UNIVERSIDADE DE SÃO PAULO

UMA MODELAGEM DO DOMÍNIO DE ENGENHARIA REVERSA DE SOFTWARE UTILIZANDO O MÉTODO OOHDM

VALÉRIA DELISANDRA FELTRIM

RENATA PONTIM DE M. FORTES

Nº 40

NOTAS

Instituto de Ciências Matemáticas de São Carlos

Instituto de Ciências Matemáticas de São Carlos

ISSN - 0103-2577

UMA MODELAGEM DO DOMÍNIO DE ENGENHARIA REVERSA DE SOFTWARE UTILIZANDO O MÉTODO OOHDM

VALÉRIA DELISANDRA FELTRIM

RENATA PONTIM DE M. FORTES

Nº 40

NOTAS DO ICMSC
Série Computação

São Carlos
jun.1998

Uma modelagem do domínio de Engenharia Reversa de Software utilizando o método OOHDM[1]

Valéria Delisandra Feltrim
Renata Pontin de M. Fortes
{vfeltrim, renata}@icmsc.sc.usp.br

Departamento de Ciências da Computação e Estatística
Instituto de Ciências Matemáticas e de Computação
Universidade de São Paulo

Resumo

O processo de engenharia reversa de software visa a recuperação de informações do software com base em uma intensa interação com os documentos e o software propriamente dito. Os hiperdocumentos tem se apresentado como alternativa apropriada para o suporte a esse tipo de interação com documentos. Dessa forma, este trabalho apresenta uma modelagem conceitual e navegacional do domínio de processo de engenharia reversa de software baseado no método de engenharia reversa Fusion-RE/I. Para a modelagem foi utilizado o método para projeto de aplicações hipermidia OOHDM, método o qual também é abordado neste trabalho.

Palavras-Chave: Processo de Engenharia Reversa de Software, Modelagem de Hiperdocumentos, Métodos para Projeto de Hiperdocumentos, OODHM.

1. Introdução

Tanto a manutenção como o reuso de componentes de software requerem uma análise do sistema alvo e uma recuperação do seu projeto para um melhor entendimento do sistema por parte do engenheiro de software. Tal tarefa de recuperação não é trivial, pois muitas vezes a documentação disponível é pouca ou nenhuma, ou ainda desatualizada, e o código fonte não retrata explicitamente decisões tomadas durante a etapa de projeto do sistema. Nesse ponto, como

[1] Este trabalho tem o apoio financeiro da FAPESP, #97/12999-8

auxílio às tarefas de recuperação das informações de projeto e análise do sistema alvo, surgem os métodos e técnicas de engenharia reversa, como ferramenta à recuperação de informações relevantes e à uma compreensão adequada do sistema em si.

Em [5] encontramos engenharia reversa como "o processo de exame e compreensão do sistema existente, para recapturar ou recriar os requisitos atualmente implementados pelo sistema, apresentando-os em um grau ou nível mais alto de abstração. Não envolve mudanças no sistema ou criação de um novo sistema".

Engenharia reversa de software é um processo de examinação, não um processo de mudança ou duplicação [3].

Em termos comparativos, tendo definido engenharia progressiva como um processo que inicia-se na análise dos requerimentos, passa pelo projeto até a implementação do sistema, a engenharia reversa é vista como o oposto da engenharia progressiva.

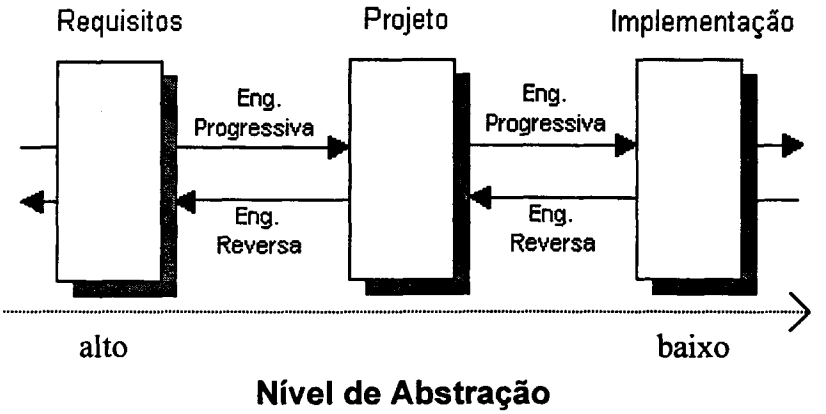


Figura 1: Processo de Eng. Progressiva x Eng. Reversa

O processo de engenharia progressiva é caracterizado por uma seqüência de atividades que podem ser descritas através de um modelo de ciclo de vida. Na literatura encontramos diferentes formalizações desses modelos, sendo que embora todos passem por atividades básicas que apresentam-se em todas as formalizações, a existência de interações entre as partes de cada modelo diferem, bem como a existência de atividades e produtos específicos a cada formalização.

Na engenharia progressiva, o sistema é o resultado do processo de desenvolvimento. Na engenharia reversa, o sistema geralmente é o ponto inicial do processo [3]. A Figura 1, adaptada de [3], representa esquematicamente, uma forma comparativa das direções inversas por quais se orientam as etapas envolvidas em cada uma das engenharias. O processo de engenharia reversa

caracteriza-se pelas atividades retroativas do ciclo de vida, que partem de um baixo nível de abstração para um alto nível de abstração.

Embora exista a caracterização de um processo para a engenharia reversa, não existe a formalização de um "modelo de ciclo de vida" para as atividades envolvidas na engenharia reversa de software. Existem, por outro lado, métodos de engenharia reversa que impõem uma ordem na execução das atividades propostas por cada método, bem como estabelecem os produtos resultantes da realização de tais atividades. Neste contexto, este trabalho apresenta uma modelagem do processo de engenharia reversa através de um hiperdocumento de apoio às atividades desenvolvidas durante o processo; trata-se justamente de uma tentativa de formalizar o relacionamento dessas atividades, seus subprodutos e interações. Para tal, nos apoiaremos no método de engenharia reversa Fusion-RE/I [5], o qual servirá como base para o estabelecimento das atividades envolvidas e dos modelos gerados ao final do processo.

Este trabalho apresenta uma primeira modelagem do domínio de processo de engenharia reversa de software segundo o método Fusion-RE/I. Para essa modelagem foi adotado o método OOHDM – *Object Oriented Hypermidia Design Method*, um método para modelagem de aplicações hipermídia orientado a objetos. Uma vez que o método de Engenharia Reversa utilizado, o Fusion-RE/I, também é orientado a objetos, a utilização do OOHDM nos pareceu mais natural. Esta modelagem deverá ser validada através da realização do exercício de engenharia reversa sobre o ambiente hipertexto SASHE [10], uma vez que o processo a se realizar será uma instância do referido modelo proposto.

A fim de estabelecer o método de modelagem hipermídia a ser utilizado para modelar o domínio do processo de engenharia reversa, foi feito um estudo de algumas abordagens presentes na literatura que pudessem satisfazer os nossos objetivos. Dessa forma, na seção a seguir são brevemente discutidos quatro outros métodos para a modelagem conceitual de dados hipertexto.

