistema inicia-se com a subdivisão do sistema em subsistemas sendo cada um desses composto por elementos que possuem as mesmas características: funcionalidade, localização física ou execução em mesmo *hardware*. O fluxo de informações existente entre os subsistemas são apresentados em um DFD. É importante notar que esse fluxo de informações deve ser baixo, pois se houver muita dependência entre os subsistemas, provavelmente eles formam um único subsistema.

Os subsistemas para o sistema de biblioteca poderiam ser: empréstimos, reservas, pesquisas, cadastros e emissão de relatórios.

A segunda etapa do projeto do sistema identifica as concorrências inerentes, que ocorre quando dois objetos recebem eventos ao mesmo tempo sem interagirem. (Rumbaugh, 1994) indica o modelo dinâmico - o diagrama de estados - como base para essa identificação.

A terceira etapa do projeto do sistema é a alocação dos subsistemas aos processadores, verificando quatro itens: i) decidir se serão necessários vários processadores, ii) decidir se a implementação será em *hardware* ou em *software*, iii) alocar os subsistemas aos processadores, observando a satisfação das necessidades de desempenho e a diminuição da comunicação interprocessadores e iv) organizar a conectividade física desses processadores.

A quarta etapa do projeto do sistema consiste em decidir sobre armazenamento dos dados, se será utilizado um SGBD (Sistema Gerenciador de Banco de Dados) ou arquivos simples. Os autores do método discutem as vantagens e desvantagens de se usar um ou outro.

A quinta etapa do projeto do sistema consiste em identificar recursos como: unidades físicas, espaço, nomes lógicos e acesso a dados compartilhados e determinar mecanismos para controlar o acesso a eles.

A sexta etapa do projeto do sistema trata da escolha de uma implementação de controle de *software*. Existem dois tipos de controle: externo e interno. O controle seqüencial baseado em procedimento, seqüencial baseado em eventos ou concorrente são tipos de controle externo. E os controles internos são: chamadas de procedimentos, chamadas intertarefas quase concorrentes e chamadas intertarefas concorrentes. Os autores advertem que os subsistemas podem ser implementados de maneiras diferentes, entretanto, o projetista, geralmente, adota uma única.

A sétima etapa do projeto do sistema consiste em tratar as condições extremas do sistema, tais como: inicialização (de constantes, parâmetros, variáveis globais, objetos, etc.), término (liberar recursos externos) e falhas (tratar qualquer término inesperado do processamento do sistema).

A oitava etapa do projeto do sistema consiste em estabelecer prioridades em relação a vários aspectos: memória, tempo, espaço, *hardware*, *software*, eficiência e manutenibilidade. O objetivo é fazer um balanceamento adequado entre esses aspectos a fim de obter uma arquitetura consistente, sem desperdícios. Às vezes, um aspecto pode ser sacrificado a fim de valorizar outro.

O projeto de objetos possui oito etapas, a modelagem estática está presente da primeira a terceira etapa e a modelagem dinâmica auxilia da quarta a última etapa. Estas etapas devem ser documentadas, contudo, o método não fornece diretrizes para se fazer essa documentação.

O projeto de objetos inicia-se com a identificação das operações das classes para atualização do diagrama de classes, que foi transportado da análise para o projeto de objetos. Esta identificação é feita com a ajuda das ações e atividades do modelo dinâmico - diagrama de estados - e dos processos do modelo funcional.

A figura 3.21 ilustra o diagrama de classes referente ao da análise representado pela figura 3.14, com o acréscimo das operações colocadas na parte inferior da classe, por exemplo: obter_qtdd_tombos ( ) da classe Ficha_tombo e obter_nr_tombo ( ) da classe Ficha_tombo.

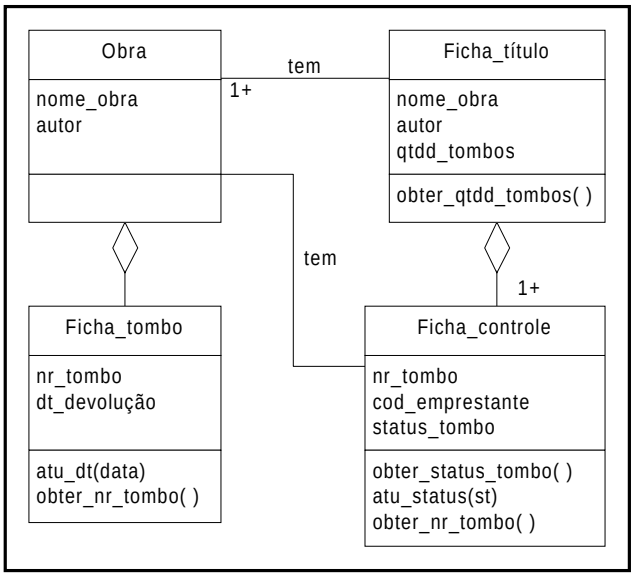



Figura 3.21 - Parte do diagrama de classe

O projeto de objetos prossegue com a definição de cada operação do modelo funcional na forma de um algoritmo. Para tanto, é levado em consideração: i) a escolha do algoritmo que minimize o custo da implementação das operações, ou seja, é verificado a necessidade de um algoritmo complexo ou simples e sua eficiência, ii) a escolha da estrutura de dados para a implementação do algoritmo, iii) definição de classes e operações internas, que acontece quando se faz a especificação das operações, pois novas classes podem ser necessárias para o armazenamento de resultados intermediários e novas operações podem surgir da decomposição das operações de alto nível e iv) atribuir as operações aos objetos alvos verdadeiros, pois muitas vezes, uma operação pode afetar mais de um objeto.

A terceira etapa do projeto de objetos consiste em otimizar o projeto, obtendo um equilíbrio entre a eficiência e a clareza. As diretrizes são: acrescentar associações redundantes com a criação de índices; reorganizar a ordem de execução de um algoritmo para melhorar a eficiência e armazenar atributos derivados do processamento de outros atributos para evitar o reprocessamento, que devem ser atualizados sempre que os atributos base forem alterados.

A quarta etapa do projeto de objetos trata da implementação do controle, ou seja, da implementação do modelo dinâmico definido na análise, definindo se será utilizado um sistema baseado em procedimento, um sistema baseado em eventos ou tarefas concorrentes. Nesta etapa acrescenta-se detalhes ao que foi definido no projeto do sistema.

A quinta etapa do projeto de objetos faz o ajustamento de herança, observando: i) as operações idênticas entre as classes e que podem ser colocadas em uma classe comum ii) os comportamentos comuns para colocá-los em uma superclasse, deixando as especialidades para a subclasse, com isso a superclasse poderá ser reaproveitada em outros projetos e iii) a utilização da delegação ao invés da herança, para evitar o uso incorreto desta quando uma subclasse não é uma forma da superclasse em todos os aspectos.

A sexta etapa do projeto de objetos consiste em formular uma estratégia para a implementação das associações, se elas serão bidirecionais ou com apenas uma direção, o que a torna mais simples. Contudo, é importante verificar se os requisitos não serão alterados, necessitando que a implementação seja bidirecional. Várias estratégias, que podem ser encontradas em (Rumbaugh, 1994).

A sétima etapa do projeto de objetos trata da representação de objetos, porque embora as classes tenham sido modeladas a nível de outras classes, tudo deve ser implementado em termos de tipos de dados primitivos: inteiros, *strings*, tipos enumerados.

A oitava etapa do projeto de objetos consiste em fazer o empacotamento físico (de programas), observando: i) ocultamento de informações (de atributos e alguns métodos) para determinar a visibilidade dos métodos; ii) coerência de entidades para verificar se as entidades não modelam vários propósitos ao mesmo tempo, fazendo separação entre a implementação (execução de algoritmos) e a política (tomada de decisões), o que aumenta o reuso; e iii) construção de módulos para facilitar o trabalho em equipes e para que futuras modificações afetem poucos módulos, deste modo, as classes de um módulo devem estar ligadas e terem o mesmo propósito.

### *3.4.4 Transição entre as fases*

O método inicia-se com os clientes e talvez os desenvolvedores gerando o enunciado do problema, que pode ser inconsistente, incompleto e informal. Por isso, a análise do sistema é realizada tentando tornar esse enunciado em verdadeiros requisitos.

Os autores do método sugerem que ao terminar a modelagem do sistema, nas três dimensões apresentadas - estática, dinâmica e funcional -, seja feita uma checagem entre os modelos para excluir erros e inconsistências refazendo, se necessário, alguns desses modelos para torná-los corretos. Quando estiverem corretos, os requisitos serão atualizados e/ou redefinidos e isto será discutido com o cliente para verificar se é o que se deseja.

### *3.4.5 Aplicabilidade*

O método OMT é uma boa opção para analistas experientes em modelagem com métodos estruturados de análise (sem muita experiência em programação), pois o seu enfoque é na abstração do domínio do problema e não na implementação. Isso facilita o trabalho dos analistas com várias opções de representação das associações entre objetos.

Esse método não é uma boa opção para programadores que estejam aprendendo a modelar, porque o seu enfoque não é direcionado às linguagens de implementação, mas sim às abstrações de alto nível. Os programadores tendem a querer mapear as representações de análise para algo no domínio de implementação, mas os modelos de análise não têm este objetivo, gerando alguma confusão nos primeiros projetos.

### *3.4.6 Pontos positivos*

O método OMT possui uma ótima explanação sobre os conceitos de orientação a objetos. Isso auxilia bastante a compreensão do método. O método também possui uma extensa notação para a análise com riqueza sintática e semântica, que facilita a modelagem de uma variedade de sistemas.

Três representações permitem que o modelo de objetos se torne mais claro quanto as associações entre as classes, são eles: atributos de ligação, associação ternária e qualificação (já explicados no capítulo 2).

Durante a fase de análise, o método é bastante prescritivo, o que facilita o desenvolvimento dos modelos e diagramas, que permitem uma boa abstração do que o sistema deve fazer, porque oferece uma seqüência de passos bem definidos.

### *3.4.7 Pontos negativos*

Deveria haver alguma modelagem para auxiliar os desenvolvedores na identificação dos requisitos do sistema juntamente com os usuários.

Durante a fase de projeto, o método OMT deixa muito a desejar, por não possuir a mesma riqueza sintática e semântica utilizada na análise. São fornecidas apenas sugestões, passos a serem seguidos, de como evoluir os modelos de análise para os de projeto, contudo, não há exemplificações do que está sendo sugerido, o que dificulta o estudo e aplicação do mesmo.

O diagrama de instância é bastante simples comparado ao que os outros métodos oferecem.

## *3.5 OOAD - Object Oriented Analysis and Design*

### *3.5.1 Descrição*

OOAD - *Object Oriented Analysis and Design* - Análise e Projeto Orientado a Objeto, é um método orientado a objeto desenvolvido por Grady Booch. Esse método foi inicialmente criado somente com o micro processo, recebendo o nome de OOD (*Object Oriented Design*), sendo apresentado em (Booch, 1991). Em (Booch, 1994), ele incorpora ao seu método o macro processo e o renomeia para OOAD. Assim, o método OOAD é composto pelo macro processo e pelo micro processo.

O micro processo é a descrição das atividades diárias de um único desenvolvedor ou pequeno grupo de desenvolvedores de *software*. A figura 3.22 ilustra o micro processo que abrange quatro atividades descritas abaixo.

a) identificação de classes e objetos para estabelecer os limites do problema, identificando as classes e objetos do domínio do problema, esta é uma fase de descobrimento.

b) identificação da semântica das classes e objetos para estabelecer os atributos e as operações da cada abstração identificada anteriormente, fazendo uma distribuição adequada.

c) identificação dos relacionamentos entre classes e objetos para definir os limites das abstrações e reconhecer os colaboradores de cada abstração identificada nas fases anteriores.

d) implementação de classes e objetos para representar cada abstração e mecanismos concretamente de maneira mais eficiente.

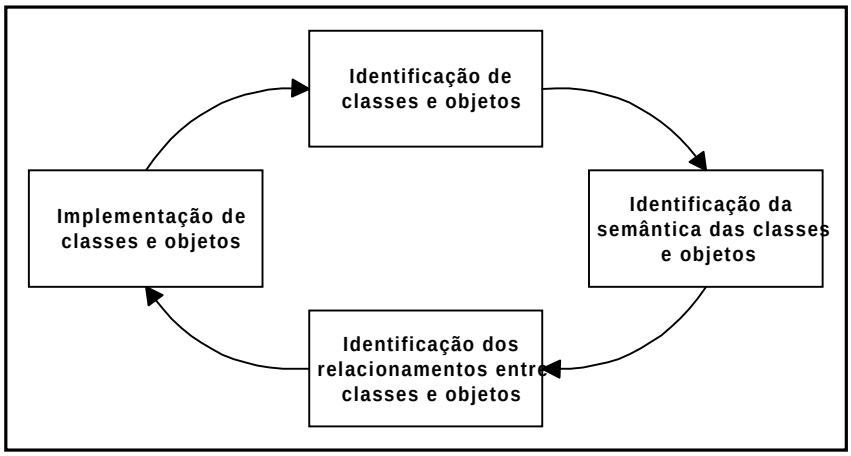



Figura 3.22 - Micro processo (Booch, 1994)

O macro processo serve como uma estrutura de controle para o micro processo. A principal preocupação do macro processo é com o gerenciamento técnico da equipe de desenvolvimento e com práticas básicas tais como: gerência de configuração, garantia de qualidade, ensaios de código e documentação. As fases do macro processo são: conceitualização, análise, projeto, evolução e manutenção.

A figura 3.23 ilustra a abordagem adotada pelo método, que consiste em aplicar as atividades do micro processo dentro de cada fase do macro processo, permitindo o refinamento dos modelos a cada fase. *A filosofia básica do macro processo é o desenvolvimento incremental* (Booch, 1994).

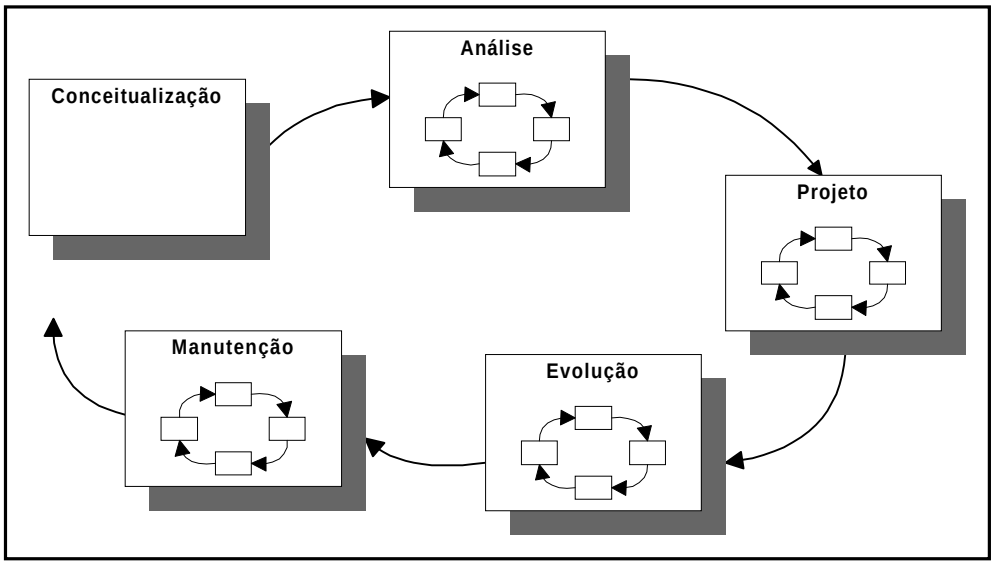



Figura 3.23 - Macro processo (Booch, 1994)

### *3.5.2 Características*

O método adota a abordagem orientada a comportamento, onde a preocupação básica está com as operações do sistema.

A primeira versão do método fazia menção somente a modelos de projeto. Na segunda versão houve amadurecimento dos modelos de projeto e inclusão dos modelos de análise. Assim, seu foco está na fase de projeto pela riqueza sintática e semântica.

Para todos os diagramas sugeridos pelo método - diagrama de classe, diagrama de objeto, diagrama de interação, diagrama de transição de estado, diagrama de módulo e diagrama de processo -, descritos na próxima sessão, deve ser feita uma especificação de forma textual podendo seguir o modelo que é sugerido pelo método em (Booch, 1994).

### *3.5.3 Fases*

**a) Análise de requisitos**

A análise de requisitos é denominada por (Booch, 1994) de conceitualização e tem o propósito de estabelecer o núcleo dos requisitos do sistema. Nem todos os projetos requerem esta fase.

Não é apresentada nenhuma modelagem para essa fase, todavia uma prototipação é indicada para fazer levantamento dos requisitos do sistema. Especialmente se o sistema for de grande escala, pois é melhor descobrir o quanto antes que as funcionalidades, performance, tamanho e complexidade estão erradas, do que em fases futuras quando o impacto técnico e financeiro é maior.

**b) Análise**

Na análise a identificação de classes e objetos é aplicada para descobrir as abstrações que formam o vocabulário do domínio do problema, para começar a restringi-lo, decidindo o que é e o que não é de interesse. A identificação da semântica das classes e objetos é aplicada para alocar as responsabilidades a diferentes procedimentos do sistema. A identificação dos relacionamentos entre classes e objetos é aplicada para especificar associações entre classes e objetos (incluindo certos relacionamentos importantes de herança e agregação), as quais especificam alguma dependência semântica entre duas abstrações e alguma habilidade de navegar de uma entidade para outra. A implementação de classes e objetos é aplicada para fornecer um refinamento de abstrações existentes o suficiente para revelar novas classes e objetos no próximo nível de abstração.

Nesta fase, desenvolvem-se os cenários que capturam as descrições das funções do sistema. São desenvolvidos os cenários básicos para ilustrar os procedimentos principais e cenários secundários para mostrar procedimentos sob condições especiais. A modelagem de um cenário foi incorporada ao método tendo como base as definições feitas por (Jacobson, 1992) no que diz respeito as descrições de um *use case*, portanto um exemplo pode ser encontrado na figura 3.3.

Como complementação aos cenários, são desenvolvidos os diagramas de objeto, que apresentam os objetos que colaboram para realizar um determinado cenário e a interação entre eles através da passagem de mensagens. Como este diagrama será acrescido de detalhes na fase de projeto, sua descrição mais detalhada e uma exemplificação encontram-se no item c) dessa seção.

Os cenários podem ser representados pelo diagramas de interação, assim como pelos diagramas de objetos. Todavia, estes são mais abrangentes do que aqueles. A vantagem de se construir o diagrama de interação, é que com ele é mais fácil de compreender a passagem e a seqüência de mensagens entre os objetos. Se este diagrama se tornar muito complexo, pode-se incluir *scripts* (descrições) do lado esquerdo do diagrama, descrevendo de forma textual a interação entre os objetos.

A figura 3.24 apresenta parte do diagrama de interação para o cenário emprestar obra. Os objetos estão na parte superior da figura, as descrições na parte esquerda e as mensagens estão entre os objetos.

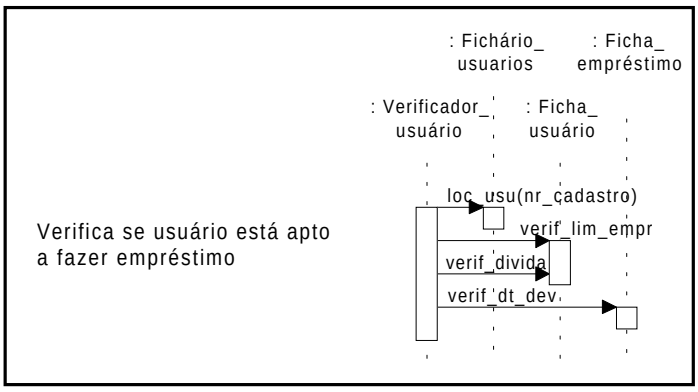



Figura 3.24 - Parte do diagrama de interação - cenário emprestar obra

Adicionalmente, são incluídos diagramas de classes apresentando as classes que formam o domínio do problema e seus relacionamentos - os dois componentes essenciais deste diagrama -. É um diagrama estático e bastante rico em notação, porque vários aspectos podem ser modelados, tais como: metaclasse, classe parametrizada, conteúdo físico, restrições, chaves, papéis, atributos de associação e notas, além dos essenciais citados anteriormente.

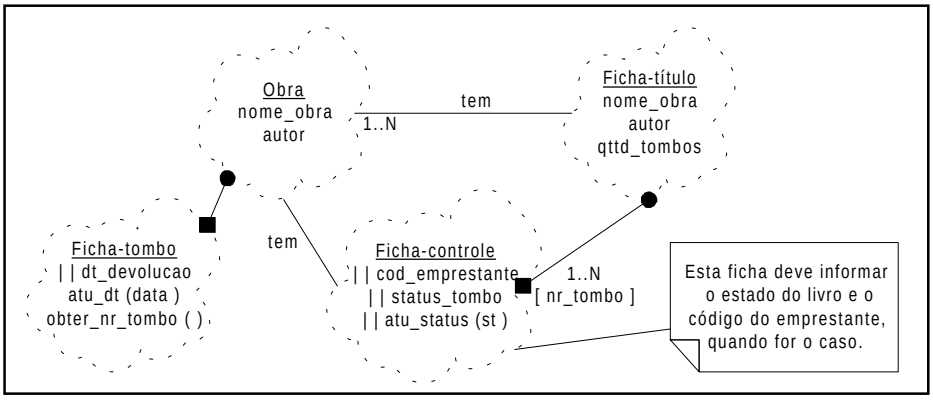



Figura 3.25 - Parte do diagrama de classe

A figura 3.25 apresenta parte do diagrama de classe para o sistema de biblioteca, que modela as classes, uma nota para descrever algo importante, seus relacionamentos simples e de agregação com conteúdo físico por valor. Isso significa, por exemplo, que se a Ficha_título for excluída todas as Ficha_controle referente àquela serão excluídas também.

**c) Projeto**

No projeto a identificação de classes e objetos serve para descobrir abstrações que são elementos da solução. A identificação da semântica das classes e objetos serve para conseguir uma separação clara de relações entre as partes da solução. A identificação dos relacionamentos entre classes e objetos serve para especificar colaborações que formam os mecanismos de nossa arquitetura e para agrupar as classes de mais alto nível em categorias e módulos dentro de subsistemas. A implementação de classes e objetos é aplicada para criar representações tangíveis das abstrações no suporte de sucessivos refinamentos de versões executáveis no macro processo.

O propósito do projeto é criar uma arquitetura, baseando-se no procedimento que o modelo de análise requer, para desenvolver a implementação e para estabelecer uma política tática comum que deve ser usada para disparar elementos comuns do sistema.

O projeto arquitetural só pode começar quando a equipe de desenvolvimento tem um entendimento razoável dos requisitos do sistema. O projeto focaliza a estrutura tanto estática como dinâmica. Análise mais aprofundada pode ocorrer durante a fase de projeto, principalmente para explorar áreas de incerteza a respeito do comportamento desejado do sistema. Contudo, o projeto serve, principalmente, para criar o esqueleto concreto sobre o qual todo o resto da implementação se projeta.

Durante o projeto é feita uma descrição da arquitetura, utilizando os diagramas de classes e os diagramas de objetos para mostrar as estruturas das classes e dos objetos, respectivamente, destacando o agrupamento das classes em categorias de classes (arquitetura lógica), e dos módulos aos subsistemas (arquitetura física).

O diagrama de objetos proposto por (Booch, 1994) é o mais completo do que os propostos pelos demais métodos aqui estudados (por ex.: OMT - diagrama de instâncias). Este diagrama, além de apresentar os objetos das classes e seus relacionamentos, modela a visibilidade entre os objetos e a sincronização das mensagens intercambiadas entre os objetos.

Os diagramas de objetos são constituídos dos objetos e seus relacionamentos, e são utilizados para complementar os cenários. Estes diagramas apresentam a situação do sistema em determinados momentos. A composição destes diagramas também pode incluir a representação de papéis, chaves, restrições, fluxo de dado, visibilidade entre os objetos, identificação de objetos ativos ou seqüenciais, sincronização e seqüência de mensagens em segundos, possibilitando visualizar maiores detalhes.

Dois objetos estão sincronizados quando um objeto (objeto ativo[11]) passa uma mensagem para outro objeto (objeto passivo[12]) através de um relacionamento (ligação).

A sincronização pode ser de três tipos: seqüencial, guardada e síncrona (Booch, 1994).

i) seqüencial - único objeto ativo;

ii) guardada - vários objetos ativos, onde eles se comunicam para garantir exclusão mútua;

iii) síncrona - vários objetos ativos, onde o objeto passivo garante a exclusão mútua.

Parte do diagrama de objetos para o sistema de biblioteca é ilustrado pela figura 2.15. A concorrência é mapeada de cinco maneiras: sem concorrência, síncrono, *balking*, *timeout* e assíncrono, descritos com mais detalhes na seção 2.2.5.

Os diagramas de módulos são compostos por módulos e suas dependências e mostram como classe e objetos são alocados para módulos no projeto físico do sistema. As dependências entre os módulos se referem a compilação dos programas, e é o único relacionamento existente entre os módulos. Estes módulos podem ser o programa principal, as especificações, programas secundários e subsistemas. Este diagrama pode acrescentar ainda tipos de módulos particulares da linguagem a ser utilizada, se necessário.

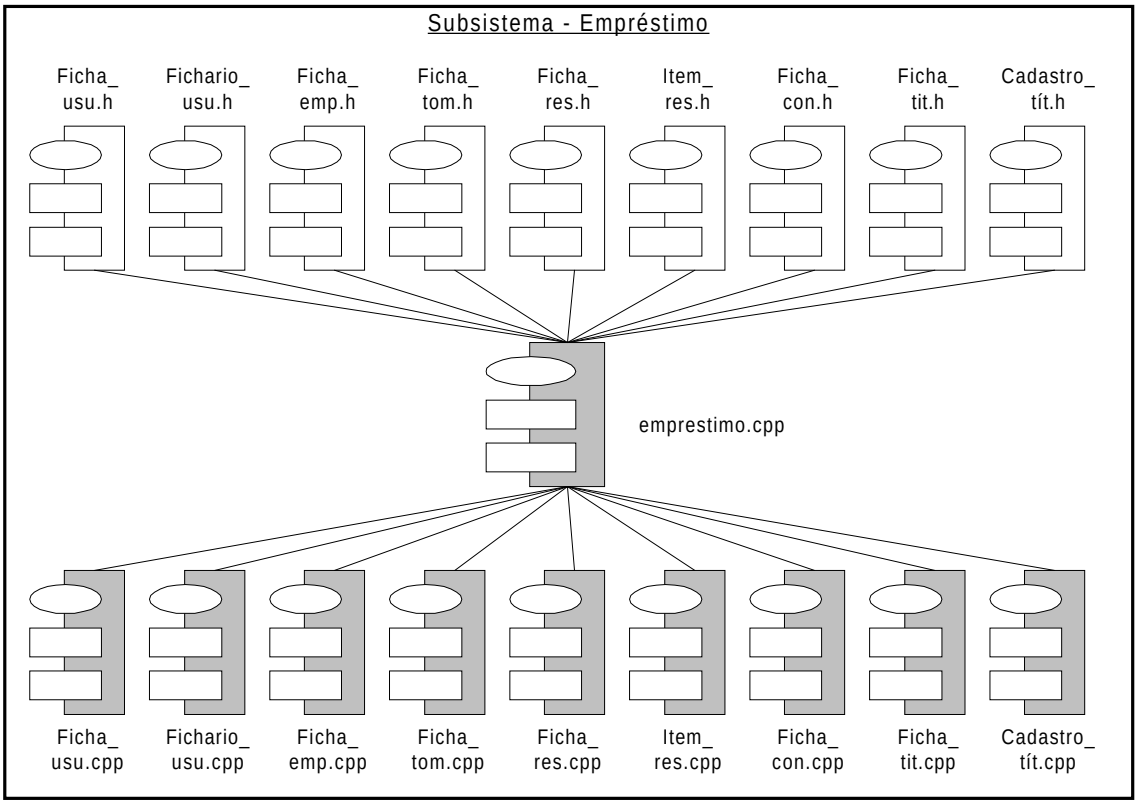



Figura 3.26 - Diagrama de módulos para o subsistema empréstimo

A figura 3.26 ilustra o diagrama de módulos para o subsistema Empréstimo, onde Emprestimo.cpp é um programa fonte que utiliza os arquivos de especificação (**.h**[13]), listados na

[11] Objeto ativo é o objeto que solicita uma chamada de uma operação de outro objeto (objeto passivo).
[12] Objeto passivo é o objeto que executará a operação que foi solicitada pelo objeto ativo.

parte superior da figura, e outros programas fontes (**.cpp)** que contém implementação das operações.

Os diagramas de processo que capturam os aspectos físicos são compostos pelos processadores (*hardware* capaz de executar programas), os dispositivos (*hardware* incapaz de executar programas, ex.: modem, impressora, terminal) e suas conexões, os quais formam a plataforma de execução do sistema.

Segundo (Booch, 1994), pode-se incorporar ao diagrama: i) uma representação alternativa dos processadores e dispositivos, facilitando que o usuário final do sistema compreenda melhor o aspecto físico, ii) uma representação para agrupar processadores, dispositivos e conexões, simplificando o diagrama e iii) determinar escalonamento de processos, especificando como: executivo ou manual e preemptivo, não preemptivo e cíclico. Maiores detalhes ver (Booch, 1994) e (Tanenbaum, 1995). Apesar das sugestões de (Booch, 1994), não é especificado pelo método uma modelagem ou notação para a captura dos itens ii) e iii).

A figura 3.27 ilustra parte do diagrama de processo para o sistema de biblioteca, com os relacionamentos entre os processadores, com seus processos listados abaixo de cada um e um dispositivo (a impressora) com representação alternativa.

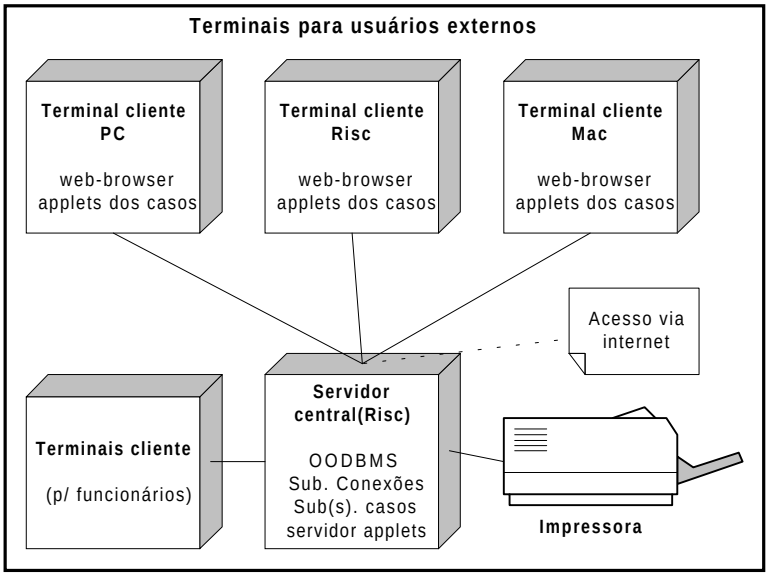



Figura 3.27 - Parte do diagrama de processo

Os diagramas de transição de estado são utilizados para mostrar os vários estados dos objetos de uma classe, os eventos que causam as transições de um estado para outro e as ações que resultam dessa mudança. Estes diagramas só são utilizados para modelar os objetos que possuem um comportamento dinâmico expressivo.

A figura 3.28 apresenta parte do diagrama de transição de estado para o objeto Ficha_controle. O estado inicial é o Disponível e o objeto é destruído depois de estar no estado Extraviado. A transição entre os estados é nomeada pelo evento seguido ou não de uma ação.

[13] As extensões **.h** e **.cpp** são designação para arquivos de especificação e de programa fonte, respectivamente, desenvolvidos em linguagem C++.

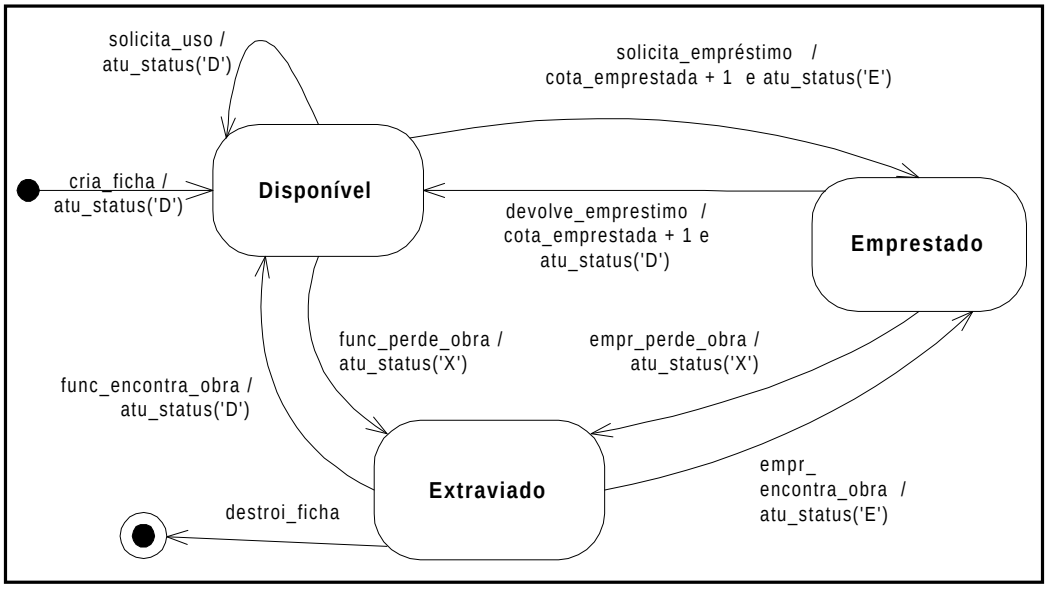



Figura 3.28 - Parte do diagrama de transição de estado

**d) Outras**

Vale observar que não faz parte do escopo do trabalho a análise das fases de evolução e manutenção, embora elas sejam aqui descritas para oferecer ao leitor uma visão completa do método.

O propósito da fase de evolução é produzir uma implementação através de sucessivos refinamentos da arquitetura do sistema, conduzindo a produção do sistema.

O propósito da fase manutenção é gerenciar o sistema gerado pela evolução, fornecendo uma continuação da fase anterior. De fato, mudanças mais localizadas são feitas ao sistema, como adicionar novos requisitos e eliminar erros protelados. O objetivo de continuar a evolução do sistema, seus produtos são idênticos aqueles da fase anterior, com uma adição: uma lista (*punch list*) de novas tarefas, que serve como veículo para coletar erros e aumentar requisitos, tanto que eles podem ser priorizados para futuras versões.

### *3.5.4 Transição entre as fases*

(Booch, 1994) destaca: *Faça a conceitualização somente quando os riscos do projeto forem relativamente altos ou quando for necessário inventar um vínculo inicial entre o cliente e a organização de desenvolvimento; senão, passe diretamente para a análise.*

*Não é possível nem desejado realizar a análise completa antes de permitir que o projeto comece* (Booch, 1994). Neste sentido, o projeto se inicia tão logo se tenha um modelo de análise razoavelmente completo.

A proposta desse método é a aplicação das fases repetidas vezes, onde a transição de uma fase para a próxima não exige que a anterior tenha sido totalmente concluída. Com esse enfoque, os modelos vão sendo aperfeiçoados e detalhados a cada fase e a cada interação.

### *3.5.5 Aplicabilidade*

Não é um método prescritivo, como os métodos OOA-OOD (Coad/Yourdon) e OMT (Rumbaugh), sendo portanto difícil a sua aplicação por equipes inexperientes. É um método bom para equipes de programadores experientes (em linguagens O.O.), que julguem não ser necessária a elaboração de modelos de requisitos e análise e visualizam o projeto de uma forma interativa com a implementação. Analistas podem ter certa dificuldade no aprendizado, por se tratar de um método direcionado para a implementação, apesar de apresentar outras fases.

Aplicável para organizações que pretendem construir uma família de programas e àquelas que tem um investimento significante de capital, devido a repetição do macro processo depois que o produto é liberado (Booch, 1994).

### *3.5.6 Pontos positivos*

O método possui uma boa descrição e exemplificação dos conceitos de orientação a objetos realizados de maneira bastante didática. Apresenta também conceitos avançados e seus respectivos elementos notacionais não encontrados em outros métodos.

O modelo de objetos do OOAD, além de modelar os objetos e seus relacionamentos, como apresentado pelos outros métodos, é o único que aborda a sincronização e a seqüência de troca de mensagem entre os objetos.

O método apresenta dois diagramas não descritos pelos outros métodos: o diagrama de processo e o diagrama de módulo, os quais adicionam detalhes de projeto. Assim, é um método que oferece bastante modelagem e descrição sobre o projeto.

### *3.5.7 Pontos negativos*

O método OOAD poderia ser mais prescritivo para facilitar sua compreensão e seu uso, tendo em vista que possui conceitos únicos, quer dizer, não encontrados nos outros métodos aqui estudados. Além disso, uma modelagem para a análise de requisitos seria útil para que se pudesse utilizá-lo como único método em todo desenvolvimento.

Os autores fazem referências a conceitos e diagramas utilizados e/ou definidos por outros autores, que descrevem outros métodos, tais como OOSE e OMT. Estes diagramas e conceitos, entretanto, não são explicados nem descritos no livro de Booch (Booch, 1994), o que dificulta a utilização do método por falta de explicações e pela necessidade de buscar outras fontes para que se tenha um esclarecimento.

## *3.6 OOA-OOD - Object Oriented Analysis / Object Oriented Design*

### *3.6.1 Descrição*

OOA-OOD (*Object Oriented Analysis and Object Oriented Design*) - Análise e Projeto Orientado a Objetos é um método orientado a objetos desenvolvido por Peter Coad e Edward

Yourdon em 1991 e 1992, tratando das fases de análise e projeto. O método é apresentado em (Coad, 1992) cobrindo a fase de análise e em (Coad, 1993) cobrindo a fase de projeto.

O método OOA-OOD tem seu desenvolvimento baseado em um modelo multicamadas na análise, que se expande para o projeto como o modelo multicamadas e multicomponentes.

O modelo multicamadas apresenta as camadas de classe e objetos, assunto, estrutura, atributo e serviço. O modelo multicomponentes retrata os componentes: domínio do problema, interação humana, gerenciamento de dados e gerenciamento de tarefas. Uma melhor descrição se encontra nas seções a seguir.

### *3.6.2 Características*

O método adota a abordagem orientada a dados, *a modelagem de uma classe implica no conhecimento prévio de seus atributos e serviços* (Silveira, 1995). A preocupação básica está com os dados e suas estruturas armazenados pelo sistema.

O foco do método é tanto na análise quanto no projeto, entretanto esse método não apresenta modelos ricos em notação. Caracteriza-se por ser o método mais prescritivo em relação aos métodos aqui apresentados, indicando com detalhes todas as atividades que devem ser executadas e os seus subprodutos.

Foi um dos primeiros métodos orientados a objetos a serem desenvolvidos (Fichman, 1992). Desta forma, tornou-se uma referência para a criação dos outros métodos, os quais puderam melhorar os pontos deficientes do método base, assim como aperfeiçoar o método como um todo.

Durante a explanação do método em (Coad, 1992) e (Coad, 1993), os autores sugerem o reuso, insistindo que o desenvolvedor faça uma busca em projetos anteriormente desenvolvidos. Eles indicam que o método tanto pode ser utilizado com o modelo de cascata (*waterfall*), como com o espiral ou com o incremental, que dependerá do enfoque a ser dado.

### *3.6.3 Fases*

**a) Análise de requisitos**

Os autores não oferecem modelagem (modelo ou diagrama) para captura e/ou identificação dos requisitos, porém, deixam claro a necessidade da criação de um documento que relata os requisitos, e que de preferência, se compreenda o domínio do problema antes de iniciar o desenvolvimento.

**b) Análise**

Durante a fase de análise é criado o modelo de análise, também chamado de modelo multiníveis ou multicamadas, porque é acrescido de detalhes a cada nível. Os níveis são: assunto, classe&objeto, estrutura, atributo e serviço, ilustrada pela figura 3.29. Eles não possuem seqüência definida, ficando a critério de cada analista a seqüência para construção do modelo.

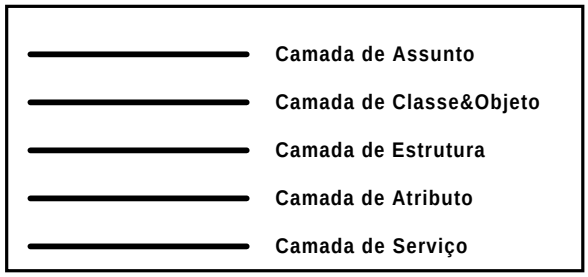



Figura 3.29 - Modelo multiníveis - análise (Coad, 1992)

No nível de assunto procura-se agrupar as classes&objetos formando assuntos (subsistemas), que facilitam a compreensão do sistema como um todo. No nível de classe&objeto[14] identifica-se as classes e os objetos do domínio do problema. No nível de estrutura faz-se o relacionamento entre as classes com a estrutura de generalização-especialização (herança - simples ou múltipla) ou com a estrutura todo-parte (agregação). No nível de atributo identifica-se primeiramente, os dados (informações de estado) de cada objeto que são pertinentes ao domínio do problema e depois as conexões de ocorrência existentes entre os objetos, para que cada objeto possa *cumprir suas responsabilidades* (Coad, 1991), ou seja, possa realizar todas as suas tarefas. Se necessário, pode ser feita a descrição de cada atributo em uma ou duas linhas sendo que restrições podem ser especificadas. No nível de serviço define-se, primeiramente, as operações (os serviços) de cada classe e depois as conexões de mensagem que modelam a dependência de processamento de um objeto, indicando quais serviços ele necessita.

A figura 3.30 apresenta uma parte do modelo multicamadas para o sistema de biblioteca, para o nível de classe&objeto. Neste nível só há a identificação das classes, que na figura são seis: Ficha_tombo, Ficha_título, Obra, Ficha_controle, Ficha_reservas e Item_reserva.

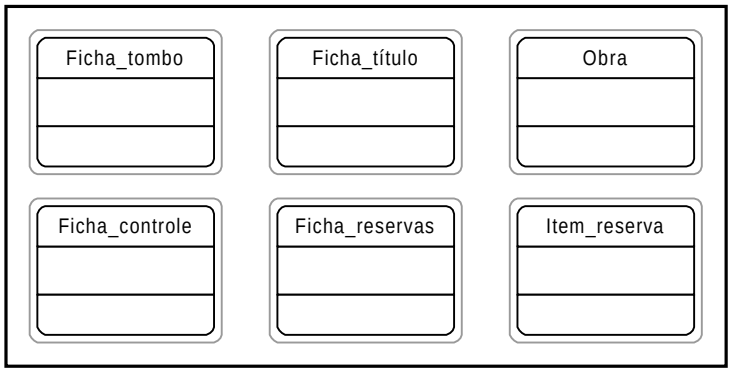



Figura 3.30 - Parte do modelo de análise - nível classe&objeto

A figura 3.31 apresenta a mesma parte do modelo de análise, ilustrado pela figura 3.30, com a adição dos níveis de atributo e assunto. Nestes níveis, acrescenta-se a identificação dos

[14] Classe&objeto é representado graficamente (como mostra a figura A.20 do apêndice A), modelando a classe e os objetos dessa classe. Assim, pode-se mapear explicitamente um objeto para outro, uma classe para outra, ou um objeto para uma classe.

atributos de cada classe, expressos na parte central de cada classe, o assunto ao qual as classes pertencem e as conexões de ocorrência com as devidas cardinalidades.

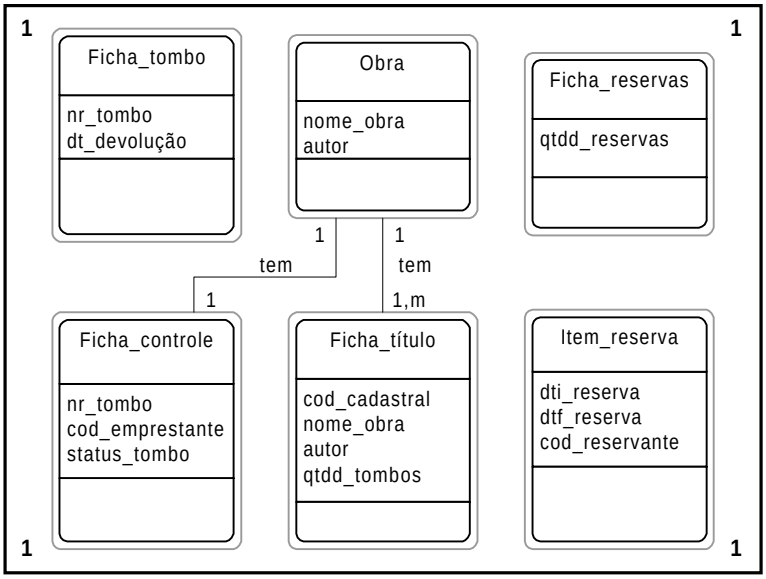



Figura 3.31 - Parte do modelo de análise - nível classe&objeto, assunto e atributo

A figura 3.32 apresenta parte do modelo de análise abrangendo todos os níveis. Inclui-se a identificação dos serviços (operações) na parte inferior das classes, as conexões de mensagem entre as classes&objetos e os relacionamentos de generalização-especialização e todo-parte.

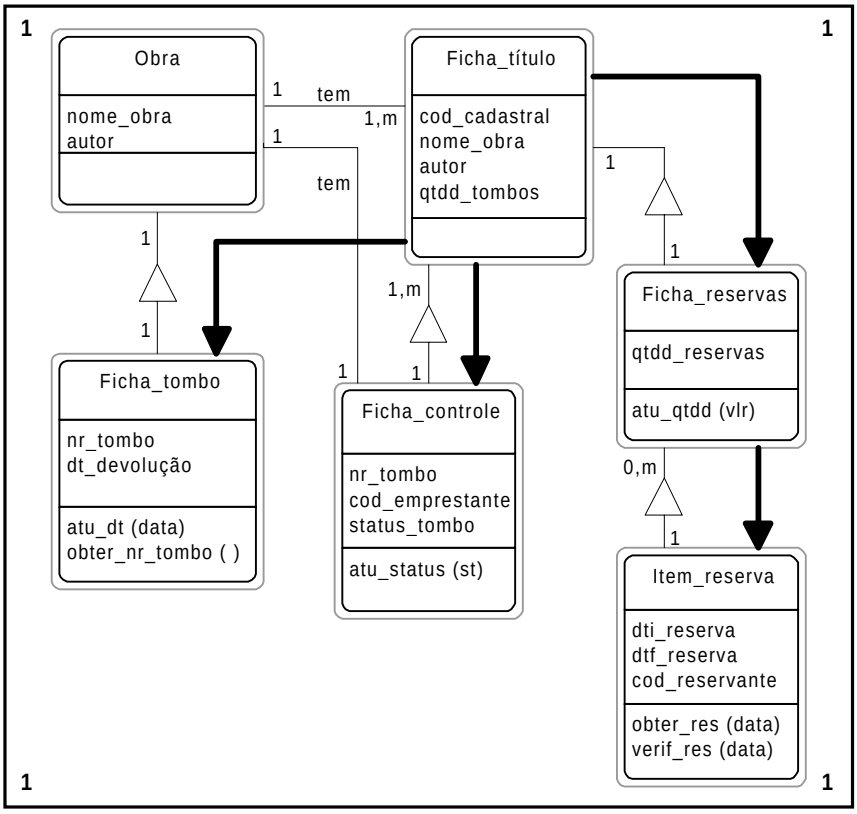



Figura 3.32 - Parte do modelo de análise - completo

Durante a análise é criado o diagrama de estado do objeto que modela os estados de um objeto, apresentando apenas notação para o estado (retângulo) e para as transições (setas), sendo mais simples que os apresentados pelos outros métodos, como mostra a figura 3.33.

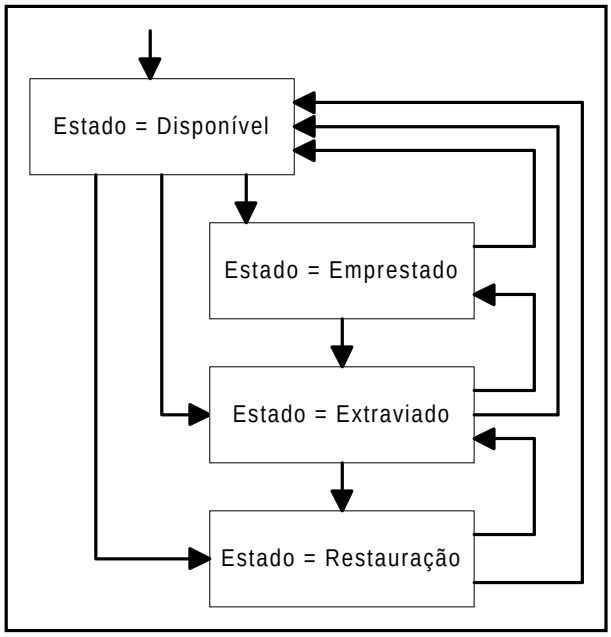



Figura 3.33 - Diagrama de estado do objeto obra

Através da figura 3.33 nota-se que o objeto Obra possui quatro estados e as possibilidades de transição entre estes estados. O estado Disponível é o estado inicial pois há seta direcionada a ele que não parte de outro estado.

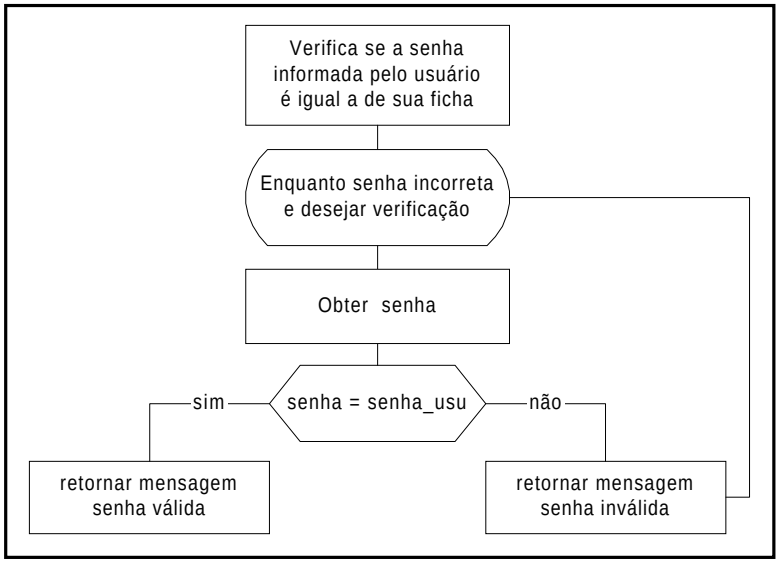



Figura 3.34 - Diagrama de serviço - verif_senha_usu (senha)

Outro diagrama desenvolvido na análise é o diagrama de serviço, utilizado para especificar todos os serviços do modelo a nível de algoritmo, possuindo bloco de texto, condição, *loop* e conector. A figura 3.34 ilustra o diagrama do serviço verif_senha_usu (senha) do objeto

Ficha_usuário, possuindo quatro blocos de texto (retângulos), uma condição - senha = senha_usu -, seis conectores e um loop que realizará enquanto a senha for incorreta e desejar a verificação. A saída do serviço é uma mensagem, ou indicando que a senha não confere ou que a senha está correta.

Ao final da criação do modelo de análise e dos outros dois diagramas apresentados (diagrama de estado do objeto e diagrama de serviço), deve-se fazer uma especificação de cada classe, podendo seguir o modelo sugerido pelos autores.

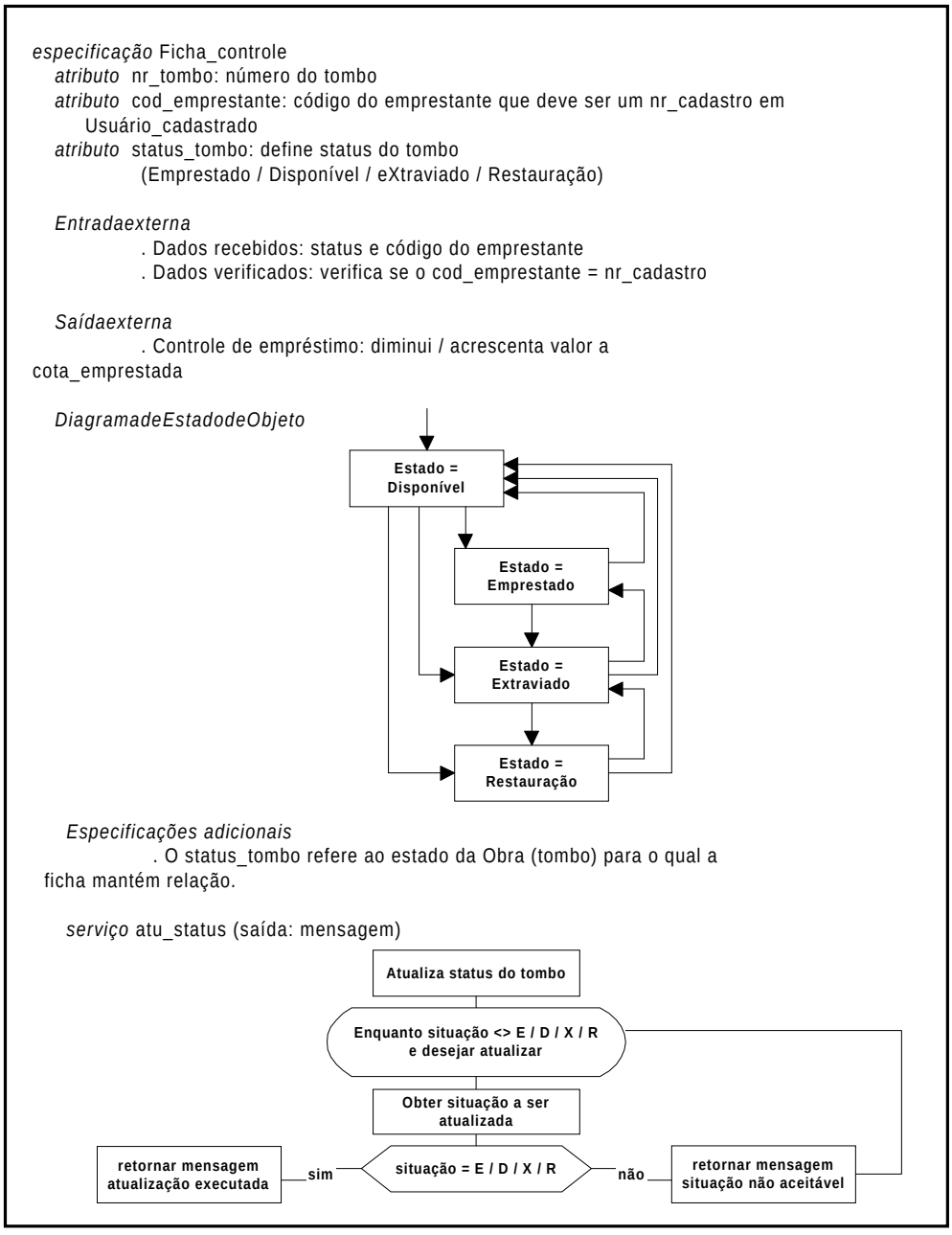



Figura 3.35 - Especificação da classe ficha_controle

A figura 3.35 apresenta a especificação da classe Ficha_controle, que apresenta três atributos, entrada e saída externa, diagrama de estado do objeto, especificações adicionais e um serviço com seu diagrama de serviço.

**c) Projeto**

Durante a fase de projeto é criado o modelo de projeto também chamado de modelo multicamadas e multicomponentes. A figura 3.36 ilustra o modelo de projeto que possui quatro componentes: domínio do problema, interação humana, gerenciamento de tarefas e gerenciamento de dados.

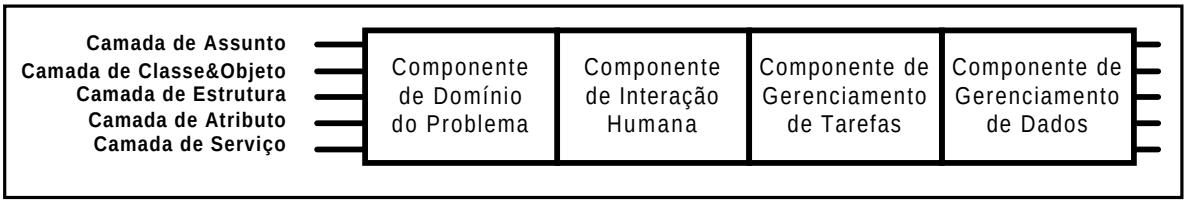



Figura 3.36 - Modelo multicamadas, multicomponentes - projeto (Coad, 1993)

O modelo de análise apresenta cinco camadas/níveis e se encaixa no modelo de projeto no componente domínio do problema, que será acrescido de detalhes de projeto como por exemplo: acomodar níveis de herança, melhorar performance, reutilizar classes de projeto e programação, etc., sendo chamado, a partir disso, de modelo de projeto, utilizando a mesma notação da análise.

No projeto, a modelagem de interface é feita com a especificação detalhada e o desenho de telas e relatórios no componente interação humana. Deste modo, será possível visualizar como uma pessoa comandará o sistema e como o sistema apresentará as informações.

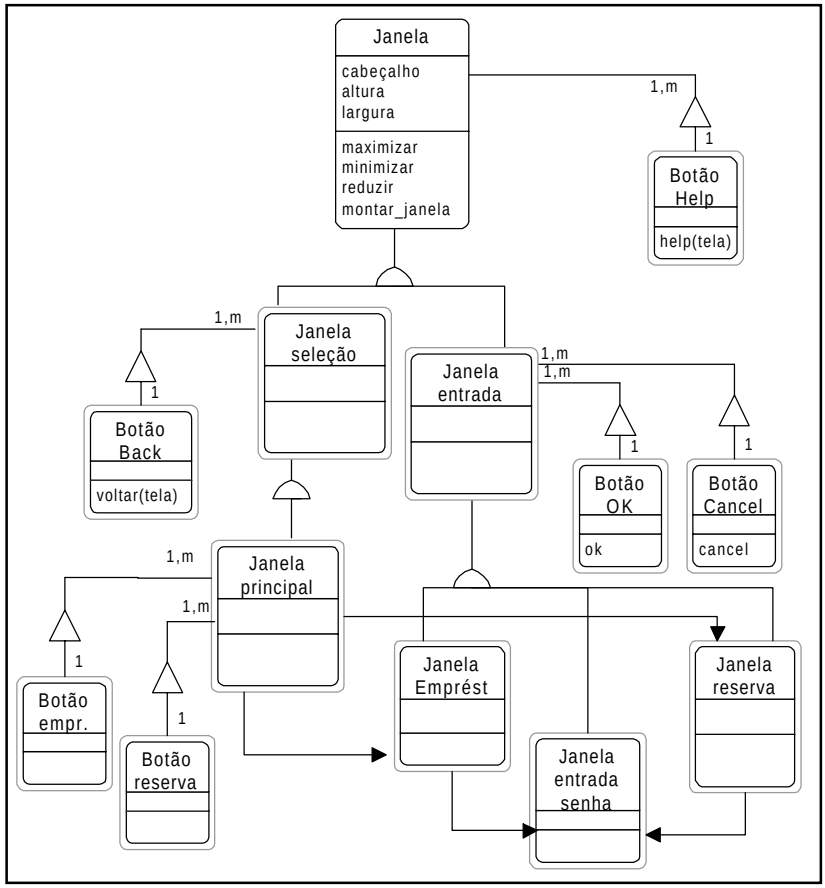



Figura 3.37 - Modelo de projeto - componente interação humana

A figura 3.37 mostra parte do modelo de projeto para o componente interação humana, que especifica a composição e apresentação das telas do sistema. Uma ilustração semelhante a tela original pode ser desenhada, com o objetivo de certificar sua validade junto ao usuário.

Durante a criação do componente gerenciamento de tarefa é necessário identificar as tarefas dirigidas por tempo e por evento, a prioridade de cada tarefa, as tarefas críticas, a tarefa que coordenará as outras e descrever cada uma delas.

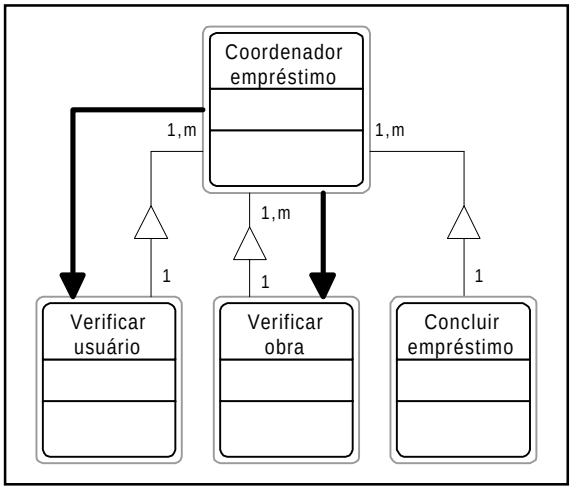



Figura 3.38 - Modelo de projeto - componente gerenciamento de tarefas

A figura 3.38 ilustra parte do modelo de projeto para o componente gerenciamento de tarefas para a tarefa realizar empréstimo. Uma tarefa coordenadora foi identificada, Coordenador de empréstimo, para coordenar três tarefas necessárias para se efetuar um empréstimo, são elas: verificar usuário, verificar obra e concluir empréstimo.

```
Nome: Verificar obra
Descrição: Esta tarefa é responsável por fazer
   verificações relativas a obra
Prioridade: Média
Serviços incluídos:
 . Cadastro_títulos.loc_ficha (cod_cadastral)
 . Ficha_título.obter_qtdd_tombos
 . Ficha_controle.obter_status_tombo
 . Ficha_reservas.loc_res (cod_cadastral)
 . Item_reserva.verif_tombo_res (nr_tombo)
 . Item_reserva.verif_cod_reserv (nr_cadastro)
Coordenada por:
 . controlada por tempo
Comunica via:
 . envia valores (nr_tombo) a outra tarefa
   (Concluir empréstimo)
```

Figura 3.39 - Modelo de projeto - especificação de tarefas

Para todas as tarefas identificadas é necessário fazer uma descrição sobre cada uma. A figura 3.39 apresenta a descrição para a tarefa Verificar obra, seguindo modelo sugerido pelos

autores do método, que relata nome, descrição, prioridade, serviços incluídos, coordenada por evento / tempo e sua comunicação (de quem recebe e para quem envia os valores).

Durante o desenvolvimento do componente denominado gerenciamento de dados, é necessário definir o *layout* (arquivo simples, relacional ou baseado em objeto) dos dados e também projetar os serviços correspondentes a fim de que os objetos possam ser armazenados.

### *3.6.4 Transição entre as fases*

A passagem da análise para o projeto significa uma expansão progressiva do modelo. Durante a análise, o objetivo é modelar o domínio do problema e as responsabilidades do sistema. Durante o projeto, o objetivo é expandir o modelo de análise com aspectos de implementação, adicionando os componentes: interação humana, gerenciamento de tarefas e gerenciamento de dados.

Desta forma, toda semântica e sintática utilizada na análise é utilizada no projeto também, permitindo uma melhor compreensão do desenvolvimento e evitando uma transformação de notação entre as fases.

### *3.6.5 Aplicabilidade*

É uma boa opção para equipes iniciantes em tecnologia de objetos, pois sua notação é simples e suficiente para boa parte dos projetos que tenham enfoque nos dados armazenados pelo sistema e sem muita dinâmica.

O método é aplicável a grandes equipes de desenvolvimento, utilizando a análise e depois o projeto. Sendo útil, também, às pequenas equipes e onde se utilize a prototipação, fazendo uma intercalação entre a análise e o projeto, um pouco de cada durante todo o desenvolvimento (Coad, 1992).

### *3.6.6 Pontos positivos*

O método possui a representação gráfica para classe&objeto que permite além do mapeamento entre classes e entre objetos, o mapeamento explícito de um objeto para uma classe, o que não é encontrado nos outros métodos, possibilitando a diferenciação entre a classe abstrata e a concreta.

O mesmo esquema notacional é aplicado à análise e ao projeto, acrescentando cada vez mais detalhes ao modelo, o que facilita o entendimento do sistema durante seu desenvolvimento, pois não possui notações diferentes para o mesmo aspecto somente por estar em fases distintas.

A notação diferenciada para mensagens permite uma rápida visualização da troca de mensagens entre os objetos. É importante para não misturar com os outros relacionamentos.

### *3.6.7 Pontos negativos*

A cardinalidade apresentada pelo método é diferente da utilizada pelos outros métodos, dificultando o entendimento do modelo, pois ela se apresenta invertida. A figura 3.40 apresenta um

exemplo, onde o Acervo possui uma ou mais (1,N) Obras; a figura da esquerda está modelada pelo método OMT (utilizada também pelos demais métodos) e a da direita pelo método OOA-OOD.

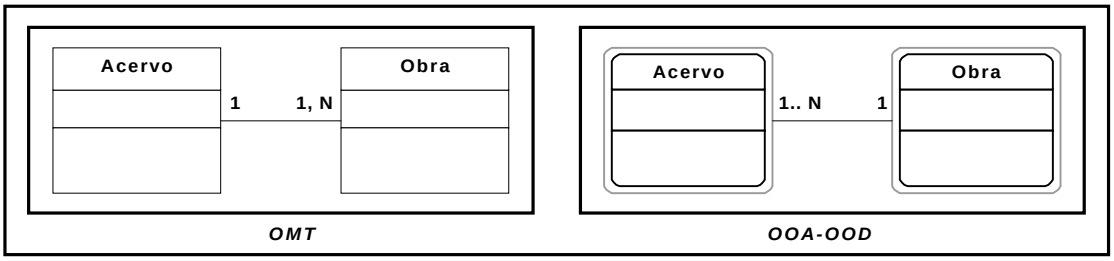



Figura 3.40 - Exemplificação da cardinalidade

O método OOA-OOD oferece pouca modelagem para o desenvolvimento de um sistema. A parte estática apresenta poucos modelos e a notação utilizada neste modelos é muito restrita. A parte dinâmica é representada juntamente com a estática, não apresentando notação suficiente, falta modelos específicos para a modelagem, tais como: cenários ou diagramas de eventos.

## *3.7 Fusion*

### *3.7.1 Descrição*

Fusion é um método orientado a objeto, desenvolvido por Derek Coleman, Patrick Arnold, Stephanie Bodoff, Chris Dollin, Helena Gilchrist, Fiona Hayes e Paul Jeremaes em 1994. O método está descrito no livro *Object-Oriented Development: the fusion method* e sua tradução para o português em (Coleman, 1996).

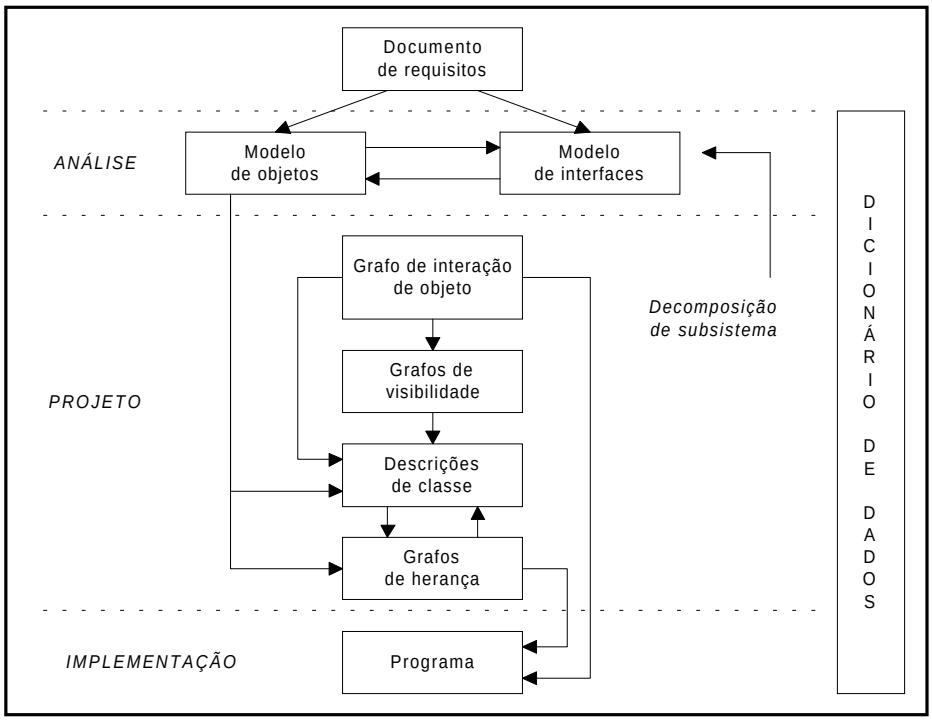



Figura 3.41 - Método Fusion (Coleman, 1996)

O Fusion aborda as fases de análise, projeto e implementação, deixando claro que *a captura de requisitos será executada por um usuário, que deve fornecer o documento inicial de requisitos* (Coleman, 1996), portanto o Fusion não faz modelagem da análise de requisitos. A figura 3.41[15] ilustra o processo desse método.

