Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação

Desenvolvimento de Sistema de Informação de Fábrica em OOHDM - um Estudo de Caso

Viviane Sayuri Tonaki
Kattiana Fernandes Constantino
Renata Pontin de Mattos Fortes

Notas Didáticas

São Carlos - SP
Setembro/2003

Resumo

Estas notas didáticas apresentam os conceitos de Engenharia de Web para criar aplicativos Web com qualidade, reportando um estudo de caso que foi realizado a partir do desenvolvimento de um Sistema de Informação de Fábrica, adotando-se o método OOHDM – *Object Oriented Hypermedia Design Method* [SR95]. O processo de engenharia começou com a formulação do problema a ser resolvido, os requisitos foram analisados e os projetos arquitetural, navegacional e de interface foram elaborados. Dessa forma, neste relatório são também apresentadas as principais atividades do método de modelagem OOHDM e os respectivos artefatos modelados no estudo de caso. Finalmente, o Sistema de Informação projetado deverá ser implementado usando linguagens e ferramentas especializadas, associadas com a Web.

Sumário

1 Introdução e Motivação **1**

2 Engenharia de Web **3**

2.1 Introdução 3

2.2 A Evolução da Web 3

2.3 Panorama 5

2.4 Características de Aplicativos para a Web 5

2.5 Modelo de Processo da *Web Engineering* 7

2.6 Alguns Métodos de Projeto de Aplicações Web 9

3 O método OOHDM **11**

3.1 Introdução 11

3.2 Um panorama do Método OOHDM 11

3.3 Projeto Conceitual 13

3.4 Projeto Navegacional 16

3.5 Projeto de Interface Abstrata 18

3.6 Implementação 20

4 Modelagem **24**

4.1 Introdução 24

4.2 Contextualização da aplicação 25

4.3 Levantamento de Requisitos e Projeto Conceitual 26

4.3.1 Identificação dos Atores e Tarefas 27
4.3.2 Especificação dos Cenários 28
4.3.3 Especificação dos Casos de Uso 34
4.3.4 Identificação dos UIDs 42
4.3.5 Validação dos Casos de Uso e dos UIDs 44
4.3.6 Especificação do Esquema Conceitual 45
4.4 Projeto Navegacional 48
4.4.1 Projeto da Navegação da Tarefa 49
4.4.2 Projeto da Navegação da Aplicação 54
4.4.3 Especificação Navegacional da Aplicação 55
4.4.4 Especificação Final para a Navegação da Aplicação 59
4.5 Outros Métodos de Levantamento de Requisitos 59

5 Conclusão **62**

A UIDs **64**

B Cartões de Especificação **73**

Referências Bibliográficas **82**

Lista de Figuras

2.1 Um *framework* para o processo de Engenharia de Web [Pre02] 7

3.1 Esboço da metodologia OOHDM [Ros96] 12
3.2 Esquema Conceitual da aplicação "Guia de Profissões" 15
3.3 Esquema Navegacional da Aplicação "Guia de Profissões" 17
3.4 Diagrama de Configuração da classe Profissão - "Guia de Profissões" 19

4.1 Fases do Desenvolvimento e Implementação de Aplicações Web [GSV00] . . 25
4.2 Fases da Análise de Requisitos e Projeto Conceitual [GSV00] 27
4.3 UID referente ao Caso de uso 1 44
4.4 Esquema Conceitual da Fundição 47
4.5 Principais fases do Projeto Navegacional [GSV00] 48
4.6 Passos para projetar a navegação de uma tarefa [GSV00] 49
4.7 Cartão de especificação para listas [GSV00] 52
4.8 Cartão de contexto ou grupo de contexto [SV] 52
4.9 Cartão de estrutura de acesso [SV] 53
4.10 Esquema de Contexto da fundição 56
4.11 Esquema Navegacional da fundição 58

A.1 UID referente ao Caso de uso 2 64
A.2 UID referente ao Caso de uso 3 65
A.3 UID referente ao Caso de uso 4 65

A.4 UID referente ao Caso de uso 5 66
A.5 UID referente ao Caso de uso 6 66
A.6 UID referente ao Caso de uso 7 67
A.7 UID referente ao Caso de uso 8 67
A.8 UID referente ao Caso de uso 9 68
A.9 UID referente ao Caso de uso 10 68
A.10 UID referente ao Caso de uso 11 69
A.11 UID referente ao Caso de uso 12 69
A.12 UID referente ao Caso de uso 13 70
A.13 UID referente ao Caso de uso 14 70
A.14 UID referente ao Caso de uso 15 71
A.15 UID referente ao Caso de uso 16 71
A.16 UID referente ao Caso de uso 17 72
A.17 UID referente ao Caso de uso 18 72

Lista de Tabelas

4.1 Tabela de Mapeamento Conceitual−Navegacional para a fundição 60

Capítulo

1

Introdução e Motivação

Com o desenvolvimento de aplicativos Web a cada dia mais complexos, muitas empresas e pessoas têm se interessado cada vez mais a entrarem no “novo mundo da computação”, que de forma coloquial é chamado de “mundo da Internet”. Os aplicativos Web, destinados a atender uma ampla variedade de domínios de informação, usam ainda diversas mídias como imagem, som, animações, além de proporcionarem que dados de conteúdo dinâmico sejam disponibilizados com extrema rapidez. Os aplicativos Web, que inicialmente eram somente de conteúdo textual, se tornam serviços completos. Assim, grandes e complexos aplicativos têm sido construídos, e seus projetos possuem níveis de dificuldade de desenvolvimento, manutenção, gerenciamento e controle de qualidade, muito semelhantes aos de software convencionais. Um agravante é que muito do que tem sido feito para Web possui pouca disciplina e preocupação mínima com técnicas e métodos padronizados. Neste cenário, surge a necessidade de Engenharia para aplicativos que são construídos, considerando-se a Web – comumente denominada de Engenharia de Web.

A Engenharia de Web não é um clone perfeito da Engenharia de Software, mas toma emprestado dela muitos conceitos e princípios fundamentais, enfatizando as mesmas atividades técnicas e de gestão. Há diferenças sutis no modo pelo qual essas atividades são conduzidas, mas a principal filosofia, que determina uma abordagem disciplinada para o desenvolvimento de um sistema baseado em computador, é idêntica.

Para Pressman, vale a pena investir em engenharia para qualquer produto ou sistema importante. Antes de começar a construí-lo, é melhor entender o problema, projetar uma solução viável, implementá-la de maneira sólida e testá-la completamente. Deve também, controlar mudanças que irão ocorrer durante o desenvolvimento e ter algum mecanismo para garantia da qualidade.

Pressman [Pre02] propõe um *framework* para Engenharia de Web[1], constituindo-se de atividades técnicas de Projeto Arquitetural, Projeto Navegacional, Projeto de Interface, paralelamente com atividades não-técnicas como Projeto e Produção do Conteúdo. Na literatura existem diversos métodos para as atividades técnicas. Schwabe propôs o método OOHDM - *Object Oriented Hypermedia Design Method* [SR95], para o desenvolvimento de aplicaçôes hipermídia, tal método é compreendido por quatro atividades: Modelagem Conceitual, Projeto de Navegação, Projeto de Interface Abstrata e Implementação.

Nestas notas didáticas é apresentado o conceito de Engenharia de Web, o método de modelagem OOHDM e a modelagem de uma aplicação para uma fundição como exemplo. Deve-se ressaltar que a aplicação para a fundição faz parte de um projeto apoiado pela Fapesp que está sendo desenvolvido paralelamente, sob a orientação do Prof^{o} Dr. Marcos Nereu Arenales, e cujo título é "A programação da produção em fundições".

Este trabalho está organizado da seguinte forma: capítulo 2 apresenta os conceitos, processos e equipe de engenharia da web; o capítulo 3 descreve o método de modelagem OOHDM e suas principais atividades; e o capítulo 4 apresenta o estudo de caso para o desenvolvimento de um Sistema de Informação para uma fundição usando a modelagem OOHDM. Finalmente, no capítulo 5, concluímos o estudo realizado.

[1]Neste trabalho, entendemos que o termo Engenharia de Web referencia o processo de gerenciamento, suporte e desenvolvimento de aplicativos que deverão ser executados considerando-se a tecnologia de Web, ou seja, Internet, *browsers*, arquitetura cliente-servidor, etc.

Capítulo

2

Engenharia de Web

"Os princípios de engenharia, planejar antes de projetar e projetar antes de construir, se mantiveram ao longo de todas as transições de tecnologia anteriores; eles vão igualmente sobreviver a esta transição." ***Watts Humphrey***

2.1 Introdução

Este capítulo apresenta os conceitos de Engenharia de Web caracterizando as etapas do seu processo, e descrevendo as atividades técnicas e de gestão.

2.2 A Evolução da Web

A Web foi inicialmente concebida com o intuito de compartilhar informações científicas entre alguns poucos cientistas. O conteúdo era estático e apenas textual, não havia imagens, sons, animações ou conteúdo gerado dinamicamente para cada usuário, a interação era limitada, a navegabilidade era fácil, alto desempenho era desejável, mas não essencial, os sites eram desenvolvidos por apenas uma pessoa ou um pequeno grupo.

Mas ocorreu uma evolução! A partir do início da década de 90, pode-se dizer que a World Wide Web e a Internet têm obrigado que a cada dia, mais e mais pessoas entrem no mundo da computação. Hoje é possível comprar ações e fundos mútuos, baixar músicas, ver filmes, vender artigos pessoais, reservar passagens aéreas, conhecer pessoas, fazer transações bancárias, assistir palestras e inúmeros outros usos que não poderiam ser imaginados dez anos atrás. Murugesan [MD00] destaca que a Internet levou apenas quatro anos para estar em 30% dos lares americanos. É um tempo bem curto quando comparado a outros produtos: o telefone levou 40 anos, o rádio levou 35 anos, o videocassete demorou 20 anos, a televisão 26 anos e o próprio computador levou 19 anos.

Escopo e complexidade foram aumentando, pequenos serviços foram cedendo espaço para grandes aplicativos, e com isso também aumentou a complexidade dos projetos e as dificuldades de desenvolvimento, manutenção, gerenciamento e controle de qualidade. Tudo feito com pouca disciplina, sem preocupação com técnicas e métodos padronizados. A atitude usual parecia ser *"termine rapidamente o sistema e coloque-o no mercado. Depois nós arrumamos os problemas (e entendemos melhor, o que realmente era necessário ser construído)"*, gerando aplicativos com grande probabilidade de problemas como baixo desempenho e falhas. Na Web problemas como esses são ainda mais graves que no software tradicional, pois como afirma Nielsen *"a distância entre um site e seus concorrentes é sempre de apenas poucos cliques"*.

Se a abordagem ad-hoc *(implementar e testar)* atual para o desenvolvimento Web persistir, a infra-estrutura de aplicações Web, poderá levar a algo que poderia ser chamado de uma "rede embaraçada". Este termo tem sido associado às diversas aplicações Web desenvolvidas de forma rudimentar e que possuem uma probabilidade muito alta de fracasso. Para piorar, a medida em que os sistemas Web se tornam cada vez mais complexos, uma falha em um sistema pode se propagar para vários outros.

Na opinião de Roger Pressman [Pre02], vale a pena investir em engenharia para qualquer produto ou sistema importante. Antes de começar a construí-lo, é melhor entender o problema, projetar uma solução viável, implementá-la de maneira sólida e testá-la completamente. Deve também, controlar mudanças que irão ocorrer durante o desenvolvimento e ter algum mecanismo para garantia da qualidade. E aí surge a Engenharia para a Web.

2.3 Panorama

Sistemas e aplicações baseados na Web produzem uma complexa matriz de conteúdo e funcionalidade para uma ampla população de usuários finais. A engenharia de Web é o processo usado para criar aplicativos de alta qualidade, e mesmo não sendo um clone perfeito da engenharia de software, a engenharia de Web toma emprestado muitos conceitos e princípios fundamentais, enfatizando as mesmas atividades técnicas e de gestão. Há diferenças sutis no modo pelo qual essas atividades são conduzidas, mas a filosofia que prevalece, de que consiste de uma abordagem disciplinada para o desenvolvimento de um sistema baseado em computador, é idêntica.

A engenharia de Web aplica uma abordagem genérica combinada com estratégias, táticas e métodos especializados. O processo de engenharia de Web começa com uma formulação do problema a ser resolvido pela Web. O projeto é planejado e os requisitos são analisados. Os projetos arquitetural, navegacional e de interface são conduzidos. O sistema é implementado usando linguagens e ferramentas especializadas, associadas com a Web, e o teste começa. Como os aplicativos para a Web evoluem continuamente, são necessários mecanismos para controle de configuração, garantia de qualidade e contínuo suporte a esta evolução.[Pre02]

A engenharia de Web *(Web Engineering)* diz respeito ao estabelecimento e uso de princípios científicos sólidos, de engenharia e de gestão, e abordagens disciplinadas e sistemáticas para que o desenvolvimento, a disponibilização e a manutenção de sistemas e aplicações de alta qualidade baseados na Web sejam bem sucedidos [Pre02].

2.4 Características de Aplicativos para a Web

Segundo Pressman [Pre02], as seguintes características podem ser encontradas na grande maioria dos aplicativos Web:

- **Rede Intensiva**: pois utilizam recursos de rede por natureza (Internet, Intranet ou Extranet), e por estar em uma rede, deve atender às necessidades de diversas comunidades de clientes e usuários.

- **Dirigido por Conteúdo**: em muitos casos, uma das funções primárias do aplicativo para a Web é utilizar hipermídia para apresentar textos, gráficos, e vídeo para os usuários.

- **Evolução Contínua**: ao contrário dos aplicativos convencionais que evoluem através de uma série de versões planejadas e lançadas em determinados intervalos de tempo, os aplicativos para a Web evoluem continuamente, ou seja, de forma muito rápida e com extrema freqüência.

Alguns autores costumam comparar a evolução de aplicativos para a Web com jardinagem. Desenvolver um site consiste em criar uma infra-estrutura (planejar o jardim) e então *"plantar"* as informações que irão crescer e florescer neste jardim. Com o tempo, o jardim *(o site)* irá evoluir, mudar e crescer. Um bom planejamento inicial permitirá que este crescimento ocorra de forma controlada e consistente.

Pressman [Pre02] destaca ainda as principais diferenças entre desenvolver um aplicativo para a Web e desenvolver um software tradicional:

- **Imediatismo**: o tempo que um site completo precisa para ficar pronto pode ser apenas de alguns poucos dias ou semanas. Os desenvolvedores devem, portanto, utilizar métodos de planejamento, análise, projeto, implementação e teste que estejam adaptados para estes cronogramas.

- **Segurança**: os aplicativos para a Web estão disponíveis via rede, que permite acesso a uma grande variedade de tipos de usuário. Portanto, é preciso implementar medidas rígidas de segurança no aplicativo e na infra-estrutura do mesmo.

- **Estética**: quando um aplicativo é projetado para vender produtos ou idéias, a estética pode ser tão importante para o seu sucesso, quanto o projeto técnico.

As seguintes características gerais se aplicam a todos os aplicativos Web, mas com diferentes graus de influência nos seus projetos:

- ***Informacional***: cujo conteúdo é disponibilizado somente para leitura; assim, geralmente devem ser fornecidas navegação e ligações simples.

- ***Download***: que permite ao usuário baixar informação de um servidor adequado.

- ***Adaptável***: em que o usuário adapta o conteúdo às suas necessidades específicas.

- ***Interação***: em que a comunicação entre uma comunidade de usuários ocorre por intermédio de salas de bate-papo, quadros de aviso ou mensagens instantâneas.

- ***Entrada de Usuário***: em que a entrada é baseada em formulários, principal mecanismo para comunicar o que é necessário.
- ***Orientada a Serviço***: que fornece serviços ao usuário.
- ***Portal***: que orienta o usuário para outros conteúdos ou serviços da Web, se fora do domínio da aplicação do portal.
- ***De Acesso à Base de Dados***: em que o usuário consulta uma grande base de dados e extrai informação.
- ***Data Warehousing***: em que o usuário consulta uma coleção de grandes bases de dados e extrai informação oriunda de diversas bases de dados.

2.5 Modelo de Processo da *Web Engineering*

A medida em que os aplicativos para a Web evoluem de conteúdo estático para dinâmico, cresce a necessidade de aplicar princípios sólidos de gestão e engenharia. É necessário, portanto, um modelo do processo de desenvolvimento que supra tais necessidades de maneira eficiente.

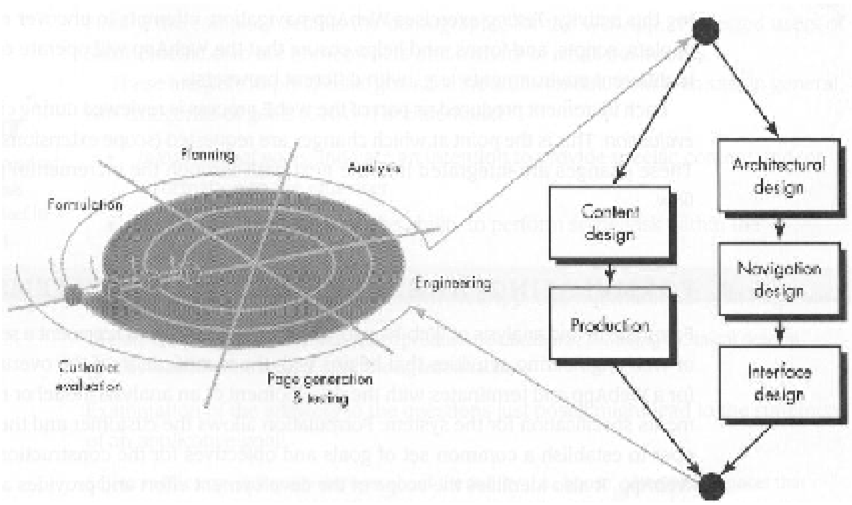

Figura 2.1: Um *framework* para o processo de Engenharia de Web [Pre02]

A figura 2.1 apresenta um *framework* proposto por Pressman, para o processo da engenharia de Web. O processo começa pela *Formulação* - atividade que identifica as metas e os objetivos do aplicativo e estabelece o escopo do primeiro incremento. O *Planejamento* estima o custo global do projeto, avalia os riscos associados com o esforço de desenvolvimento e define um cronograma bem definido para o primeiro incremento e um menos definido para os demais. A *Análise* estabelece os requisitos técnicos do aplicativo e identifica os itens de conteúdo que serão incorporados. Os requisitos para o projeto gráfico também são definidos.

A atividade de *Engenharia* incorpora duas tarefas paralelas: *Projeto de Conteúdo e Produção* são tarefas desenvolvidas pelos membros *"não-técnicos"* da equipe. O objetivo destas tarefas é projetar, produzir e/ou adquirir todo o conteúdo de texto, gráfico, audio e vídeo, que deve ser integrado no aplicativo. Ao mesmo tempo, um conjunto de tarefas de projeto técnico é conduzido.

Geração de Página é uma atividade de construção que conta intensamente com o uso de ferramentas automáticas para a criação dos aplicativos Web. O conteúdo definido na atividade de engenharia é combinado com o projeto arquitetural, de navegação e de interface, para produzir páginas da Web executáveis em HTML, XML e outras linguagens orientadas a procedimentos (por exemplo, Java). A integração com componentes de *Middleware* (por exemplo: CORBA, DCOM ou *JavaBeans*) também é realizada durante essa atividade.

A atividade de *Teste* exercita a navegação do aplicativo Web; tenta descobrir erros de *applets*, *scripts* e formulários, e ajuda a garantir a correta operação do aplicativo em diferentes ambientes *(browsers)*.

Cada incremento produzido como parte do processo é revisto durante a *Avaliação do Cliente*. Neste ponto, mudanças são pedidas (extensões do escopo ocorrem). Estas mudanças serão integradas ao sistema no próximo ciclo do processo incremental.

McDonald e Welland [MW01] destacam que as equipes de desenvolvimento para aplicativos Web são normalmente menores que as equipes de desenvolvimento de software tradicional. Em ambos os casos as equipes são gerenciadas em pequenos grupos, mas a semelhança acaba aí, pois no desenvolvimento de software tradicional as equipes são divididas em unidades menores para resolver diferentes problemas e executar diferentes tarefas. Mas no desenvolvimento para a Web, as equipes são divididas em grupos multidisciplinares, que construirão diferentes seções do aplicativo para a Web, mas em geral irão trabalhar em problemas similares. No decorrer do desenvolvimento de um software tradicional, as equipes devem interagir entre si, isto normalmente é feito através de interfaces pré−definidas, com cada equipe vendo o trabalho das outras equipes como caixas−pretas Na engenharia para a Web as equipes precisam se comunicar ainda mais, com o objetivo de reduzir o esforço e garantir consistência.