Na Seção 3 é descrito o OOHDM, visto que, como já mencionado acima, o OOHDM foi o método de modelagem de dados que melhor se enquadrou nos objetivos propostos. Finalmente na Seção 4 é apresentada a modelagem OOHDM do domínio do processo de engenharia reversa de software segundo o método Fusion-RE/I, seguido pela conclusão do trabalho.

2. Modelagem de Hiperdocumentos

Diversos modelos lógicos de dados têm sido desenvolvidos para hiperdocumentos. Eles se baseiam principalmente na noção de composição e *links* rotulados/ tipados. Os *links*

representam um papel de conexão semântica em sistemas hipertexto, segundo Wang [16] e são o suporte às conexões diretas entre os nós.

Os trabalhos que propõem a modelagem lógica dos dados de um hiperdocumento visam fornecer uma forma de planejar o hiperespaço de nós e *links* e de adicionar recursos não contemplados na concepção original, tais como a definição de abstrações e o controle de versões.

Na realidade, muito embora muitos dos trabalhos não tratem exclusivamente da modelagem lógica do hiperdocumento, pois também fornecem especificações de características funcionais para sistemas hipertexto ou hipermídia, eles se fundamentam prioritariamente em modelagem de dados.

Segundo Fortes [6], os trabalhos encontrados na literatura com propostas de modelagem lógica de um hiperdocumento se baseiam em um dos dois enfoques de modelagem lógica de dados, a saber:

- **Estruturação das ligações** contidas no hiperdocumento - se referem aos modelos de hiperdocumentos que consideram principalmente a definição dos *links*. Uma vez que seja conhecido o domínio da aplicação ou da informação à qual o hiperdocumento servirá de base, os modelos sob este enfoque de estruturação de *links* visam definir os diversos tipos de ligação que darão a estrutura geral de um hiperdocumento, baseados na identificação das abstrações das informações (como composições, generalizações e outras).
- **Estruturação dos nós** contidos no hiperdocumento - se referem aos modelos para hiperdocumentos que se orientam principalmente por mecanismos que garantam que os nós sejam documentos logicamente estruturados. Documentos ODA e SGML são estruturados e bem definidos. Nessa linha, o surgimento de padrões e de meta-linguagens que os suportem têm se destacado.

Sob o enfoque de estruturação dos nós contidos no hiperdocumento, as pesquisas encontradas na literatura tiveram origem na busca por uma representação abstrata do hiperdocumento que permitisse sua transmissão e armazenamento em diferentes plataformas, sejam elas de hardware ou de software. Essa representação abstrata do hiperdocumento, que inclui a estruturação de todos os seus componentes (nós e *links*), garante que os nós sejam documentos logicamente estruturados.

A modelagem de um hiperdocumento, sob o enfoque de estruturação das ligações, tem o objetivo de organizar de forma coerente as estruturas do hiperdocumento, ou seja, as ligações entre os nós, de forma a permitir um melhor entendimento pelo leitor do hiperespaço planejado pelo autor e auxiliar o autor no próprio desenvolvimento do hiperdocumento.

Nas próximas subseções são apresentados os modelos encontrados na literatura que se apresentam como principais referências para a modelagem conceitual (cujo o enfoque é o de estruturação dos nós) de um hiperdocumento, com recursos para definição de esquemas conceituais de dados.

2.1. HDM

O primeiro modelo formal de projeto de hiperdocumento voltado especialmente para aplicações hipermídia foi o HDM *(Hypertext Design Model)* de Garzotto [7]. Ele sugere a utilização de uma abordagem estruturada para o desenvolvimento de hiperdocumentos, com a modelagem do domínio de informações de uma aplicação como suporte para a autoria das páginas e de suas ligações. Esta proposta visa diminuir os dois problemas típicos da utilização de sistemas hipertexto, desorientação do usuário e sobrecarga cognitiva.

O HDM prescreve a definição de um esquema de aplicação, o qual descreve todas as classes de elementos de informação em termos das suas características de apresentação comuns, suas estruturas de organização internas, e os tipos de suas interconexões mútuas. O HDM é principalmente um dispositivo de modelagem. Ele fornece maneiras de descrever de forma concisa e independente de sistema, aplicações hipertexto existentes ou a serem desenvolvidas [7].

Dentre as inovações de HDM estão: a noção de perspectiva, a identificação de diferentes categorias de *links* (estruturais, de aplicação e de perspectiva) com diferentes papéis de representação, a distinção entre a hiperbase (como é denominada a base de documentos) e as estruturas de acesso e a possibilidade de integrar facilmente a estrutura de uma aplicação hipertexto à sua semântica de navegação.

Uma das principais vantagens da modelagem através de HDM para projeto e construção de aplicações hipertexto é a derivação automática da definição de um número significativo de *links* a partir de uma descrição em nível de projeto conceitual.

2.2. RMM

A motivação para a proposta do método RMM *(Relationship Management Methodology)* [11], está na necessidade de técnicas para gerenciamento de projeto e de desenvolvimento formal de sistemas, para assegurar que um produto hipermídia alcance seus objetivos e seja terminado dentro das expectativas de tempo e de orçamento previstas. Para tanto, as técnicas convencionais utilizadas na indústria de software devem ser alteradas para atender os seguintes requisitos:

- O projeto de aplicações hipermídia em geral envolve pessoas com habilidades bastante distintas: autores, bibliotecários, projetistas de conteúdo, artistas, músicos e programadores;
- O projeto de aplicações hipermídia inclui a captura e organização de uma estrutura de um domínio complexo além de ter que torná-la clara e acessível aos usuários;
- Os aspectos de multimídia das aplicações hipermídia acrescentam diversas dificuldades.

Isakowitz et al. [11], criadores do RMM, ainda destacam que o projeto hipermídia é um desafio que atualmente está qualificado mais como arte do que ciência. Finalmente, a necessidade de prototipação e testes intensivos com o usuário é ainda mais pronunciada porque a tolerância a erros que o usuário de hipermídia possui é bastante baixa.

O RMM se destina prioritariamente à classe de aplicações que exibem uma estrutura regular no domínio de interesse, ou seja, em domínios onde existam classes de objetos cujos relacionamentos possam ser definidos entre essas classes e múltiplas instâncias de objetos dentro de cada classe. Além disso, muitas aplicações hipermídia pertencentes a essa classe possuem dados "voláteis" que requerem freqüentes atualizações, o que por sua vez configura a necessidade de meios para sistematizar e automatizar ambos: o desenvolvimento inicial e os subseqüentes processos de atualização. Nesse contexto, o RMM se apresenta como uma abordagem bastante adequada para o projeto de aplicações hipermídia contendo informações estruturadas com alta volatilidade.

O modelo de dados utilizado como base para o RMM é chamado RMDM *(Relationship Management Data Model)*. O RMDM fornece uma linguagem para descrever os objetos de informação e os mecanismos de navegação utilizados em aplicações hipermídia. Trata-se de uma linguagem gráfica, semelhante à utilizada nos diagramas do modelo entidade-relacionamento de Base de Dados, que inclui representações para primitivas do domínio e de acesso.

2.4. OOHDM

O OOHDM é um método voltado para o desenvolvimento de aplicações hipermídia que descreve as tarefas a serem executadas desde a análise de domínio da aplicação até a sua implementação [14]. Ele é um descendente direto do HDM [7], incorporando uma série de novos conceitos vindos sobretudo da orientação a objetos.

