3
OOHDM e SHDM

Com a disseminação em massa, desde a década de 80, de ambientes hipertexto e hipermídia, principalmente a Web, foi identificada a necessidade de elaborar métodos que estruturassem de maneira mais formal o projeto de aplicações para esses ambientes. Neste capítulo veremos uma descrição resumida dos métodos OOHDM, idealizado por Daniel Schwabe e Gustavo Rossi em 1995, e SHDM, idealizado por Fernanda Lima e Daniel Schwabe em 2003. Referências mais completas e detalhadas sobre estes métodos podem ser encontradas em [Schwabe & Rossi, 1998], [Lima, 2003] e [Szundy, 2004].

3.1. OOHDM

OOHDM (Object-Oriented Hypermedia Design Method) é um método baseado em modelos e que utiliza técnicas de orientação a objetos para o projeto de aplicações hipermídia. O grande diferencial deste método para construção de aplicações hipermídia, comparado a métodos tradicionais de engenharia de software, é a destacada relevância dada ao aspecto navegacional da aplicação. Para isso, o método introduz conceitos básicos importantes como:

- classes em contexto e objetos navegacionais: visões das classes e dos objetos conceituais em um contexto navegacional;
- contextos navegacionais: define um espaço de objetos navegacionais relacionados por um critério definido no projeto da aplicação, e que pode ser percorrido pelo usuário geralmente de maneira linear e uniforme;

A primitiva de contexto navegacional é especialmente importante, pois permite ao projetista da aplicação definir conjuntos de objetos relacionados que,

se bem implementados, podem ser examinados rapidamente pelo usuário, o que é uma tarefa bastante comum dentro de uma aplicação hipermídia.

Além disso, um contexto navegacional influencia diretamente a maneira como os objetos navegacionais são visualizados pelo usuário. Um objeto navegacional pode tomar diferentes formas de apresentação dependendo do contexto navegacional em que se encontra, seja apresentando informações extras, suprimindo informações redundantes ou outras formas de adaptação. Essa capacidade polimórfica dos objetos navegacionais se mostra muito eficaz em aplicações hipermídia.

3.2. SHDM

SHDM (Semantic Hypermedia Design Method) é uma evolução do método OOHDM que leva em consideração os formalismos e primitivas introduzidas pela Web Semântica. O método permite projetar aplicações hipermídia baseadas em ontologias, descritas através de metadados. Além disso, introduz o conceito de navegação facetada, oferecendo primitivas de construção correspondentes. O método SHDM mantém a mesma estrutura de fases de modelagem já citadas para o método OOHDM, mantendo a separação entre modelos conceituais e navegacionais, contextos navegacionais e outros conceitos do método original. No entanto, SHDM enriquece esses modelos através de construções com maior poder expressivo e formaliza alguns novos conceitos que constituem uma evolução real do método original.

3.3. Fases

Os métodos OOHDM e SHDM prevêem 5 fases principais: levantamento de requisitos, projeto conceitual, projeto navegacional, projeto de interface abstrata e implementação. Em cada fase um modelo é produzido e refinado, gerando um conjunto de artefatos relacionados. Apesar de apresentadas de maneira seqüencial, estas 5 etapas não precisam ser executadas na exata ordem em que aparecem aqui e apenas uma única vez, como em um processo em cascata. Estas etapas podem ser executadas de maneira iterativa e cíclica, dependendo da metodologia de processo de desenvolvimento adotada.

3.3.1. Levantamento de Requisitos

A etapa de levantamento de requisitos tem como objetivo principal a identificação dos atores e tarefas a serem apoiadas pela aplicação. Para isso, devem ser produzidos descrições dos cenários e casos de uso, que são apresentados através de narrativas em formato textual. Além destes, devem ser produzidos ainda os diagramas de interação com o usuário (User Interaction Diagrams - UIDs) que servirão de base para uma validação através de representação gráfica do comportamento da aplicação. Podemos visualizar um exemplo deste tipo de diagrama a seguir:

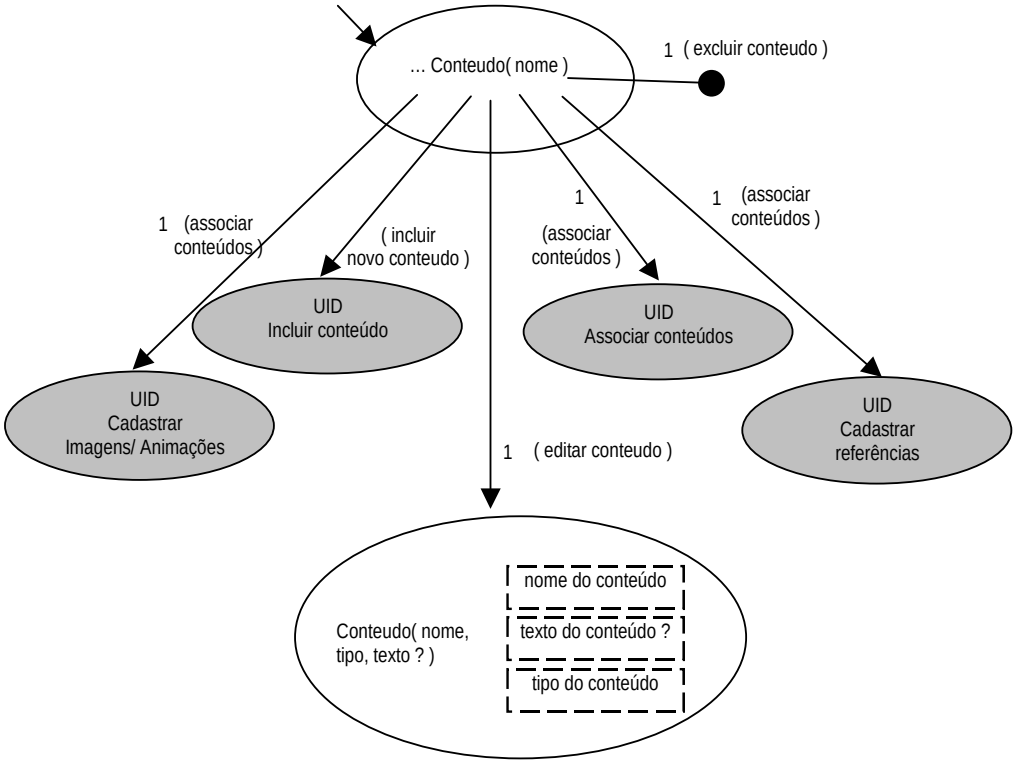

Figura 5 - Exemplo de UID como apresentado por [Leite 01]

Para detalhes sobre a notação deste diagrama, ver [Vilain, 2002].

3.3.2. Projeto Conceitual

Na etapa de projeto conceitual, é produzido o modelo conceitual da aplicação, sem levar em consideração ainda aspectos navegacionais ou relativos a plataforma de implementação. No método OOHDM, este modelo conceitual pode ser representados através de primitivas de orientação a objeto, como classes e

atributos, utilizando notação UML pura. No método SHDM, o modelo conceitual é definido através de uma ontologia, definidas em linguagem RDFS ou OWL.

3.3.3. Projeto Navegacional

A etapa de projeto navegacional introduz a criação do modelo de navegação, mapeado a partir do modelo conceitual. O modelo navegacional permite definir primitivas que irão representar o projeto de navegação da aplicação. Para isso, o modelo conceitual sofre uma série de transformações para geração deste novo modelo de navegação, onde classes conceituais são mapeadas em classe navegacionais, relações entre as classes são mapeadas em elos, que por sua vez permitem definir novos tipos de atributos navegacionais como âncoras e índices.

Além do mapeamento do esquema conceitual em um esquema navegacional, que pode ser representado visualmente através de uma versão customizada da notação UML, a modelagem navegacional deve produzir ainda o esquema de contextos navegacionais, utilizando notação própria, e que irá permitir a representação de primitivas navegacionais como *landmarks*, estruturas de acesso e contextos de navegação. Este esquema permite a representação concisa de todos os caminhos de navegação oferecidos pela aplicação. Exemplos deste esquema serão apresentados mais adiante nesta dissertação.

3.3.4. Projeto de Interface Abstrata

A etapa de projeto de interface abstrata deve produzir o modelo abstrato de interface, onde são definidas as visões que o usuário poderá obter sobre as instâncias de objetos navegacionais, que chamamos de nós. Os elementos de uma interface abstrata são mapeados entre as primitivas definidas no modelo navegacional e elementos de apresentação abstratos independentes de plataforma, representados através de ADVs (*abstract data views*). A seguir podemos ver um exemplo de ADV:

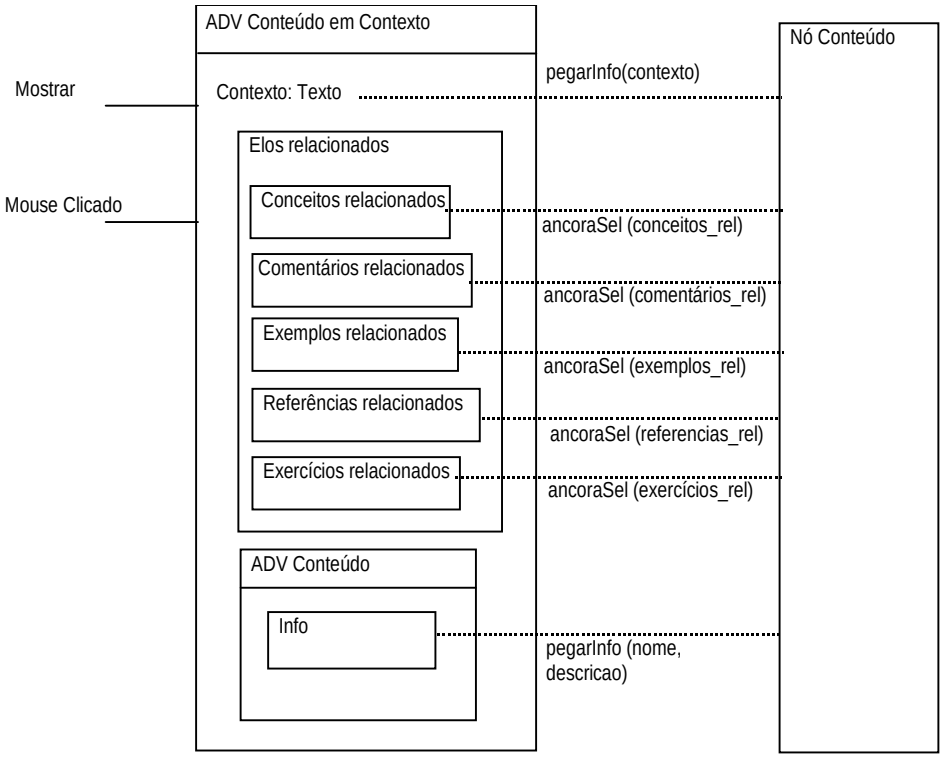

Figura 6 - Exemplo de ADV como apresentado em [Leite 01]

Na etapa de implementação, tais modelos abstratos são utilizados para mapear os elementos de apresentação abstratos em elementos de apresentação concretos, dependentes da plataforma de implementação. O projeto de interface abstrata, incluindo os mapeamentos feitos a partir de primitivas navegacionais e para elementos de apresentação concretos, pode ser visto em grande detalhe em [Moura, 2004].

3.3.5. Implementação

A etapa de implementação tem como objetivo transformar todos os modelos definidos nas etapas anteriores em uma aplicação executável sobre uma plataforma de hardware e software escolhida. Para isso pode ser feito uso de frameworks, ferramentas, linguagens de programação e ambientes de desenvolvimento que podem reduzir o esforço necessário para a execução desta etapa. A seguir podemos ver o exemplo de uma tela de aplicação projetada com o método OOHDM e implementada em plataforma Web utilizando *framework* ASP.NET.