Na teoria, a maioria das atividades de gerenciamento de projeto utilizadas na Engenharia de Software pode ser utilizada também na Engenharia para a Web. Mas na prática, a abordagem é consideravelmente diferente.

O desenvolvimento de aplicativos para a Web é uma área relativamente nova e há poucos dados históricos que podem ser utilizados para fazer estimativa. Até agora, nenhum tipo de métrica foi publicado e ainda há pouca discussão de como devem ser estas métricas. Com isso, estimativas são baseadas apenas em experiências com projetos similares. Mas quase todo aplicativo para a Web quer inovar em alguma coisa, oferecendo algo novo e diferente. Isto acaba fazendo com que estimativas baseadas em experiência com outros projetos, apesar de úteis, estejam sujeitas a uma alta margem de erro.

2.6 Alguns Métodos de Projeto de Aplicações Web

O uso de um modelo (ou grupo de modelos) durante o desenvolvimentos de aplicativos hipermídia para a Web visa ajudar a disciplinar a atividade de autoria [SCGP92], para que:

- Seja possível descrever a aplicação independentemente de sua implementação;
- A Evolução da aplicação seja mais facilmente fornecida, desde que esteja devidamente documentado as decisões de projetos no nível correto de abstrações;

Os métodos de projeto existentes estendem o modelo de dados hipermídia com primitivas de mais alto nível, oferecendo ferramentas conceituais para a construção de abstrações (no sentido de abstrações de dados, onde a implementação não é apenas ocultada, mas também relegada a uma etapa mais adiante).

Em HDM [GPS93], por exemplo, um esquema hipermídia é descrito como um conjunto de tipos de entidade e suas relações (elos aplicativos); as entidades são constituídas por componentes (hierarquicamente estruturados) e cada componente pode ser visto sob diferentes perspectivas através do uso de Unidades, o equivalente HDM dos nós hipermídia. Estruturas de Acesso, como índices e roteiros guiados, são utilizadas para especificar uma forma alternativa (normalmente atalhos) para acessar unidades de informação.

RMM [ISB95] estende HDM acrescentando estruturas de roteiros guiados e índices mais ricos, e definindo um modelo metodológico detalhado, onde o projeto é encaminhado a partir das entidades, através do projeto de relações e de navegação, terminando no projeto da interface do usuário e na implementação.

Em EORM [Lan94], as aplicações hipermídia são modeladas utilizando-se de uma extensão do modelo ORM [RBP$^+$91] e as relações são enriquecidas com semânticas de navegação.

Uma abordagem bastante distinta foi observada em Trellis [SF89] [SFR92]. A estrutura navegacional de uma aplicação hipermídia é descrita utilizando uma Rede de Petri. Neste caso, embora seja possível computar diferentes propriedades da aplicação através da análise da Rede de Petri correspondente, perde-se alguma força de modelagem devido à natureza das redes. Em [ZP92] por sua vez, as estruturas de navegação e de interface da aplicação são modeladas utilizando Statecharts [HPSS87]. Embora os Statecharts compartilhem de algumas propriedades das Redes de Petri, sua força expressiva é maior pelo fato de oferecerem mecanismos de aninhagem diferentes. Ambos os casos demonstram que as técnicas formais adequam-se bem aos projetos de aplicações hipermídia.

O que falta aos métodos acima é uma abordagem sistemática e abrangente, na qual todos os aspetos do empreendimento de projeto hipermídia sejam considerados e as decisões de projetos sejam gravadas com o intuito de serem mais tarde rastreadas. Ao mesmo tempo, enquanto Trellis e outras abordagens baseadas em máquinas de estado enfatizam modelagem navegacional e, em alguns casos, modelagem de interface, infelizmente ignoram as relações entre componentes hipermídia e suas contrapartes no mundo real. Ademais, nenhum dos métodos anteriormente mencionados oferece alto nível de princípios arquitetônicos para a constituição da estrutura geral de uma aplicação hipermídia [Ros96].

O OOHDM visa fornecer primitivas de projeto e mecanismos de abstração de alto nível. Visa, ainda, proporcionar uma transição simples e elegante das descrições de modelos abstratos de domínio para projetos de aplicações hipermídia concretos, ligando as abordagens modernas de engenharia de software às peculiaridades do campo das aplicações de hipermídia. Finalmente, deve ser independente da implementação, o que significa que deve possibilitar a construção de um projeto abstrato e, em seguida, uma implementação, usando diferentes plataformas de implementação [Ros96].

Capítulo

3

O método OOHDM

"Um modelo formal ajuda, na medida em que permite a abstração e a separação entre a estrutura e o conteúdo do hipertexto; ele nos proporciona uma melhor compreensão da semântica de navegação e fornece uma interpretação consistente do modelo de implementação" ***Zheng***

3.1 Introdução

Neste capítulo é apresentado o *Object Oriented Hypermedia Design Method (OOHDM)* [SR95] e suas atividades básicas. Este método permite passar de uma especificação de requisitos a uma aplicação em uso, modelando entidades do mundo real como objetos conceituais e, mais tarde, objetos de navegação e de interface.

3.2 Um panorama do Método OOHDM

OOHDM é um método de projeto para aplicações hipermídia que descreve as tarefas a serem executadas desde a análise de domínio da aplicação até a sua implementação. O método OOHDM tem se mostrado o método mais maduro, devido a sua ampla utilização pela comunidade hipermídia para projetos de aplicações em diversos países.

O OOHDM considera o processo de desenvolvimento da aplicação hipermídia como um processo de quatro atividades, desempenhadas iterativa e incrementalmente durante o desenvolvimento; em cada etapa um modelo é construído ou enriquecido.

ATIVIDADES	PRODUTOS	MECANISMOS	INTERESSES DO PROJETO
Modelagem Conceitual	Classes, sub-sistemas, relacionamentos, perspectivas de atributos	Classificação, composição,generalização e especialização	Modelagem da semântica do domínio de aplicação
Projeto de Navegação	Nós, elos, estruturas de acesso, contextos de navegação, transformações navegacionais	Mapeamento entre objetos conceituais e de navegação.Padrões de navegação para a descrição da estrutura geral da aplicação	Leva em conta o perfil do usuário e a tarefa; ênfase em aspetos cognitivos e arquiteturais
Projeto da Interfase Abstrata	Objetos de interface abstrata, reações a eventos externos, transformações de interface	Mapeamento entre objetos de navegação e objetos de interface	Modelagem de objetos perceptíveis,implementa metáforas escolhidas. Descrição de interface para objetos navegacionais
Implementação	Aplicação em execução	Aqueles fornecidos pelo ambiente alvo	Desempenho, completitude

Figura 3.1: Esboço da metodologia OOHDM [Ros96]

Na figura 3.1 estão resumidas as quatro atividades: modelagem de domínio ou conceitual, projeto navegacional, projeto de interface abstrata e implementação, assim como os produtos gerados em cada uma e os mecanismos de abstração utilizados ao longo do processo. As setas de avanço demonstram a evolução natural do empreendimento de projeto, enquanto as setas de retorno expressam não apenas a possibilidade de "*feedback loops*" na forma mencionada em [ISB95], mas também a existência de um modelo simples, porém potente, de rastreamento que permite mapear as modificações em um modelo para partes de um outro modelo (rastreamento para trás).

Durante a modelagem conceitual, um modelo do domínio da aplicação é construído utilizando-se princípios bem conhecidos de modelagem orientada a objetos [RBP+91] estendidos por primitivas, tais como perspectivas de atributo e subsistemas. As classes conceituais podem ser construídas utilizando-se hierarquias de agregação e de generalização/especialização. Não há, nesta etapa, preocupação com os tipos de usuários e tarefas, apenas com a semântica do domínio da aplicação, o produto desta etapa é um esquema de classes e objetos construídos a partir de Sub−sistemas, Classes e Relações.

Uma das características essenciais das aplicações hipermídia é a noção de navegação. No OOHDM, uma aplicação é vista como uma visão navegacional do modelo conceitual. Esta visão é construída durante o Projeto Navegacional levando em conta os tipos de usuários ao quais a aplicação se destina e o conjunto de tarefas que deverão desempenhar utilizando−o. Diferentes modelos navegacionais podem ser construídos para o mesmo esquema conceitual, expressando, desta forma, diferentes visões (aplicações) no mesmo domínio. A estrutura navegacional de uma aplicação hipermídia é descrita por meio da definição das classes navegacionais que refletem a visão escolhida sobre o domínio da aplicação. Esta estrutura é definida em termos de contextos navegacionais, que são induzidos (de diferentes maneiras, dependendo do tipo de contexto) a partir de classes navegacionais tais como Nós e Elos. Durante o Projeto Navegacional definimos, também a maneira como a navegação irá proceder, especificando transformações no espaço navegacional, isto é, do conjunto de objetos navegacionais acessíveis em cada momento.

Durante o Projeto de Interface Abstrata, um modelo de interface é construído. Este modelo especifica que objetos de interface serão vistos pelo usuário e, particularmente, a forma que tomarão diferentes objetos navegacionais, que objetos de interface ativarão a navegação, a maneira como os objetos de interface multimídia serão sincronizados e que transformações ocorrerão na interface.

Uma clara separação entre os dois interesses (a interface abstrata e o modelo navegacional) permite construir interfaces diferentes para o mesmo modelo navegacional adequando-se, portanto, às variações de necessidades e preferências dos usuários, ou de tecnologia de interface.

Finalmente, mapeando os modelos navegacionais e de interface no ambiente de implementação escolhido, o autor produz o sistema real de hipermídia a ser disponibilizado.

3.3 Projeto Conceitual

Em uma modelagem orientada a objetos procura-se representar os aspectos estruturais e comportamentais que serão mapeados para o ambiente de implementação como estruturas de classes com seus métodos [RBP$^+$91]. Nesta fase, o domínio da aplicação é analisado e descrito utilizando os princípios de modelagem orientada a objetos apresentada em [RBP+91] *(Object Modeling Technique - OMT).*

Em OOHDM, o esquema conceitual é constituído de classes, relações e subsistemas. As classes são descritas como em modelos orientados a objetos, mas seus atributos podem ser multi−tipados. As abstrações de Agregação e Generalização/Especialização são utilizadas para aumentar o poder de representação do sistema.

O esquema consiste de um conjunto de objetos e classes unidos entre si por relacionamentos; como os objetos são instâncias de classes, um relacionamento entre as classes faz com que o relacionamento entre os objetos seja abstraido.

As classes podem relacionar−se com subsistemas. Um subsistema pode ser auto−contido, possuindo um único ponto de entrada e saída. Neste caso, age como um *"servidor de informações"*.

Contudo, do ponto de vista da hipermídia, é importante salientar que os relacionamentos não devem ser escondidos nos atributos de classes. Isto significa que no caso de um atributo representar uma entidade conceitual complexa a ser explorada na hipermídia final, deve−se especificar um relacionamento. Os relacionamentos, assim como as classes, podem conter atributos. Os Atributos de classes são tipados e representam propriedades intrínsecas ou conceituais dos objetos.

A modelagem oferece três mecanismos de abstração para lidar com a complexidade: Agregação, Generalização/Especialização e um conceito de empacotamento, os subsistemas. O primeiro é útil na descrição de Classes complexas como agregadas de classes mais simples. Já o segundo é utilizado na construção de Hierarquias de Classes e no uso de herança como um mecanismo de compartilhamento. Os subsistemas são um mecanismo de empacotamento para a abstração de modelos de domínio complexos. As classes herdam de suas super−classes atributos, estrutura de partes, relacionamentos e comportamento.

A figura 3.2 apresenta um exemplo de esquema conceitual para a aplicação "Guia de Profissões".

Podemos resumir nos seguintes itens os produtos da atividade de modelagem:

- Um esquema conceitual *(um conjunto de esquemas conceituais, sendo um para cada subsistema)*;

- Um conjunto de definições de subsistemas, classes, objetos e relacionamentos *(utilizando Cartões de Subsistemas, Classes, Objetos e Relacionamentos).*

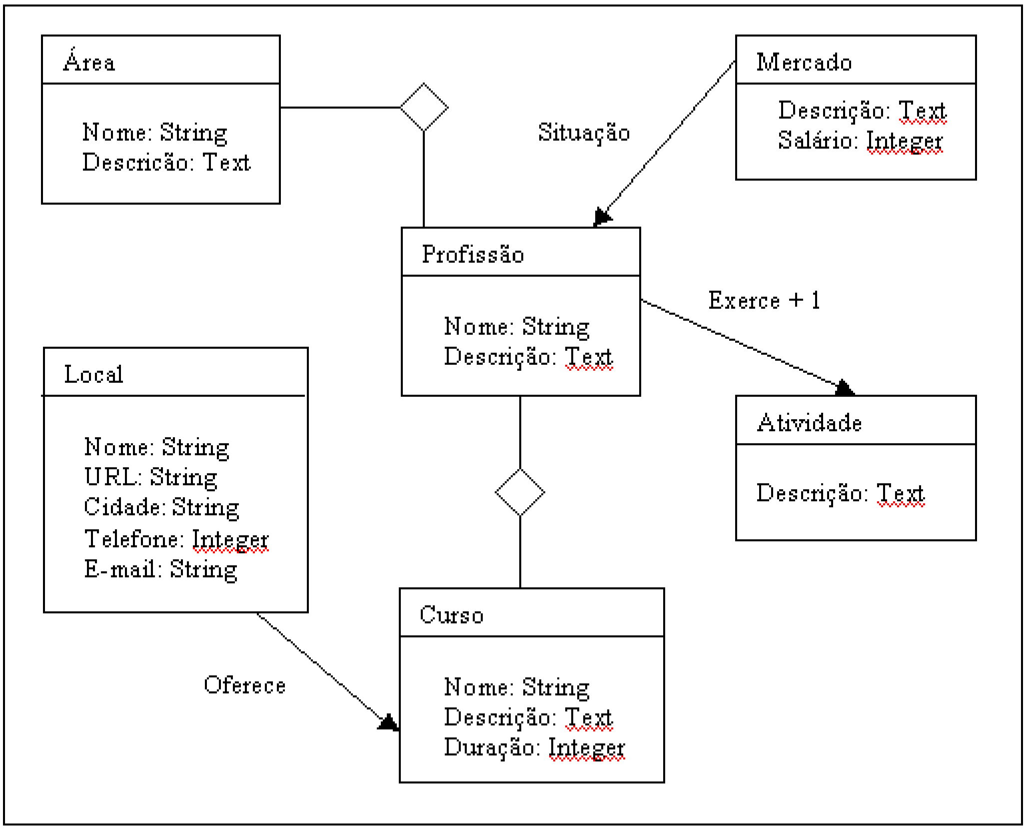

Figura 3.2: Esquema Conceitual da aplicação "Guia de Profissões"

3.4 Projeto Navegacional

Ao projetar a estrutura de navegação de um aplicativo de hipermídia, vários aspectos serão considerados, tais como:

- Quais objetos serão navegados e que atributos possuem? Quais os relacionamentos entre estes objetos e aqueles definidos no esquema conceitual? Isto será feito pela definição de nós e elos como visões orientadas a objetos das classes e relacionamentos conceituais.

- Qual é a estrutura subjacente de navegação? Em que contextos o usuário irá navegar? Conceito de contextos navegacionais, uma primitiva arquitetônica para a organização do espaço de navegação.

- É preciso decidir se os objetos navegados poderão ter aparências diferentes de acordo com o contexto em que são visitados e especificar tais diferenças claramente.

- Quais as estruturas de elo existentes entre os objetos que serão navegados (elos, caminhos, índices, roteiros guiados, etc.)?

- Como procede a navegação quando o usuário "pula" de um assunto a outro, isto é, qual o efeito da navegação nos objetos de origem e alvo e em outros objetos relacionados?

A figura 3.3 mostra o esquema de navegação da aplicação "Guia de Profissões".

Os Contextos Navegacionais são um conjunto de nós, elos e outros contextos navegacionais *(aninhados)* que auxiliam na organização dos objetos navegacionais, fornecendo espaços de navegação consistentes e, deste modo, diminuindo as chances de o usuário "perder−se no hiperespaço".

Os nós podem ser atômicos ou compostos e são descritos através de atributos e âncoras para elos no caso de serem atômicos, e como um conjunto de nós componentes no caso de serem compostos. Os Contextos Navegacionais definem um conjunto de classes "decoradoras" que adaptam cada nó no contexto para as estruturas âncoras−atributos necessárias ao contexto.

Os elos fazem a ligação entre os objetos navegacionais e podem ser de um−para−um ou um−para−muitos. O resultado da travessia de um elo é expresso pela definição das

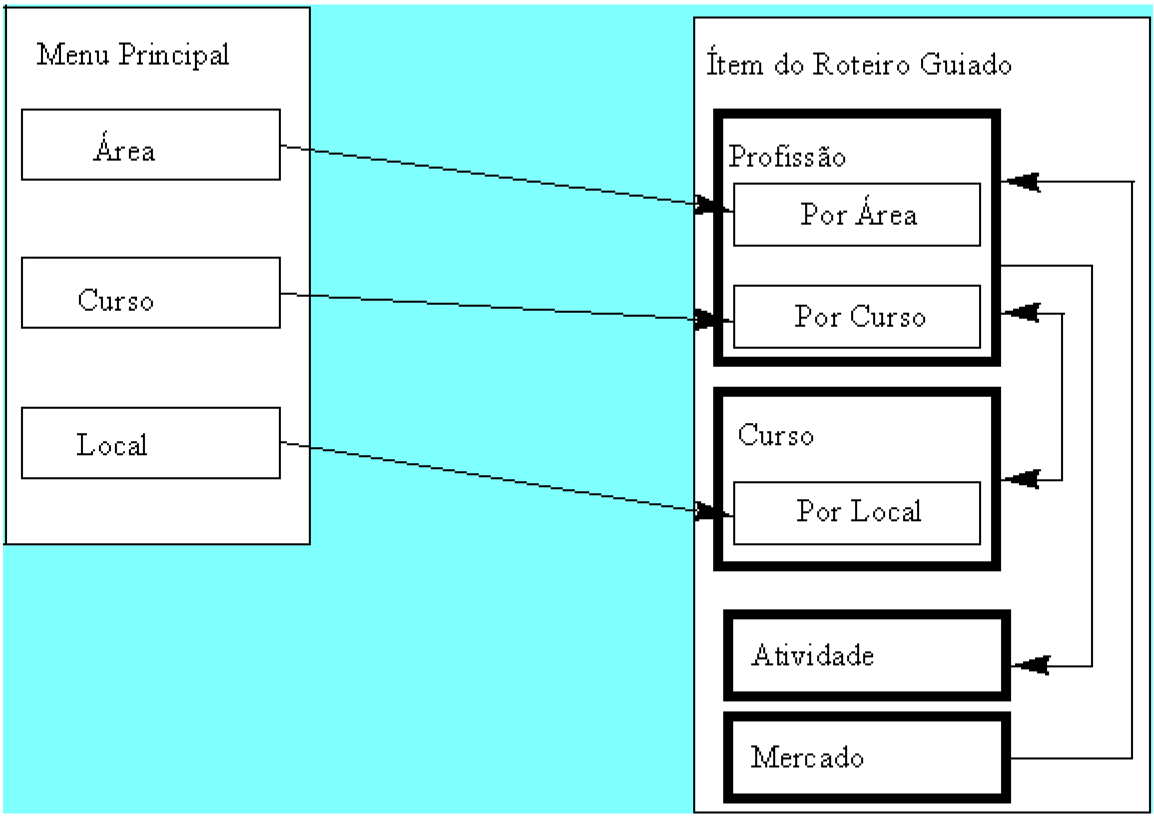

Figura 3.3: Esquema Navegacional da Aplicação “Guia de Profissões”

semânticas navegacionais, como resultado do comportamento do elo. As estruturas de acesso *(tais como os índices ou roteiros guiados)* são também definidas como classes e apresentam modos alternativos para a navegação no aplicativo hipermídia.

As classes e contextos navegacionais definem a estrutura estática da aplicação hipermídia. Contudo, é necessário também, especificar os aspectos dinâmicos da navegação. Em OOHDM, isto é feito pelo uso de Diagramas de Navegação, um modelo baseado em máquinas de estado no qual são descritas as transformações no estado do espaço navegacional.