O OOHDM propõe que o desenvolvimento de aplicações hipermídia seja um processo dividido em quatro etapas: modelagem conceitual, modelagem da navegação, projeto abstrato da interface e implementação.

Na próxima seção serão detalhados os aspectos de cada etapa da modelagem segundo a metodologia OOHDM, sendo dado ênfase as etapas desenvolvidas no exemplo apresentado na Seção 4.

3. O Método de Projeto para Aplicações Hipermídia OOHDM

O OOHDM considera o processo de desenvolvimento de uma aplicação hipermídia como um processo de quatro atividades, desempenhadas em uma mistura de estilos iterativos e incrementais de desenvolvimento; em cada etapa um modelo é construído ou enriquecido [12]. Na figura abaixo, é mostrado um esboço do OOHDM, destacando cada uma das quatro fases que compõem o método, juntamente com os produtos gerados em cada fase, os mecanismos de abstração utilizados na fase correspondente, e qual o foco empregado em cada uma das fases do método.

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 da 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 2: Esboço da Metodologia OOHDM [12]

A modelagem conceitual dos dados é feita utilizando-se os princípios da modelagem orientada a objetos, onde o principal objetivo é construir um modelo do domínio da aplicação. O resultado é um modelo conceitual composto por subsistemas, classes e relacionamentos.

Na modelagem da navegação é feita a descrição da estrutura navegacional da aplicação, em termos de contextos navegacionais, levando em conta os tipos de usuários da aplicação e suas tarefas.

Nós representam janelas lógicas (ou visões) sobre as classes conceituais definidas durante a análise do domínio. Os *links* são derivados dos relacionamentos conceituais. Diferentes modelos navegacionais podem ser feitos para o mesmo modelo conceitual, expressando diferentes visões do mesmo domínio.

Definidos os nós e os links, pode ser modelado o movimento dentro do espaço navegacional, independente do modelo conceitual.

No projeto da interface abstrata é construído um modelo especificando quais objetos de interface serão vistos pelo leitor, e particularmente, a forma de apresentação dos diferentes objetos navegacionais, as transformações causadas pela ativação desses objetos, quais objetos ativarão a navegação, a maneira como os objetos multimídia serão sincronizados e quais transformações ocorrerão na interface.

Podem ser feitos diferentes projetos de interface abstrata para o mesmo projeto navegacional, de acordo com as preferências dos leitores.

Na etapa de implementação é feito o mapeamento dos objetos de interface para objetos de implementação.

3.1. Modelagem Conceitual

Nesta fase é construído um modelo conceitual do domínio em questão. O foco nesta fase e modelar os objetos que constituem o domínio e os relacionamentos entre eles. Na modelagem conceitual pode-se observar uma maior preocupação com a estrutura dos objetos do que com seu comportamento. Para tal é empregada a abordagem orientada a objetos, segundo a notação do método OMT [13], com a adição de alguns elementos novos, como o conceito de subsistemas e de perspectiva de atributos.

Um subsistema representa um esquema completo de modelagem hipermídia de um domínio que está inserido dentro de outro domínio. Dessa forma, podemos definir subsistema como abstrações de um sistema conceitual completo [12]. Pode existir relacionamentos entre

classes e subsistemas, sendo que pode haver apenas um ou múltiplos pontos de entrada para o subsistema.

O modelo é definido através de classes, subsistemas, e relações organizados de acordo com a semântica do domínio. Na Seção 4 é apresentado um modelo conceitual para o domínio do processo de engenharia reversa de software.

As classes são organizadas através dos mecanismos de abstração de composição e especialização/generalização, como acontece em outros métodos orientado a objetos. A cardinalidade das relações também consta do modelo, expressando o número de objetos que participam das mesmas.

Os atributos representam propriedades dos objetos e possuem um tipo. O tipo do atributo pode especificar um relacionamento implícito ou a aparência que o atributo terá na aplicação final. Cada aparência possível é denominada perspectiva do atributo. Um atributo pode múltiplas perspectivas (um exemplo dessa característica é um atributo "descrição" em uma Obra de Arte poder ser visto como um texto ou uma imagem [12]). Esse conceito de perspectiva de atributo foi herdado do HDM de Garzotto [7]. As diferentes perspectivas de um atributo são mostradas entre colchetes ([...]), sendo que a perspectiva *default* é precedida de um sinal +. A perspectiva *default* é a que estará presente em todas as instâncias da classe definida.

O OOHDM usa cartões como um mecanismo extra para documentar classes e relacionamentos, sendo que cada tipo de cartão possui características próprias para se documentar cada tipo de primitiva. A principal justificativa apresentada para o uso de cartões é a sua facilidade de manipulação.

Segundo Rossi [12], ao término dessa etapa do método teremos concluído como produtos um modelo conceitual, ou um conjunto de esquemas conceituais, sendo um para cada subsistema, e um conjunto de definições de subsistemas, classes, objetos e relacionamentos especificados em Cartões de Subsistemas, Classes, Relacionamentos e Objetos. Estes cartões são mostrados na respectivamente nas Figuras 3, 4, 5 e 6, encontradas originalmente em [12].

Nome de Sub-Sistema	
Inclui (nome das classes)	
Relação Classe Relacionado com	
Pontos de Entrada	
Comentários (planos de implementação, etc)	
Trace para trás	Trace para frente

Figura 3: Cartão de Subsistema

Nome da Classe	Herda de:
Atributos e Partes	
Comportamento	
Classe Relação Relacionado con	
Parte-de:	
Comentários :(Planos de implementação, etc)	
Trace para trás	Trace para frente

Figura 4: Cartão de Classe

Nome da Relação-Cardinalidade	
Atributos	
Com Relaciona	
Comentários (planos de implementação, etc)	
Trace para trás	Trace para frente

Figura 5: Cartão de Relacionamento

Nome de Objeto (Classe)	
Acrescenta: (atributos e partes)	
Valores de Partes e atributos	
Relações	
Comentarios:	
Trace para trás	Trace para frente

Figura 6: Cartão de Objeto

3.2. Modelagem Navegacional

O modelo navegacional especifica a estrutura da navegação, ou seja, os objetos que efetivamente serão visualizados pelo usuário na aplicação final. Durante o projeto de navegação, as classes são mapeadas para nós, enquanto os relacionamentos em do modelo conceitual são mapeados para elos.

O OOHDM propõe a construção de um modelo conceitual que represente de forma generalizada todo o domínio em estudo. Diferentes visões navegacionais serão "derivadas" desse modelo conceitual, levando em conta os perfis dos futuros usuários, especificando quais porções de informação serão visualizadas e como estarão conectadas no hiperdocumento. O modelo conceitual passa a funcionar como um repositório compartilhado de modelagem, a partir do qual

diferentes visões podem ser construídas [12]; cada uma delas constituirá um tipo distinto de aplicação hipermídia implementando uma visão diferente do mesmo domínio.

A modelagem navegacional gera dois produtos principais: um esquema de classes navegacionais e um esquema contextual ou de contextos de navegação. Neles são descritos as classes de nós, as classes de ligações, as estruturas de acesso e os contextos de navegação que compõem a aplicação [2].