### *3.7.2 Características*

O método possui abordagem orientada a dados, onde a preocupação principal é a identificação dos dados armazenados pelo sistema. Isso é caracterizado pela descrição dos primeiros modelos a serem criados no início do desenvolvimento, onde não há menção às operações do sistema.

O foco está tanto na fase de análise, quanto na fase de projeto, pois o *Fusion* está baseado em métodos com foco nestas fases, método OMT e método OOAD, respectivamente.

(Coleman, 1996) classifica seu método como sendo de segunda geração e diz: *ele integra e amplia técnicas existentes*. Como o próprio nome sugere Fusão, do inglês Fusion, o método integra aspectos positivos de três métodos: OMT, Booch e métodos formais e a técnica CRC - Classe Responsabilidade Colaborador[16]. Esta integração é ilustrada pela figura 3.42.

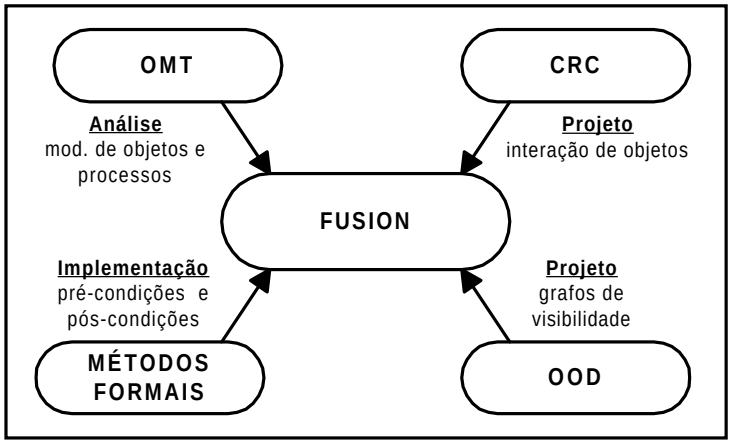



Figura 3.42 - Composição do método Fusion (Coleman, 1996)

Para a composição do *Fusion* adotou-se: i) o modelo de objetos baseado no mesmo modelo aplicado ao método OMT, ii) modelo de operações com influência dos DFD's do método OMT, iii) a interação de objetos baseado na técnica CRC, iv) as pré-condições e pós-condições dos métodos formais e v) os grafos de visibilidade do método OOD (primeira versão do método OOAD).

### *3.7.3 Fases*

**a) Análise de requisitos**

Não há modelagem para a análise de requisitos, a fase de análise terá como base um documento de requisitos definido pelo usuário.

---

[15] A figura original (Coleman, 1996) - versão traduzida - pág.: 311, lê-se 'Documento de requisitos', onde na figura 3.41 lê-se 'Grafo de interação de objeto'. Deve haver um erro gráfico na figura citada.

[16] CRC é uma técnica exploratória e não um método completo, que foi projetada originalmente para ensinar os conceitos básicos do projeto orientado a objeto (Coleman, 1996).

**b) Análise**

Durante a modelagem estática são construídos os seguintes modelos: modelo de objetos, o modelo de objetos do sistema e, em paralelo, um dicionário de dados.

A figura 3.43 ilustra parte do modelo de objetos construído para o sistema de biblioteca, baseando no documento de requisitos, onde serão modeladas as classes (ex.: Acervo_obras, Cadastro_títulos, Item_reserva, Funcionário, etc.), seus relacionamentos (ex.: tem, atualização e consulta), agregação (modelada como uma classe dentro da outra, ex.: o relacionamento entre Acervo_obras e Obra), atributos (ex.: qtdd_obras, autor, nr_tombo) e cardinalidade (ex.: 1, +, *).

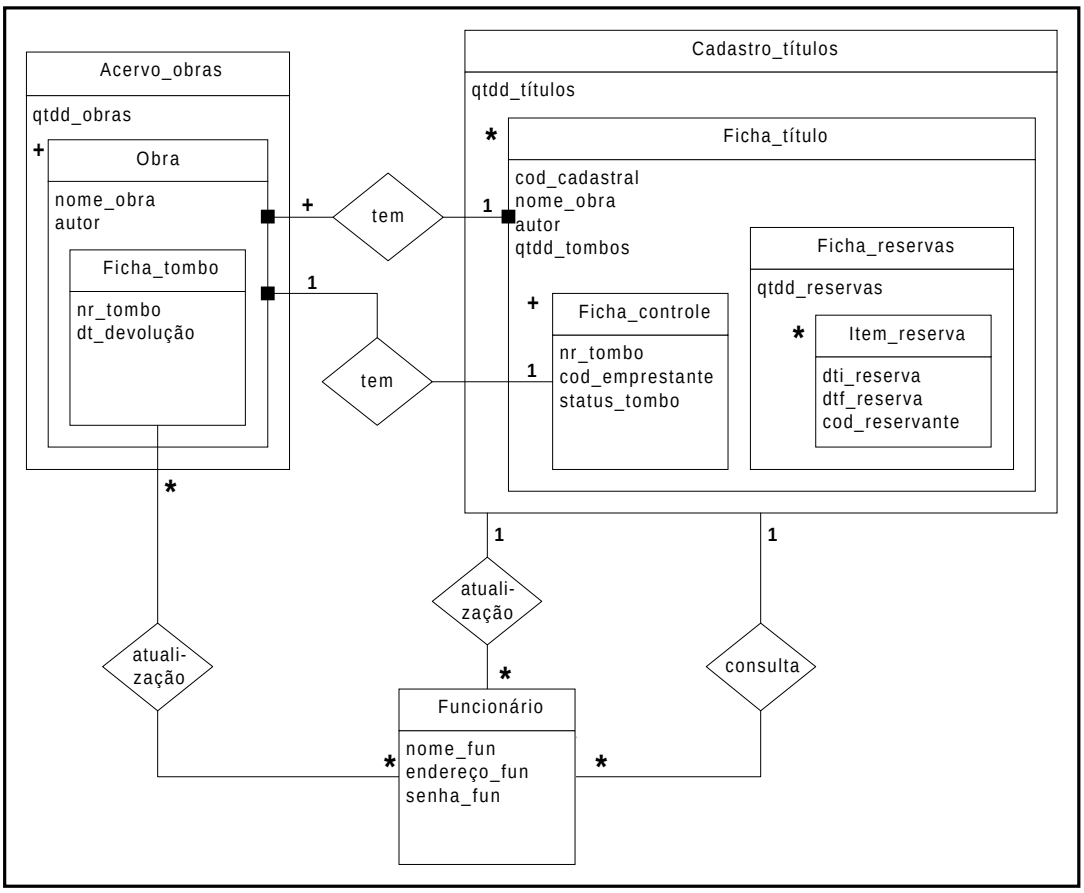



Figura 3.43 - Parte do modelo de objetos

Com base no modelo de objetos desenvolve-se o dicionário de dados, que será atualizado durante todo o desenvolvimento. Esse dicionário deve conter todas as definições necessárias para o entendimento do sistema que está sendo modelado devendo servir para fazer a verificação de consistência entre os modelos.

Para delimitar a fronteira do sistema e identificar as classes e relacionamentos que serão implementados, é criado o modelo de objeto do sistema, onde se faz uma separação entre as classes e relacionamentos que pertencem ao sistema e ao ambiente. A figura 3.44 ilustra parte do modelo de objeto do sistema para o sistema de biblioteca, onde as classes que pertencem ao sistema estão modeladas dentro do retângulo tracejado e do lado de fora o ambiente é modelado.

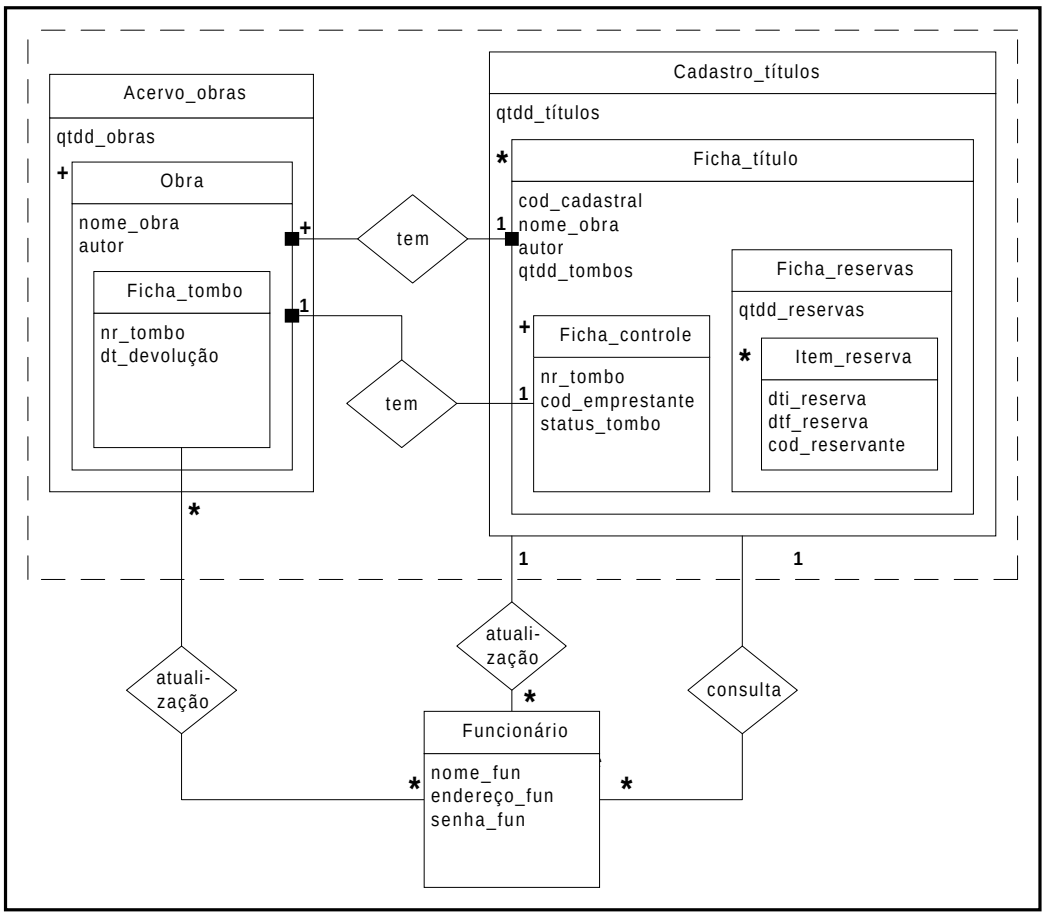



Figura 3.44 - Parte do modelo de objeto do sistema

A modelagem dinâmica do sistema é feita desenvolvendo-se o modelo de interface e seus componentes. O modelo de interface[17] modela o comportamento do sistema, identificando eventos e estados. O modelo de interface utiliza o modelo de ciclo-de-vida e o modelo de operações para capturar aspectos diferentes deste comportamento. Não existe uma ordem para o desenvolvimento, entretanto é sugerido que primeiro se faça o modelo ciclo-de-vida, porque ele pode auxiliar na construção do outro modelo.

O modelo ciclo-de-vida descreve o comportamento do sistema de maneira mais ampla, mostrando como o sistema se comunica com os agentes desde sua criação até seu término, ou seja, define as seqüências permissíveis de interação entre eles. Este modelo é excelente para capturar, além da ordenação, repetição e alternação dependentes das operações do sistema. O modelo ciclo-de-vida é composto por várias expressões ciclo-de-vida que generalizam os cenários, ou seja, definem um conjunto de cenários.

A figura 3.45 mostra o exemplo de um modelo de ciclo-de-vida incompleto, que se tornaria completo com a definição de todas as expressões ciclo-de-vida de todos os cenários do sistema.

[17] É interessante observar, que a interface do sistema, sugerida pelo método, é composta pelo conjunto de operações que o sistema pode responder (operação do sistema, ou eventos de entrada), e pelos eventos que este sistema pode gerar (eventos ou eventos de saída).

```
lifecicle  Sistema_biblioteca: Iniciação. (Empréstimo || Reserva || Pagamento ||
                    Devolução || Pesquisa)*

Iniciação = solicita_cadastro. #tipo_cadastro. tipo_cadastro. (dados_func |
          dados_usu | dados_obra)

Empréstimo = solicita_empréstimo. #solicita_dados_emp. dados_empréstimo.
             #obra_liberada

Reserva = (f_solicita_reserva | u_solicita_reserva). #solicita_dados_res.
   dados_reserva. (#f_reserva_efetuada | #u_reserva_efetuada)

Pagamento = . . .

Devolução = . . .

Pesquisa = . . .
```

Figura 3.45 - Modelo ciclo-de-vida incompleto

A expressão ciclo-de-vida Reserva se inicia com o evento de solicitação de reserva pelo funcionário f_solicita_reserva ou pelo usuário u_solicita_reserva, seguido pela solicitação do sistema dos dados da reserva #solicita_dados_res, pelo envio dos dados pelo usuário dados_reserva e, finalmente, pelo envio da mensagem reserva efetuada ao solicitante #f_reserva_efetuada ou #u_reserva_efetuada.

**Operation**: dados_reserva
**Description**: Usuário_cadastrado fornece dados para que a reserva seja efetuada.

**Reads**: supplied cod_cadastral, supplied data_reserva, supplied senha; supplied nr_cadastro; qtdd_tombos, nr_tombo, status_tombo, dt_devolução, qtdd_reserva, dt_reserva, cod_reservante.

**Changes**: incrementar 1 em qtdd_reservas,
criação de uma nova reserva com os dados já informados pelo usuário

**Sends**: usuário_cadastrado: {reserva_efetuada}

**Assumes**: a qtdd_tombos deve ser maior que 0, então para todos os nr_tombos obtém-se o status_tombo. Se estiver emprestado, verifica-se se a dt_devolução for menor que a data_reserva, concluir reserva, senão procura outro tombo. Se estiver disponível, verifica existência de reserva. Se não existir reserva, concluir reserva, senão verifica se o nr_cadastro for diferente de cod_reservante, concluir reserva, senão reserva não pode ser feita.
Antes de concluir reserva verificar se a senha_usu é igual a senha.

**Result**: -

Figura 3.46 - Esquema de operação - dados_reserva

O segundo modelo de interfaces, o modelo de operações, é composto por esquemas de operações. Um exemplo de um esquema para a operação dados_reserva é apresentado pela figura 3.46. Para a operação dados_reserva são necessários o cod_cadastral, a data_reserva, a senha e o nr_cadastro como parâmetros e o restante indicado na cláusula *Reads* é obtido pelo sistema. A qtdd_reservas será alterada, um novo item_reserva será criado e o usuário_cadastrado receberá uma mensagem (reserva_efetuada). A cláusula *Assumes* descreve as condições sob as quais uma operação do sistema pode ser ativada. A cláusula Results descreve como o estado do sistema será alterado por uma operação do sistema e quais os eventos que serão enviados aos agentes.

Ao final, todas as operações do sistema devem ser descritas de forma mais detalhada, sendo utilizado um esquema para cada operação. Neste esquema são informados: nome, descrição, nome de itens a serem acessados pela operação, nome de itens a serem acessados e modificados, nome de todos os eventos e agentes que a operação possa enviar, lista de pré-condições e pós-condições. Este esquema é o que melhor captura os comportamentos condicionados pelos eventos do sistema.

Além desses modelos, podem ser utilizados os cenários e os diagramas de tempo para deixar bem claro a comunicação entre os agentes e o sistema.

Os cenários são criados para cada função do sistema, por exemplo: fazer empréstimo ou fazer reserva. Estes cenários mostram os agentes (entidades externas) e a seqüência de informações envolvidas entre estes e o sistema de uma forma textual. Este cenário é idêntico aos cenários descritos pelos métodos OOSE e OOAD.

Cada cenário é representado graficamente pelos diagramas de tempo, onde as operações e os eventos são colocados ordenadamente em relação ao tempo, facilitando a visualização dinâmica do sistema. Um cenário poderá possuir mais de um diagrama de tempo para que a modelagem de caminhos alternativos possa ser feita.

A figura 3.47 ilustra o diagrama de tempo para o cenário de empréstimo, que foi ilustrado na figura 3.3 na seção 3.4. Este diagrama é semelhante ao diagrama de eventos do OMT, porque mostra a interação dos agentes com o sistema, não detalhando a interação dentro do sistema.

Tendo identificado os eventos, os agentes e as operações entre eles, estes devem ser acrescentados ao dicionário de dados, informando nome, categoria, agente (para o primeiro e o terceiro) e uma descrição.

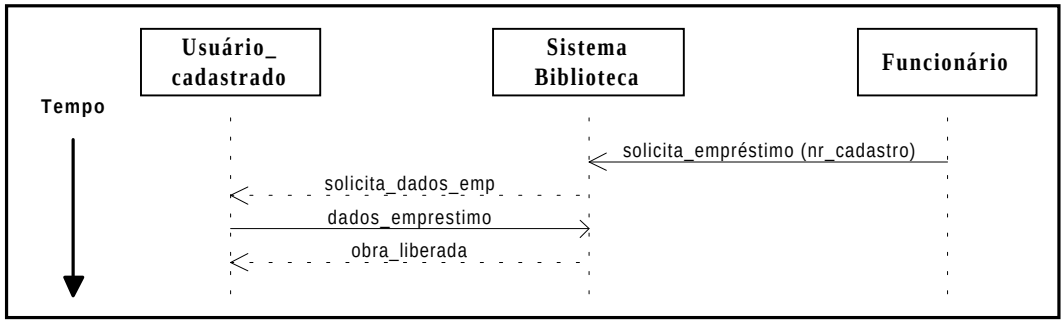



Figura 3.47 - Diagrama de tempo - cenário empréstimo

**c) Projeto**

Ao passar para a fase de projeto, o primeiro modelo que é desenvolvido é o grafo de interação de objetos para cada operação do sistema, tendo como base os esquemas de operações presentes no modelo de operação. O propósito é especificar como a funcionalidade do sistema será implementada pelos objetos contidos no sistema. Cada grafo de interação de objetos possui um texto descritivo, assim como cada operação identificada no grafo, os quais se encontram no dicionário de dados. A figura 3.48 ilustra um exemplo de grafo de interação de objeto para a operação solicita_empréstimo.

A operação solicita_empréstimo tem como parâmetro o nr_cadastro e o objeto f da classe Fichário_usuários como controlador. Este objeto acessa o objeto fu da classe Ficha_usuário, o objeto fe da classe Ficha_empréstimo e o objeto uc da classe Usuário_cadastrado, nesta ordem.

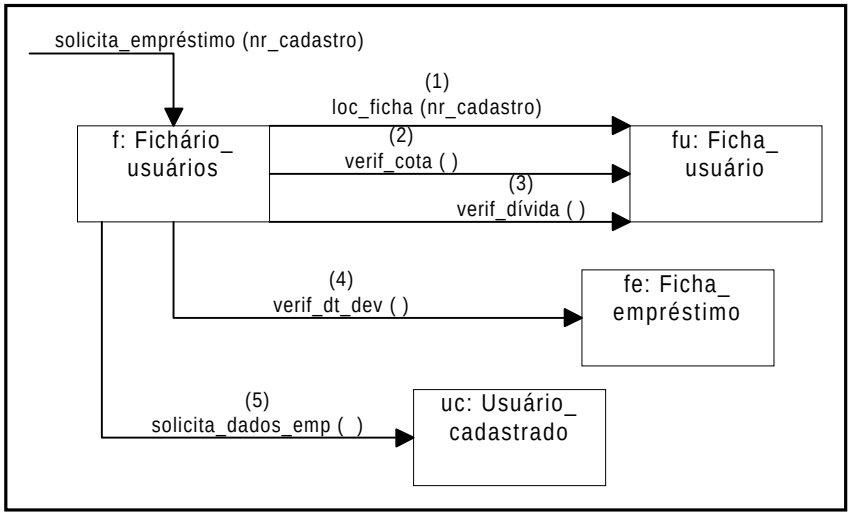



Figura 3.48 - Grafo de interação de objeto - operação solicita_empréstimo

Estes grafos definem as seqüências de mensagens entre os objetos, que são utilizados para realizar a operação. A construção desses grafos forma a primeira fase do projeto onde é considerado que todos os objetos são visíveis entre si.

Depois de gerado todos os grafos de interação de objeto, é necessário que sejam modeladas as estruturas de visibilidade entre as classes, determinando os objetos clientes e os servidores do sistema, pois nem todos os objetos são visíveis por todos os outros. Esse detalhamento é realizado sobre os grafos de interação, utilizando notação apropriada.

Posteriormente, são desenvolvidos os grafos de herança com base nos seguintes itens: i) modelo de objetos do sistema, com as estruturas generalização e especialização identificadas na análise, ii) os grafos de interação de objeto e as descrições de classes, obtendo as operações comuns para gerar classes abstratas e iii) os grafos de visibilidade, fornecendo estruturas que possuam referências em comum, facilitando a identificação de classes abstratas. O exemplo de biblioteca não possui este modelo por não haver herança entre as classes modeladas.

Após a modelagem do projeto, utilizando-se dos grafos e diagramas descritos, são desenvolvidas as descrições de todas as classes do sistema. As descrições de classe modelam: atributos de dados, as operações e seus parâmetros, atributos de objetos e informações sobre herança, obtidos através do modelo de objetos e dicionário de dados, grafos de interação de objetos, grafos de visibilidade e grafos de herança, respectivamente.

As descrições de classe compõem a base para o início da codificação, pois reúnem informações sobre as operações e suas referências. Um exemplo de descrição de classe é ilustrado pela figura 3.49 para a classe Ficha_tombo.

```
class Ficha_tombo
            attribute nr_tombo : int
            attribute dt_devolução : data

            method atu_dt (data)
            method obter_nr_tombo ( )
endclass
```

Figura 3.49 - Descrição da classe ficha_tombo

**d) Outras**

O método aborda a fase de implementação que é a tradução do que foi definido nas descrições de classe e nos grafos de visibilidade. O detalhamento na fase de projeto facilita a implementação do sistema.

### *3.7.4 Transição entre as fases*

A transição da fase de análise para o projeto, segundo os autores, é possível quando os modelos de análise estiverem bons (completos e consistentes) o suficiente para serem utilizados no projeto. A fase de projeto não necessita de modelos perfeitamente corretos, alguns erros triviais são permissíveis. Assim, os modelos de análise devem ser verificados para certificar de que estão completos, capturando todas as abstrações significativas do domínio do problema, e consistentes, significando que não há contradições entre os modelos.

### *3.7.5 Aplicabilidade*

Este método não é indicado para equipes iniciantes em orientação a objetos, pois é um método de ampla cobertura, tornando-se um método de aprendizado demorado. Também não é um método indicado para projetos de baixa complexidade, que poderão fazer uso de métodos mais simples, como por exemplo do método OOA-OOD.

É o método ideal para desenvolvedores que já utilizaram na prática métodos mais simples, e desejam evoluir para um método mais completo.

Segundo (Coleman, 1996), o Fusion pode ser aplicado na reengenharia de sistemas existentes, além do desenvolvimento de novos sistemas. O método também pode ser aplicado a tipos particulares de sistemas concorrentes.

### *3.7.6 Pontos positivos*

O método é composto por partes (diagramas/modelos) de métodos que possuem pontos positivos, melhorando as partes que não obtiveram sucesso, ou que não modelavam o desejado de maneira mais precisa.

A criação dos grafos de interação e das descrições de classe, por possuir riqueza semântica e de detalhes, facilita a fase de implementação.

Outro ponto positivo é a descrição de um *framework* para o dicionário de dados, pois este tem um papel muito importante na verificação dos modelos do sistema.

### *3.7.7 Pontos negativos*

O método Fusion não apresenta nenhum mecanismo para representação dos estados possíveis dos objetos, o que nos outros métodos é possível através dos diagramas de estados. Em alguns tipos de aplicações este tipo de representação faz falta, pois os estados de objetos individuais podem ser bastante significativos.

## *3.8 Considerações finais*

Neste capítulo foi apresentado o processo utilizado para escolher os métodos orientados a objetos, observando a cobertura do método em relação as fases de desenvolvimento, ferramentas CASE que oferece suporte para cada método e se o método faz parte de algum método integrador.

A segunda parte procurou contextualizar o sistema utilizado como exemplo durante a descrição dos métodos, que foi feita na fase seguinte com a identificação de vários aspectos, tais como: descrição, características, abrangência das fases do desenvolvimento, transição entre as fases, aplicabilidade, pontos positivos e negativos; os quais possibilitam uma comparação entre os métodos, que é realizada no próximo capítulo.

# Capítulo 4

## *Comparação entre os métodos orientados a objetos*

Neste capítulo são apresentadas algumas tabelas que reúnem vários aspectos capturados pelos métodos orientados a objetos. Estes aspectos foram identificados com ajuda da modelagem do sistema de biblioteca e que facilita a comparação entre os métodos. A cobertura oferecida pelos métodos em cada fase do desenvolvimento é resumida em uma figura, que é comentada e, posteriormente, as considerações finais referentes ao capítulo são descritas.

### *4.1 Comparação entre os métodos*

A tabela 4.1 apresenta uma descrição explicativa dos aspectos capturados pelos métodos, que foram utilizados para verificar quais deles os métodos abordavam, permitindo compará-los.

A tabela 4.2 apresenta para cada método a fase do desenvolvimento e os modelos ou diagramas onde cada aspecto da tabela 4.1 é descrito. Por exemplo, o aspecto identificação dos requisitos é modelado pelo OOSE na fase de Análise, mais especificamente durante a Análise de requisitos, pelo modelo *use case*.

Com o objetivo de comparar os métodos foi estabelecido o seguinte critério: o método que possuir um modelo com maior quantidade de elementos semânticos, ou seja, maior abrangência dentro do que o modelo propõe, recebe modelagem com nível MÁXIMO e os demais métodos serão avaliados tendo como parâmetro o método de nível MÁXIMO. Por exemplo, se o método OOSE possuir 10 elementos ponderados e recebe 100 % (MÁXIMO), o método OOA-OOD com 6 elementos ponderados recebe 60 %. O critério representa uma regra de três simples, 10 está para 100, assim como 6 está para X, então X = (6 x 100) / 10 => X = 60 %, como mostra a figura 4.1.

$$\begin{array}{l} 10 \;-\; 100 \\ 6 \;-\; X \end{array}$$

$$X = \frac{6 \times 100}{10}$$

$$X = 60\ \%$$

Figura 4.1 - Critério para nivelamento

*Tabela 4.1*
*Descrição explicativa dos aspectos capturados pelos métodos*

| **Aspectos capturados** | **Descrição explicativa** |
|---|---|
| Identificação de requisitos | Modelagem dos requisitos do sistema, identificados pelo usuário ou pelo analista e confirmados pelo usuário. |
| Identificação de eventos | Modelagem da interação que o usuário terá com o sistema. Envio e recebimento de dados ao sistema. |
| Interface do sistema | Modelagem das telas e relatórios que aparecem como interface entre o sistema e o usuário; descrições e/ou gráficos. |
| Estrutura de classes | Identificação das classes do sistema e os relacionamentos entre elas. |
| Estrutura de objetos | Identificação dos objetos do sistema e os relacionamentos entre eles. |
| Descrição de classes | Resumo abrangendo todas as características das classes, tais como: objetivo, atributos e operações, associações e restrições. |
| Identificação de atributos | Identificação dos atributos de cada classe. |
| Identificação de operações | Identificação das operações de cada classe. |
| Troca de mensagens | Modelagem da troca de mensagens entre os objetos do sistema. |
| Relacionamentos | Aborda os relacionamentos existentes de acordo com cada método e seus acompanhamentos que melhor os definem (papel, qualificador, etc). |
| Estrutura de agregação | Modelagem do relacionamento de agregação entre as classes. |
| Estrutura de herança | Modelagem do relacionamento de herança entre as classes. |
| Modelagem de estados | Modelagem dos estados que os objetos de cada classe pode assumir, como os eventos, transições, ações, condições e atividades. |
| Aspectos funcionais | Identificação de funções do sistema. (Nível mais baixo de abstração) |
| Identificação de funcionalidades | Identificação de funcionalidades do sistema. (Nível mais alto de abstração) |
| Criação de subsistemas | Identificação de subsistemas, para facilitar o entendimento do sistema como um todo. |
| Alocação de subsistemas | Alocação de subsistemas a processadores. |
| Estabelecer prioridades | Modelagem das prioridades que os processos tem em relação a utilização do processador. |
| Detalhamento de operações | Modelagem do detalhamento das operações, na forma de algoritmo, apresentando parâmetros de entrada e saída e repetição. |
| Tratamento de concorrência | Modelagem da concorrência entre os objetos, quando dois objetos executam operações ao mesmo tempo. |
| Foco - Análise de requisitos | Apresenta o foco que o método oferece a fase, com a apresentação dos modelos definidos, e assim, acredita-se apresentar uma boa documentação na fase em que possui foco. |
| Foco - Análise | |
| Foco - Projeto | |
| Abordagem - Dados | Define a abordagem que o método possui, se orientada a dados ou orientada ao comportamento; de acordo com a indicação do que deve ser feito primeiro no desenvolvimento. |
| Abordagem - Comportamental | |

*Tabela 4.2*
*Fase e modelos/diagramas onde os aspectos são capturados pelos métodos*

| | **MÉTODOS** | | | | |
|---|---|---|---|---|---|
| **Aspectos capturados** | **OOSE** | **OMT** | **OOAD** | **OOA-OOD** | **Fusion** |
| Identificação dos requisitos | Análise (Análise requis.)<br>Modelo use case | - | - | - | - |
| Identificação de eventos | Projeto<br>Descr. *Use case* / Diagrama de interação | Análise<br>Cenário (textual) / Diagr. de eventos | Análise<br>Cenário (textual) / Diagr, interação | - | Análise<br>Cenário (textual) / Diagr. de tempo |
| Interface do sistema | Análise (Análise requis.)<br>Descrições interface / Modelo Análise | Análise<br>Telas e suas seqüências | Análise<br>Cenário (textual) | Projeto<br>Mod. Proj. - Interação Humana | - |
| Estrutura de classes | Análise (Análise requis.)<br>Modelo domínio do problema | Análise<br>Diagrama de classes | Análise<br>Diagrama de classes | Análise<br>Mod. Análise - classe&objetos | Análise<br>Modelo de objetos |
| Estrutura de objetos | Análise<br>Modelo de análise | Análise<br>Diagrama de instância | Análise<br>Diagrama de objetos | Análise<br>Mod. Análise - classe&objetos | Projeto<br>Grafo interação de objetos |
| Descrição de classes | Análise<br>Descrição de objeto | Análise<br>Dicionário de dados | Análise<br>Especificações (textual) | Análise<br>Especificação de classes | Projeto<br>Descrições de classes |
| Identificação de atributos | Análise<br>Modelo de análise | Análise<br>Diagrama de classes | Análise<br>Diagrama de classes | Análise<br>Mod. Análise - atributos | Análise<br>Modelo de objetos |
| Identificação de operações | Projeto<br>Diagrama de interação | Projeto de objetos<br>Diagrama de classes | Análise<br>Diagrama de classes | Análise<br>Mod. Análise - serviços | Projeto<br>Grafo interação de objetos |
| Troca de mensagens | Projeto<br>Diagrama de interação | Análise<br>Diagrama de instância | Análise<br>Diagr. objetos / Diagr. interação | Análise<br>Mod. Análise - serviços | Projeto<br>Grafo interação de objetos |
| Relacionamentos | Análise (Análise requis.)<br>Modelo domínio do problema | Análise<br>Diagrama de classes | Análise<br>Diagrama de classes | Análise<br>Mod. Análise - classe&objetos | Análise<br>Modelo de objetos |
| Estrutura de agregação | Análise<br>Modelo domínio problema | Análise<br>Diagrama de classes | Análise<br>Diagrama de classes | Análise<br>Mod. Análise - estrutura | Análise<br>Modelo de objeto |
| Estrutura de herança | Análise<br>Modelo domínio problema | Análise<br>Diagrama de classes | Análise<br>Diagrama de classes | Análise<br>Mod. Análise - estrutura | Projeto<br>Grafos de herança |
| Modelagem de estados | Projeto<br>Grafo transição de estado | Análise<br>Diagrama de estados | Projeto<br>Diagrama transição de estado | Análise<br>Diagrama de estado do objeto | - |
| Aspectos funcionais | - | Análise<br>Diagrama de fluxo de dados | - | - | Análise<br>Modelo de operações |
| Identificação de funcionalidades | Análise (Análise requis.)<br>Modelo use case / Descrição | Análise<br>Cenário (textual) / Diagr. de eventos | Análise<br>Cenários (textual) | - | Análise<br>Cenários (textual) |
| Criação de subsistemas | Análise<br>Modelo de análise | Projeto de sistema<br>Diagrama de fluxo de dados | Projeto<br>Diagrama de módulos | Análise<br>Mod. Análise - assunto | - |
| Alocação de subsistemas | Projeto<br>- | Projeto de sistema<br>- | Projeto<br>Diagrama de processo | - | - |
| Estabelecer prioridades | - | Projeto de sistema<br>- | - | Projeto<br>Mod. Proj. - Gerenciam. Tarefas | - |
| Detalhamento de operações | Projeto<br>Diagrama de interação | Projeto de objetos<br>Algoritmos | Análise<br>Especificação de operação | Análise<br>Diagrama de serviço | Análise<br>Modelo de operações |
| Tratamento de concorrência | - | Projeto de sistema<br>- | Projeto<br>Diagrama de objetos | - | - |
| Foco - Análise de requisitos | Análise de requisitos<br>modelos tabela 4.22 | - | - | - | - |
| Foco - Análise | Análise<br>modelos tabela 4.23 | Análise<br>modelos tabela 4.23 | Análise<br>modelos tabela 4.23 | Análise<br>modelos tabela 4.23 | Análise<br>modelos tabela 4.23 |
| Foco - Projeto | Projeto<br>modelos tabela 4.24 | Projeto<br>modelos tabela 4.24 | Projeto<br>modelos tabela 4.24 | Projeto<br>modelos tabela 4.24 | Projeto<br>modelos tabela 4.24 |
| Abordagem - dados | - | Análise<br>tabela 4.25 | - | Análise<br>tabela 4.25 | Análise<br>tabela 4.25 |
| Abordagem - comportamental | Análise<br>tabela 4.26 | - | Análise<br>tabela 4.26 | - | - |

A tabela 4.3 apresenta o nível de modelagem e as faixas adotadas para comparar os métodos. A modelagem com nível MÁXIMO engloba de 75 a 100 % de aproveitamento, o nível MÉDIO ALTO inclui os métodos com 50 a 74.9 %, o nível MÉDIO BAIXO engloba os métodos com 25 a 49.9 % e o nível MÍNIMO inclui os métodos com 0.1 a 24.9 % de modelagem, todos em relação ao método com nível MÁXIMO. O nível inexistente é designado ao método que não possui modelagem.

*Tabela 4.3*
*Nível de modelagem*

| Nível | Valor |
|---|---|
| MÁXIMO | 75 – 100 % |
| MÉDIO ALTO | 50 – 74.9 % |
| MÉDIO BAIXO | 25 – 49.9 % |
| MÍNIMO | 0.1 – 24.9 % |
| - (inexistente) | 0 |

O critério apresentado pela figura 4.3 possibilita verificar mais intuitivamente a comparação realizada entre os métodos, que é ilustrada pela tabela 4.27.

A seguir cada um dos aspectos identificados na tabela 4.2 serão detalhados quanto aos elementos notacionais oferecidos pelos métodos e comentados quanto ao nível recebido segundo o critério estabelecido anteriormente.

É importante ressaltar que os elementos notacionais oferecidos devem ser bem explicados pelos autores em seus livros fundamentais. A avaliação não é feita somente sobre a notação, mas também sobre o detalhamento conceitual oferecido. Este último é o mais importante, tendo em vista, que normalmente a notação a ser utilizada de agora em diante será a UML (detalhes no apêndice D).

Outro aspecto importante é que os elementos notacionais apresentados em cada tabela obterá peso (1) ou peso (2). O peso (2) indica que o elemento será mais utilizado no desenvolvimento de sistemas de complexidade média, tendo em vista o aspecto em questão. O peso (1) refere-se ao elemento adicional oferecido pelo método, o qual será utilizado para detalhar ainda mais a modelagem, portanto, nem sempre será necessário.

*Tabela 4.4*
*Identificação dos requisitos*

| OOSE | OMT | OOAD | OOA-OOD | Fusion |
|---|---|---|---|---|
| (2) *use case*<br>(2) atores<br>(2) descrição | – | – | – | – |
| 6 - 100 % | - | - | - | - |

Pela análise da tabela 4.4, observa-se que o método OOSE é o único que apresenta modelagem para a identificação dos requisitos, com o modelo *use case* e descrição de cada *use case*, recebendo nível MÁXIMO. Enquanto que os outros métodos apenas indicam a necessidade de um documento textual ou o uso de prototipação sem apresentar modelagem.

*Tabela 4.5*
*Identificação de eventos*

| OOSE | OMT | OOAD | OOA-OOD | Fusion |
|---|---|---|---|---|
| (2) descr. *use case*<br>(1) tempo<br>(2) objetos<br>(2) sinal<br>(1) borda<br>(1) script | (2) cenário<br>(1) tempo<br>(1) agente externo<br>(2) eventos<br>(1) sistema | (2) descr. *use case* | _ | (2) cenário<br>(1) tempo<br>(1) agentes<br>(2) eventos<br>(1) sistema |
| 9 - 100 % | 7 - 77.8 % | 2 - 22.2 % | - | 7 - 77.8 % |

O método OOA-OOD não possui modelagem e o método OOAD refere aos modelos do método OOSE. Assim, os demais métodos apresentam modelagem de nível MÁXIMO para a identificação de eventos, conforme mostra a tabela 4.5, porque permitem a visualização imediata dos eventos.

*Tabela 4.6*
*Interface do sistema*

| OOSE | OMT | OOAD | OOA-OOD | Fusion |
|---|---|---|---|---|
| (2) desenho das telas<br>(2) descr. interface<br>(1) objetos interface | (2) desenho das telas<br>(2) fluxo de informação e controle | (2) desenho das telas | (2) desenho das telas<br>(2) estrutura classes | _ |
| 5 - 100 % | 4 - 80 % | 2 - 40 % | 4 - 80 % | - |

A tabela 4.6 apresenta a interface do sistema, onde o método Fusion não modela nem faz referência ao aspecto. Os demais métodos modelam telas e fazem descrições, onde OOSE apresenta o objeto de interface peculiar a ele. Os métodos OOAD e OOA-OOD sugerem o uso de prototipação para auxiliar o trabalho. Os métodos OOSE, OMT e OOA-OOD recebem nível MÁXIMO e o método OOAD recebe nível MÉDIO BAIXO.

*Tabela 4.7*
*Estrutura de classes*

| OOSE | OMT | OOAD | OOA-OOD | Fusion |
|---|---|---|---|---|
| (2) objeto entidade<br>(2) atributo<br>(1) objeto interface<br>(1) objeto controle<br>(1) tipo de atributo<br>(1) cardinalidade atr. | (2) classe<br>(2) atributo<br>(2) operação<br>(2) restrições<br>(1) ordenação | (2) classe<br>(2) atributo<br>(2) operação<br>(1) notas<br>(2) restrições<br>(2) propriedades<br>(1) contr. exportação<br>(1) classe parametriz.<br>(1) categoria classe<br>(1) metaclasse<br>(1) aninhamento class | (2) classe<br>(2) atributo<br>(2) operação<br>(2) classe&objeto | (2) classe<br>(2) atributo<br>(2) cardinalidade |
| 8 - 50 % | 9 - 56.3 % | 16 - 100 % | 8 - 50 % | 6 - 37.5 % |

Através da tabela 4.7, observa-se que o método OOAD possui maior quantidade de elementos notacionais recebendo nível MÁXIMO para a estrutura de classes, por ser o mais completo. Os métodos OOSE, OMT e OOA-OOD recebem nível MÉDIO ALTO porque possui menos elementos. O método Fusion possui modelagem MÍNIMA.

*Tabela 4.8*
*Estrutura de objetos*

| OOSE | OMT | OOAD | OOA-OOD | Fusion |
|---|---|---|---|---|
| (2) objetos interface<br>(2) objetos controle<br>(2) objetos entidade<br>(2) relacionamentos | (2) objetos<br>(2) relacionamentos<br>(1) valores atributos | (2) objetos<br>(2) relacionamentos<br>(1) chave<br>(2) restrição<br>(2) visibilidade<br>(1) notas | (2) objetos<br>(2) relacionamentos | (2) objetos<br>(2) relacionamentos<br>(1) valores<br>(1) coleção objetos<br>(2) visibilidade |
| 8 - 80 % | 5 - 50 % | 10 – 100 % | 4 - 40 % | 8 - 80 % |

Para a estrutura de objetos, os métodos OOSE, OOAD e Fusion têm nível MÁXIMO, pois são mais abrangentes neste aspecto como mostra a tabela 4.8. O método OOSE possui uma abordagem bastante diferente dos outros métodos com a criação de vários tipos de objetos. O método OMT recebe nível MÉDIO ALTO e o método OOA-OOD possui fraca modelagem e obtém nível MÉDIO BAIXO.

*Tabela 4.9*
*Descrição de classes*

| OOSE | OMT | OOAD | OOA-OOD | Fusion |
|---|---|---|---|---|
| (2) desc. obj entidade<br>(2) desc. obj controle<br>(2) desc. obj interface | (2) texto descritivo | (2) texto descritivo<br>(2) atributos<br>(2) operações<br>(1) restrições<br>(1) máquina estados<br>(1) contr. exportação<br>(1) cardinalidade<br>(1) parâmetros<br>(1) persistência<br>(1) concorrência | (1) notas<br>(2) atributos<br>(2) operações<br>(1) especificações adicionais<br>(1) diagrama estado<br>(1) entrada externa<br>(1) saída externa<br>(1) requisit. memória<br>(1) requisitos tempo | (2) superclasse<br>(2) atributos<br>(2) operações<br>(1) tipo de atributo<br>(1) mutabilidade (const / var)<br>(1) parâmetros |
| 6 - 46.2 % | 2 - 15.4 % | 13 - 100 % | 11 - 84.6 % | 9 - 69.2 % |

Através da análise da tabela 4.9, para a descrição de classes, nota-se que o método OOAD e OOA-OOD apresentam nível MÁXIMO, por oferecerem bons modelos para descrever as classes. O método Fusion apresenta um modelo mais simples e recebe nível MÉDIO ALTO. Nível MÍNIMO para o método OOSE e nível MÍNIMO para o método OMT porque apenas menciona o nome que é dado ao modelo, onde se descreve as classes, sem apresentar nenhuma definição formal.

*Tabela 4.10*
*Troca de mensagens*

| OOSE | OMT | OOAD | OOA-OOD | Fusion |
|---|---|---|---|---|
| (2) mensagens<br>(2) objetos<br>(1) seqüência<br>(1) descrição<br>(1) parâmetros | (2) relacionamentos<br>(2) objetos | (2) mensagens<br>(2) objetos<br>(1) seqüência<br>(1) descrição<br>(1) parâmetros<br>(2) sincronização | (2) mensagens<br>(2) objetos | (2) mensagens<br>(2) objetos<br>(1) seqüência<br>(1) retornos<br>(1) parâmetros<br>(1) restrição |
| 7 - 77.8 % | 4 - 44.4 % | 9 - 100 % | 4 - 44.4 % | 8 - 88.9 % |

Os métodos OOSE, OOAD e Fusion possuem nível MÁXIMO para a troca de mensagens, como mostra a tabela 4.10, por modelarem o aspecto com bons elementos de notação, os demais métodos modelam as mensagens sem nenhum detalhe e recebem nível MÉDIO BAIXO.

*Tabela 4.11*
*Relacionamentos*

| OOSE | OMT | OOAD | OOA-OOD | Fusion |
|---|---|---|---|---|
| (2) cardinalidade<br>(2) relacionamentos<br>(2) relacion. *extends*<br>(2) relacion. *uses* | (2) cardinalidade<br>(2) relacionamentos<br>(1) papel<br>(2) atributo ligação<br>(1) chave candidata<br>(2) restrição<br>(2) assoc. ternária<br>(1) qualificador | (2) cardinalidade<br>(1) rel. instanciação<br>(1) papel<br>(2) atrib. associação<br>(1) chave<br>(2) restrição<br>(1) relac. associação<br>(1) relac. metaclasse<br>(1) relacion. usa<br>(1) relacion. tem | (2) cardinalidade<br>(2) relacionamentos | (2) cardinalidade<br>(2) relacionamentos<br>(1) papel<br>(2) atrib. relacionam.<br>(1) marcador totalidd<br>(2) relac. visibilidade |
| 8 - 61.5 % | 13 - 100 % | 13 – 100 % | 4 - 30.8 % | 10 - 76.9 % |

A tabela 4.11 apresenta os elementos notacionais oferecidos pelos métodos para modelagem dos relacionamentos. Os métodos OMT e OOAD modelam com nível MÁXIMO. Os métodos OOSE e Fusion possuem menos elementos e recebem nível MÉDIO ALTO. O método OOA-OOD recebe nível MÉDIO BAIXO devido a fraca modelagem.

*Tabela 4.12*
*Estrutura de agregação*

| OOSE | OMT | OOAD | OOA-OOD | Fusion |
|---|---|---|---|---|
| (2) objeto<br>(2) agregação | (2) classe<br>(2) agregação | (2) classe<br>(2) agregação<br>(1) conteúdo p/ valor<br>(1) conteúdo p/ refer. | (2) classe<br>(2) agregação | (2) classe<br>(2) agregação |
| 4 - 66.7 % | 4 - 66.7 % | 6 – 100 % | 4 - 66.7 % | 4 - 66.7 % |

Pela análise da tabela 4.12, o método OOAD possui nível MÁXIMO por apresentar melhor modelagem ao aspecto de agregação, com dois elementos a mais que os demais métodos não oferecem, os quais recebem nível MÉDIO ALTO.

*Tabela 4.13*
*Estrutura de herança*

| OOSE | OMT | OOAD | OOA-OOD | Fusion |
|---|---|---|---|---|
| (2) classe<br>(2) herança<br>(2) herança múltipla | (2) classe<br>(2) herança<br>(2) herança múltipla<br>(1) classe disjunta<br>(1) classe não disj.<br>(1) discriminador | (2) classe<br>(2) herança<br>(2) herança múltipla | (2) classe<br>(2) herança<br>(2) herança múltipla | (2) classe<br>(2) herança<br>(2) herança múltipla<br>(1) classe disjunta<br>(1) classe não disj. |
| 6 - 66.7 % | 9 - 100 % | 6 - 66.7 % | 6 - 66.7 % | 8 - 88.9 % |

Para a estrutura de herança, apresentada pela tabela 4.13, os métodos OMT e Fusion destacam-se com vários elementos semânticos recebendo nível MÁXIMO, os demais métodos apresentam modelos mais simples e obtêm nível MÉDIO ALTO.

A tabela 4.14 apresenta a modelagem de estados, onde os métodos OOSE, OMT e OOAD possuem nível MÁXIMO, em que o primeiro apresenta um modelo bastante diferente dos dois últimos. O método OOAD apresenta modelo idêntico ao OMT, mas modela menos aspectos

referentes ao estado dos objetos. O método OOA-OOD obtém nível MÉDIO BAIXO porque modela somente os estados e suas transições. O método Fusion não apresenta modelagem.

*Tabela 4.14*
*Modelagem de estados*

| OOSE | OMT | OOAD | OOA-OOD | Fusion |
|---|---|---|---|---|
| (2) estado<br>(2) início<br>(1) rótulo<br>(2) destruir objeto<br>(2) enviar mensagem<br>(2) decisão (cond.)<br>(1) receber mensag.<br>(1) retorno mensag.<br>(1) enviar sinal<br>(1) receber sinal<br>(2) executar tarefa | (2) estado<br>(2) início<br>(2) transição<br>(2) fim<br>(2) evento<br>(2) condição<br>(2) atividade<br>(1) ação de transição<br>(1) ação de entrada<br>(1) ação de saída<br>(1) super-estado<br>(1) sincronizar contr.<br>(1) concorr. diagrama | (2) estado<br>(2) início<br>(2) transição<br>(2) fim<br>(2) evento<br>(2) guarda (condição)<br>(2) atividade<br>(1) ação de entrada<br>(1) ação de saída<br>(1) história<br>(1) super-estado | (2) estado<br>(2) início<br>(2) transição | _ |
| 17 - 85 % | 20 - 100 % | 18 - 90 % | 6 - 30 % | - |

A tabela 4.15 apresenta aspectos funcionais, onde os métodos OMT e Fusion são os únicos a apresentar modelagem funcional e bem detalhada, obtendo nível MÁXIMO.

*Tabela 4.15*
*Aspectos funcionais*

| OOSE | OMT | OOAD | OOA-OOD | Fusion |
|---|---|---|---|---|
| _ | (1) descrição<br>(2) processos<br>(2) atores<br>(2) fluxo de dados<br>(2) fluxo de controle<br>(2) depósito de dados | _ | _ | (1) descrição<br>(2) leitura de valor<br>(2) alteração de valor<br>(2) comunic. agente<br>(1) pré-condições<br>(1) pós-condições |
| - | 11 - 100 % | - | - | 9 - 81.8 % |

A identificação de funcionalidades, apresentada pela tabela 4.16, mostra que os métodos OOSE, OMT e Fusion modelam este aspecto com mais elementos notacionais que os demais. O método OOAD recebe nível MÍNIMO e o método OOA-OOD não apresenta modelagem.

*Tabela 4.16*
*Identificação de funcionalidades*

| OOSE | OMT | OOAD | OOA-OOD | Fusion |
|---|---|---|---|---|
| (1) descr. *use cases*<br>(2) *use case*<br>(2) atores<br>(2) interação | (1) descrição<br>(2) agentes externos<br>(2) sistema<br>(2) interação | (1) descrição | _ | (1) descrição<br>(2) agentes externos<br>(2) sistema<br>(2) interação |
| 7 - 100 % | 7 - 100 % | 1 - 14.3 % | - | 7 - 100 % |

Para a criação de subsistemas, apresentada pela tabela 4.17, com exceção do método Fusion, que não possui modelagem, todos os métodos subdividem sistemas complexos e grandes para sua simplificação.

*Tabela 4.17*
*Criação de subsistemas*

| OOSE | OMT | OOAD | OOA-OOD | Fusion |
|---|---|---|---|---|
| (2) subsistemas | (2) processo | (2) subsistemas | (2) assunto | _ |
| 2 - 100 % | 2 - 100 % | 2 - 100 % | 2 - 100 % | - |

A tabela 4.18 apresenta a alocação de subsistemas, onde observa-se que o método OOAD é o único a oferecer modelagem. Entretanto os métodos OOSE e OMT descrevem sobre o assunto.

*Tabela 4.18*
*Alocação de subsistemas*

| OOSE | OMT | OOAD | OOA-OOD | Fusion |
|---|---|---|---|---|
| _ | _ | (2) processador<br>(2) processos | _ | _ |
| - | - | 4 - 100 % | - | - |

Somente o método OOA-OOD apresenta modelagem para estabelecer prioridades, apresentada pela tabela 4.19, obtendo nível MÁXIMO. Entretanto o método OMT, assim como os demais métodos, faz descrição da necessidade de se modelar sem apresentar modelos.

*Tabela 4.19*
*Estabelecer prioridades*

| OOSE | OMT | OOAD | OOA-OOD | Fusion |
|---|---|---|---|---|
| _ | _ | _ | (2) descrição<br>(2) estrutura classes | _ |
| - | - | - | 4 - 100 % | - |

A tabela 4.20 apresenta o detalhamento de operações, observa-se que os métodos OOSE, OOAD e Fusion recebem nível MÁXIMO, porque apresentam bons esquemas para detalhar as operações. O método OOA-OOD recebe nível MÉDIO ALTO porque apesar de ter uma boa modelagem, esta não é tão abrangente quanto a dos outros métodos. O método OMT possui nível MÉDIO BAIXO porque descreve o assunto sugerindo a utilização de algoritmos.

*Tabela 4.20*
*Detalhamento de operações*

| OOSE | OMT | OOAD | OOA-OOD | Fusion |
|---|---|---|---|---|
| (2) descrição<br>(2) operação<br>(2) parâmetros<br>(2) condição<br>(1) objetos<br>(1) *loop* | (2) descrição<br>(2) operação | (2) descrição<br>(2) operação<br>(2) argumentos<br>(1) pré-condições<br>(1) pós-condições<br>(1) controle export.<br>(2) concorrência<br>(1) tempo complexidade | (2) bloco (descrição)<br>(2) operação<br>(1) *loop*<br>(2) condição<br>(1) sequência | (2) descrição<br>(2) operação<br>(2) parâmetros<br>(1) pré-condição<br>(1) pós-condição<br>(1) vlr. acessados<br>(1) vlr. modificados<br>(1) vlr. enviados usuario |
| 10 - 83.3 % | 4 - 33.3 % | 12 - 100 % | 8 - 66.7 % | 11 - 91.7 % |

Sobre o tratamento de concorrência, apresentado pela tabela 4.21, observa-se que o método OOAD possui excelente modelagem e recebe nível MÁXIMO. Os demais métodos não apresentam modelagem, todavia o método OMT discorre sobre o assunto.

*Tabela 4.21*
*Tratamento de concorrência*

| OOSE | OMT | OOAD | OOA-OOD | Fusion |
|---|---|---|---|---|
| – | – | (2) mensag. síncrona<br>(2) mensag. simples<br>(2) mensag. balking<br>(2) mensag. c/ tempo<br>(2) mens. assíncrona | – | – |
| - | - | 10 - 100 % | - | - |

O método OOSE é o único a possuir foco na análise de requisitos, com vários modelos oferecidos, como pode-se observar na tabela 4.22.

*Tabela 4.22*
*Foco - análise de requisitos*

| OOSE | OMT | OOAD | OOA-OOD | Fusion |
|---|---|---|---|---|
| (2) modelo *use case*<br>(2) descr. *use case*<br>(2) modelo domínio problema<br>(2) modelo interface | – | – | – | – |
| 8 - 100 % | - | - | - | - |

O foco na fase de análise é dado somente pelos métodos OMT e Fusion, que oferecem vários modelos. Os métodos OOSE, OOAD e OOA-OOD apresentam poucos modelos e recebem nível MÉDIO BAIXO, como apresenta a tabela 4.23.

*Tabela 4.23*
*Foco - análise*

| OOSE | OMT | OOAD | OOA-OOD | Fusion |
|---|---|---|---|---|
| (2) modelo de análise<br>(2) descrição objetos<br>(1) diagrama p/ detalhar atributo | (2) diagr. de classes<br>(1) diagr. instâncias<br>(1) diagr. de interface<br>(1) dicionário dados<br>(1) cenários<br>(2) diagr. de eventos<br>(1) diag. fluxo evento<br>(2) diagr. de estados<br>(1) diagr. valores de entrada e saída<br>(2) diagr. fluxo dados<br>(1) descr. funções | (2) diagr. de classes<br>(2) diagr. de objetos<br>(2) diagr. interação | (2) modelo de análise<br>(2) diagr. de serviços<br>(1) diagr. estado obj.<br>(2) especif. classes | (1) dicionário dados<br>(2) modelo de objetos<br>(2) mod. obj. sistema<br>(1) modelo interface<br>(2) mod. ciclo-d-vida<br>(2) diagr. de tempo<br>(1) esquema operação |
| 5 - 33.3 % | 15 - 100 % | 6 - 40 % | 7 - 46.7 % | 11 - 73.3 % |

Como mostra a tabela 4.24, o foco na fase de projeto é melhor abordado pelos métodos OOAD com refinamento dos dois primeiros modelos e criação de outros três e OOA-OOD com o modelo de projeto dividido em quatro componentes, recebendo nível MÁXIMO. O método OOSE com três modelos recebe nível MÉDIO ALTO. O método Fusion com dois modelos recebe nível MÉDIO BAIXO e o método OMT oferece somente o refinamento do diagrama de classes com adição das operações.

*Tabela 4.24*
*Foco - projeto*

| OOSE | OMT | OOAD | OOA-OOD | Fusion |
|---|---|---|---|---|
| (2) modelo de projeto<br>(2) diagr. interação *use case*<br>(2) grafo transição estado | (1) diagr. de classes | (1) diagr. de classes<br>(2) diagr. de objetos<br>(2) diagr. transição estado<br>(2) diagr. módulos<br>(2) diagr. processos | (1) mod. projeto CDP<br>(2) mod. projeto CIH<br>(2) mod. projeto CGT<br>(2) mod. proj. CGD | (2) grafo inter. obj.<br>(2) descr. de classe |
| 6 - 66.7 % | 1 - 11.1 % | 9 - 100 % | 7 - 77.8 % | 4 - 44.4 % |

A tabela 4.25 retrata o quê e como os métodos, no caso OMT, OOA-OOD e Fusion, modelam, que os caracterizam com enfoque orientado a dados. Todos os mencionados acima tem como preocupação primeira, a identificação de classes e objetos, dos atributos e dos relacionamentos. Somente depois é que se identifica as operações, é importante notar que o OMT só faz essa identificação na fase de projeto.

*Tabela 4.25*
*Abordagem - dados*

| OOSE | OMT | OOAD | OOA-OOD | Fusion |
|---|---|---|---|---|
| – | (2) ident. classe/obj.<br>(2) ident. associações<br>(2) ident. atributos | – | (2) id. classe&objetos<br>(2) ident. estruturas<br>(2) ident. atributos | (2) ident. classe/obj.<br>(2) ident. atributos<br>(2) ident. relacionam. |
| - | 6 - 100 % | - | 6 - 100 % | 6 - 100 % |

Os métodos OOSE e OOAD possuem uma abordagem comportamental, como pode ser observado pela tabela 4.26. Estes métodos primeiramente, procuram identificar as funcionalidades do sistema, descrevê-las, para posteriormente identificar classes, atributos e relacionamentos, com base nas funcionalidades definidas.

*Tabela 4.26*
*Abordagem - comportamental*

| OOSE | OMT | OOAD | OOA-OOD | Fusion |
|---|---|---|---|---|
| (2) modelo *use cases*<br>(1) descr. *use cases*<br>(2) descr. interface<br>(2) modelo domínio problema | – | (2) diagrama objetos<br>(2) máquina estados | – | – |
| 7 - 100 % | - | 4 - 57.1 % | - | - |

Observe que os aspectos identificação de atributos e identificação de operações da tabela 4.2, não serão descritos separadamente como os demais, porque a modelagem é de nível MÁXIMO para todos os métodos, para os dois aspectos citados acima, pois todos os métodos apresentam notação e captura desses itens com o mesmo detalhe.

A tabela 4.27 apresenta o nível de modelagem de um método em relação aos outros para cada um dos aspectos descritos, sumariando o que foi apresentado pelas tabelas 4.4 até 4.26.

A seguir é apresentada a cobertura que cada método possui em relação as fases do desenvolvimento.

*Tabela 4.27*
*Resumo dos níveis de modelagem recebidos pelos métodos para cada aspecto capturado*

| | **MÉTODOS** | | | | |
|---|---|---|---|---|---|
| **Aspectos capturados** | **OOSE** | **OMT** | **OOAD** | **OOA-OOD** | **Fusion** |
| Identificação de requisitos | MÁXIMO | - | - | - | - |
| Identificação de eventos | MÁXIMO | MÁXIMO | MÍNIMO | - | MÁXIMO |
| Interface do sistema | MÁXIMO | MÁXIMO | MÉDIO BAIXO | MÁXIMO | - |
| Estrutura de classes | MÉDIO ALTO | MÉDIO ALTO | MÁXIMO | MÉDIO ALTO | MÍNIMO |
| Estrutura de objetos | MÁXIMO | MÉDIO ALTO | MÁXIMO | MÉDIO BAIXO | MÁXIMO |
| Descrição de classes | MÉDIO BAIXO | MÍNIMO | MÁXIMO | MÁXIMO | MÉDIO ALTO |
| Troca de mensagens | MÁXIMO | MÉDIO BAIXO | MÁXIMO | MÉDIO BAIXO | MÁXIMO |
| Relacionamentos | MÉDIO ALTO | MÁXIMO | MÁXIMO | MÉDIO BAIXO | MÉDIO ALTO |
| Estrutura de agregação | MÉDIO ALTO | MÉDIO ALTO | MÁXIMO | MÉDIO ALTO | MÉDIO ALTO |
| Estrutura de herança | MÉDIO ALTO | MÁXIMO | MÉDIO ALTO | MÉDIO ALTO | MÁXIMO |
| Modelagem de estados | MÁXIMO | MÁXIMO | MÁXIMO | MÉDIO BAIXO | - |
| Aspectos funcionais | - | MÁXIMO | - | - | MÁXIMO |
| Ident. de funcionalidades | MÁXIMO | MÁXIMO | MÍNIMO | - | MÁXIMO |
| Criação de subsistemas | MÁXIMO | MÁXIMO | MÁXIMO | MÁXIMO | - |
| Alocação de subsistemas | - | - | MÁXIMO | - | - |
| Estabelecer prioridades | - | - | - | MÁXIMO | - |
| Detalhamento operações | MÁXIMO | MÉDIO BAIXO | MÁXIMO | MÉDIO ALTO | MÁXIMO |
| Tratamento concorrência | - | - | MÁXIMO | - | - |
| Foco - análise requisitos | MÁXIMO | - | - | - | - |
| Foco - análise | MÉDIO BAIXO | MÁXIMO | MÉDIO BAIXO | MÉDIO BAIXO | MÉDIO ALTO |
| Foco - projeto | MÉDIO ALTO | MÍNIMO | MÁXIMO | MÁXIMO | MÉDIO BAIXO |
| Abordagem - dados | - | MÁXIMO | - | MÁXIMO | MÁXIMO |
| Abord. - comportamental | MÁXIMO | - | MÉDIO ALTO | - | - |

## *4.2 Cobertura dos métodos*

Os cinco métodos orientados a objetos apresentados no capítulo 3, seções 3.3, 3.4, 3.5, 3.6 e 3.7, cobrem diferentes fases do desenvolvimento, que é ilustrado pela figura 4.2. Esta figura apresenta as fases do desenvolvimento de *software* tendo como base o modelo cascata (Pressman, 1995). O objetivo é destacar a cobertura que o método oferece em cada uma das fases. Isto foi possível depois do estudo e comparação dos métodos.

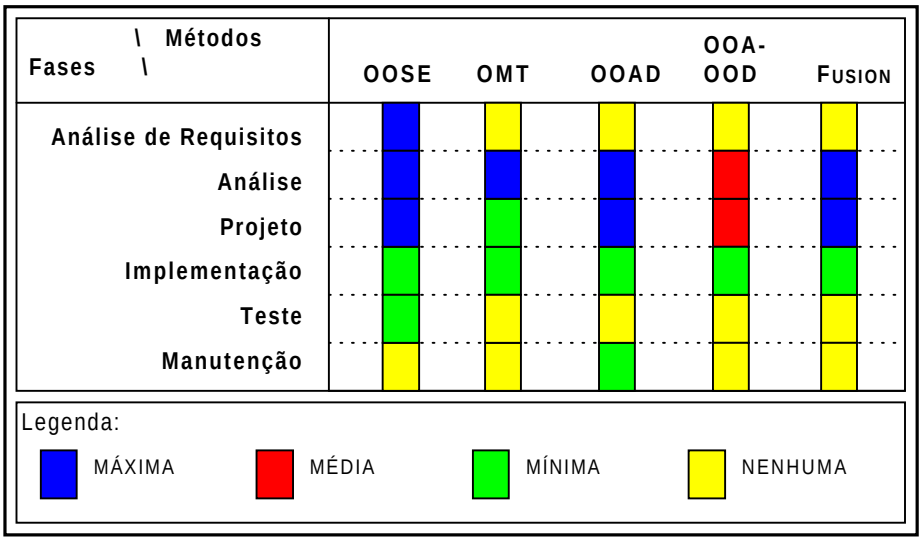



Figura 4.2 - Cobertura dos métodos

O primeiro item da legenda, MÁXIMO, reflete que o método faz uma descrição da fase, apresenta muitos elementos de notação e oferece exemplos conforme a descrição na literatura proposta (Jacobson, 1992), (Rumbaugh, 1994), (Booch, 1994), (Coad, 1992) e (Coad, 1993) e (Coleman, 1996). O segundo item, MÉDIO, indica que há descrição da fase, apresentando modelos com poucos elementos de notação. O terceiro item, MÍNIMO, revela que o método somente descreve a fase sem apresentar modelos ou exemplos. O quarto item, NENHUMA, indica que o método não menciona a fase, não há descrição, exemplificação ou modelos.

Pela figura 4.2 pode-se notar que na fase de análise de requisitos, o método OOSE apresenta cobertura MÁXIMA com vários modelos. O método OOAD descreve a fase e os demais não possuem modelagem. Todos os métodos tem cobertura MÁXIMA na fase de análise, com exceção de OOA-OOD, que oferece pouca notação. Na fase de projeto, os métodos OOSE, OOAD e Fusion se destacam com cobertura MÁXIMA. O método OMT cobre a fase com descrição de passos e o método OOA-OOD cobre com pouca notação. Todos os métodos descrevem a fase de implementação. Com exceção do método OOSE, que descreve a fase de teste e do método OOAD, que descreve a fase de manutenção, os outros métodos não mencionam estas duas últimas fases.

## *4.3 Considerações finais*

Um critério foi definido para possibilitar a comparação entre os métodos. Verifica-se que cada método possui uma melhor modelagem em determinados aspectos. Assim, reforça-se a afirmação de que não existe um método melhor, mas sim um método mais adequado ao desenvolvimento de um determinado domínio de aplicação, que possua características mais apropriadas àquele método.

Com a finalidade de fazer um resumo, foi apresentado a cobertura que os métodos possuem em relação a cada fase do desenvolvimento.

Através do estudo completo dos métodos, pode-se retirar algumas conclusões enumeradas separadamente pelos métodos como segue.

**1. Método OOSE:**

i) É o único a oferecer modelagem e orientação sobre a identificação dos requisitos, os demais partem do princípio de que os requisitos estão bem definidos e devidamente documentados;

ii) A modelagem sugerida define que alguns modelos serão feitos na sub-fase de análise de requisitos, e que alguns modelos de projeto, geralmente definidos pelos outros métodos nesta fase, são feitos na fase de análise, assim, obtém-se um tempo precioso que pode ser despendido durante a fase de projeto.

**2. Método OMT:**

i) Possui modelos abrangentes na fase de análise, onde o sistema é visto por três diferentes dimensões: estático, dinâmico e funcional; onde os demais só apresentam os dois primeiros. Entretanto, é um método com praticamente nenhuma modelagem na fase de projeto;

ii) É um método prescritivo, porque informa exatamente qual é a seqüência de passos a seguir para gerar os modelos .

**3. Método OOAD:**

i) Sua primeira versão continha somente a fase de projeto, em 1991. Posteriormente, foi incorporado a fase de análise. Isso justifica sua forte modelagem em projeto;

ii) Oferece modelos de projeto que capturam aspectos que os outros métodos não mencionam, como visibilidade e concorrência;

iii) Direciona para o uso de técnicas de outros metodologistas como: CRC - Classe Responsabilidade Colaborador[18] - Wirfs-Brock, e cenários e *use cases* - Jacobson.