3.5 Projeto de Interface Abstrata

Para especificar o modelo abstrato de interface é necessário definir metáforas de interface e descrever suas propriedades estáticas e dinâmicas, assim como seus relacionamentos com o modelo navegacional de uma forma independente de implementação. É preciso especificar:

- a aparência de cada objeto navegacional que será percebido pelo usuário, isto é, a representação de seus atributos (incluindo as âncoras);
- os outros objetos de interface para oferecer as diversas funções da aplicação, como barras de menus, botões de controle e menus;
- os relacionamentos entre os objetos de interface e os navegacionais, descrevendo assim o modo com que um evento externo, afetará a navegação;
- as transformações de interface ocorridas pelo efeito da navegação ou de eventos externos no comportamento de diferentes objetos de interface;
- a sincronização de alguns objetos de interface, especialmente quando há meios dinâmicos, como áudio e vídeo, envolvidos.

Em OOHDM, *Abstract Data Views* (ADV) são usados para especificar o modelo de interface abstrata. O modelo de projeto ADV foi criado originalmente para especificar, clara e formalmente, a separação entre a interface do usuário e os componentes de um sistema de software, e para oferecer um método de projeto independente de implementação, gerando graus mais altos de reuso de componentes de projeto e de interface.

Os ADVs são geralmente usados para representar interfaces entre duas mídias diferentes. Um ADV, quando usado em um projeto de aplicação hipermídia, pode ser visto como um objeto de interface contendo um conjunto de atributos que definem suas propriedades de percepção e o conjunto de eventos com os quais pode lidar, como os eventos gerados pelo usuário. Valores de atributos podem ser definidos como constantes, definindo, então, um estilo específico de aparência como posição, cor ou som.

Os Diagramas de Configuração são úteis na expressão de padrões de comunicação entre objetos, em termos de serviços oferecidos e requeridos. No contexto de OOHDM, o interesse concentra-se no modo como o usuário irá interagir com o aplicação hipermídia e, especialmente, em quais objetos de interface causarão a navegação.

Os ADVcharts (notação em diagramas seguindo a noção dos ADVs) suportam o aninhamento de estados e ADVs, permitindo a expressão da associação entre eventos externos e ADVs. Os ADVcharts expressam os diagramas de navegação, isto é, descrevem as transformações no nível da interface de usuário e seu impacto nos objetos navegacionais. Um ADVchart é composto por ADVs, estados, atributos e transições. O aninhamento de ADVs permite mostrar a estrutura de agregação dos objetos de interface.

A figura 3.4 mostra o diagrama de configuração da classe Profissão da aplicação exemplo.

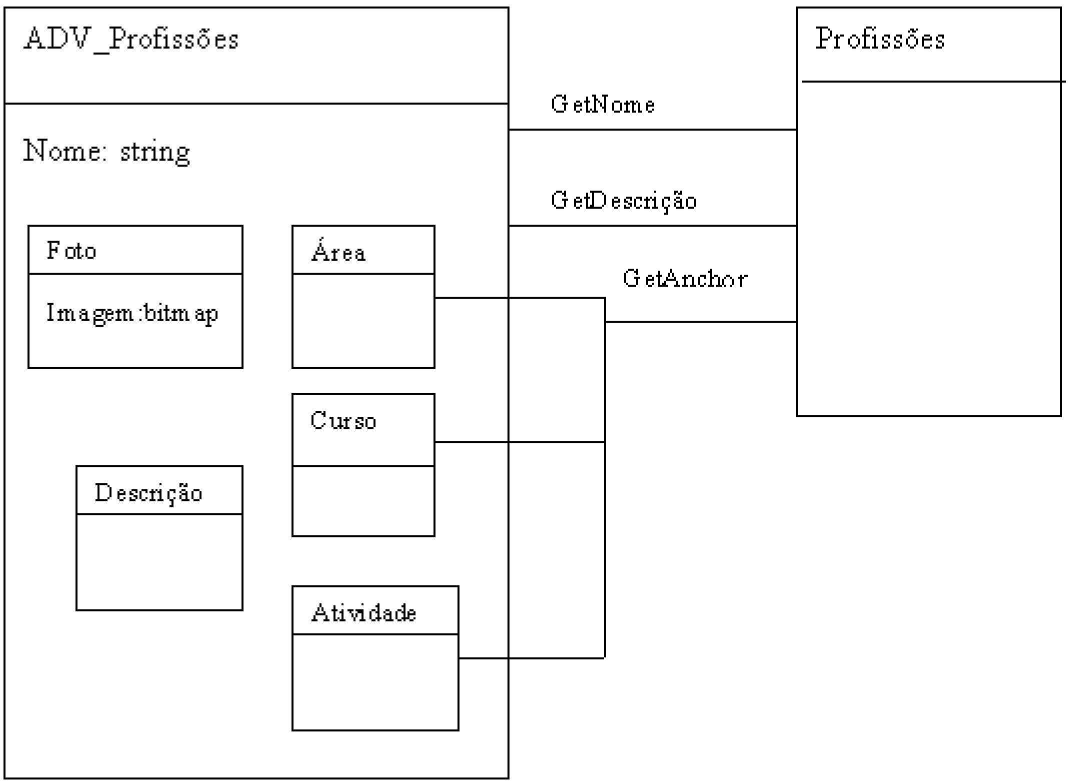

Figura 3.4: Diagrama de Configuração da classe Profissão - “Guia de Profissões”

A definição do modelo de interface de uma aplicação hipermídia com OOHDM implica em:

- definir a estrutura geral da interface da aplicação;
- definir, em cada nó, objetos de interface apropriados para atributos, âncoras, etc.;
- especificar o diagrama de configuração, mostrando relacionamentos estáticos entre ADVs e ADOs (*Abstract Data Objects* - sendo estes objetos navegacionais);
- mostrar relacionamentos estáticos entre os componentes de um ADV. Deve ser possível expressar todas as restrições estruturais existentes entre os objetos de interface;
- especificar, para cada ADV significativo, o ADVchart mostrando a dinâmica da aplicação hipermídia

Deve estar claro que, apesar de ser independente de implementação, a especificação da interface abstrata deve considerar certos aspectos do ambiente de implementação alvo, para que a especificação seja realista.

3.6 Implementação

A implementação de uma aplicação hipermídia de modo a resultar em um aplicativo de fato utilizável não é tarefa simples. Muitas questões técnicas e não−técnicas devem ser resolvidas. Uma vez que o ambiente de implementação tenha sido escolhido, o projeto deve ser mapeado para artefatos de implementação e todos os componentes hipermídia devem ser instanciados.

Basicamente, é necessário definir os objetos de interface de acordo com a especificação da interface abstrata, implementar transformações da forma como foram definidas nos ADVcharts e fornecer suporte para a navegação através da rede hipermídia.

A informação fornecida pela especificação baseada em ADVs e, em especial, a estrutura aninhada dos ADVs, oferece uma indicação sobre quais os objetos de interface que precisam ser definidos.

Os aspectos dinâmicos da interface lidam tanto com transformações de interface dentro de um ADV (por exemplo, mostrando/ocultando algum atributo), como com transformações de interface envolvendo navegação. Em ambos os casos, a natureza orientada a eventos dos ADVcharts e o poder expressivo de seu modelo de transição de estados permite a implementação destas transformações em um estilo simples, de baixo para cima.

Em aplicações hipermídia, a navegação ocorre porque o usuário seleciona uma âncora em um nó. É possível implementar comportamento navegacional mais sofisticado, como, por exemplo, navegação baseada em tempo, ou baseada em mídia (quando um evento determinado ocorre na mesma mídia, podemos proceder de modo a seguir um elo). Ambos os tipos de comportamento navegacional são especificados através de ADVcharts, já que as transições de estado e o evento gerador de cada mudança estão documentados na especificação da transição.

Para manter a informação de rastreabilidade acumulada durante o processo do projeto, devemos documentar, também, as decisões de implementação. Apesar de esta documentação poder ser facilmente automatizada e introduzida no ambiente de implementação, podemos gerar cartões simples ao implementar a aplicação.

Ambientes de desenvolvimento para aplicações hipermídia tem sido implementados, visando o suporte completo de desenvolvimento de aplicações hipermídia orientada a objetos a partir de uma modelagem totalmente orientada a objetos. Durante a realização do estudo de caso reportado neste relatório, foram estudados alguns artigos que descrevessem tais ambientes– KATIANA, e a seguir são brevemente apresentados.

O **OOHDM−XWeb** [SM01] é um ambiente de desenvolvimento criado para suporte à implementação de aplicações hipermídia, projetadas de acordo com as primitivas do método OOHDM, a partir de sua especificação na linguagem declarativa **OOHDM−ML** [SM01].

O ambiente **OOHDM−Web** 2.0 [SPM99] é baseado na linguagem Lua e no ambiente CGILua [HBI97]. Com o OOHDM−Web, o projetista cria páginas (*templates*) que são uma combinação de HTML com chamadas a funções de uma biblioteca. Essas funções acessam os objetos navegacionais armazenados em um banco de dados relacional.

Web−Composition Repository [GWG97]é uma ferramenta chave para o acesso sistemático de reuso de código. É usado para armazenar, gerenciar e reusar um grande número de componentes **WCML**. Conseqüentemente, facilita o reuso de componentes da Engenharia de Web.

Cocoon[1] é um *framework* Java para o desenvolvimento de aplicações Web. Consiste de uma ferramenta *opensource* desenvolvida pela *Apache Software Foundation*; o Projeto

[1] http://www.ibm.com/developerWorks

Cocoon foi originalmente proposto e implementado em janeiro de 1999 por Stefano Mazzocchi; começou como um *servlet* para a aplicação de estilos XSL estáticos, mas tem se tornado cada vez mais poderoso conforme novas características são adicionadas.

O Cocoon permite a separação de conteúdo *(informação real oferecida por um web site (ex. lista de preços))*, layout *(informação aplicada ao conteúdo para mostrá–lo)* e lógica *(funcionalidade necessária para fornecer serviços e interação dinâmica aos usuários)* em áreas distintas. Em termos práticos, no contexto do Cocoon, é possível associar conteúdo a XML, apresentação a XSL e lógica a XSP.

Diferentemente do HTML, as tags XML podem ser criadas para expressar com precisão o conteúdo de um documento. Já o HTML define tags para dizer como mostrar graficamente esse conteúdo *(informação de layout)*. Cocoon usa a informação contida em arquivos XSL para aplicar a informação de layout no conteúdo descrito por um arquivo XML. Porém, um arquivo XML também pode ser transformado em outro arquivo XML sem informação de layout. De fato, um arquivo XML pode passar por sucessivas transformações antes de ser gerado HTML. Isso é útil para capturar dados externos que utilizam tags que não são entendidas. Através das transformações intermediárias, esses dados externos vão sendo "traduzidos" para uma estrutura conhecida. Obter esses dados com o Cocoon e mostrar numa página com o estilo do site é bastante fácil, e uma vez feito, é igual para várias outras fontes de dados que utilizam o mesmo formato.

Cocoon foi projetado especialmente para permitir trabalho conjunto de **desenvolvedores** *(criam a lógica e modelos de objeto, cuidam das funcionalidades (actions))*, **analistas de negócios** *(se preocupam com que deve ser apresentado e com layout (generators XML))*, **designers** *(cuidam da aparência final (transformers XSLT))* e **administradores** *(cuidam do sitemap que mapeia os diferentes* ***pipelines*** *no Cocoon)*. Isto traz como grande vantagem, o fato de uma pessoa não prejudicar o trabalho da outra, além de evitar o "retrabalho", principalmente quando é necessário inserir mudanças ou novas funcionalidades na aplicação web, garantindo uma boa Engenharia de Web: *flexibilidade*, *manutenibilidade* e *escalabilidade*.

O conceito de separação de conteúdo, *layout* e lógica permite o planejamento, o desenvolvimento e a manutenção dessas três áreas independentemente. Com isso, é possível a reutilização cada vez maior do trabalho realizado entre essas diferentes áreas, o que permite atualizações muito mais rápidas e diminui o tempo total de desenvolvimento. Este conceito é muito valorizado, pois mudanças são necessárias e como a *Web Engineering* é o processo de "projetar para mudar" *(evolução contínua)*, as vantagens, que o Cocoon oferece, tornam mais flexível o processo de desenvolvimento da aplicação.

Capítulo

4

Modelagem

4.1 Introdução

A maioria dos atuais métodos de projeto hipermídia, como HDM, RMM, EORM, ou até mesmo versões do OOHDM, fornecem ao projetista, modelos e as correspondentes notações para especificar o projeto e implementação de aplicações. Mas ajudam pouco em relação ao levantamento de requisitos. O método apresentado em [GSV00] tenta resolver este problema, empregando para tal **Cenários** [Car95], **Casos de Uso** [JBR99] e *User Interaction Diagrams (**UIDs**)* [VSdS00b] [VSdS00a]. O método mostra como coletar informações e como sintetizá-las em um Projeto Conceitual e em um Projeto Navegacional para uma aplicação com tais requisitos, como representado na figura 4.1.

Nas próximas seções são apresentados os passos para as fases da área sombreada da figura 4.1 utilizando o método apresentado em [GSV00], além da modelagem de um aplicativo Web para uma fundição como exemplo que foi elaborada por – VIviane.

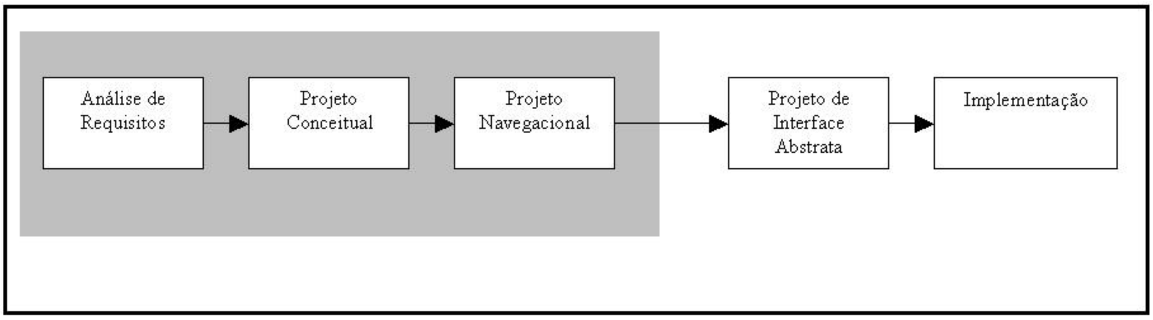

Figura 4.1: Fases do Desenvolvimento e Implementação de Aplicações Web [GSV00]

4.2 Contextualização da aplicação

O estudo de caso realizado foi o de desenvolvimento de um Sistema de Informação de uma fundição cativa de grande porte, cujo domínio de informações se refere ao problema da programação da produção. Atualmente, as decisões tomadas na fundição são baseadas apenas no bom senso do engenheiro de produção. Um modelo matemático de programação inteira mista foi proposto e resolvido com o auxílio de um pacote de otimização inteira. A partir dos resultados computacionais obtidos, foi possível concluir que o modelo matemático proposto representa bem o problema de programação da produção na fundição estudada e permite um considerável aumento na produção [Ara03].

Para resolver este modelo, foi utilizada a linguagem de modelagem AMPL [FGK] com o *solver* CPLEX 7.5 [S.A], que não tem uma interface gráfica muito amigável. O objetivo então deste estudo de caso foi o de criar uma interface amigável para esse sistema de informação (considerando-se ainda que o sistema seria utilizado via Web) de modo a viabilizar a implantação desta abordagem de solução na prática industrial.

Os dados necessários para a resolução do modelo matemático e portanto, para a aplicação, são:

- os nomes das peças a serem produzidas,
- o tipo de liga de que a peça é feita (são duas ligas),
- quantidade em estoque de cada peça,
- quantidade em atraso de cada peça,
- demanda de cada peça (fixa em um mês),

- estoque máximo de cada peça, quantidade máxima por lote de cada peça,
- custo de estocagem de cada peça,
- penalidade por atraso de cada peça,
- peso de cada árvore,
- número de cavidades de cada árvore,
- custo e capacidade de produção de determinada peça em determinada máquina (são três máquinas),
- número de horas de funcionamento de cada máquina (o primeiro dia é dividido em três turnos e os 4 dias seguintes possuem valores iguais),
- número de horas de paralisação de cada máquina (o primeiro dia é dividido em três turnos e os 4 dias seguintes possuem valores iguais),
- número de horas de todos os fornos (o primeiro dia é dividido em três turnos e os 4 dias seguintes possuem valores iguais),
- capacidade total de armazenamento,
- capacidade de armazenamento de cada liga,
- capacidade total de reposição e
- capacidade de reposição de cada liga.

4.3 Levantamento de Requisitos e Projeto Conceitual

Notadamente, os projetos que são bem sucedidos começam com um adequado levantamento de requisitos dos usuários e outros interessados. O levantamento de requisitos, segundo o método [GSV00] é dividido nas seguintes fases (figura 4.2): identificação dos atores e tarefas, especificação dos cenários, especificação dos casos de uso, especificação dos UIDs, e validação dos casos de uso e dos UIDs. O projeto conceitual apresenta uma única fase: a de especificação do Esquema Conceitual.

Nas subseções a seguir, são apresentadas todas as fases referentes ao levantamento de requisitos.

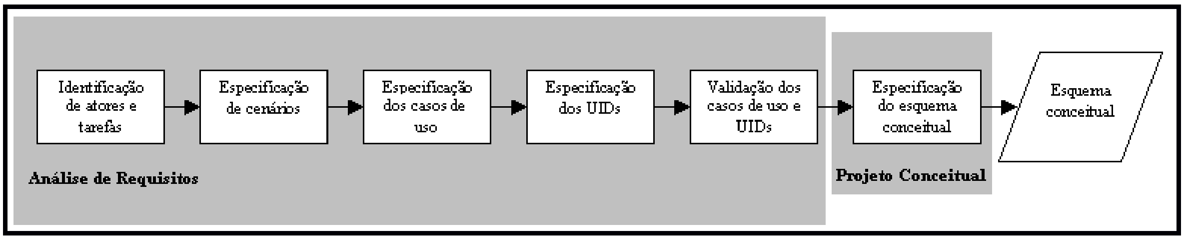

Figura 4.2: Fases da Análise de Requisitos e Projeto Conceitual [GSV00]

4.3.1 Identificação dos Atores e Tarefas

Nesta fase, o projetista interage com o domínio para identificar os atores e as tarefas que serão apoiadas pela aplicação. Esta interação é realizada por meio de análises de documentos e entrevistas com os usuários.

Atores são entidades que trocam informações com a aplicação. Um ator representa um papel específico de um usuário da aplicação. Na aplicação para a fundição, para o primeiro estágio de informações a serem supridas, foi identificado somente um ator:

- Usuário 1 – Funcionário da fábrica

De fato, foi também observado que um segundo ator-usuário, responsável por cadastrar os funcionários da fábrica, seria necessário. No entanto, as tarefas que este responsável executaria foram alocadas ao próprio desenvolvedor do Banco de Dados da aplicação. Posteriormente, outras tarefas administrativas e mesmo de consultas aos dados fornecidos pela aplicação, irão requerer novas versões da aplicação.

O termo Tarefa significa o objetivo que o usuário deseja alcançar usando a aplicação, representam potenciais cenários. As tarefas que foram identificadas como necessárias para o Sistema de Informação da fundição foram:

- Inserir parâmetros
- Excluir parâmetros
- Alterar parâmetros
- Consultar parâmetros
- Inserir peça
- Excluir peça

- Alterar peça
- Consultar peças
- Solicitar cálculo da produção de peças
- Confirmar peças produzidas
- Consultar peças produzidas

4.3.2 Especificação dos Cenários

Nesta fase cada usuário especifica textualmente, ou verbalmente, os cenários que descrevem suas tarefas, sendo especificadas assim, as informações que serão trocadas entre o usuário e a aplicação, não sendo necessário abordar aspectos de interface nem de navegação. Cenários são descrições narrativas de como a aplicação precisa ser utilizada [Car95], e pode ser especificada tanto por usuários quanto pelo projetista.

As tarefas identificadas na fase anterior podem ajudar os usuários na determinação dos cenários relevantes. A descrição de cada um dos cenários utilizados pela aplicação do estudo de caso encontram-se a seguir.

Cenário 1 (Usuário 1): Entrar no sistema.

- **Contexto:** Sou usuário do sistema de cálculo da produção de peças, para utilizá-lo devo entrar com uma senha que deve ser validada pelo sistema.
- **Objetivo:** Entrar no sistema de cálculo da produção de peças.
- **Descrição:** Eu entro com o meu login e minha senha, se forem válidos, é permitida a minha entrada no sistema.

Cenário 2 (Usuário 1): Inserir peça.