O esquema de classes navegacionais especifica classes de nós e as ligações entre elas, ou seja, as classes do domínio que serão efetivamente navegadas, quais informações pertencentes a classe serão visualizadas pelo usuário alvo, e quais as conexões que fazem sentido naquela visão navegacional.

O esquema contextual define as estruturas de acesso que estarão disponíveis para o acesso das informações representadas no esquema de classes navegacionais e os contextos em que cada classe de nó poderá ser acessada, especificando desta forma, quais objetos podem ser acessados dentro de um mesmo contexto.

Na Seção 4 são mostrados o esquema navegacional e o esquema contextual para o modelo conceitual elaborado para o domínio do processo de engenharia reversa de software.

Um nó define uma visão de um objeto pertencente a classes do modelo conceitual. Uma classe de nó é caracterizada pela classe do domínio da qual foi gerada, pelos atributos que possuí e pelas classes de ligação com as quais se relaciona. Pode ser um mapeamento direto da classe conceitual ou uma combinação das suas características. Classes de nós não possuem múltiplas perspectivas, sendo que uma classe conceitual com mais de uma perspectiva irá gerar mais de uma classe de nó, ou diferentes atributos para cada uma das perspectivas possíveis.

Uma classe de ligação implementa um relacionamento definido no esquema conceitual. As classes de ligação são especificadas através de seus atributos, seu comportamento, sua cardinalidade, e seus objetos de ligação (fonte e destino da ligação).

As estruturas de acesso constituem os pontos de entrada para as informações presentes no hiperdocumento, e podem ser vistas como índices, roteiros guiados, *timelines*, etc.

Os contextos de navegação procuram "aninhar" os objetos que fazem parte de um mesmo assunto, promovendo assim, um maior conforto para o usuário que navega pelo hiperdocumento, pois haverá um menor risco de sentir-se perdido dentro da rede de ligações do hiperdocumento. A idéia de contextos navegacionais foi herdada do Modelo de Contextos Aninhados - MCA de Casanova [1].

Dessa forma, podemos definir um contexto de navegação como um conjunto de objetos que compõem tópicos relacionados.

Segundo Rossi [12], no processo de desenvolvimento OOHDM, uma vez que entendemos e especificamos quais os objetos navegacionais que acessaremos durante a navegação, devemos organizá-los em contextos navegacionais coerentes. O mesmo objeto navegacional pode aparecer em diferentes contextos e novos objetos navegacionais (como elos de contexto) podem ser definidos durante a criação do contexto navegacional.

Por exemplo, tendo um conjunto de obras de arte, podemos acessar uma obra pelo seu tema, e a partir disso, continuar acessando todas as obras do mesmo tema. Assim, as obras seriam acessadas no contexto "obras de arte por tema", e nesse contexto haveria elos de contextos interligando todas as instâncias do objeto "obra de arte". Do mesmo modo, poderíamos acessar obras de arte pelo ano em que foram feitas, e da mesma forma, continuar acessando todas as obras realizadas naquele ano. Teríamos dessa forma outro contexto, "obras de arte por ano". No entanto, uma obra pode ser acessada dentro do contexto "obras por tema", e a partir desse ponto, permitir o acesso às obras datadas do mesmo ano.

Há muitos tipos de contextos navegacionais, variando de acordo com o modo como são originados. Alguns tipos podem ser inferidos do esquema de classes navegacionais. Em Rossi [12] são comentados algumas fontes para a definição de contextos navegacionais.

Um contexto navegacional arbitrário é definido pela escolha arbitrária de seus componentes, como acontece em um roterio guiado. Várias instâncias de classes navegacionais diferentes podem fazer parte de um roteiro guiado.

Um contexto navegacional derivado de classe é aquele formado por todas as instâncias de uma determinada classe de nó. Este tipo de contexto pode ser especializado, de forma que apenas as instâncias que satisfaçam uma determinada condição pertençam ao contexto, como por exemplo, todas as obras de arte sobre um determinado tema. Assim, mais de um contexto pode ser especificado para uma mesma classe de nó.

Um contexto navegacional derivado de um elo é especificado a partir de uma relação *um-para-muitos*. O contexto é constituído pelo conjunto de nós que podem ser acessados a partir de um determinado elo. Um exemplo é uma obra de arte que pode ser exibida em muitas exposições, constituindo dessa forma, um elo *um-para-muitos* entre "obra de arte" e "exposição". O conjunto de todas as exposições em que a obra foi exibida forma um contexto navegacional.

Os nós compostos (agregações) também geram um tipo de contexto navegacional, sendo que o conjunto das partes componentes do nó formam o contexto.

Também são utilizados cartões para a especificação de contextos navegacionais, como nos modelos anteriores. Nos cartões são especificados nome e tipo de contexto, atributos do contexto, as instâncias de classes pertencentes ao contexto juntamente com algum predicado que especialize o contexto, se houver uma especialização no contexto. Também são incluídos os contextos que permitem o acesso a partir do contexto sendo descrito, o modo como os elementos do contexto estão organizados para a navegação, entre outros, como comentários e outros elementos de modelagem relacionados ao contexto (*trace*). A Figura 7 apresenta um cartão de contexto.

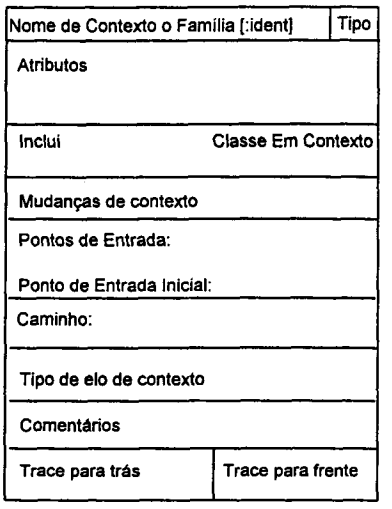


Figura 7: Cartão de Contexto [12]

A definição completa de como procede a navegação dentro dos contextos navegacionais e estruturas de acesso encontram-se em [12].

3.3. Projeto da Interface Abstrata

Segundo Rossi [12], os métodos atuais de projeto de hipermídia tendem a negligenciar a questão da definição de interface para aplicações hipermídia, e as interfaces humano-computador de aplicações desse tipo costumam ser totalmente construídas com uso de ferramentas dependentes da implementação e do ambiente. A ênfase encontrada nesses métodos está nos projetos conceitual e navegacional, sem considerar os problemas associados à interface.

Numa terceira etapa do método OOHDM, encontra-se a definição de um projeto de interface abstrata. Diz-se "interface abstrata" devido ao fato de, embora a interface da aplicação seja especificada, isso é feito sem levar em conta decisões de implementação e de forma

independente do ambiente no qual a interface será implementada. Dessa forma, mais de um projeto de interface pode ser definido para o mesmo projeto navegacional, apresentando diferentes aspectos de interface relacionados aquela visão navegacional.

Para tal especificação, o OOHDM utiliza o modelo de ADVs – *Abstract Data Views*, para definir os aspectos estruturais da interface. Um ADV é um objeto abstrato, com um estado e uma interface que interage com outros ADVs ou ADOs. Diz-se abstrato pelo fato de, embora tenha estado e defina uma interface, não especifica questões de implementação. Um ADO - *Abstract Data Object* pode ser visto como um *container* de informações estruturadas e que é considerado um ADV que não dá suporte aos eventos de interface. Um ADV pode apresentar na interface dados vindos de um ou mais ADOs e um ADO pode ser referenciado por um ou mais ADVs [2].