**4. Método OOA-OOD:**

i) Foi um dos primeiros métodos orientados a objetos a ser desenvolvido (entre as décadas de 80 e 90) (Fichman, 1992), se tornando uma referência para a criação dos outros métodos, os

[18] A técnica CRC é uma técnica exploratória e não um método completo, que foi projetada originalmente para ensinar os conceitos básicos do projeto orientado a objeto (Coleman, 1996).

quais puderam melhorar os pontos deficientes do método base, assim como aperfeiçoar o método como um todo;

ii) Apresenta os modelos multicamadas (análise) e multicomponentes (projeto), onde este último conterá camadas, quer dizer, durante o projeto faz-se o refinamento do modelo de análise;

iii) É um método bastante simples e prescritivo, sendo indicado para desenvolvedores inexperientes.

**5. Método Fusion:**

i) É um método integrador, porque reúne elementos de análise do OMT (modelo de objetos e características de DFD), de projeto do Booch (visibilidade), técnica CRC e as pré e pós condições dos métodos formais, além de apresentar outros modelos interessantes;

ii) É bastante completo, mas não oferece cobertura a todos os aspectos.

Com base na comparação feita entre os métodos, o capítulo 5 apresenta as questões a serem apresentadas ao engenheiro de *software*, a teoria de função de crença, o algoritmo utilizado e todos os detalhes pertinentes.

# Capítulo 5

# *Proposta de um sistema de apoio a decisão utilizando teoria de função de crença*

Neste capítulo é apresentada uma proposta de um sistema, que tem como objetivo auxiliar o engenheiro de *software* a decidir qual método orientado a objetos é o mais adequado no desenvolvimento de um determinado domínio de aplicação.

Para alcançar esse objetivo, o sistema de apoio a decisão (SAD) gera um questionário que é respondido pelo engenheiro de *software* tendo como base o sistema a ser desenvolvido. O SAD faz uma análise sobre as respostas obtidas e fornece uma pontuação aos métodos, entre 0-100, em cada fase do desenvolvimento[19].

Esta análise é realizada com a ajuda da teoria de função de crença definida na próxima seção.

## *5.1 Teoria de função de crença*

Como a teoria de função de crença é utilizada neste trabalho, são descritos seus principais conceitos para melhor compreensão do assunto, visto que esta teoria comumente não é utilizada na engenharia de *software*.

As fontes de consulta para descrição dos conceitos foram (Carvalho, 1996), onde se apresenta uma aplicação da teoria; e (Silva, 1991), onde se obtém um vasto detalhamento sobre o assunto.

A teoria de Dempster e Shafer ou teoria da evidência, mais conhecida como teoria de função de crença é definida em termos de domínios discretos finitos, conhecidos como quadro de discernimento e denotado por Θ.

Um quadro de discernimento (Θ) ou quadro é um conjunto finito de proposições, mutuamente exclusivas, das quais a única verdadeira é desconhecida.

Neste trabalho o quadro utilizado é definido como:

Θ = { a, b, c, d, e } onde: a = método OOSE - Jacobson et. al.;
b = método OMT - Rumbaugh et. al.;
c = método OOAD - Booch;
d = método OOA-OOD - Coad & Yourdon;
e = método Fusion - Coleman et. al..

[19] Os métodos tratados são os mesmos abordados no capítulo 3. As fases de desenvolvimento são as limitadas pelo estudo dos métodos: análise de requisitos, análise e projeto.

Um subconjunto $\mathbb{A}$ de $\Theta$ pode ser visto como uma disjunção das proposições que lhe pertencem. Desta forma, $\mathbb{A}$ também é uma proposição com as seguintes propriedades:

- Se p $\in$ $\mathbb{A}$ e p é a proposição verdadeira, então $\mathbb{A}$ também é uma proposição verdadeira;
- Se $\mathbb{B}$ é um subconjunto de $\Theta$ e $\mathbb{A} \subset \mathbb{B}$ então $\mathbb{A} \rightarrow \mathbb{B}$.

A função de Cr: $2^{\Theta} \rightarrow [0,1]$ satisfaz as seguintes condições:

- Cr ($\varnothing$) = 0
- Cr ($\Theta$) = 1

Para cada subconjunto $\mathbb{A}$ de $\Theta$, o número Cr($\mathbb{A}$) pode ser interpretado como o grau de crença de que a proposição verdadeira está em $\mathbb{A}$. A função de crença Cr pode ser expressa por uma função de densidade de probabilidade Pr da seguinte forma:

Seja $\mathbb{A}$ um subconjunto de $\Theta$ e $\mathbb{X}$ um subconjunto aleatório de $\Theta$, $\mathbb{X} \neq \varnothing$, então:

$$Cr(\mathbb{A}) = Pr[\mathbb{X} \subset \mathbb{A}] = \Sigma \{ Pr[\mathbb{X} = \mathbb{B}] \mid \mathbb{B} \subset \mathbb{A} \}$$

Cr ($\mathbb{A}$) é a probabilidade de $\mathbb{X}$ ser um subconjunto de $\mathbb{A}$.

Para cada função de crença Cr há uma única função m : $2^{\Theta} \rightarrow [0,1]$, chamada massa básica de crença, ou massa de crença (m), tal que m distribui a massa de crença unitária entre os subconjuntos de $\Theta$. Assim, m($\mathbb{A}$) = Pr ($\mathbb{X} = \mathbb{A}$), para todo $\mathbb{A} \subset \Theta$, onde $\mathbb{X}$ é o conjunto aleatório associado a Cr. Suas propriedades são:

$m(\varnothing) = 0$

$\Sigma \{ m(\mathbb{A}) \mid \mathbb{A} \subset \Theta \} = 1$

A quantidade m($\mathbb{A}$) é chamada de crença básica em $\mathbb{A}$ e mede a crença que é atribuída exatamente a $\mathbb{A}$ e a nenhum de seus subconjuntos próprios. A função m($\mathbb{A}$) pode ser vista como a probabilidade do método a ser exatamente igual a $\mathbb{A}$. A função de crença Cr pode ser obtida de sua massa básica de crença m pela equação:

$$Cr(\mathbb{A}) = \Sigma \{ m(\mathbb{B}) \mid \mathbb{B} \subset \mathbb{A} \}, \text{ para todo } \mathbb{A} \subset \Theta.$$

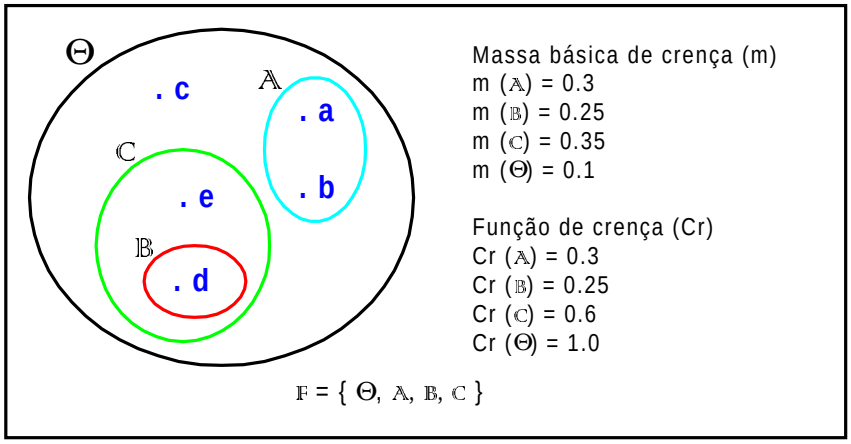



Figura 5.1 - Crença básica em $\mathbb{A}$

A figura 5.1 ilustra um exemplo sobre a massa básica de crença e função de crença. Pela figura observa-se que a massa de crença em $\mathbb{A}$ é de 0.3 e em $\mathbb{B}$ é de 0.25. A função de crença em $\mathbb{A}$

e $\mathcal{B}$ continuam com os mesmos valores atribuídos a suas massas de crenças, mas a função de crença de $\mathcal{C}$ soma as massas de crença de $\mathcal{B}$ e $\mathcal{C}$, porque $\mathcal{B}$ é subconjunto de $\mathcal{C}$. Assim, a função de crença no $\Theta$ é o somatório de todas as massas de crença de seus subconjuntos.

Ao conjunto $\mathcal{F} = \{ \mathcal{A} \mid \mathcal{A} \subset \Theta \text{ e } m(\mathcal{A}) > 0 \}$ é dado o nome de conjunto focal de Cr e seus elementos são chamados elementos focais de Cr.

Apesar da teoria possibilitar trabalhar com todos os subconjuntos de um dado $\Theta$, neste trabalho será aplicado a crença em subconjuntos atômicos de $\Theta$ e no próprio $\Theta$. Isto é, cada subconjunto contém apenas um elemento, um método.

A seguir é apresentado o algoritmo utilizado para combinar as funções de crença, ele é conhecido como Regra de Dempster ou soma ortogonal.

## *5.2 Algoritmo adotado - Regra de Dempster*

Visando obter a combinação das funções de crença definidas, será utilizada regra de Dempster ou regra para formação da soma ortogonal.

- Sejam $Cr_1$ e $Cr_2$ duas funções de crença sobre o $\Theta$

- Seja
  $$K^{-1} = \Sigma \{ m_1(\mathcal{A}_1) \text{ x } m_2(\mathcal{A}_2) \mid \mathcal{A}_1 \in \mathcal{F}_1, \ \mathcal{A}_2 \in \mathcal{F}_2, \ \mathcal{A}_1 \cap \mathcal{A}_2 \neq \varnothing \}$$

- Se $K^{-1} > 0$ e $Cr_1$ e $Cr_2$ são suportadas por evidências independentes, então elas são combináveis e a massa básica de crença m da função resultante é representada por
  $$m(\mathcal{A}) = K \text{ x } \Sigma \{ m_1(\mathcal{A}_1) \text{ x } m_2(\mathcal{A}_2) \mid \mathcal{A}_1 \in \mathcal{F}_1, \ \mathcal{A}_2 \in \mathcal{F}_2, \ \mathcal{A}_1 \cap \mathcal{A}_2 = \mathcal{A} \}$$
  Donde se conclui que a soma ortogonal $Cr = Cr_1 \oplus Cr_2$ tem um conjunto focal
  $$\mathcal{F} = \{ \mathcal{A}_1 \cap \mathcal{A}_2 \mid \mathcal{A}_1 \in \mathcal{F}_1, \ \mathcal{A}_2 \in \mathcal{F}_2, \ \mathcal{A}_1 \cap \mathcal{A}_2 \neq \varnothing \}$$

As funções de crença definidas neste trabalho são chamadas bayesianas, porque todos os elementos focais são atômicos, exceto $\Theta$. Deste modo, o algoritmo a ser utilizado é uma especialização da soma ortogonal, sendo definido em linguagem natural e ilustrado pela figura 5.2.

O algoritmo definido na figura 5.2 combina duas funções de crença bayesianas, onde **m1** e **m2** são vetores que armazenam uma função de crença cada. Estas funções são retiradas do arquivo **AFC** que armazena todas as funções de crença. O algoritmo combina as funções de crença de uma mesma fase. Quer dizer que ao final, obter-se-á a crença nos métodos mais indicados para cada fase (análise de requisitos, análise e projeto). A matriz **mf** armazena a função de crença final para cada fase do desenvolvimento. O procedimento **Normaliza** faz a normalização da função de crença final de cada fase e é executado antes dos valores serem armazenados em **mf**. É importante ressaltar que a normalização é necessária para redistribuir a massa contida em subconjuntos vazios. Isso deve-se a definição $m(\varnothing) = 0$.

```
Principal: SomaOrtogonal
início
        // definição das incógnitas utilizadas
        AFC     { arquivo de funções de crença }
        m1, m2 { vetor local de 6 posições }
        mf      { matriz global de 6 linhas e 3 colunas }
        i, fa, fg  { variáveis globais de controle - indice, fase atual, fase guardada}

        // execução do algoritmo
        m1 ← proximafuncao(AFC)
        fg ← faseatual
        enquanto AFC ≠ final faça
                m2 ← proximafuncao(AFC)
                fa ← faseatual
                se fa ≠ fg
                        então
                                Normaliza(m1)
                                ArmazenaFase(mf, m1)
                                m1 ← m2
                                fg ← fa
                        senão
                                para i de 1 até 5 faça                          1
                                    m1[i] ← m1[i] * m2[i] + m1[i] * m2[0] + m1[0] * m2[i]
                                fim para
                                m1[0] ← m1[0] * m2[0]                           2
                fim se;
        fim enquanto
        Normaliza(m1)
        ArmazenaFase(mf, m1)
fim.

Procedimento: Normaliza (vetor)
início
        SM ← 0;
        para i de 0 até 5 faça
                SM ← SM + m1[i]
        fim para
        para i de 0 até 5 faça
                m1[i] ← m1[i] / SM
        fim para
    fim.

Procedimento: Armazena (matriz por referência, vetor)
início
        para i de 0 até 5 faça
                matriz[i,fa] ← m1[i]
        fim para
fim.
```

Figura 5.2 - Algoritmo em linguagem natural do algoritmo especializado da soma ortogonal

Visando uma melhor compreensão do algoritmo adotado, segue um rastreamento para duas funções de crença.

Sejam m1 e m2 o mapeamento de duas funções de crença:

| Função de crença m1 | Função de crença m2 |
|---|---|
| $m_1(\Theta) = 0$ | $m_2(\Theta) = 0.14$ |
| $m_1(\{a\}) = 0.23$ | $m_2(\{a\}) = 0.27$ |
| $m_1(\{b\}) = 0.18$ | $m_2(\{b\}) = 0.17$ |
| $m_1(\{c\}) = 0.28$ | $m_2(\{c\}) = 0.21$ |
| $m_1(\{d\}) = 0.18$ | $m_2(\{d\}) = 0$ |
| $m_1(\{e\}) = 0.13$ | $m_2(\{e\}) = 0.21$ |

Antes de executar a linha indicada pelo número **1** na figura 5.2, os valores dos vetores m1 e m2 encontram-se como apresentados pela figura 5.3, seguindo o mapeamento definido anteriormente.

| | Θ | {a} | {b} | {c} | {d} | {e} | |
|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | |
| **m1** | 0 | 0.23 | 0.18 | 0.28 | 0.18 | 0.13 | $\mathbb{F}_1$ = { {a}, {b}, {c}, {d}, {e} } |
| **m2** | 0.14 | 0.27 | 0.17 | 0.21 | 0 | 0.21 | $\mathbb{F}_2$ = { {a}, {b}, {c}, {e}, Θ} |

Figura 5.3 - Mapeamento das funções de crença m1 e m2

Após a combinação das duas funções, o vetor m1 apresenta-se como ilustrado pela figura 5.4, após a execução da linha indicada pelo número **2** na figura 5.2.

| | Θ | {a} | {b} | {c} | {d} | {e} | |
|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | |
| **m1** | 0 | 0.0943 | 0.0558 | 0.0980 | 0.0252 | 0.0455 | $\mathbb{F}_1$ = { {a}, {b}, {c}, {d}, {e} } |

Figura 5.4 - Vetor m1 antes da normalização

Após a normalização realizada pelo procedimento Normaliza(vetor), apresentado pela figura 5.2, o vetor m1 encontra-se com os valores como ilustrado pela figura 5.5.

| | Θ | {a} | {b} | {c} | {d} | {e} | |
|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | |
| **m1** | 0 | 0.30 | 0.17 | 0.31 | 0.08 | 0.14 | $\mathbb{F}_1$ = { {a}, {b}, {c}, {d}, {e} } |

Figura 5.5 - Vetor m1 após normalização

Se houvessem mais evidências, o resultado obtido na figura 5.5 seria combinado com a próxima evidência, e assim por diante; como definido pelo algoritmo.

Com o conhecimento da função de crença e do algoritmo, a próxima seção explica com mais detalhes a proposta do sistema de apoio a decisão.

## *5.3 Detalhamento da proposta*

A proposta realizada neste trabalho teve como fonte inspiradora a proposta de Nascimento (Nascimento, 1990). Esta proposta tem como objetivo indicar o método estruturado, o modelo de processo e a abordagem mais indicados no desenvolvimento de um sistema específico. Para tanto, é apresentado um questionário com o objetivo de identificar quais são as características do sistema a ser desenvolvido. Também neste trabalho é utilizado uma técnica de Inteligência Artificial, mais especificamente, um sistema especialista[20] para auxiliar no processo de indicação do método, modelo de processo e abordagem mais apropriados.

A proposta descrita no presente trabalho tem como objetivo apresentar uma pontuação dos métodos orientados a objetos que indica os mais adequados ao desenvolvimento de um determinado domínio de aplicação. Será, então, utilizada a teoria de função de crença para ajudar nesta indicação.

No capítulo 3, foram descritos cinco métodos orientados a objetos e, paralelamente, desenvolvida a modelagem de partes de um sistema de biblioteca. Essa modelagem facilitou o estudo dos métodos e da identificação de seus pontos positivos e negativos.

Após finalização deste estudo foi possível relacionar os aspectos modelados pelos métodos, realizado no capítulo 4. Para cada aspecto foram relatados os elementos notacionais oferecidos e definidos por cada método, expressos em tabelas. Essas tabelas identificam os métodos que melhor modelam cada aspecto.

Neste capítulo são apresentadas as questões definidas com base no trabalho citado anteriormente (Nascimento, 1990) e com ajuda de engenheiros de *software* com grande experiência no desenvolvimento de sistemas. Estas questões possibilitam identificar as características do sistema a ser desenvolvido.

Muitas vezes uma característica do sistema exige que o método modele mais de um dos aspectos definidos no capítulo 4. Por exemplo, se houver muito desconhecimento sobre os requisitos do sistema, o método deve modelar a identificação dos requisitos e das funcionalidades e ter o foco em análise de requisitos. Deste modo, cada questão definida neste trabalho, geralmente, faz o mapeamento para mais de um aspecto capturado pelo método.

A próxima seção apresenta todas as questões e os aspectos relacionados e o mapeamento das evidências (características) aos métodos.

[20] *Um sistema especialista soluciona problemas que normalmente são solucionados por "especialistas" humanos* (Rich, 1994). No trabalho em questão são utilizadas regras de produção, que expressam certeza sobre determinadas evidências. Deste modo, pode-se obter o conhecimento armazenado, segundo regras definidas pelo especialista. Maiores detalhes (Nascimento, 1990).

## *5.4 As questões e o mapeamento das crenças*

Esta seção apresenta 22 questões que compõem o questionário. Cada uma é do tipo que exige resposta Sim ou Não, ou do tipo que exige um grau entre zero e 100, que determina a porcentagem de conhecimento sobre o assunto abordado pela pergunta. Cada questão se encontra em tabela separada, que segue o modelo definido pela tabela 5.1.

*Tabela 5.1*
*Tabela modelo para apresentação das questões*

<table>
<tr><td>A</td><td colspan="3">B</td><td>C</td></tr>
<tr><td colspan="2" rowspan="2">D</td><td>E</td><td colspan="2">F</td></tr>
<tr><td colspan="3">G</td></tr>
<tr><td colspan="5">H</td></tr>
</table>

A tabela 5.1 define cada uma das células utilizadas, como segue:

**A** - número da questão;

**B** - descrição da questão;

**C** - indica o tipo da questão: SN, quando a questão exigir resposta do tipo Sim/Não;

GR, quando for solicitado ao usuário informar o grau de conhecimento;

**D** - relação dos aspectos capturados e respectivas tabelas do capítulo 4, que são utilizadas como parâmetro para determinar os graus de crença nos métodos - peso, número da tabela, descrição do aspecto -;

**E** - número da próxima pergunta se a resposta for SIM ou grau g > 0 %;

**F** - número da próxima pergunta se a resposta for NÃO, grau g = 0 % ou não responder a questão;

**G** - mapeamento para os métodos;

**H** - observações sobre a pergunta, que engloba uma explicação para auxiliar o engenheiro de *software* a responder a pergunta, quando houver necessidade.

As seguintes observações são de grande relevância para melhor entendimento dos critérios adotados:

1. Não há dúvida ou incerteza quanto as respostas oferecidas pelo engenheiro de *software*, que tem como base o sistema que ele pretende desenvolver;

2. Uma única característica ou questão não determina um método ou subconjunto destes, pois todos os métodos podem ser usados em qualquer desenvolvimento, entretanto, o SAD sugere os métodos que oferecem melhor modelagem e suporte conceitual para desenvolver aquele tipo de sistema;

3. Quando o tipo de resposta for SN (Sim/Não), atribui-se grau 100 para Sim e grau zero para Não;

4. O mapeamento é dado pela seguinte fórmula:

$$m(A) = \{ m_2(A) + (m_1(A) - m_2(A)) * g / 100 \mid A \in \{\{a\},\{b\},\{c\},\{d\},\{e\},\Theta\}\}$$

onde: $m_1$ ($\mathcal{A}$) é o mapeamento máximo para $\mathcal{A}$;
$m_2$ ($\mathcal{A}$) é o mapeamento mínimo para $\mathcal{A}$;
g é o grau informado.

5. O engenheiro de *software* não é obrigado a responder a pergunta. Se ele não responder, o mapeamento definido à questão será desconsiderado e a próxima questão é apresentada.

Antes da apresentação das questões, é necessário uma explanação sobre a relação das questões com os aspectos capturados pelos métodos e o mapeamento para os métodos. Existem dois casos: no primeiro, todos os métodos oferecem modelagem em pelo menos um aspecto definido para a questão. No segundo caso, um ou mais métodos não modelam todos os aspectos definidos para a questão. Isto necessita de um mapeamento para o elemento Θ para que o método não seja excluído, como descrito anteriormente.

Outro ponto importante é que dentre o conjunto de tabelas relacionadas para modelagem de uma determinada característica, pode existir casos onde algumas tabelas são mais importantes, recebendo peso 2. Caso contrário, a tabela recebe peso 1 e quer dizer que a modelagem desse aspecto é menos importante para a característica em questão. Cada questão é apresentada em uma tabela, seguindo o modelo definido na tabela 5.1.

A questão 1 se encaixa no primeiro caso e refere-se a sete tabelas, conforme a seguinte descrição: peso 2, tabela 4.7 - Estrutura de classes; peso 1, tabela 4.9 - Descrição de classes; peso 2, tabela 4.11 - Relacionamentos; peso 1, tabela 4.12 - Estrutura de agregação; peso 1, tabela 4.13 - Estrutura de herança; peso 2, tabela 4.22 - Foco - análise de requisitos e peso 2, tabela 4.23 - Foco - análise, conforme indica célula D da tabela 5.2.

*Tabela 5.2*
*Questão 1 - Análise de requisitos*

| 1 | Qual é o grau estimado de desconhecimento do domínio do problema? | | GR |
|---|---|---|---|
| (2) 4.7 - Estrutura de classes<br>(1) 4.9 - Descrição de classes<br>(2) 4.11 - Relacionamentos<br>(1) 4.12 - Estrutura de agregação<br>(1) 4.13 - Estrutura de herança<br>(2) 4.22 - Foco - análise de requisitos<br>(2) 4.23 - Foco - análise | 2 | m({a}) = 0.21 * g / 100<br>m({b}) = 0.22 * g / 100<br>m({c}) = 0.23 * g / 100 | 2<br>m({d}) = 0.15 * g / 100<br>m({e}) = 0.19 * g / 100<br>m(Θ) = 1 - g / 100 |
| Informe se a equipe tem conhecimento sobre o tipo de sistema que se quer desenvolver, ou já desenvolveu sistema semelhantes ou idênticos, ou ainda, se tem algum membro da equipe que conhece o domínio do problema. | | | |

A relação entre essa questão e os aspectos mencionados, deve-se ao fato de que, se não existe conhecimento sobre o domínio do problema, então é necessário que se tenha boa análise, foco em análise de requisitos e em análise, detalhando-a bastante com estrutura de classes e seus relacionamentos. As estruturas de agregações e de heranças e as classes também são importantes. Tudo isso facilita a verificação da corretude da modelagem sobre o domínio.

Para obter um mapeamento da questão aos métodos, primeiro obtém-se uma média dos valores encontrados nas tabelas relacionadas do capítulo 4, como mostra a tabela 5.3 na linha nove,

levando-se em consideração os pesos de cada tabela. Cada valor médio é dividido pela soma de todos os valores médios, obtendo as massas de crença de cada método entre zero e um e soma um.

*Tabela 5.3*
*Valores obtidos para a questão 1*

| Tabela | OOSE | OMT | OOAD | OOA-OOD | Fusion |
|---|---|---|---|---|---|
| 4.7 (2) | 50 | 56.3 | 100 | 50 | 37.5 |
| 4.9 (1) | 46.2 | 15.4 | 100 | 84.6 | 69.2 |
| 4.11 (2) | 61.5 | 100 | 100 | 30.8 | 76.9 |
| 4.12 (1) | 66.7 | 66.7 | 100 | 66.7 | 66.7 |
| 4.13 (1) | 66.7 | 100 | 66.7 | 66.7 | 88.9 |
| 4.22 (2) | 100 | - | - | - | - |
| 4.23 (2) | 33.3 | 100 | 40 | 46.7 | 73.3 |
| Média | 60.84 | 63.15 | 67.88 | 43 | 54.56 |
| Equivalente a 1 | 0.21 | 0.22 | 0.23 | 0.15 | 0.19 |

As massas de crença obtidas são: $m(\{a\}) = 0.21$ $m(\{d\}) = 0.15$
$m(\{b\}) = 0.22$ $m(\{e\}) = 0.19$
$m(\{c\}) = 0.23$ $m(\Theta) = 0$

Seguindo a fórmula definida anteriormente (seção 5.4, item 4), o mapeamento se apresenta como descrito na célula G da tabela 5.2.

*Tabela 5.4*
*Questão 2 - Análise de requisitos*

| 2 | Qual é o grau estimado de desconhecimento sobre os requisitos do sistema? | GR |
|---|---|---|
| (2) 4.4 - Identificação de requisitos<br>(2) 4.16 - Identificação de funcionalidades<br>(2) 4.22 - Foco - análise de requisitos | 3 | 3 |
| | $m(\{a\}) = 0.55 * g / 100$ $m(\{d\}) = 0$<br>$m(\{b\}) = 0.18 * g / 100$ $m(\{e\}) = 0.18 * g / 100$<br>$m(\{c\}) = 0.03 * g / 100$ $m(\Theta) = 1 - 0.94 * g / 100$ | |
| • Indique 0, se o usuário tem certeza do que o sistema deve fazer, quais suas funcionalidades, entradas e saídas.<br>• Indique 100, se o usuário se sente inseguro e não consegue dizer exatamente o que o sistema deve fazer, ou seja, quando o usuário não sabe absolutamente nada. | | |

A questão 2 baseia-se nas tabelas 4.4, 4.16 e 4.22. Quando os requisitos do sistema não estão definidos é interessante adotar um método que ajude na identificação dos requisitos e das funcionalidades do sistema, e que possua foco em análise de requisitos. Assim a tarefa se torna mais fácil e rápida porque são apresentados os passos a serem seguidos para sua realização.

Esta questão pertence ao segundo caso. O processo de obtenção das médias é o mesmo definido para a questão 1 e os valores são baseados nas tabelas descritas na célula D da tabela 5.4. Segundo definido anteriormente, não é permitido o mapeamento somente para os métodos que apresentam modelagem para este aspecto. Desta forma, é adicionado à soma das médias o valor dez, que representa um pequeno grau para o mapeamento do conjunto $\Theta$. O que permite a não exclusão do método. Os valores são descritos pela tabela 5.5.

*Tabela 5.5*
*Valores obtidos para a questão 2*

| Tabela | OOSE | OMT | OOAD | OOA-OOD | Fusion |
|---|---|---|---|---|---|
| 4.4 (2) | 100 | - | - | - | - |
| 4.16 (2) | 100 | 100 | 14.3 | - | 100 |
| 4.22 (2) | 100 | - | - | - | - |
| Média | 100 | 33.33 | 4.77 | - | 33.33 |
| Equivalente a 1 | 0.55 | 0.18 | 0.03 | - | 0.18 |

As massas de crença obtidas são: m({a}) = 0.55 m({d}) = 0
m({b}) = 0.18 m({e}) = 0.18
m({c}) = 0.03 m(Θ) = 0.06

O restante das questões obedecem a um desses dois critérios, que pode ser verificado pelo mapeamento obtido.

*Tabela 5.6*
*Questão 3 - Análise de requisitos*

| 3 | O tempo de vida útil estimado para o sistema é maior que um ano? | | SN |
|---|---|---|---|
| (2) 4.22 - Foco - análise de requisitos<br>(2) 4.23 - Foco - análise<br>(2) 4.24 - Foco - projeto | 4 | 4 | |
| | m({a}) = 0.29 * g / 100<br>m({b}) = 0.16 * g / 100<br>m({c}) = 0.20 * g / 100 | m({d}) = 0.18 * g / 100<br>m({e}) = 0.17 * g / 100<br>m(Θ) = 1 - g / 100 | |
| Qual é o tempo que se pretende utilizar o sistema, ou seja, o tempo que o sistema vai servir aos usuários.<br>Exemplos:<br>• Menos de 1 ano: um sistema que se executa as inscrições em um evento, terá uma vida curta, pois finalizado o evento não haverá mais necessidade de utilizá-lo.<br>• Mais de 1 ano: um sistema bancário tem vida longa, porque é desenvolvido para atender aos usuários por um tempo maior. Os procedimentos utilizados não se modificam como retirar extrato/saldo, transferência entre contas, pagamentos, saques, etc. | | | |

Caso o sistema a ser desenvolvido deva possuir o tempo de vida útil maior que um ano, significa que o mesmo deve ser bem documentado para que a manutenção seja mais rápida, além do mais, sistemas com tempo de vida útil muita grande tendem a possuir maior rotatividade na equipe. Então, escolhe-se preferencialmente um método com foco em todas as seguintes fases do desenvolvimento: análise de requisitos, análise e projeto, pois assim possuirá uma documentação adequada do sistema.

*Tabela 5.7*
*Questão 4 - Análise*

| 4 | Qual a porcentagem estimada do sistema que será reutilizada de sistemas já desenvolvidos? | | GR |
|---|---|---|---|
| (2) 4.10 - Troca de mensagens<br>(2) 4.17 - Criação de subsistemas | 5 | 5 | |
| | m({a}) = 0.21 * g / 100<br>m({b}) = 0.17 * g / 100<br>m({c}) = 0.23 * g / 100 | m({d}) = 0.17 * g / 100<br>m({e}) = 0.22 * g / 100<br>m(Θ) = 1 - g / 100 | |

Se durante o desenvolvimento de um sistema forem reutilizadas partes de sistemas já desenvolvidos, é interessante que o sistema a desenvolver e os já desenvolvidos tenham divisão em subsistemas, porque isso facilita o reuso. Além de que a troca de mensagens é muito importante

para a comunicação entre os subsistemas. Se o sistema a ser reutilizado não tiver sido desenvolvido com idéia de reuso, a prática do mesmo se torna inviável.

*Tabela 5.8*
*Questão 5 - Análise*

| 5 | O sistema todo ou parte dele será reutilizado no desenvolvimento de outros sistemas? | | SN |
|---|---|---|---|
| (2) 4.17 - Criação de subsistemas<br>(1) 4.18 - Alocação de subsistemas<br>(2) 4.23 - Foco - análise<br>(2) 4.24 - Foco - projeto | 6 | 6 | |
| | m({a}) = 0.19 * g / 100<br>m({b}) = 0.20 * g / 100<br>m({c}) = 0.28 * g / 100 | m({d}) = 0.22 * g / 100<br>m({e}) = 0.11 * g / 100<br>m(Θ) = 1 - g / 100 | |

Se já existe a intenção de reusar o sistema em outros desenvolvimentos, a tarefa se tornará mais simples se o sistema possuir uma boa documentação para a análise e o projeto, contendo divisão do sistema em subsistemas. É importante também ter definido a alocação de subsistemas para processadores.

*Tabela 5.9*
*Questão 6 - Análise*

| 6 | Qual é o grau estimado de interação entre o usuário e o sistema? | | GR |
|---|---|---|---|
| (2) 4.6 - Interface do sistema | 7 | 8 | |
| | m({a}) = 0.32 * g / 100<br>m({b}) = 0.26 * g / 100<br>m({c}) = 0.13 * g / 100 | m({d}) = 0.26 * g / 100<br>m({e}) = 0<br>m(Θ) = 1 - 0.97 * g / 100 | |
| Informe proporcionalmente o tempo que o usuário irá interagir com o sistema através de entrada e recuperação de dados.<br>Exemplos:<br>• Grau de intensidade 10 – 40 %: um sistema de ajuste de temperatura tem baixa interação com o usuário, porque somente é necessário o nivelamento da temperatura desejada.<br>• Grau de intensidade 50 – 70 %: um sistema de folha de pagamento tem uma média interação, porque a entrada e recuperação de dados mais pesada acontece uma vez por mês.<br>• Grau de intensidade 80 – 100 %: um sistema de controle bancário, tem alta interação, porque o usuário enquanto utiliza o sistema está entrando com dados (senha, valores) ou selecionando opções. | | | |

Quando o sistema deve possuir muita interação com o usuário, o método deve sugerir modelagem para a interface do sistema para que esta interação seja o mais próximo possível da que foi idealizada pelo solicitador do sistema, referência tabela 5.9.

*Tabela 5.10*
*Questão 7 - Análise*

| 7 | A interface com os usuários terá padrão *Windows*? | | SN |
|---|---|---|---|
| (2) 4.5 - Identificação de eventos<br>(2) 4.10 - Troca de mensagens | 8 | 8 | |
| | m({a}) = 0.28 * g / 100<br>m({b}) = 0.19 * g / 100<br>m({c}) = 0.19 * g / 100 | m({d}) = 0.08 * g / 100<br>m({e}) = 0.26 * g / 100<br>m(Θ) = 1 - g / 100 | |

A interface tendo o padrão *Windows*, o método deve oferecer modelagem para identificação de eventos, já que este tipo de sistema utiliza-se de janelas e botões e troca de mensagens entre os objetos definidos para interface, referência tabela 5.10.

*Tabela 5.11*
*Questão 8 - Análise*

| 8 | Qual é o grau estimado de intensidade (entre 0 e 100%) para a entrada de dados? | | GR |
|---|---|---|---|
| (2) 4.6 - Interface do sistema | 9 | 9 | |
| (2) 4.10 - Troca de mensagens<br>Identificação de atributos | m({a}) = 0.27 * g / 100<br>m({b}) = 0.19 * g / 100<br>m({c}) = 0.21 * g / 100 | m({d}) = 0.19 * g / 100<br>m({e}) = 0.14 * g / 100<br>m(Θ) = 1 - g / 100 | |
| A pergunta quer saber se antes de obter uma resposta do sistema, existe muita entrada de dados. A resposta deve ser dada em porcentagem.<br>Exemplos:<br>• Grau de intensidade 80 - 100%: um sistema como o do IRRF antigo, em que os dados de entrada eram digitados, antes de se obter um resultado é considerado de grande volume.<br>• Grau de intensidade 50 - 70%: um sistema de controle de estoque cuja empresa recebe as mercadorias diariamente, apresenta uma grande entrada de dados.<br>• Grau de intensidade 10 - 40%: 1) um sistema de controle de estoque cuja entrada de dados ocorre mensalmente, em um dia determinado; 2) um sistema de folha de pagamento onde a entrada de dados também é realizada mensalmente. | | | |

Quando existe muita entrada de dados, o sistema deve ser desenvolvido observando principalmente a interface do sistema, a identificação de atributos, porque muitos desses dados serão armazenados e a troca de mensagens para que os objetos possam se comunicar corretamente.

*Tabela 5.12*
*Questão 9 - Análise*

| 9 | Em relação ao sistema como um todo, qual é a porcentagem estimada de cálculos (0% - 100%) que devem ser executados? | | GR |
|---|---|---|---|
| (2) 4.15 - Aspectos funcionais | 10 | 10 | |
| | m({a}) = 0<br>m({b}) = 0.52 * g / 100<br>m({c}) = 0 | m({d}) = 0.43 * g / 100<br>m({e}) = 0<br>m(Θ) = 1 - 0.95 * g / 100 | |
| Informe se o sistema fará cálculos com uso de matrizes, funções trigonométricas, ou outros cálculos do gênero.<br>Exemplos:<br>• Porcentagem entre 80 - 100%: um sistema científico exige grande volume de cálculos.<br>• Porcentagem entre 50 - 70%: um sistema de folha de pagamento apresenta muitos cálculos, mas eles são simples.<br>• Porcentagem entre 10 - 40%: um sistema de controle de estoque apresenta pouquíssimos cálculos. | | | |

Quando o sistema possui muita transformação de dados, como cálculos, o método deve oferecer modelos que abrangem os aspectos funcionais para possibilitar a modelagem das funções que modificam os valores.

Quando os dados armazenados pelo sistema são bastante importantes, o seu desenvolvimento deve possuir abordagem orientada a dados, onde a preocupação se focaliza na identificação e descrição desses dados, ou seja, na descrição das classes.

*Tabela 5.13*
*Questão 10 - Análise*

| 10 | Qual é o grau estimado de importância dos dados no sistema? | | GR |
|---|---|---|---|
| (2) 4.9 - Descrição de classe<br>(2) 4.25 - Abordagem - dados | 11 | 11 | |
| | m({a}) = 0.08 * g / 100<br>m({b}) = 0.19 * g / 100<br>m({c}) = 0.16 * g / 100 | m({d}) = 0.30 * g / 100<br>m({e}) = 0.27 * g / 100<br>m(Θ) = 1 - g / 100 | |
| No caso de perda dos dados, informe qual é o grau de prejuízo para o sistema.<br>Exemplos:<br>• Grau entre 80 - 100%: um sistema que controla uma caldeira, apresenta nos dados a garantia de que não haverá uma explosão ou perda do material, pois serão informados os valores máximo e mínimo permitidos. O mesmo se verifica para um sistema bancário, onde as operações não poderão ser refeitas. Os dados nestes casos são importantíssimos.<br>• Grau entre 50 - 70%: um sistema de folha de pagamento apresenta resultados a partir dos dados entrados. Se houver perda dos mesmos, eles terão que ser reentrados com um esforço considerável. O mesmo se verifica em um sistema de controle de estoque.<br>• Grau entre 10 - 40%: um sistema que apresenta a estrutura de um shopping, como suas lojas e caminho a seguir, não possui alterações nos dados (durante a interface com o usuário). Se caso os dados forem perdidos podem ser obtidos de um backup sem problema algum, porque não será necessário reentrada de dados. Os dados neste caso, têm importância mínima. | | | |

Um sistema que pode sofrer manutenções quanto a modificações de requisitos durante a vida útil, deve ser desenvolvido sobre a orientação de um método que ofereça modelagem detalhada de classes e objetos, criação e alocação de subsistemas, bem como possuir um enfoque em análise de requisitos e análise. Tudo isso permitirá uma manutenção mais rápida, simples e segura.

*Tabela 5.14*
*Questão 11 - Análise*

| 11 | Pode haver mudanças na definição do sistema durante sua vida útil? | | SN |
|---|---|---|---|
| (2) 4.7 - Estrutura de classes<br>(2) 4.8 - Estrutura de objetos<br>(2) 4.17 - Criação de subsistemas<br>(1) 4.18 - Alocação de subsistemas<br>(1) 4.22 - Foco - análise de requisitos<br>(2) 4.23 - Foco - análise | 12 | 12 | |
| | m({a}) = 0.22 * g / 100<br>m({b}) = 0.21 * g / 100<br>m({c}) = 0.27 * g / 100 | m({d}) = 0.16 * g / 100<br>m({e}) = 0.14 * g / 100<br>m(Θ) = 1 - g / 100 | |
| Informe se já existe a intenção de modificar ou incluir alguma funcionalidade ou estrutura ao sistema.<br>• Exemplo: um sistema bancário apresenta grandes mudanças devido aos valores das taxas que variam ou das modificações nas leis. | | | |

Quando o sistema faz controle de equipamentos e/ou máquinas, o método deve oferecer uma abordagem comportamental, preocupando-se principalmente com as operações necessárias; o detalhamento das operações e a troca de mensagens. A modelagem de estados também se torna importante para que compreenda o sistema melhor e possa modelá-lo de forma mais correta, porque o controle se baseia nos estados dos objetos.

*Tabela 5.15*
*Questão 12 - Projeto*

| 12 | O sistema faz controle de máquinas e equipamentos em processos industriais? | SN |
|---|---|---|
| (2) 4.10 - Troca de mensagens<br>(2) 4.14 - Modelagem de estado<br>(1) 4.20 - Detalhamento de operações<br>(2) 4.26 - Abordagem - comportamental | 13<br>m({a}) = 0.30 * g / 100<br>m({b}) = 0.16 * g / 100<br>m({c}) = 0.30 * g / 100 | 13<br>m({d}) = 0.11 * g / 100<br>m({e}) = 0.13 * g / 100<br>m(Θ) = 1 - g / 100 |

Se o controle de tempo real fizer parte do sistema e imprescindível estabelecer as prioridades dos objetos, assim como, tratar a concorrência existente entre os objetos. Abrangendo esses dois pontos o sistema possuirá condições de apresentar as respostas necessárias em tempo hábil.

*Tabela 5.16*
*Questão 13 - Projeto*

| 13 | O sistema deve fazer controle de tempo real? | SN |
|---|---|---|
| (2) 4.19 - Estabelecer prioridades<br>(2) 4.21 - Tratamento de concorrência | 14<br>m({a}) = 0<br>m({b}) = 0<br>m({c}) = 0.45 * g / 100 | 14<br>m({d}) = 0.45 * g / 100<br>m({e}) = 0<br>m(Θ) = 1 - 0.9 * g / 100 |

Um sistema com armazenamento distribuído deve ter uma estrutura de objetos e troca de mensagens bem modeladas, porque existirão muitos objetos e consequentemente muitas mensagens entre eles, para que o sistema atenda a um evento buscando respostas nas várias partes distribuídas do banco de dados. Além disso, um método que tenha abordagem orientada a dados pode auxiliar na modelagem dos dados.

*Tabela 5.17*
*Questão 14 - Projeto*

| 14 | O armazenamento de dados será distribuído? | SN |
|---|---|---|
| (2) 4.8 - Estrutura de objetos<br>(2) 4.10 - Troca de mensagens<br>(1) 4.25 - Abordagem - dados | 15<br>m({a}) = 0.18 * g / 100<br>m({b}) = 0.17 * g / 100<br>m({c}) = 0.23 * g / 100 | 15<br>m({d}) = 0.16 * g / 100<br>m({e}) = 0.26 * g / 100<br>m(Θ) = 1 - g / 100 |
| Informe se o banco de dados estará dividido em várias CPU's.<br>• Exemplo: um sistema bancário poderá ter sua base de dados distribuída pelas cidades, de acordo com os clientes de cada uma. Assim, teria um banco de dados em Brasília para todos os clientes desta cidade, e assim por diante.<br>Obs.: O sistema estará distribuído, mas não estará isolado, porque se um cliente de Anápolis tentar retirar dinheiro em Brasília será perfeitamente possível. | | |

Se a arquitetura do sistema para o acesso aos dados é de clientes remotos com mais de um servidor, é necessário a modelagem da estrutura dos objetos, a troca de mensagens entre eles e o tratamento de concorrência, além de identificar os subsistemas e alocá-los aos processadores apropriados. Esta completa modelagem permite que o acesso aos dados seja distribuído, pois será feito um controle apropriado.

*Tabela 5.18*
*Questão 15 - Projeto*

| 15 | A arquitetura a ser utilizada para o acesso aos dados é de clientes remotos com mais de um servidor? | SN |
|---|---|---|
| (2) 4.8 - Estrutura de objetos<br>(2) 4.10 - Troca de mensagens<br>(2) 4.17 - Criação de subsistemas<br>(2) 4.18 - Alocação de subsistemas<br>(2) 4.21 - Tratamento de concorrência | 16 | 17 |
| | m({a}) = 0.20 * g / 100<br>m({b}) = 0.15 * g / 100<br>m({c}) = 0.38 * g / 100 | m({d}) = 0.14 * g / 100<br>m({e}) = 0.13 * g / 100<br>m(Θ) = 1 - g / 100 |

O desenvolvimento de um protocolo aborda a modelagem de estado e troca de mensagens e requer uma abordagem comportamental. Estes aspectos detalham a comunicação e verificação existente em um protocolo.

*Tabela 5.19*
*Questão 16 - Projeto*

| 16 | O sistema necessitará do desenvolvimento de protocolos entre clientes e servidores ou entre servidores? | SN |
|---|---|---|
| (2) 4.10 - Troca de mensagens<br>(2) 4.14 - Modelagem de estados<br>(1) 4.26 - Abordagem - comportamental | 20 | 20 |
| | m({a}) = 0.29 * g / 100<br>m({b}) = 0.20 * g / 100<br>m({c}) = 0.29 * g / 100 | m({d}) = 0.10 * g / 100<br>m({e}) = 0.12 * g / 100<br>m(Θ) = 1 - g / 100 |

Se a arquitetura para o acesso aos dados é de clientes remotos com um único servidor, a modelagem deve cobrir a estrutura de objetos, troca de mensagens e tratamento de concorrência. Como será utilizado somente um servidor, não torna-se necessário a modelagem de subsistemas e sua alocação a processadores, com a ênfase definida para questão 15.

*Tabela 5.20*
*Questão 17 - Projeto*

| 17 | Então, a arquitetura para o acesso aos dados é de clientes remotos com servidor central? | SN |
|---|---|---|
| (2) 4.8 - Estrutura de objetos<br>(2) 4.10 - Troca de mensagens<br>(2) 4.21 - Tratamento de concorrência | 18 | 19 |
| | m({a}) = 0.20 * g / 100<br>m({b}) = 0.12 * g / 100<br>m({c}) = 0.37 * g / 100 | m({d}) = 0.10 * g / 100<br>m({e}) = 0.21 * g / 100<br>m(Θ) = 1 - g / 100 |

Se houver a necessidade do desenvolvimento de protocolos entre os clientes e o servidor, os mesmos aspectos cobertos para a questão 16 devem também ser modelados aqui.

*Tabela 5.21*
*Questão 18 - Projeto*

| 18 | O sistema necessitará do desenvolvimento de protocolos entre clientes e servidor? | SN |
|---|---|---|
| (2) 4.10 - Troca de mensagens<br>(2) 4.14 - Modelagem estados<br>(1) 4.26 - Abordagem - comportamental | 20 | 20 |
| | m({a}) = 0.29 * g / 100<br>m({b}) = 0.20 * g / 100<br>m({c}) = 0.29 * g / 100 | m({d}) = 0.10 * g / 100<br>m({e}) = 0.12 * g / 100<br>m(Θ) = 1 - g / 100 |

O uso de clientes e servidor locais para o acesso aos dados requer uma modelagem igual a abordada para a questão 17, sendo descrita na tabela 5.22.

*Tabela 5.22*
*Questão 19 - Projeto*

| 19 | Para o acesso aos dados, a arquitetura utilizada será de clientes e servidor local? | SN |
|---|---|---|
| (2) 4.8 - Estrutura de objetos<br>(2) 4.10 - Troca de mensagens<br>(2) 4.21 - Tratamento de concorrência | 20<br>m({a}) = 0.20 * g / 100<br>m({b}) = 0.12 * g / 100<br>m({c}) = 0.37 * g / 100 | 20<br>m({d}) = 0.10 * g / 100<br>m({e}) = 0.21 * g / 100<br>m(Θ) = 1 - g / 100 |

Quanto maior for o grau de pesquisa no conjunto de dados, maior será a necessidade de uma modelagem completa da estrutura de objetos e do tratamento de concorrência, definindo se haverá vários acessos simultâneos ao mesmo objeto ou não (tanto aos atributos quanto às operações). A troca de mensagens também deve ser modelada.

*Tabela 5.23*
*Questão 20 - Projeto*

| 20 | Informe o grau estimado de pesquisa realizada no conjunto de dados armazenados. | GR |
|---|---|---|
| (2) 4.8 - Estrutura de objetos<br>(1) 4.10 - Troca de mensagens<br>(2) 4.21 - Tratamento de concorrência | 21<br>m({a}) = 0.19 * g / 100<br>m({b}) = 0.12 * g / 100<br>m({c}) = 0.39 * g / 100 | 21<br>m({d}) = 0.10 * g / 100<br>m({e}) = 0.20 * g / 100<br>m(Θ) = 1 - g / 100 |
| Indique se haverá necessidade de acesso ao banco de dados.<br>Exemplos:<br>• Grau 0%: um sistema que controla temperatura de um caldeirão não apresenta quase nenhum acesso, pois os dados são obtidos por entidades externas.<br>• Grau ente 10 - 50%: um sistema de folha de pagamento baseia-se em cálculos sobre os dados armazenados, mas não há pesquisa nos dados e, sim, uma busca desses valores.<br>• Grau entre 60 - 100%: um sistema de controle de biblioteca baseia-se fundamentalmente em verificações na base de dados, porque requer verificações sobre empréstimos, reservas, quantidade emprestada a um determinado usuário, etc. | | |

Se o banco de dados orientado a objetos será utilizado pelo sistema, então a modelagem deve ser detalhada na estrutura de classes e dos objetos, onde se modela os relacionamentos e a visibilidade, e no tratamento de concorrência, modelando a concorrência entre os objetos.

*Tabela 5.24*
*Questão 21 - Projeto*

| 21 | Pretende-se utilizar Banco de Dados Orientados a Objetos? | SN |
|---|---|---|
| (1) 4.7 - Estrutura de classes<br>(2) 4.8 - Estrutura de objetos<br>(2) 4.21 - Tratamento de concorrência | 0<br>m({a}) = 0.18 * g / 100<br>m({b}) = 0.13 * g / 100<br>m({c}) = 0.41 * g / 100 | 22<br>m({d}) = 0.11 * g / 100<br>m({e}) = 0.17 * g / 100<br>m(Θ) = 1 - g / 100 |

Se o banco de dados relacional é que será utilizado, então a modelagem sobre os relacionamento deve ser enfatizada, assim como a abordagem orientada a dados. Estes são os pontos fortes em um sistema relacional, o relacionamento dos dados armazenados.

*Tabela 5.25*
*Questão 22 - Projeto*

| | | |
|---|---|---|
| 22 | Então, pretende-se utilizar Banco de Dados Relacional? | SN |
| (2) 4.11 - Relacionamentos | 0 | 0 |
| (2) 4.25 - Abordagem - dados | m({a}) = 0.09 * g / 100<br>m({b}) = 0.30 * g / 100<br>m({c}) = 0.15 * g / 100 | m({d}) = 0.20 * g / 100<br>m({e}) = 0.26 * g / 100<br>m(Θ) = 1 - g / 100 |

## 5.5 Considerações finais

As perguntas apresentadas avaliam a parte técnica de um desenvolvimento, então, para uma completa visão seria necessário ainda a avaliação da parte gerencial, que não é abordada neste trabalho.

Como complemento dessa abordagem, poderia ser aplicado a teoria de função de perda, que determina a perda associada a todos os casos onde o SAD sugere um método e o engenheiro de *software* adota outro. Ao final é adotado o método que oferece menor perda.

Com a proposta definida, o próximo capítulo descreve o protótipo desenvolvido, que implementa a proposta.

# Capítulo 6

## *SGMOO: o protótipo*

O Sistema Gestor de Métodos Orientados a Objetos - SGMOO é um protótipo que fornece a indicação dos métodos mais adequados ao desenvolvimento de um determinado sistema, mediante respostas obtidas através do questionário gerado pelo protótipo. O SGMOO é uma implementação da proposta definida no capítulo anterior.

O presente capítulo apresenta o desenvolvimento do SGMOO, sua arquitetura de implementação, que apresenta o seu funcionamento em alto nível de abstração, descrição da base de conhecimento, ou seja, como os dados são armazenados e como eles se relacionam, as telas desenvolvidas durante a implementação, os resultados obtidos pela aplicação de quatro simulações de casos reais e as considerações finais.

São aplicados ao SGMOO simulações de casos reais porque a busca por descrições de casos reais em empresas de desenvolvimento ou organizações, seria um trabalho muito dispendioso de tempo.

### *6.1 Descrição do uso do protótipo*

O objetivo principal do SGMOO[21] é apresentar a interface do sistema para os dois tipos de usuários, como descritos a seguir, e a implementação do algoritmo de Dempster e Shafer (seção 5.2). A criação de um sistema completo abrangendo todos os detalhes não se caracteriza uma meta.

A figura 6.1 apresenta um esquema de interação entre os dois tipos de usuários e o SGMOO, apresentando interface para o especialista em métodos e para o engenheiro de *software*, separadamente.

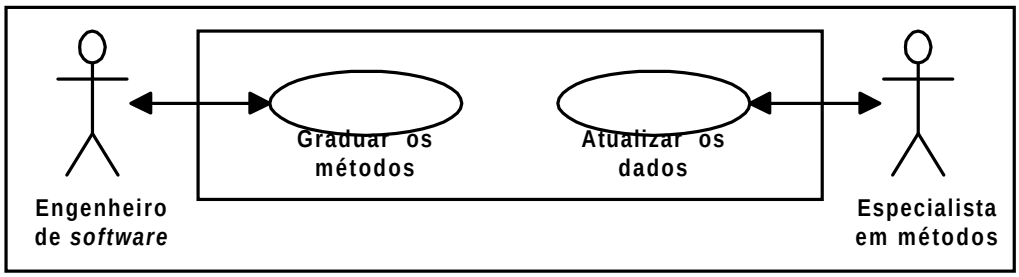



Figura 6.1 - Interação entre os dois tipos de usuários e o SGMOO

[21] O SGMOO foi desenvolvido em *Visual Basic for Application*. Esta linguagem oferece fácil manuseamento e rápido desenvolvimento.

O especialista em métodos atualizará os dados necessários para a graduação dos métodos, como: os métodos disponíveis, as questões para o questionário, as fases de desenvolvimento oferecidas e o mapeamento das funções de crença. A qualquer momento esta atualização será possível.

Visando graduar os métodos, o SGMOO gera um questionário disponível para o engenheiro de *software*, que fornece as respostas tendo como base o sistema que ele pretende desenvolver. Com as respostas, que caracterizam as evidências (definidas no capítulo 5), aplica-se o algoritmo de Dempster e Shafer (figura 5.2) sobre os mapeamentos das funções de crença. O resultado será um quadro com a graduação da aplicabilidade de cada método em cada fase do desenvolvimento do sistema em questão e, também, o método mais graduado em todas as fases podendo ser utilizado em todo o desenvolvimento. Este quadro é ilustrado pela figura 6.2.

**Resultado**

| | An. Req. | Análise | Projeto | Geral |
|---|---|---|---|---|
| Todos | 0,00 | 0,00 | 0,00 | 0,00 |
| OOSE - Jacobson et. al. | 31,30 | 28,60 | 13,10 | 23,08 |
| OMT - Rumbaugh et. al. | 15,90 | 28,60 | 5,30 | 4,73 |
| OOAD - Booch | 19,30 | 28,60 | 64,70 | 70,41 |
| OOA-OOD - Coad & Yourdon | 16,70 | 14,30 | 3,70 | 1,78 |
| Fusion - Coleman et. al. | 16,70 | 0,00 | 13,10 | 0,00 |

Figura 6.2 - Quadro de graduação dos métodos em cada fase do desenvolvimento

Neste quadro estão relacionados todos os métodos e suas crenças, que definem o quanto o método é apropriado para o desenvolvimento de um sistema em uma dada fase e o método que pode ser utilizado em todas as fases (Geral)[22].

A arquitetura de implementação do SGMOO é apresentada a seguir.

## *6.2 Arquitetura de implementação do SGMOO*

A figura 6.3 apresenta a arquitetura do protótipo composta por três módulos:

1. **serviços para o engenheiro** que apresenta o questionário, obtém e armazena as respostas, faz a combinação das funções de crença e envia ao engenheiro de *software* a graduação de cada método, separadamente, por fases;
2. **base de conhecimento** que armazena as fases, os métodos, as questões e as funções de crenças definidas pelo especialista, bem como um arquivo temporário das respostas obtidas do engenheiro de *software*;

[22] Este resultado geral é a combinação dos resultados obtidos para cada fase, de acordo com o algoritmo de Dempster e Shafer seção 5.2.

3. **serviços para o especialista** que permite o especialista em métodos fazer manutenções na base de conhecimento, com inclusões, exclusões e alterações.

A **base de conhecimento** armazena as fases, os métodos e as questões identificadas através de entrevistas com engenheiros de *software*, que possuem larga experiência no desenvolvimento de diversos domínios da aplicação, considerando os aspectos relevantes no desenvolvimento. Também armazena um arquivo temporário das respostas obtidas através do questionário e as funções de crença definidas pelo especialista mediante um estudo aprofundado dos métodos.

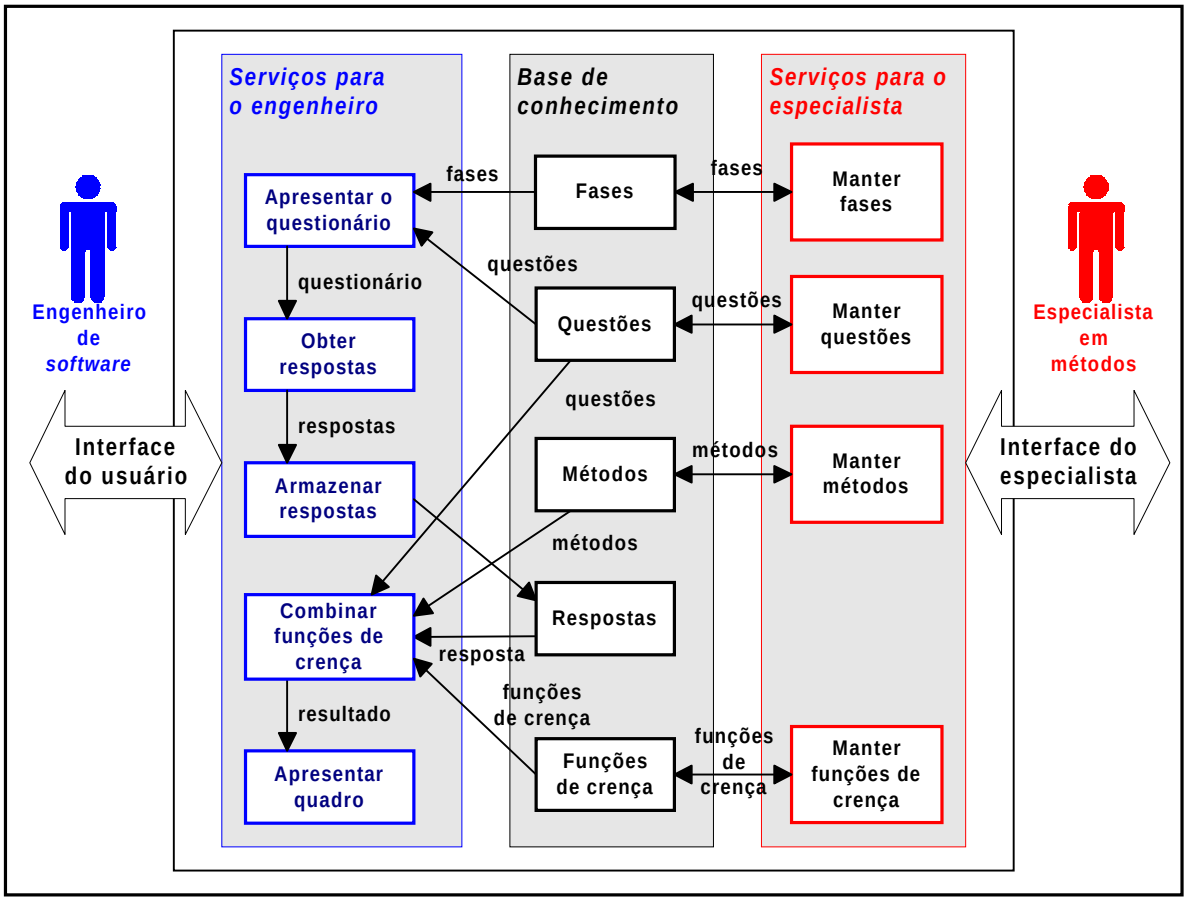



Figura 6.3 - Arquitetura do protótipo

Somente o especialista em métodos poderá fazer manutenções na base de conhecimento através do módulo **serviços para o especialista**.

A interação entre o engenheiro de *software* e o sistema será através do módulo **serviços para o engenheiro**. Este módulo inclui os seguintes serviços: a) apresentar o questionário para o engenheiro de *software*, com base nas perguntas definidas pelo especialista, que são separadas por fases, mas que são transparentes ao engenheiro de *software*, b) obter as respostas do questionário, possibilitando determinar, além das características do sistema, a próxima pergunta, dependente da resposta anterior, c) armazenar as respostas temporariamente para posterior serviço de combinar as funções de crença, para apresentar o quadro ao engenheiro de *software*.

O resultado obtido é uma associação bem fundamentada, que pode ser utilizada como guia no desenvolvimento de qualquer sistema, gerando uma qualidade e rendimento superior aos

encontrados atualmente, devido ao fato de se utilizar o método mais adequado ao domínio da aplicação. Entretanto, esse guia não garante o sucesso do desenvolvimento, o qual será medido também por outros aspectos, como: tamanho e treinamento da equipe de desenvolvimento, alterações de requisitos, etc.

Uma das principais propriedades do protótipo é prover permissão de futuras inclusões ou modificações na base de conhecimento, com informações sobre novas fases, questões, métodos e funções de crença, seja pelo aperfeiçoamento do estudo feito ou pelo estudo de novos métodos.

A seguir são apresentados detalhes da base de conhecimento, sua estrutura e seus relacionamentos.

## *6.3 Descrição da base de conhecimento*

A base de conhecimento é composta por cinco tabelas: Crença, Fase, Método, Pergunta e Resposta (o arquivo temporário). O nome dos campos[23], os tipos de dados e uma descrição para cada tabela, citada anteriormente, são relacionados pelas tabelas 6.1 a 6.5.

A tabela 6.1 armazena os mapeamentos das crenças nos métodos para cada tipo de resposta de cada pergunta. Se o mapeamento for máximo recebe código 1, se for mínimo recebe código 2. O valor da crença varia entre 0 e 1, como já definido no capítulo 5.

*Tabela 6.1*
*Crença*

| Nome do campo | Tipo de dados | Descrição |
|---|---|---|
| codperg 🔑 | Número | Código da pergunta |
| codcren 🔑 | Número | Código da crença (1 ou 2) |
| codmeto 🔑 | Número | Método |
| vlrcren | Número | Valor da crença |

A tabela 6.2 armazena as fases que são relacionadas na apresentação do quadro de graduação. Neste trabalho, são abordadas somente as fases de análise de requisitos, análise e projeto.

*Tabela 6.2*
*Fase*

| Nome do campo | Tipo de dados | Descrição |
|---|---|---|
| codfase 🔑 | Número | Código da fase |
| desfase | Texto | Descrição da fase |

A tabela 6.3 armazena os métodos que o SGMOO utiliza e para os quais pode-se fazer mapeamentos. Este trabalho aborda cinco métodos cujos códigos são assim descritos:

0 - Todos; 1 - OOSE; 2 - OMT;
3 - OOAD; 4 - OOA-OOD; 5 - Fusion.

[23] Os campos que apresentam uma chave, indicam que fazem parte da chave primária da tabela.

*Tabela 6.3*
*Método*

| Nome do campo | Tipo de dados | Descrição |
|---|---|---|
| codmeto | Número | Código do método |
| desmeto | Texto | Descrição do método |

A tabela 6.4 relaciona as perguntas que formam o questionário. As perguntas podem ser do tipo Sim ou Não ou para informar o grau. Ao final de cada resposta é feita uma próxima pergunta. Na maioria das vezes isso é independente da resposta, ou seja, a próxima pergunta é única; mas em outras são consideradas determinantes, quando a próxima pergunta depende da resposta anterior. A observação da pergunta é como uma ajuda para que o engenheiro de *software* possa compreender melhor o que está sendo questionado. Cada pergunta pertence a uma determinada fase, o que torna-se transparente através do questionário, somente o especialista em métodos tem conhecimento.

*Tabela 6.4*
*Pergunta*

| Nome do campo | Tipo de dados | Descrição |
|---|---|---|
| codperg | Número | Código da pergunta |
| desperg | Texto | Descrição da pergunta |
| tipperg | Texto | Tipo da pergunta (SN - Sim/Não, GR - Informar o grau) |
| propergS | Número | Próxima pergunta (100% ou outra <> 0% - Sim) |
| propergN | Número | Próxima pergunta (0% - Não - Não informado) |
| obsperg | Texto | Observação da pergunta |
| codfase | Número | Código da fase |

A tabela 6.5 armazena informações sobre as respostas obtidas através do questionário, armazenando o grau informado para a pergunta e qual a pergunta anterior, para correção de eventual erro na resposta anterior.

*Tabela 6.5*
*Resposta*

| Nome do campo | Tipo de dados | Descrição |
|---|---|---|
| codperg | Número | Código da pergunta |
| graresp | Número | Grau da resposta |
| anterior | Número | Pergunta anterior |
| codcren | Número | Código da crença |

A figura 6.4 apresenta os relacionamentos entre as tabelas descritas anteriormente. Cada pergunta pode ter uma resposta informada pelo engenheiro. É importante observar que a pergunta deve pertencer a uma fase, podendo ter várias crenças para cada resposta e dois alto-relacionamentos. Isto ocorre porque a pergunta seguinte depende da resposta informada. Cada crença é mapeada somente para um método.

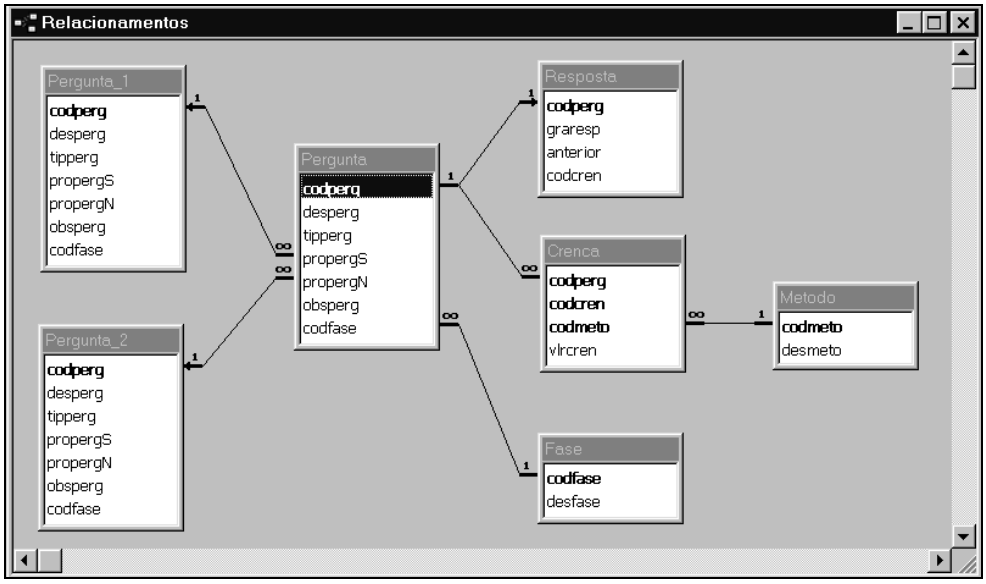



Figura 6.4 - Relacionamentos das tabelas da base de conhecimento

## *6.4 Apresentação das telas*

Nesta seção são apresentadas as telas mais importantes do protótipo, seguindo uma rápida explanação sobre cada uma.

A figura 6.5 apresenta a tela inicial do protótipo onde define qual a categoria do usuário a utilizar o sistema, pode ser o engenheiro de *software* representado pelo botão da esquerda ou o especialista representado pelo botão da direita. O botão na parte inferior permite sair do sistema.


Figura 6.5 - Tela inicial do protótipo

As próximas três figuras são apresentadas ao especialista em métodos.

A figura 6.6 apresenta o formulário de menu de navegação para o especialista em métodos, onde ele poderá modificar os formulários listados no menu ou fazer inclusão/alteração de perguntas e crenças. A última opção do menu retorna a tela inicial.