- **Contexto:** Fui entrar com os parâmetros das peças e não encontrei a peça desejada na lista de peças, então preciso incluir esta peça.
- **Objetivo:** Incluir uma nova peça.
- **Descrição:** Escolho a opção para inserir peças, entro com o nome da peça, liga de que a peça é feita, o custo e a capacidade de produção para cada máquina, quanto eu validar a inserção, a aplicação deve guardar os dados.

Cenário 3 (Usuário 1): Excluir peça.

- **Contexto:** Quando eu cadastrar uma peça erroneamente, devo ter a oportunidade de poder excluí-la.

- **Objetivo:** Excluir uma peça que foi inserida erroneamente.

- **Descrição:** Escolho a opção para excluir peças e entro com o nome da peça, é interessante que eu veja de que liga a peça é feita e ao validar a exclusão, a aplicação exclui os dados.

Cenário 4 (Usuário 1): Alterar peça.

- **Contexto:** Quando eu cadastrar uma peça erroneamente, devo ter a oportunidade de poder alterá-la.

- **Objetivo:** Alterar dados de uma peça que foram inseridos erroneamente.

- **Descrição:** Escolho a opção para alteração de peças e entro com o nome da peça, é interessante que eu veja de que liga a peça é feita e o custo e a capacidade de produção para cada máquina. Quando eu confirmar a alteração, a aplicação deve alterar os dados.

Cenário 5 (Usuário 1): Consultar peças.

- **Contexto:** Pode ser necessário, eu visualizar todas as peças cadastradas.

- **Objetivo:** Consultar peças cadastradas.

- **Descrição:** Solicito visualizar as peças cadastradas, e a aplicação exibe todas as peças. É interessante que seja exibido a respectiva liga de que cada peça é feita, e imprimir a consulta também.

Cenário 6 (Usuário 1): Entrar com os dados das peças que são utilizados como parâmetro.

- **Contexto:** Quero entrar com os dados referentes às peças que são necessários para calcular quais peças deverão ser produzidas, em quais turnos e em quais máquinas. Como a produção das peças é fixa no período de um mês, os dados deverão ser inseridos a cada mês.

- **Objetivo:** Entrar com os dados referentes às peças, utilizados na resolução do modelo matemático, para um novo mês de trabalho.

- **Descrição:** Escolho a opção para entrar com os parâmetros de peças, entro com o mês, o ano e os dados referentes às peças do novo mês de trabalho. Os dados que deverei inserir são: nome da peça (caso a peça não esteja na lista de nomes de peças, o usuário deverá inserir a peça antes), quantidade em estoque de cada peça, quantidade em atraso de cada peça, demanda de cada peça, estoque máximo de cada peça, quantidade máxima por lote de cada peça, custo de estocagem de cada peça, penalidade por atraso de cada peça, peso da árvore e número de cavidades da árvore. Ao confirmar a inserção, a aplicação deve armazenar os dados e novos dados de peças poderão ser inseridos.

Cenário 7 (Usuário 1): Excluir dados que são utilizados como parâmetro.

- **Contexto:** Posso ter entrado com uma peça erroneamente em um mês de trabalho, então é necessário que eu tenha a possibilidade de excluí-la.
- **Objetivo:** Excluir dados de uma peça que são utilizados como parâmetro.
- **Descrição:** Escolho a opção para excluir peças que são utilizadas como parâmetros, entro com o mês, o ano e escolho o nome da peça que desejo excluir. Os dados da peça (nome da peça, quantidade em estoque, quantidade em atraso, demanda, estoque máximo, quantidade máxima por lote, custo de estocagem, penalidade por atraso, peso da árvore e número de cavidades da árvore) são exibidos. Se eu fizer a confirmação de exclusão, a aplicação deve excluir os dados.

Cenário 8 (Usuário 1): Alterar dados de alguma peça que são utilizadas como parâmetro.

- **Contexto:** Inseri erroneamente dados de alguma peça que são utilizados como parâmetro, então devo ter a possibilidade e alterá-los.
- **Objetivo:** Alterar dados de peças que foram inseridos erroneamente.
- **Descrição:** Escolho a opção para alterar peças que são utilizadas como parâmetros, entro com o mês e ano desejados e escolho o nome da peça. É interessante que os dados sejam exibidos: quantidade em estoque, quantidade em atraso, demanda, estoque máximo, quantidade máxima por lote, custo de estocagem, penalidade por atraso, peso da árvore e número de cavidades da árvore. Quando eu validar a alteração, a aplicação deve alterar os dados.

Cenário 9 (Usuário 1): Alterar dados do custo de produção.

- **Contexto:** Às vezes é necessário que eu altere os dados do custo de produção, ou porque foram inseridos erroneamente ou para melhorar a solução da programação da produção.
- **Objetivo:** Alterar dados do custo de produção.

- **Descrição:** Seleciono a opção de alteração do custo da produção e a aplicação deve exibir o custo de produção de todas as peças por máquina e permitir alteração. Caso eu confirme, a aplicação deve alterar os dados.

Cenário 10 (Usuário 1): Alterar dados da capacidade de produção.

- **Contexto:** Às vezes é necessário que eu altere os dados da capacidade de produção, ou porque foram inseridos erroneamente ou para melhorar a solução da programação da produção.
- **Objetivo:** Alterar dados da capacidade de produção.
- **Descrição:** Seleciono a opção para alterar a capacidade de produção e a aplicação deve exibir a capacidade de produção de todas as peças por máquina e permitir a alteração. Caso eu confirme, a aplicação deve alterar os dados.

Cenário 11 (Usuário 1): Alterar o número de horas do funcionamento das máquinas.

- **Contexto:** Às vezes é necessário que eu altere o número de horas de funcionamento de alguma máquina, ou porque foram inseridos erroneamente ou para melhorar a solução da programação da produção.
- **Objetivo:** Alterar dados do número de horas de funcionamento das máquinas.
- **Descrição:** Seleciono a opção para a alteração e a aplicação deve exibir o número de horas de funcionamento de todas as máquinas por turnos e dias e permitir a alteração. Caso eu confirme, a aplicação deve alterar os dados.

Cenário 12 (Usuário 1): Alterar o número de horas de paralisação das máquinas.

- **Contexto:** Às vezes é necessário que eu altere o número de horas de paralisação de alguma máquina, ou porque foram inseridos erroneamente ou para melhorar a solução da programação da produção.
- **Objetivo:** Alterar o número de horas de paralisação das máquinas.
- **Descrição:** Seleciono a opção para a alteração e a aplicação deve exibir o número de horas de paralisação de todas as máquinas por turnos e dias e permitir a alteração. Caso eu confirme, a aplicação deve alterar os dados.

Cenário 13 (Usuário 1): Alterar o número de horas dos fornos.

- **Contexto:** Às vezes é necessário que eu altere o número de horas dos fornos, ou porque foram inseridos erroneamente ou para melhorar a solução da programação da produção.
- **Objetivo:** Alterar o número de horas dos fornos.
- **Descrição:** Seleciono a opção para a alteração e a aplicação deve exibir o número de horas dos fornos por turnos e dias e é permitir a alteração. Caso eu confirme, a aplicação deve alterar os dados.

Cenário 14 (Usuário 1): Alterar dados das capacidades que são utilizados como parâmetro.

- **Contexto:** Às vezes é necessário que eu altere os dados das capacidades, ou porque foram inseridos erroneamente ou para melhorar a solução da programação da produção.
- **Objetivo:** Alterar os dados das capacidades.
- **Descrição:** Seleciono a opção para a alteração e a aplicação deve exibir a capacidade total de armazenamento, capacidade de armazenamento de cada liga, capacidade total de reposição e capacidade de reposição de cada liga e permitir alteração. Caso eu confirme, a aplicação deve alterar os dados.

Cenário 15 (Usuário 1): Alterar Mês e/ou Ano.

- **Contexto:** Posso ter entrado com os dados de Mês e/ou Ano errados e devo ter a possibilidade de alterá-los.
- **Objetivo:** Alterar Mês e/ou Ano.
- **Descrição:** Seleciono a opção para a alteração, entro com o mês e ano antigos e depois com o mês e ano novos. Quando eu confirmar a alteração, a aplicação deve alterar os dados.

Cenário 16 (Usuário 1): Consultar os dados que são utilizados como parâmetro.

- **Contexto:** É necessário que eu possa consultar os dados que são utilizados como parâmetros em determinado mês de produção.
- **Objetivo:** Consultar os dados que são utilizados como parâmetro.
- **Descrição:** Seleciono a opção de consulta, entro com o mês e ano desejados e a aplicação deve exibir os dados. Seria interessante que os dados fossem exibidos por partes: primeiramente os dados das peças (nome das peças, quantidade em estoque de cada peça, quantidade em atraso de cada peça, demanda de cada peça, estoque máximo de cada peça, quantidade

máxima por lote de cada peça, custo de estocagem de cada peça, penalidade por atraso de cada peça, peso da árvore e nro de cavidades), depois dados do custo de produção (custo de produção de todas as peças por máquina), depois dados da capacidade de produção (capacidade de produção de todas as peças por máquina), e por fim dados do nro de horas (nro de horas de funcionamento de todas as máquinas por turnos e dias, nro de horas de paralisação de todas as máquinas por turnos e dias e nro de horas dos fornos por turnos e dias) e das capacidades (capacidade total de armazenamento, capacidade de armazenamento de cada liga, capacidade total de reposição e capacidade de reposição de cada liga).

Cenário 17 (Usuário 1): Solicitar o cálculo da produção das peças.

- **Contexto:** Desejo calcular as peças que deverão ser produzidas no dia corrente de trabalho, para tanto, preciso solicitar que o cálculo seja feito.
- **Objetivo:** Calcular a produção das peças para um dia, do mês atual.
- **Descrição:** Escolho a opção para o cálculo da produção, entro com o dia e solicito o cálculo. A aplicação deve retornar as peças sugeridas para produção, de acordo com turno e máquina, é interessante que eu tenha a possibilidade de alterar o resultado das peças calculadas, validar a produção e também imprimir a sugestão de produção das peças.

Cenário 18 (Usuário 1): Confirmar produção.

- **Contexto:** As peças já foram produzidas e desejo confirmar um dia de produção. Posso alterar dados de peças que foram sugeridas para produção, mas que não foram produzidas.
- **Objetivo:** Confirmar um dia de produção.
- **Descrição:** Seleciono a opção para confirmar a produção e entro com o dia. A aplicação deve exibir a lista de peças sugeridas para produção e permitir a alteração dessas quantidades. Se eu fizer a confirmação da produção, a aplicação atualiza os dados das peças.

Cenário 19 (Usuário 1): Consultar as peças que foram produzidas.

- **Contexto:** Às vezes preciso saber como anda o desempenho do setor de produção de peças, então é necessário fazer algumas consultas.
- **Objetivo:** Visualizar de maneira mais amigável os dados da produção de peças.
- **Descrição:** Seleciono a opção para consultas, digito o dia, mês e ano desejados e a aplicação deve exibir as seguintes consultas: para cada peça a quantidade que está atrasada, para cada peça a quantidade que foi produzida por turno e por máquina.

4.3.3 Especificação dos Casos de Uso

Um caso de uso é a forma de utilização da aplicação [JBR99]. Assim, os casos de uso apresentam a interação entre o usuário e a aplicação, sem considerar os aspectos internos da aplicação.

Cenários que descrevem a mesma tarefa são agrupados em um único caso de uso. A descrição do caso de uso deve incluir a informação apresentada em todos os cenários relatados. O projetista deve identificar quais itens de dados mostrados no cenário são relevantes. Um caso de uso pode incluir informações de outros casos de uso.

Se vários tipos de usuários executam a mesma tarefa, os cenários para estes tipos de usuários são agrupados em um caso de uso, identificando os papéis aos quais eles pertencem. A seguir são apresentados todos os casos de uso da aplicação:

Caso de uso 1: Inserir peça.

- Cenários: 1.1 / 1.2
- Descrição:
  1. O usuário entra com uma senha.
  2. A aplicação valida a senha.
  3. Se a senha for válida, o usuário entra na aplicação.
  4. O usuário seleciona Incluir peça.
  5. O usuário entra com os dados (nome da peça, a que tipo de liga pertence a peça e o custo e capacidade de produção por máquina).
  6. Se a inserção foi confirmada, a aplicação armazena os dados.

No Caso de uso 1, o número dos cenários indicam respectivamente: Usuário 1, no Cenário 1 (1.1) e Usuário 1 no Cenário 2 (1.2).

Caso de uso 2: Excluir peça.

- Cenários: 1.1 / 1.3
- Descrição:
  1. O usuário entra com uma senha.

2. A aplicação valida a senha.
3. Se a senha for válida, o usuário entra na aplicação.
4. O usuário seleciona Excluir peça.
5. O usuário entra com o nome da peça.
6. A aplicação retorna o tipo de liga da peça.
7. Se a exclusão foi confirmada, a aplicação exclui os dados.

Caso de uso 3: Alterar peça.

- Cenários: 1.1 / 1.4
- Descrição:

1. O usuário entra com uma senha.
2. A aplicação valida a senha.
3. Se a senha for válida, o usuário entra na aplicação.
4. O usuário seleciona Alterar peça.
5. O usuário entra com o nome da peça.
6. A aplicação retorna o tipo de liga da peça e o custo e capacidade de produção por máquina e permite alteração.
7. Se a alteração foi confirmada, a aplicação altera os dados.

Caso de uso 4: Consultar peças.

- Cenários: 1.1 / 1.5
- Descrição:

1. O usuário entra com uma senha.
2. A aplicação valida a senha.
3. Se a senha for válida, o usuário entra na aplicação.
4. O usuário seleciona Consultar peças.
5. A aplicação retorna os nomes de todas as peças e a respectiva liga de que a peça é feita.

Caso de uso 5: Inserir parâmetros.

- Cenários: 1.1 / 1.6
- Descrição:
  1. O usuário entra com uma senha.
  2. A aplicação valida a senha.
  3. Se a senha for válida, o usuário entra na aplicação.
  4. O usuário seleciona a opção Inserir parâmetros.
  5. A aplicação requisita a entrada dos dados.
  6. O usuário entra com o mês, ano e com os dados das peças (nome das peças, quantidade em estoque de cada peça, quantidade em atraso de cada peça, demanda de cada peça, estoque máximo de cada peça, quantidade máxima por lote de cada peça, custo de estocagem de cada peça, penalidade por atraso de cada peça, peso de cada árvore e número de cavidades de cada árvore).
  7. Se a inserção foi confirmada, a aplicação insere os dados.

Caso de uso 6: Excluir parâmetros.

- Cenários: 1.1 / 1.7
- Descrição:
  1. O usuário entra com uma senha.
  2. A aplicação valida a senha.
  3. Se a senha for válida, o usuário entra na aplicação.
  4. O usuário seleciona a opção Excluir parâmetros.
  5. A aplicação requisita a entrada de mês, ano e nome da peça.
  6. O usuário entra com o mês, ano e nome da peça.
  7. A aplicação retorna os dados relativos à solicitação (a que tipo de liga pertence a peça, quantidade em estoque, quantidade em atraso, demanda, estoque máximo, quantidade máxima por lote, custo de estocagem, penalidade por atraso, peso de cada árvore e número de cavidades da árvore) e solicita confirmação de exclusão.

8. Se a exclusão foi confirmada, a aplicação exclui os dados.

Caso de uso 7: Dados de peças.

- Cenários: 1.1 / 1.8
- Descrição:

1. O usuário entra com uma senha.
2. A aplicação valida a senha.
3. Se a senha for válida, o usuário entra na aplicação.
4. O usuário seleciona a opção Alterar parâmetros de Dados de peças.
5. A aplicação requisita a entrada de mês, ano e nome da peça.
6. O usuário entra com o mês, ano e nome da peça.
7. A aplicação retorna os dados da peça (quantidade em estoque, quantidade em atraso, demanda, estoque máximo, quantidade máxima por lote, custo de estocagem, penalidade por atraso, peso de cada árvore, número de cavidades da árvore) e permite alteração.
8. Se a alteração foi confirmada, a aplicação altera os dados.

Caso de uso 8: Custo de produção.

- Cenários: 1.1 / 1.9
- Descrição:

1. O usuário entra com uma senha.
2. A aplicação valida a senha.
3. Se a senha for válida, , o usuário entra na aplicação.
4. O usuário seleciona a opção Alterar parâmetros do Custo de produção.
5. A aplicação retorna o custo de produção de todas as peças por máquina e permite alteração.
6. O usuário faz as alterações necessárias.
7. Se a alteração foi confirmada, a aplicação altera os dados.

Caso de uso 9: Capacidade de produção.

- Cenários: 1.1 / 1.10
- Descrição:
  1. O usuário entra com uma senha.
  2. A aplicação valida a senha.
  3. Se a senha for válida, o usuário entra na aplicação.
  4. O usuário seleciona a opção Alterar parâmetros da capacidade de produção.
  5. A aplicação retorna a capacidade de produção de todas as peças por máquina e permite alteração.
  6. O usuário faz as alterações necessárias.
  7. Se a alteração foi confirmada, a aplicação altera os dados.

Caso de uso 10: Alterar parâmetros de Horas de Funcionamento.

- Cenários: 1.1 / 1.11
- Descrição:
  1. O usuário entra com uma senha.
  2. A aplicação valida a senha.
  3. Se a senha for válida, o usuário entra na aplicação.
  4. O usuário seleciona a opção Alterar parâmetros das Horas de Funcionamento.
  5. A aplicação retorna o número de horas de funcionamento de todas as máquinas por turnos e permite alteração.
  6. O usuário faz as alterações necessárias.
  7. Se a alteração foi confirmada, a aplicação altera os dados.

Caso de uso 11: Alterar parâmetros de Horas de Paralisação.

- Cenários: 1.1 / 1.12
- Descrição:
  1. O usuário entra com uma senha.
  2. A aplicação valida a senha.

3. Se a senha for válida, o usuário entra na aplicação.
4. O usuário seleciona a opção Alterar parâmetros das Horas de Paralisação.
5. A aplicação retorna o número de horas de paralisação de todas as máquinas por turnos e dias.
6. O usuário faz as alterações necessárias.
7. Se a alteração foi confirmada, a aplicação altera os dados.

Caso de uso 12: Alterar parâmetro de Horas dos Fornos.

- Cenários: 1.1 / 1.13
- Descrição:
  1. O usuário entra com uma senha.
  2. A aplicação valida a senha.
  3. Se a senha for válida, o usuário entra na aplicação.
  4. O usuário seleciona a opção Alterar parâmetros das Horas dos Fornos.
  5. A aplicação retorna o número de horas dos fornos por turnos e dias, e permite alteração.
  6. O usuário faz as alterações necessárias.
  7. Se a alteração foi confirmada, a aplicação altera os dados.

Caso de uso 13: Alterar parâmetros - Capacidades.

- Cenários: 1.1 / 1.14
- Descrição:
  1. O usuário entra com uma senha.
  2. A aplicação valida a senha.
  3. Se a senha for válida, o usuário entra na aplicação.
  4. O usuário seleciona a opção Alterar parâmetros das Capacidades.
  5. A aplicação retorna a capacidade total de armazenamento, capacidade de armazenamento de cada liga, capacidade total de reposição e capacidade de reposição de cada liga, e permite alteração.

6. O usuário faz as alterações necessárias.
7. Se a alteração foi confirmada, a aplicação altera os dados.

Caso de uso 14: Alterar Mês e/ou Ano.

- Cenários: 1.1 / 1.15
- Descrição:

1. O usuário entra com uma senha.
2. A aplicação valida a senha.
3. Se a senha for válida, o usuário entra na aplicação.
4. O usuário seleciona a opção Alterar parâmetros Mês/Ano.
5. A aplicação requisita a entrada de mês e ano antigos e mês e ano novos.
6. O usuário entra como mês e ano antigos e mês e ano novos.
7. Se a alteração foi confirmada, a aplicação altera os dados.

Caso de uso 15: Consultar parâmetros

- Cenários: 1.1 / 1.16
- Descrição:

1. O usuário entra com uma senha.
2. A aplicação valida a senha.
3. Se a senha for válida, o usuário entra na aplicação.
4. O usuário seleciona a opção Consultar parâmetros.
5. A aplicação requisita a entrada de mês e ano.
6. O usuário entra com o mês e ano.
7. A aplicação retorna os dados que são utilizados como parâmetro: dados das peças (nome das peças, quantidade em estoque de cada peça, quantidade em atraso de cada peça, demanda de cada peça, estoque máximo de cada peça, quantidade máxima por lote de cada peça, custo de estocagem de cada peça, penalidade por atraso de cada peça, peso de cada árvore, número de cavidades de cada árvore), dados do custo de produção (custo de produção de todas as

peças por máquina), dados da capacidade de produção (capacidade de produção de todas as peças por máquina), dados do nro de horas (nro de horas de funcionamento de todas as máquinas por turnos e dias, nro de horas de paralisação de todas as máquinas por turnos e dias e nro de horas dos fornos por turnos e dias) e dados das capacidades (capacidade total de armazenamento, capacidade de armazenamento de cada liga, capacidade total de reposição e capacidade de reposição de cada liga).