Segundo Castro [2], sob a perspectiva de projeto de aplicações hipermidia, um ADV pode ser visto como um objeto de interface compreendendo um conjunto de atributos que definem suas propriedades de percepção e um conjunto de eventos que podem ser manipulados por ele. Assim, para o OOHDM, os ADOs são definidos pelos nós, ligações e estruturas de acesso enquanto os ADVs especificam sua apresentação para o usuário. Os ADVs podem utilizar mecanismos de agregação e generalização/especialização para sua especificação.

A interação entre os ADVs e os objetos navegacionais (ADOs) é especificada através de Diagramas de Configuração. Esses diagramas definem os eventos externos que interagem com o ADV e as mensagens geradas e envias para o ADO proprietário do ADV, para que o mesmo possa responder aos eventos. Um exemplo de diagrama de configuração é mostrado na Figura 8, onde as setas contínuas representam eventos externos e as setas pontilhadas mensagens geradas.

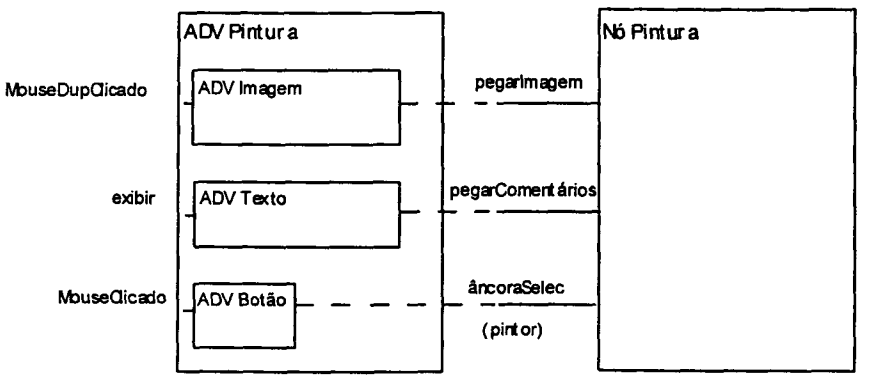
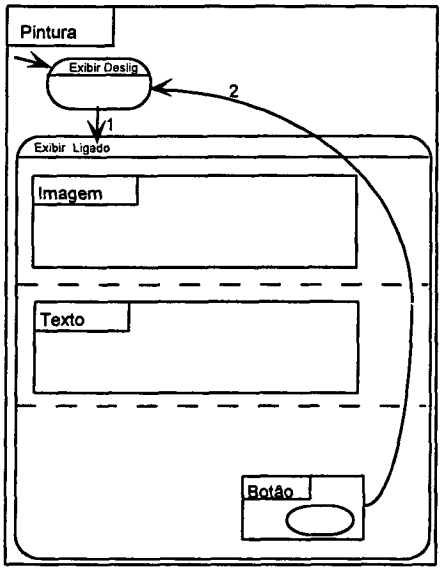


Figura 8: Diagrama de Configuração

O comportamento dinâmico da interface é especificado através de *ADVCharts*, que define a dinâmica de apresentação dos ADVs. Sendo uma generalização dos *StateCharts* [8] e dos *ObjectCharts* [4], os *ADVCharts* especificam as mudanças de estado do ADV após os eventos manipulados por ele, bem como seus ADVs aninhados [2].

Um *ADVchart* é composto por ADVs, estados, atributos e transições. Os estados são conectados uns aos outros por meio de transições. Neste formalismo, uma transição é especificada pela afirmação do evento que a dispara, das suas pré-condições, que devem ser satisfeitas para que a transição seja desencadeada, e das suas pós-condições, a lista de ações a serem executadas a cada vez que a transição é disparada [12]. Assim como em *StateCharts* e Diagramas de Navegação, os ADVs e os estados podem ser aninhados em modo And ou Xor.

Um exemplo de um *ADVChart* é apresentado na Figura 9 para o ADV mostrado no exemplo de diagrama de configuração (Figura 8).



Transições

1:
Evento: Exibir
Pré-Cond:
Pos-Cond: contextoPerceptivo = contextoPerceptivo + Pintura

2:
Evento: MouseClicado
Pré-Cond: foco (Botão)
Pos-Cond: contextoPerceptivo = contextoPerceptivo -Pintura; dono.âncoraSelec (pintor)

Figura 9: *ADVChart* para o ADV Pintura

"ContextoPerceptivo" é uma variável reservada que especifica quais os ADVs e elementos de interface que estão perceptíveis pelo usuário. Quando um ADV deixa de ser exibido, ele é retirado da lista ContextoPerceptivo.

Como mostrado na figura acima, os ADVs são apresentados como caixas retangulares, enquanto os estados são mostrados como caixas arredondadas. As setas representam as transições de estado, sendo que o nome, pré-condições e pós-condições das transições são mostradas ao lado do *ADVChart*.

A semântica de um ADV e um estado é diferente, uma vez que um ADV é um objeto que abstrai estrutura e comportamento, os estados apenas expressam aspectos comportamentais da interface [12].

O produto gerado pelo projeto da interface abstrada, é um conjunto de ADVs, diagramas de configuração e ADVCharts. Para complementar esse conjunto, existem cartões de projeto para

a especificação do comportamento estático e dinâmico de um ADV. Ambos os tipos de cartões complementam o diagrama apresentado anteriormente, fornecendo informações completas sobre a interface.

3.4. Implementação

Fica óbvio que um projeto OOHDM favorece sua implementação em um ambiente orientado a objetos, uma vez que, durante todas as etapas da modelagem da aplicação foram utilizados conceitos do paradigma de orientação a objetos. Dessa forma, os objetos modelados poderiam ser diretamente implementados como nós, e a especificação da interface também seria direta através da implementação dos *ADVCharts*.

No entanto, os ambientes para desenvolvimento de aplicações hipermídia não apresentam tais características, ou apresentam alguns conceitos orientados a objetos, mas não suportam todo o paradigma.

Em Rossi [12] é discutido a implementação de um projeto OOHDM em duas plataformas: em um ambiente de desenvolvimento hipermídia tradicional e em um ambiente orientado a objetos de propósito geral.

A primeira abordagem permite uma fácil tradução das instâncias das classes de navegação para nós. O problema dessa implementação é a falta dos conceitos orientados a objetos, já que esse tipo de ambiente é essencialmente baseado em nós e ligações [2]. A criação de mais de uma visão navegacional da mesma base de dados implicaria na duplicação de informações, uma vez que os objetos de dados e de interface estão presentes no mesmo objeto, o nó. Também no caso de uma alteração na apresentação de uma classe navegacional também implicaria na atualização do conteúdo de todos os nós instâncias da classe, pois os dados e a interface estão implementados no mesmo nó.