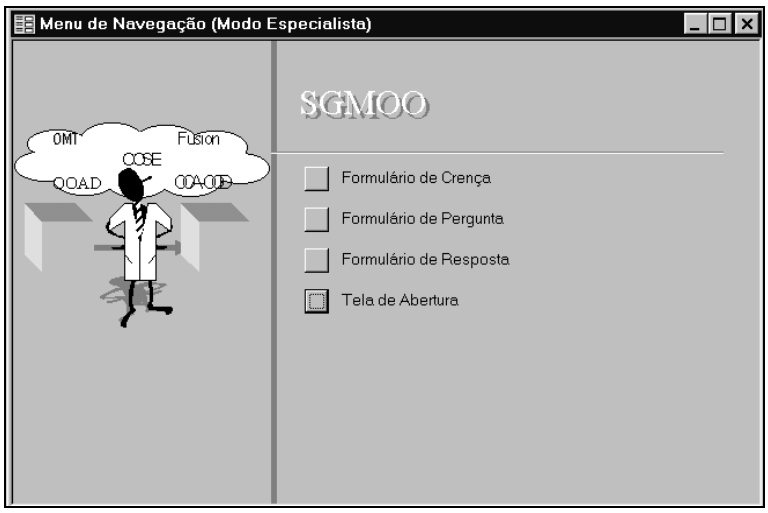



Figura 6.6 - Formulário de menu de navegação para o especialista

A figura 6.7 apresenta o formulário para entrada das perguntas - ou questões - a serem utilizadas no questionário. Cada pergunta pertence a uma fase, tem um tipo (GR/SN) e um campo de observação, que funciona como uma explicação da pergunta para auxiliar o engenheiro de *software* a responder o questionário.

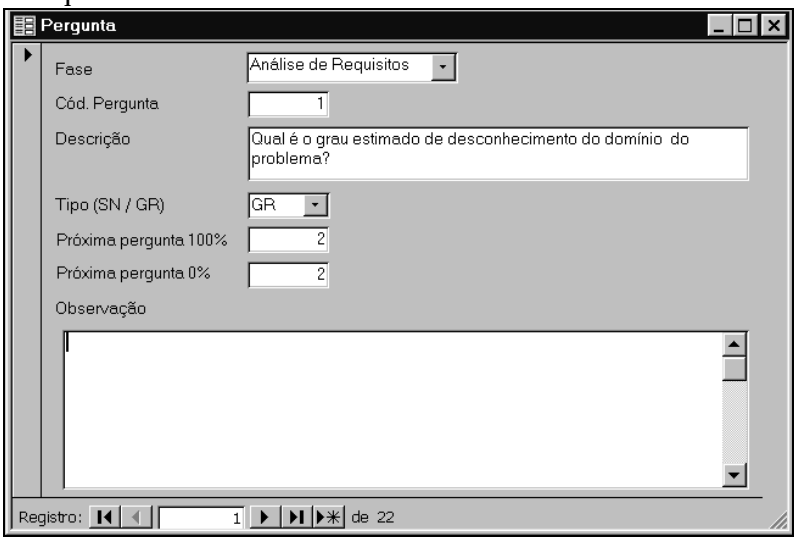



Figura 6.7 - Formulário para entrada das perguntas

A figura 6.8 apresenta o formulário para entrada das crenças definidas para cada resposta (máximo/mínimo). Uma crença é definida para um nível e para um método atribuindo seu valor.

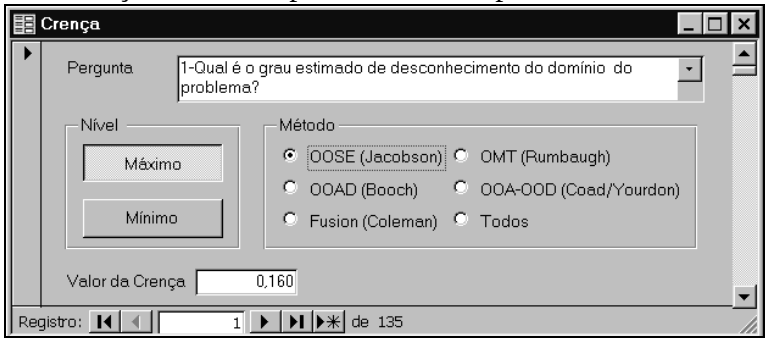



Figura 6.8 - Formulário para entrada das crenças

As próximas quatro figuras são apresentadas ao engenheiro de *software*.

A figura 6.9 ilustra o formulário para apresentação do questionário quando o tipo de pergunta é para informar o grau.

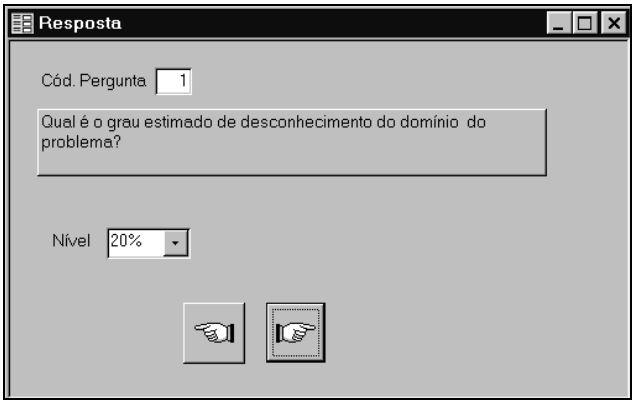



Figura 6.9 - Formulário I de apresentação do questionário

A figura 6.10 apresenta um segundo formulário para apresentação do questionário quando o tipo de pergunta é SN (sim/não).

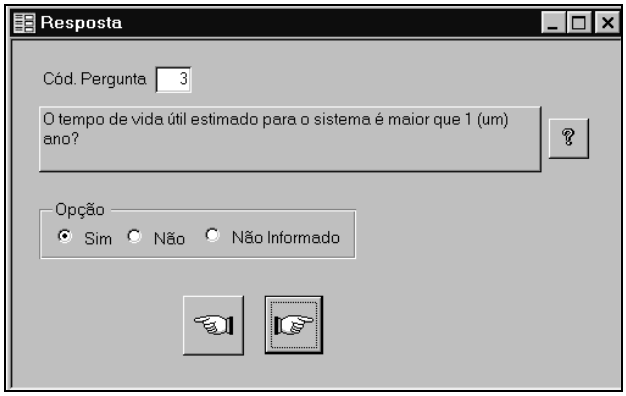



Figura 6.10 - Formulário II de apresentação do questionário

Para ambos os formulários (figura 6.9 e 6.10) estão definidos dois aspectos: i) um botão de *help*, ilustrado na figura 6.10, que será acionado se estiver definido na tabela de Perguntas a necessidade de explicação e ii) o usuário poderá prosseguir com a próxima pergunta ou voltar para corrigir eventual resposta informada incorretamente, através dos botões ilustrados na parte inferior de ambas figuras.

A figura 6.11 apresenta a explanação da pergunta quando definida no campo 'observação' da tabela Pergunta.

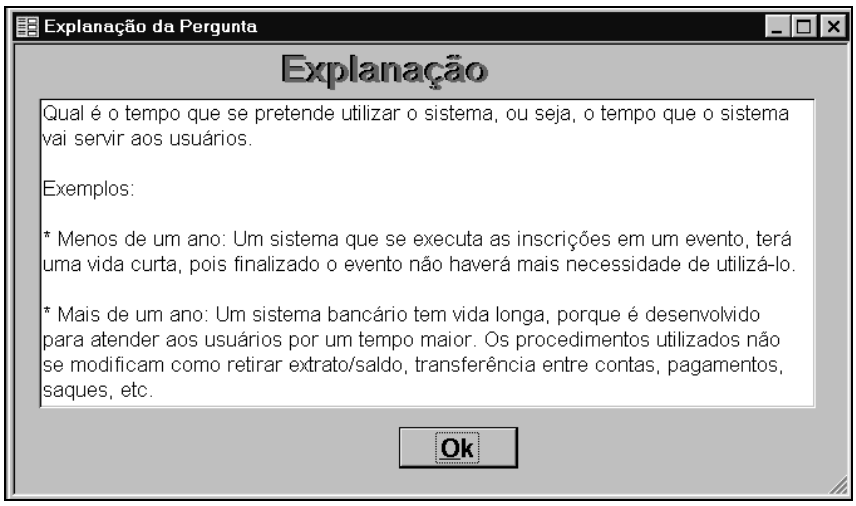



Figura 6.11 - Formulário de apresentação da explanação da pergunta

A figura 6.12 ilustra o formulário apresentado ao engenheiro de *software* assim que o questionário todo é respondido. Posteriormente, são combinadas as funções de crença com base nas respostas obtidas.

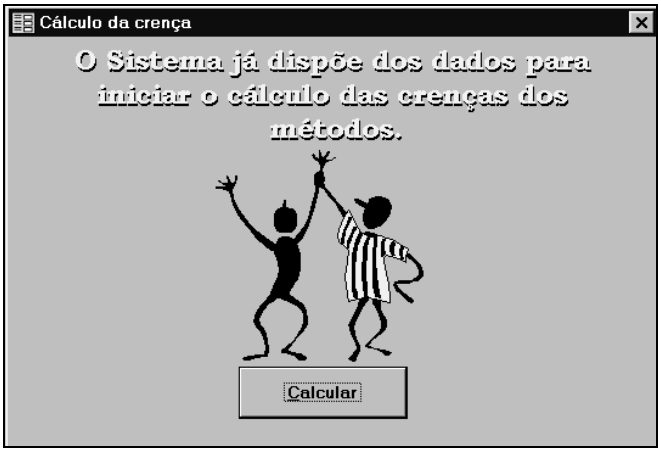



Figura 6.12 - Formulário para início do algoritmo

## *6.5 Descrição dos casos reais*

Serão aplicados a simulação de quatro casos reais. Estes casos foram identificados com a ajuda de engenheiros de *software* que têm grande experiência no desenvolvimento de sistemas, porque a aplicação de casos reais engloba entrevistas com diversos usuários; além do mais o objetivo principal deste trabalho não é fazer uma vasta validação sobre o protótipo. Os casos são: *i)* sistema de contabilidade, *ii)* sistema de controle de biblioteca, *iii)* sistema de controle de passagens aéreas e *iv)* sistema de controle industrial. A seguir são apresentadas uma breve descrição das funcionalidades dos sistemas utilizados para simulação e suas características.

- Sistema de contabilidade - sistema contábil de uma empresa que registra as notas fiscais de produtos vendidos e/ou adquiridos, atualização de patrimônio ativo e passivo, elaboração de balanços e balancetes e controle de caixa. A interação entre o usuário e o sistema é média, assim

como a pesquisa realizada nos dados armazenados, a entrada de dados deste tipo de sistema é alta, com uso considerável de cálculos, os dados têm importância média.

- Sistema de controle de biblioteca - sistema para controlar cadastro de funcionários, usuários e obras, empréstimos, reservas, devoluções, pagamento de multas, envio de obras para restauração e manter relatórios diários para melhor controle. A interação entre o usuário e o sistema é baixa, assim como a entrada de dados no sistema, possui pouquíssimos cálculos, os dados são muito importantes e a pesquisa realizada nos dados armazenados é bastante alta.
- Sistema de controle de passagens aéreas - sistema para controlar reservas, cancelamentos e venda de passagens aéreas, controle de embarque, pagamentos efetuados e a efetuar. A interação entre o usuário e o sistema é média, assim como a entrada de dados no sistema, possui pouquíssimos cálculos, os dados são extremamente importantes e a pesquisa realizada nos dados armazenados é relevante.
- Sistema de controle industrial - sistema que controla um processo químico, em que várias substâncias são combinadas e é feito um controle de temperatura e de outras variáveis como: PH, concentração de $O_2$, etc. Quase não há interação entre o usuário e o sistema e nem entrada de dados no sistema, possui bastante cálculos, os dados têm importância mínima, assim como a pesquisa realizada nos dados armazenados. O sistema não será do tipo cliente-servidor.

Características gerais sobre os sistemas são: i) todos os sistemas com exceção do sistema industrial terá padrão *Windows* e poderá ter mudanças na sua definição durante sua vida útil; ii) somente o sistema industrial fará controle de máquinas e equipamentos e tempo real.

A tabela 6.6 apresenta as respostas obtidas sobre as questões para cada um dos sistemas descritos acima.

*Tabela 6.6*
*Respostas das questões para cada sistema*

| | Contabilidade | Biblioteca | Passagens aéreas | Industrial |
|---|---|---|---|---|
| Questão 1 | 30 % | 10 % | 70 % | 50 % |
| Questão 2 | 20 % | 0 % | 50 % | 80 % |
| Questão 3 | Sim | Sim | Sim | Sim |
| Questão 4 | 90 % | 50 % | 10 % | 0 % |
| Questão 5 | Sim | Sim | Não | Não |
| Questão 6 | 60 % | 40 % | 50 % | 10 % |
| Questão 7 | Sim | Sim | Sim | Não |
| Questão 8 | 80 % | 30 % | 50 % | 10 % |
| Questão 9 | 60 % | 10 % | 10 % | 70 % |
| Questão 10 | 50 % | 90 % | 100 % | 30 % |
| Questão 11 | Sim | Sim | Sim | Não |
| Questão 12 | Não | Não | Não | Sim |
| Questão 13 | Não | Não | Não | Sim |
| Questão 14 | Não | Não | Sim | Não |
| Questão 15 | Não | Não | Sim | Não |
| Questão 16 | - | - | Não | - |
| Questão 17 | Não | Sim | - | Não |
| Questão 18 | - | Sim | - | - |
| Questão 19 | Sim | - | - | Não |
| Questão 20 | 50 % | 90 % | 70 % | 10 % |
| Questão 21 | Sim | Sim | Sim | Não |
| Questão 22 | - | - | - | Sim |

## *6.6 Resultados obtidos pela análise do SGMOO*

A figura 6.13 apresenta o resultado obtido pela análise do SGMOO para o sistema de contabilidade.

**Resultado**

| | An. Req. | Análise | Projeto | Geral |
|---|---|---|---|---|
| Todos | 0,00 | 0,00 | 0,00 | 0,00 |
| OOSE - Jacobson et. al. | 31,30 | 28,60 | 13,10 | 23,08 |
| OMT - Rumbaugh et. al. | 15,90 | 28,60 | 5,30 | 4,73 |
| OOAD - Booch | 19,30 | 28,60 | 64,70 | 70,41 |
| OOA-OOD - Coad & Yourdon | 16,70 | 14,30 | 3,70 | 1,78 |
| Fusion - Coleman et. al. | 16,70 | 0,00 | 13,10 | 0,00 |

Figura 6.13 – Resultado apresentado pelo SGMOO para o sistema de contabilidade

Os seguintes aspectos podem ser identificados:

- Para modelar a análise, o SGMOO indicou o uso de qualquer um dos seguintes métodos: OOSE, OMT ou OOAD, não indicando o método Fusion. O empate deve-se ao fato de que o método deve modelar subsistemas, devido ao alto reuso, a interface e possibilitar uma boa documentação.
- Interessante observar que para a fase de análise, o método Fusion apresenta modelagem mais deficiente do que os métodos para quase todos os aspectos[24], por isso não houve pontuação.

A figura 6.14 apresenta o resultado obtido pela análise do SGMOO para o sistema de biblioteca.

**Resultado**

| | An. Req. | Análise | Projeto | Geral |
|---|---|---|---|---|
| Todos | 0,00 | 0,00 | 0,00 | 0,00 |
| OOSE - Jacobson et. al. | 29,00 | 20,70 | 11,40 | 10,54 |
| OMT - Rumbaugh et. al. | 16,00 | 24,10 | 2,40 | 1,40 |
| OOAD - Booch | 20,10 | 34,50 | 80,50 | 86,51 |
| OOA-OOD - Coad & Yourdon | 17,90 | 10,30 | 0,80 | 0,16 |
| Fusion - Coleman et. al. | 17,00 | 10,30 | 4,90 | 1,40 |

Figura 6.14 – Resultado apresentado pelo SGMOO para o sistema de biblioteca

[24] Referências são as tabelas 5.7 a 5.14.

Os seguintes aspectos podem ser identificados:

- O método OOAD foi o mais indicado para ser utilizado tanto na análise quanto no projeto. O sistema não requer cálculos, alta entrada de dados e a interação é baixa. Necessita ser bem documentado e ter modelagem apropriada para reuso e mudança de requisitos. Por isso, o método OOAD é o mais indicado para análise.
- Se o engenheiro não optar pelo método OOAD, pode-se decidir entre os métodos OOSE e OMT que tecnicamente estão empatados.

A figura 6.15 apresenta o resultado obtido pela análise do SGMOO para o sistema de passagens aéreas.

**Resultado**

| | An. Req. | Análise | Projeto | Geral |
|---|---|---|---|---|
| Todos | 0,00 | 0,00 | 0,00 | 0,00 |
| OOSE - Jacobson et. al. | 36,80 | 17,10 | 10,00 | 15,18 |
| OMT - Rumbaugh et. al. | 15,90 | 24,80 | 4,60 | 4,34 |
| OOAD - Booch | 17,70 | 22,90 | 73,20 | 71,33 |
| OOA-OOD - Coad & Yourdon | 13,60 | 12,40 | 3,20 | 1,20 |
| Fusion - Coleman et. al. | 16,10 | 22,90 | 8,90 | 7,95 |

Figura 6.15 –Resultado apresentado pelo SGMOO para o sistema de passagens aéreas

Os seguintes aspectos podem ser identificados:

- Para o desenvolvimento desse sistema pode-se escolher entre os métodos OMT, OOAD e Fusion para a fase de análise, que praticamente estão empatados na graduação. São métodos que apresentam boa documentação para essa fase. O sistema não será desenvolvido tendo o reuso como meta e nem apresenta muitos cálculos, tem sua principal preocupação nos dados e na interação.
- Importante observar que para a modelagem do sistema na análise, praticamente pode-se utilizar qualquer um dos métodos. Isso deve-se ao fato de cada método modelar melhor determinados aspectos. Entretanto, o SGMOO indica o método que tenha melhor modelagem em todos os aspectos.

A figura 6.16 apresenta o resultado obtido pela análise do SGMOO para o sistema industrial. Os seguintes aspectos podem ser identificados:

- Para modelagem desse sistema na fase de análise não são indicados os métodos OOSE, OOAD e Fusion. O principal nesse sistema é a modelagem dos cálculos e a importância dos dados, onde é necessário ter uma boa descrição das classes. Deste modo, os métodos mais indicados são OMT e Fusion.

- Interessante observar que o método OOA-OOD foi bem graduado para a fase de projeto, pelo fato de que poucas características para o projeto são informadas e uma delas - estabelecer prioridades - é modelada unicamente por esse método.
- Apesar do método OOA-OOD não ter obtido pontuação como OOAD para a fase de projeto, o engenheiro de *software* pode pensar em utilizar o método OOA-OOD na análise e no projeto, principalmente se a equipe de desenvolvimento for inexperiente com orientação a objetos.

**Resultado**

| | An. Req. | Análise | Projeto | Geral |
|---|---|---|---|---|
| Todos | 0,00 | 25,20 | 0,00 | 0,00 |
| OOSE - Jacobson et. al. | 46,80 | 2,60 | 5,70 | 14,07 |
| OMT - Rumbaugh et. al. | 14,90 | 34,60 | 10,00 | 16,92 |
| OOAD - Booch | 13,00 | 2,80 | 52,50 | 36,31 |
| OOA-OOD - Coad & Yourdon | 10,00 | 31,50 | 24,90 | 26,81 |
| Fusion - Coleman et. al. | 15,40 | 3,30 | 7,00 | 5,89 |

Figura 6.16 –Resultado apresentado pelo SGMOO para o sistema industrial

Analisando os resultados obtidos observa-se os seguintes pontos genéricos:

- O método OOSE foi o mais indicado para a fase de análise de requisitos em todos os quatro casos. Isto deve-se ao fato desse método oferecer uma modelagem abrangente para essa fase e haver necessidade de iniciar o desenvolvimento por essa fase;
- O método OOAD foi o mais indicado para a fase de projeto também para os quatro casos. Este método possui ampla modelagem nessa fase;
- O método OOAD também foi o mais indicado para ser utilizado durante todo o desenvolvimento de todos os casos. Isto deve-se ao fato de que este método possui alta graduação na fase de projeto.

Um ponto já identificado que pode ser melhorado:

- Para o sistema de contabilidade, o SGMOO não deveria graduar o método OOAD tão distante dos demais para a fase de projeto, pelo fato do sistema não apresentar características de projeto e dinâmica muito relevantes - tempo real, sistema distribuído, desenvolver protocolo -. Deste modo, outros métodos poderiam ser utilizados.

## *6.7 Considerações finais*

O SGMOO pode auxiliar bastante o engenheiro de *software* a tomar a decisão de escolher qual método orientado a objetos é mais indicado para o desenvolvimento de determinado sistema.

Os resultados obtidos através das simulações de casos reais são aceitáveis e, em muitos aspectos, refletem o estudo realizado. Entretanto, os resultados fornecidos deveriam ser aplicados em um desenvolvimento real, permitindo ao especialista em métodos uma análise ao final do desenvolvimento para melhorar o SGMOO, o mapeamento das crenças ou os aspectos que devem ser modelados para cada característica de um sistema.

O próximo capítulo apresenta as conclusões finais deste trabalho.

# Capítulo 7

# *Conclusão*

O capítulo final deste trabalho é divido em duas seções. A primeira descreve uma visão geral dos objetivos, desenvolvimento e contribuições ao conhecimento. A segunda destaca os trabalhos futuros que podem ser desenvolvidos como extensão ou complementação deste.

## *7.1 Visão geral*

Esta seção faz uma revisão resumida do trabalho como um todo e apresenta as contribuições ao conhecimento.

### *7.1.1 Revisão*

Como já mencionado, o objetivo principal dessa pesquisa foi elaborar uma proposta para auxiliar os desenvolvedores de sistema a escolher o método orientado a objetos mais indicado para o desenvolvimento de um determinado domínio de aplicação. Para alcançá-lo, foi necessário o estudo sobre os conceitos de modelagem e orientação a objetos, bem como dos métodos orientados a objetos. Paralelamente ao estudo e descrição dos métodos, foi desenvolvida a modelagem de partes de um sistema de biblioteca segundo as diretrizes de cada um dos métodos. O objetivo da modelagem não foi ser completa, mas ajudar a identificar pontos positivos e negativos dos métodos.

Em seqüência, os métodos foram comparados com base nos elementos notacionais e explicações conceituais oferecidas sobre alguns aspectos de um desenvolvimento. Estes aspectos foram identificados com o auxílio de engenheiros de *software* experientes em desenvolvimento e no trabalho de Nascimento (Nascimento, 1990).

Com o auxílio desses mesmos engenheiros foi selecionado um conjunto de questões que possibilita a identificação das características de um sistema. Cada característica informa que o método deve possuir modelagem para certos aspectos. Desta forma, cada característica foi relacionada a um ou mais dos aspectos referenciados anteriormente. Estas questões são respondidas pelo engenheiro de *software*, que tem como base o sistema que ele pretende desenvolver.

Ainda visando alcançar o objetivo traçado, foi estudado a técnica de Inteligência Artificial denominada teoria de função de crença. Com esta teoria pode-se fazer o mapeamento das funções de crença aos métodos para cada característica do sistema. Posteriormente, a combinação destas funções indica o método mais apropriado para cada fase do desenvolvimento do sistema em questão.

### *7.1.2 Métodos orientados a objetos*

Pelo estudo realizado pode-se concluir que o método OOSE é o único a oferecer ampla modelagem para análise de requisitos, os demais iniciam pela fase de análise. Este método realiza tarefas na fase de análise de requisitos que os outros métodos executam na análise, realizando durante a análise algumas tarefas do projeto, tornando assim seu projeto bem detalhado. Não é um método prescritivo mas é bastante completo e complexo, por isso é indicado para equipes de desenvolvimento com mais de dois componentes e que possuam treinamento em todas as fases de um desenvolvimento. É indicado para sistemas que necessitem de especialistas do domínio porque oferece modelagem para facilitar esta tarefa e para sistemas técnicos e administrativos porque nestes tipos de sistemas pode se utilizar o reuso de componentes com maior facilidade, por não se distinguirem tanto um dos outros mesmo em organizações diferentes.

O método OMT modela o sistema através de três diferentes dimensões: estática, dinâmica e funcional, os outros se preocupam somente com as duas primeiras. A dimensão funcional é bastante criticada entre os metodologistas que trabalham com orientação a objetos, mas que pode ser muito útil para modelar sistemas com muita transformação de dados. O método não apresenta praticamente nenhuma modelagem para a fase de projeto. É um método bastante prescritivo, explicando o que se deve fazer durante o desenvolvimento. É indicado para equipes que tenham experiência em análise. Caso a experiência seja em projeto pode ocorrer dificuldades tendo em vista que o método direciona o desenvolvimento para a abstração do problema. Sistemas com muita modelagem de dados e que possuam muitos cálculos podem utilizar-se desse método.

O método OOAD sofreu uma extensão da primeira versão. Primeiramente foram definidos modelos para a fase de projeto, posteriormente incorporou-se modelos para a fase de análise. O projeto aborda aspectos que não são modelados pelos demais métodos, como visibilidade e concorrência (com exceção do método Fusion). Utiliza-se técnicas e diagramas definidos por outros metodologistas como CRC e *use cases*. Não é um método prescritivo e é bastante complexo, sendo portanto indicado para equipes experientes. Indicado para sistemas de tempo real, ou que possuam muita concorrência ou que tenha a arquitetura distribuída.

O método OOA-OOD foi um dos primeiros métodos orientados a objetos a serem desenvolvidos servindo de base para os que vieram posteriormente. O método sugere o refinamento dos modelos de análise (multicamadas) durante o projeto (modelo multicomponentes). É um método prescritivo, que se preocupa com o reuso de sistemas e é bastante simples. É indicado para desenvolvedores inexperientes, também para grandes equipes (análise depois projeto) ou para pequenas (intercalação entre análise e projeto). Deve ser utilizado para desenvolvimento de sistemas simples devido a pouca notação oferecida.

O método Fusion caracteriza-se por ser um método integrador, reunindo elementos dos métodos OMT, OOAD, formais e da técnica CRC, complementando com modelos próprios. É um método quase completo, porque falta a cobertura para alguns aspectos. Não é um método prescritivo e é de ampla cobertura, por isso é indicado para equipe que já possuem experiência em

desenvolvimento orientado a objetos. Pode ser aplicado em sistemas concorrentes por oferecer notação para a modelagem.

### *7.1.3 Conceitos e notação*

Alguns conceitos apresentados pelos métodos, apesar de se referirem ao mesmo conteúdo são nomeados diferentemente, o que pode gerar uma certa confusão quando se estuda mais de um método. Uma solução que poderia ser adotada, seria uma padronização da nomeação dos conceitos, assim como aconteceu com a notação empregada, o que facilitaria o estudo de vários métodos.

Problema semelhante ocorre com a notação empregada pelos autores dos métodos. Alguns conceitos são graficamente representados de diferentes formas. Exemplo: um triângulo ligando duas classes, indica herança para o método OMT, o que já significa agregação para o método OOA-OOD. Para o método Fusion, a agregação é representada quando se coloca uma classe dentro da outra e para o método OOAD, a agregação é uma linha com um círculo preenchido no lado da classe que agrega; onde este mesmo símbolo é utilizado pelo método OMT para definir cardinalidade de 0 ou mais em um relacionamento. Este tipo de problema exige atenção redobrada do desenvolvedor.

Tentando solucionar esse problema, Grady Booch, Ivar Jacobson e James Rumbaugh, com ajuda de um conjunto de metodologistas e organizações, definiram uma linguagem padrão de notação chamada UML – *Unified Modeling Language*, que foi adotada como padrão pela OMG. Maiores detalhes podem ser obtidos no apêndice D.

A UML, como uma notação padrão, tem o propósito de unificar as notações existentes para a modelagem orientada a objetos, facilitando a comunicação entre analistas e projetistas que utilizem métodos diferentes, e consequentemente notações diferentes, em seus desenvolvimentos. Além de facilitar a compreensão de todos os modelos, pois os modelos da estrutura estática de um sistema, por exemplo, possuirão a mesma sintática e semântica.

É importante ressaltar que a UML não é um método porque não oferece diretrizes de como conduzir o desenvolvimento. Ela é uma notação padrão que pode substituir as notações definidas pelos autores dos métodos. Por exemplo, pode-se utilizar o método OMT com a notação UML em um determinado desenvolvimento.

### *7.1.4 Função de crença*

A busca pela interação entre duas áreas do conhecimento é muito importante e necessária, porque uma pode complementar a outra, como foi aplicado neste caso, onde a área de Inteligência Artificial auxiliou a área de Engenharia de *Software*.

A teoria de função de crença com o algoritmo de Dempster e Shafer, soma ortogonal, permitiu a identificação do método mais indicado através da combinação entre as várias evidências, identificadas pelas respostas fornecidas diante das perguntas apresentadas. A partir das várias

evidências que podem indicar métodos diferentes, obtém-se uma evidência que expressa o conhecimento atual.

A função de perda poderia ser também incorporada ao trabalho, possibilitando uma completa decisão sobre qual método ser adotado, pois assim poderia sugerir o método e calcular qual seria a perda se fosse utilizado um método diferente. Neste sentido seria adotado o que oferecesse menor perda. Entretanto, este trabalho se propôs a sugerir um método onde a decisão final ficaria a cargo do engenheiro de *software*, usuário do protótipo. Isto se deve ao fato do SGMOO não levar em conta aspectos gerenciais que são bastante relevantes em um desenvolvimento.

### *7.1.5 SGMOO*

O SGMOO alcança seu objetivo inicial de implementar a proposta definida neste trabalho. Ele apresenta interfaces separadas para cada tipo de usuário, possibilitando a indicação do método mais apropriado no desenvolvimento de um sistema específico e o aperfeiçoamento das crenças definidas, inclusões/alterações de questões, métodos e fases.

Para aperfeiçoar o SGMOO deve haver uma realimentação do protótipo. Isso se torna possível através do acompanhamento do especialista em um desenvolvimento com os métodos sugeridos pelo protótipo para o sistema em questão. Isto permitirá verificar a eficiência para futuras modificações na base de dados ou no aperfeiçoamento do modelo proposto, como ilustra a figura 7.1.

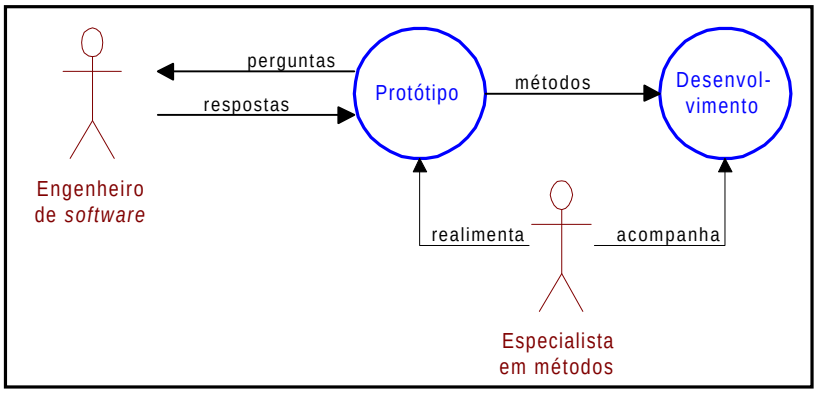



Figura 7.1 - Possível realimentação do protótipo

## *7.2 Contribuições ao conhecimento*

O presente trabalho contribui com a área de Engenharia de *Software*, principalmente, auxiliando no desenvolvimento de aplicações. Ele oferece um estudo aprofundado de cinco métodos orientados a objetos e a comparação entre eles, que permite ao engenheiro de *software* uma visão geral e detalhada desses métodos.

Quando o engenheiro de *software* tem que decidir sobre qual método utilizar no desenvolvimento de um sistema específico, ele possui duas saídas: i) escolher um método

aleatoriamente, podendo ocasionar uma escolha errada ou ii) estudar todos os possíveis métodos para escolher o mais adequado. Neste sentido o SGMOO busca auxiliá-lo nessa tarefa dispendiosa de tempo.

Nos últimos anos muitas pesquisas vêm sendo realizadas a cerca dos métodos orientados a objetos e suas comparações (Brinkkemper, 1995), (Fichman, 1992), (Marques), (Reinoso, 1994) e (Silveira, 1995). O presente trabalho também realizou uma comparação. Entretanto, os primeiros trabalhos comparam os métodos a nível de modelos e diagramas. Neste trabalho a comparação é realizada a nível dos elementos notacionais desses diagramas e base conceitual oferecida pelos métodos.

Diferentemente destes primeiros trabalhos (Nascimento, 1990) propôs o agrupamento de duas grandes áreas do conhecimento (Engenharia de *Software* e Inteligência Artificial). Foram comparados métodos estruturados e utilizado sistemas especialistas. Da mesma forma, este trabalho agrupa essas duas áreas, comparando métodos orientados a objetos e utilizando teoria de função de crença. A principal razão pela escolha da teoria de função de crença deve-se ao fato de que está possui um suporte teórico para tratamento de incerteza.

## *7.3 Propostas para trabalhos futuros*

Tendo em vista o tempo limitado para desenvolver e concluir a dissertação, alguns aspectos do trabalho podem ser acrescentados ou aperfeiçoados, como:

i) acrescentar novos métodos orientados a objetos, através de um estudo aprofundado e de sua comparação com os outros, para permitir ao engenheiro de *software* uma visão mais abrangente dos métodos;

ii) acrescentar novas perguntas devido ao surgimento de novos domínios de aplicação, onde necessita englobar suas características;

iii) acrescentar ao mapeamento de crenças os aspectos da parte gerencial de um desenvolvimento como quantidade de pessoal na equipe, conhecimento que a equipe possui sobre os métodos a serem utilizados e mesmo sobre o domínio do problema, etc.;

iv) devido a inclusão de novos métodos e/ou perguntas é necessário, também, a inclusão de funções de crença referentes ao mapeamento das respostas aos métodos;

v) inclusão de funções de perda que viabilizam identificar qual é a perda que se tem quando o protótipo sugere um método, mas o usuário opta por outro; e, assim, decidir pelo método que ofereça menor perda;

vi) detalhar o critério adotado de ponderação entre os elementos notacionais, ou seja, aumentar a faixa dos pesos, ao invés de somente 1 e 2. Isto possibilita uma melhor ponderação entre os elementos;

vii) sistematizar integração com o SMM, para que este considere a escolha de métodos orientados a objetos. O presente trabalho será, então, adicionado a subfunção

*Building Strategy* em *Technical Environment Selection*, onde os métodos são escolhidos. Maiores detalhes (Nascimento, 1992).

viii) realizar uma realimentação do SGMOO para aperfeiçoar a proposta.

Outras idéias ainda podem ser extraídas deste trabalho, como:

i) a proposta de um novo método, porque com um estudo aprofundado pode-se ter o conhecimento suficiente para saber escolher as melhores partes de um método que se aplicará a determinado domínio de aplicação;

ii) desenvolvimento de um produto final, tendo como base a arquitetura definida para o SGMOO, apresentada neste trabalho. Isso auxiliaria o engenheiro de *software* na tarefa de escolha de um método, permitindo ganho de tempo e um produto final com mais qualidade.

## *7.4 Considerações finais*

Espera-se que o estudo e o protótipo tragam os seguintes resultados concretos: *a*) suporte os gerentes e desenvolvedores, através da utilização de processos mais confiáveis que reflitam ganho de qualidade no processo de desenvolvimento e consequentemente dos produtos gerados; *b*) conscientização de que a escolha de um método ou um conjunto deles não é a solução para todos os tipos de sistemas, uma vez que cada sistema possui suas próprias características, exigindo assim, uma escolha mais adequada de métodos garantindo a qualidade do produto final.

É interessante notar que embora possam ser desenvolvidos métodos abrangentes, os primeiros métodos (primeira geração) continuarão a ser utilizados, porque modelam bem determinados domínios de aplicação. A escolha do método não será somente devido ao que o método oferece, mas também será levado em consideração outros aspectos, como o tamanho e conhecimento da equipe de desenvolvimento. Neste sentido, acredita-se não ter muita necessidade a padronização de um método, porque mesmo com algumas falhas os primeiros métodos serão utilizados. Entretanto, uma padronização poderia ser feita em relação aos conceitos utilizados, pois é enorme a quantidade de nomes atribuídos ao mesmo conceito pelos métodos, assim como foi feito a padronização para a notação (UML), maiores detalhes apêndice D.

Finalmente, pode-se tomar como base os argumentos de (Fowler, 1997):

> *Eu não acredito que se possa ter um único processo para o desenvolvimento de software. Vários fatores associados com o desenvolvimento de software induz em diferentes tipos de processos. Estes fatores incluem o tipo de software que está sendo desenvolvido (tempo-real, sistema de informação, produto para desktop), a escala (único desenvolvedor, equipe de desenvolvimento pequena, equipe com 100 ou mais membros) e assim por diante*.

# Bibliografia


**ANDERSSON**, M. *Uses Cases in Object-Oriented Analysis*. Sweden, Lund University. 1995. <http://www.efd.lth.se/~d87man/EXJOBB/Title_Abstract_Preface.html>.

**BAKKER**, G. *OMT Object Model: Notation, Concepts and Constructs*. 1996. <http://www.comp.mq.edu.au/courses/comp331/OMT/notation.html>.

**BIENVENU**, M. Systems Design in ObjecTime. *Byte*, v.20, n.12, p.189-190, dez.1995.

**BOOCH**, G. *Object Oriented Design with Applications*. California: *The Benjamin/Cummings Publishing Company, Inc*., 1991.

_____ *Object Oriented Analysis and Design with Applications*. 2.ed. Canadá: Addison Wesley, 1994.

_____ *Object Solutions: Managing the Object Oriented Project*. Menlo Park: Addison-Wesley, 1996.

_____ Quality Software and the UML. *Object Magazine*, Canadá, v.7, n.1, p.78-80, mar.1997.

**BORGES**, G. do E. S., **NASCIMENTO**, M. E. M. *Uma avaliação de métodos orientados a objetos e modelos de processo*. Brasília: UnB, 1997.

_____ *Sistema baseado em conhecimento para selecionar métodos de desenvolvimento de software orientados a objeto*. In: 50ª. Reunião Anual da SBPC - Sociedade Brasileira para o Progresso da Ciência, 1998a. Natal: UFRN, 1998. p.1054-1055.

_____ *SGMOO: Sistema Gestor de Métodos Orientados a Objetos baseado em conhecimento*. In: III Workshop de Teses em Engenharia de *Software*, 1998. Maringá: 1998b.

**BRINKKEMPER**, S., **HONG**, S., **BULTHUIS**, A., **GOOR**, G. v.d. *Object Oriented Analysis and Design Methods – a Comparative Review*. Enschede: University of Twente. 1995. <http://wwwis.cs.utwente.nl:8080/dmrg/OODOC/ oodoc/oo.html>.

**Business on the Net.** *BYTE*, USA, v.21, n.8, ago.1996.

**CAMARÃO**, P. C. B. *Glossário de Informática*. Rio de Janeiro: LTC - Livros Técnicos e Científicos, 1993.

**CAMPOS**, F. B. *Sistematizando o Processo de Desenvolvimento de Software - Uma abordagem Orientada ao Reuso*. Brasília: UnB, 1997.

**CAPRETZ**, L. F., **LEE**, P. A. Object Oriented Design: guidelines and techniques, *Information and Software Technology*, v.35, n.4, abr.1993.

**CARVALHO**, R. R. A. *Função de Crença como Ferramenta para Selecionar Diagnósticos em Raciocínio Baseado em Casos*. Brasília: UnB, 1996.

**CHOOSING REUSABLE COMPONENTS**. *Software Development*, USA, v.3, n.4, abr.1995.

**COAD**, P., **YOURDON**, E. *Análise Baseada em Objetos*. Tradução por CT Informática. Rio de Janeiro: Editora Campus, 1992.

_____ *Projeto Baseado em Objetos*. Tradução por Vandenberg Dantas de Souza. Rio de Janeiro: Editora Campus, 1993.

**COLEMAN**, D. et alii. *Desenvolvimento Orientado a objetos: o método Fusion*. Tradução por Geraldo Costa Filho. Rio de Janeiro: Editora Campus, 1996.

**CROSS - PLATAFORM DEVELOPMENT**. *Dr. Dobb's Journal*. Canadá: v.20, n.228, issue 3, mar.1995.

**DATE**, C. J. *Introdução a Sistema de Banco de Dados*. Tradução por Hélio Auto Gouveia. Rio de Janeiro: Editora Campus, 1989.

**DICK**, K. *Trajectory Analysis: ODBMS Vendors*. USA: <http://www.odbmsfacts.com./trajectory/trajintr.html>.

**ENCRYPTION, ERROR CORRECTION, & COMPRESSION**. *Dr. Dobb's Journal*. Canadá: v.22, issue 1, jan.1997.

**FICHMAN**, R. G. **KEMERER**, C. F. Object-Oriented and Conventional Analysis and Design Methodologies - Comparison and Critique. *IEEE Computer*, Massachusetts, v.25, n.10, out.1992.

**FOWLER**, M., **SCOTT**, K. *UML Distilled - Applying the Standard Object Modeling Language*. USA: Addison Wesley Longman, Inc., 1997.

**HAINES**, J. *Persistent Object Stores Applications*. nov.1995. <http://demos.anu.edu.au/jason/thesis/thesis/www>

**HODGSON**, J. *Software Engineering Process Models*. 1995. <http://www.sju.edu/~jhodson/se/models.html>

**HUMPHREY**, W. S. *Managing Software Process*. Canadá: Addison-Wesley, 1989.

**JACOBSON**, I. *Object-Oriented Software Engineering: A use case driven approach*. USA: ACM Press, 1992.

**KAN**, S. H. *Metrics and Models in Software Quality Engineering*. Massachusetts: Addison-Wesley, 1995.

**KEYES**, J. *Software Engineering Productivity Handbook*. USA: McGraw Hill, 1993.

**KHOSHAFIAN**, S. *Banco de Dados Orientado a Objeto*. Tradução por Tryte Informática. Rio de Janeiro: Infobook, 1994.

**KIM**, W. *Introduction to object-oriented databases. Cambridge*: The MIT Press, 1990.

**KITCHENHAM**, B. A. Evaluating Software Engineering Methods and Tool - Part 1: The Evaluation Context and Evaluation Methods. *ACM SIGSOFT - Software Engineering Notes*, Inglaterra, v.21, n.1, p.11-15, jan.1996a.

_____ Evaluating Software Engineering Methods and Tool - Part 2: Selecting an Appropriate Evaluation Method - Technical Criteria. *ACM SIGSOFT - Software Engineering Notes*, Inglaterra, v.21, n.2, p.11-15, mar.1996b.

_____ Evaluating Software Engineering Methods and Tool - Part 3: Selecting an Appropriate Evaluation Method - Pratical Issues. *ACM SIGSOFT - Software Engineering Notes*, Inglaterra, v.21, n.4, p.9-12, jul.1996c.

**MARQUES**, A. F. D. M. R. *Descrição de Metodologias de Análise orientadas a Objetos*. Braga: Universidade do Minho. <http://alfa.di.uminho.pt/~mesafm/ analiOO.html>

**MARTIN**, J. *Princípio da Análise e Projeto baseado em Objetos*. Tradução por Cristina Bazán. Rio de Janeiro: Editora Campus, 1994.

**McCLURE**, S. *Object Database vs. Object-Relational Databases*. International Data Corporation: ago.1997. <http://www.cai.com/products/jasmine/analyst/idc/ 14821E.htm>

**McFARLAND**, G., **RUDMIK**, A. *Object-Oriented Database Management Systems - A Critical Review*. Technology Assessment: set.1993. <http://www.dacs.com/techs/OODBMS/ oodbms.toc.html>

**McGEE**, M. D. e **BOYCE**, J. *Microsoft Access for Windows - Step by Step*. Washington: Microsoft Press, 1993.

**MICROKERNELS AND OPERATING SYSTEMS**. *Dr. Dobb's Journal*. Canadá: v.19, n.214, issue 5, mai.1994.

**MONTGOMERY**, S. L. *Information Engineering: analysis, design and implementation*. Cambridge: Academic Press, 1994.

**NASCIMENTO**, A. R. C. *An Expert System to Advise on Information System Development Approaches*. Manchester: UMIST, 1990.

**NASCIMENTO**, M. E. M. *SMM (Software Management Model): A Multidimensional and Integrative Software Development Management Model*. Manchester: UMIST, 1992.

**NASCIMENTO**, M. E. M. e **NASCIMENTO**, A. R. C. *Um modelo orientado a multi-métodos, multi-técnicas e multi-ferramentas para o desenvolvimento de software*. In: Textos em Ciência da Computação, 1993. p.11-20.

**ODMG**. <http://www.odmg.org/>

**OLIVEIRA**, W. V. *Fundamentos de Metodologia Científica*. Brasília: UnB, 1997.

**OODB**. <http://www.mb2.tu-chemmitz.de/oodb/oodb2.html>

**ORFALI**, Robert et.alii. *The Essential Distributed Objects Survival Guide*. USA: Wiley Computer Publishing, 1996.

**PATTERNS**. *Object Magazine*, USA, v.7, n.1, mar.1997.

**PEREIRA**, C. E. *Métodos de Análise de Sistemas de Tempo Real usando Técnicas de Orientação a Objetos*. In: X Simpósio Brasileiro de Engenharia de *Software*, 1996. São Carlos: Departamento de Ciências de Computação e Estatística, 1996.

**PERRY**, G. M. *Access 2 for Windows - Técnicas de Programação*. Tradução por Jorge Chedid. Rio de Janeiro: Axcel Books, 1994.

**PRESSMAN**, R. S. *Engenharia de Software*. Tradução por José Carlos Barbosa dos Santos. 3. ed. São Paulo: Makron Books, 1992.

**QUATRANI**, T. *Visual Modeling with Rational Rose and UML*. USA: Addison Wesley Longman, Inc., 1998.

***RATIONAL SOFTWARE CORPORATION***. *UML past, present and future*. 1997a. <http://www.rational.com/uml/html/summary/summary5.html>

_____ *UML Frequently Asked Questions (FAQs)*. 1997b. <http://www.rational.com/uml/faq.html>.

_____ *UML Proposal to the Object Management Group, in response to the AO&D Task Force's RFP - 1. Version 1.1*. v.1.1, set.1997c. <http://www.rational.com/uml/>

**REINOSO**, G. B., **HEUSER** C. A. *Aplicação comparativa de Técnicas de Análise Orientada a Objetos*. Porto Alegre: UFRGS, 1994. T.I. n.416.

_____ *Comparando o Processo de Modelagem de Técnicas de Análise Orientada a Objetos*. In: VIII Simpósio Brasileiro de Engenharia de *Software*, 1994. Curitiba: Centro Internacional de Tecnologia de *Software*, 1994.

**RICH**, E. e **KNIGHT**, K. *Inteligência Artificial*. Tradução por Maria Cláudia Santos Ribeiro Ratto. 2. ed. São Paulo: Makron Books, 1994.

**RODRIGUES**, S. R., **MONARD**, M. C. *Sistemas baseados em conhecimento - Conceitos fundamentais e aplicações*. In: 1ª Jornada USP - SUCESU-SP de Informática e Telecomunicações, 1993. São Paulo: ICMSC, 1993.

**RUMBAUGH**, J. et alii. *Modelagem e Projetos baseados em Objetos*. Tradução por Dalton Conde de Alencar. Rio de Janeiro: Editora Campus, 1994.

**SHLAER**, S., **MELLOR**, S. J. *Análise de Sistemas Orientada para Objetos*. Tradução por Anna Terzi Giova. São Paulo: Editora McGraw-Hill, 1990.

**SILVA**, R. P., **PRICE**, R. T. *Avaliação de metodologias de análise e projeto orientadas a objetos voltadas ao desenvolvimento de aplicações, sob a ótica de sua utilização no desenvolvimento de frameworks orientados a objetos*. Porto Alegre: UFRGS, 1996.

**SILVA**, W. T. *Algoritmos para Raciocínio Evidencial usando Funções de Crença*. Rio de Janeiro: PUC-RJ, 1991.

**SILVEIRA**, I. F. *Análise de Sistemas Orientada a Objetos: Um Estudo Comparativo das Metodologias de Wirfs-Brock e de Coad/Yourdon*. Porto Alegre: UFRGS, 1995.

**SOMMERVILLE**, I. *Software Engineering*. 4. ed. USA: Addison-Wesley, 1992.

**TANENBAUM** A. S. *Sistemas Operacionais Modernos*. Tradução por Nery Machado Filho. Rio de Janeiro: Prentice Hall do Brasil, 1995.

**THAYER**, R. H. *Tutorial: Software Engineering Project Management*. Los Alamitos: IEEE Computer Society Press, 1988.

**WEISS**, S. M., **KULIKOWSKI**, C. A. *Guia prático para projetar Sistemas Especialistas*. Tradução por Carlos Octário Pavel. Rio de Janeiro: LTC - Livros Técnicos e Científicos Editora S. A., 1988.

**YONG**, C. S. *Banco de Dados - Organização, sistemas, administração*. 5. ed. São Paulo: Editora Atlas, 1988.

# Apêndice A

## *Resumo da notação utilizada pelos métodos*

Este apêndice apresenta o resumo da notação utilizada pelos métodos orientados a objetos descritos neste trabalho, com o objetivo de facilitar a compreensão da sintática e da semântica dos modelos e diagramas gerados a partir da modelagem do sistema utilizado como exemplo. Deste modo, o leitor não necessitará buscar os livros fundamentais dos métodos para obter o resumo da notação dos mesmos.

### *8.1 Notação do método OOSE*

As figuras de A.1 - A.3 apresentam a notação utilizada pelo método OOSE durante as fases de requisitos, análise e projeto.

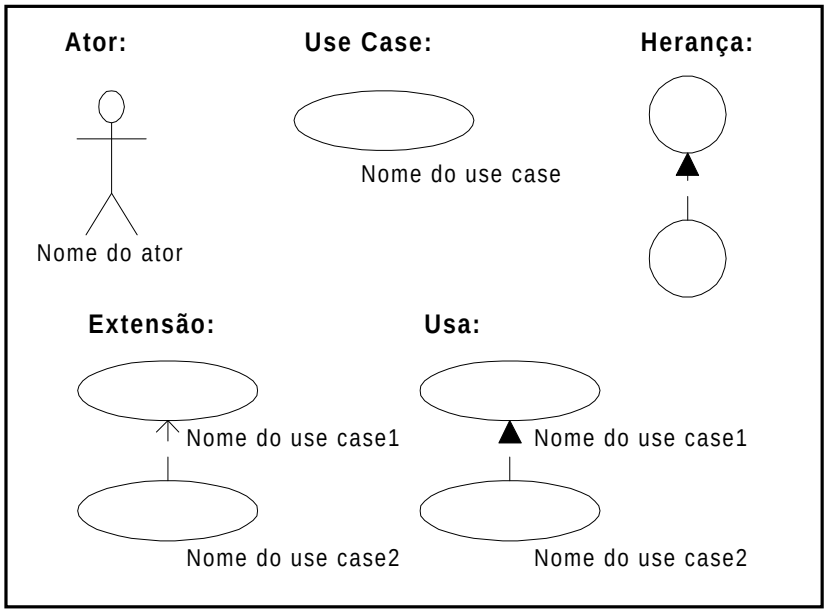



Figura A.1 - Notação utilizada no modelo de requisitos (Jacobson, 1992)

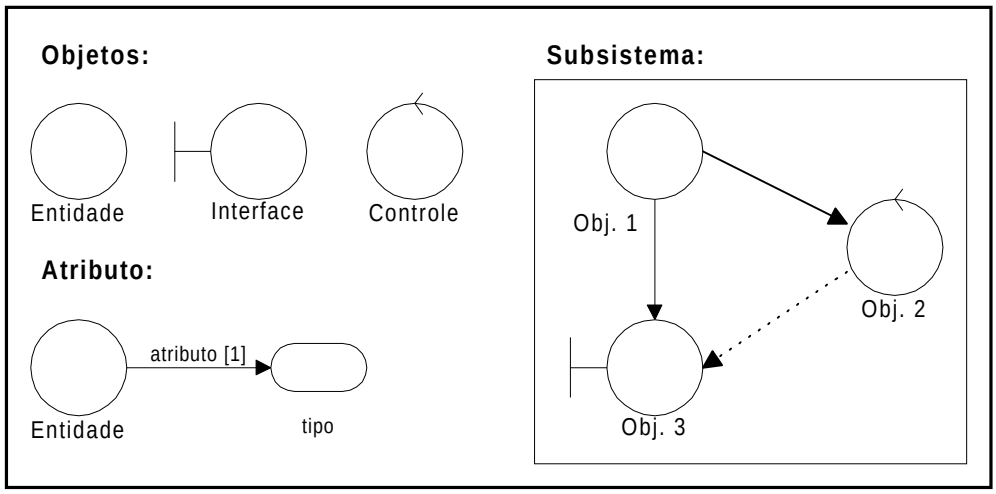



Figura A.2 - Notação utilizada no modelo de análise (Jacobson, 1992)

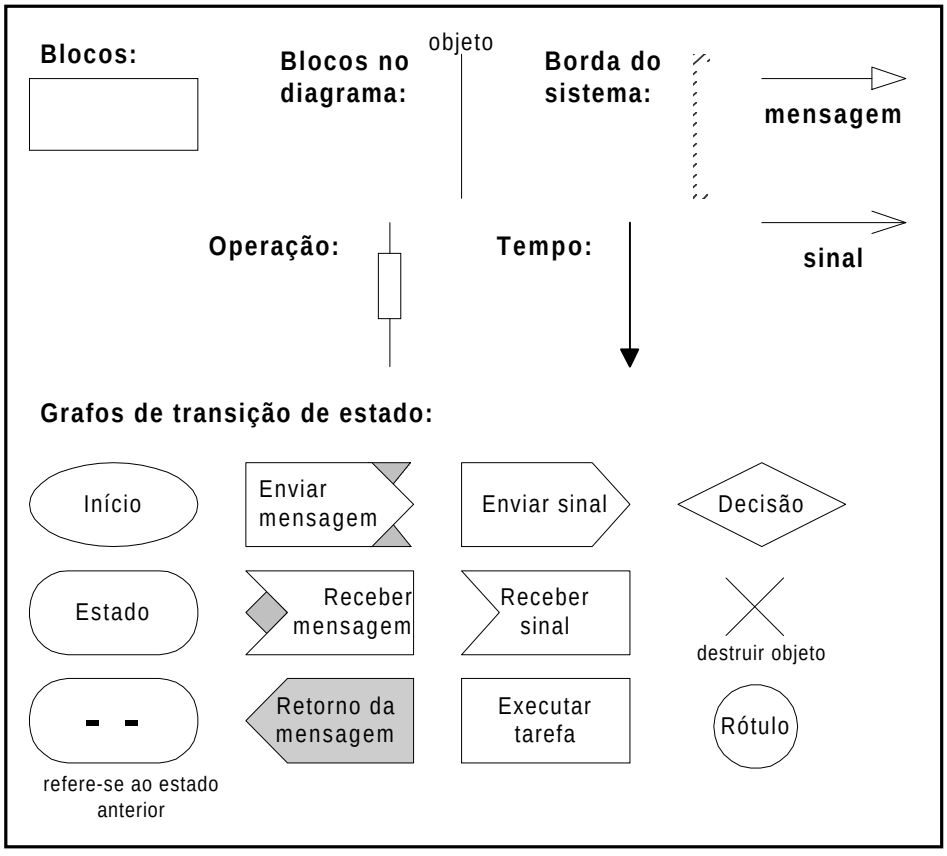



Figura A.3 - Notação utilizada na fase de projeto (Jacobson, 1992)

## *8.2 Notação do método OMT*

As figuras de A.4 - A.10 apresenta a notação utilizada pelo método OMT no modelo de objetos (modelagem comum e avançada), modelo dinâmico e modelo funcional; ambos na fase de análise.

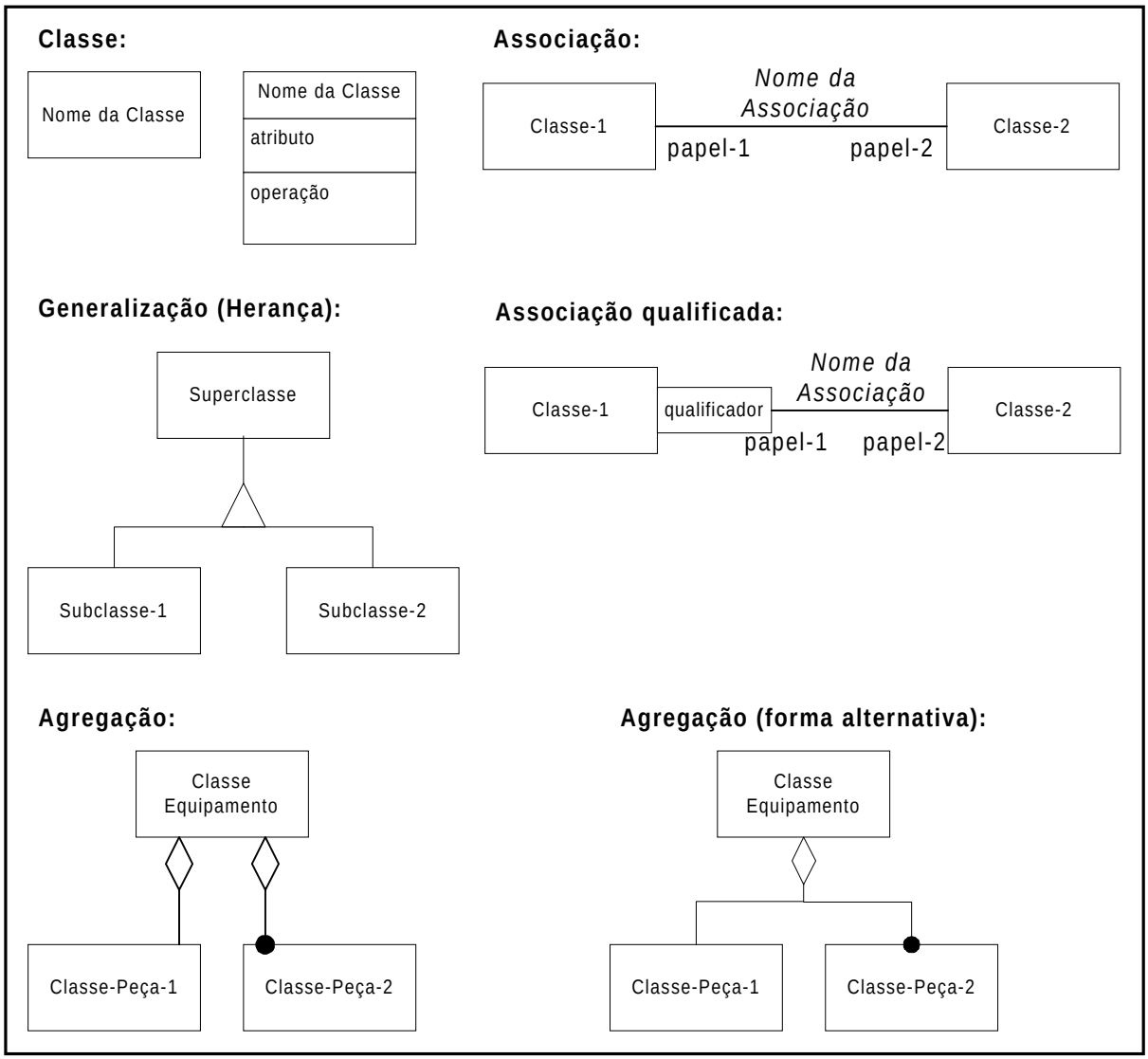



Figura A.4 - Notação utilizada no modelo de objetos (Rumbaugh, 1994)

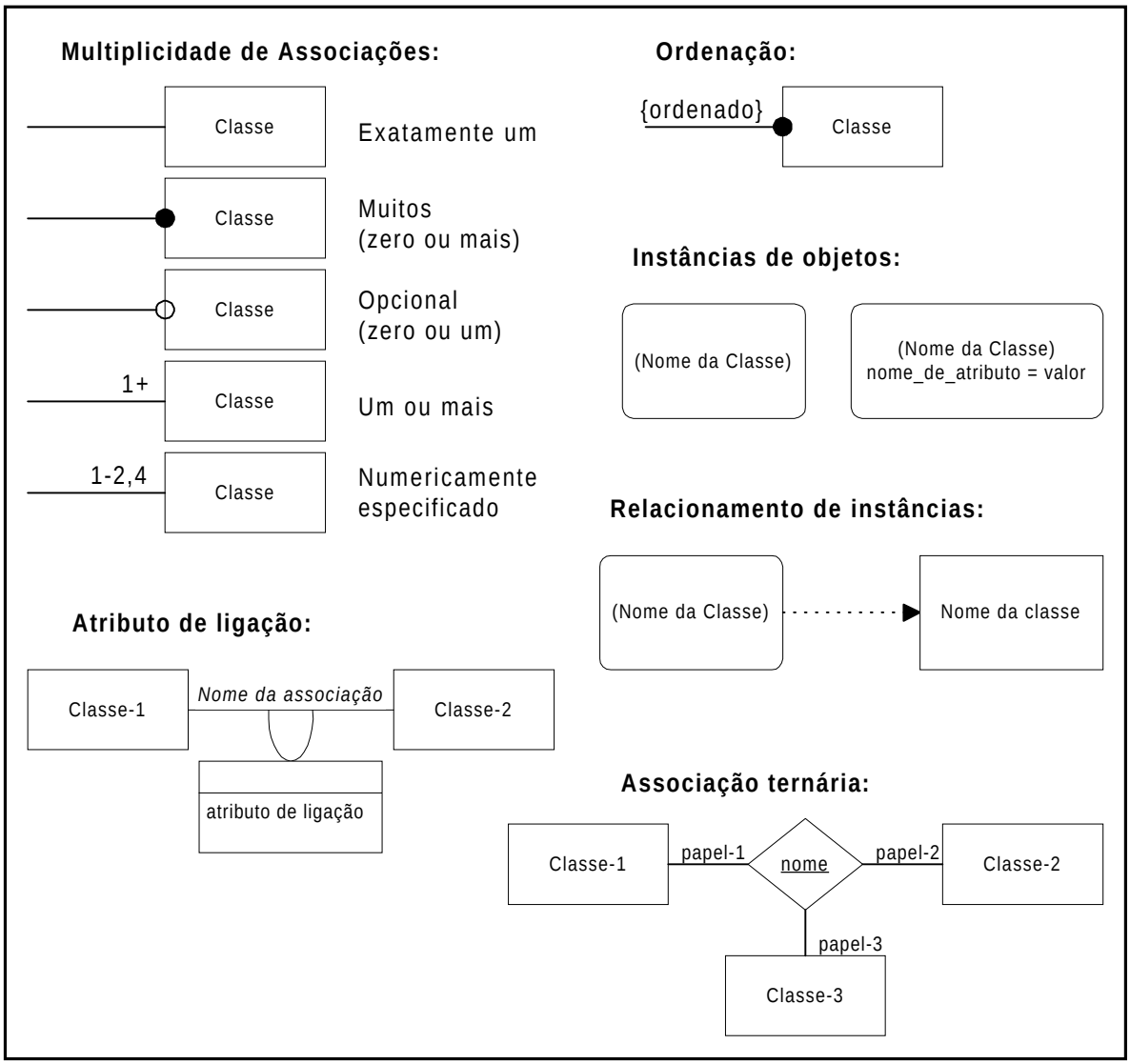



Figura A.5 - Notação utilizada no modelo de objetos - cont. (Rumbaugh, 1994)

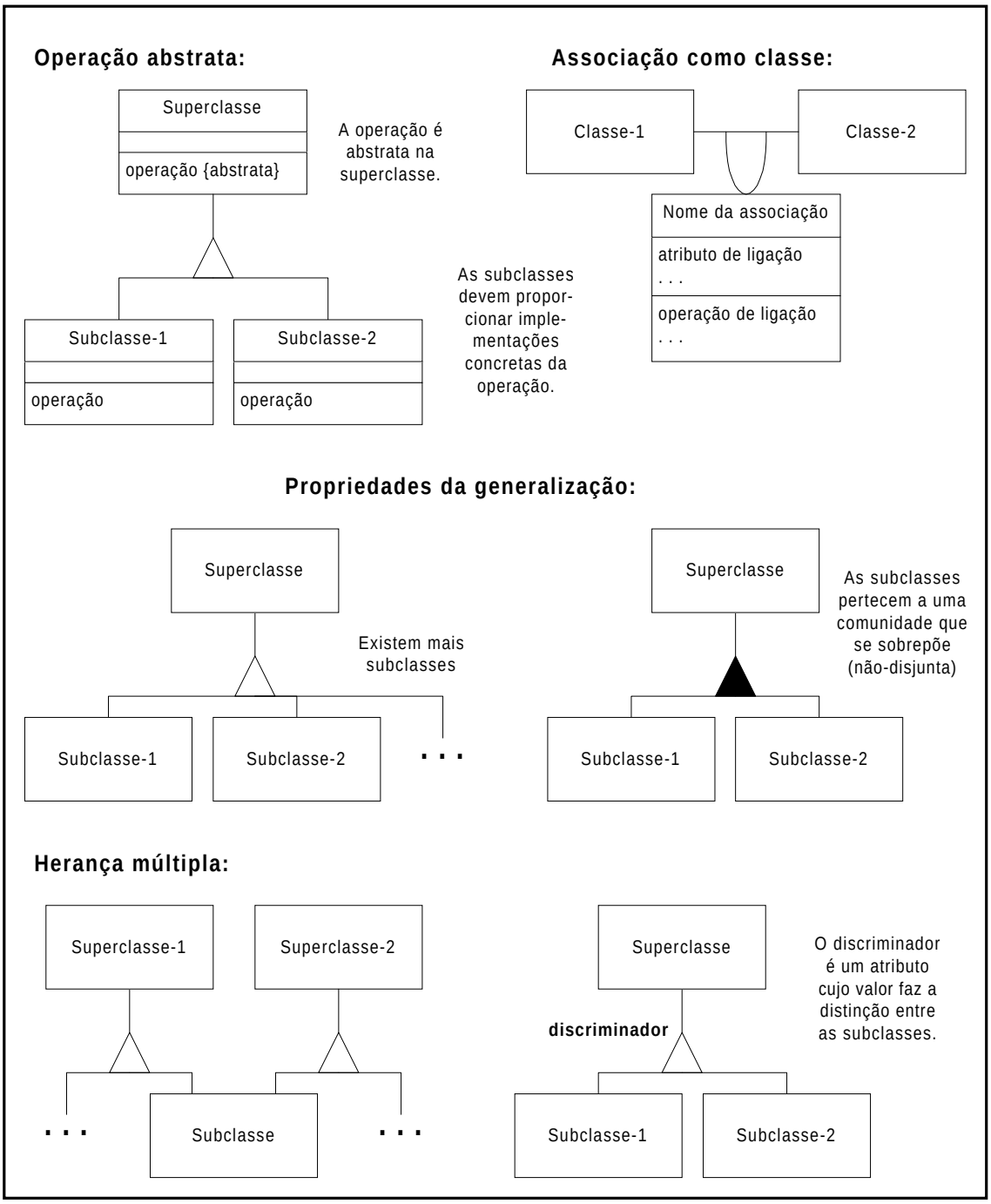



Figura A.6 - Notação utilizada no modelo de objetos - conceitos avançados (Rumbaugh, 1994)

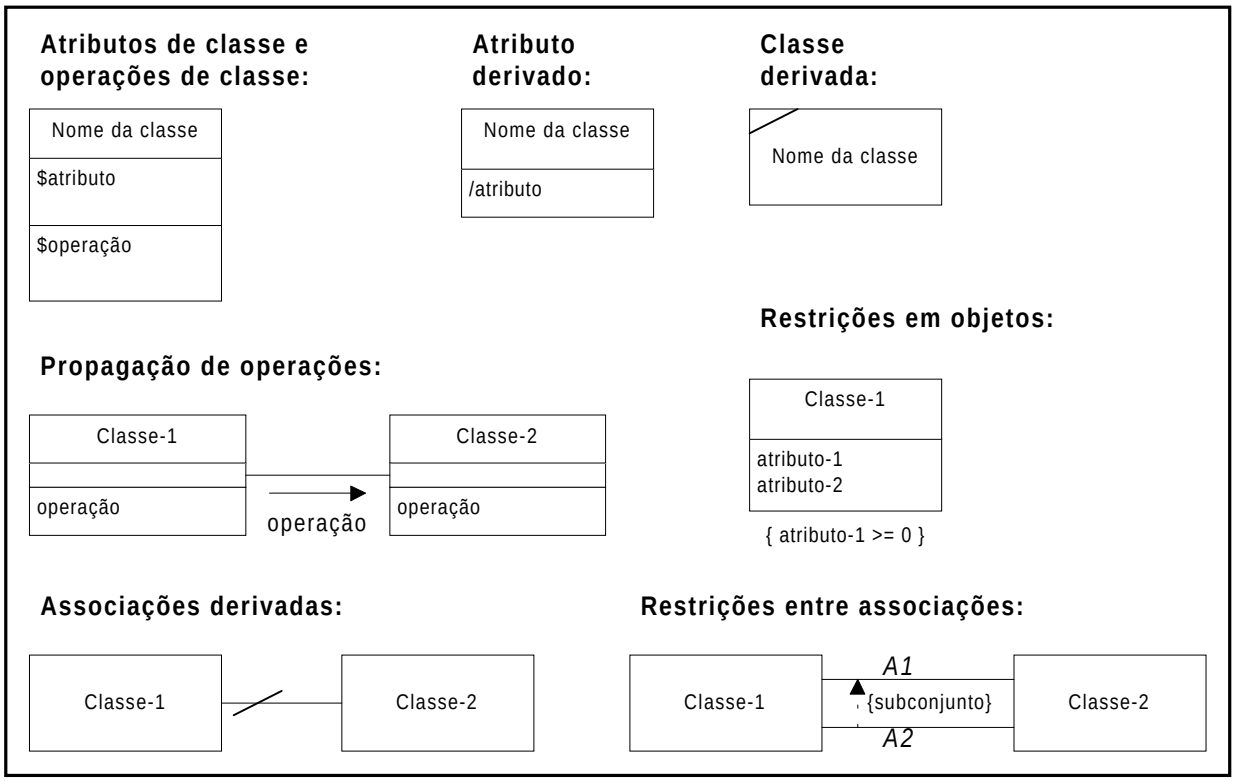



Figura A.7 - Notação utilizada no modelo de objetos - conceitos avançados - cont.