Caso de uso 16: Solicitar cálculo da produção de peças.

- Cenários: 1.1 / 1.17
- Descrição:

1. O usuário entra com uma senha.
2. A aplicação valida a senha.
3. Se a senha for válida, o usuário entra na aplicação.
4. O usuário seleciona Calcular produção.
5. A aplicação solicita a entrada do dia.
6. O usuário entra com o dia.
7. A aplicação passa os parâmetros para um outro programa que faz o cálculo e retorna os resultados.
8. O usuário visualiza as peças sugeridas para produção, pode alterar o resultado e pode validar a produção.
9. Se a produção foi validada, a aplicação armazena os dados e retorna à tela principal, se a operação foi cancelada, apenas retorna à tela principal.

Caso de uso 17: Confirmar peças produzidas.

- Cenários: 1.1 / 1.18
- Descrição:

1. O usuário entra com uma senha.
2. A aplicação valida a senha.
3. Se a senha for válida, o usuário entra na aplicação.

4. O usuário seleciona Confirmar produção.
5. A aplicação solicita o dia.
6. O usuário entra com o dia.
7. A aplicação retorna as peças e respectivas quantidades produzidas, por máquina e turno e permite alteração.
8. Se os dados forem confirmados, a aplicação atualiza os dados.

Caso de uso 18: Consultar peças produzidas.

- Cenários: 1.1 / 1.19
- Descrição:
    1. O usuário entra com uma senha.
    2. A aplicação valida a senha.
    3. Se a senha for válida, o usuário entra na aplicação.
    4. O usuário seleciona a opção Consultar produção.
    5. A aplicação requisita a entrada de dia, mês e ano.
    6. O usuário entra com o dia, mês e ano.
    7. A aplicação retorna as seguintes consultas: para cada peça a quantidade que está atrasada, para cada peça a quantidade que foi produzida por turno e por máquina, para cada peça a quantidade que foi produzida por turno, para cada máquina a quantidade de peças que foram produzidas por turno.

4.3.4 Identificação dos UIDs

Para cada caso de uso, é definido um UID [VSdS00b]. UIDs representam graficamente a interação entre o usuário e a aplicação, descritas textualmente nos casos de uso. Diferentes usos dos UIDs podem ser encontrados em [VSdS00a]. Este diagrama apenas descreve a troca de informação entre o usuário e a aplicação, sem considerar especificamente aspectos internos da aplicação. *Unified Modeling Language(UML)* [BRJ99] não apresenta um diagrama parecido. Diagramas UML representando interações, como os Diagramas de Seqüência, Diagramas de Colaboração, Diagramas de Estado, e Diagramas de Atividades, consideram a troca de mensagens entre os objetos do sistema. Então, são mais apropriadamente utilizados na fase de projeto, desde que na fase de análise de requisitos ainda não há um conhecimento dos objetos.

Em [VSdS00a], é apresentado um método para mapear um caso de uso para um UID. O método apresenta os seguintes passos:

1. Identificar as informações (itens de dados) trocadas entre o usuário e a aplicação. É também importante identificar que itens de dados são entrados pelo usuário e quais são retornados pelo sistema.

2. Separar os itens de dados dentro de estados de interação. Itens de dados são incluídos no mesmo estado de interação, a menos que eles dependam de itens de dados anteriores. Neste caso, os itens de dados dependentes são incluídos em outro estado de interação.

3. Distinguir os itens de dados fornecidos pelo usuário e os itens de dados retornados pela aplicação. Itens de dados fornecidos pelo usuário são colocados em retângulos com uma borda contínua (no caso de serem obrigatórios) ou com uma borda tracejada (se forem opcionais). Os itens retornados pela aplicação são colocados diretamente na interação, sem retângulos em volta. Os itens de dados que são associados com um elemento são colocados dentro de parênteses e um conjunto de itens de dados é representado utilizando três pontos antes do nome do conjunto.

4. Conectar os estados de interação com transições (representadas por setas). Quando interações são conectadas, a interação destino deve ter um número seqüencial maior que a interação de origem. É possível conectar uma interação com duas ou mais interações, representando assim várias alternativas de interações sucessoras. Neste caso, a informação fornecida pelo usuário determina qual interação será a próxima. Se a mudança de interação é o resultado de uma seleção de elementos, o número de elementos selecionados é colocado junto com a seta, e a origem desta seta é um conjunto a partir do qual os elementos selecionados fazem parte. A interação inicial sempre tem uma seta sem origem.

5. As operações executadas sobre os itens de dados precisam ser identificadas. Uma operação é representada em um diagrama por uma linha com um “bullet” Se depois

da execução da operação ocorre outra interação, então a operação é posta juntamente com a seta, ao invés da linha.

6. Requisitos não−funcionais não são especificados nos UIDs. Embora casos de uso não especifiquem estes requisitos, às vezes um requisito não−funcional pode aparecer. Neste caso, ele deve aparecer junto ao UID como texto.

É apresentado a seguir o UID referente somente ao Caso de uso 1. Os UIDs restantes estão em A.

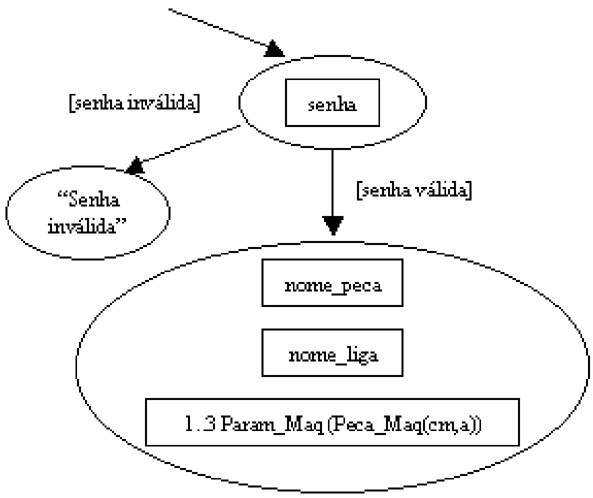

Figura 4.3: UID referente ao Caso de uso 1

Na primeira interação da figura 4.3, o usuário precisa entrar com a senha. Se a senha é inválida, a aplicação retorna uma mensagem de erro, se for válida, o usuário entra com o nome da peça, o nome da liga de que a peça é feita, e custo e capacidade de produção da peça para cada máquina.

4.3.5 Validação dos Casos de Uso e dos UIDs

Nesta fase o projetista interage com cada usuário para validar os casos de uso e os UIDs, exibindo-os aos usuários para verificar se eles concordam. O usuário valida somente os casos de uso e diagramas relacionados com o papel que ele executa. Antes de começar a validação com qualquer usuário, os casos de usos devem ter sido modificados de acordo com o resultado da validação anterior.

4.3.6 Especificação do Esquema Conceitual

O Projeto Conceitual é composto de uma única fase, a especificação do Esquema Conceitual.

O Esquema Conceitual é obtido a partir dos UIDs de acordo com algumas regras. Algumas destas regras são adaptações de técnicas de esquemas de normalização padrão [EN94], então podem ser aplicadas aos UIDs. As regras mais importantes são apresentadas abaixo (mais detalhes podem ser encontrados em [VSdS00b]).

1. Para cada UID, definir uma classe para cada elemento, conjunto de elementos e elementos específicos. Para cada classe definida, assumir a existência de um identificador de atributo OID.

2. Para cada item de dado do conjunto de elementos que aparece em cada UID, ou item de dado entrados pelo usuário, definir um atributo de acordo com o seguinte:

    - Verificar se o item de dado é funcionalmente dependente (ver em [EN94]) do atributo OID em cada classe, i.e., que OID−>item de dado. Verificar que o item de dado não é transitivamente dependente (ver em [EN94]) do OID. Se estas condições são satisfeitas, então o item de dado deve se tornar um atributo da classe.

    - Verificar se o item de dado é funcionalmente dependente do atributo OID de duas ou mais diferentes classes e se o item de dado não é transitivamente dependente dos OIDs. Se essas condições forem satisfeitas, então o item de dado deve se tornar um atributo da associação entre estas classes.

    - Se o item de dado não é funcionalmente dependente do OID de nenhuma classe existente, ou se somente é transitivamente dependente de existentes OIDs, uma nova classe precisa ser criada para ser definida com seu próprio OID. O item de dado precisa ser adicionado como um atributo desta nova classe criada.

3. Para cada item de dado entrado pelo usuário, representado por um retângulo simples, definir um atributo de acordo com o seguinte:

    - Verificar se o item de dado é funcionalmente dependente (ver em [EN94]) do atributo OID em cada classe, i.e., que OID−>item de dado. Verificar que o item de dado não é transitivamente dependente (ver em [EN94]) do OID. Se estas condições são satisfeitas, então o item de dado deve se tornar um atributo da classe.

- Verificar se o item de dado é funcionalmente dependente do atributo OID de duas ou mais diferentes classes e se o item de dado não é transitivamente dependente dos OIDs. Se essas condições forem satisfeitas, então o item de dado deve se tornar um atributo da associação entre estas classes.
- Se o item de dado não é funcionalmente dependente do OID de nenhuma classe existente, ou se somente é transitivamente dependente de existentes OIDs, uma nova classe precisa ser criada para ser definida com seu próprio OID. O item de dado precisa ser adicionado como um atributo desta nova classe criada.

4. Para cada atributo cujo item de dado correspondente aparece em uma estrutura que não corresponde à sua classe, tentar definir uma associação entre as classes e a classe do conjunto de elementos. Se a classe do atributo não está relacionada com a classe que representa o conjunto, então verificar se a classe do atributo está relacionada com a classe de outro atributo presente no conjunto. Verificar se esta associação resultante é semanticamente correta (i.e. se faz sentido no domínio que está sendo modelado).

5. Para cada transição do estado de interação (representada por uma seta), se existirem diferentes classes representando o estado de interação fonte e o estado de interação destino, definir uma associação entre as classes correspondentes. Verificar se a associação resultante é semanticamente correta (i.e. se faz sentido no domínio que está sendo modelado).

6. Para cada opção que aparece anexada a uma transição de estado nos UIDs, verificar se existe uma operação que deve ser criada para alguma das classes correspondentes aos estados de interação.

Ao final do processo é necessário fazer os ajustes necessários no diagrama de classes resultante como: identificar generalizações e agregações, definir cardinalidades ainda não definidas para as associações, eliminar ciclos redundantes de associações, verificar se faltou atributos para uma classe ou ainda se uma classe foi modelada como atributo de outra classe.

Seguindo este método, foi elaborado o Esquema Conceitual apresentado na figura 4.4:

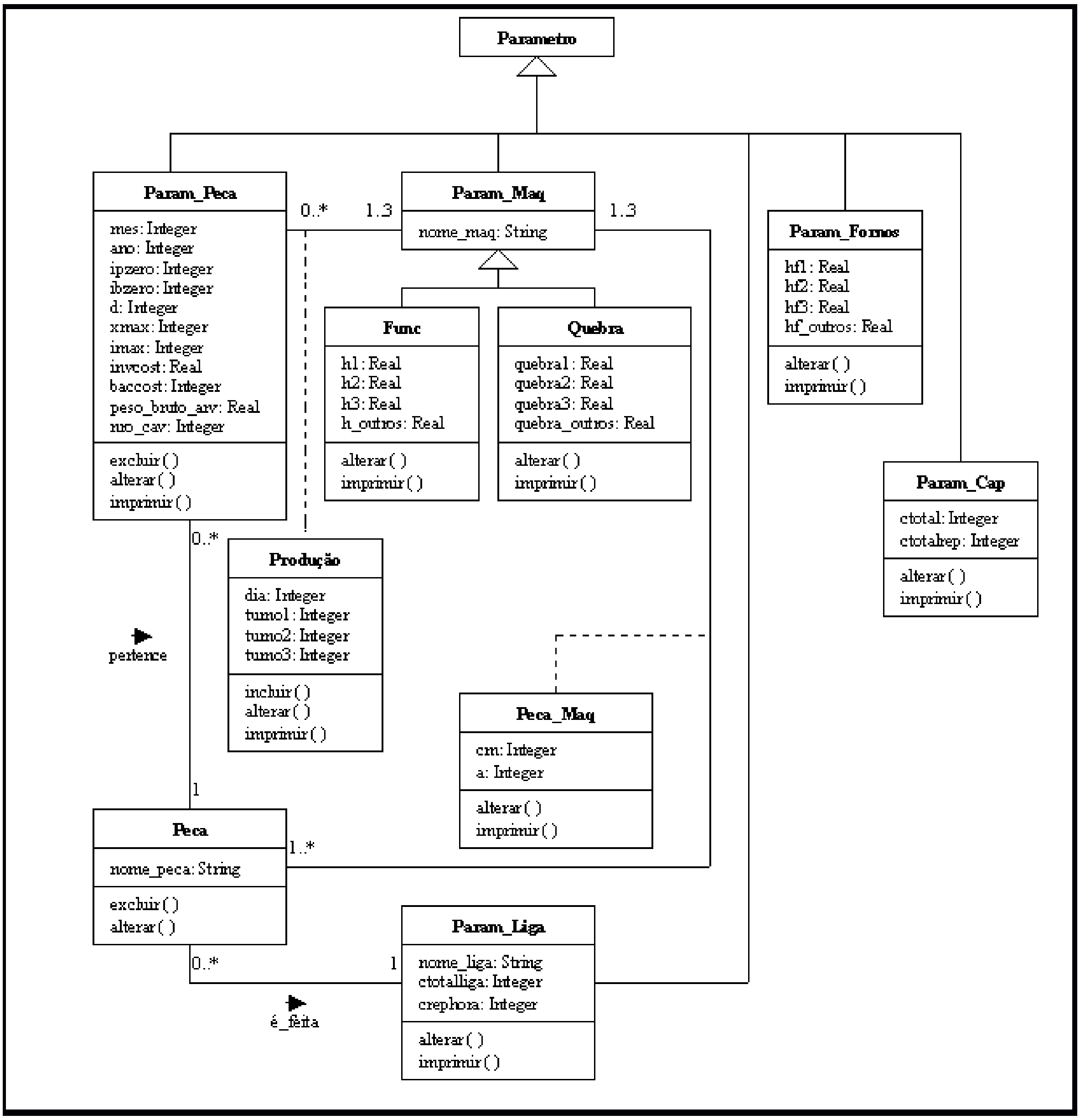

Figura 4.4: Esquema Conceitual da Fundição

O Parâmetro pode ser do tipo Param_Peca, Param_Maq, Param_Liga, Param_Fornos e Param_Cap. Param_Maq pode ser do tipo Func e Quebra. Uma peça pode conter zero ou mais parâmetros de peças e deve pertencer a uma liga. A produção se dá no relacionamento entre Param_Peca e Param_Maq e os parâmetros de capacidade e custo de produção entre Peca e Param_Maq.

No método OOHDM os UIDs são re−examinados depois de concluída esta fase para definir visões navegacionais que constituirão o diagrama de classes navegacionais. Além do mais, as interações são examinadas, e algumas são mapeadas através de passos de navegação que determinarão a topologia navegacional da aplicação Web final.

4.4 Projeto Navegacional

O objetivo desta fase é definir uma visão navegacional sobre um domínio conceitual considerando os perfis dos usuários e as tarefas que devem ser apoiadas, utilizando para tanto o material produzido na fase de levantamento de requisitos descrito anteriormente.

As quatro principais fases do Projeto Navegacional são mostradas na figura 4.5:

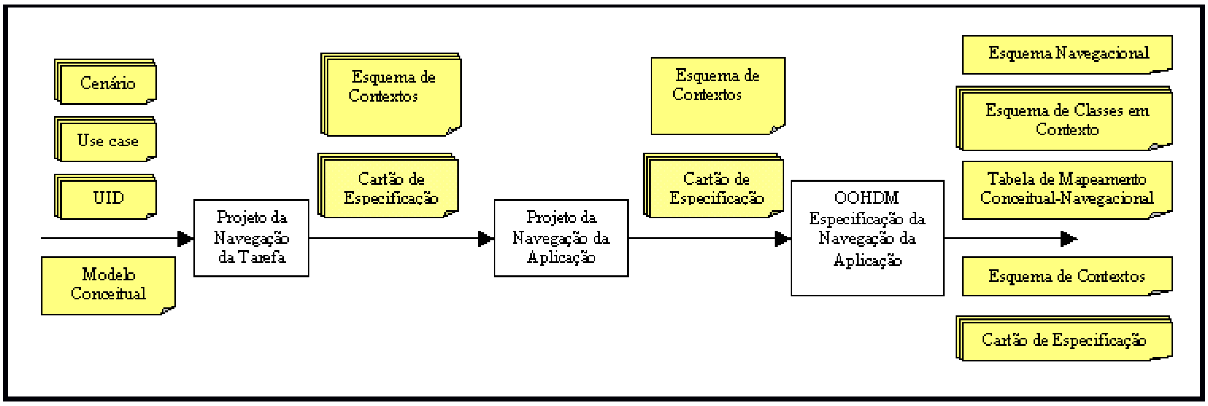

Figura 4.5: Principais fases do Projeto Navegacional [GSV00]

No primeiro estágio, a navegação de cada tarefa (representado por um caso de uso) é projetada, representada em um Esquema de Contexto e correspondentes Cartões de Especificação. A informação navegacional necessária para esta especificação pode ser obtida ou pela análise dos cenários ou por meio do reuso de soluções de projeto conhecidos pelo projetista.

Até este ponto o projetista já possui uma visão navegacional de todas as partes individuais do sistema. O projeto navegacional de toda a aplicação, é obtida por unificação de soluções para cada tarefa. Uma vez que o processo de união foi concluído, a navegação será completamente especificada.

Cada uma destas sub-tarefas são examinadas em mais detalhes nas próximas seções.

4.4.1 Projeto da Navegação da Tarefa

Para cada tarefa, a seqüência de navegação mais apropriada que apóia o usuário é determinada. O projetista pode se basear no UID, desde que tenha sido validado com os usuários e represente assim uma forma aceitável de realização da tarefa. Apesar disso, o projetista não deve se basear somente no UID, também pode consultar descrições de cenários ou reutilizar soluções baseadas em sua própria experiência ou aplicando padrões de projetos [RSL99].

Há tarefas em que o usuário precisa trabalhar com um conjunto de objetos, que poderão ser explorados de diferentes formas, de acordo com o objetivo do usuário. O OOHDM tem um projeto para cada conjunto, chamado Contexto Navegacional. A estrutura navegacional de uma aplicação é caracterizada como um grupo de contextos em que os objetos serão acessados.

Os principais passos do projeto de navegação da tarefa estão representados na figura 4.6:

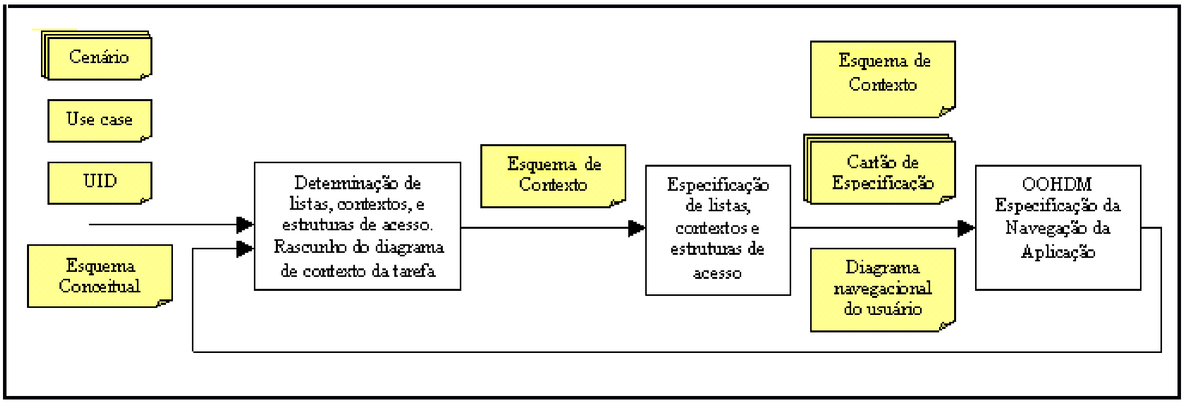

Figura 4.6: Passos para projetar a navegação de uma tarefa [GSV00]

Determinação de listas, contextos e estruturas de acesso

Cada conjunto representado em um UID é analisado para determinar que tipo de primitiva se tornará: uma estrutura de acesso, um contexto ou uma lista de atributo (ou um conjunto ordenado de elementos). A solução será representada em um Esquema de Contexto de Tarefa.