A implementação do projeto OOHDM em um ambiente de programação orientado a objeto de propósito geral tem como vantagem a disponibilidade para a implementação de todos os conceitos OO utilizados durante o projeto. O problema dessa abordagem encontra-se na complexidade de implementação, principalmente no que diz respeito ao comportamento navegacional existente nos hipertextos [2], e no esforço de codificação requisitado.

4. Modelagem do Domínio de Processo de Engenharia Reversa de Software

4.1. Esquema Conceitual

O modelo de classes do processo de engenharia reversa (PER) foi derivado do estudo desse processo sob a perspectiva do método de engenharia reversa Fusion-RE/I, segundo demonstrado na Figura 10. Os elementos presentes no modelo podem ser agrupados em duas hierarquias principais: uma hierarquia de tarefas de engenharia reversa e uma hierarquia de produtos gerados.

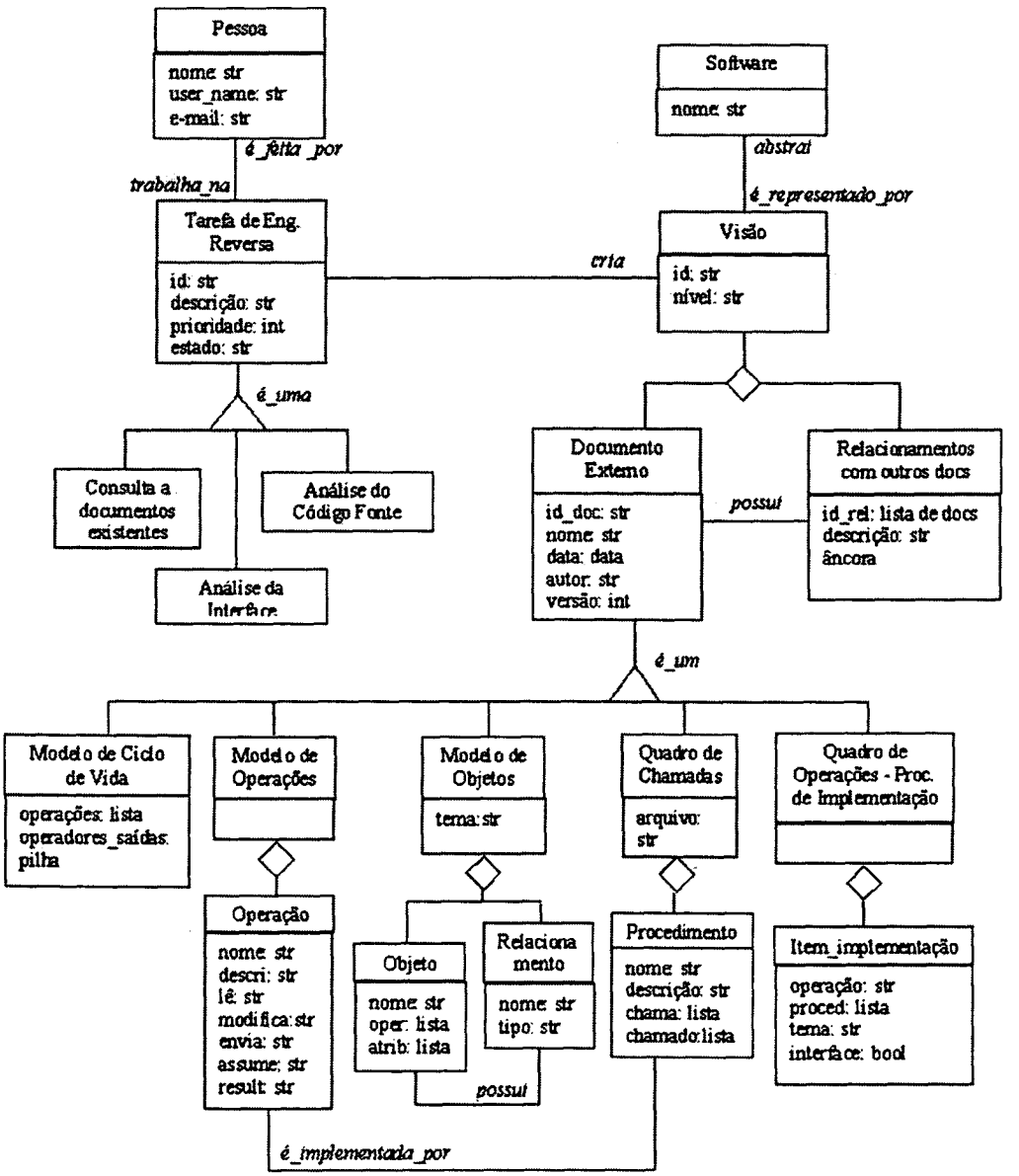


Figura 10: Modelo Conceitual do PER

As classes *Pessoa* e *Tarefa de Engenharia Reversa* (TER) definem a parte do modelo correspondente ao gerenciamento de projeto. Como TER corresponde a atividade de recuperação de informações relevantes à geração de uma visão do sistema, por isso o relacionamento um-para-um entre TER e *Visão*. *Pessoa* está relacionada com TER apesar dessa tarefa poder ser vista como mais de uma atividade. Na modelagem, entretanto, foi considerado TER como uma tarefa que pode ser desenvolvida por uma única pessoa e não existe a necessidade de se controlar a execução da TER por mais de uma pessoa, daí o relacionamento muitos-para-um entre TER e *Pessoa*. São as TER que conectam a parte de projeto e os produtos gerados. Uma TER pode ser uma: *Consulta a Documento Existente*, *Análise da Interface*, ou *Análise do Código Fonte*. Essas tarefas são descritas pelo método Fusion-RE/I, sendo que, segundo o método, apresentam uma escala de prioridade para a realização de cada uma delas. No decorrer do processo, no entanto, essa prioridade pode mudar de acordo com as necessidades do engenheiro reverso, e certas tarefas podem ser novamente realizadas.

Uma *Visão* define uma interpretação do sistema sob Engenharia Reversa. Essa interpretação corresponde a um nível de abstração, sendo que pode ser estrutural, funcional ou de domínio. O Fusion-RE/I propõe recuperação de visões tanto ao nível estrutural como funcional. Assim, a representação de um sistema é composta por um conjunto de visões, e isso é expresso pelo relacionamento um-para-muitos entre *Software* e *Visão*.

Uma *Visão* é um agregado de documentos e dos relacionamentos existentes entre esses documentos. Um documento é um *Documento Externo* que descreve informações recuperadas por uma TER. *Relacionamento com outros Docs* descreve uma ligação entre um documento e seus documentos relacionados.

Documento Externo é um dos modelos ou quadros propostos como resultado do Fusion-RE/I. No esquema conceitual (modelo de classes), todos os documentos são implicitamente relacionados através de seus atributos. As operações contidas no *Modelo de Ciclo de Vida* são as mesmas que compõem o *Modelo de Operações*. Da mesma forma, cada operação descrita em *Operação* e constituinte do *Modelo de Operações* e é parte de um *Objeto*. *Modelo de Objetos* é um agregado de objetos relacionados sendo que o atributo *Tema* indica o tema que inspirou o *Modelo de Objetos* em questão. O *Quadro de Operações-Procedimentos de Implementação* é um agregado de *Item*, onde cada *Item* compõe uma entrada do quadro. As operações citadas nesse modelo também são as mesmas descritas nos modelos mencionados anteriormente. Da mesma forma, os procedimentos presentes em cada *Item* também são descritos em *Procedimento*. A agregação de *Procedimento* compõe os *Quadros de Chamadas*, cujo atributo *arquivo* indica o arquivo onde se encontra o código que implementa *Procedimento*. O relacionamento explícito entre *Operação* e *Procedimento* se faz necessário uma vez que, apesar de existir o

relacionamento (uma operação é implementada por um procedimento), não existem atributos relacionados. O relacionamento um-para-muitos se explica por uma operação ser implementada por um ou mais procedimentos.