(Rumbaugh, 1994)

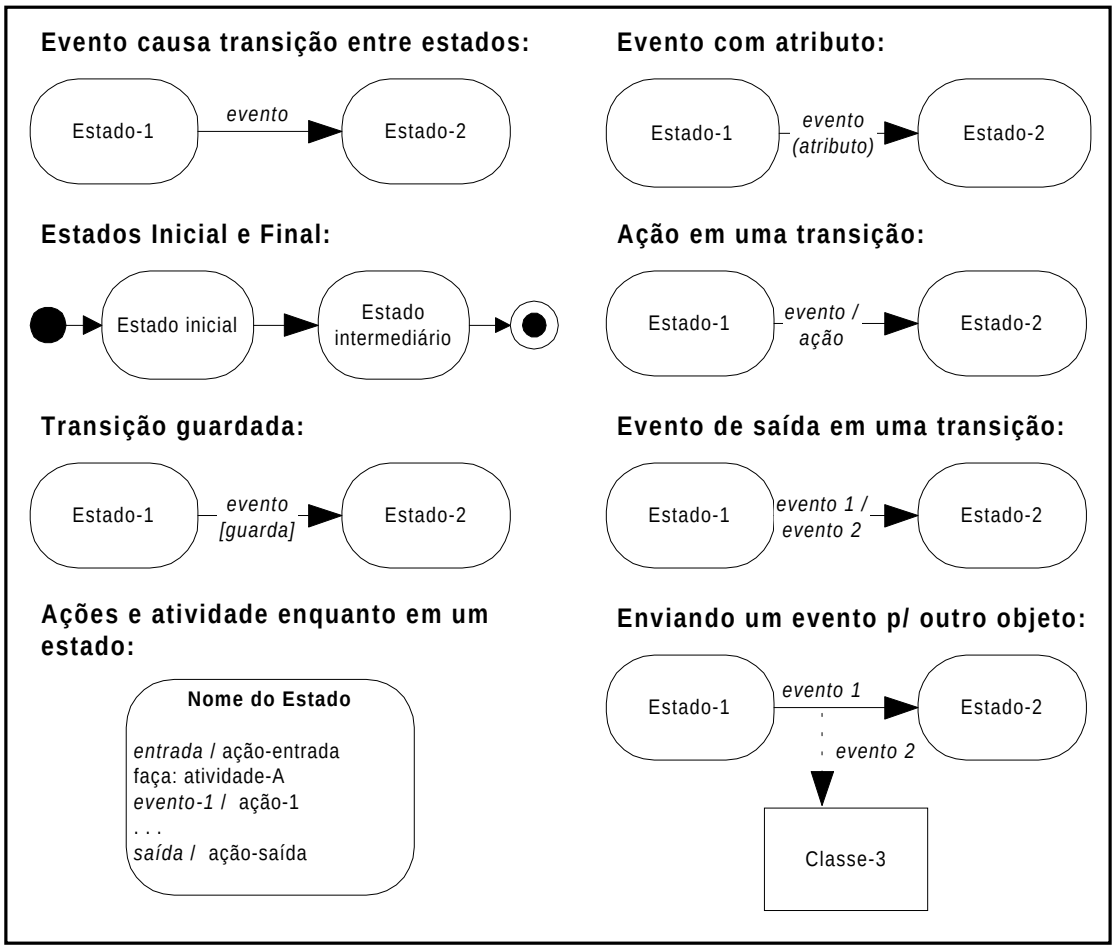



Figura A.8 - Notação utilizada no modelo dinâmico (Rumbaugh, 1994)

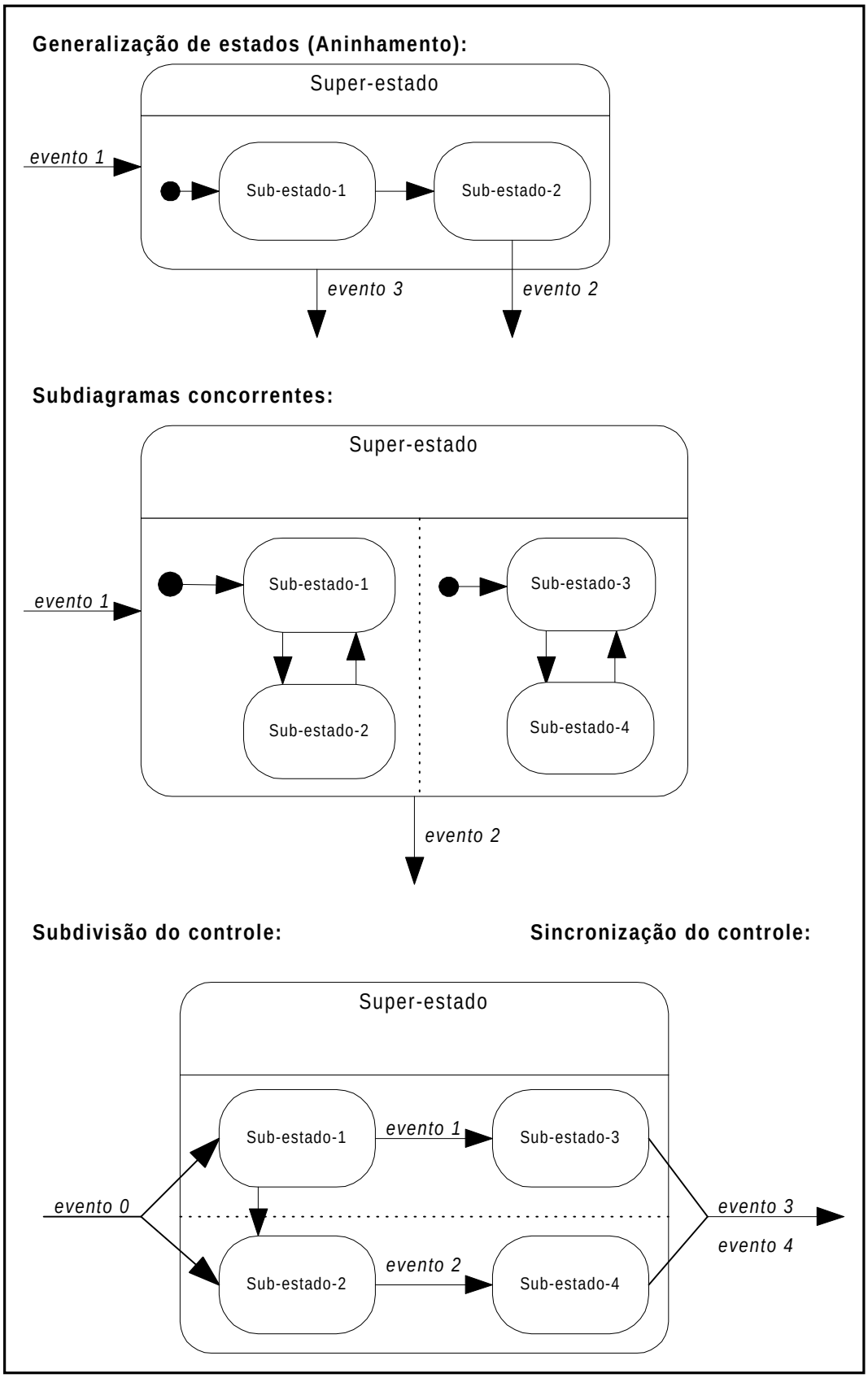



Figura A.9 - Notação utilizada no modelo dinâmico - cont. (Rumbaugh, 1994)

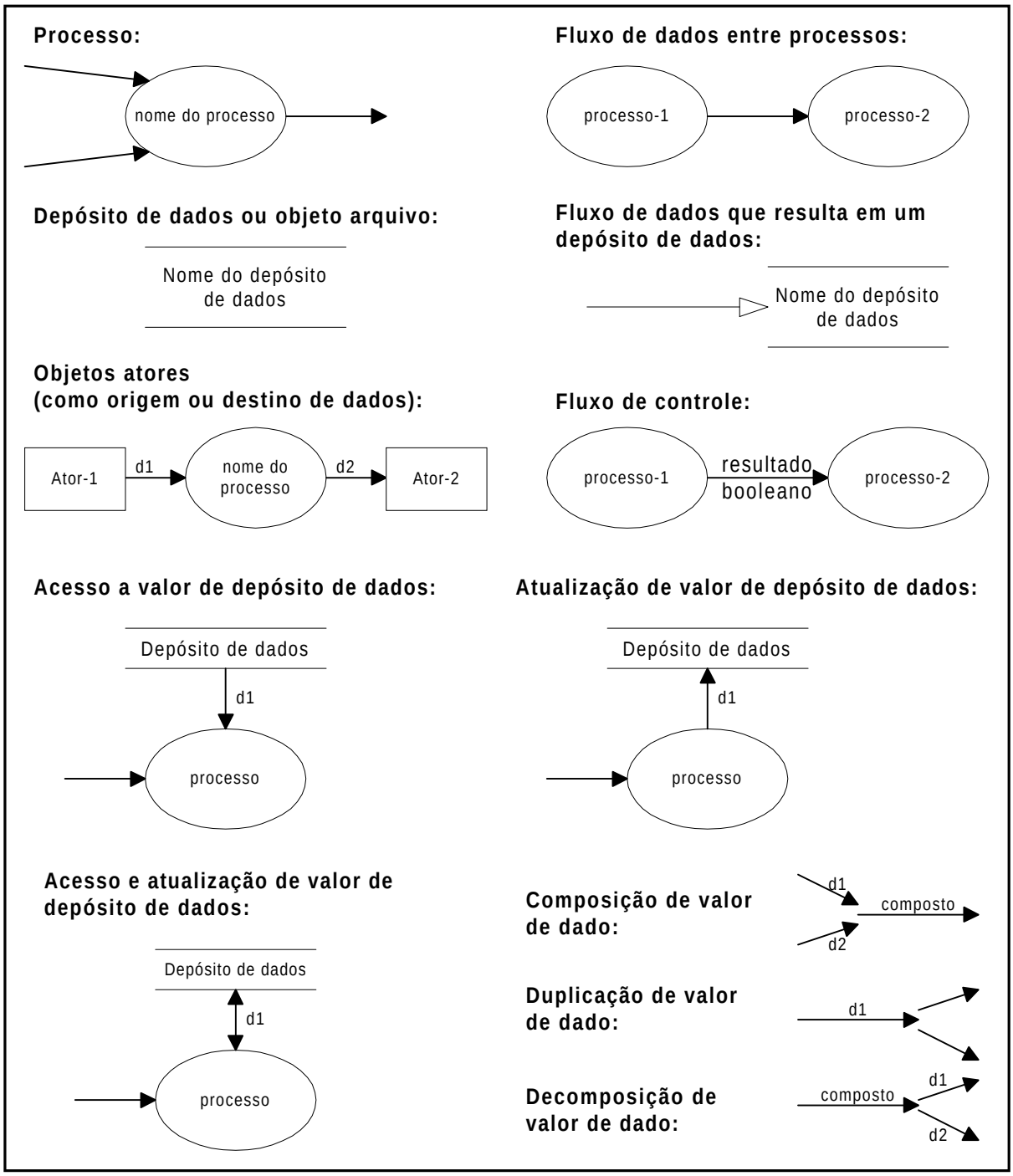



Figura A.10 - Notação utilizada no modelo funcional (Rumbaugh, 1994)

## *8.3 Notação do método OOAD*

As figuras de A.11 - A.16 apresentam a notação utilizada pelo método OOAD para a fase de análise e projeto.

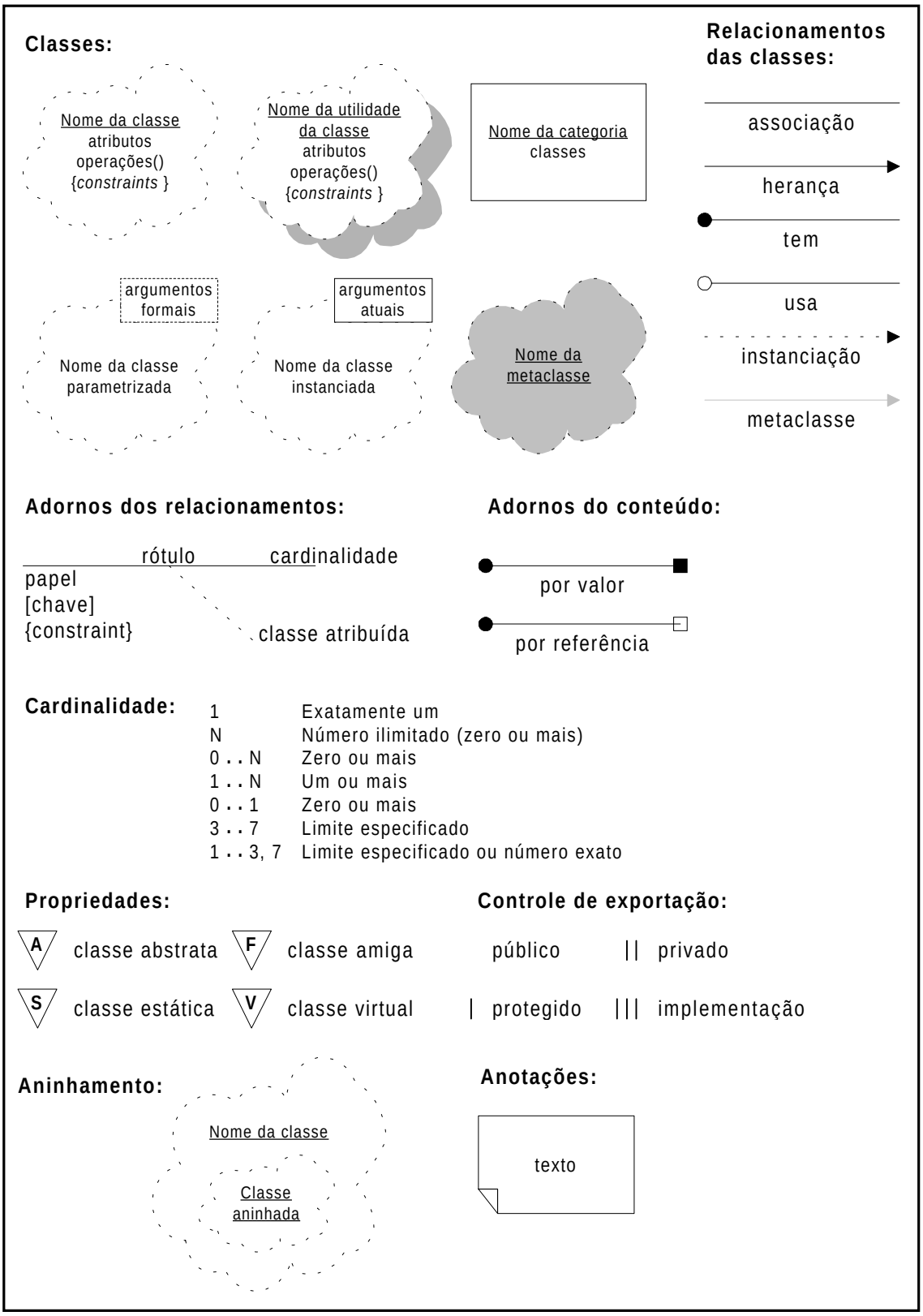



Figura A.11 - Notação utilizada no diagrama de classes (Booch, 1994)

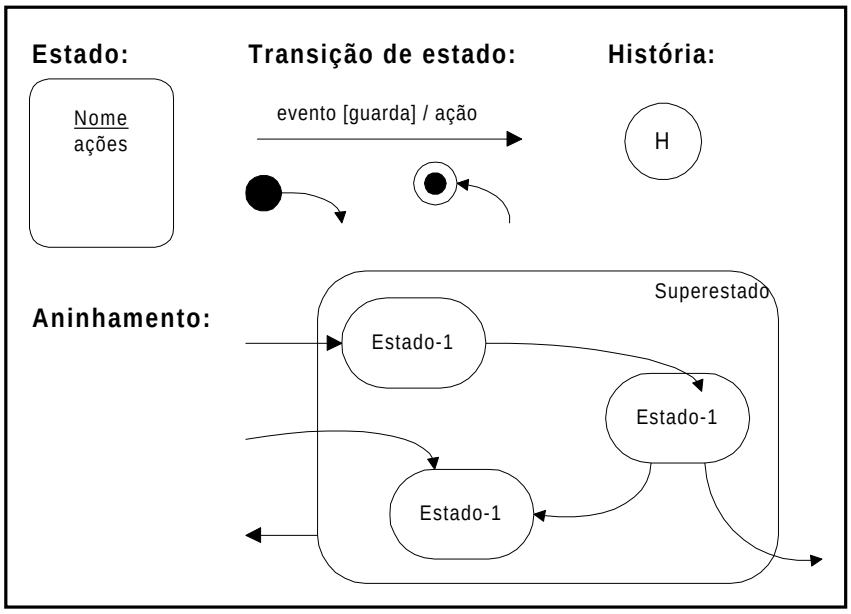



Figura A.12 - Notação utilizada no diagrama de transição de estados (Booch, 1994)

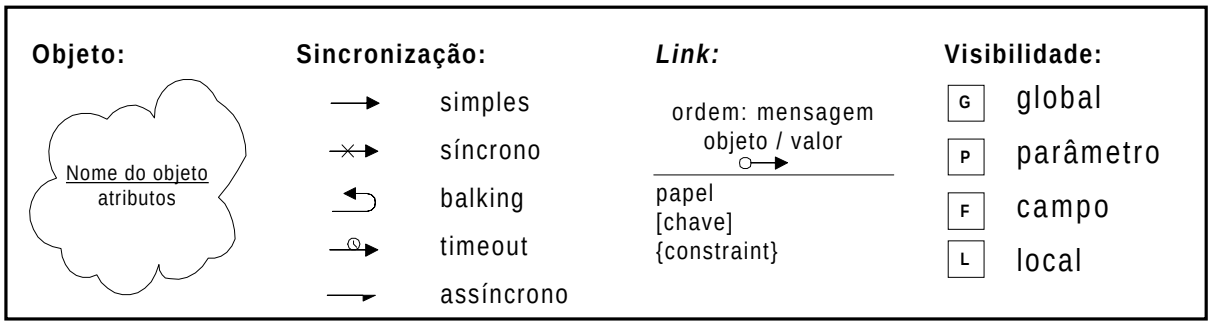



Figura A.13 - Notação utilizada no diagrama de objetos (Booch, 1994)

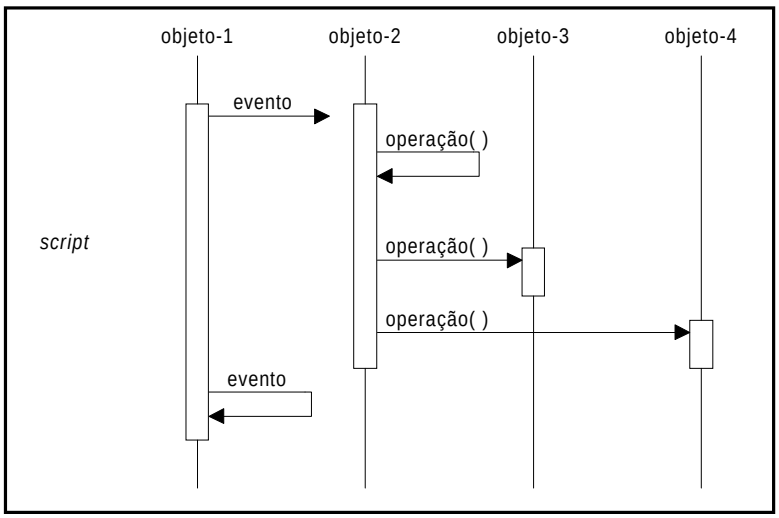



Figura A.14 - Notação utilizada no diagrama de interação (Booch, 1994)

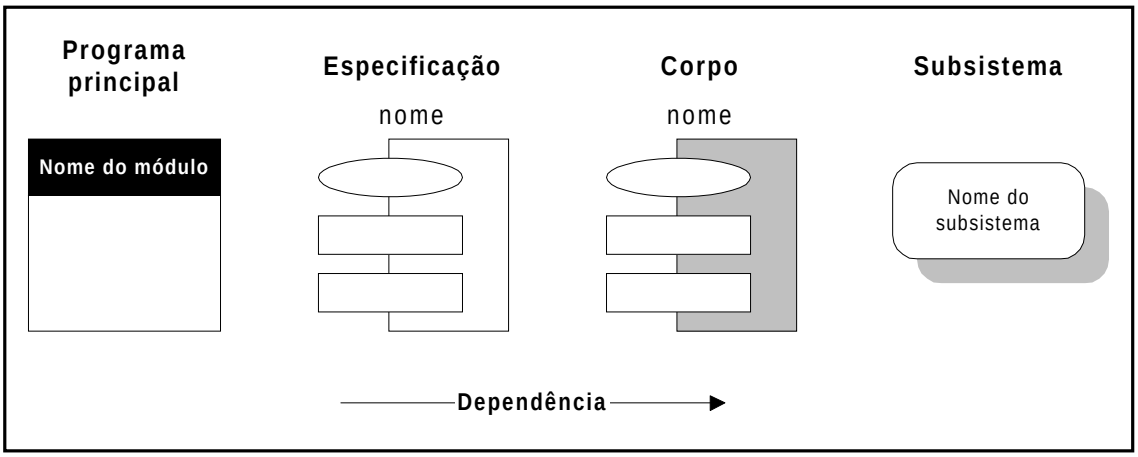



Figura A.15 - Notação utilizada no diagrama de módulo (Booch, 1994)

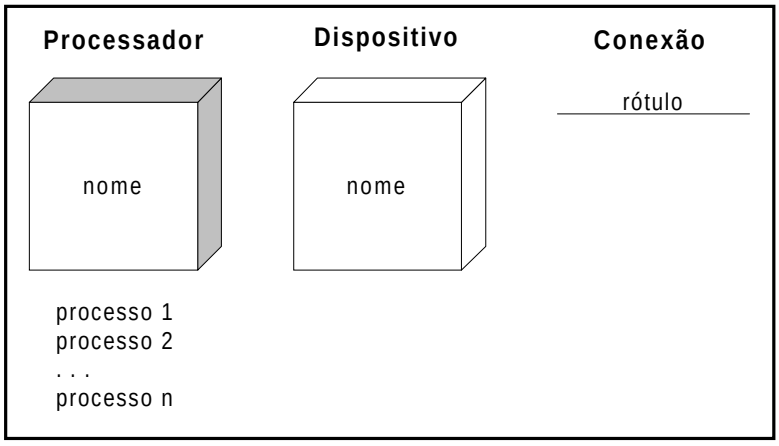



Figura A.16 - Notação utilizada no diagrama de processo (Booch, 1994)

## *8.4 Notação do método OOA-OOD*

As figuras de A.17 - A.20 apresentam a notação utilizada pelo método OOA-OOD, na fase de análise.

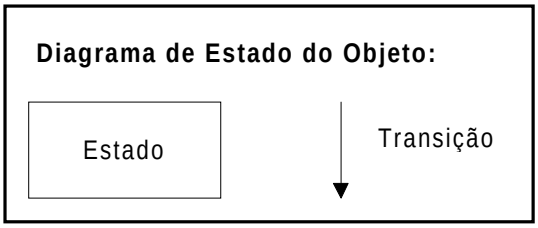



Figura A.17 - Notação utilizada no diagrama de estado do objeto (Coad, 1992)

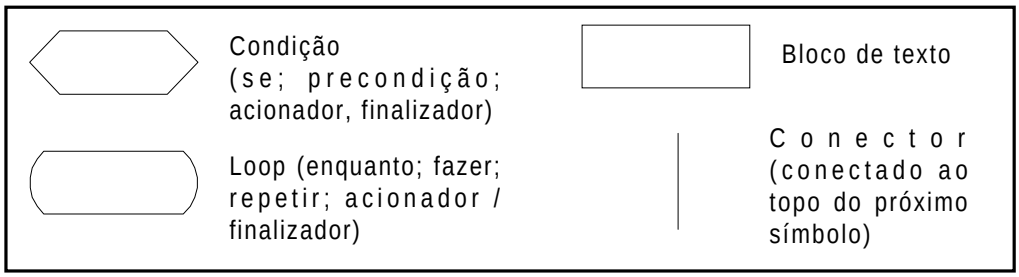



Figura A.18 - Notação utilizada no diagrama de serviço ( Coad, 1992)

**Especificação de Classe&Objeto:**

especificação
*atributo*
*atributo*
*Entrada externa*
*Saída externa*
*Diagrama de Estado de Objeto*
*Especificações adicionais*
*notas*
serviço <nome & Diagrama de Serviço>
serviço <nome & Diagrama de Serviço>

e, na medida do necessário,
*Códigos de acompanhamento*
*Códigos de estado aplicáveis*
*Requisitos de tempo*
*Requisitos de memória*

Figura A.19 - Notação utilizada na especificação classe&objeto ( Coad, 1992)

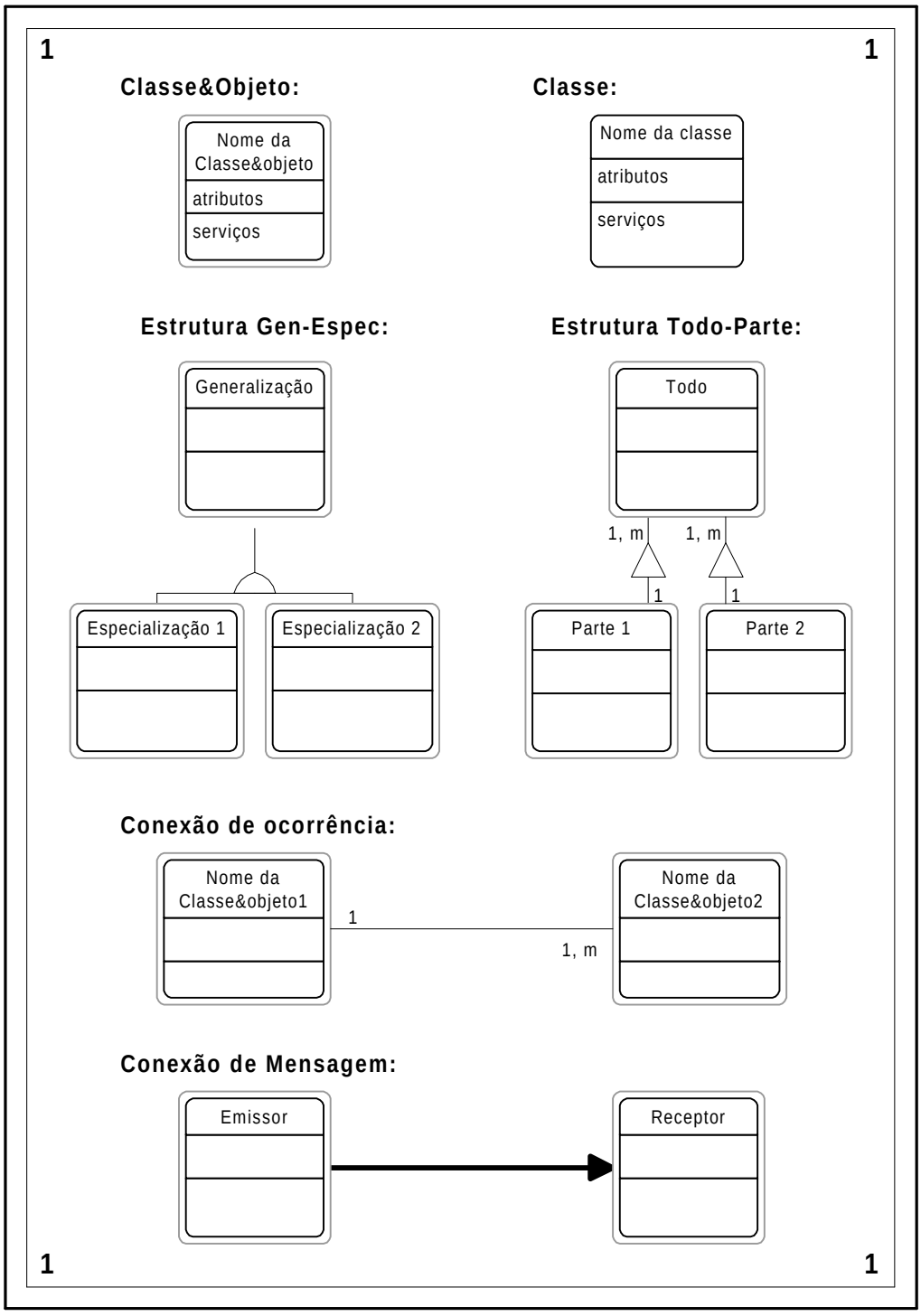



Figura A.20 - Notação utilizada no modelo de análise (Coad, 1992)

## *8.5 Notação do método* Fusion

As figuras de A.21 - A.26 apresentam a notação utilizada pelo método *Fusion*, nos modelos e grafos.

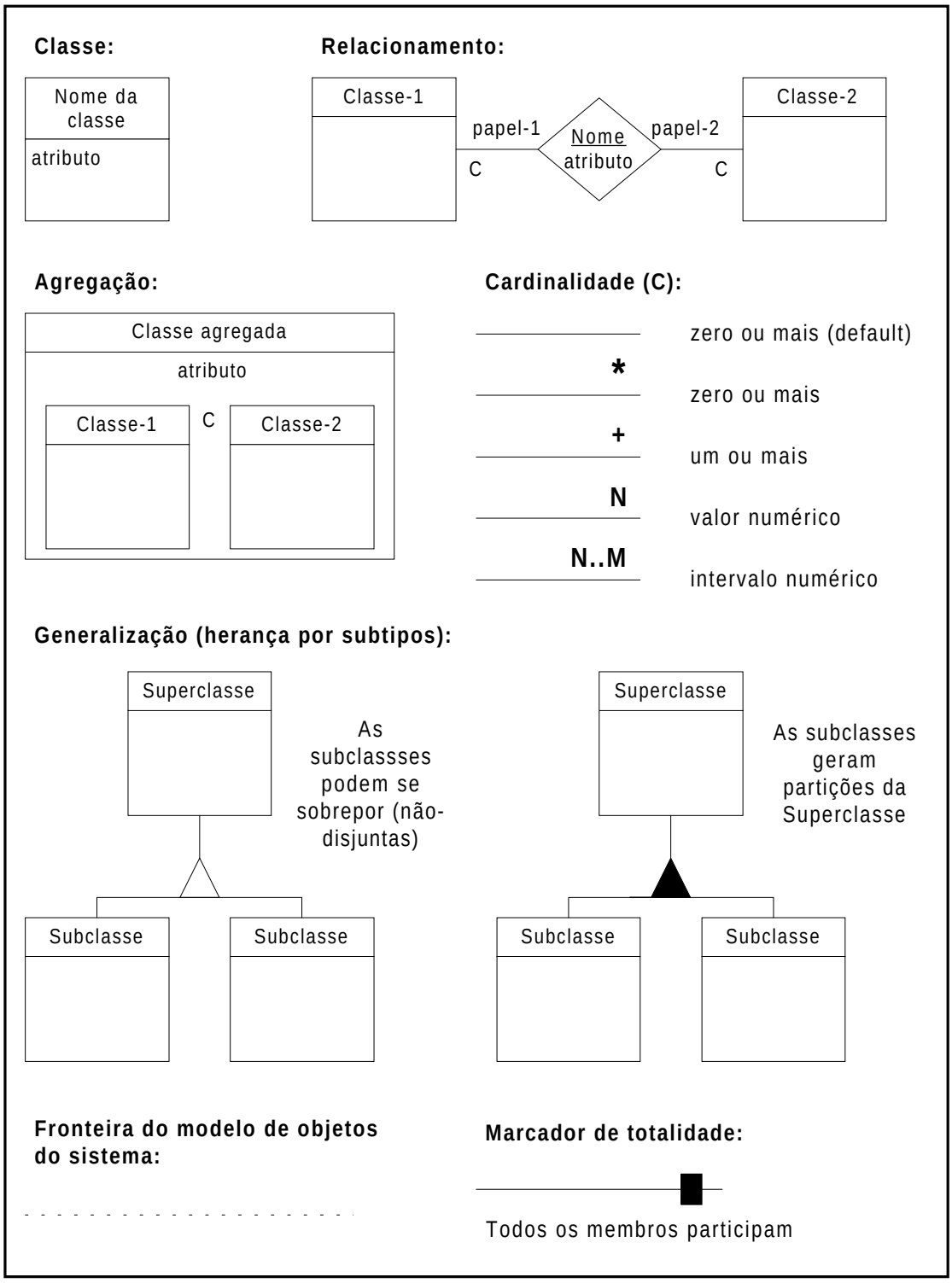



Figura A.21 - Notação utilizada no modelo de objetos (Coleman, 1996)

**Modelo Ciclo-de-vida:**

**lifecicle** [*Nome*:] *Expressão_Regular*
(*NomeLocal* = *Expressão_Regular*)*

*Expressões_Regulares*:

| | |
|---|---|
| Nome | Qualquer nome de evento (operação), nome local ou evento de saída |
| Concatenação | *x.y* |
| Alternância | *x* \| *y* |
| Repetição | |
| Zero ou mais | *x** |
| Uma ou mais | *x*+ |
| Opcional | [*x*] |
| Intercalação | \| \| |
| Agrupamento | (*x*) |

**Modelo de Operação:**

| | |
|---|---|
| **Operation:** | *identificador da operação* |
| **Description**: | *<texto> Descrição da operação* |
| **Reads**: | *<valores fornecidos> <componentes do estado>* |
| **Changes**: | *<valores fornecidos> <componentes do estado>* |
| **Sends**: | *<comunicação com agente>* |
| **Assumes**: | *<asserções>* (*pré-condições*) |
| **Result**: | *<asserções>* (*pós-condições*) |

Figura A.22 - Notação utilizada no modelo de interfaces (Coleman, 1996)

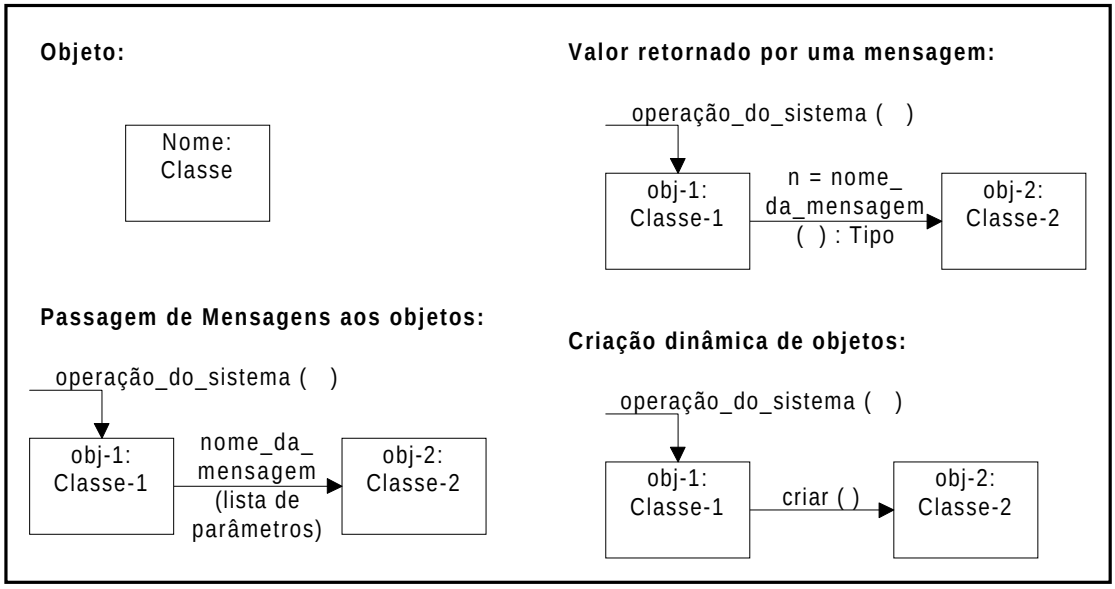



Figura A.23 - Notação utilizada nos grafos de interação de objetos ( Coleman, 1996)

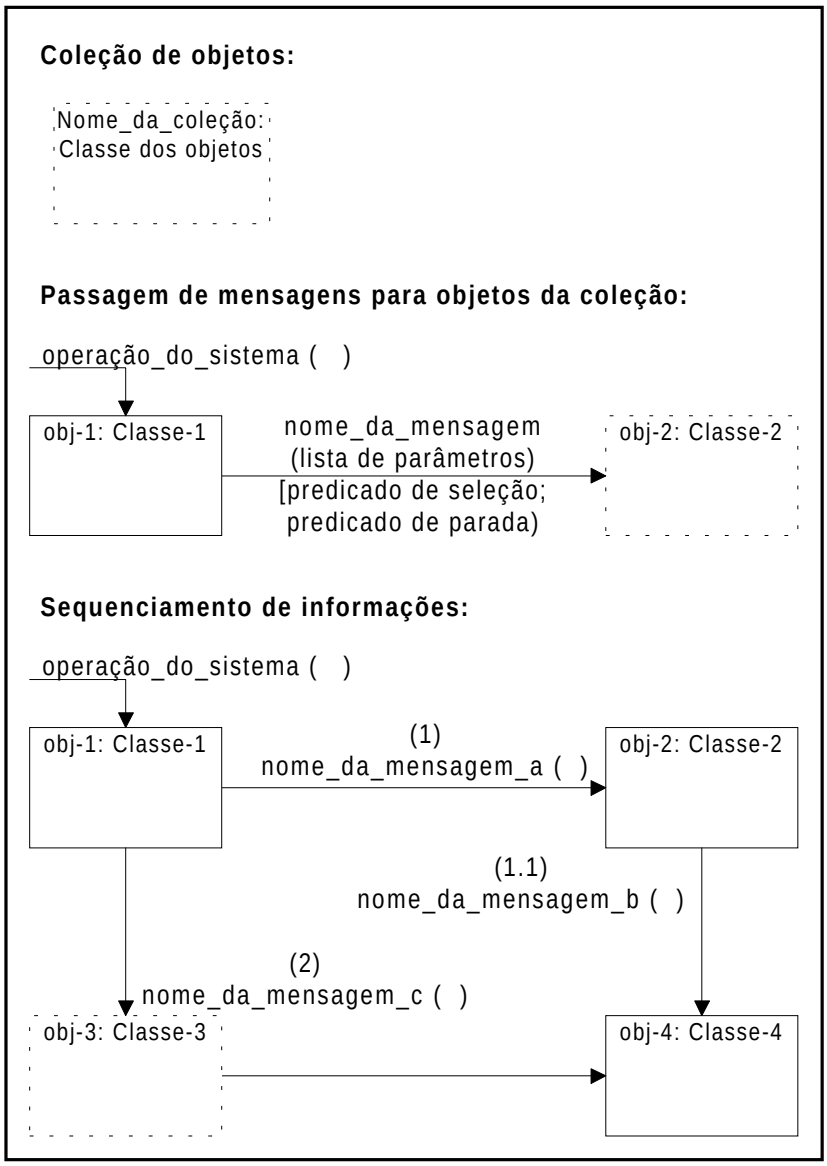



Figura A.24 - Notação utilizada nos grafos de interação de objetos - cont. ( Coleman, 1996)

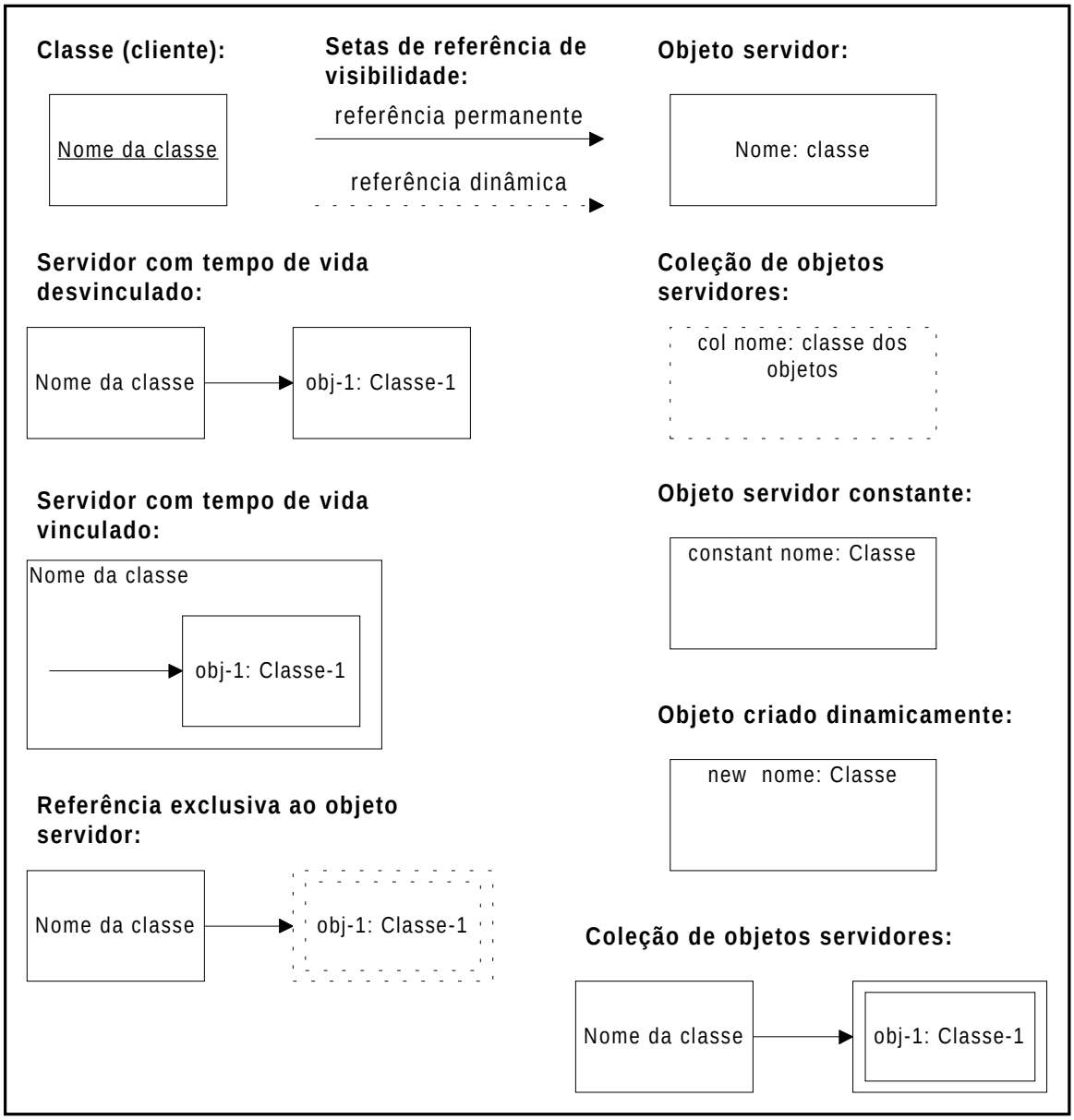



Figura A.25 - Notação utilizada nos grafos de visibilidade (Coleman, 1996)

**Descrição de classe:**

classe <*NomeDaClasse*> [ isa <*NomesDasSuperClasses*> ]
    // para cada atributo
    attribute [Mutabilidade] <*nome_do_atrib*> : [*Compartilhamento*] [*Vinculação*] <*Tipo*>
    .
    .
    // para cada método
    [method] <*nome_do_método*> <*lista_de_argumentos*> [:<*Tipo*>]
    .
    .
endclass

Figura A.26 - Notação utilizada na descrição de classes ( Coleman, 1996)

# Apêndice B

## *Requisitos do sistema de biblioteca*

Este apêndice apresenta os requisitos do sistema de biblioteca facilitando a compreensão do sistema que serviu de exemplo para a modelagem, seguindo os cinco métodos orientados a objetos: OOSE, OMT, OOAD, OOA-OOD e Fusion.

É um sistema de biblioteca simples que controla cadastros (de obras, usuários e funcionários), empréstimos, reservas, cobranças, pagamentos (de dívidas) e emissão de relatórios, bem como aplicação de verificações para empréstimo e reserva.

### *9.1 Objetivos do documento*

Levantar os requisitos necessários a um sistema de automação dos procedimentos relacionados ao empréstimo de livros e demais publicações em uma biblioteca genérica. Este documento deve definir as funções e interfaces desejadas para os sistemas que venham a se basear nesta especificação. Deve ser descrito "o que" se deseja do sistema, que será a referência para as fases posteriores no processo de desenvolvimento. Os conceitos e termos utilizados devem ser o mais próximos possíveis do domínio do problema (biblioteca). Não devem ser utilizados nesta fase termos muito técnicos pertencentes ao domínio de implementação. O documento final deve ser compreensível por especialistas da área de bibliotecas, de modo que possa ser criticado e aperfeiçoado. Este documento visa uma especificação genérica para o problema de automação de empréstimos em bibliotecas. Detalhes de uma implementação específica serão considerados na fase de Levantamento de Requisitos Específicos (LRE). Ao final deste documento as seguintes questões devem estar claramente respondidas: i) quais funções o sistema deve executar?, ii) quais interfaces são necessárias?, iii) quais informações o sistema armazena, produz e consome?, iv) quais as restrições que se aplicam ao sistema? e v) Quais os relacionamentos entre os agentes externos e o sistema?

### *9.2 Glossário básico*

Segue uma lista dos conceitos básicos utilizados no documento:

**Acervo**: conjunto de obras da biblioteca;

**Cadastro**: informações organizadas na forma de um conjunto de registros de estrutura definida.

**Intranet**: rede interna à organização baseada nos protocolos e serviços TCP-IP;

**ISBN**: codificação classificatória internacional atribuída a livros;

**ISSN:** codificação internacional atribuída a periódicos;

**Obra**: referência genérica a livros, periódicos, teses ou qualquer outro documento sob a guarda da biblioteca;

**Tombo**: número de registro sequencial atribuído a uma obra quando a mesma é cadastrada na biblioteca.

## *9.3 Descrição*

O cadastramento de obras e o controle de empréstimos são as atividades principais de qualquer biblioteca. A medida que o volume de obras em uma biblioteca aumenta e o número de usuários também, as atividades de controle deste acervo vão se tornando intensas, dificultando a administração da biblioteca. Os usuários também enfrentam problemas, pois têm dificuldade em localizar as obras desejadas, tendo que muitas vezes se deslocar até a biblioteca e descobrir que a obra desejada não está no cadastro ou está emprestada para outro usuário. São muitos os sistemas informatizados existentes que tratam deste problema, sendo na sua maioria sistemas de grande porte que suportam grandes bibliotecas.

A proposta deste projeto é de um sistema que suporte tanto pequenas como grandes bibliotecas. O sistema deverá ser capaz de ser instalado em microcomputadores (para uma pequena biblioteca), em estações de maior capacidade (para uma biblioteca média), e em estações de alta capacidade (para uma grande biblioteca). Portanto o sistema proposto deverá ser portável para diferentes plataformas de *hardware*/sistema-operacional.

O sistema deverá permitir aos usuários que tiverem acesso à rede *Internet*, acesso a funcionalidades básicas como: pesquisa às obras, reservas, etc., sem a necessidade de se deslocarem até a biblioteca.

Além da portabilidade, o sistema deverá ter projeto modular que permita uma configuração simples para pequenas bibliotecas e configurações mais complexas para grandes bibliotecas.

A biblioteca pode manter em seu acervo os seguintes tipos de obras:

(i) Impressos (papel): livros, teses, monografias, periódicos, enciclopédias, dicionários;

(ii) Mídia digital (disquetes e CDs): *softwares* educacionais, utilitários, enciclopédias, e etc..

(iii) Mídia visual (fitas de vídeo e DVDs ): cursos dirigidos, programas educacionais, etc.

(iv) Publicações na forma de arquivos digitais[25] públicos: teses, monografias, artigos técnicos, etc.

Os sistemas desenvolvidos para esta área deverão dar suporte aos procedimentos de controle do acervo de livros/publicações das instituições e às operações básicas a serem executadas pelos operadores e usuários do sistema.

[25] São arquivos digitais de textos (por exemplo: um documento gerado pelo WORD no formato .doc) que podem ser "copiados" para consulta e pesquisa sem a necessidade do procedimento de empréstimo.

## *9.4 Identificação do sistema e dos agentes externos*

O sistema informatizado é composto por um conjunto de equipamentos centrais, equipamentos remotos, módulos de *software*, módulos de dados e documentos.

O equipamento central armazenará os módulos de informações que conterão todas as informações relativas ao acervo e aos empréstimos de livros, unicamente acessível pelo pessoal de suporte técnico.

Os equipamentos remotos serão os terminais que permitirão o acesso dos usuários e operadores ao sistema. Poderão estar distribuídos pela biblioteca ou departamento (*Intranet*), ou em qualquer lugar por meio da *Internet*.

Os módulos de *software* implementarão as funcionalidades do sistema podendo estar associados ao equipamento central ou aos equipamentos remotos.

Os documentos componentes do sistema são desde os documentos de projeto até os manuais de operação e manutenção do sistema.

Os agentes do sistema são os agentes humanos que interagem com o sistema, são os operadores, usuários, administradores e pessoal de suporte.

Os operadores são os funcionários da biblioteca responsáveis pelas funções rotineiras como empréstimo de livros, cobrança de multas, cadastramento de usuários, cadastramento de livros, reservas e outros.

Os usuários são o público em geral que tiver acesso à biblioteca para fazer consultas e no local da biblioteca utilizar as obras. O público em geral não pode retirar as obras da biblioteca na forma de empréstimo. Os usuários cadastrados podem fazer reservas e empréstimos.

Os administradores são as pessoas que gerenciam a biblioteca e terão acesso a dados globais como: quantidade de livros emprestados, índices de atraso na devolução, índice de perdas de livros, livros mais solicitados, áreas mais solicitadas, etc.

Pessoal de suporte são os profissionais de informática que farão o acompanhamento do sistema durante sua vida útil, administração dos bancos de dados, configurações de redes, manutenções e evoluções no sistema.

## *9.5 Funcionalidades e restrições genéricas do sistema*

As funcionalidades e as restrições genéricas do sistema são divididas nos seguintes itens: i) funcionalidades disponíveis aos usuários, ii) restrições impostas aos usuários que desejem fazer reservas, iii) restrições impostas às obras a serem reservadas, iv) funcionalidades disponíveis aos operadores, v) restrições impostas aos usuários que desejem fazer empréstimo, vi) restrições impostas às obras a serem emprestadas; vii) funcionalidades disponíveis aos administradores e viii) funcionalidades disponíveis ao pessoal de suporte.

**i) Funcionalidades disponíveis aos usuários**

1. "Navegar" pelo tutorial da biblioteca e manual de uso do sistema;
2. Verificar a disponibilidade de determinada obra no acervo da biblioteca;
3. Verificar a lista de obras disponíveis por determinado autor ou assunto (palavra chave);
4. Verificar a disponibilidade para empréstimo de uma obra específica;
5. Visualizar a ficha de resumo de determinada obra;
6. Acessar o documento digital (arquivo de texto) de determinada obra quando disponível (teses, monografias e etc.);
7. Impressão da ficha cadastral da obra;
8. Observação: as funcionalidades disponíveis aos usuários poderão ser acessadas por meio de terminais colocados na biblioteca para este fim ou por meio de microcomputadores pessoais interligados à *Internet*.

**ii) Restrições impostas aos usuários que desejem fazer reservas:**

1. Ser um usuário cadastrado;
2. Não estar devendo nenhuma obra e nenhuma multa.

**iii) Restrições impostas as obras a serem reservadas**

1. só será efetuada a reserva se houver disponibilidade de títulos daquela obra no período desejado.

**iv) Funcionalidades disponíveis aos operadores**

1. Emprestar obras para usuários cadastrados;
2. Receber retorno de obra/tombo emprestado;
3. Cobrar multa por atraso na devolução;
4. Imprimir recibo da multa, lista de obras que estão emprestadas a determinado usuário ou ficha de identificação do tombo/obra
5. Cadastrar novo usuário, nova obra/tombo ou novo tombo de obra já cadastrada;
6. Excluir usuário, obras ou tombo do cadastro;
7. Fazer reserva de obras/tombo;
8. Gerar lista de usuários devedores de obras/tombos;
9. Gerar arquivos texto para cobranças de obras/tombos emprestados não devolvidos;
10. Observação: as funcionalidades disponíveis aos operadores serão executadas em terminais localizados na própria biblioteca, não estando disponíveis via *Internet*, mas apenas na *Intranet* da biblioteca.

**v) Restrições impostas aos usuários que desejem fazer empréstimo**

1. Ser um usuário cadastrado;
2. Não estar devendo nenhuma obra e nenhuma multa;
3. Não ter excedido cota de empréstimo.

**vi) Restrições impostas as obras a serem emprestadas**

1. Número de obras reservadas menor que número de tombos disponíveis;
2. Obra/tombo não estar em restauração;
3. Observação: o fato de uma obra ter sido reservada não implica necessariamente que a mesma será emprestada. As condições de empréstimo serão checadas no momento do empréstimo.

**vii) Funcionalidades disponíveis aos administradores**

1. Estatísticas básicas do movimento de empréstimos: obras mais solicitadas, obras extraviadas, obras em restauração, total de empréstimos (diário, semanal, mensal e anual).
2. Estatísticas de atrasos e multas: tempo médio de atraso nas devoluções, listagem dos usuários atrasados.
3. Listagens de controle: listagem do cadastro de usuários, listagem do cadastro de obras.
4. Observação: as funcionalidades de administração serão acessíveis em equipamento remoto (*Intranet*) e as listagens serão geradas na impressora matricial conectada ao equipamento central.

**viii) Funcionalidades disponíveis ao pessoal de suporte**

1. Relatórios de gerenciamento do sistema: logs de falhas ocorridas em acessos aos dados, logs de falhas de comunicação (*Intranet* e *Internet*), disponibilidade de disco rígido livre no equipamento central, disponibilidade de memória livre (média) no equipamento central, tempo médio das consultas locais, tempo médio das consultas remotas.
2. Funções de manutenção: *back-ups* das bases de dados, restauração das bases de dados.
3. Observação: estas funcionalidades estarão disponíveis apenas no equipamento central.

## *9.6 Interfaces entre os agentes externos e o sistema*

A interface gráfica entre os agentes e o sistema será baseada em telas gráficas, orientadas por ícones, caixas de seleção e menus do tipo "*Pull-Down*". Deverá ter a capacidade de ser visualizada em estações *Risc/Unix* ou em PCs/*Windows*, com acessos ao equipamento central através da *Intranet* da biblioteca ou através da *Internet*.

## *9.7 Identificação dos elementos de informação*

As principais informações do sistema devem ser organizadas na forma de um conjunto de "fichários". Cada fichário conterá informação específicas, como por exemplo: fichário de obras catalogadas na biblioteca, fichário de usuários registrados, fichário de funcionários registrados e outros que forem necessários.

As informações em uma biblioteca normalmente são organizadas na forma de fichas com diversos tipos de dados, que passam a ser descritos a seguir.

**i) Fichas de identificação de obras**

Estas fichas são as fichas coladas nas contracapas traseiras das obras com a identificação da mesma. A figura B.1 ilustra o mínimo de informação que deve constar desta ficha. As obras iguais terão a mesma catalogação mas números de tombo diferentes.

**Obra**: (nome da obra)

**Autor**: (Autor ou autores da obra)

**Número do tombo**: (numeração seqüencial atribuída as Obras cadastradas)

**Data para devolução**: (data em que deve ser efetuada a devolução do tombo)

**Catalogação bibliográfica**: (número de identificação única atribuído a Obra, em função da área, assunto, etc.)

Figura B.1 - Definição da ficha_tombo

**ii) Fichas de controle de empréstimo**

São as fichas destacáveis presentes na contracapa traseira das obras antes de serem emprestadas. Esta ficha é utilizada como documento que comprova o empréstimo, é retirada do tombo no ato do empréstimo e retornando ao mesmo após a devolução. Elas contêm a mesma informação das fichas de identificação de obras, com o acréscimo de dois itens para anotações, como ilustrado pela figura B.2.

**Obra**: (nome da obra)

**Autor**: (Autor ou autores da obra)

**Número do tombo**: (numeração seqüencial atribuída as Obras cadastradas)

**Data para devolução**: (data em que deve ser efetuada a devolução do tombo)

**Catalogação bibliográfica**: (número de identificação única atribuído a Obra, em função da área, assunto, etc.)

**Código identificador do emprestante**: (número cadastral do solicitante do empréstimo)

**Campo para assinatura do emprestante**

Figura B.2 - Definição da ficha_empréstimo

O sistema automatizado deve oferecer uma alternativa ao uso das fichas, através de senhas e assinaturas digitais, mantendo a confiabilidade do sistema de fichas. O sistema deve ser implementado de modo a permitir o uso simultâneo das fichas e assinatura digital, até que se tenha confiança total no novo sistema.

**iii) Fichas de catalogação de obras**

Estas fichas permitem os usuários procurar pela obra desejada, elas são únicas para cada código de catalogação bibliográfica, ou seja, uma mesma ficha pode corresponder a vários tombos de mesma catalogação. Estas fichas contêm as informações ilustradas pela figura B.3.

| | |
|---|---|
| **Título da Obra**: | |
| **Autor**: | |
| **Edição**: | **Volume**: |
| **Local da edição**: | **Editora**: |
| **Ano**: | **Número de páginas**: |

Figura B.3 - Descrição da ficha_título

Observações: a medida que o sistema automatizado for entrando em operação não será mais necessário o uso destas fichas na forma de papel, sendo possível a sua visualização pelo sistema e a sua impressão caso desejado.

Alguns tipos específicos de obras poderão ter necessidade de alguma informação adicional, podendo ser criadas fichas especiais para tal.

**iv) Ficha de cadastro de usuário**

Esta ficha contém os dados individuais de cada usuário cadastrado no sistema, tais como dados pessoais e relativos a empréstimo, como ilustrado pela figura B.4.

| |
|---|
| **Número no cadastro**: (número de cadastro gerado pelo sistema) |
| **Nome completo do usuário**: |
| **Nome para o sistema**: (nome com 8 caracteres) |
| **R.G.:** |
| **Endereço**: |
| **Telefone**: |
| **CEP**: |
| **E-mail**: |
| **Cota para empréstimo**: (quantidade de obras emprestadas) |
| **Cota emprestada**: (quantidade de obras que podem ser emprestadas) |
| **Dívida total**: (valor total que o usuário deve, relativo a devoluções de empréstimos atrasadas) |
| **Tipo**: (o tipo de usuário determina a quantidade de dias para o empréstimo) |
| **Senha**: (senha informada pelo usuário para verificações posteriores) |

Figura B.4 - Descrição da ficha_Usuário

## *9.8 Aspectos gerais sobre o sistema*

**i) Capacidade**

O sistema deverá ter capacidade para na sua configuração mínima armazenar informações de 1.000 (mil) obras e ter até 250 (duzentos e cinquenta) usuários cadastrados. Na sua configuração máxima 1.000.000 (um milhão) de obras e 25.000 (vinte e cinco mil) usuários. A fase de Levantamento de Requisitos Específicos (LRE) é que deverá definir a faixa de capacidade para cada projeto (biblioteca) específico. A fase de Definição da Arquitetura de Implementação (DAI) deverá

levar em conta a escalabilidade do sistema, permitindo a evolução do sistema para equipamentos de maior capacidade sem a necessidade de alterações de *software*. Portanto os módulos de *software* produzidos para este sistema deverão ser portáveis para diferentes plataformas de *hardware*/sistema-operacional.

**ii) Desempenho**

Desconsiderando a possibilidade de consultas simultâneas em uma *Intranet* com baixo tráfego, o tempo médio de consultas e pesquisas não deve ultrapassar os 10 segundos. Com o crescimento da base de obras cadastradas este tempo de pesquisa tenderá a crescer e quando atingir os 10 segundos seria um indicativo da necessidade de evolução da plataforma de *hardware*.

**iii) Procedimentos contra perdas de informações**

O sistema deverá ter procedimentos automatizados que permitam a recuperação de todas as informações do sistema no caso de ocorrerem as seguintes falhas: perda parcial do disco rígido, perda total do disco rígido e problema de CPU.

Para atingir tal nível de confiabilidade deverá ser feito pelo menos uma cópia diária (por exemplo em fita DAT) de todas os empréstimos e devoluções efetuados no dia e de todos os cadastramentos feitos. Também deverá ser feita periodicamente uma cópia de todo o cadastro e do estado dos empréstimos.

Para evitar a possibilidade de perdas entre as cópias diárias, deverão ser feitas cópias contínuas de todas as transações que alterem as informações do sistema em um dispositivo magnético (disquete ou *Zip-Drive*) de maneira que a qualquer momento que ocorra uma falha haja possibilidade de recuperação das operações realizadas no dia.

**iv) Tratamento de erros**

Todos o erros ocorridos no sistema deverão ser armazenados em arquivos de LOG, com identificação do módulo causador, tipo do erro e a hora de ocorrência com o objetivo de facilitar a identificação de problemas de *software* e *hardware* que estejam restringindo as funcionalidades do sistema. Estes arquivos terão um procedimento de *back-up* individualizado, sendo eliminados do disco rígido quando copiados.

**v) Restrições gerais do sistema**

O sistema não se aplica a quaisquer outras necessidades de uma biblioteca que não estejam diretamente relacionadas à guarda e empréstimo de obras.

O acesso via *Internet* implicará ao usuário ter instalado em seu computador um *software* "navegador" atualizado para permitir as consultas remotas. O tempo de consulta via *Internet* não é determinístico, pois uma série de fatores, fora do controle da biblioteca, podem influenciar.

Não haverá possibilidade de cadastro de usuários via *Internet*, o cadastramento deverá ser feito pessoalmente pelo usuário na biblioteca, com a apresentação dos documentos necessários.

# Apêndice C

## *Modelagem do sistema pelos métodos*

Neste apêndice se encontram todos os modelos e/ou diagramas criados para o sistema de biblioteca (descrito no apêndice B), durante o estudo de cada um dos métodos.

O objetivo deste apêndice é apresentar os modelos e diagramas sugeridos por cada método, tentando apresentar todos os seus aspectos para melhor ilustrá-los. Eventualmente, isso não será possível devido ao exemplo utilizado, visto que um método aborda várias dimensões de um sistema, alguma desta dimensão pode não ser aplicável ao exemplo.

É importante ressaltar que nem todos os requisitos do sistema (apêndice B) foram modelados, haja visto que a modelagem completa do sistema demandaria um tempo precioso e seria desnecessário, pois o objetivo dessa modelagem é facilitar o estudo e comparação dos métodos. Neste sentido o enfoque foi dado a parte de empréstimo e reserva, que são as que oferecem mais dimensões a serem tratadas.

As seções a seguir apresentam os modelos separados por métodos da seguinte maneira:

- C.2 - Modelagem em OOSE;
- C.3 - Modelagem em OMT;
- C.4 - Modelagem em OOAD;
- C.5 - Modelagem em OOA-OOD;
- C.6 - Modelagem em Fusion.

A seção C.7 descreve o dicionário de dados para o sistema, abordando todas as classes, atributos, operações, associações e eventos, referentes aos modelos apresentados nas seções anteriores.

## 10.1 Modelagem em OOSE

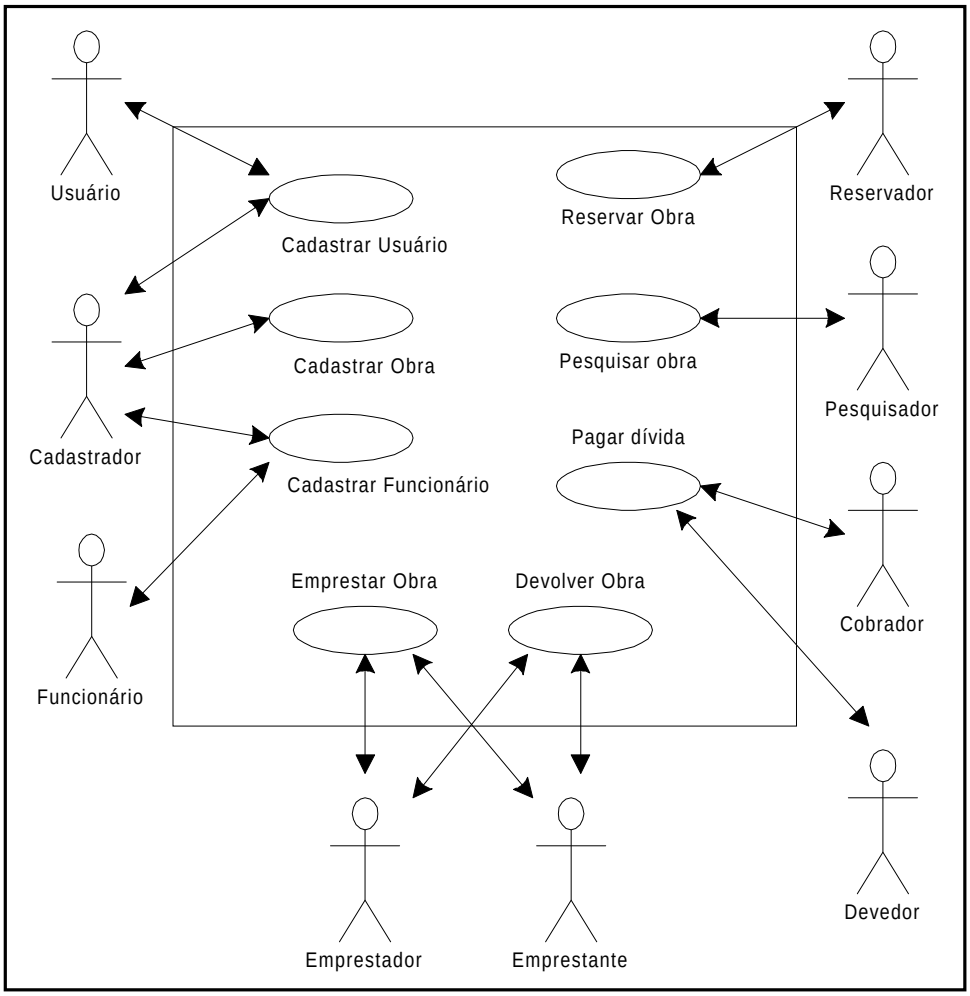



Figura C.1 - Modelo *use case* - delimitação do sistema

***Use Case: Emprestar obra***

O funcionário (emprestador) solicita empréstimo de obra ao sistema. O sistema solicita ao emprestador o número de cadastro do emprestante, para verificar se o mesmo está apto a fazer empréstimo. Esta verificação é feita observando se o emprestante já excedeu o limite máximo para empréstimo, se possui alguma dívida pendente e se está com alguma obra emprestada cuja data de devolução já esteja vencida.

Se o usuário estiver apto, o sistema solicita e o usuário fornece o código cadastral da obra. Verifica se o código cadastral é válido, e é verificado, também, se existe algum tombo daquela obra disponível, se existir, verifica se existe reserva para o tombo. Se existir reserva, observar se o reservante é o mesmo usuário que o emprestante, se for, o tombo é liberado, se não for, o tombo não pode ser liberado e toda verificação é feita para outro tombo. Se existir é solicitado ao emprestante sua senha, que é verificada na ficha do usuário.