Um contexto é um conjunto de objetos que estão relacionados de acordo com algum aspecto. Um contexto pode conter objetos de uma única classe ou pode conter objetos de várias classes. Os objetos de um contexto podem fazer parte de vários contextos. Independente do contexto no qual o objeto está sendo acessado, ele apresenta todos os atributos definidos no Esquema de Navegação, entretanto, ele pode apresentar aparência e atributos diferentes de acordo com o contexto [SV]. Um contexto é representado por um retângulo contendo um identificador. Os contextos em geral são colocados dentro de um outro retângulo sombreado, que representa a classe navegacional dos seus elementos, que apresenta um identificador em itálico.

As estruturas de acesso são índices que permitem o acesso aos contextos, são representadas por retângulos com linhas densas tracejadas [SV].

As seguintes regras, apresentadas em [GSV00], podem ser aplicadas durante a análise de cada UID:

Regra 1: Mapeamento da interação que apresenta um objeto em um contexto da classe objeto.

Todo objeto é navegável dentro de um contexto. Portanto, qualquer interação que apresenta um objeto, deve ser mapeado em um contexto onde o objeto será navegado.

O próximo passo é determinar os elementos de cada contexto. Os elementos de um contexto geralmente têm algumas características em comum: eles pertencem à mesma classe, têm atributos com o mesmo valor ou são ligados com um dado objeto por meio de um mesmo relacionamento. Por outro lado, um conjunto é arbitrário, então cada elemento precisa ser enumerado.

Para determinar os elementos de um contexto é necessário analisar a fonte da interação no UID. Os seguintes *guidelines* podem ser utilizados:

- Se a fonte da interação está contida em um conjunto de objetos que tem uma característica comum, os elementos do contexto destino são estes mesmos objetos.
- Se a fonte da interação é outro objeto, os elementos do contexto são aqueles que são ligados ao objeto fonte por meio de um certo relacionamento.
- Se for uma interação inicial, ainda não é possível especificar os objetos do contexto.

Regra 2: Mapeamento de um conjunto que é parte de uma informação de um objeto, em uma estrutura de acesso, um contexto ou uma lista de atributos.

Para determinar o mapeamento de cada conjunto é necessário analisar a importância desta informação com respeito à execução das tarefas que utilizarão o objeto. Para determinar este grau de importância, é necessário analisar a descrição de cada caso de uso e os respectivos cenários, para melhor entender a tarefa e os objetivos do usuário.

- Informação muito importante deve aparecer integralmente como listas que são atributos do objeto.
- Informação importante pode aparecer como estruturas de acesso junto ao objeto. Neste caso, o projetista deve decidir que atributos serão apresentados na estrutura.
- Informação complementar pode aparecer como uma âncora que aponta para uma estrutura de acesso ou para um elemento no contexto.

Regra 3: Mapeamento de uma interação de entrada de dados seguido por uma interação que apresenta um conjunto de objetos, em uma estrutura de acesso de conjuntos de objetos.

A escolha da estrutura de acesso, depende da possibilidade de os dados fornecidos pelo usuário terem sido gerados pela aplicação, e se estes dados pertencem aos objetos do conjunto destino ou de outros objetos ligados à eles.

Quando a aplicação não é capaz de gerar os resultados da entrada de dados do usuário de antemão, a interação de entrada de dados e o conjunto resultante são mapeados em uma estrutura de acesso em que os elementos serão computados a partir da entrada do usuário (estrutura dinâmica).

Quando a aplicação sabe de antemão todas as possíveis entradas do usuário válidas, as interações são mapeadas em uma estrutura de acesso.

Uma estrutura de acesso pode ser de dois tipos: simples ou hierárquica. Isso irá depender do dado de entrada e dos objetos do conjunto destino. Uma estrutura de acesso hierárquica é uma seqüência de estruturas simples onde a seleção de cada estrutura é o parâmetro de entrada para a seguinte [RSL99].

Especificação de listas, contextos e estruturas de acesso

A construção do Esquema de Contexto das Tarefas, é completada com a especificação das listas, contextos e estruturas de acesso em seus respectivos cartões. Nestes cartões são documentados dados que não estão representados no diagrama de contexto: as características dos elementos do conjunto, a ordenação, o tipo de navegação interna de contexto, etc.

É definido um cartão de especificação para cada lista (figura 4.7) que é um atributo objeto, para cada contexto (figura 4.8) representado no Esquema de Contexto e para cada tipo de estrutura de acesso (figura 4.9) utilizado no Esquema de Contexto, independentemente da estrutura estar sendo representada graficamente ou sendo descrita como um atributo da classe em alguma visão de contexto.

Lista:
Atributos e Operações:
Regra conceitual:

Figura 4.7: Cartão de especificação para listas [GSV00]

No cartão de especificação para listas apresentado na figura 4.7, é preciso ter informação a respeito dos atributos, operações e regra conceitual para cada lista.

Contexto:	
Elementos:	
Parâmetros:	
Classes em Contexto:	
Navegação Interna:	
Restrições de Uso Usuário:	 Permissão:
Operações:	
Comentários:	

Figura 4.8: Cartão de contexto ou grupo de contexto [SV]

Os nomes dos campos do cartão de contexto (figura 4.8) são bastante significativos:

- **Elementos:** explicita, de maneira mais formal, todos os elementos que farão parte do contexto. Quando o contexto é enumerado, os elementos devem ser especificados explicitamente.

- **Parâmetros:** apresenta os parâmetros que são passados durante a definição do contexto. O parâmetro pode ser uma instância de classe ou o valor de um atributo. Esse campo é utilizado somente pelos grupos de contexto. Na especificação de um contexto simple, esse campo permanece vazio.

- **Classes em contexto:** apresenta a lista das classes em contexto definidas para o contexto em questão. Esta lista inclui uma classe em contexto para cada classe participante do contexto, se necessário.

- **Navegação interna:** define qual o tipo de navegação permitida entre os elementos do contexto (sequencial, circular, por índice ou uma combinação destas). No caso de um contexto que apresenta a navegação sequencial ou circular deve ser especificado o critério de ordenação dos elementos. Um mesmo contexto pode ter diversos critérios de ordenação, sendo que todos os critérios devem ser especificados e o critério *default* deve ser indicado pelo caracter "+".

- **Restrições de uso:** permite indicar as permissões de acesso para cada classe de usuários do contexto. Todos os usuários de um contexto, de uma mesma classe, possuem as mesmas permissões de acesso.

- **Operações:** apresenta as operações do contexto, ou seja, as operações que manipulam os elementos do contexto.

Estrutura de acesso:	
Tipo:	
Parâmetros:	
Elementos:	
Atributos:	Destino:
Ordenação:	
Restrições de Uso Usuário:	 Permissão:
Depende de:	
Influencia:	

Figura 4.9: Cartão de estrutura de acesso [SV]

Os nomes dos campos do cartão de estrutura de acesso (figura 4.9) significam:

- **Tipo:** tipo da estrutura de acesso, pode ser simples, hierárquico ou dinâmico.

- **Parâmetros:** especifica os parâmetros necessários para determinar os elementos do índice.

- **Elementos:** especifica os elementos que serão mostrados no índice.

- **Atributos:** especifica os atributos de cada objeto que será mostrado no índice. Os atributos que são usados para acessar o contexto devem apresentar um destino, ou seja, os elementos que serão acessados a partir do índice. Quando vários seletores apresentam um mesmo destino, eles são especificados em uma lista onde os seletores são separados por vírgula.

- **Ordenação:** especifica o critério de ordenação dos elementos no índice. Quando um contexto é hierárquico, devem ser especificados os critérios de ordenação para os elementos de cada nível de hierarquia.
- **Restrições de Uso:** especifica as diferentes permissões de acesso ao índice.
- **Depende de** e **Influencia** apresentam os elementos criados durante a modelagem que ao serem modificados influenciam neste índice ou que são influenciados por uma modificação deste índice.

Todos os Cartões de Especificação para o exemplo da fundição são apresentados em B.

Validação da Navegação de Cada Tarefa

Uma vez que as navegações das tarefas tenham sido projetadas, o projetista pode validá−las com os usuários, utilizando para tal, o Esquema de Contextos como ferramenta de comunicação.

Durante a validação, o projetista deve registrar cada entrevista separadamente, assim, mais tarde, se baseando nesses registros, e em sua própria experiência, o projetista poderá optar por uma solução, tentando reconciliar as diversas propostas dos usuários entrevistados. Se a validação resultar em grandes mudanças, é necessário construir um novo esquema de contextos e repete−se a validação; caso contrário, o projetista pode passar para o próximo passo do processo.

4.4.2 Projeto da Navegação da Aplicação

O Esquema de Contexto da aplicação é sintetizado por meio de sucessivas uniões de Esquemas de Contextos das Tarefas. Para cada passo da união, um novo Esquema de Contexto é construído, sendo definidos: seus contextos, suas estruturas de acesso e as visões do objeto para cada contexto.

O primeiro passo é unificar as soluções de cada grupo de usuário, começando com o grupo que executa as tarefas principais. As tarefas nas quais um mesmo objeto navegacional é acessado, são unificados em seqüência. Ao invés de manter a solução para cada grupo de usuário separadamente, até o final, é preferível unificá-los tão logo cada solução tenha sido definida.

A mesma classe navegacional pode aparecer em contextos originando diferentes diagramas parciais, então, é feita a análise se alguns destes contextos podem ser substituídos por um único contexto. Para tal análise, são analisadas as tarefas que cada contexto suporta, assim como as visões do objeto e permissões de usuários nestes contextos. Eles serão substituídos por um único contexto somente se o contexto resultante suporta todas as tarefas, e se as visões do objeto e a permissão do usuário não conflitam.

Também é feita a análise da possibilidade da unificação de caminhos de acesso conduzindo aos contextos originais e se é possível generalizar as navegações originais destes contextos dentro do diagrama resultante. Desde que eles são agora parte do mesmo diagrama, é necessário explorar as possibilidades de navegação entre eles.

O Esquema de Contexto da fundição na figura 4.10 suporta todas as tarefas executadas pelo usuário da aplicação. Este esquema apresenta uma generalização de contextos, onde as subclasses são incluídas dentro do retângulo representante da superclasse, que é representada por uma tonalidade de cinza diferente. A linha tracejada entre os contextos representa que não é possível navegar a partir de um elemento de um contexto para o outro contexto. Os contextos de criação, exclusão e modificação de objetos são representados por um retângulo com bordas arredondadas e uma barra vertical preenchida no lado direito. A barra vertical preenchida indica que o contexto é dinâmico, ou seja, os elementos são definidos ou alterados em tempo de execução. As setas com um círculo na extremidade (implementando o "design pattern" *Landmark*) indicam que as estruturas de acesso ou contexto podem ser acessados a partir de qualquer lugar da aplicação.

4.4.3 Especificação Navegacional da Aplicação

A especificação do projeto navegacional OOHDM, é um conjunto de modelos: uma Tabela de Mapeamento Conceitual−Navegacional, um Esquema Navegacional, vários Esquemas de Classes em Contexto, um Esquema de Contexto, e vários Cartões de Especificação, que durante a apresentação da documentação de um projeto, devem ser apresentados nesta ordem. O Esquema de Contexto e os cartões são obtidos no processo de união.

A seguir são descritos os modelos do Esquema Navegacional, dos Esquemas de Classes em Contexto e a Tabela de Mapeamento Conceitual−Navegacional.

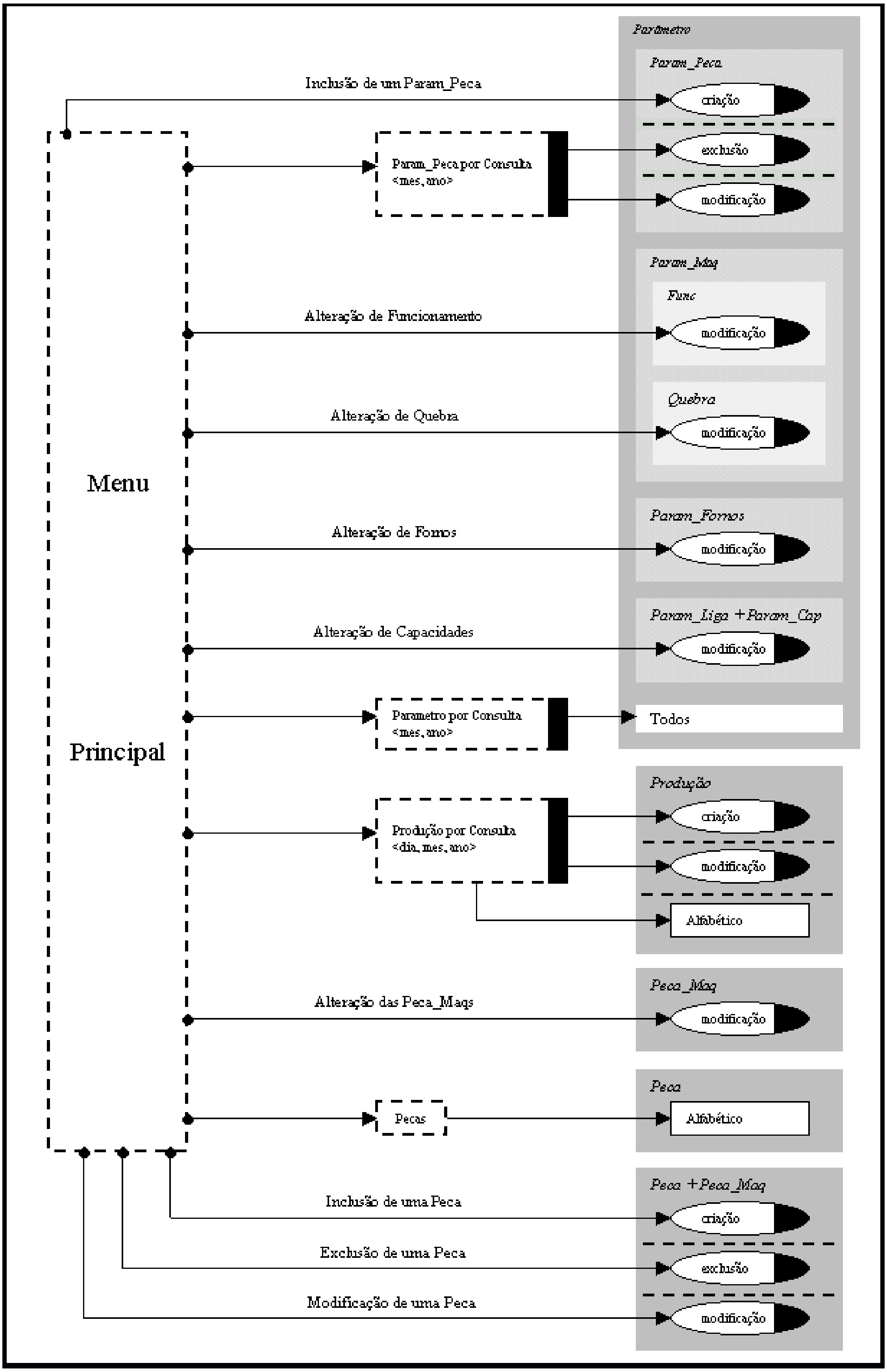

Figura 4.10: Esquema de Contexto da fundição

Esquema Navegacional

Classes navegacionais descrevem a informação contida por meio da qual os usuários irão navegar. Estas classes constituem visões das classes conceituais: uma classe navegacional pode apresentar alguma informação de uma classe conceitual ou uma união de conteúdos de várias classes conceituais. Classes navegacionais são chamadas nós; relacionamentos navegacionais são chamados elos; e atributos que ativam navegações, são chamados de âncoras.

Cada classe de um Esquema de Contexto é um nó. As navegações entre as classes dos Esquemas de Contexto podem ser elos. É necessário analisar a regra de seleção do contexto destino, especialmente quando isso envolve navegação entre contextos da mesma classe.

A figura 4.11 apresenta o Esquema Navegacional obtido para a fundição.

Esquema de Classes em Contexto

Os esquemas de classes de contexto são derivados do Esquema de Contextos e dos Cartões de Especificação. É definida uma Classe em Contexto para a classe navegacional de um objeto, quando um objeto pode ser acessado em contextos diferentes, e é necessário, ou conveniente, apresentar seus dados e suas navegações de um modo diferente em cada um desses contextos.

Na aplicação para a fundição como os dados não precisam ser mostrados de modo diferente nos contextos em que aparecem, não foi necessário a criação deste esquema.

Tabela do Mapeamento Conceitual–Navegacional

O próximo passo é especificar o mapeamento conceitual para cada nó e elo.

A classe conceitual base para cada nó é determinada. Depois, por comparação, os atributos e operações do nó que não aparecem na classe base são identificados e uma regra de mapeamento para cada um é expresso. A classe conceitual base de uma classe navegacional é a derivada da mesma interação no UID durante a síntese do Esquema de Conceitual.

Os atributos e operações de um nó, podem não aparecer em sua classe conceitual base por duas razões: foram definidas durante o Projeto de Navegação, como as âncoras, listas e estruturas de acesso, ou pertencem à outra classe conceitual relacionada à classe base.

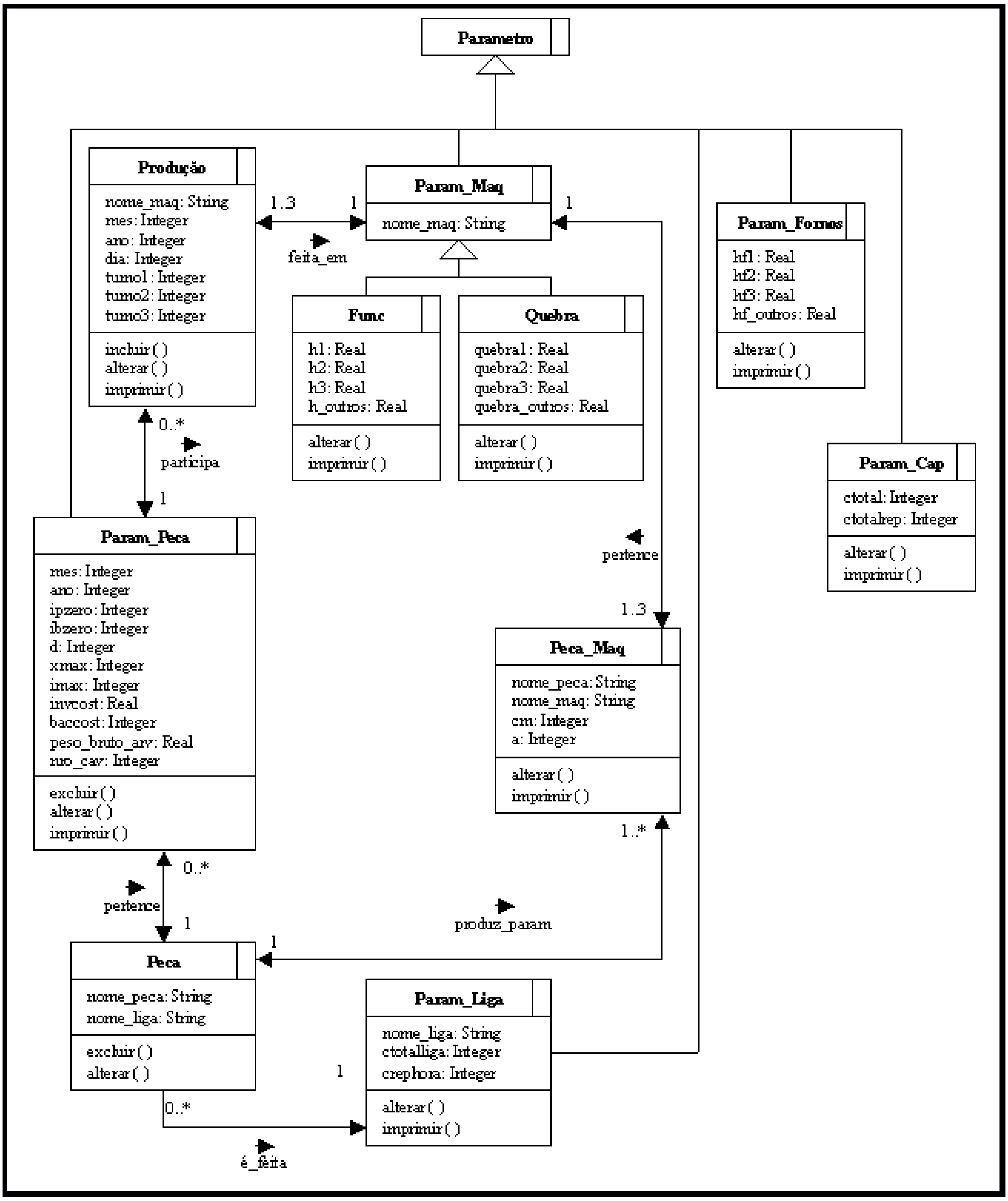

Figura 4.11: Esquema Navegacional da fundição

A maioria dos elos são derivadas diretamente das relações conceituais. Em alguns casos, contudo, certos elos correspondem à composição das relações conceituais, e suas regras de derivação devem também ser especificadas na Tabela de Mapeamento Conceitual−Navegacional.