4.2. Esquema de Classes Navegacionais

Como pode ser visualizado na Figura 11, o esquema de classes navegacionais é derivado do esquema conceitual descrito anteriormente, e é composto pelas classes que serão efetivamente navegadas, ou seja, serão visualizadas pelo usuário do aplicativo.

Dentro do processo descrito anteriormente, as classes navegáveis serão os produtos gerados pelas TER (modelos e quadros). Com exceção do *Modelo de Ciclo de Vida* e do *Modelo de Objetos*, todos os outros produtos são uma agregação de outros objetos. Dessa forma, as classes que compõem a agregação é que serão navegados, e não a classe agregada.

Assim, as classes componentes do esquema navegacional são *Operação*, *Objeto*, *Procedimento*, *Item*, *Modelo de Objetos* e *Modelo de Ciclo de Vida*. Nesse esquema tornam-se explícitos os relacionamentos que no esquema conceitual estão implícitos por meio dos atributos contidos nas classes. Esses relacionamentos são expressos pelas setas rotuladas. O direcionamento das setas foi decidido considerando os objetos *Operação*, *Objeto*, *Procedimento*, *Modelo de Ciclo de Vida* e *Modelo de Objetos* como possíveis entradas para o início da navegação. A partir disso os outros objetos foram relacionados de forma que todos pudessem ser alcançados e os relacionamentos fossem coerentes.

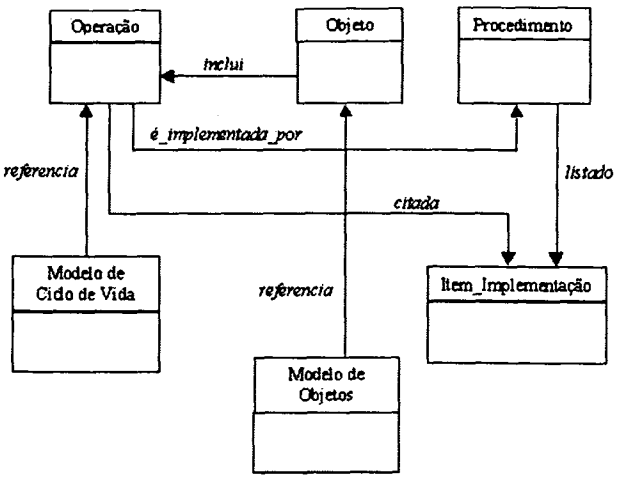


Figura 11: Modelo de Classes Navegacionais

4.3. Esquema Contextual

Os contextos navegacionais são geralmente induzidos pelas classes navegacionais e expressam a estrutura navegacional geral da aplicação, enquanto as classe navegacionais especificam os objetos que serão vistos pelo usuário [12].

Os contextos são um conjunto de instancias relacionadas, como, por exemplo, todos procedimentos em um determinado arquivo, todos os procedimentos com uma determinada data, etc. No esquema contextual as classes navegáveis são mostradas juntamente com os contextos em podem aparecer na aplicação. Os contextos são representados por caixas pontilhadas que aparecem dentro da classe a qual o contexto pertence. Para simplificação do diagrama, os contextos podem ser agrupados em quadros pontilhados maiores quando conveniente. As setas indicam navegação ou mudança de contexto. Quando uma seta chega a um agrupamento de contexto, isto indica que qualquer um dos contextos pertencentes ao agrupamento estará disponível para a navegação, sendo que os contextos da classe que não estiverem no agrupamento não serão visíveis. Os índices também aparecem com caixas pontilhadas entre as setas de navegação.

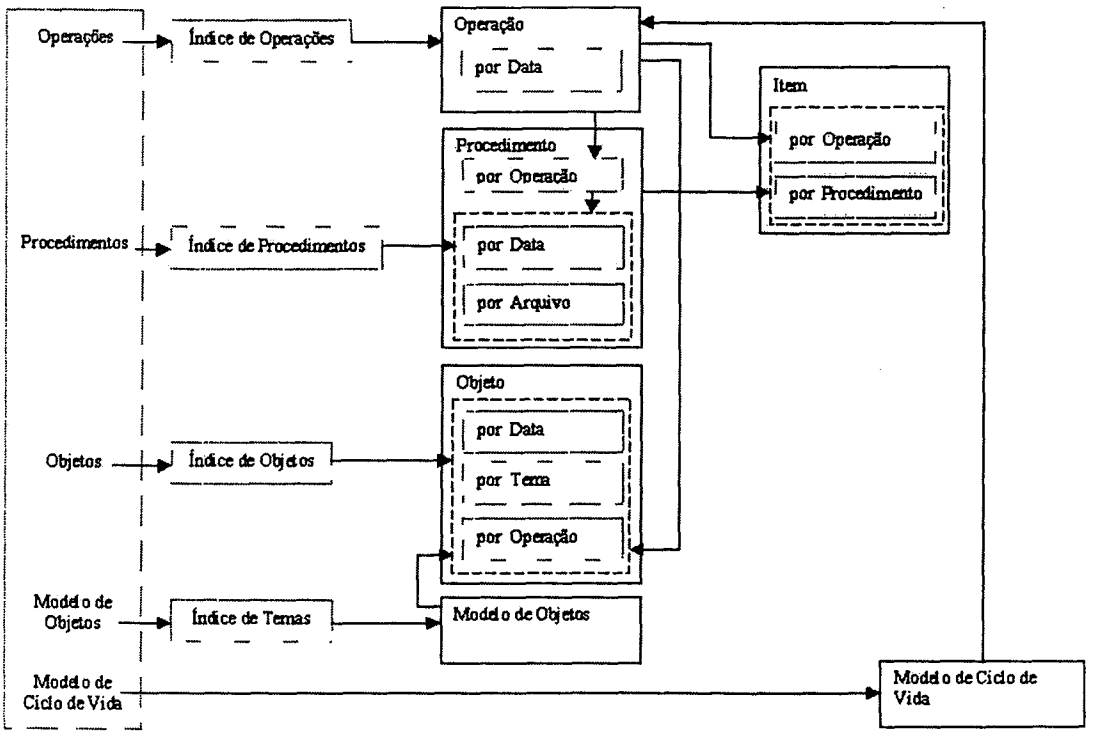


Figura 12. Modelo Contextual

No esquema contextual apresentado na Figura 12, o índice à esquerda é o principal, a partir do qual podem ser acessados diversos índices intermediários. Cada classe pode então ser navegada nos contextos mostrados na figura. Um exemplo de navegação em diferentes contextos pode ser visualizado na classe *Procedimento*. Uma vez que os procedimentos são acessados através do índice de procedimentos, cada instância da classe *Procedimento* pode ser navegada tanto no contexto *por data* como no contexto *por arquivo*. No entanto, quando a classe *Procedimento* é acessada através da classe *Operação*, a classe *Procedimento* passa a poder ser navegada no contexto *por operação*, uma vez que uma operação pode ser implementada por mais de um procedimento. A partir desse contexto, a classe *Procedimento* pode ser navegada também nos contextos *por data* e *por arquivo*. Dessa forma, o contexto *por operação* só é visível quando a classe é acessada através da classe *Operação*, e não aparece quando a classe é acessada através do índice.