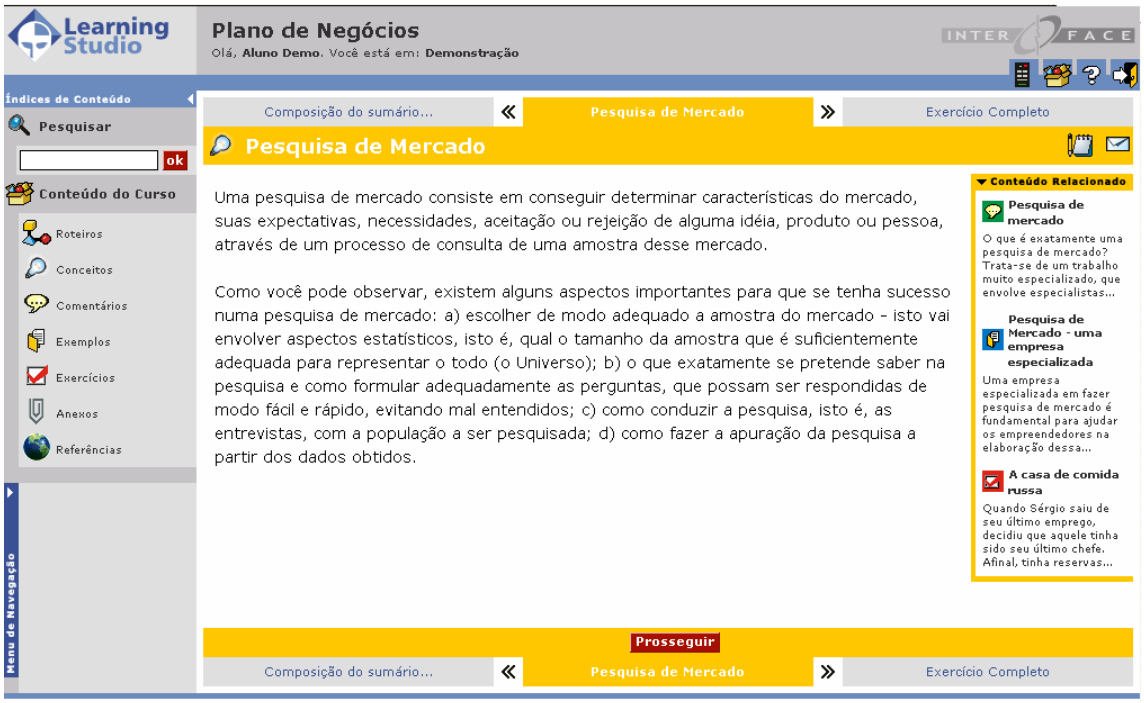

Figura 7 - Exemplo de aplicação OOHDM implementada em plataforma ASP.NET

3.4. Modelagem Navegacional

Um modelo navegacional constitui uma visão específica mapeada sobre as primitivas do modelo conceitual, definindo quais informações e de que forma estas informações poderão se acessadas através do ato de navegação pelo usuário da aplicação. O modelo navegacional tem como objetivo dar suporte ao usuário nas tarefas definidas para a aplicação em questão e por conta disso, mais de um modelo navegacional pode ser construído em função da variedade de tipos de usuário e tarefas a serem suportadas pela aplicação.

O modelo navegacional pode ser representado através do esquema de classes navegacionais e pelo esquema de contextos navegacionais. Através do primeiro é possível visualizar quais informações do modelo poderão ser acessadas, enquanto que através do segundo é possível ter uma melhor visão dos caminhos navegacionais possíveis que poderão ser usados para fazer este acesso.

Nas seções a seguir, apresentaremos de forma resumida, uma descrição sobre as primitivas navegacionais introduzidas pelos métodos OOHDM e SHDM e utilizadas na etapa de projeto navegacional para a construção de um modelo navegacional.

3.4.1. Classes Navegacionais

Classes navegacionais são primitivas navegacionais construídas em função das classes definidas no modelo conceitual, transformadas através da perspectiva de tarefas a serem executadas pelo usuário da aplicação. A definição de uma classe navegacional é composta por seus atributos, operações e elos, além das relações de especialização/generalização que podemos utilizar para constituir uma heranças de classes. Às instâncias de classes navegacionais chamamos de nós.

O mapeamento entre as classes conceituais e navegacionais ocorre de acordo com o tipo de acesso que será feito às informações definidas no modelo conceitual. Isto na prática significa que classes conceituais podem se transformar em simples atributos no modelo navegacional ou que atributos podem se transformar em classes navegacionais, entre muitos outros tipos de mapeamentos possíveis. Por exemplo, se tivermos definidas classes “Livro” e “Autor”, com uma relação do tipo “possui” no modelo conceitual, mas na aplicação em questão não desejarmos realizar navegação sobre nós da classe “Autor”, poderíamos então simplificar a classe “Autor” e sua relação “possui” transformando-os apenas em um atributo da versão navegacional da classe “Livro”.

3.4.2. Atributos Navegacionais

Atributos navegacionais ajudam a compor a definição de uma classe navegacional. Podemos classificar estes atributos em 4 categorias:

- **Atributos Simples:** estes atributos possuem tipos diretamente mapeados com os tipos dos atributos no modelo conceitual, tais como “data”, “texto”, “booleano”, “inteiro”. Além disso, existem tipos específicos para utilização em aplicações hipermídia, que incluem “imagem”, “som”, “email”, “animação”, entre outros.
- **Atributos Computados:** estes atributos são definidos a partir de uma função computacional executada dinamicamente para determinação de seu valor durante o acesso ao nó na aplicação. O uso deste tipo de atributo foi introduzido pelo ambiente HyperDE.

- **Âncoras:** este tipo de atributo define um possível caminho de navegação entre o nó da classe navegacional sob o qual ele está definido e um outro nó ou estrutura de acesso. Este caminho é definido em função de uma relação navegacional, ao que chamamos de elo, e seu cálculo ocorre em tempo de execução da aplicação, podendo ser parametrizado.
- **Índices:** este tipo de atributo define uma associação entre a classe navegacional sob o qual ele é definido e uma estrutura de acesso da aplicação. Os valores do índice associado são incorporados ao nó durante a navegação e calculados em tempo de execução, podendo receber parâmetros para este cálculo.