A tabela 4.1, apresenta o mapeamento do Esquema Conceitual para o Esquema Navegacional. Somente as classes e atributos que não são mapeados diretamente para o esquema navegacional são mostrados na tabela.

4.4.4 Especificação Final para a Navegação da Aplicação

Os elementos, parâmetros e regras de seleção dos cartões de especificação de estruturas de acesso e contextos, devem ser baseados nas classes navegacionais (nós e elos). As regras conceituais que não aparecem mais nestes cartões, desde que foram criados somente para ajudar no mapeamento conceitual.

Nesta etapa, os cartões de especificação são revisados com o objetivo de eliminar qualquer diferença com o Modelo Navegacional.

4.5 Outros Métodos de Levantamento de Requisitos

Várias abordagens utilizam o uso de cenários e casos de uso para o levantamento de requisitos, mas de formas diferentes. Em [SMMM], cenários são utilizados para validar requisitos e são automaticamente gerados a partir dos casos de uso explicitados pelos usuários. Os cenários utilizados em [ECTB], diferentemente dos cenários descritos neste capítulo, descrevem os aspectos de navegação e de interface, já que são usados na reutilização de um Web site existente.

Em [IB99], o levantamento de requisitos é baseado nos casos de uso derivados a partir de histórias dos usuários. Uma história do usuário pode ser visto como um cenário mais simples, onde um agente executa uma ação física sobre um objeto. Contudo, em ambos, histórias de usuários e casos de uso, as interações entre os usuários e a aplicação são especificadas somente com uma descrição textual.

No Processo Unificado [JBR99], os requisitos também são definidos por casos de uso, mas cenários não são utilizados para especificá-los. Se necessário, protótipos de interface do usuário podem ser utilizados para entender e especificar os requisitos, mas não há diagramas que focam especificamente na interação entre o usuário e a aplicação.

O método WSDM [Tro00], para desenvolvimento de Web sites somente para leitura,

Objetos Mapeados	**Regra de Mapeamento**
Conceitual: Produção	
Navegacional: Produção	Mapeamento direto
Param_Peca participa Produção	Mapeamento direto
Param_Maq feita_em Produção	Mapeamento direto
Conceitual: Peca_Maq	
Navegacional: Peca_Maq	Mapeamento direto
Peca produz_param Peca_Maq	Mapeamento direto
Peca pertence_a Param_Maq	Mapeamento direto
Conceitual: Param_Peca.mês	
Navegacional: Param_Peca.mês	Mapeamento direto
Produção.mês	Selecionar PP.mês de PP:Param_Peca, PM: Param_Maq ONDE PP produção PM
Conceitual: Param_Peca.ano	
Navegacional: Param_Peca.ano	Mapeamento direto
Produção.ano	Selecionar PP.ano de PP:Param_Peca, PM: Param_Maq ONDE PP produção PM
Conceitual: Param_Maq.nome_maq	
Navegacional: Param_Maq.nome_maq	Mapeamento direto
Produção.nome_maq	Selecionar PM.nome_maq de PM:Param_Maq, PP: Param_Peca ONDE PP produção PM
Conceitual: Peca.nome_peca	
Navegacional: Peca.nome_peca	Mapeamento direto
Peca_Maq.nome_peca	Selecionar PM.nome_peca de PM:Peca_Maq, PL: Param_Liga ONDE PM peca_maq PL
Conceitual: Param_Liga.nome_peca	
Navegacional: Param_Liga.nome_peca	Mapeamento direto
Peca_Maq.nome_liga	Selecionar PL.nome_liga de PM:Peca_Maq, PL: Param_Liga ONDE PM peca_maq PL

Tabela 4.1: Tabela de Mapeamento Conceitual−Navegacional para a fundição

propõem uma abordagem similar a apresentada neste capítulo. No WSDM, a navegação para cada classe é projetada separadamente e o modelo de navegação do Web site é a coleção de rastros de navegação, disjuntas de várias classes, ligadas por um elemento inicial. Desde que diferentes grupos de usuários podem precisar navegar através de um mesmo rastro, a estrutura navegacional da aplicação não deve ser disjunta. O método OOHDM permite especificar aplicações com comportamento e navegação complexos e não somente para leitura.

Capítulo

5

Conclusão

A evolução rápida da Web e o impacto que tem causado nos últimos anos são significativos, porém, a maneira desordenada com que os aplicativos são construídos preocupa.

O processo de construção de aplicações hipermídia não é intrinsecamente diferente daquele utilizado na construção das aplicações convencionais. Existe um crescente consenso acerca do tipo de atividades que devem ser desempenhadas com respeito ao produto de software: modelagem ou análise, projeto, implementação, teste e manutenção. A modelagem conceitual ou de domínio destina-se à compreensão do domínio problema e à construção de modelos adequados deste domínio, enquanto o projeto lida com abstrações no universo do software e tende a maximização da modularidade e do reuso. O modelo do projeto é independente da implementação no sentido em que, embora possa levar em consideração algumas configurações de implementação, não é condicionado por um ambiente de implementação em particular. Finalmente, durante a implementação, abstrações de projeto são convertidas em artefatos concretos de implementação. Teste e manutenção podem ser integradas com sucesso ao processo de desenvolvimento através do reconhecimento de que o ciclo de vida de um produto de software é um contínuo em que o software evolui na medida em que novas exigências aparecem.

As técnicas criadas e já consolidadas na Engenharia de Software podem ser utilizadas para Engenharia de Web, mas é preciso fazer algumas adaptações, respeitando as características dos aplicativos Web. Pressman propôs um *framework* para o processo de Engenharia de Web que inclui Análise de Requisitos, Projeto Arquitetural, Projeto Navegacional, Projeto de Interface, Produção e Testes.

Nestas notas didáticas foi descrito um método de modelagem proposto por Schwabe [SR95] para o processo de Engenharia de Web, assim como o exemplo da modelagem de um aplicativo Web para uma fundição. Um dos motivos da escolha deste método em especial, é sua ampla utilização pela comunidade hipermídia em projetos de aplicações em diversos países.

O método OOHDM *Object Oriented Hypermedia Design Method* possui quatro atividades: a Modelagem Conceitual (classe, atributos e associações), o Projeto Navegacional, o Projeto de Interface Abstrata e a Implementação. Foi verificada uma carência no processo de implementação, principalmente no que diz respeito aos ambientes de desenvolvimento de aplicações hipermídia, como por exemplo, HTML, *ToolBook*. Um ambiente completo de desenvolvimento de aplicações hipermídia orientada a objetos, facilitaria e daria continuidade a uma modelagem totalmente orientada a objetos.

Conforme demonstrado neste estudo, o OOHDM é uma técnica poderosa para modelagem de aplicações, mas poderia ser melhor aproveitada caso fosse possível implementar exatamente conforme modelado. Assim, um próximo estudo será realizado para se relacionar e transformar os artefatos de modelagem para os respectivos componentes do software a ser desenvolvido.