A navegação mostrada nesse esquema é a mesma do esquema de classes navegáveis, porém em maior nível de detalhamento. Neste diagrama são visualizados os contextos em que cada classe pode ser navegada.

5. Conclusões e Trabalhos Futuros

Neste trabalho foi apresentada uma modelagem conceitual e navegacional do domínio de Engenharia Reversa de Software utilizando o método de projeto de aplicações hipermídia OOHDM [12], sendo a modelagem conceitual representada por um diagrama de classes e a modelagem navegacional composta pelo diagrama navegacional, contendo as classes do domínio que serão efetivamente navegadas, e pelo diagrama contextual, que especifica o contexto de navegação dessas classes.

Para tal modelagem, nos apoiamos nas etapas descritas pelo método de engenharia reversa Fusion-RE/I [5], um método para recuperação de projeto orientado a objetos desenvolvido no ICMC-SC. Por ambos os métodos serem orientados a objetos, a escolha do OOHDM nos pareceu mais adequada. De fato pode-se observar que apesar do domínio de informações em questão ser abstrato, por meio da abordagem de orientação a objetos os resultados da modelagem se mostraram representativos dos principais documentos produzidos pelo Fusion-RE/I, bem como seus relacionamentos.

Na linha dos próximos passos desta pesquisa, esta modelagem servirá de apoio ao projeto de adequação do ambiente protótipo SASHE ao domínio da engenharia reversa de software.

6. Referências Bibliográfica

[1] Casanova, M. A.; Tucherman, L; Lima, M. J.; Rangel, J. L.; Rodriguez, N. R.; Soares, L. F. G. *The Nested Context Model for Hyperdocuments*. Third ACM Conference on Hypertext, *Proceedings*, San Antonio, Texas, 1991, pp. 193-201.

[2] Cerqueira, A. A. C. *HOOT: Integrando Hipermídia e Bancos de Dados Orientados a Objetos*. Dissertação de Mestrado. UFRJ, Ago 1997.

[3] Chikofsky, E.J.; Cross II, J.H. *Reverse Engeneering and Design Recovery: A Taxonomy*. IEEE Software, v.7, n.1, pp. 13-7, 1990.

[4] Coleman, D. et al. *Desenvolvimento Orientado a Objetos: O Método Fusion*. Ed. Campos, Rio de Janeiro, 1996.

[5] Costa, Rejane M. *Um método de Engenharia Reversa para Auxiliar a Manutenção de Software*. Dissertação de mestrado, ICMSC–USP, São Carlos-SP, 1997.

[6] Fortes, R.P.M. *Análise e Avaliação de Hiperdocumentos: Uma Abordagem Baseada na Representação Estrutural*. Tese de Doutorado, IFSC-USP, São Carlos-SP, 1996.

[7] Garzotto, F.; Paolini, P.; Schwabe, D. *HDM - A Model-Based Approach to Hypertext Application Design*. ACM Transactions on Information Systems, v.11, n.1, p.1-26, January 1993.

[8] Harel, D. *Statecharts: A Visual Formalism for Complex Systems*. Science of Computer Programming, v.8, pp.231-274, 1987.

[9] Nunes, M.G.V.; Hasegawa, R.; Vieira, F.M.C. *Hip/Windows: Um Ambiente de Autoria de Hiperbases Multimídia*, Relatório Técnico, nº 38, ICMSC–USP, São Carlos-SP, 1996.

[10] Nunes, M.G.V.; Hasegawa, R.; Vieira, F.M.C.; Santos, G.H.R.; Fortes, R.P.M. *SASHE: Sistema de Autoria e Suporte Hipermídia para Ensino*. Notas, nº 33, ICMSC–USP, São Carlos-SP, 1997.

[11] Isakowitz, T.; Stohr, E.A., Balasubramanian, P. *RMM: A Methodology for Structured Hypermedia Design*. Communications of the ACM, v.38, n.8, p.34-44, August 1995.

[12] Rossi, G. *Um método orientado a objetos para o projeto de aplicações hipermídia*. Tese de Doutorado. Departamento de Informática Pontifícia Universidade Católica, Rio de Janeiro, 1996.

[13] Rumbaugh, J.; Blaha, M.; Premerlani, W.; Eddy, F.; Lorensen, W. *Object Oriented Modeling and Design*. Prentice Hall, 1991.

[14] Schwabe, D.; Rossi, G.; Barbosa, S.D.J. *Systematic Hypermedia Application Design with OOHDM* In: Hypertext'96, Washington DC, USA, March 1996. *Proceedings.* NewYork, ACM Press, 1996. p.116-28.

[15] Soares, L.F.G. et al. *HyperProp: Uma Visão Geral.* In: I Workshop em Sistemas Hipermídia Distribuídos, *Anais*, ICMSC-USP, São Carlos-SP, pp. 1-12, 1995.

[16] Wang, B.; Hitchcock, P.; Holden, T. *An Object Oriented Database Approach for Supporting Hypertext.* LNCS, p.601-28, 1992.

NOTAS DO ICMSC

SÉRIE COMPUTAÇÃO

039/98 FERREIRA, V.G.;MELLO, O .D.; OLIVEIRA, J.N.; FORTUNA, A . O . - Tópicos teóricos e computacionais em escoamentos de fluidos.

038/98 CAVICHIA, M.C.; ARENALES, M.N. - Piecewise linear programming via interior points.

037/98 REZENDE, S. O. ; PUGLIESI, J.B. - Aquisição de conhecimento explícito ou manual - versão 1.0

036/97 SOSSOLOTE, C.R.C; RINO, L.H.M; ZAVAGLIA, C.; NUNES, M.G.V. - As manifestações morfossintáticas da linguagem UNL no português do Brasil.

035/97 FRANCO, N.B. - A volterra integral equation arising from the propagation of non-linear waves

034/97 MARTINS, A.L.; OLIVEIRA, M.C.F.; MINGHIM, R. - Visualização Científica em mecânica dos fluidos.

033/97 NUNES, M.G.V.; HASEGAWA, R.; VIEIRA, F.M.C.; SANTOS, G.H.R.; FORTES, R.P.M. - SASHE: Sistema de autoria e suporte hipermídia para ensino [1].

032/97 DIOGO, T.; FRANCO, N..F. - High order product integration methods for Volterra integral equation with weakly singular Kernel.

031/96 ARENALES, M.N.; MORABITO, R. - A theory of the generation of cutting and packing patterns Part I: foundations.

030/96 TOMÉ, M..F.; CASTELO FILHO, A. ; MURAKAMI, J.; CUMINATO, J.A.; MINGHIM, R.; OLIVEIRA, M.C.F. - Numerical simulation of axisymmetric free surface flows.