Se a senha estiver correta, o sistema preenche a ficha de empréstimo e atualiza a ficha de controle do tombo, a ficha de usuário e a ficha de tombo. O emprestador libera a obra.

---

***Use Case: Reservar obra***

O reservador (funcionário ou usuário) solicita reserva ao sistema. O sistema solicita ao reservador o número de cadastro do usuário. É verificado se o usuário está apto a fazer reserva, verificando se não possui dívida e se não está com alguma obra emprestada cuja data de devolução já esteja vencida.

Se o usuário estiver apto, o sistema solicita e o usuário faz a entrada do código cadastral da obra e a data para reserva. Verifica-se se o código cadastral é válido; é verificado, também, se existe algum tombo daquela obra disponível na data e se não existe reserva na data para o usuário (para validar a inexistência de mais de uma reserva para o mesmo usuário na mesma data).

Se existir o tombo referido acima, o usuário entra com sua senha, que é verificada com a senha do cadastro do usuário.

Se a senha estiver correta, a reserva é efetuada, atualizando a ficha de reserva e o item de reserva.

Figura C.2 - Descrições dos *use cases* - curso básico

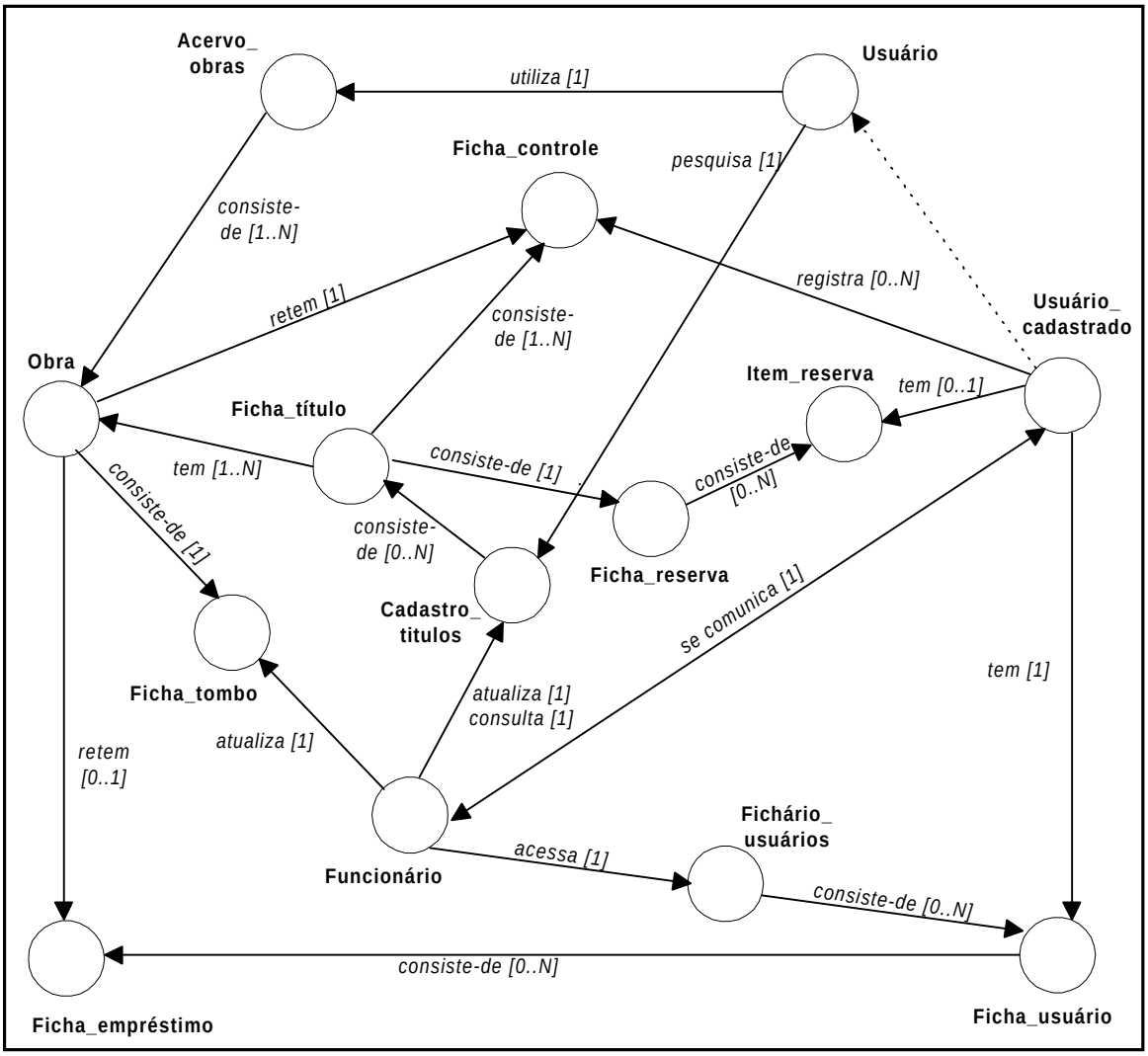
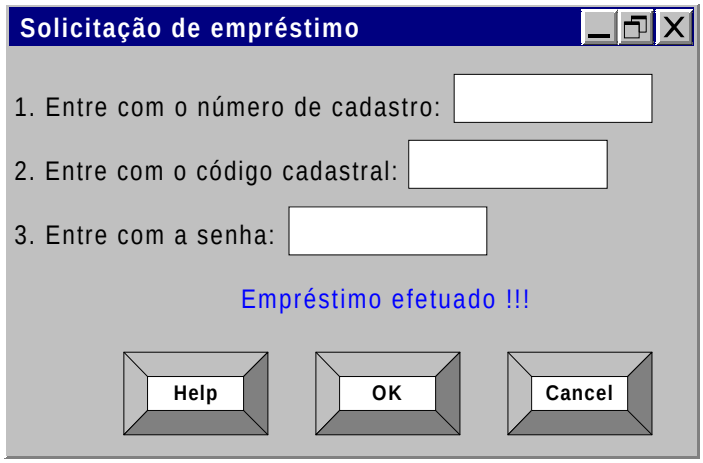



Figura C.3 - Modelo do domínio do problema




Quando o usuário solicitar empréstimo, a tela So licitação de empréstimo aparecerá, solicitando o número do cadastro do usuário. Então, serão feitas verificações internas referente ao usuário.

Se durante estas verificações for encontrado algum problema para o empréstimo, uma mensagem de erro será mostrada; se não, será solicitado o código cadastral da obra que se deseja fazer o empréstimo. Então, verificações internas referente a obra desejada serão feitas.

Se estas verificações não tiverem sucesso, uma mensagem de erro será mostrada; se não, a senha do usuário será solicitada. Então, a senha será verificada.

Se a senha estiver incorreta, aparecerá uma mensagem de erro; se não, a tela Empréstimo efetuado será apresentada.

O botão *Help* (ajuda) serve para explicar tudo que estiver na tela: dados, botões, processamento.

O botão OK serve i) para iniciar processamentos internos de verificações referentes aos dados informados pelo usuário e deve ser clicado após o dado solicitado ter sido informado; ou ii) para finalizar o empréstimo.

O botão *Cancel* (cancela) serva para cancelar o empréstimo.

Figura C.4 - Modelo de interface

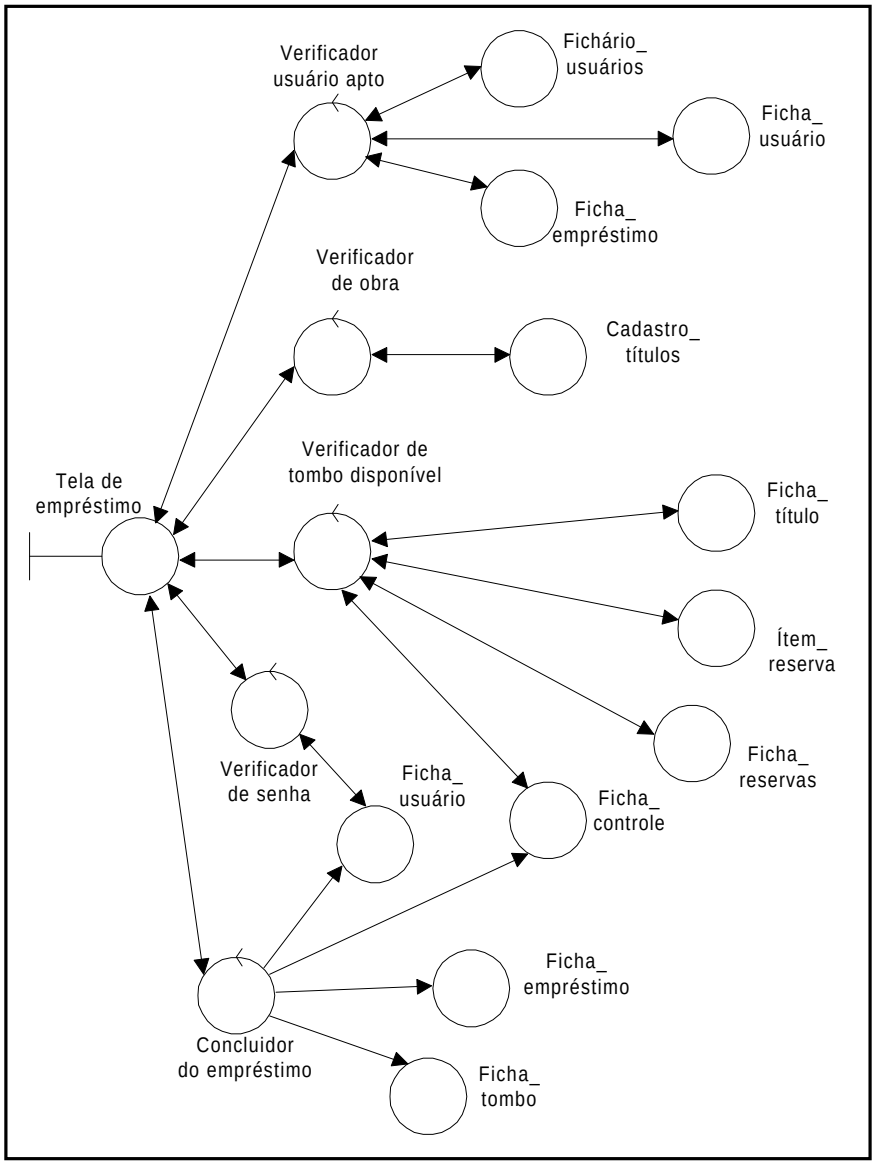



Figura C.5 - Modelo de análise - *use case*: emprestar obra

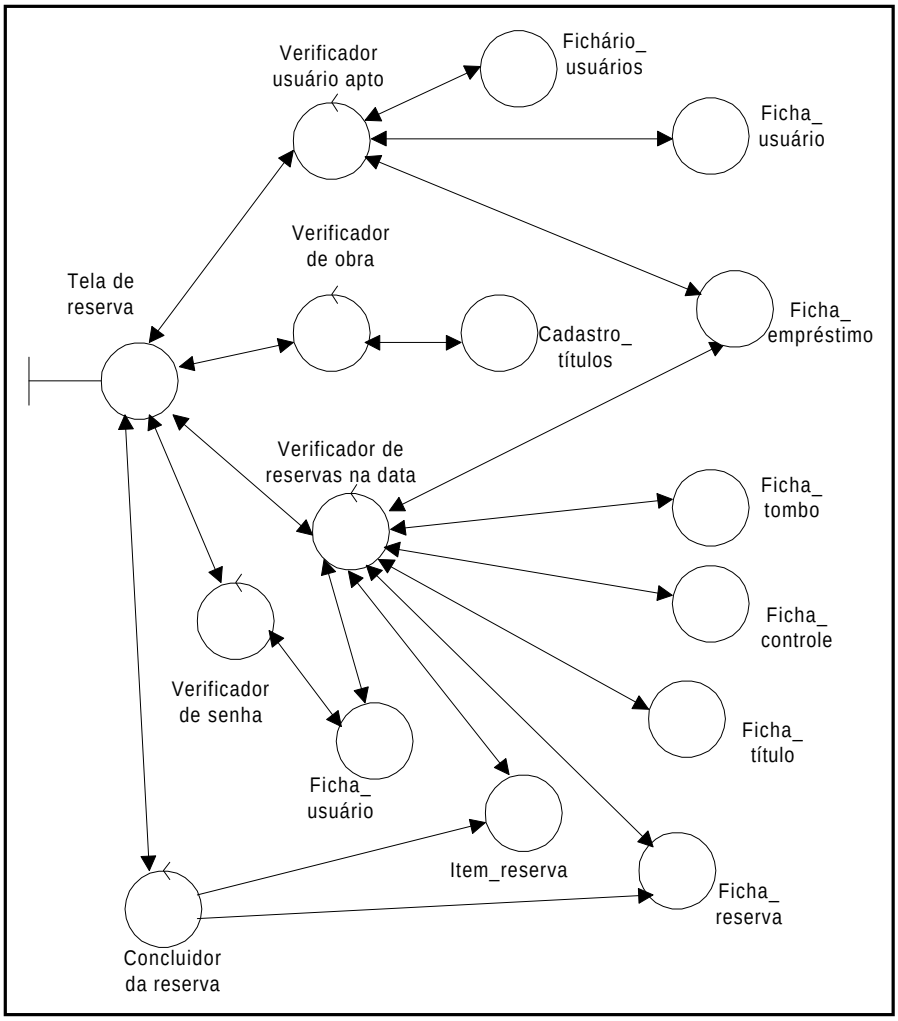



Figura C.6 - Modelo de análise - *use case*: reservar obra

***Objeto de interface: Tela de reserva***

1o. Pede-se o código do usuário e aciona o objeto verificador de usuário apto.
    Se não estiver apto: mensagem de erro.
2o. Pede-se o código cadastral da obra e aciona o objeto verificador de tombo.
    Se não existir: mensagem de erro.
3o. Pede-se a data para reserva e aciona o objeto verificador de reservas na data.
    Se não existir espaço para mais reservas: mensagem de erro.
4o. Pede-se a senha do usuário e aciona o objeto verificador de senha.
    Se senha não estiver correta: mensagem de erro.
5o. Aciona o objeto concluidor da reserva.

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

***Objeto de controle: Verificador de usuário apto***

1o. Localiza-se o código do usuário no objeto fichário_usuários.
    Se não localizado: mensagem de erro.
2o. Verifica a dívida_total e cota_empréstimos do objeto ficha_usuário.
    Se possui dívidas ou cota_emprestada ultrapassou cota_empréstimo:
    mensagem de erro.
3o. Verifica dt_devolução do objeto ficha_empréstimo.
    Se possui empréstimo vencido: mensagem de erro.

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

***Objeto de controle: Concluidor de empréstimo***

1o. Adiciona 1 a cota_emprestada no objeto ficha_usuário.
2o. Altera o status_obra no objeto ficha_controle.
3o. Altera objeto ficha_empréstimo.
4o. Altera dt_devolução no objeto ficha_tombo.

Figura C.7 - Descrições do papel e das responsabilidades dos objetos

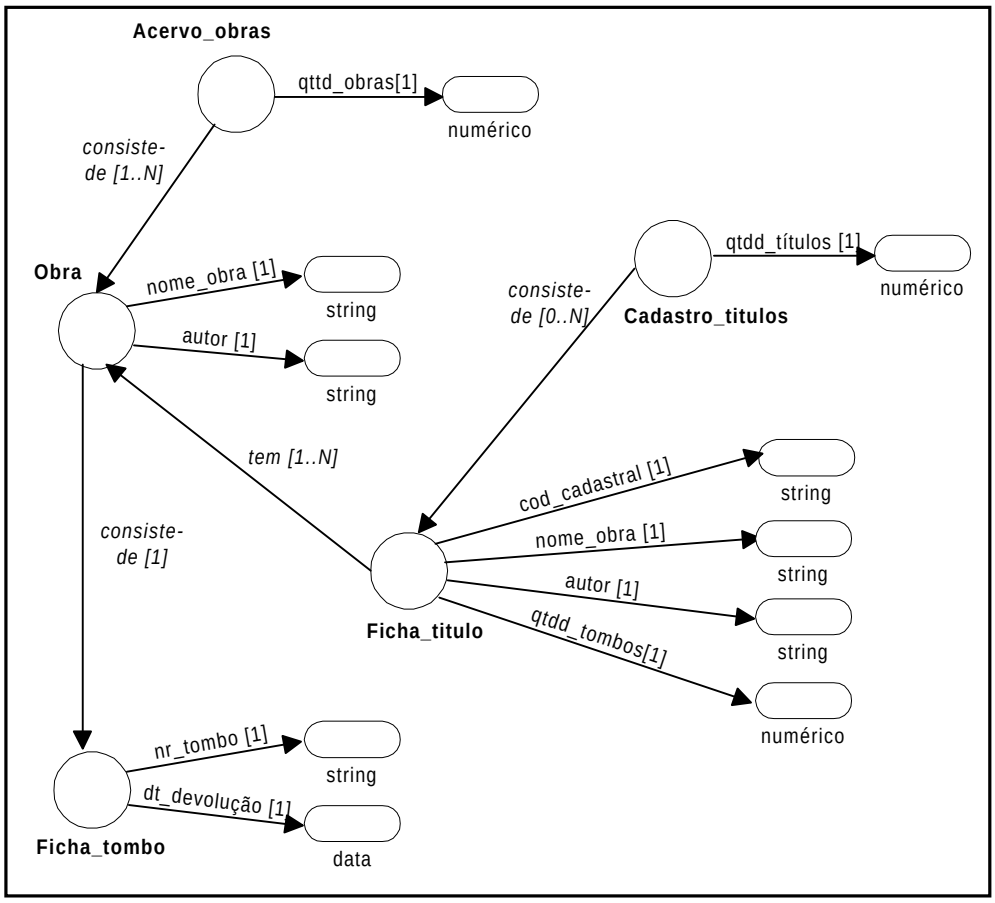



Figura C.8 - Representação dos atributos dos objetos entidade

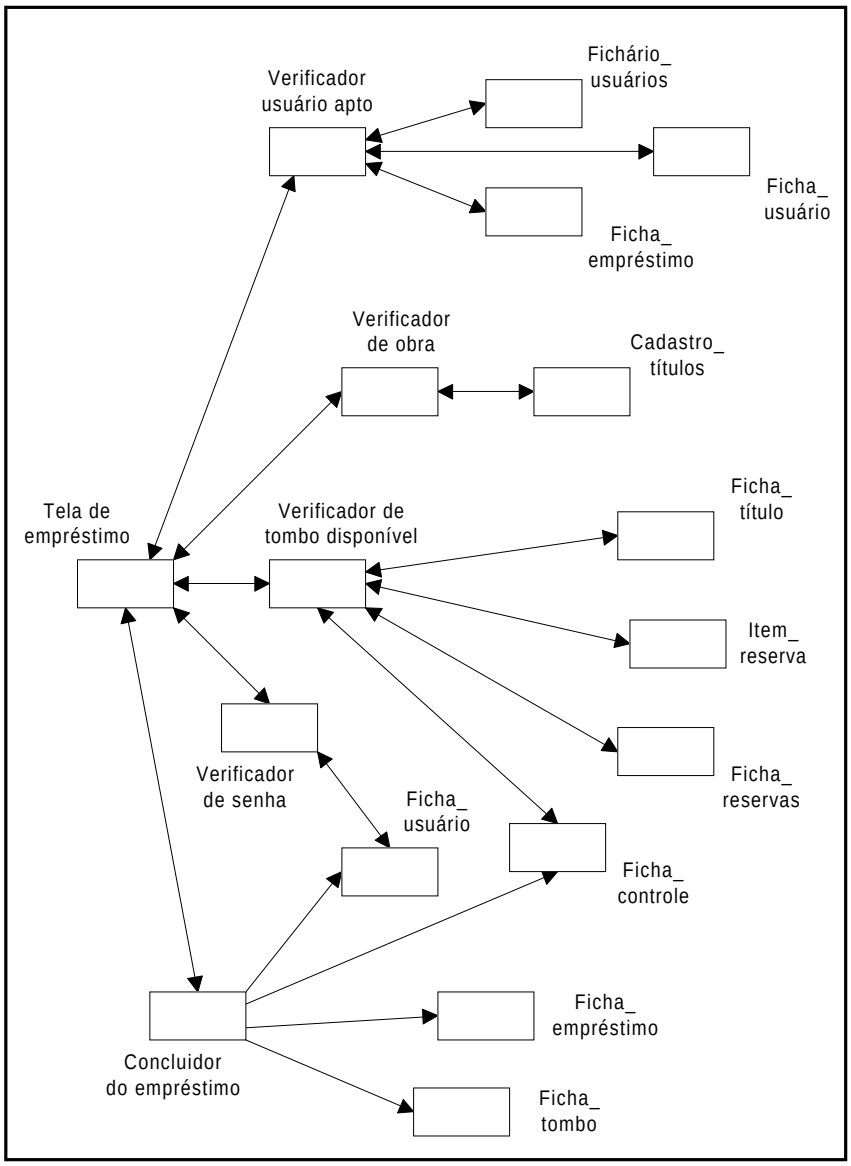



Figura C.9 - Modelo de projeto - *use case*: emprestar obra

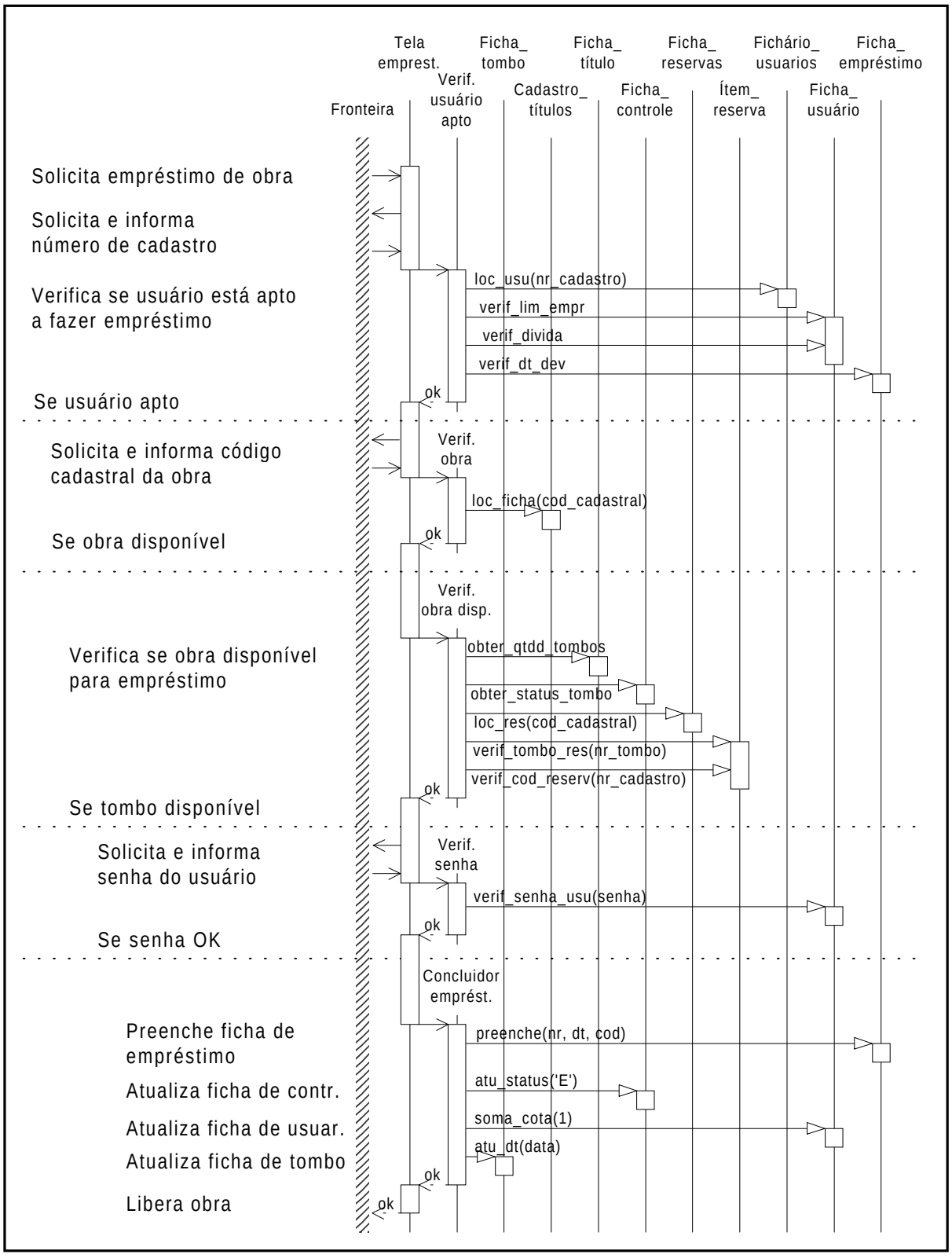



Figura C.10 - Diagrama de interação - *use case*: emprestar obra (curso básico)

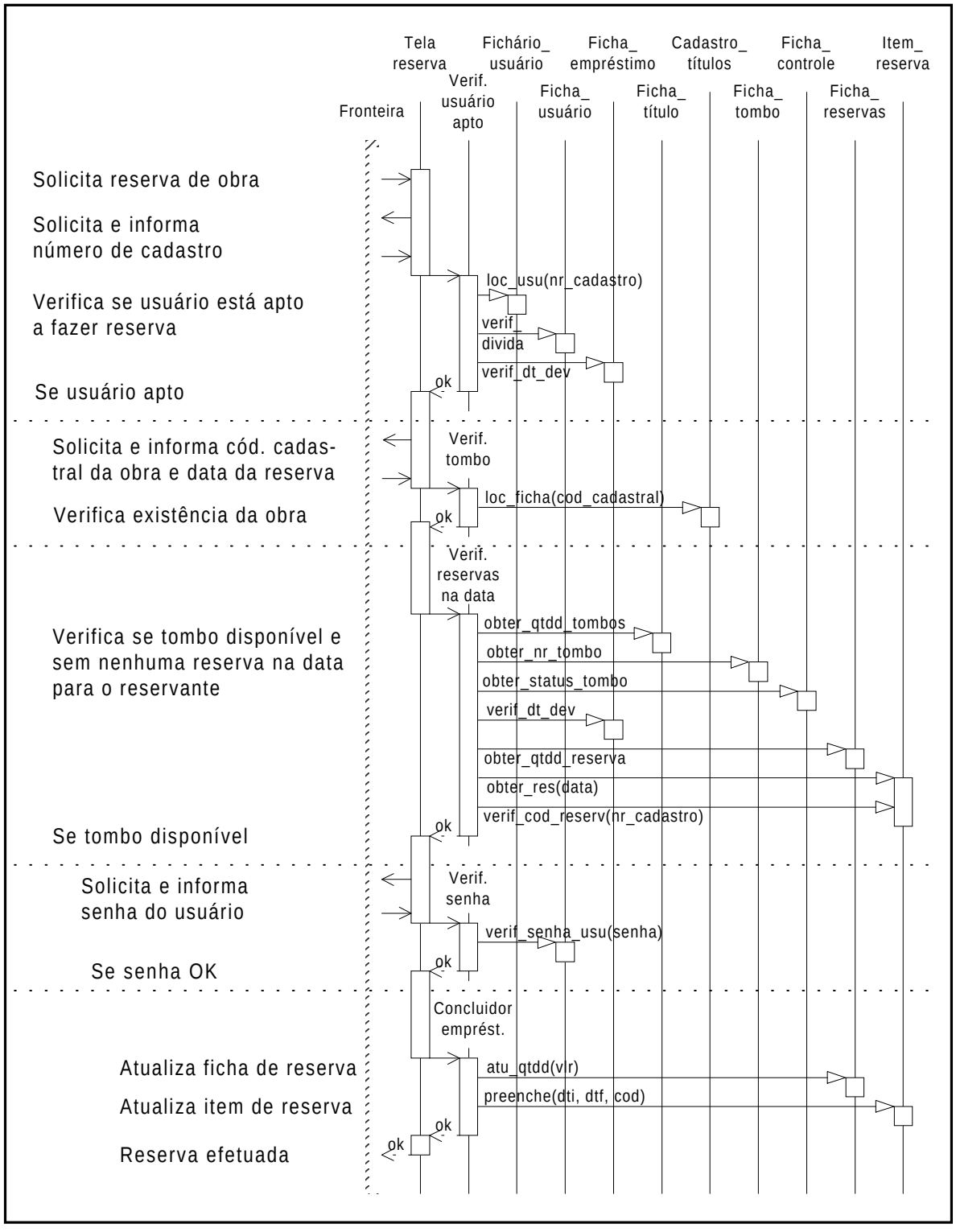



Figura C.11 - Diagrama de interação - *use case*: reservar obra (curso básico)

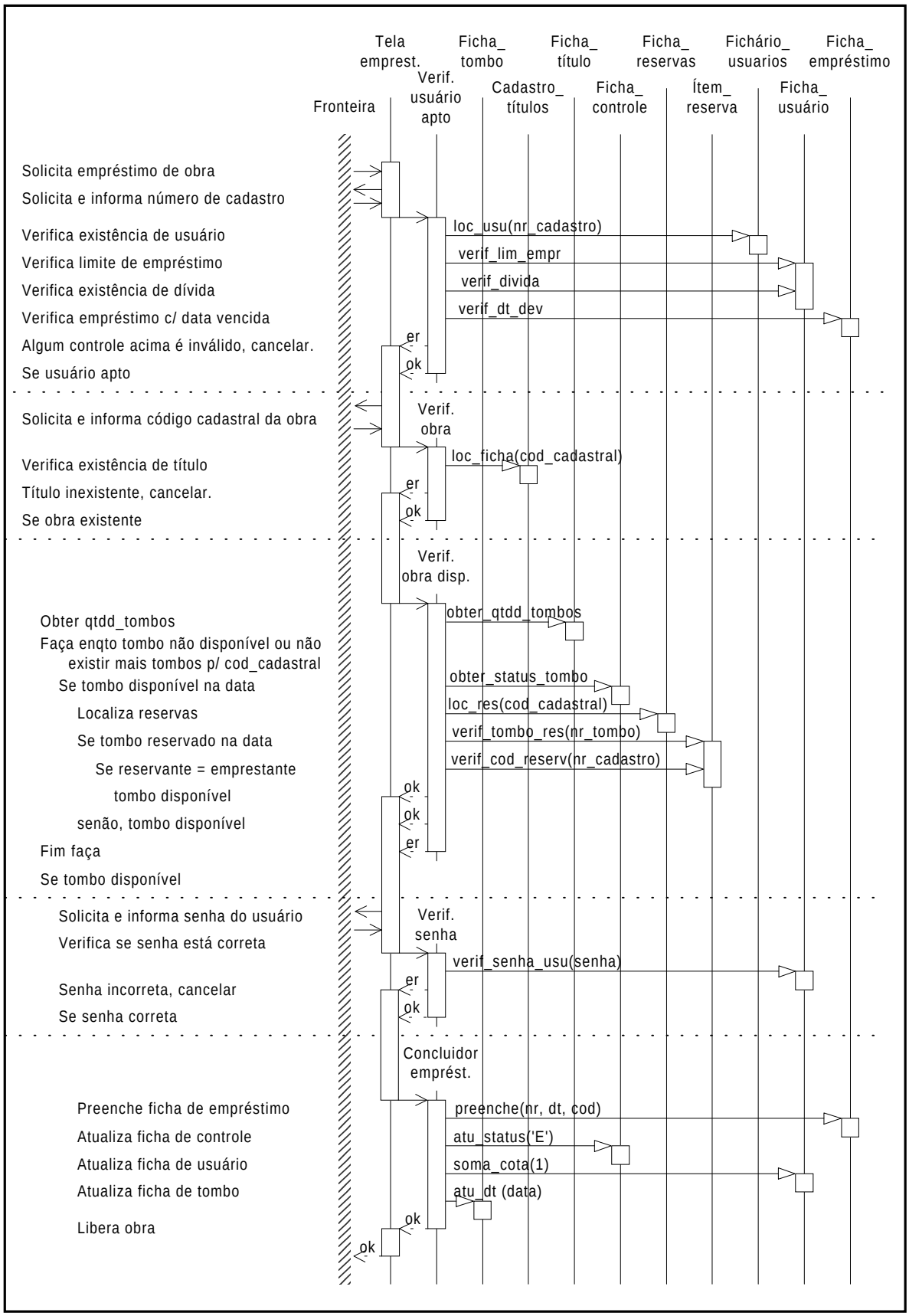



Figura C.12 - Diagrama de interação - *use case*: emprestar obra (curso alternativo)

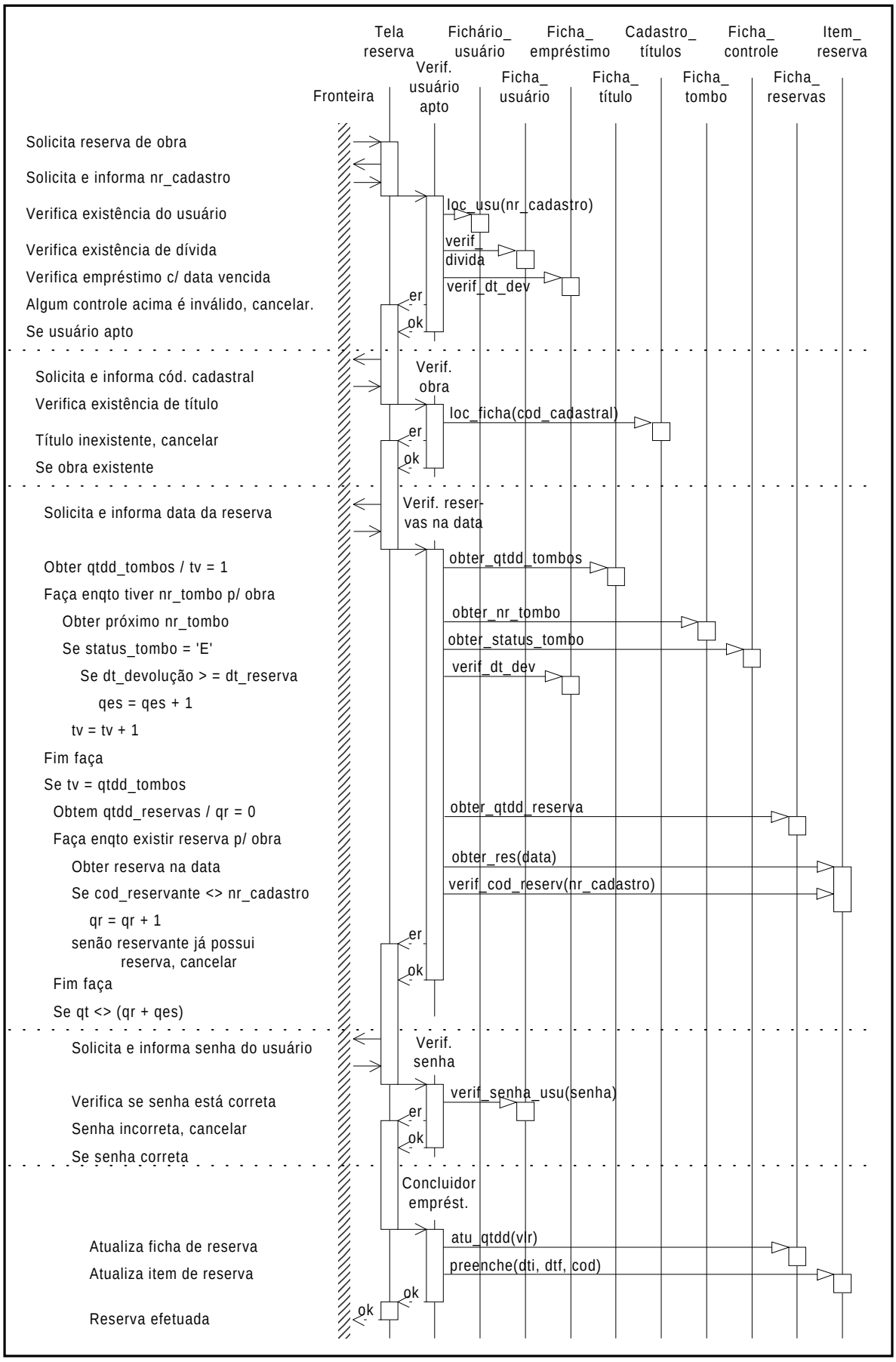



Figura C.13 - Diagrama de interação - *use case*: reservar obra (curso alternativo)

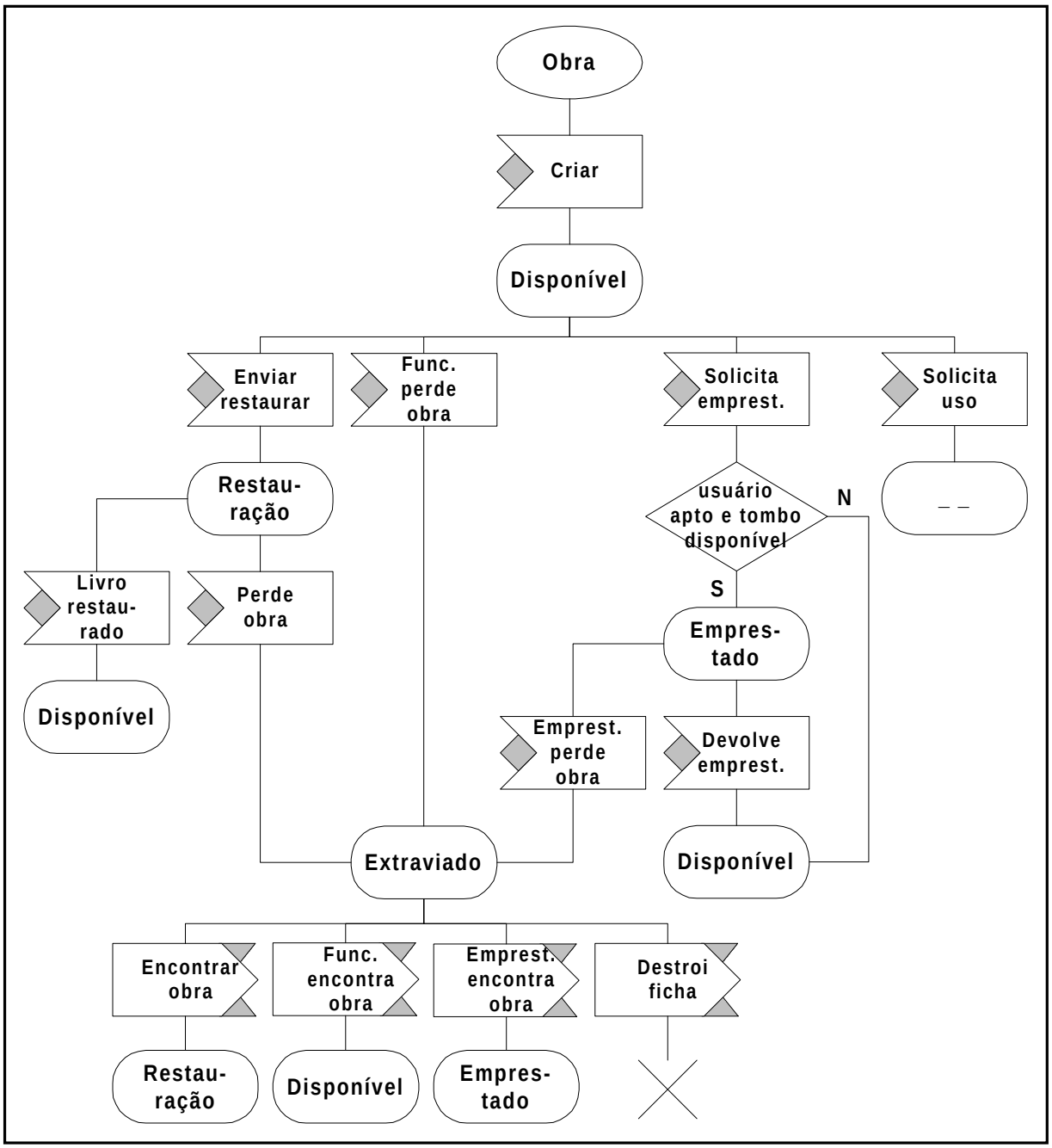



Figura C.14 - Grafo de transição de estado - objeto ficha_controle

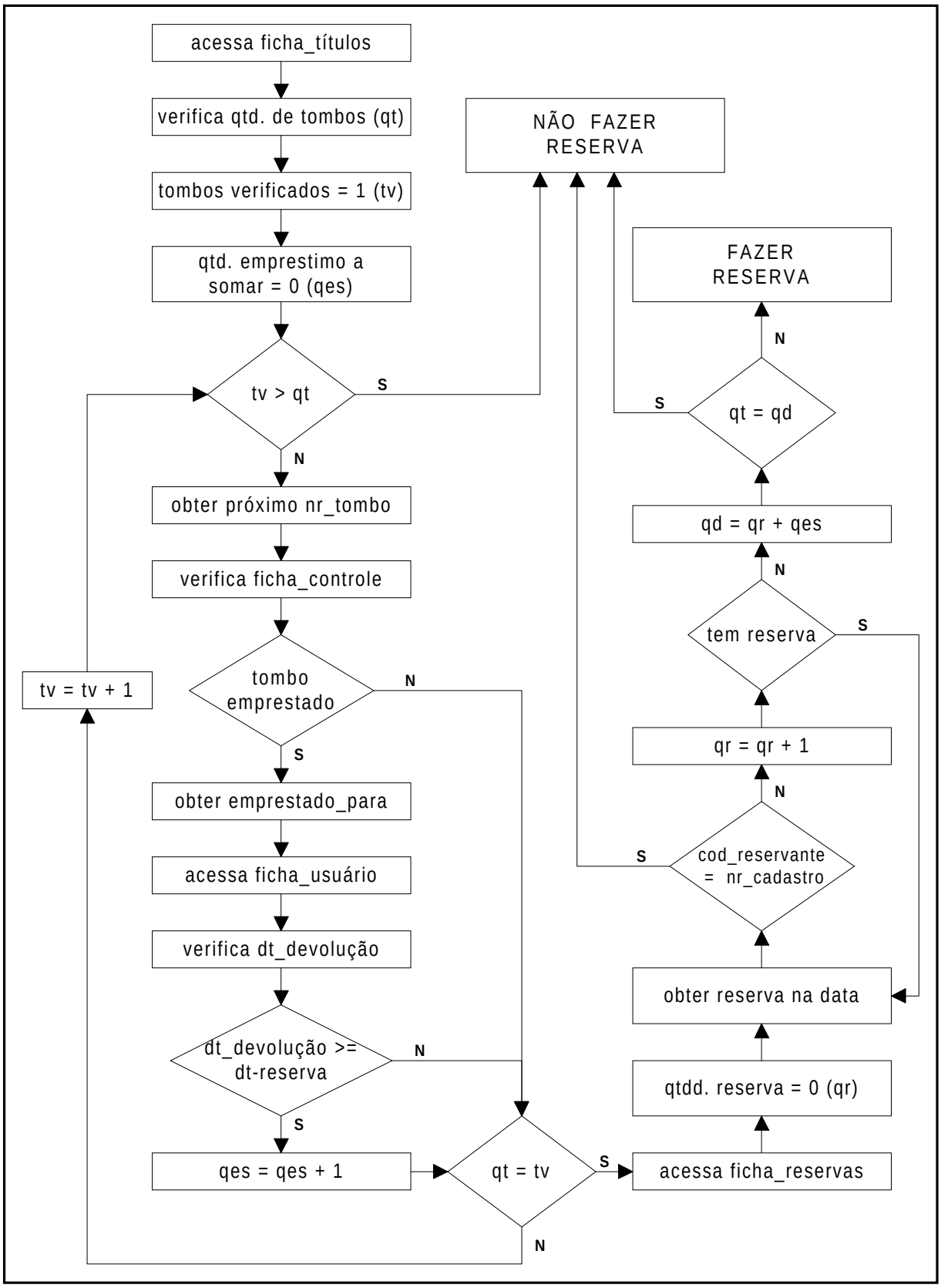



Figura C.15 - Fluxograma da reserva

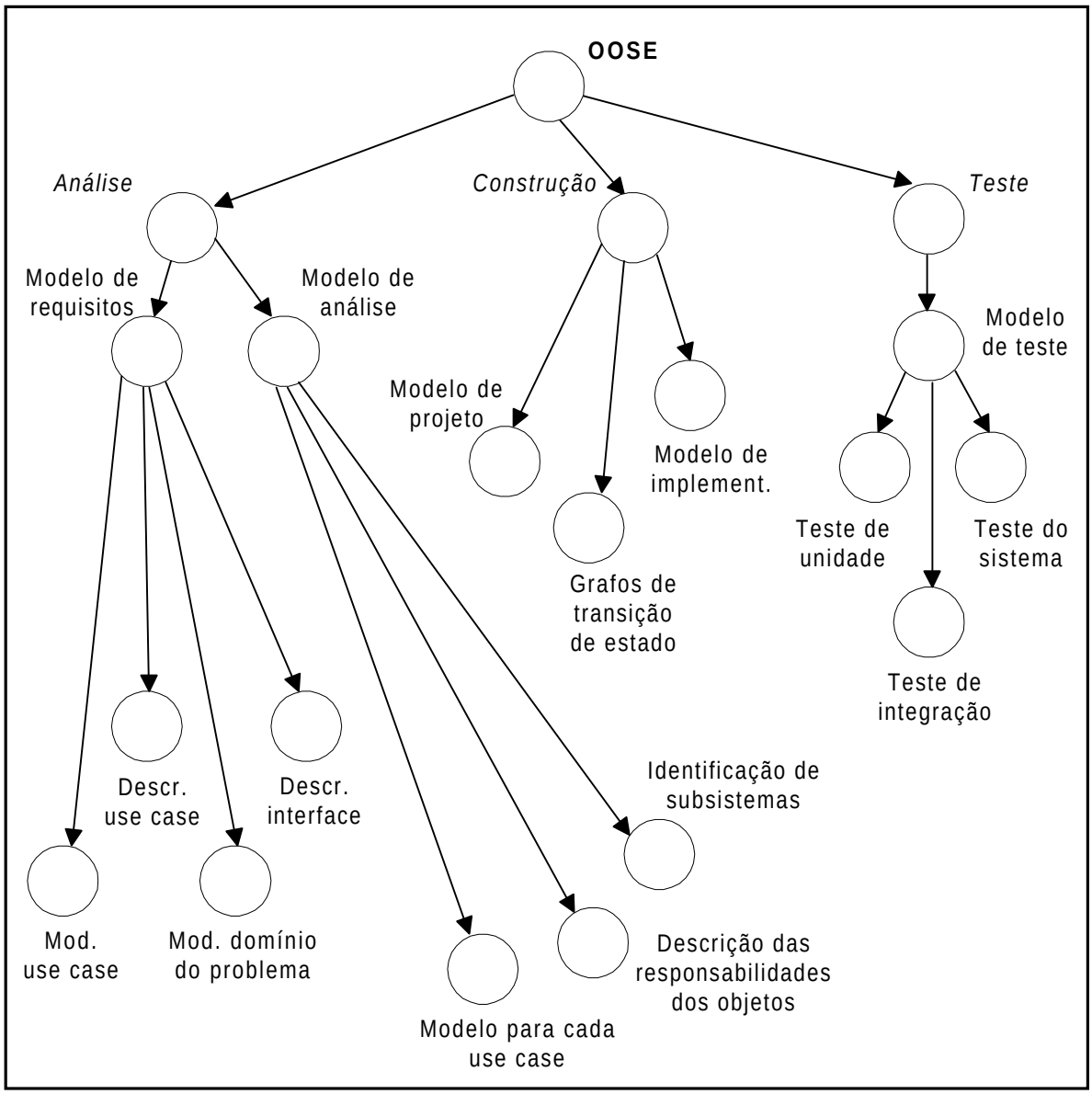



Figura C.16 - Resumo gráfico dos modelos/diagramas gerados pelo OOSE

## 10.2 Modelagem em OMT

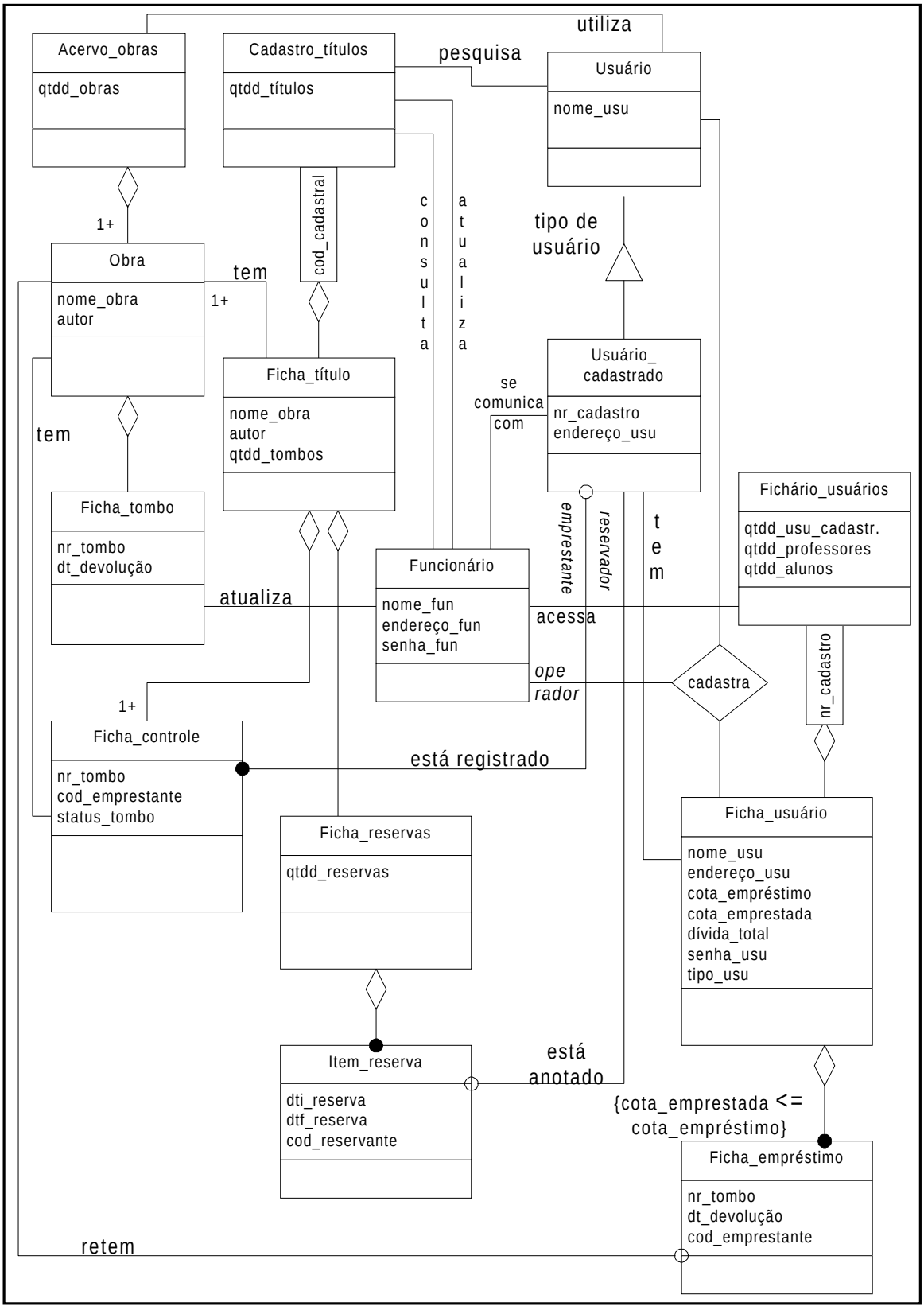



Figura C.17 - Diagrama de classes - análise

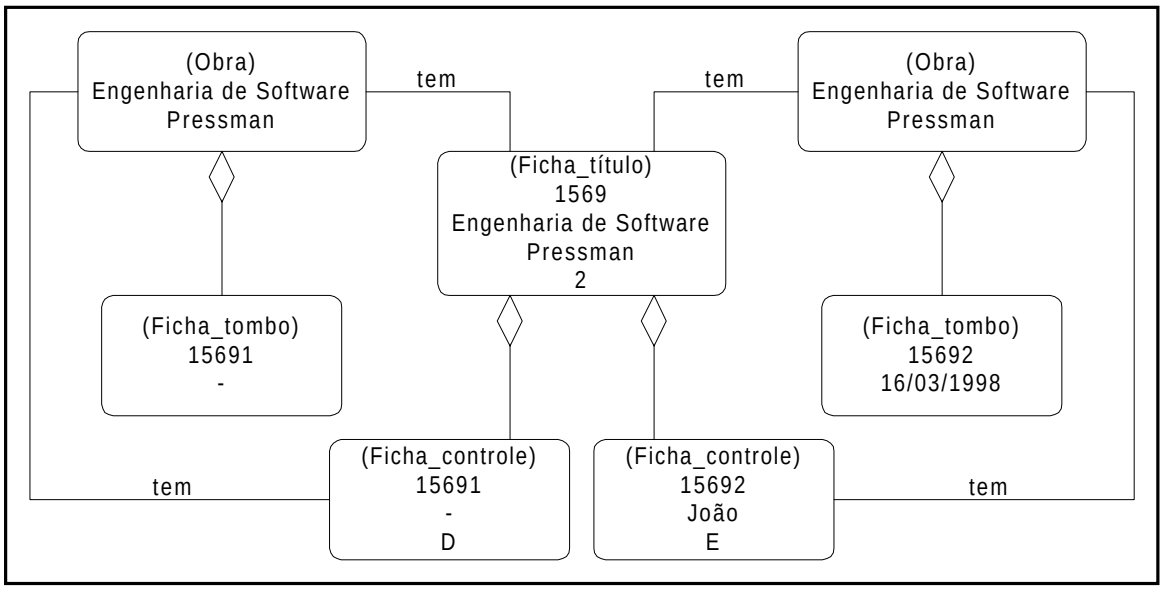



Figura C.18 - Diagrama de instâncias


Figura C.19 - Diagrama de interface

**Classes**
- ACERVO_OBRAS: possui informações totalizadoras das OBRAS.
- CADASTRO_TÍTULOS: contém informações totalizadoras sobre FICHA_TÍTULO.
- FICHA_CONTROLE: cada tombo possui a sua. Armazena o status de cada tombo (D - Disponível, E - Emprestado, R - Restauração) e código do emprestante.
- FICHA_EMPRÉSTIMO: armazena informações sobre empréstimo de um determinado usuário. Esta ficha ou estará junto a OBRA ou junto a FICHA_USUÁRIO, quando o tombo estiver emprestado.
- FICHA_RESERVAS: possui informações totalizadoras de reservas e vários ITEM_RESERVA por obra.
- FICHA_TÍTULO: armazena informações sobre os títulos das obras, possui várias FICHA_CONTROLE e várias FICHA_RESERVAS e se relaciona com FICHA_TOMBO. Poderá existir mais de uma OBRA para cada FICHA_TÍTULO.
- FICHA_TOMBO: fixado a cada tombo, possui data de devolução. Funcionário atualiza esta ficha quando o empréstimo é efetuado.
- FICHA_USUÁRIO: contém informações sobre o usuário, tais como: dívida, cota máxima para empréstimo, cota emprestada, senha e tipo de usuário (P - Professor, A - Aluno). Possui várias FICHA_EMPRÉSTIMO, quando o usuário possuir algum empréstimo. Só é permitido o empréstimo se a cota emprestada for menor ou igual a cota de empréstimo.
- FICHÁRIO_USUÁRIOS: contém informações totalizadoras sobre FICHA_USUÁRIO, tais como quantidade de usuário, quantidade de professores e quantidade de alunos.
- FUNCIONÁRIO: armazena informações gerais dos funcionários (operadores, administradores e pessoal de suporte). Os funcionários atualizam os dados, gerenciam o sistema e controlam problemas de mal funcionamento.
- ITEM_RESERVA: armazena informações sobre as reservas, como: data inicial, data final e código do reservante.
- OBRA: armazena dados gerais sobre as obras. Cada OBRA contém uma FICHA_TOMBO, que fica colada a ela, tem associadas a ela, uma FICHA_CONTROLE e uma FICHA_TÍTULO. Também possui uma FICHA_EMPRÉSTIMO, quando a obra não está emprestada. É a obra física, o tombo.
- USUÁRIO: possui informações sobre usuários, que não estão cadastrados. São usuários que simplesmente utilizam ou fazem pesquisa ao ACERVO.
- USUÁRIO_CADASTRADO: possui informações sobre usuários cadastrados no sistema e que podem reservar, emprestar, devolver. Cada usuário cadastrado tem uma FICHA_USUÁRIO. Ele pode estar anotado como emprestante em FICHA_CONTROLE ou como reservante em ITEM_RESERVA.

**Atributos**
autor: nome do autor da obra.
cod_cadastral: código cadastral da obra. Obras com o mesmo título, possuem o mesmo cod_cadastral.
dt_devolução: data da devolução da obra.
nome_usu: nome do usuário.
nr_cadastro: número do cadastro do usuário.
nr_tombo: número do tombo. Obras com o mesmo título, possuem nr_tombo diferentes.
qtdd_obras: quantidade de obras no acervo.
senha_usu: senha do usuário.

**Operações**
obter_status_tombo( ): verifica o status do tombo e o retorna.
verif_cod_reserv(nr_cadastro): com o nr_cadastro do usuário é verificado se ele é igual ao cod_reservante para saber se o reservante já possui alguma reserva.
loc_usu(nr_cadastro): localiza o usuário no fichário do usuários através do nr_cadastro para saber se ele está cadastrado como usuário cadastrado.

**Associações**
está anotado: um usuário pode estar anotado em um item de reserva, quando o usuário faz uma reserva.
pesquisa: um usuário pode fazer uma pesquisa no cadastro de títulos.
utiliza: um usuário pode utilizar o acervo de obras.
atualiza: o funcionário atualiza a ficha de tombo, quando um empréstimo é efetuado.

Figura C.20 - Dicionário de dados

· Funcionário solicita cadastro.
· SB solicita escolha do cadastro; funcionário escolhe o cadastro.
· SB solicita dados para cadastro; funcionário / usuário informa dados para cadastro.
· SB solicita senha do funcionário / usuário; funcionário / usuário informa senha.
· SB registra o cadastro.

· Funcionário solicita relatórios.
· SB solicita a escolha do relatório; funcionário escolhe o relatório.
· SB solicita datas inicial e final; funcionário informa datas.
· SB emite o relatório.

· Funcionário solicita empréstimo.
· SB solicita nr_cadastro; usuário_cadastrado informa nr_cadastro.
· SB solicita cod_cadastral da obra; o usuário_cadastrado informa cod_cadastral da obra.
· SB solicita senha do usuário; usuário_cadastrado informa senha.
· SB libera obra.

· Usuário_cadastrado / Funcionário solicita reserva.
· SB solicita data para reserva; o usuário_cadastrado informa data para reserva.
· SB informa que reserva foi efetuada.

Figura C.21 - Cenário normal do sistema de biblioteca

· Funcionário solicita cadastro.
· SB solicita escolha do cadastro; funcionário escolhe o cadastro.
· SB solicita dados para cadastro; funcionário / usuário informa dados para cadastro.
· SB solicita senha do funcionário / usuário; funcionário / usuário informa senha.
· senha inválida.
· SB registra o cadastro.

· Funcionário solicita relatórios.
· SB solicita a escolha do relatório; funcionário escolhe o relatório.
· SB solicita datas inicial e final; funcionário informa datas.
· Data inválida.
· SB emite o relatório.

· Funcionário solicita empréstimo.
· SB solicita nr_cadastro; usuário_cadastrado informa nr_cadastro.
· nr_cadastro é inválido.
· usuário não está permitido a fazer empréstimo.
· SB solicita cod_cadastral da obra; o usuário_cadastrado informa cod_cadastral da obra.
· cod_cadastral é inválido.
· tombo não disponível.
· SB solicita senha do usuário; usuário_cadastrado informa senha.
· senha é inválida.
· SB libera obra.

· Usuário_cadastrado / Funcionário solicita reserva.
· SB solicita data para reserva; o usuário_cadastrado informa data para reserva.
· tombo não disponível na data.
· reservante já possui reserva.
· SB informa que reserva foi efetuada.