Apêndice

A

UIDs

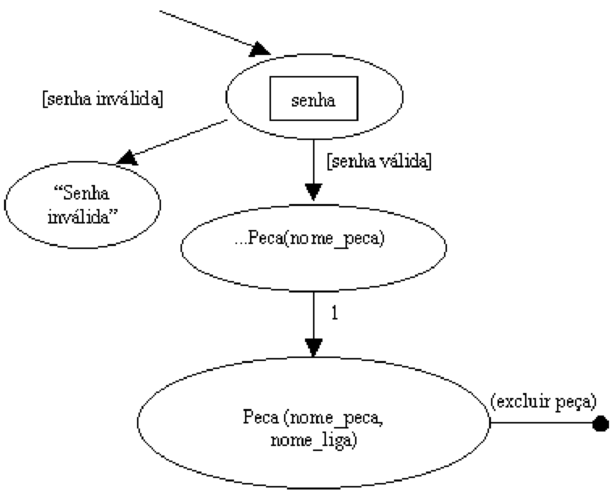

Figura A.1: UID referente ao Caso de uso 2

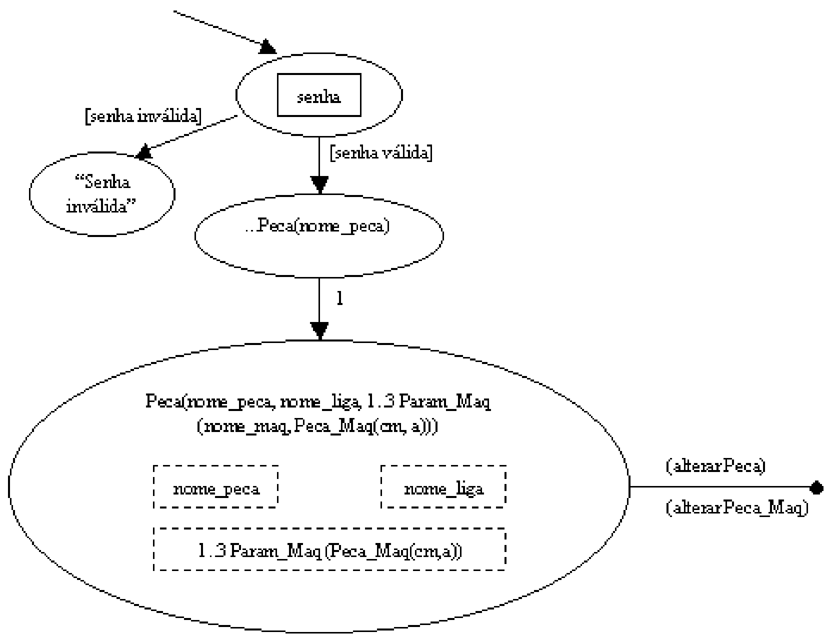

Figura A.2: UID referente ao Caso de uso 3

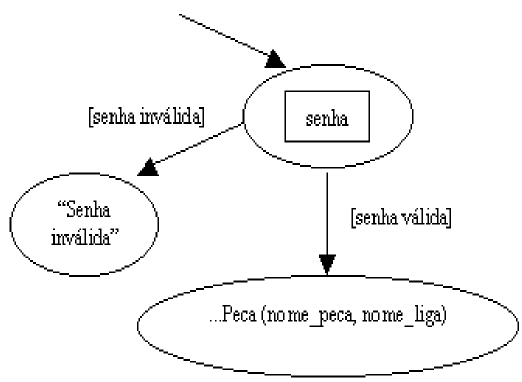

Figura A.3: UID referente ao Caso de uso 4

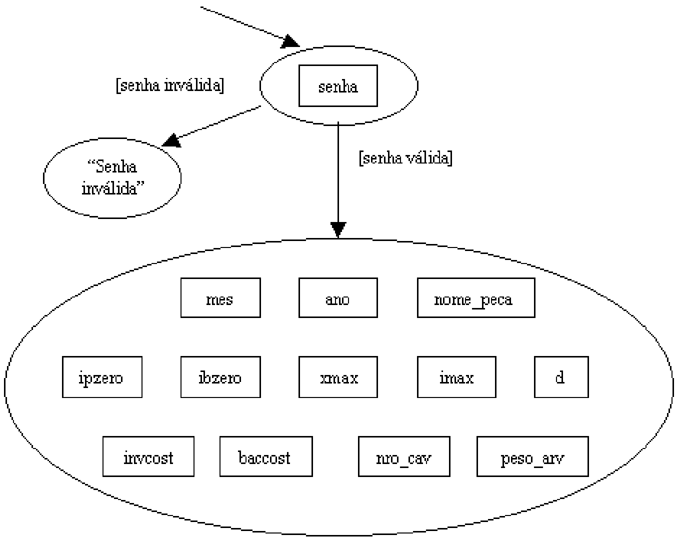

Figura A.4: UID referente ao Caso de uso 5

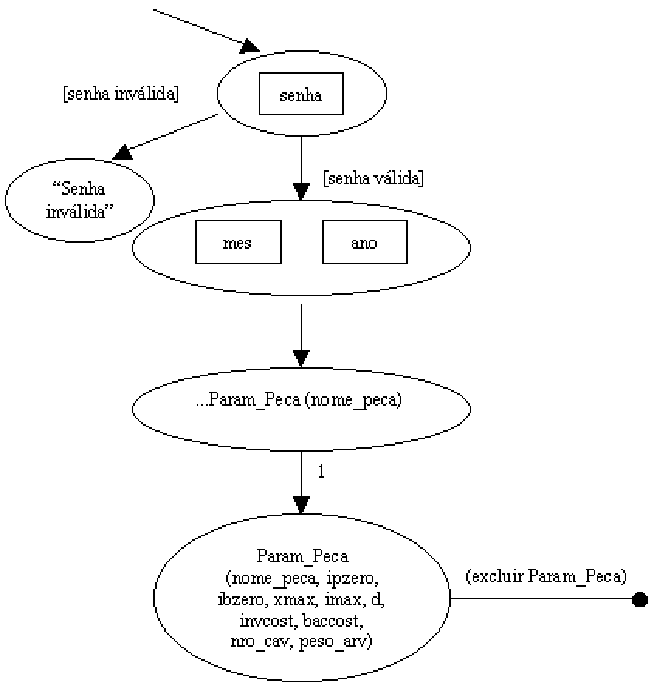

Figura A.5: UID referente ao Caso de uso 6

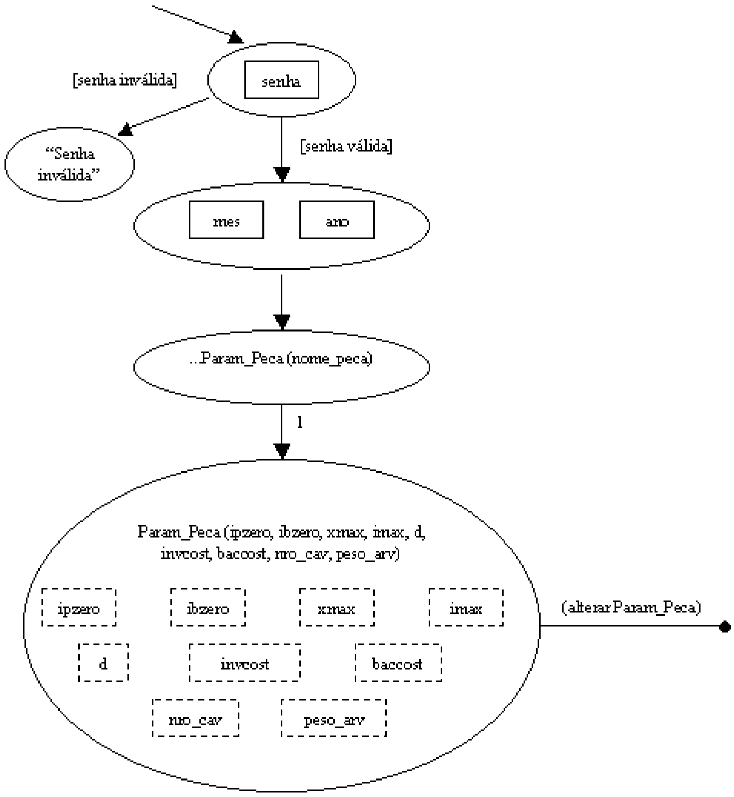

Figura A.6: UID referente ao Caso de uso 7

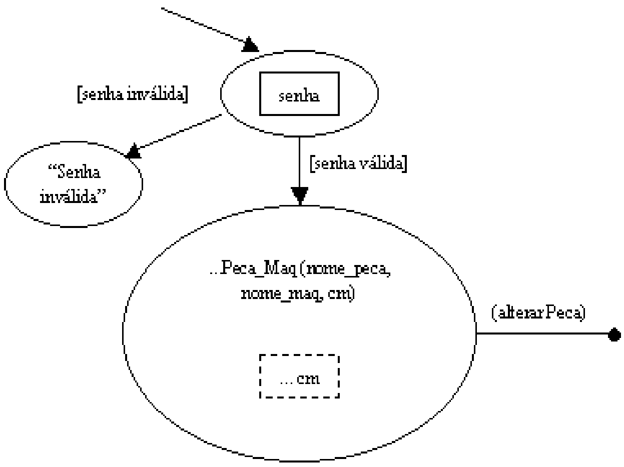

Figura A.7: UID referente ao Caso de uso 8

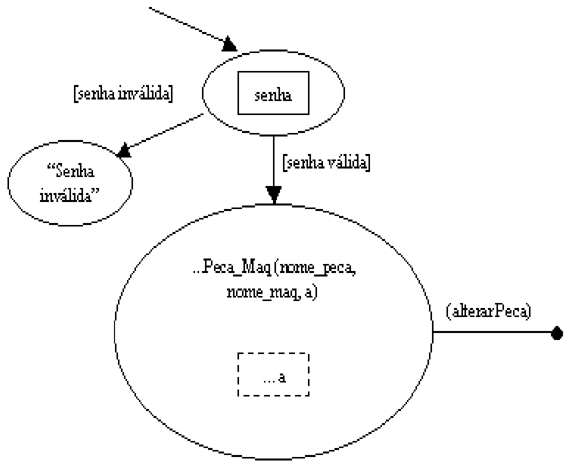

Figura A.8: UID referente ao Caso de uso 9

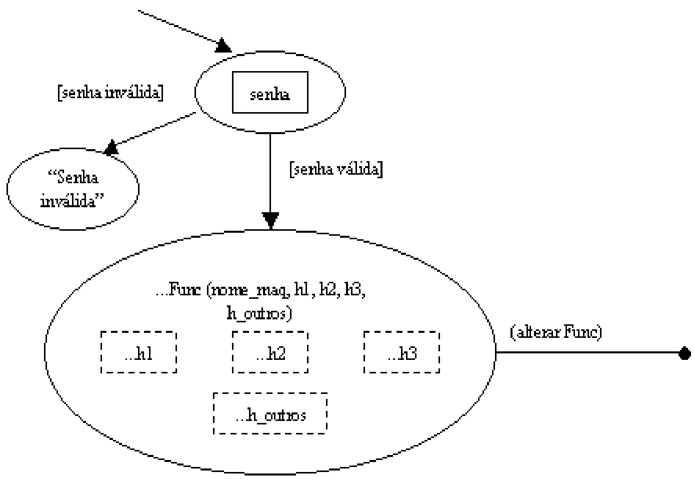

Figura A.9: UID referente ao Caso de uso 10

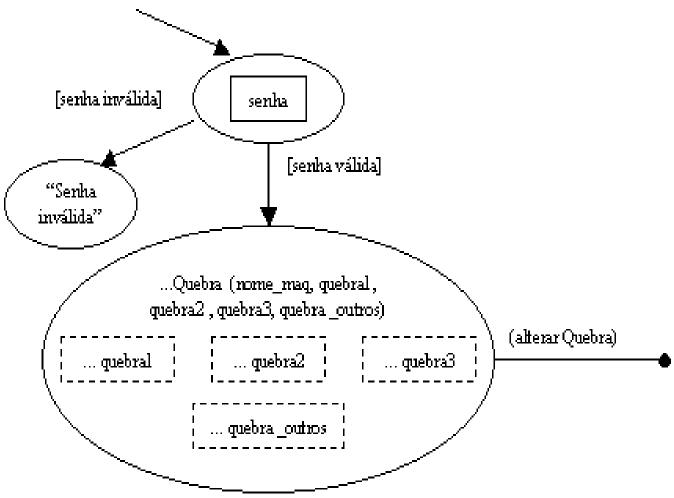

Figura A.10: UID referente ao Caso de uso 11

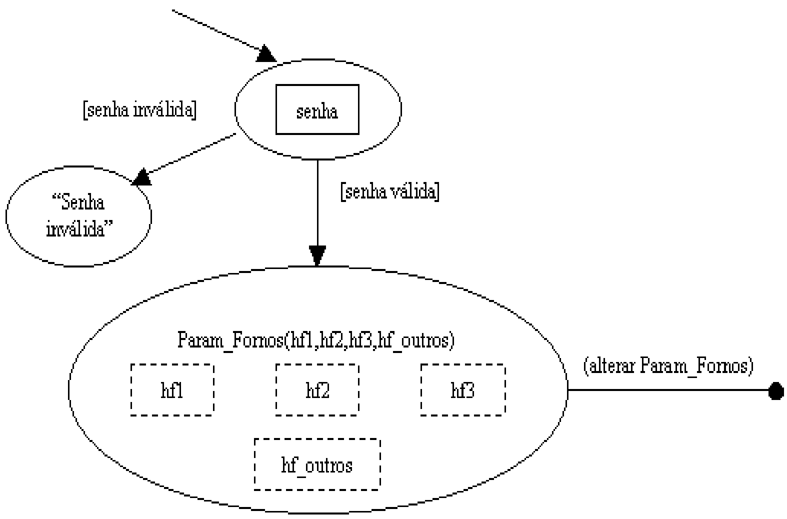

Figura A.11: UID referente ao Caso de uso 12

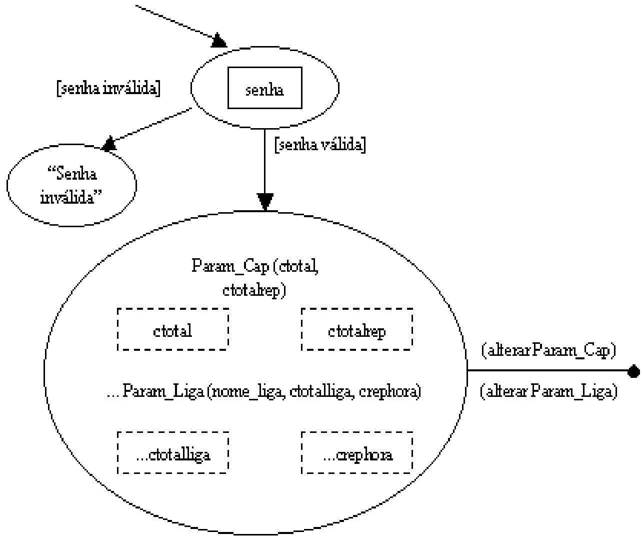

Figura A.12: UID referente ao Caso de uso 13

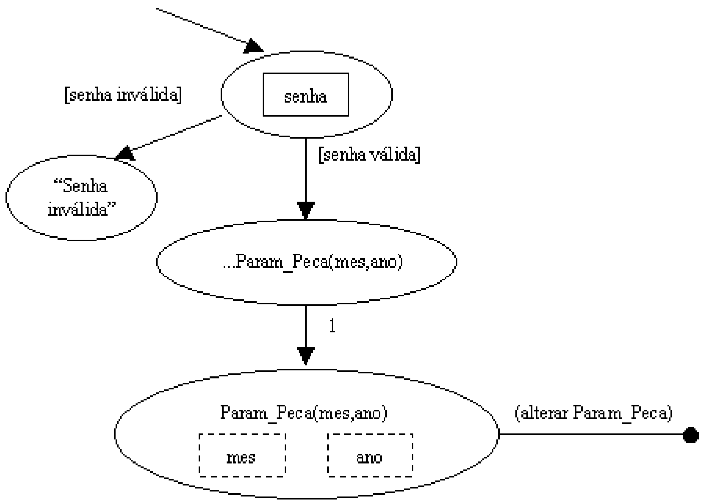

Figura A.13: UID referente ao Caso de uso 14

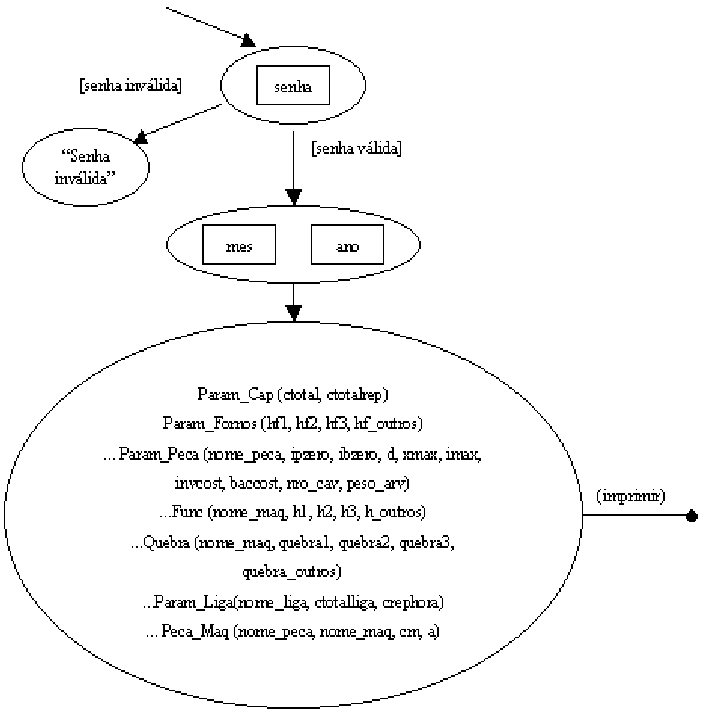

Figura A.14: UID referente ao Caso de uso 15

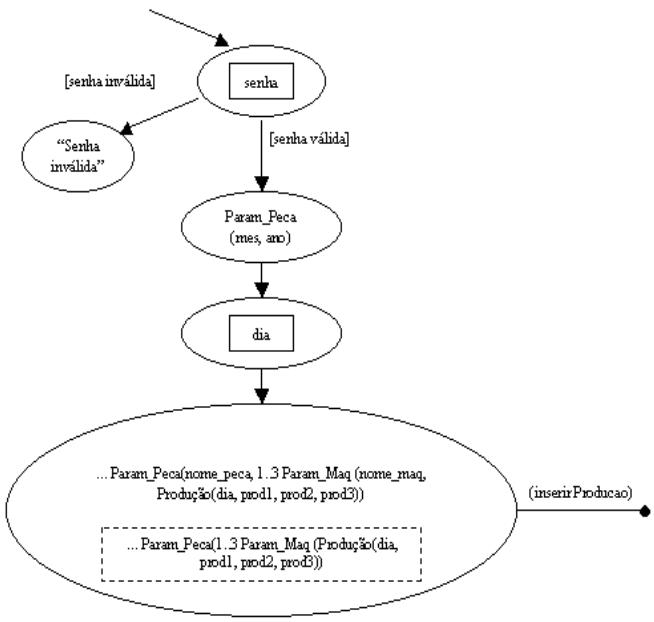

Figura A.15: UID referente ao Caso de uso 16

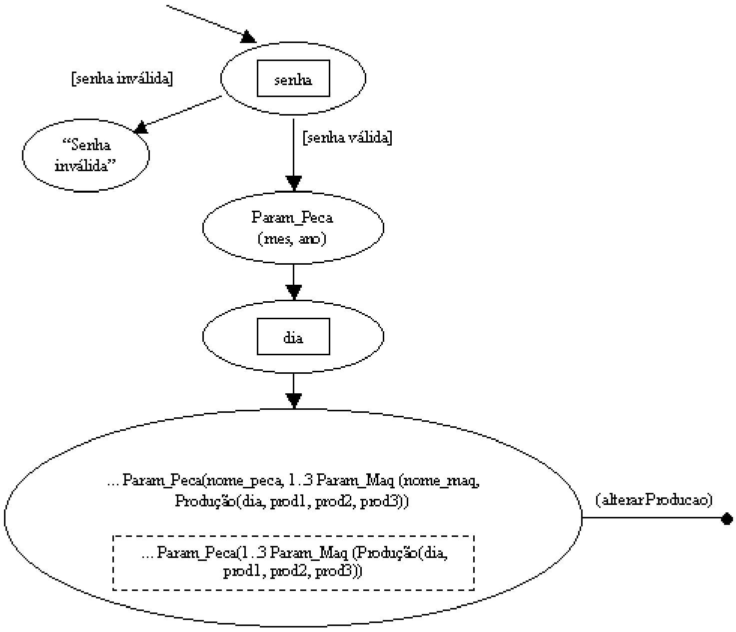

Figura A.16: UID referente ao Caso de uso 17

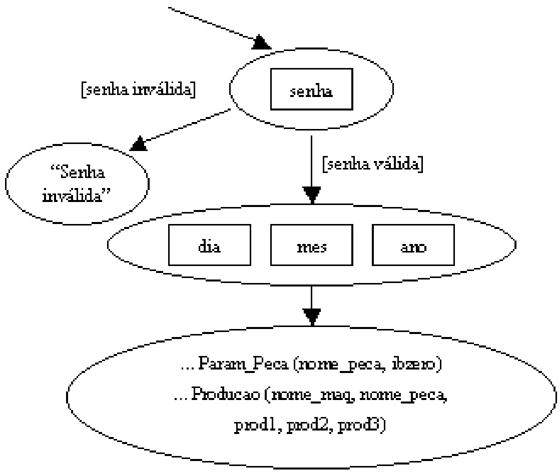

Figura A.17: UID referente ao Caso de uso 18

Apêndice

B

Cartões de Especificação

Cartões de Contexto:

Contexto: criação de um parâmetro de peça	
Elementos: P: Param_Peça	
Parâmetros:	
Classes em Contexto:	
Navegação Interna:	
Restrições de Uso Usuário: todos	Permissão: todas
Operações: incluir um param_peca	
Comentários: O usuário só poderá incluir o param_peca se não existir outro com mesmo nome peca em determinado mês e ano.	

Contexto: exclusão de um parâmetro de peça	
Elementos: P: Param_Peça	
Parâmetros:	
Classes em Contexto:	
Navegação Interna:	
Restrições de Uso Usuário: todos	Permissão: todas
Operações: excluir param_peca	
Comentários: O usuário só poderá excluir o param_peca se este estiver cadastrado.	

Contexto: modificação de um parâmetro de peça	
Elementos: P: Param_Peça	
Parâmetros:	
Classes em Contexto:	
Navegação Interna:	
Restrições de Uso Usuário: todos	Permissão: todas
Operações: alterar param_peca	
Comentários: O usuário só poderá modificar o param_peca se este estiver cadastrado.	

Contexto: modificação de um parâmetro de horas de funcionamento	
Elementos: F: Func	
Parâmetros:	
Classes em Contexto:	
Navegação Interna:	
Restrições de Uso Usuário: todos	Permissão: todas
Operações: alterar func.	
Comentários:	

Contexto: modificação de um parâmetro de horas de quebra	
Elementos: Q: Quebra	
Parâmetros:	
Classes em Contexto:	
Navegação Interna:	
Restrições de Uso Usuário: todos	Permissão: todas
Operações: alterar quebra	
Comentários:	

Contexto: modificação de um parâmetro de fornos	
Elementos: F: Param_fornos	
Parâmetros:	
Classes em Contexto:	
Navegação Interna:	
Restrições de Uso Usuário: todos	Permissão: todas
Operações: alterar param_fornos	
Comentários:	

Contexto: modificação de um parâmetro de capacidades	
Elementos: L: Param_liga, C: Param_cap	
Parâmetros:	
Classes em Contexto:	
Navegação Interna:	
Restrições de Uso Usuário: todos	Permissão: todas
Operações: alterar capacidades	
Comentários:	

Contexto: Consulta de tudo
Elementos: P: Parametro, PP: Param_Peca, PeM: Peca_Maq, F: Func, Q:Quebra, PL:Param_Liga, PF:Param_Fornos, PC:Param_Cap
Parâmetros:
Classes em Contexto:
Navegação Interna: por índice
Restrições de Uso Usuário: todos Permissão: todas
Operações:
Comentários:

Contexto: criação de uma produção
Elementos: P: Produção
Parâmetros:
Classes em Contexto:
Navegação Interna:
Restrições de Uso Usuário: todos Permissão: todas
Operações: incluir uma produção
Comentários: O usuário só poderá incluir o param_peca se não existir outro com mesmo nome_peca em determinado mês e ano.

Contexto: modificação de produção
Elementos: P: Produção
Parâmetros:
Classes em Contexto:
Navegação Interna:
Restrições de Uso Usuário: todos Permissão: todas
Operações: modificar produção
Comentários: O usuário só poderá modificar produção se este estiver cadastrado.

Contexto: consulta da produção
Elementos: P: Produção
Parâmetros:
Classes em Contexto:
Navegação Interna: por índice
Restrições de Uso Usuário: todos Permissão: todas
Operações:
Comentários:

Contexto: modificação de peca_maq
Elementos: P: Peca_Maq
Parâmetros:
Classes em Contexto:
Navegação Interna:
Restrições de Uso Usuário: todos Permissão: todas
Operações: modificar peca_maq
Comentários:

Contexto: peca alfabético
Elementos: P: Peca
Parâmetros:
Classes em Contexto:
Navegação Interna: seqüencial (ordenada por P.nome_peca, Ascendente)
Restrições de Uso Usuário: todos Permissão: todas
Operações:
Comentários:

Contexto: criação de uma peça
Elementos: P: Peca, PM: Peca_Maq
Parâmetros:
Classes em Contexto:
Navegação Interna:
Restrições de Uso Usuário: todos Permissão: todas
Operações: incluir uma peca
Comentários: O usuário só poderá incluir a peca se não existir outra com mesmo nome peca.

Contexto: exclusão de peça
Elementos: P: Peca, PM: Peca_Maq
Parâmetros:
Classes em Contexto:
Navegação Interna:
Restrições de Uso Usuário: todos Permissão: todas
Operações: excluir peca
Comentários: O usuário só poderá excluir peca se este estiver cadastrado.

Contexto: modificação de peça
Elementos: P: Peca, PM: Peca_Maq
Parâmetros:
Classes em Contexto:
Navegação Interna:
Restrições de Uso Usuário: todos Permissão: todas
Operações: modificar peca
Comentários: O usuário só poderá modificar peca se este estiver cadastrado.

Cartões de Estrutura de Acesso:

Estrutura de acesso: Peças	
Tipo: simples	
Parâmetros:	
Elementos: P:Pecas	
Atributos: P.nome_peca Destino: P: Pecas onde P *pertence_a* Peça Alfabético	
Ordenação: ordenado por P.nome_peca	
Restrições de Uso Usuário: todos	 Permissão: todas
Depende de:	
Influencia:	

Estrutura de acesso: Produção por consulta	
Tipo: dinâmico	
Parâmetros:	
Elementos: P: Produção	
Atributos: P.mês, P.ano Destino: P: Produção onde P *pertence_a* Produção Criação, Modificação ou Alfabético.	
Ordenação:	
Restrições de Uso Usuário: todos	 Permissão: todas
Depende de:	
Influencia:	

Estrutura de acesso: Parâmetro por consulta	
Tipo: dinâmico	
Parâmetros:	
Elementos: P: Param_Peça	
Atributos: P.mês, P.ano Destino: P: Param_Peça onde P *pertence_a* Parâmetro Todos.	
Ordenação:	
Restrições de Uso Usuário: todos	 Permissão: todas
Depende de:	
Influencia:	

Estrutura de acesso: Param_peca por consulta
Tipo: dinâmico
Parâmetros:
Elementos: P:Param_peca
Atributos: P.mês, P.ano Destino: P: Param_Peca onde P *pertence_a* Param_Peça Exclusão ou Modificação
Ordenação:
Restrições de Uso Usuário: todos Permissão: todas
Depende de:
Influencia:

Referências Bibliográficas

[Ara03] S. Araujo. *Modelos e Métodos para o Planejamento e Programação da Produção Aplicado no Setor de Fundições.* PhD thesis, ICMC-USP, 2003.

[BRJ99] G. Booch, J. Rumbaugh, and I. Jacobson. *The Unified Modeling Language User Guide.* 1999.

[Car95] J. M. Carroll. *Scenario-Based Design: Envisioning Work and Technology in System Development.* 1995.

[ECTB] L. E. Erskine, D. R. N. Carter-Tod, and J. K. Burton. Dialogical techniques for the design of web sites. *Int. Journal Human-Computer Studies.*

[EN94] R. Elmasri and S. B. Navathe. *Navathe S.B.: Fundamentals of Database Systems.* Benjamin/Cummings, second edition, 1994.

[FGK] R. Fourer, D. M. Gay, and B. W. Kernighan. Ampl - a modeling language for mathematical programming.

[GPS93] F. Garzotto, P. Paolini, and D. Schwabe. Hdm: a model-based approach to hypertext application design. *ACM Transactions on Information Systems (TOIS)*, 11(1):1–26, 1993.

[GSV00] N. Güell, D. Schwabe, and P. Vilain. Modeling interactions and navigation in web applications. In *Proceedings of the World Wild Web and Conceptual Modeling'00 Workshop*, 2000.

[GWG97] H. Gellersen, R. Wicke, and M. Gaedke. Webcomposition: an object-oriented support system for the web engineering lifecycle. In *Selected papers from the sixth international conference on World Wide Web*, pages 1429–1437. Elsevier Science Publishers Ltd., 1997.

[HBI97] A. Hester, R. Borges, and R. Ierusalimschy. Cgilua: A multi-paradigmatic tool for creating dynamic www pages. 1997.

[HPSS87] D. Harel, A. Pnueli, J. P. Schmidt, and R. Sherman. On the formal semantics of statecharts. In *Proc. 2nd. IEEE Symposium on Logic in Computer Science*, 1987.

[IB99] M. Imaz and D. Benyon. How stories capture interactions. In *Proceedings of Human-Computer Interaction - INTERACT'99*, 1999.

[ISB95] T. Isakowitz, E. A. Stohr, and P. Balasubramanian. Rmm: a methodology for structured hypermedia design. *Communications of the ACM*, 38(8):34–44, 1995.

[JBR99] I. Jacobson, G. Booch, and J. Rumbaugh. *The Unified Software Development Process*. 1999.

[Lan94] D. Lange. An object-oriented design method fir hypermedia information systems. In *Proceedings of the 27th annual Hawaii international conference on System science*, 1994.

[MD00] S. Murugesan and Y. Deshpande. Iweb engineering for sucessfulweb application development. In *Proceedings of the 21st international conference on Software engineering*, 2000.

[MW01] A. MCDONALD and R. WELLAND. Web engineering in practice. In *Proceedings of the 21st international conference on Software engineering*, pages 693–694. The University, Glasgow G12 8QQ, 2001.

[Pre02] R. S. Pressman. *Software engineering (5nd ed.)*. McGraw-Hill, Inc., 2002.

[RBP+91] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen. *Object-oriented modeling and design*. Prentice-Hall, Inc., 1991.

[Ros96] G. Rossi. *OOHDM.* PhD thesis, PUC-Rio, 1996.

[RSL99] G. Rossi, D. Schwabe, and F. Lyardet. Improving web information systems with navigational patterns. In *Proceedings of 8th International World Wide Web Conference*, 1999.

[S.A] ILOG S.A. Ilog cplex 7.1 user's manual.

[SCGP92] D. Schwabe, A. Caloini, F. Garzotto, and P. Paolini. Hypertext development using a model-based approach. *Software Practice and Experience*, 22(11):937–962, 1992.

[SF89] P. D. Stotts and R. Furuta. Petri-net based hypertext: Document structure with browsing semantics. *ACM Transactions on Information Systems*, 7(1), 1989.

[SFR92] P. D. Stotts, R. Furuta, and J. C. Ruiz. Hyperdocuments as automata: Trace-based browsing property verification. In *Proceedings of the ACM European Conference on Hypertext*, 1992.

[SM01] D. Schwabe and A. P. Medeiros. Especificação declarativa de aplicações web em oohdm. 2001.

[SMMM] A. G. Sutcliffe, N. A. M. Maiden, S. Minocha, and D. Manuel. Supporting scenario-based requirements engineering. *IEEE Transactions on Software Engineering.*

[SPM99] D. Schwabe, R. Pontes, and I. Moura. Oohdm-web: An environment for implementation of hypermedia applications in the www. 8(2), 1999.

[SR95] D. Schwabe and G. Rossi. The object-oriented hypermedia design model. *Communications of the ACM*, 38(8):45–46, 1995.

[SV] D. Schwabe and P. Vilain. Notação da metodologia oohdm.

[Tro00] O. De Troyer. *Audience-driven Web Design in Conceptual Modeling in the Next Millenium.* CRC Press, 2000.

[VSdS00a] P. Vilain, D. Schwabe, and C. S. de Souza. A diagrammatic tool for representing user interaction in uml. In *UML 2000 - Third International Conference on the Unified Modeling Language*, 2000.

[VSdS00b] P. Vilain, D. Schwabe, and C. S. de Souza. Use cases and scenarios in the conceptual design of web applications. Technical report, PUC-Rio, 2000.

[ZP92] Y. Zheng and M-C Pong. Using statecharts to model hypertext. In *Proceedings of the ACM European Conference on Hypertext*, 1992.