Figura C.22 - Cenário especial do sistema de biblioteca

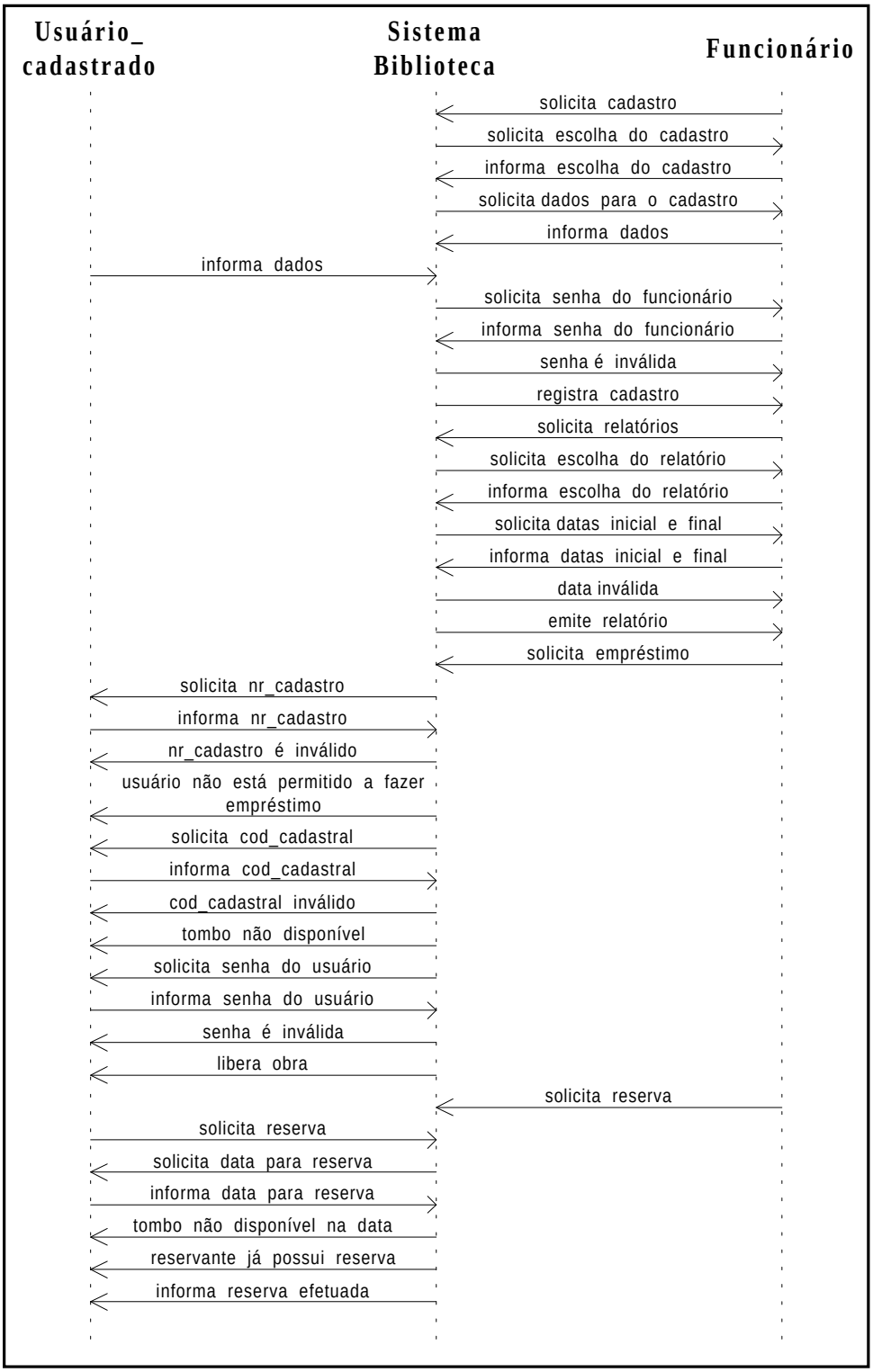



Figura C.23 - Diagrama de eventos para o cenário do sistema de biblioteca

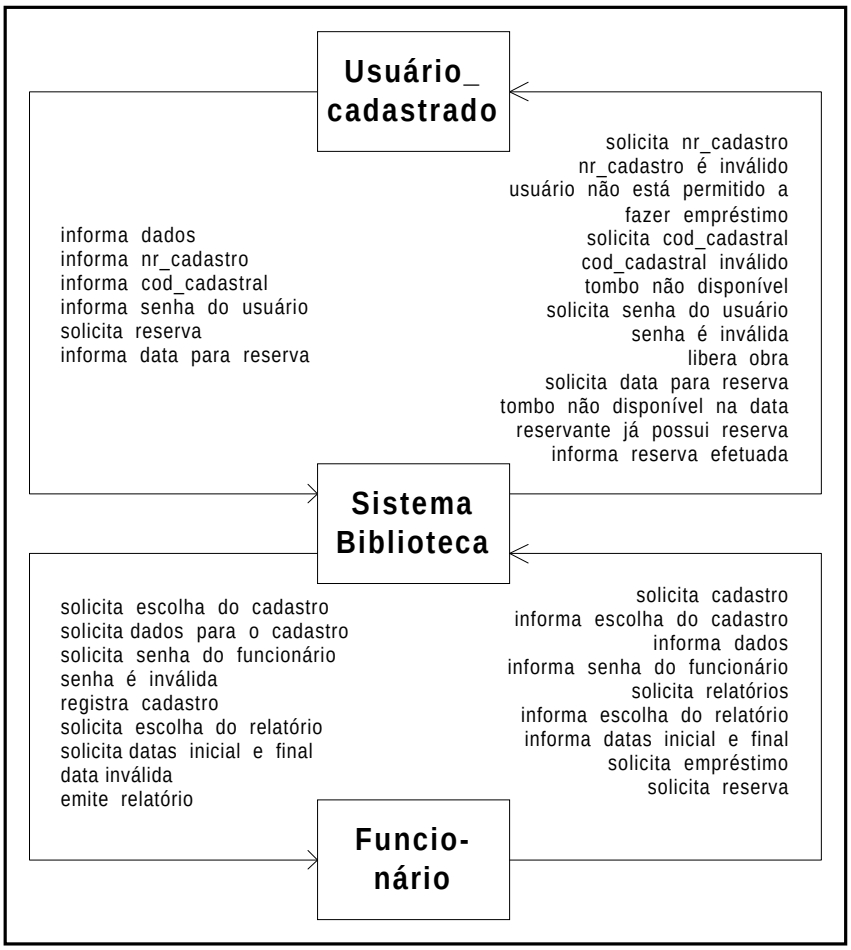



Figura C.24 - Diagrama de fluxo de eventos do sistema de biblioteca

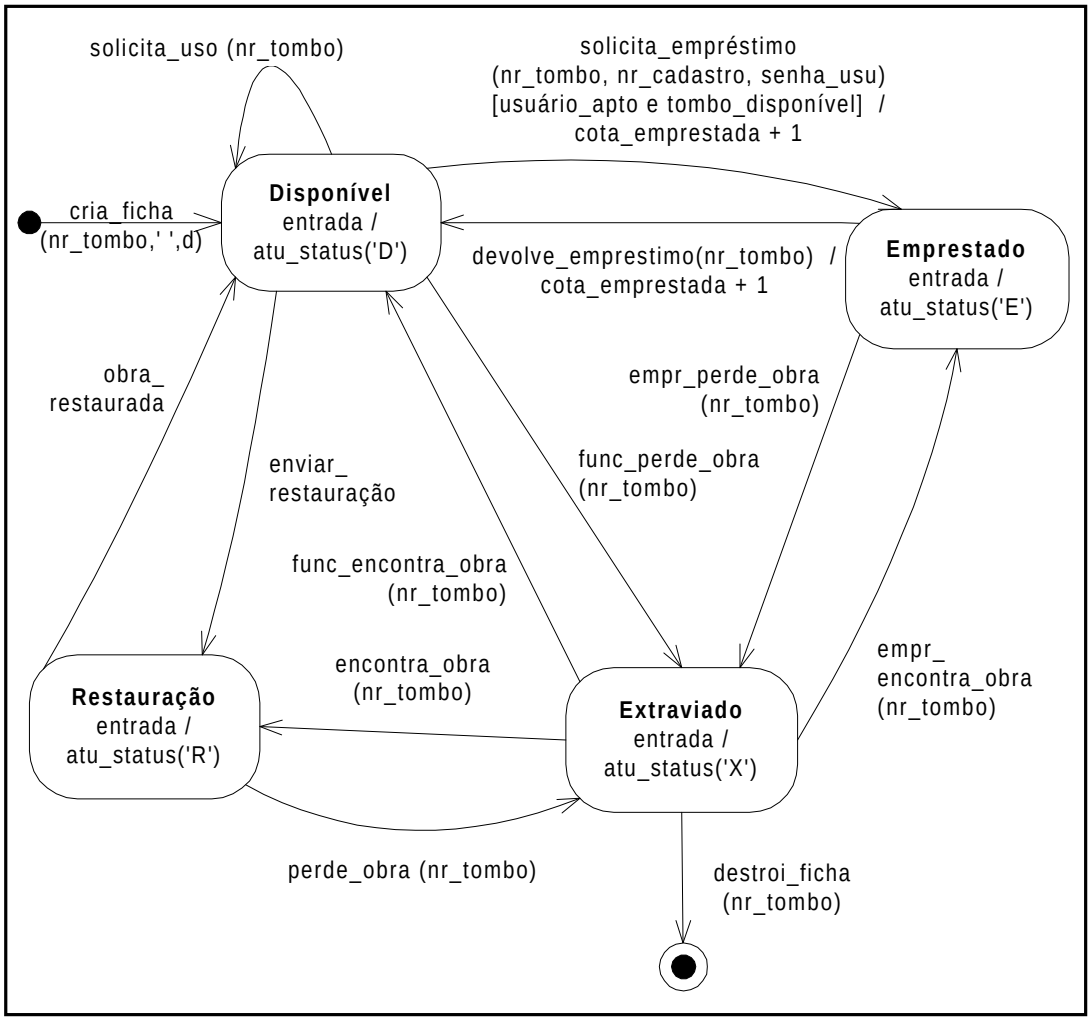



Figura C.25 - Diagrama de estado - classe: obra

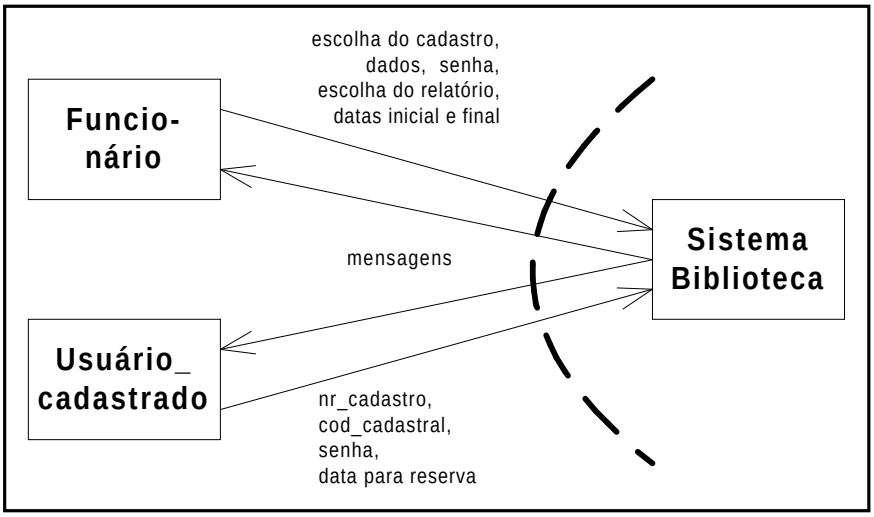



Figura C.26 - Diagrama de valores de entrada e saída

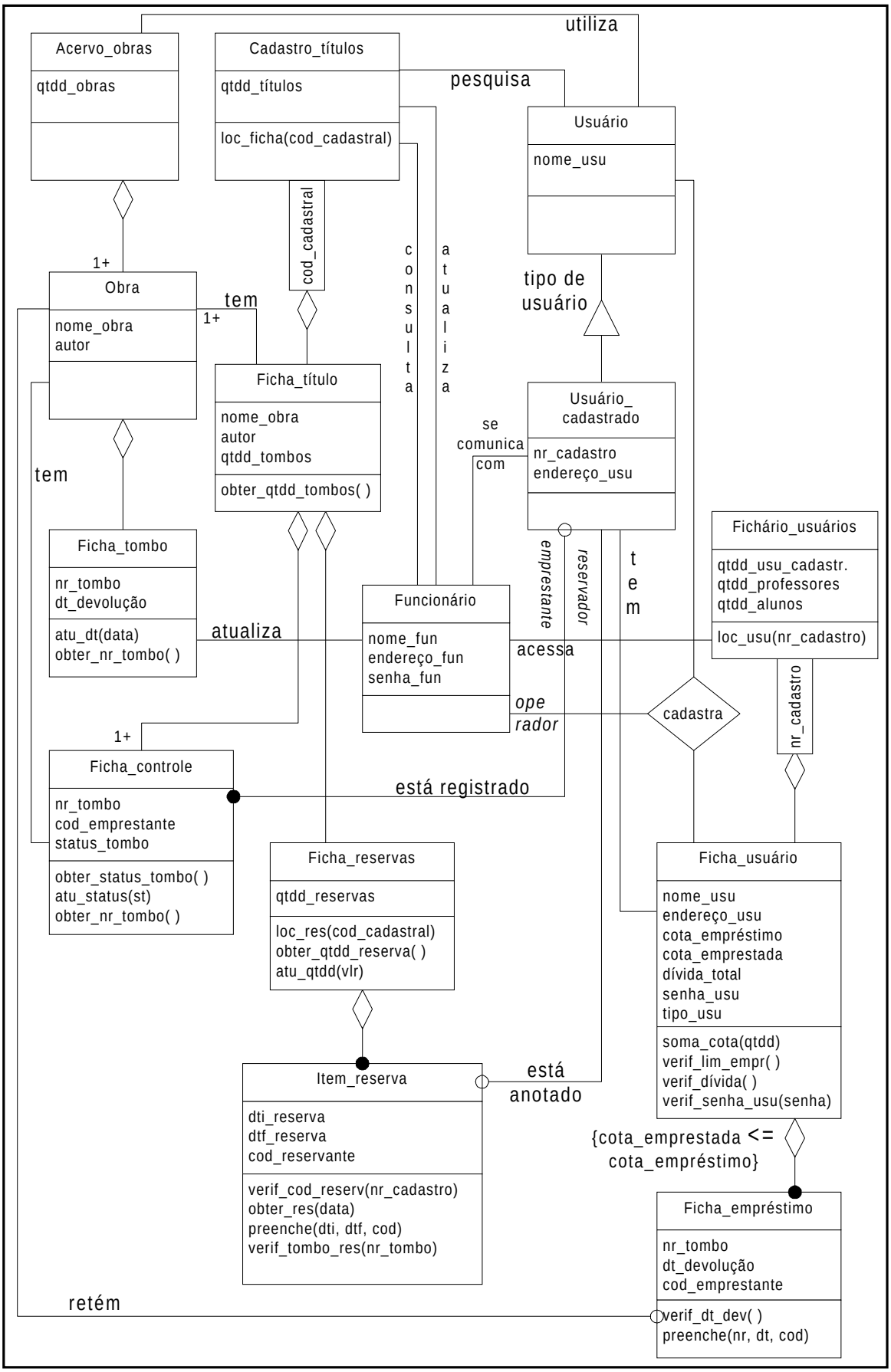



Figura C.27 - Diagrama de classes - projeto

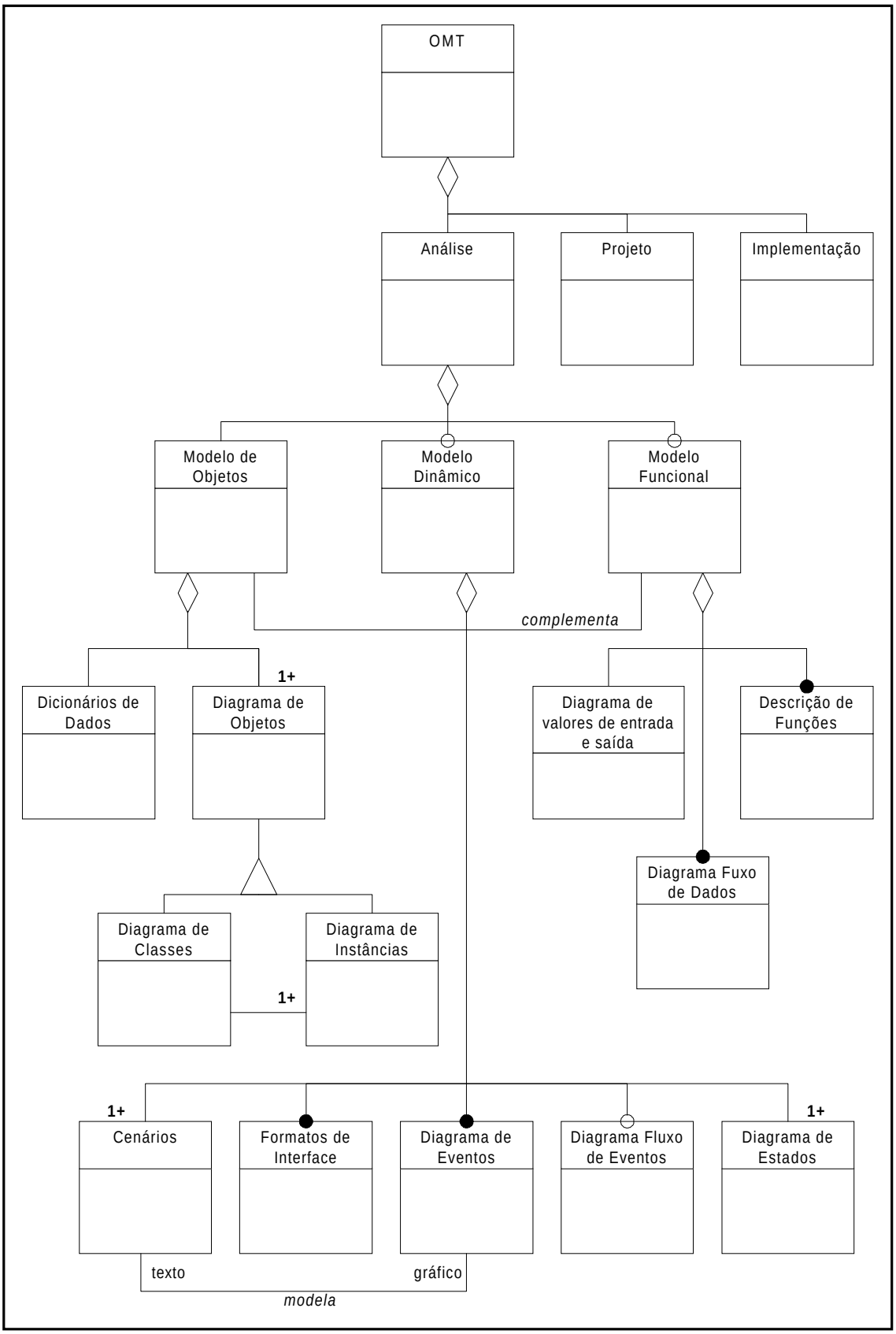



Figura C.28 - Resumo gráfico dos modelos/diagramas gerados pelo OMT - análise

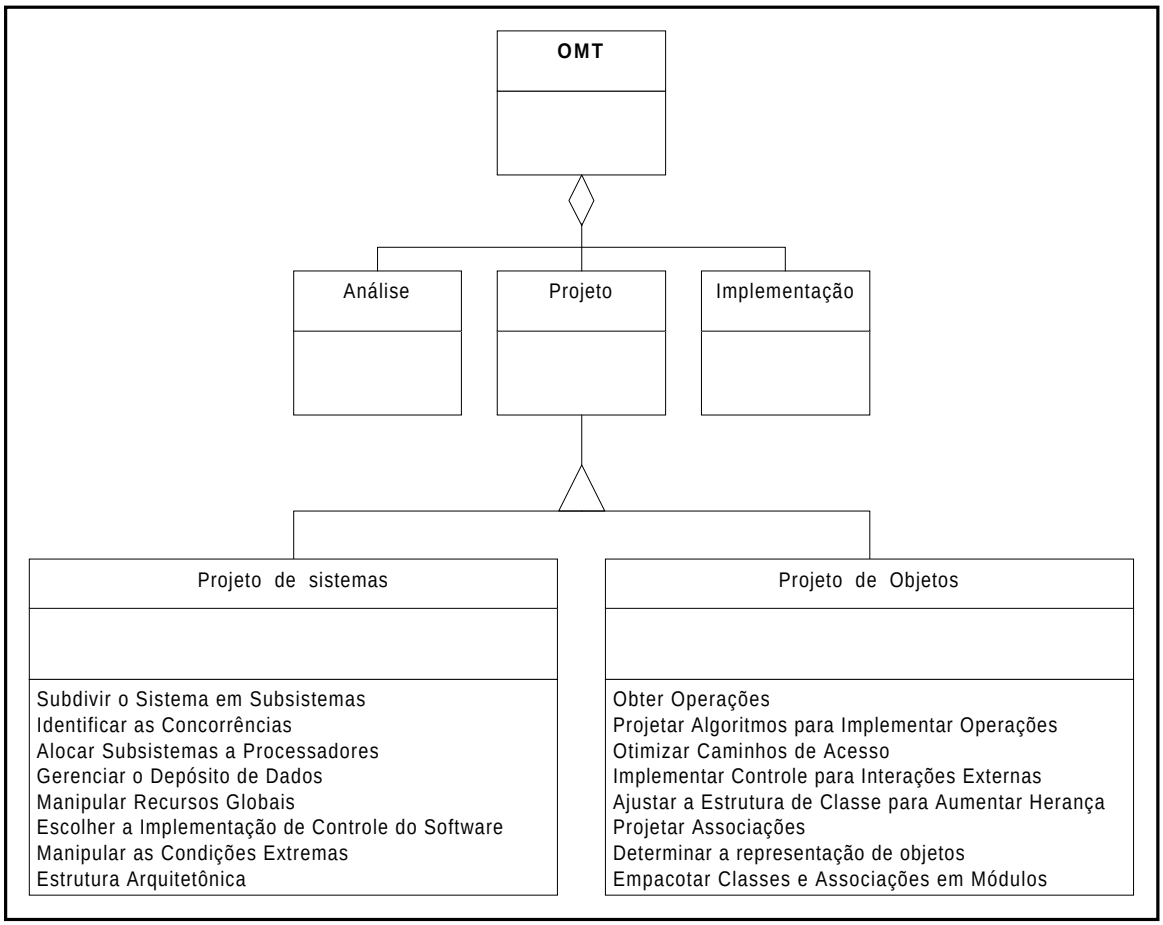



Figura C.29 - Resumo gráfico dos modelos/diagramas gerados pelo OMT - projeto

## *10.3 Modelagem em OOAD*

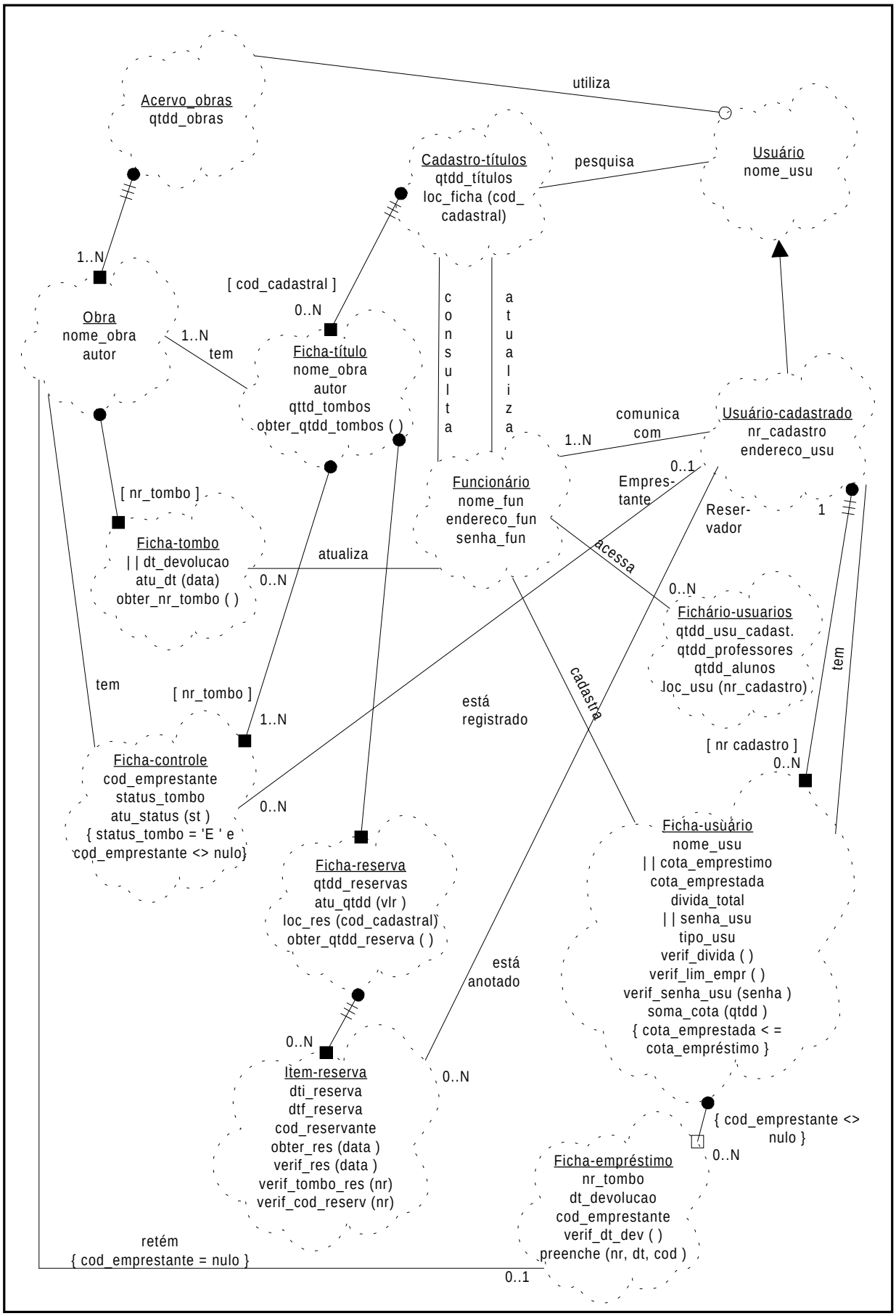



Figura C.30 - Diagrama de classes

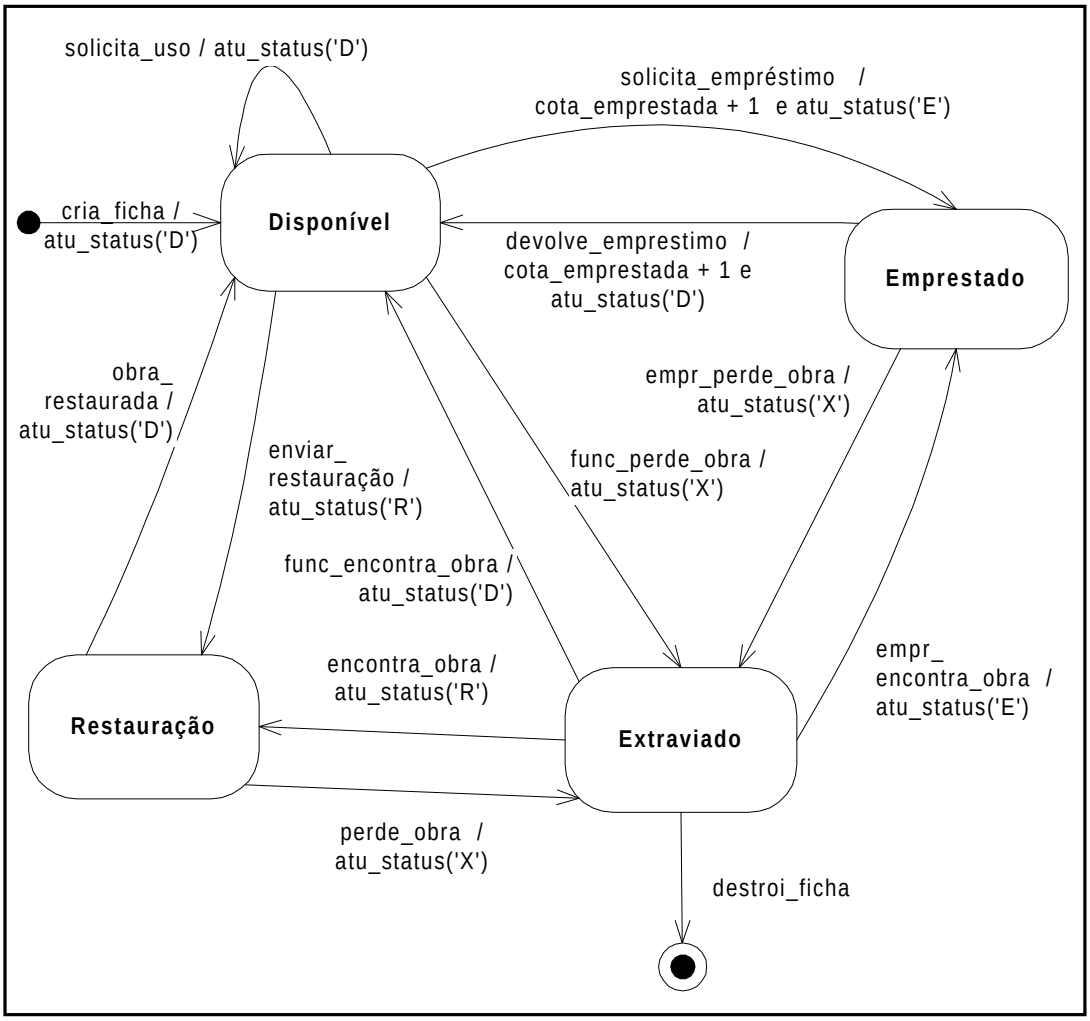



Figura C.31 - Diagrama de transição de estado de obra

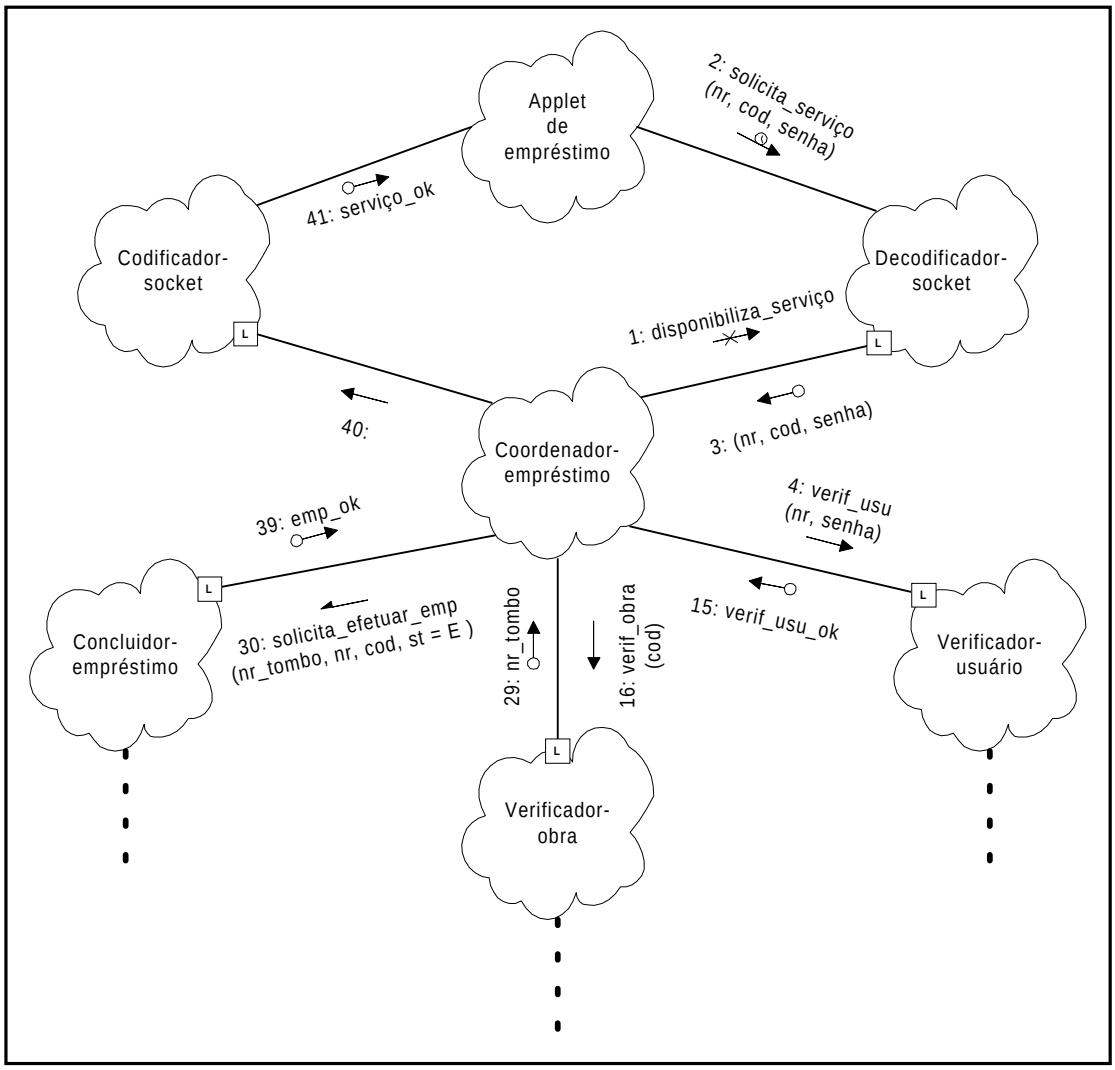



Figura C.32 - Diagrama de objetos

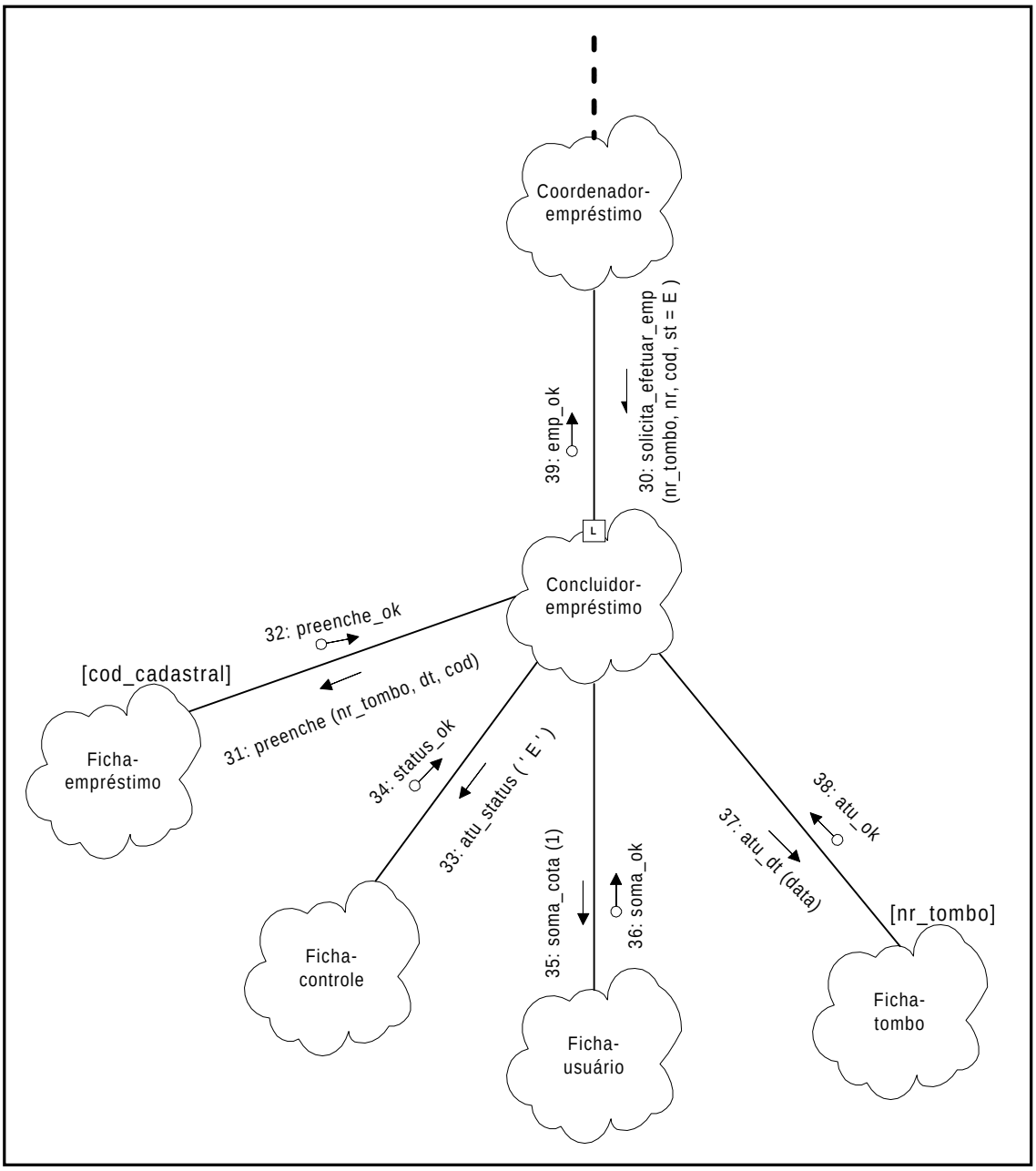



Figura C.33 - Diagrama de objetos - cont.

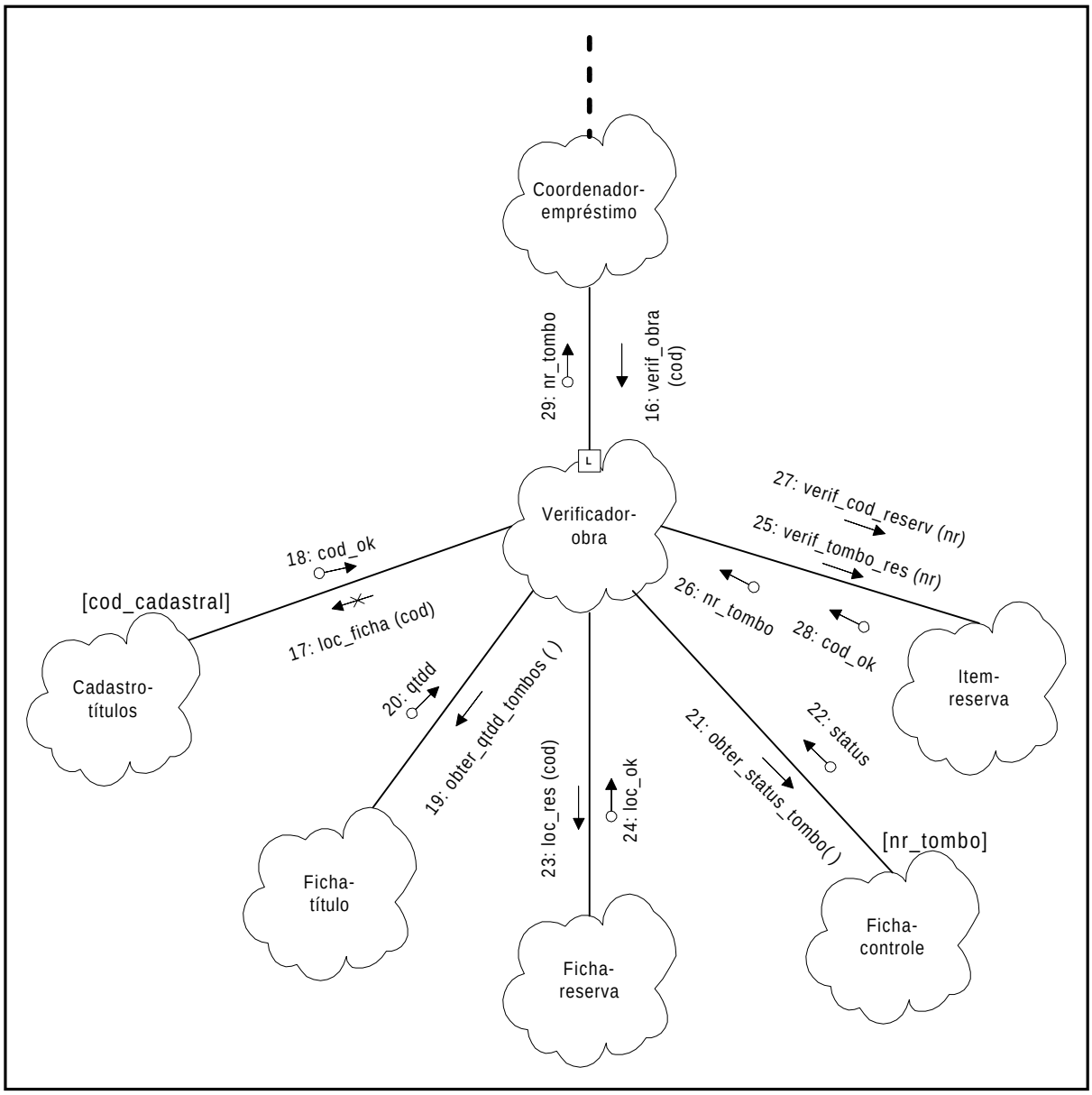



Figura C.34 - Diagrama de objetos - cont.

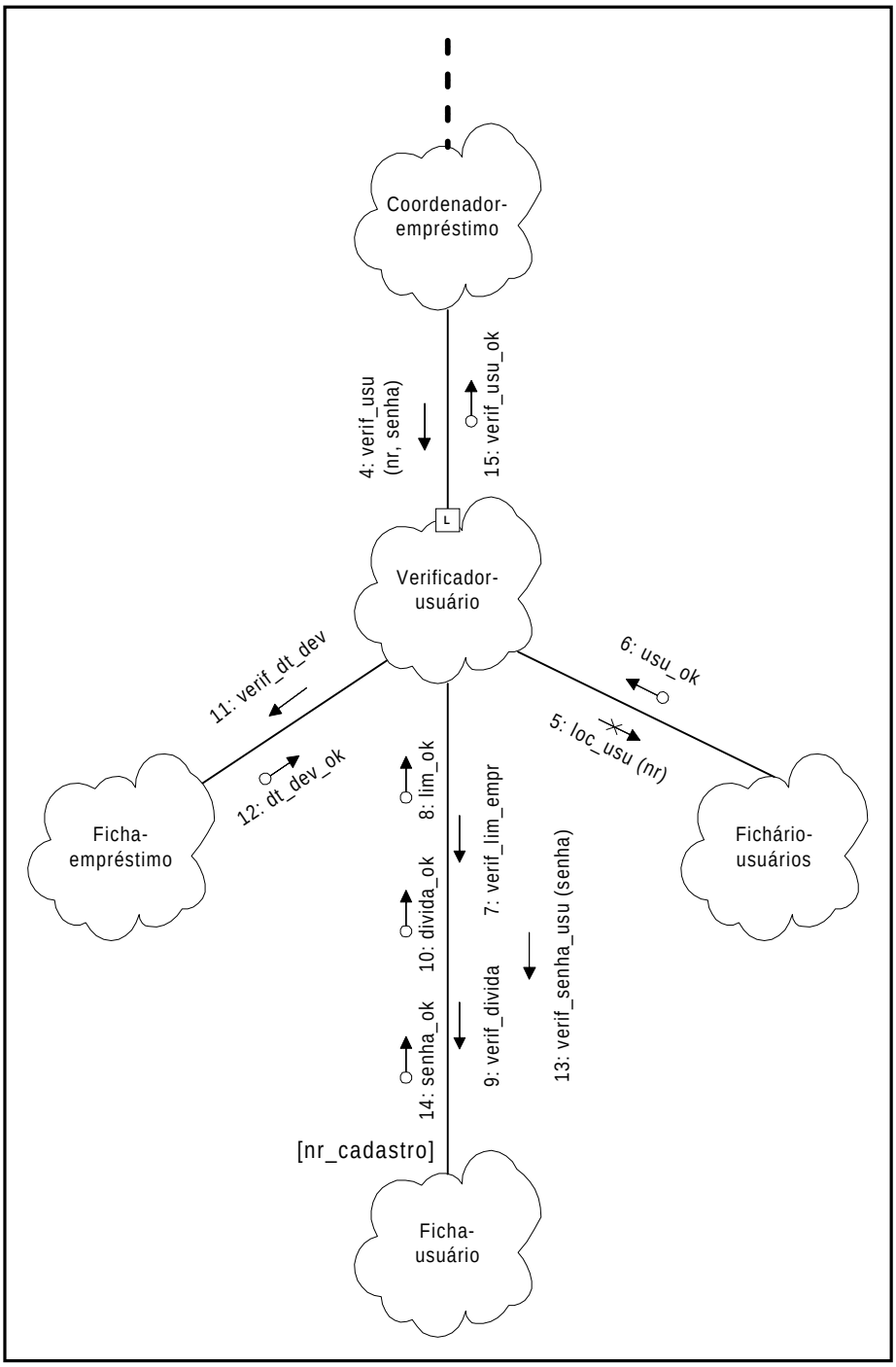



Figura C.35 - Diagrama de objetos - cont.

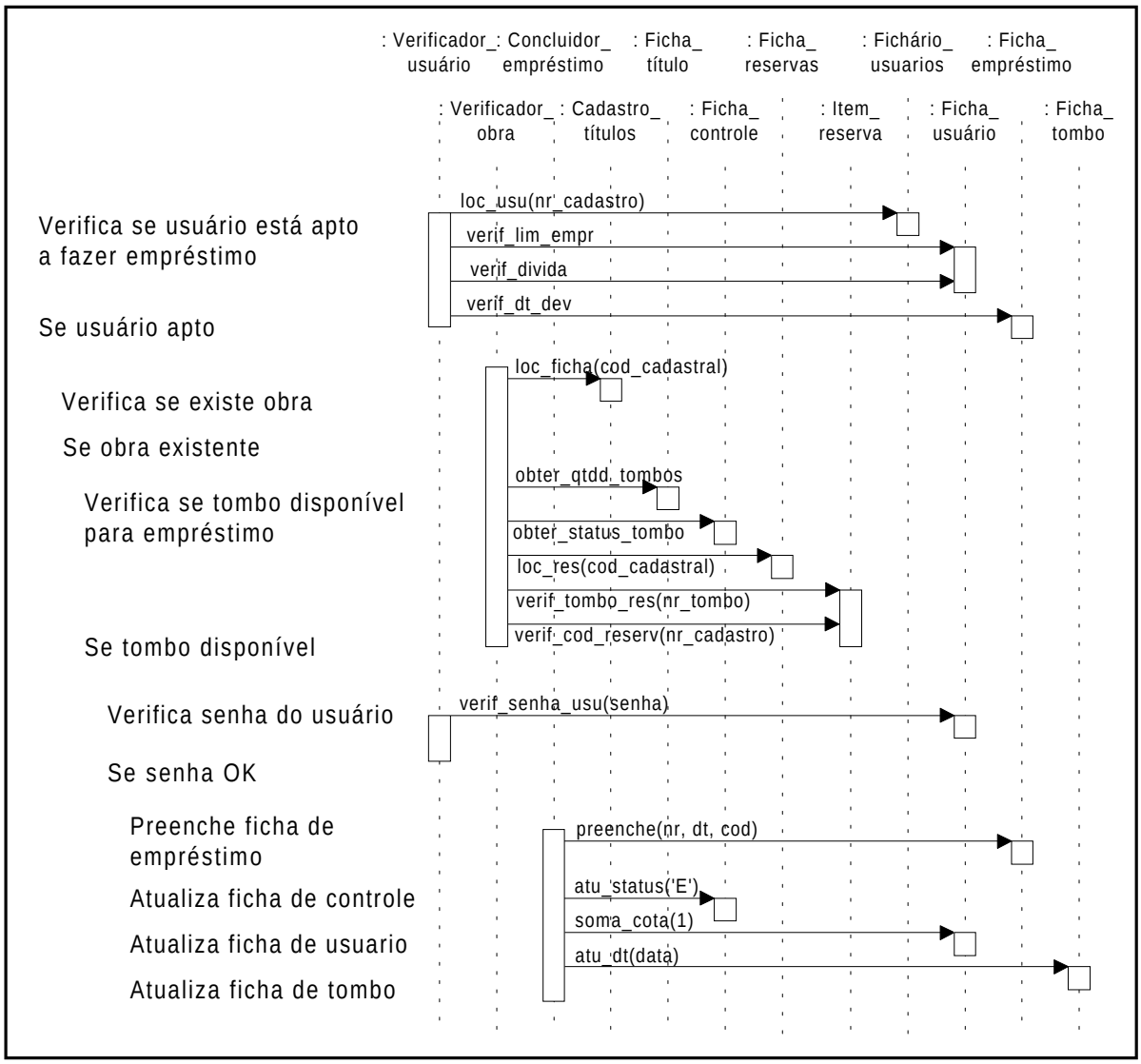



Figura C.36 - Diagrama de interação

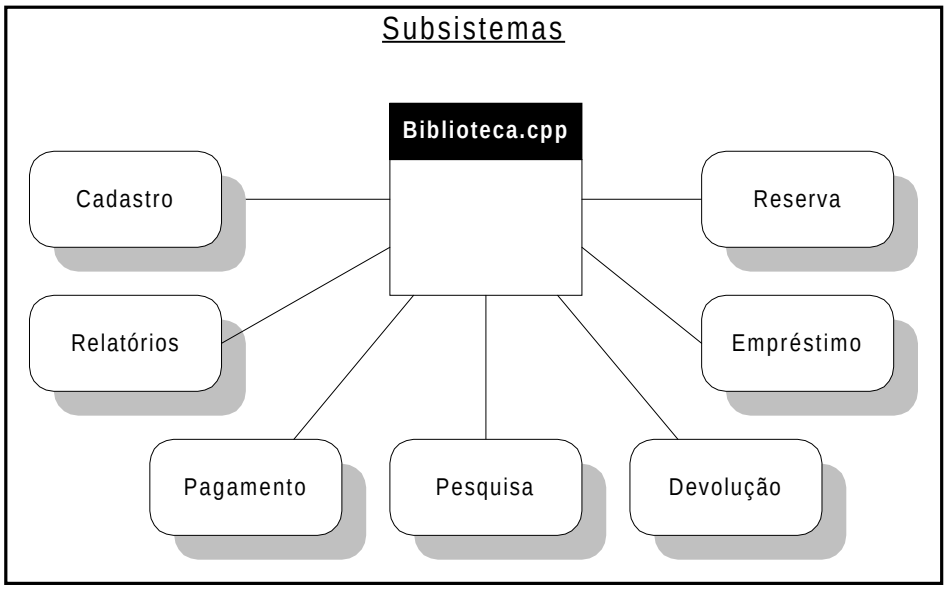



Figura C.37 - Diagrama de módulo - alto nível

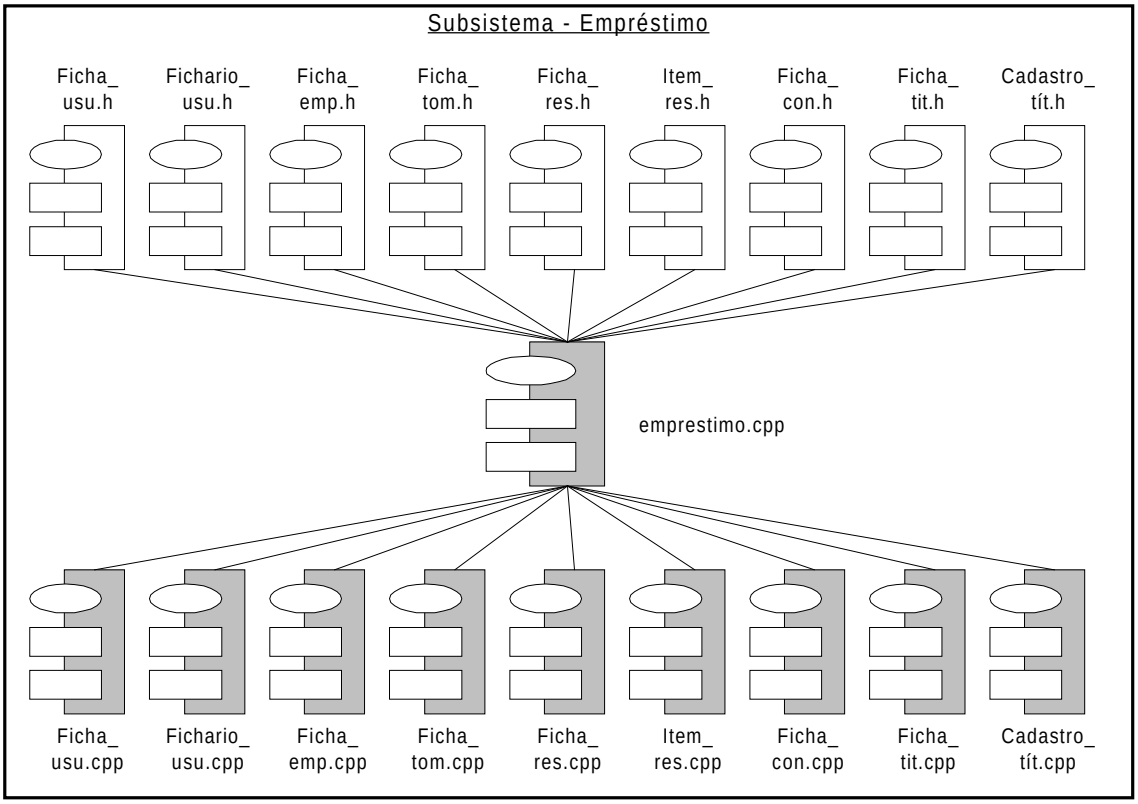



Figura C.38 - Diagrama de módulo - baixo nível

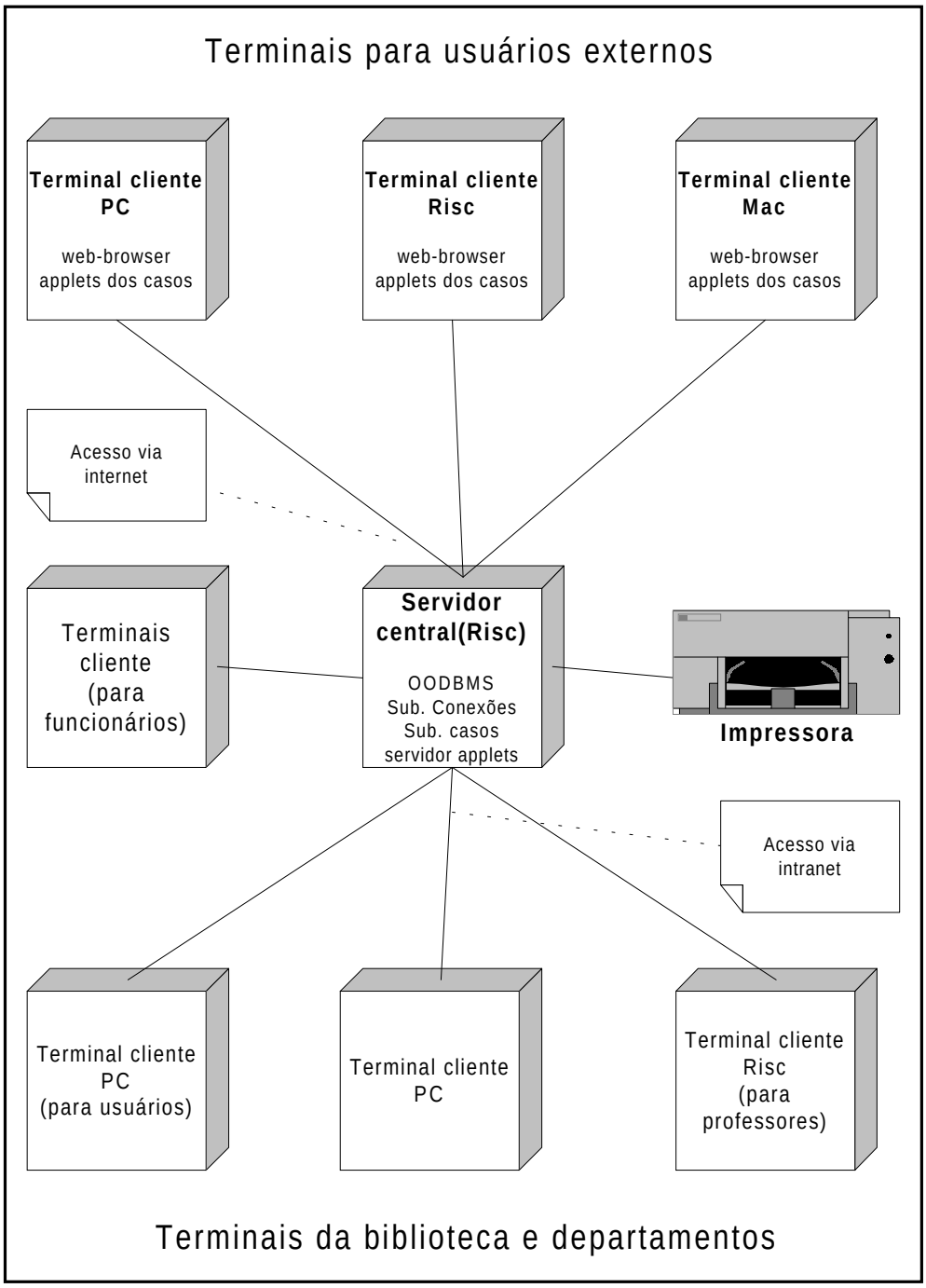



Figura C.39 - Diagrama de processo

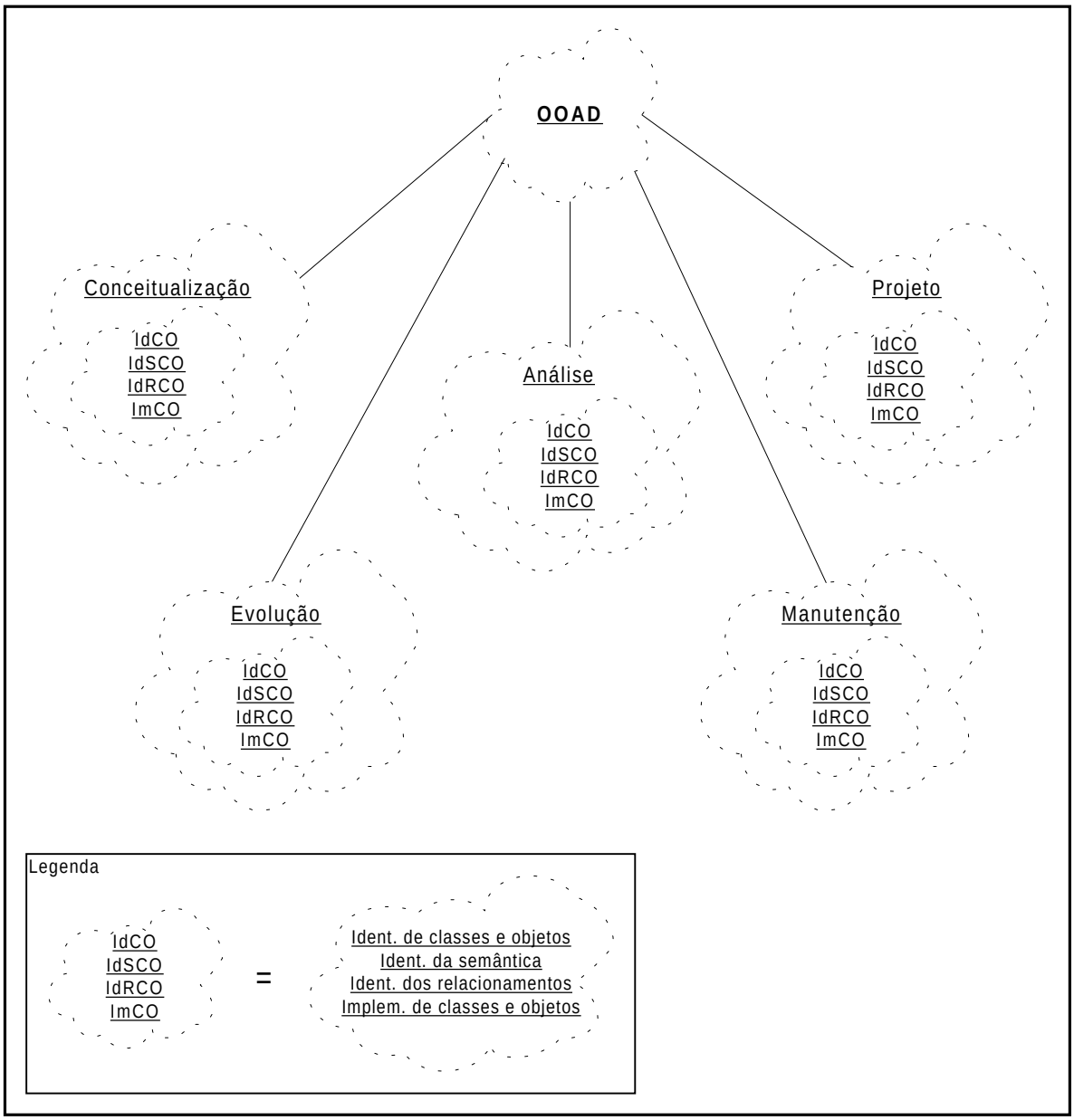



Figura C.40 - Resumo gráfico dos modelos/diagramas gerados pelo OOAD

## *10.4 Modelagem em OOA-OOD*

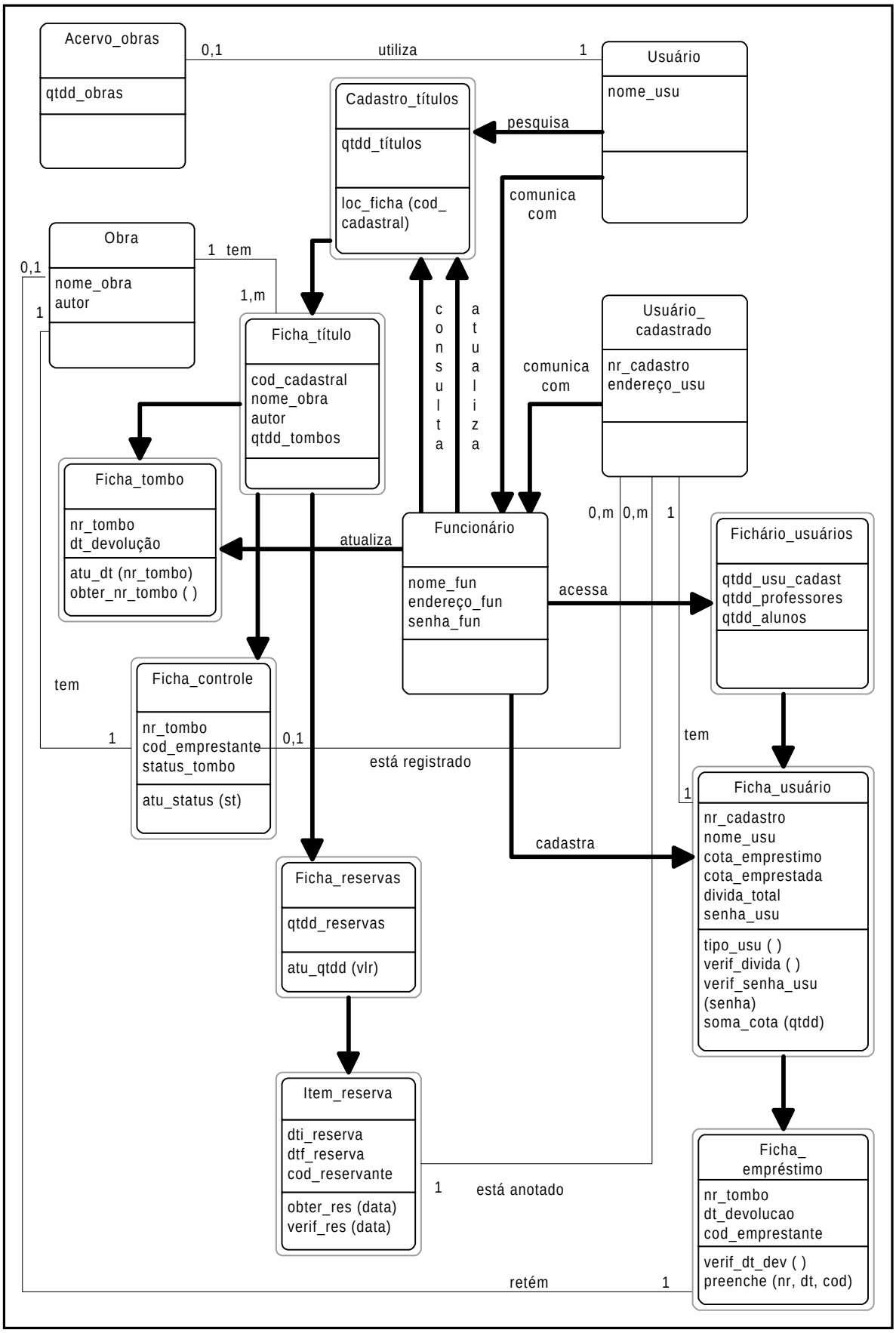



Figura C.41 - Modelo de análise - classe&objeto, atributo e serviço

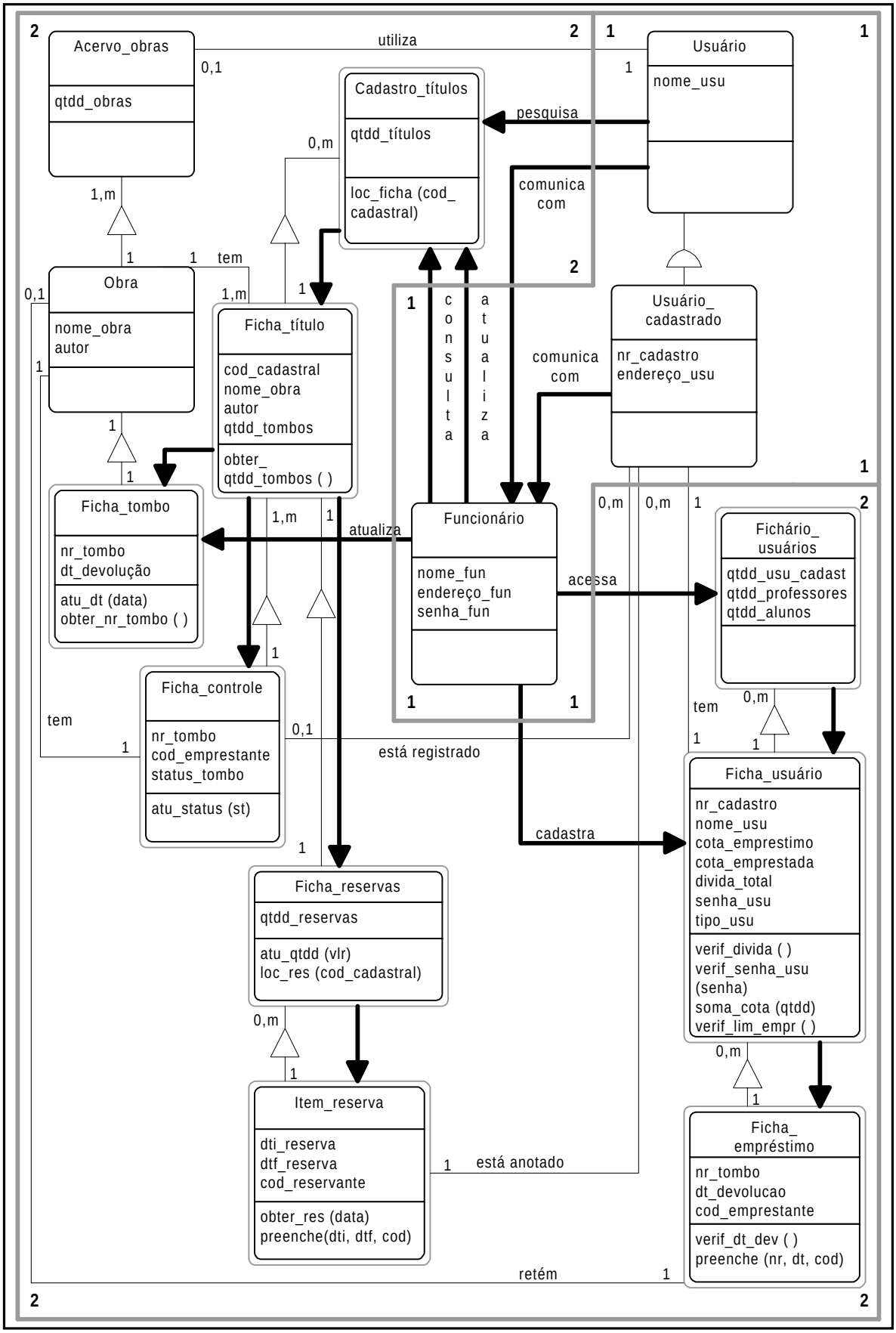



Figura C.42 - Modelo de análise - completo

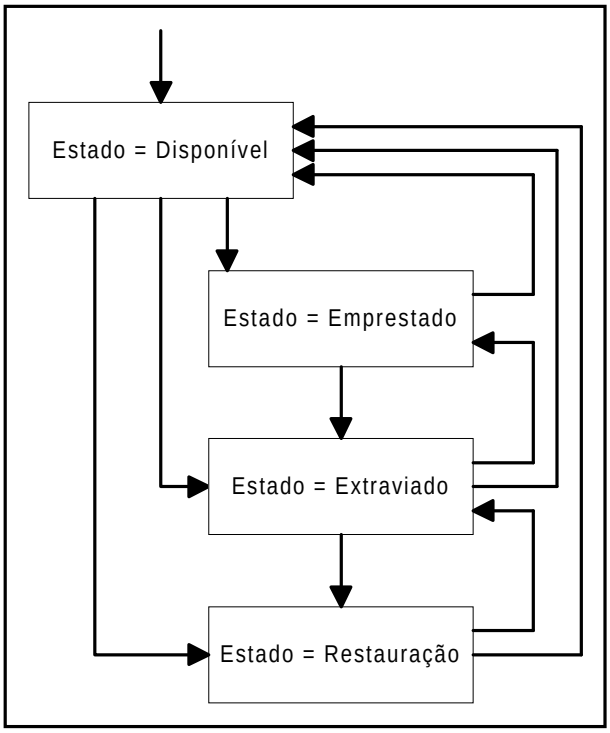



Figura C.43 - Diagrama de estado do objeto

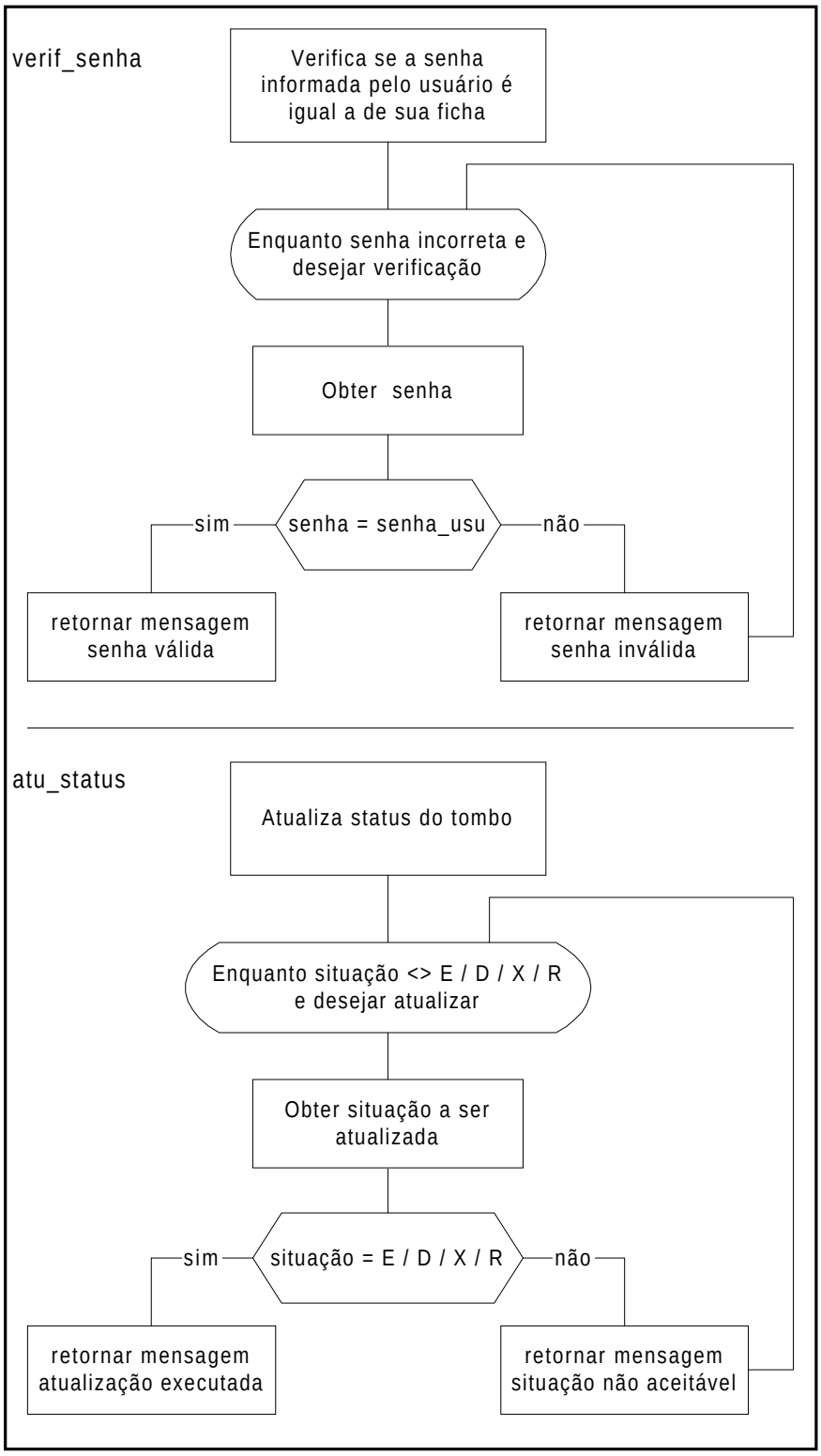


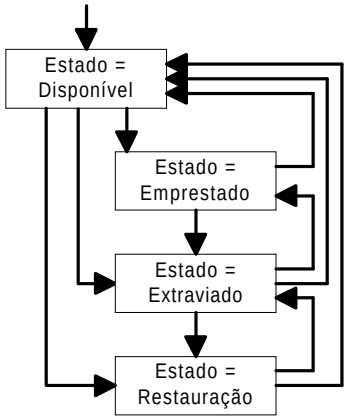

Figura C.44 - Diagrama de serviço de duas operações

*especificação* Ficha_controle
*atributo* nr_tombo: número do tombo
*atributo* cod_emprestante: código do emprestante que deve ser um
nr_cadastro em Usuário_cadastrado
*atributo* status_tombo: define status do tombo
(Emprestado / Disponível / eXtraviado / Restauração)

*Entradaexterna*
. Dados recebidos: status e código do emprestante
. Dados verificados: verifica se o cod_emprestante = nr_cadastro

*Saídaexterna*
. Controle de empréstimo: diminui / acrescenta valor a
cota_emprestada

*DiagramadeEstadodeObjeto*




*Especificações adicionais*
. O status_tombo refere ao estado da Obra (tombo) para o qual a
ficha mantém relação.

*serviço* atu_status (saída: mensagem)

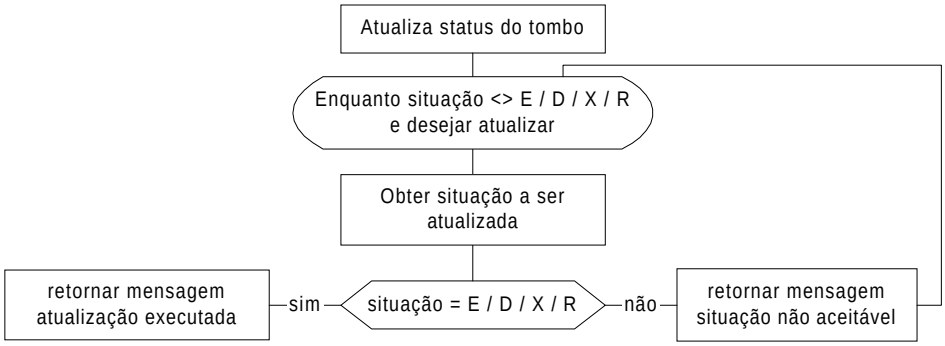



Figura C.45 - Especificação de classes

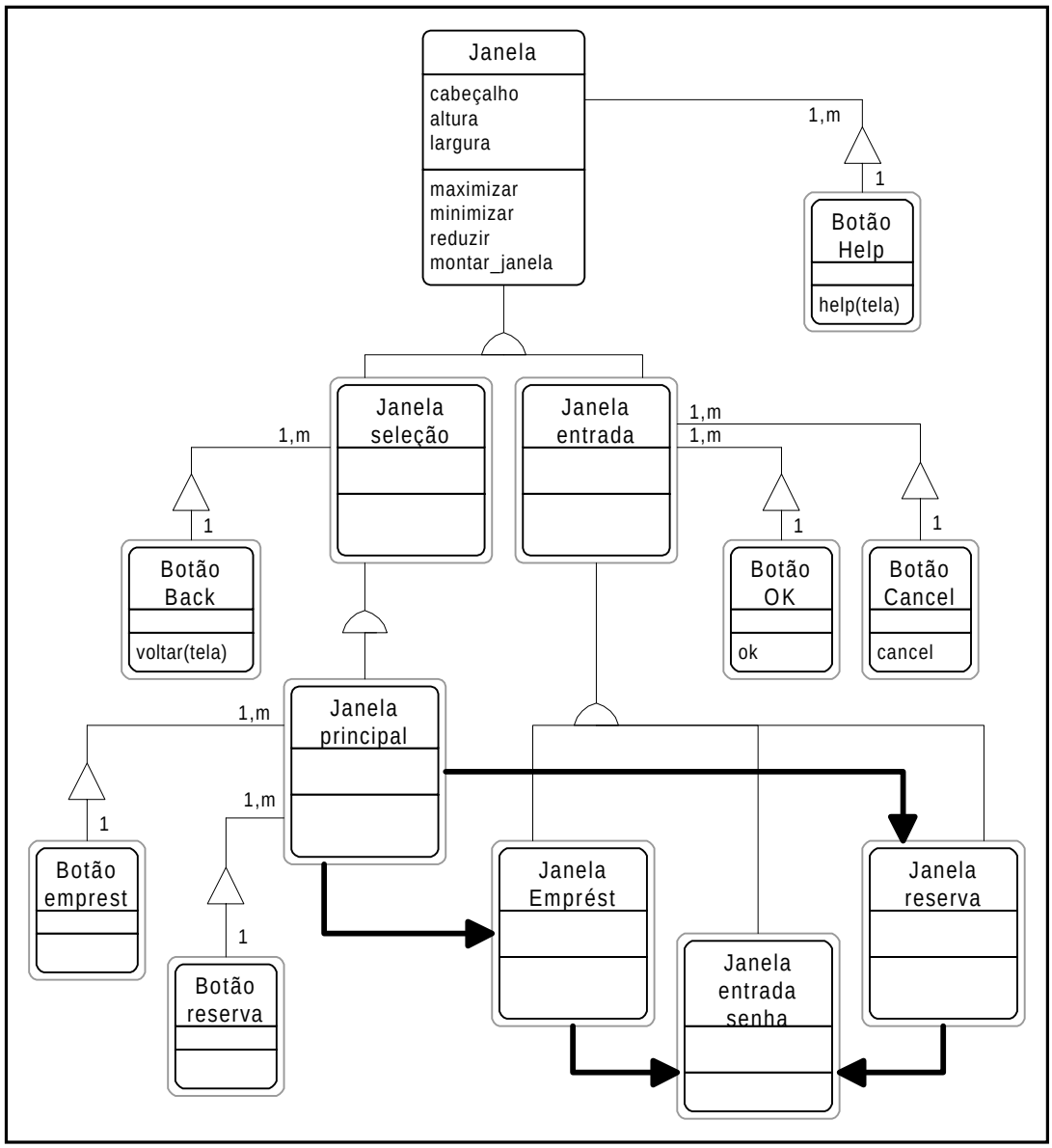
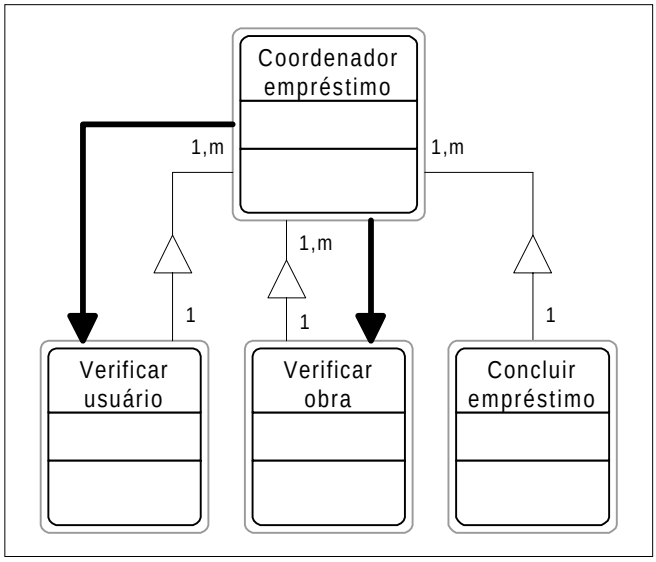



Figura C.46 - Modelo de projeto - C.I.H. expandido




Nome: Verificar usuário
Descrição: Esta tarefa é responsável por fazer verificações relativas ao usuário
Prioridade: Média
Serviços incluídos:
. Fichário_usuários. loc_usu (nr_cadastro)
. Ficha_usuário.verif_lim_empr
. Ficha_usuário.verif_divida
. Ficha_empréstimo.verif_dt_dev
. Ficha_usuário.verif_senha_usu (senha)
Coordenada por:
. controlada por eventos por evento de interação
Comunica via:
. recebe valores de interação humana

Nome: Concluir empréstimo
Descrição: Esta tarefa é responsável por concluir o empréstimo
Prioridade: Alta
Serviços incluídos:
. Ficha_empréstimo.preenche (nr, dt, cod)
. Ficha_controle.atu_status ('E')
. Ficha_usuário.soma_cota (1)
. Ficha_tombo.atu_dt (nr_tombo)
Coordenada por:
. controlada por tempo
Comunica via:
. recebe valores (nr-tombo) de outra tarefa (Verifica obra)

Nome: Verificar obra
Descrição: Esta tarefa é responsável por fazer verificações relativas a obra
Prioridade: Média
Serviços incluídos:
. Cadastro_títulos.loc_ficha (cod_cadastral)
. Ficha_título.obter_qtdd_tombos
. Ficha_controle.obter_status_tombo
. Ficha_reservas.loc_res (cod_cadastral)
. Item_reserva.verif_tombo_res (nr_tombo)
. Item_reserva.verif_cod_reserv (nr_cadastro)
Coordenada por:
. controlada por tempo
Comunica via:
. envia valores (nr_tombo) a outra tarefa (Concluir empréstimo)

Nome: Coordenador de empréstimo
Descrição: Tarefa responsável por controlar outras tarefas para efetuar um empréstimo
Prioridade: Baixa
Serviços incluídos:
. coordenar
Coordenada por:
. controlada por evento por evento de interação
Comunica via:
. recebe valores através da interação humana
. envia valores para tarefas sub-coordenadas e para a interação humana

Figura C.47 - Modelo de projeto - C.G.T. expandido

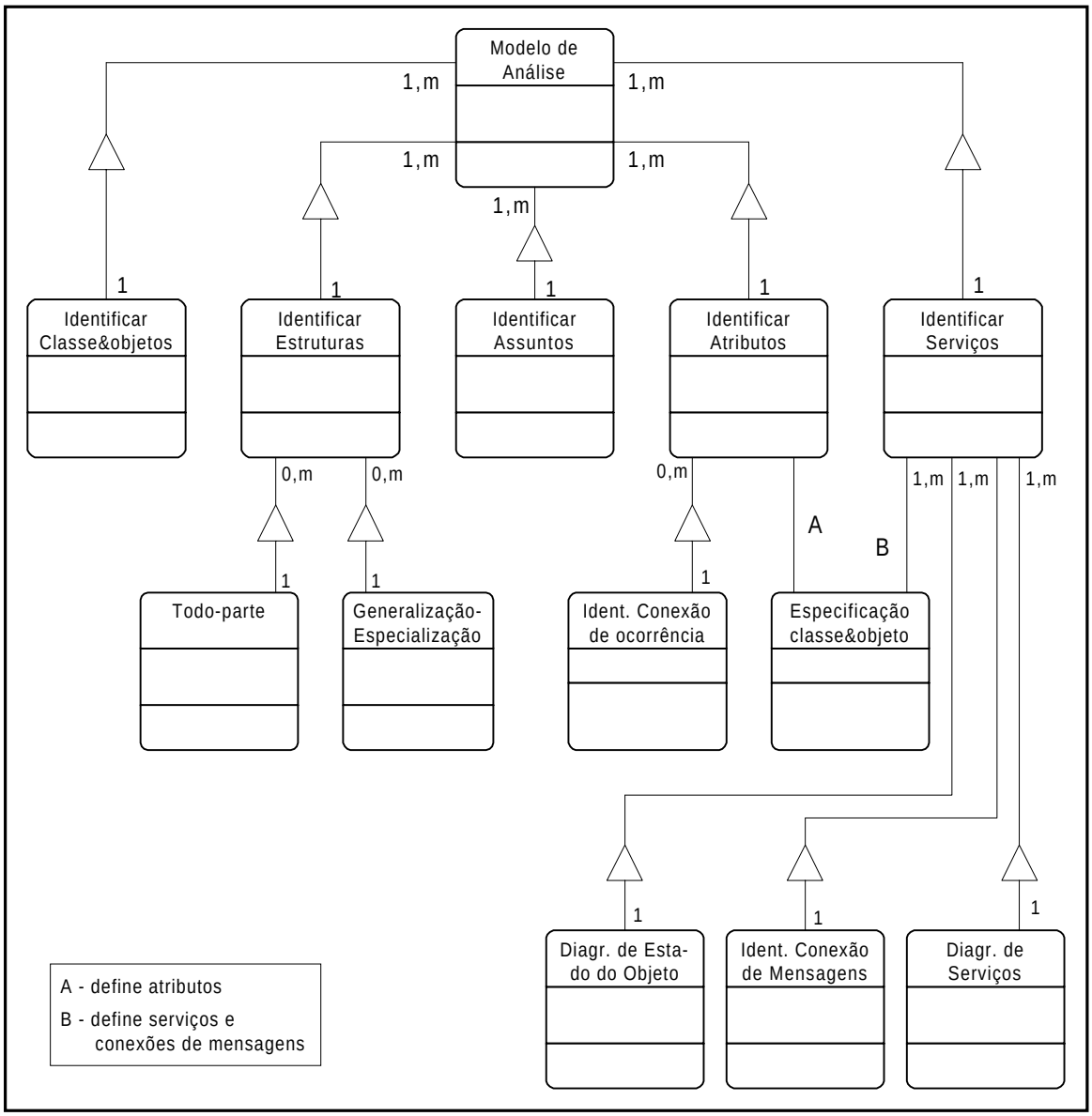



Figura C.48 - Resumo gráfico dos modelos/diagramas gerados pelo OOA-OOD - análise

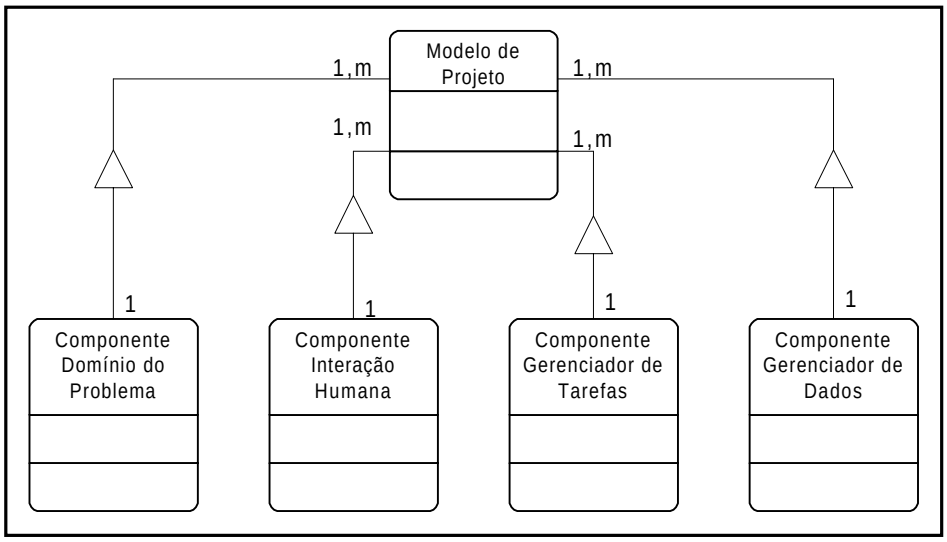



Figura C.49 - Resumo gráfico dos modelos/diagramas gerados pelo OOA-OOD - projeto

## *10.5 Modelagem em Fusion*

*Tabela C.1*
*Dicionário de dados - grafo de interação, métodos e eventos*

| **Nome** | **Categoria** | **Agente** | **Argumentos** | **Descrição** |
|---|---|---|---|---|
| dados_reserva | grafo de interação | usuário_ cadastrado | cod_cadastral, data_reserva, senha, nr_cadastro | O USUÁRIO_CADASTRADO fornece dados para que a reserva seja efetuada, antes disso várias situações são verificadas, dentre elas: tombos que estão disponíveis ou emprestados e senha do usuário. |
| verif_res | método | - | data_reserva | Classe: ITEM_RESERVA. Verifica se na data_reserva não possui uma reserva. |
| u_solicita_ reserva | evento | usuário_ cadastrado | nr_cadastro | USUÁRIO_CADASTRADO faz solicitação de reserva. |

*Tabela C.2*

*Dicionário de dados - classes, atributos, operações e associações*

| **Nome** | **Categoria** | **Descrição** |
|---|---|---|
| Acervo_obras | classe | possui informações totalizadoras das OBRAS. |
| Cadastro_títulos | classe | contém informações totalizadoras sobre FICHA_TÍTULO. |
| Fichário_usuários | classe | contém informações totalizadoras sobre FICHA_USUÁRIO, tais como quantidade de usuário, quantidade de professores e quantidade de alunos. |
| Funcionário | classe | armazena informações gerais dos funcionários (operadores, administradores e pessoal de suporte). Os funcionários atualizam os dados, gerenciam o sistema e controlam problemas de mal funcionamento. |
| Usuário | classe | possui informações sobre usuários, que não estão cadastrados. São usuários que simplesmente utilizam ou fazem pesquisa ao ACERVO. |
| Usuário_cadastrado | classe | possui informações sobre usuários cadastrados no sistema e que podem reservar, emprestar, devolver. Cada usuário cadastrado tem uma FICHA_USUÁRIO. Ele pode estar anotado como emprestante em FICHA_CONTROLE ou como reservante em ITEM_RESERVA. |
| nr_tombo | atributo | número do tombo. Obras com o mesmo título, possuem nr_tombo diferentes. |
| qtdd_obras | atributo | quantidade de obras no acervo. |
| senha_usu | atributo | senha do usuário. |
| obter_status_tombo | operação | verifica o status do tombo e o retorna. |
| verif_cod_reserv | operação | com o nr_cadastro do usuário é verificado se ele é igual ao cod_reservante para saber se o reservante já possui alguma reserva. |
| loc_usu | operação | localiza o usuário no fichário do usuários através do nr_cadastro para saber se ele está cadastrado como usuário cadastrado. |
| está_anotado | associação | um usuário pode estar anotado em um item de reserva, quando o usuário faz uma reserva. |
| pesquisa | associação | um usuário pode fazer uma pesquisa no cadastro de títulos. |
| utiliza | associação | um usuário pode utilizar o acervo de obras. |
| atualiza | associação | o funcionário atualiza a ficha de tombo, quando um empréstimo é efetuado. |

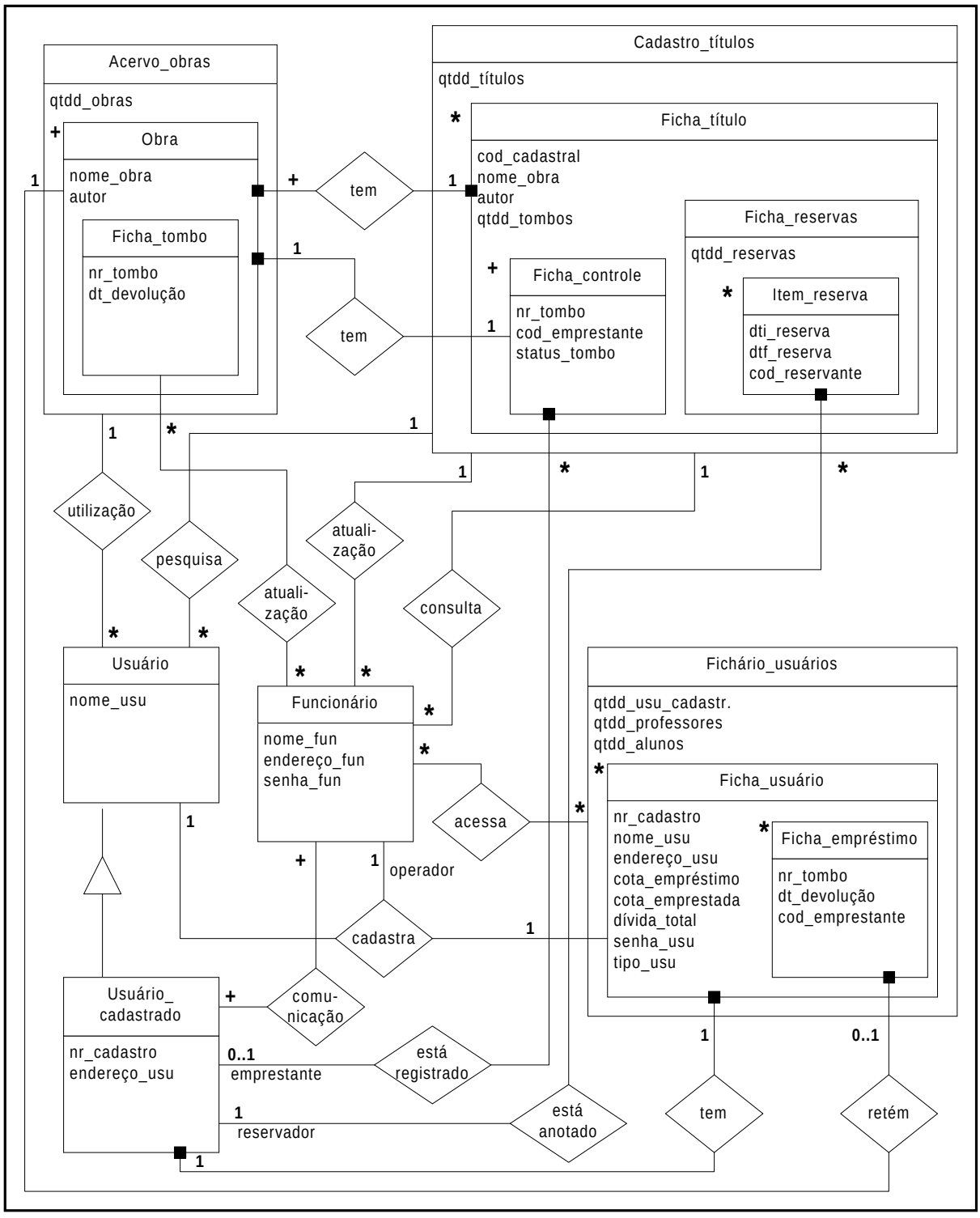



Figura C.50 - Modelo de objetos

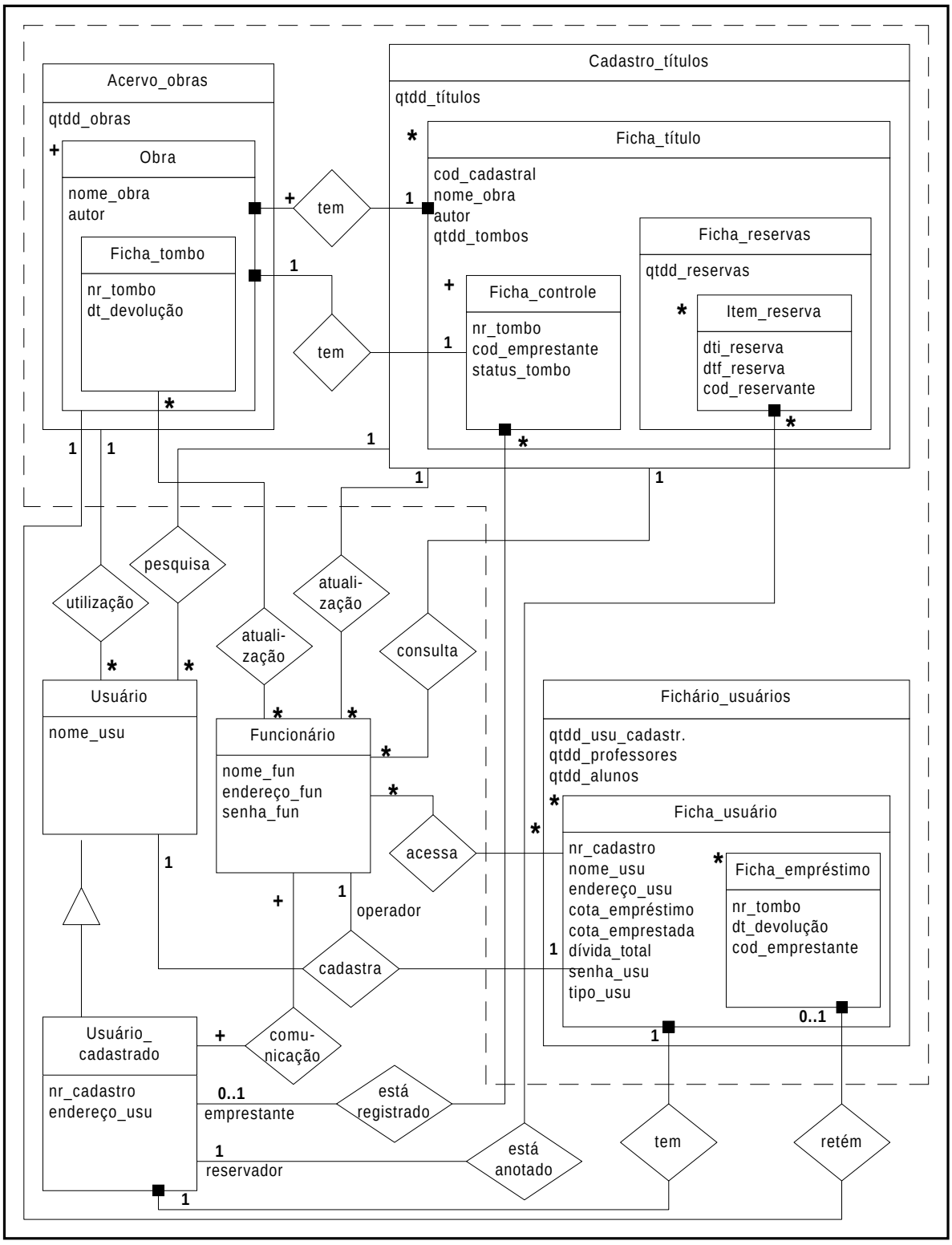



Figura C.51 - Modelo de objeto do sistema

```
lifecicle   Sistema_biblioteca: Iniciação. (Empréstimo || Reserva || Pagamento ||
                Devolução || Pesquisa)*

Iniciação = solicita_cadastro. #tipo_cadastro. tipo_cadastro. (dados_func |
          dados_usu | dados_obra)

Empréstimo = solicita_empréstimo. #solicita_dados_emp. dados_empréstimo.
             #obra_liberada

Reserva = (f_solicita_reserva | u_solicita_reserva). #solicita_dados_res.
          dados_reserva. #reserva_efetuada

Pagamento = . . .

Devolução = . . .

Pesquisa = . . .
```

Figura C.52 - Modelo ciclo-de-vida (incompleto)

## Cenário para um empréstimo

***Descrição***

O funcionário (emprestador) solicita empréstimo de obra ao sistema. O sistema solicita ao emprestador o número de cadastro do emprestante, para verificar se o mesmo está apto a fazer empréstimo. Esta verificação é feita observando se o emprestante já excedeu o limite máximo para empréstimo, se possui alguma dívida pendente e se está com alguma obra emprestada cuja data de devolução já esteja vencida.

Se o usuário estiver apto, o sistema solicita e o usuário fornece o código cadastral da obra. Verifica se o código cadastral é válido, e é verificado, também, se existe algum tombo daquela obra disponível, se existir, verifica se existe reserva para o tombo. Se existir reserva, observar se o reservante é o mesmo usuário que o emprestante, se for, o tombo é liberado, se não for, o tombo não pode ser liberado e toda verificação é feita para outro tombo. Se existir é solicitado ao emprestante sua senha, que é verificada na ficha do usuário.

Se a senha estiver correta, o sistema preenche a ficha de empréstimo e atualiza a ficha de controle do tombo, a ficha de usuário e a ficha de tombo. O emprestador libera a obra.

## Cenário para uma reserva

***Descrição***

O reservador (funcionário ou usuário) solicita reserva ao sistema. O sistema solicita ao reservador o número de cadastro do usuário. É verificado se o usuário está apto a fazer reserva, verificando se não possui dívida e se não está com alguma obra emprestada cuja data de devolução já esteja vencida.

Se o usuário estiver apto, o sistema solicita e o usuário faz a entrada do código cadastral da obra e a data para reserva. Verifica-se se o código cadastral é válido; é verificado, também, se existe algum tombo daquela obra disponível na data e se não existe reserva na data para o usuário (para validar a inexistência de mais de uma reserva para o mesmo usuário na mesma data).

Se existir o tombo referido acima, o usuário entra com sua senha, que é verificada com a senha do cadastro do usuário.

Se a senha estiver correta, a reserva é efetuada, atualizando a ficha de reserva e o item de reserva.

Figura C.53 - Descrição dos cenários - empréstimo/reserva

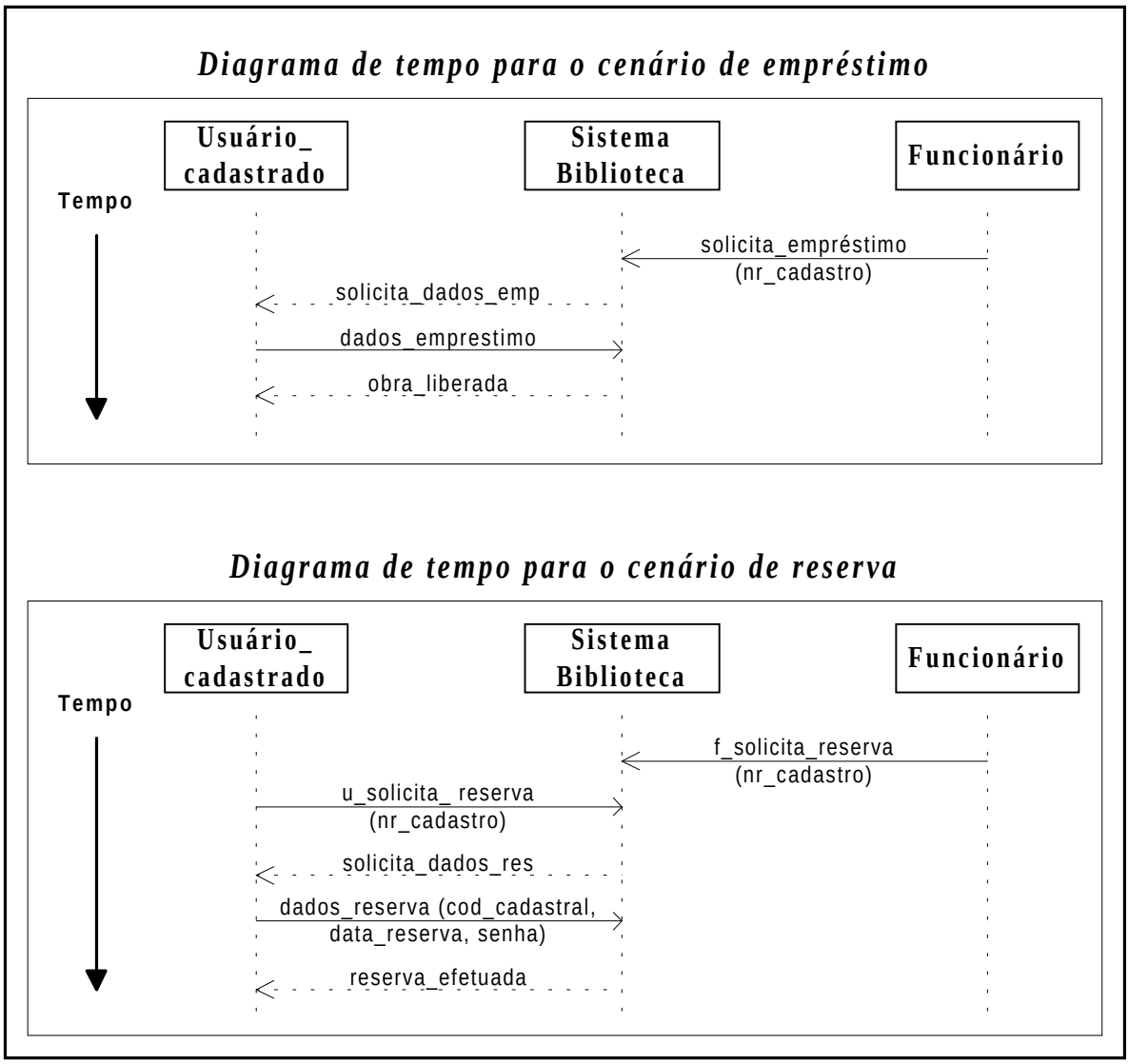



Figura C.54 - Diagrama de tempo - cenários empréstimo/reserva

**Operation**: solicita_empréstimo
**Description**: Funcionário solicita um empréstimo de obra para um usuário.

---

**Reads**: supplied nr_cadastro;
cota_empréstimo, cota_emprestada, dívida_total, dt_devolução

**Changes**: -

**Sends**: usuário_cadastrado: {solicita_dados_emp}

**Assumes**: A cota_empréstimo deve ser maior que cota_emprestada e a dívida_total deve ser igual a 0 e a dt_devolução não deve ter sido ultrapassada, ou seja a dt_devolução não pode ser menor que a data atual do sistema.

**Result**: A cota_empréstimo deve ser maior ou igual a cota_emprestada.

**Operation**: dados_reserva
**Description**: Usuário_cadastrado fornece dados para que a reserva seja efetuada.

---

**Reads**: supplied cod_cadastral, supplied data_reserva, supplied senha;
supplied nr_cadastro;
qtdd_tombos, nr_tombo, status_tombo, dt_devolução, qtdd_reserva,
dt_reserva, cod_reservante.

**Changes**: incrementar 1 em qtdd_reservas,
criação de uma nova reserva com os dados já informados pelo usuário

**Sends**: usuário_cadastrado: {reserva_efetuada}

**Assumes** : a qtdd_tombos deve ser maior que 0, então para todos os nr_tombos obtém-se o status_tombo. Se estiver emprestado, verifica-se se a dt_devolução for menor que a data_reserva, concluir reserva, senão procura outro tombo. Se estiver disponível, verifica existência de reserva. Se não existir reserva, concluir reserva, senão verifica se o nr_cadastro for diferente de cod_reservante, concluir reserva, senão reserva não pode ser feita.
Antes de concluir reserva verificar se a senha_usu é igual a senha.

**Result**: -

Figura C.55 - Esquema das operações solicita_empréstimo/dados_reserva

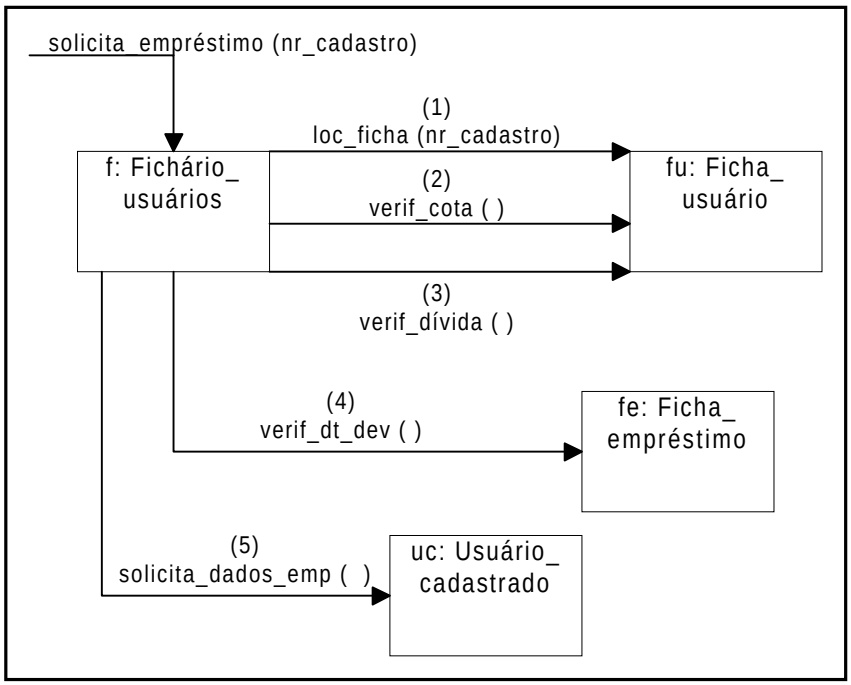



Figura C.56 - Grafo de interação de objeto - solicita_empréstimo

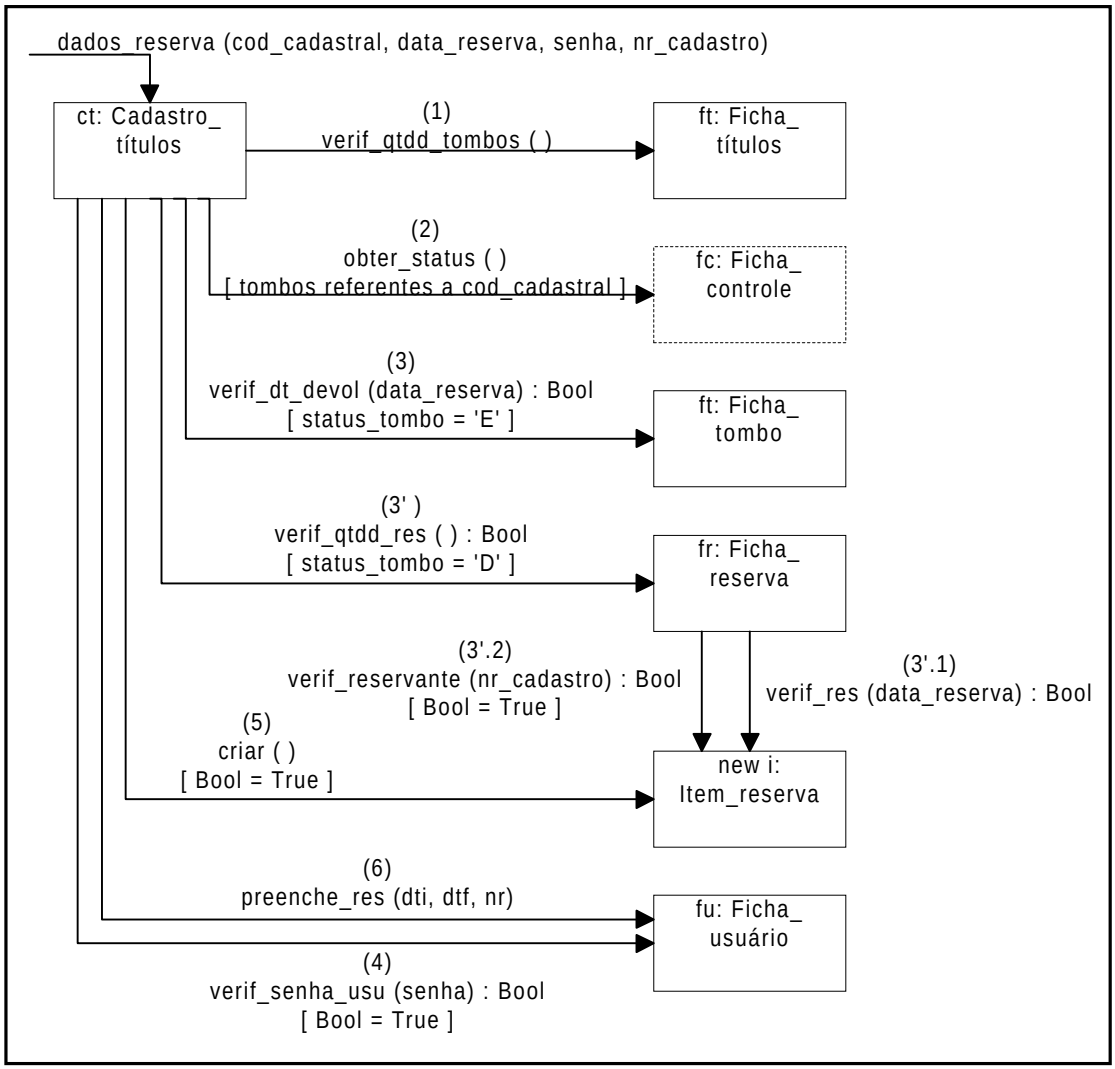



Figura C.57 - Grafo de interação de objeto - dados_reserva

```
class Ficha_controle
        attribute nr_tombo : int
        attribute cod_emprestante : int
        attribute status_tombo : char
        method atu_status (st)
endclass

class Ficha_tombo
        attribute nr_tombo : int
        attribute dt_devolução : data
        method atu_dt (data)
        method obter_nr_tombo ( )
endclass
```

Figura C.58 - Descrição de classes

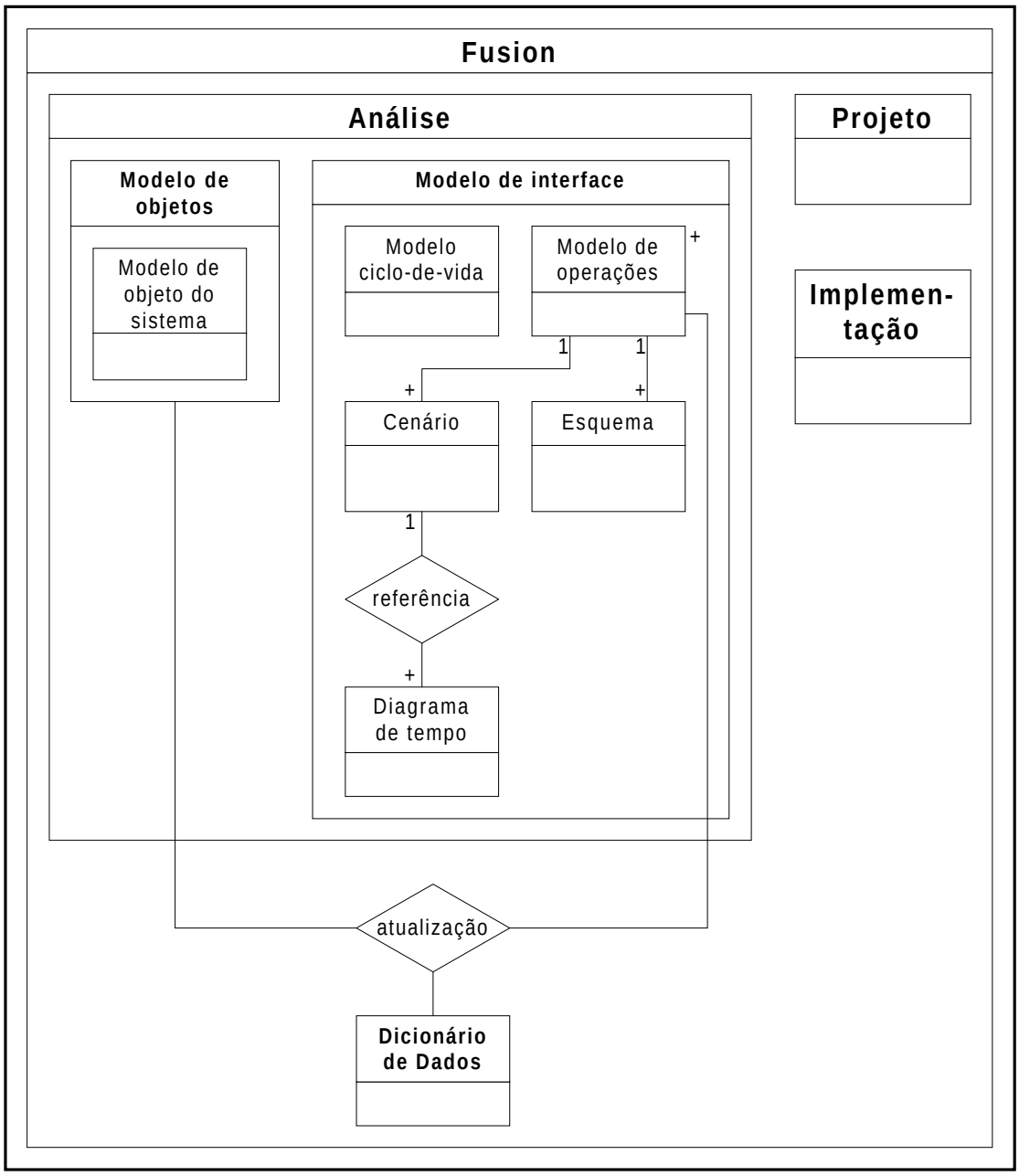



Figura C.59 - Resumo gráfico dos modelos/diagramas gerados pelo Fusion - análise

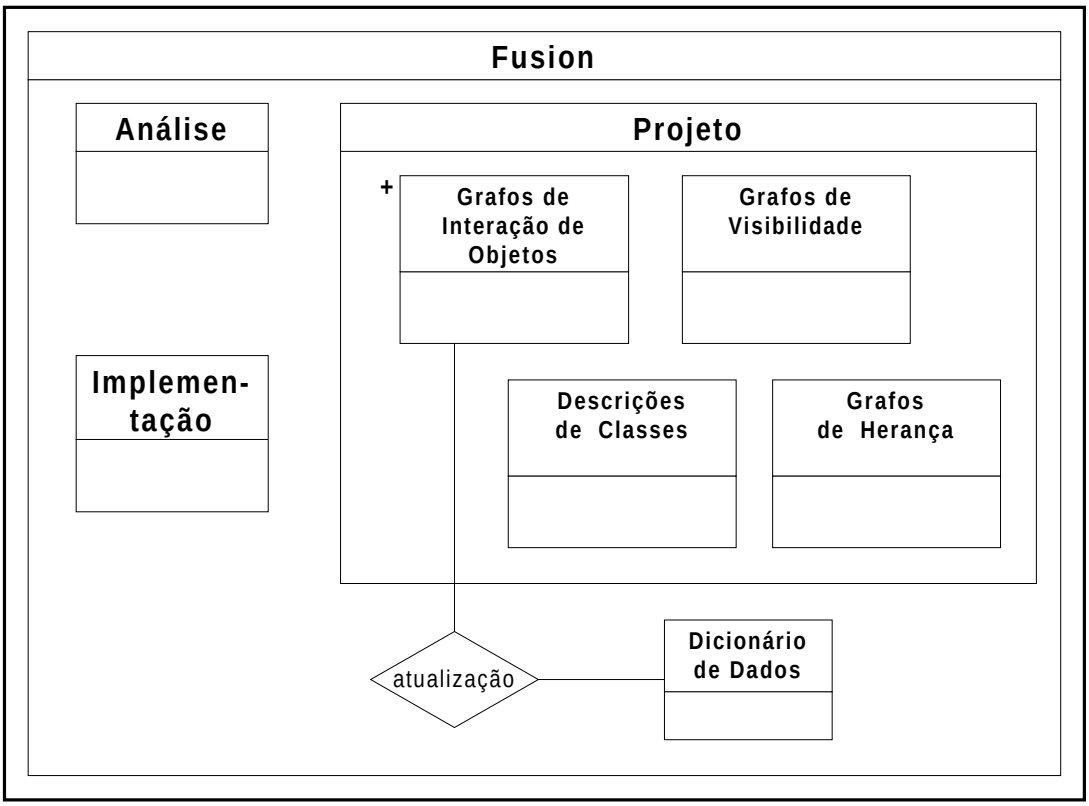



Figura C.60 - Resumo gráfico dos modelos/diagramas gerados pelo Fusion - projeto

## *10.6 Dicionário de dados do sistema*

Nesta seção é apresentado o dicionário de dados do sistema, que dá suporte aos modelos relacionados nas últimas cinco seções.

A tabela C.3 apresenta as classes, as associações entre as classes e a descrição de cada item mencionado. A tabela C.4 apresenta os atributos de cada uma das classes e uma breve descrição. A tabela C.5 apresenta as operações de cada uma das classes, a classe a que pertence, os argumentos necessários e uma descrição de cada classe. A tabela C.6 apresenta os eventos existem entre o sistema e os agentes externos, os agentes externos que criam o evento, os argumentos necessários e a descrição dos eventos.

*Tabela C.3*

*Classes, associações e descrições gerais do sistema de biblioteca*

| **Nome** | **Categoria** | **Descrição** |
|---|---|---|
| acervo_obras | Classe | possui informações totalizadoras das obras. |
| cadastro_títulos | Classe | contém informações totalizadoras sobre ficha de título. |
| ficha_controle | Classe | possui informações sobre o status do tombo e eventualmente do código do emprestante. |
| ficha_empréstimo | Classe | contém informações sobre o empréstimo de um tombo. Quando a obra estiver emprestada, esta ficha se encontrará junto a ficha de usuários, caso contrário, junto a obra física. |
| ficha_reservas | Classe | armazena informações sobre a quantidade de reservas de determinado tombo. |
| ficha_título | Classe | possui informações sobre a obra e a quantidade de tombos existente. |
| ficha_tombo | Classe | contém informações sobre a data para devolução de um tombo. Esta ficha se encontra fisicamente próxima a obra. |
| ficha_usuário | Classe | possui informações sobre usuários. |
| fichário_usuários | Classe | contém informações totalizadoras sobre ficha de usuário, tais como quantidade de usuário, e de professores e quantidade de alunos. |
| funcionário | Classe | armazena informações gerais dos funcionários (operadores, administradores e pessoal de suporte). Os funcionários atualizam os dados, gerenciam o sistema e controlam problemas de mal funcionamento. |
| item_reserva | Classe | possui informações sobre os itens da reserva de um determinado título. |
| obra | Classe | contém informações sobre as obras físicas. |
| usuário | Classe | possui informações sobre usuários, que não estão cadastrados. São usuários que simplesmente utilizam ou fazem pesquisa ao ACERVO. |
| usuário_ cadastrado | Classe | possui informações sobre usuários cadastrados no sistema e que podem reservar, emprestar, devolver. Cada usuário cadastrado tem uma FICHA_USUÁRIO. Ele pode estar anotado como emprestante em FICHA_CONTROLE ou como reservante em ITEM_RESERVA. |
| acessa | Associação | o funcionário pode acessar o fichário de usuários. |
| atualiza | Associação | o funcionário atualiza a ficha de tombo, quando um empréstimo é efetuado com a dt_devolução. |
| atualiza | Associação | o funcionário pode atualizar dados no cadastro de títulos. |
| cadastra | Associação | o funcionário como operador cadastra novo usuário com dados informados pelo usuário. |
| comunica com | Associação | o funcionário e o usuário cadastrado se comunicam ao solicitar empréstimo, reserva, devolução, fazer cobrança de título, etc.. |
| consulta | Associação | o funcionário pode consultar o cadastro de títulos. |
| está_anotado | Associação | um usuário como reservante pode estar anotado em um item de reserva, quando o usuário faz uma reserva. |
| está_registrado | Associação | um usuário como emprestante estará registrado na ficha de controle do tombo se tiver feito empréstimo deste tombo. |
| pesquisa | Associação | um usuário pode fazer uma pesquisa no cadastro de títulos. |
| retém | Associação | uma obra pode (obra está disponível) ou não reter (obra está emprestada) a ficha de empréstimo. |
| tem | Associação | uma ficha de título refere-se a uma ou mais obras. |
| tem | Associação | uma obra tem uma ficha de controle. |
| tem | Associaçã | o usuário cadastrado tem uma ficha de usuário, criada no ato do |

| | | |
|---|---|---|
| | o | cadastramento. |
| utiliza | Associaçã o | um usuário pode utilizar o acervo de obras, para consulta no local. |

*Tabela C.4*
*Atributos das classes e descrições gerais do sistema de biblioteca*

| | | |
|---|---|---|
| autor | Atributo | nome do autor da obra. |
| cod_emprestante | Atributo | código do emprestante, que deve ter um nr_cadastro equivalente em FICHA_USUÁRIO. |
| cod_reservante | Atributo | código do reservante, que deve ter um nr_cadastro equivalente em FICHA_USUÁRIO. |
| cota_emprestada | Atributo | cota de livros que já foram emprestados para o usuário. Esta cota não deve exceder a cota_empréstimo. |
| cota_empréstimo | Atributo | cota de livros que podem ser emprestados para o usuário. |
| dívida_total | Atributo | divida total do usuário. Se possuir divida não é possível fazer empréstimo nem reserva. |
| dt_devolução | Atributo | data de devolução do tombo emprestado. |
| dtf_reserva | Atributo | data final da reserva. |
| dti_reserva | Atributo | data de início da reserva. |
| endereço_fun | Atributo | endereço do funcionário para futuros contatos. |
| endereço_usu | Atributo | endereço do usuário para futuros contatos. |
| nome_fun | Atributo | nome do funcionário no cadastro. |
| nome_obra | Atributo | nome da obra no cadastro. |
| nome_usu | Atributo | nome do usuário no cadastro. |
| nr_cadastro | Atributo | número do cadastro do usuário. |
| nr_tombo | Atributo | número do tombo. Obras com o mesmo título, possuem nr_tombo diferentes. |
| qtdd_alunos | Atributo | quantidade de alunos cadastrados como usuários. |
| qtdd_obras | Atributo | quantidade de obras no acervo. |
| qtdd_professores | Atributo | quantidade de professores cadastrados como usuário. |
| qtdd_reservas | Atributo | quantidade de reservas de uma determinada obra. |
| qtdd_títulos | Atributo | quantidade de títulos disponíveis no acervo. |
| qtdd_usu_cadast. | Atributo | quantidade de usuários cadastrados. |
| qttd_tombos | Atributo | quantidade de tombos de uma determinada obra. |
| senha_fun | Atributo | senha do funcionário. |
| senha_usu | Atributo | senha do usuário. |
| status_tombo | Atributo | status do tombo - Emprestado / Disponível / eXtraviado. |
| tipo_usu | Atributo | Tipo do usuário - professor / aluno. |

*Tabela C.5*
*Operações das classes e descrições gerais do sistema de biblioteca*

| **Nome** | **Categoria** | **Classe** | **Argumentos** | **Descrição** |
|---|---|---|---|---|
| atu_dt | Operação | Ficha_ tombo | data | atualiza dt_devolução com data, quando o empréstimo é efetuado. |
| atu_qtdd | Operação | Ficha_ reserva | vlr | atualiza qtdd_reservas com +1, se houve reserva e -1 se excluiu reserva. |
| atu_status | Operação | Ficha_ controle | st | atualiza status do tombo, quando seu estado atual se modifica, ou seja, quando é emprestado, reservado ou devolvido. |
| loc_ficha | Operação | Cadastro_ títulos | cod_cadastral | localiza a ficha com o cod_cadastral em ficha de título para saber se existe. |
| loc_usu | Operação | Fichário_ usuários | nr_cadastro | localiza o usuário no fichário do usuários através do nr_cadastro para saber se ele está cadastrado como usuário cadastrado. |
| loc-res | Operação | Ficha_ reservas | cod_cadastral | localiza reserva em Item reserva para determinado codigo_cadastral. |
| obter_ qtdd_reserva | Operação | Ficha_ reservas | | obtem valor contido em qtdd_reservas |
| obter_ qtdd_tombos | Operação | Ficha_ título | | buscar valor armazenado no atributo qtdd_tombos |
| obter_ nr_tombo | Operação | Ficha_ controle | | |
| obter_nr_ tombo | Operação | Ficha_ tombo | | obtem o primeiro tombo de determinado cod_cadstral. |
| obter_res | Operação | Item_reserv a | data | verifica se existe reserva na data informada. |
| obter_status_ tombo | Operação | Ficha_ controle | | verifica o status do tombo e o retorna. |
| preenche | Operação | Item_reserv a | dti, dtf, cod | cria nova reserva com os dados informados, data inicial, data final e cod do reservante. |
| preenche | Operação | Ficha_ empréstimo | nr, dt, cod | atualiza ficha de empréstimo com os dados informados: nr do tombo, data de devolução e cod. do emprestante. |
| soma_cota | Operação | Ficha_ usuário | qtdd | adiciona ou diminui a cota_emprestada a qtdd_tombos emprestada ou devolvida, respectivamente. |
| verif_ tombo_res | Operação | Item_ reserva | nr_tombo | verifica se há tombo reservado para determinado cod-cadastral. |
| verif_ cod_reserv | Operação | Item_reserv a | nr_cadastro | com o nr_cadastro do usuário é verificado se ele é igual ao cod_reservante para saber se o reservante já possui alguma reserva. |
| verif_ lim_empr | Operação | Ficha_ usuário | | verifica se é possível fazer empréstimo, ou seja, se cota_emprestada < cota_empréstimo. |
| verif_ senha_usu | Operação | Ficha_ usuário | senha | verifica se a senha informada é igual a senha cadastrada. |
| verif_dívida | Operação | Ficha_ usuário | | verifica se o usuário possui alguma dívida |
| verif_dt_dev | Operação | Ficha_ | | verifica qual é a data de devolução de |

| | | | | |
|---|---|---|---|---|
| | | empréstimo | | determinado tombo para saber se usuário não deve algum tombo. |
| verif_res | Operação | Item_reserva | data | |

*Tabela C.6*
*Eventos que afetam o sistema e descrições gerais do sistema de biblioteca*

| **Nome** | **Categoria** | **Agente** | **Argumentos** | **Descrição** |
|---|---|---|---|---|
| dados_ empréstimo | Evento | Usuário_ cadastrado | cod_cadastral | Usuário cadastrado informa qual o cod cadastral da obra que ele deseja obter emprestada. |
| dados_reserva | Evento | Usuário_ cadastrado | cod_cadastral, data_reserva, senha | O usuário cadastrado fornece dados para que a reserva seja efetuada, antes disso várias situações são verificadas, dentre elas: tombos que estão disponíveis ou emprestados e senha do usuário. |
| f_solicita_ empréstimo | Evento | Funcioná-rio | nr_cadastro | Funcionário faz solicitação de reserva para usuário que informou seu nr_cadastro. |
| solicita_ empréstimo | Evento | Funcioná-rio | nr_cadastro | Funcionário solicita empréstimo para usuario mediante nr_cadastro. |
| u_solicita_ reserva | Evento | Usuário_ cadastrado | nr_cadastro | Usuário cadastrado faz solicitação de reserva. |

# Apêndice D

## *Unified Modeling Language - UML*

UML – *Unified Modeling Language* é uma notação e não um método, que foi proposta por Grady Booch, Ivar Jacobson e James Rumbaugh com ajuda de em conjunto de metodologistas e organizações. Esta linguagem foi adotada como padrão de notação pela OMG[26].

Esta notação foi padronizada em setembro de 1997, quando este trabalho se encontrava em fase avançada de desenvolvimento. É um assunto de extrema relevância ao contexto deste trabalho, por ter uma estreita ligação com o mesmo, pois uma notação, especialmente esta que foi padronizada, pode ser utilizada juntamente com qualquer método.

Este apêndice relata o propósito de uma notação, as idéias que formam a UML e o processo de sua criação e padronização.

### *11.1 Aspectos da notação*

Um método é composto pela notação e pelo processo, segundo (Fowler, 1997), o mais importante dos dois é a notação, pois quando duas pessoas desejam discutir o projeto, não é necessário saberem o processo e sim a notação.

Segundo (Quatrani, 1998), três facetas: notação, processo e ferramenta são necessários para que um projeto seja bem sucedido. O projeto irá falhar se: 1) souber a notação, mas não souber utilizar o processo, 2) tiver um ótimo processo e não conseguir comunicá-lo através de uma notação e 3) não houver uma boa documentação (ferramenta).

O processo deve ser aquele que melhor se adeqüe ao desenvolvimento de um determinado sistema. Uma boa ferramenta deve ser escolhida para documentar de maneira segura, fornecendo pequenos testes e verificações para que se possa obter um desenvolvimento mais rápido.

Uma notação geralmente acompanha o processo, porque o metodologista engloba em sua notação aspectos que irão capturar / modelar o que foi definido pelo processo. Entretanto, pode-se utilizar uma notação que não é a definida pelo metodologista, assim a escolha de uma notação envolve vários aspectos como: conhecimento da mesma e seu propósito. Para um bom aproveitamento é necessário verificar qual notação melhor se adequa ao propósito - sistemas complexos / simples, sistemas grandes / pequenos -, qual notação melhor modela o problema e

[26] OMG - *Object Management Group* é um grupo com mais de 800 sócios, dentre vendedores e desenvolvedores de *software* e usuários finais, podemos citar: *American Airlines*, *Canon*, *Hewlett-Packard*, *Sun Microsystems* e *Unisys Corporation*. OMG foi fundada em maio de 1989 com a missão de promover a teoria e a prática da tecnologia de objetos para o desenvolvimento de sistemas distribuídos. Maiores detalhes http:// www.omg.org.

verificar sua compatibilidade com o processo. Depois observa-se o conhecimento da equipe de desenvolvimento em relação a notação, que é muito importante para diminuição de prazos e aumento de qualidade.

Existe uma tendência em se representar os modelos de um desenvolvimento de sistemas através da UML, pois além de ser uma notação bastante abrangente é uma notação padrão. Todos os envolvidos neste projeto de desenvolvimento – analistas, projetistas, especialistas do domínio – compreenderão mais rapidamente o que está sendo modelado. Não haverá perda de tempo com a adaptação à notação, desde que os elementos representados nos modelos (classes, objetos, relacionamentos, etc.) estejam de acordo com a UML.

A UML é uma linguagem para especificar, visualizar, construir e documentar os artefatos de um sistema de *software*, como também para modelagem de sistemas onde não haverá geração de *software* (Rational, 1997b). *A UML tenta codificar as melhores práticas que os autores (Rumbaugh, Jacobson e Booch) e outros metodologistas encontraram em projetos orientados a objetos em todo o mundo* (Booch, 1997).

A UML como uma notação padrão tem o propósito de unificar as notações existentes para a modelagem orientada a objeto, especialmente as desenvolvidas por James Rumbaugh, Grady Booch e Ivar Jacobson. Esse padrão facilita a comunicação entre analistas e projetistas que utilizem em seus desenvolvimentos métodos diferentes e, provavelmente, notações diferentes. Além disso, facilita a compreensão dos modelos, por exemplo, os modelos da estrutura estática de um sistema possuirão a mesma sintática e semântica.

Dentro da mesma organização pode-se utilizar métodos diferentes para desenvolver tipos diferentes de sistema, deste modo cada equipe de desenvolvimento teria um treinamento em uma notação. Com a adoção da UML, a locação de um membro de uma equipe ao desenvolvimento de outro projeto ou organização que utilizem outros métodos não acarretará em dispensar tempo e dinheiro no treinamento deste membro em outra notação, somente no processo utilizado.

Muitos métodos juntamente com suas notações foram criados durante as décadas de 70 a 90 (Quatrani, 1998) (Fowler, 1997). Cada um deles tentava impor-se no mercado com seu foco em determinada fase do desenvolvimento ou com uma notação mais abrangente. Alguns autores (Rational, 1997a) (Quatrani, 1998) nomeiam este período de *method wars* – guerra dos métodos. A guerra das notações está finalizada com o advento da UML, mas a guerra dos métodos ainda continuará, pois o uso de cada método depende do tipo e das características do sistema que está sendo desenvolvido.

A UML representa a unificação das notações dos métodos OMT – James Rumbaugh, OOAD – Grady Booch e OOSE – Ivar Jacobson, como dito anteriormente, além de outras boas idéias de outros metodologistas como mostrado pela figura D.1.

A figura D.1 apresenta os componentes da UML e seus respectivos fornecedores, além das notações de Rumbaugh, Booch e Jacobson, algumas técnicas e diagramas serviram como entrada: classificação de Odell, pré e pós condições de Meyer, ciclo de vida dos objetos de Shlaer-Mellor, diagramas de estado de Harel, estruturas, padrões e notas de Gamma et. al., responsabilidades de

Wirfs-Brock, classes *singleton* de Embly e descrição de operação e numeração de mensagem do método Fusion.

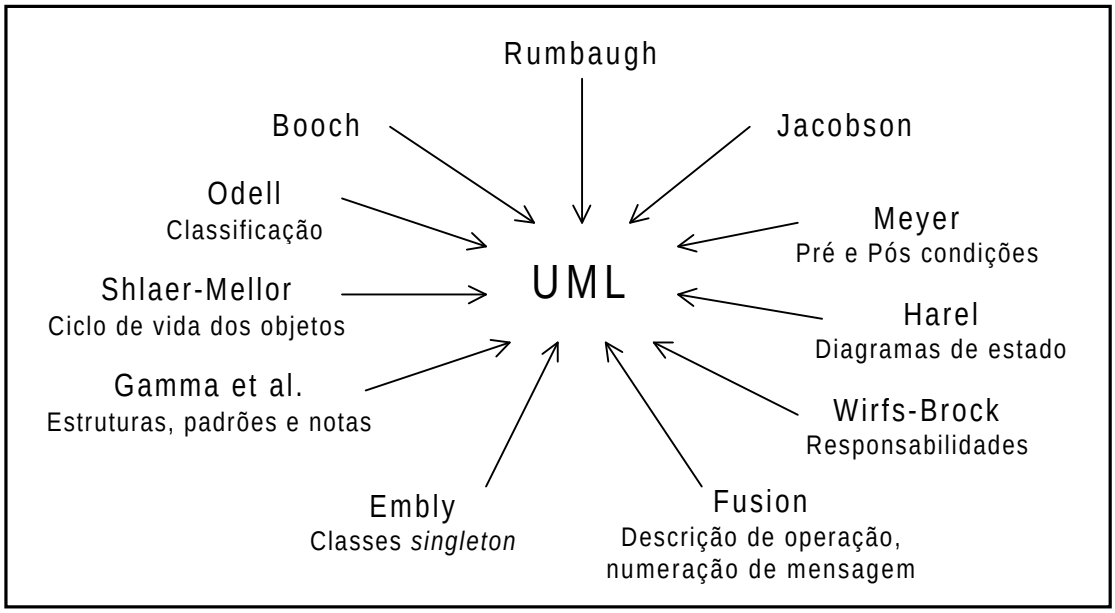



Figura D.1 - Entradas da UML (Quatrani, 1998)

O desenvolvimento da UML como mostra a figura D.2, iniciou-se quando James Rumbaugh e Grady Booch resolveram unificar suas notações em outubro de 1994. A primeira versão 0.8, chamada de Método Unificado, foi liberada ao público como rascunho em outubro de 1995. Como Rumbaugh e Booch adotaram os *use* cases de Jacobson, nada mais natural que o advento do metodologista Ivar Jacobson ao grupo. Com a chegada das idéias de Jacobson e as críticas do público surgiram as versões 0.9 e 0.91 liberadas em junho de 1996 e outubro de 1996, respectivamente (Quatrani, 1998) (Fowler, 1997) (Booch, 1997).

Enquanto a comunidade em geral continuava a realimentar o processo de criação com críticas, a *Rational* solicita um RFP[27] – *Request for Proposal* para tentar padronizar sua notação. A *Rational*, então, estabeleceu ligações com várias organizações que ajudaram na definição da versão 1.0 da UML. Dentre as organizações, podem ser citadas: *Sterling Software*, *Hewlett-Packard*, *i-Logix*, *IntelliCorp*, *IBM*, *ICON Computing*, *MCI Systemhouse*, *Microsoft*, *Oracle*, *Rational Software*, *TI* e *Unisys*. Esta versão foi enviada a OMG como resposta inicial à RFP em janeiro de 1997. Em setembro de 1997, a versão 1.1 foi enviada a OMG como reposta final à RFP, onde as seguintes organizações se juntaram ao grupo contribuindo com suas idéias: *IBM*, *ObjecTime*, *Platinum Technology*, *Petch*, *Taskon*, *Reich Technologies* e *SofTeam* (Rational, 1997c).

[27] RFP - *Request For Proposal* é distribuído por uma *Task Force* de um dos comitês de tecnologia OMG e é um pedido explícito para os membros da OMG para submeter propostas para avaliação da tecnologia e sua adoção.

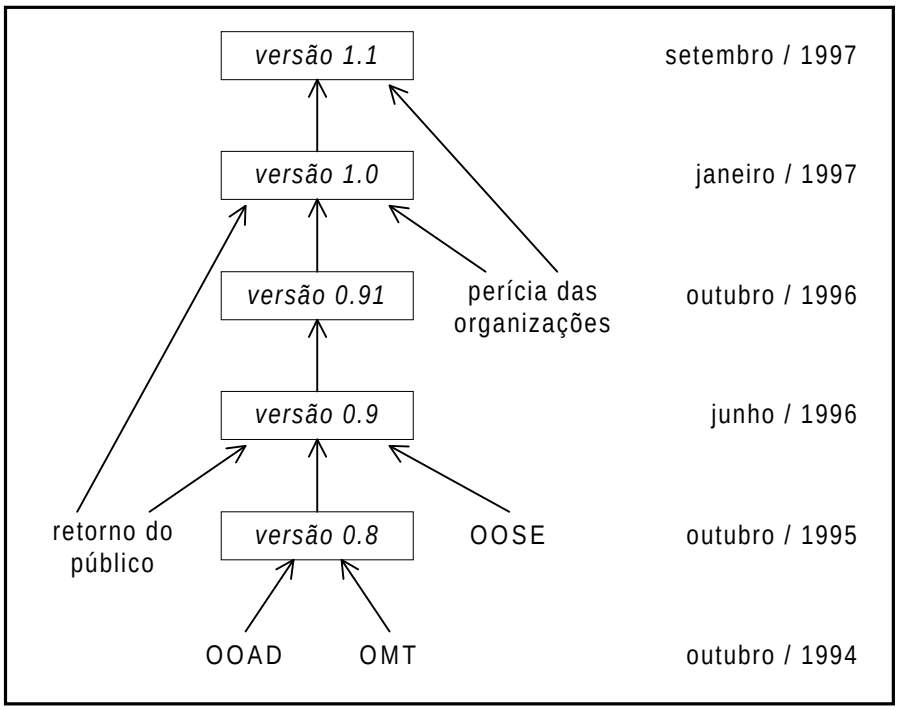



Figura D.2 - Processo de criação e padronização da UML (baseado em ( Rational, 1997a))

Cada uma dessas organizações contribuíram com a UML focalizando interesses próprios, pois perceberam que a UML seria estratégica para seus negócios, como por exemplo: HP, observando seu método *Fusion*; *Microsoft*, preocupada com seus padrões *ActiveX* e COM. Maiores detalhes ver (Rational, 1997a).

## *11.2 Considerações finais*

Uma linguagem de modelagem padronizada para o desenvolvimento sob o paradigma de orientação a objetos, vem acabar com a guerra das notações e facilitar a comunicação entre analistas e projetistas de vários domínios.

A *Rational* continuará a liderar a definição da UML e manter suas revisões que eventualmente a OMG solicitar (Rational, 1997b).

Rumbaugh, Booch e Jacobson continuam trabalhando na tentativa de unificar seus métodos que está sendo chamado de *Rational Objectory Process*. O objetivo é fornecer uma estrutura para o processo que capturará elementos comuns, permitindo que as pessoas usem técnicas que são mais apropriadas para seus projetos (Fowler, 1997).