3.4.3. Operações

As operações de uma classe navegacional são análogas a métodos definidos em um modelo orientado a objeto. Elas são definidas através de conjuntos de instruções computacionais e que podem ser ativadas durante a execução da aplicação. Geralmente, esta ativação é feita pelo usuário através de algum elemento de interface durante o acesso a um nó em determinado contexto navegacional. A operação é executada sob o escopo do nó e tem acesso aos atributos e relações definidos por sua classe navegacional, podendo potencialmente modificar seus valores. Quando executadas sob um contexto navegacional, as operações também têm acesso aos dados do contexto em questão.

3.4.4. Elos

Elos são as relações entre nós, formando potenciais caminhos navegacionais que poderão ser realizados na forma de âncoras definidas em classes navegacionais ou estruturas de acesso. Elos podem ser definidos a partir de mapeamento direto feito sobre as relações entre classes do modelo conceitual. No entanto, não necessariamente será esta a única forma de defini-los. Um elo pode ser definido ainda a partir do mapeamento indireto de um conjunto de relações entre classes do modelo conceitual como podemos ver no diagrama abaixo:

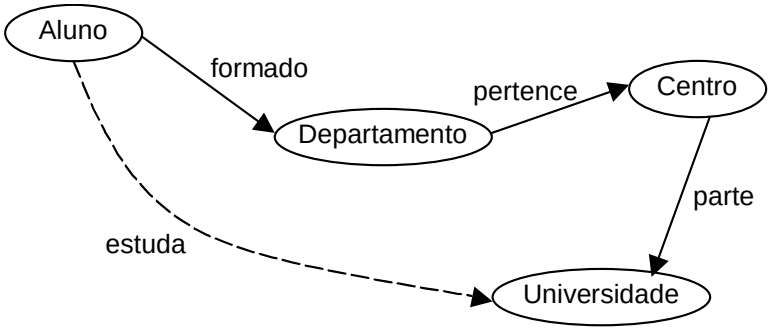

Figura 8 - Exemplo de elo definido a partir de um conjunto de relações como apresentado em [Szundy 04].

No exemplo acima, o elo “estuda” entre as classes “Aluno” e “Universidade” é definido a partir do caminho formado pelas relações “formado”, “pertence” e “parte” entre as classes “Aluno”, “Departamento”, “Centro” e “Universidade”.

Elos também podem ser definidos como especializações ou generalizações de relações entre classes do modelo conceitual. A definição de elos faz parte do modelo navegacional e são definidos estaticamente. Durate a execução, a aplicação poderá produzir instâncias destes elos, mas não novos tipos de elos.

3.4.5. Contextos Navegacionais

Um contexto navegacional é definido por um conjunto de nós. Em uma aplicação, um nó só pode ser acessado se este estiver inserido dentro de um contexto navegacional, no entanto o mesmo nó poderá fazer parte de vários contextos navegacionais. A seleção dos nós que farão parte de um contexto navegacional é feita a partir de uma expressão de consulta, utilizando-se geralmente de atributos ou meta-atributos do modelo navegacional como critérios de seleção. Esta expressão de consulta também se utiliza destes atributos para definir um ou mais critérios de ordenação, caso o acesso aos nós no contexto seja feito de forma seqüencial, o que geralmente ocorre.

Um exemplo de expressão de consulta que define um contexto navegacional pode ser: “todos os exemplos que compõem o capítulo 27”. Note que nesta expressão é utilizado como critério de seleção um meta-atributo do modelo navegacional, quando dizemos “todos os exemplos”, estamos nos referindo a todos os nós de uma determinada classe navegacional, neste caso a classe

"Exemplo". Também é feito uso de uma definição de elo, quando dizemos "que compõem", estamos nos referindo ao elo "compõe" entre a classe "Exemplo" e a classe "Capítulo". Finalmente, um valor de atributo da classe "Capítulo" também é utilizado como critério de seleção, ao especificarmos que apenas o capítulo com valor do atributo "número" igual a 27 deve ser considerado. Outros exemplos de contextos navegacionais podem ser: "disciplinas do departamento de informática", "músicas de Chico Buarque da década de 60", "corridas vencidas por Ayrton Senna em 1988", "os 10 livros de ficção mais comprados no último mês".

Um contexto navegacional possui ainda a propriedade do tipo de navegação que pode ser executada na aplicação, entre os nós que formam o contexto. Estes tipos de navegação incluem: navegação livre, onde qualquer nó do contexto pode ser acessado diretamente; navegação seqüencial, a mais comum, onde são definidas âncoras implícitas sobre o nó corrente que levam aos seus vizinhos diretos, anterior e próximo, dentro do contexto; navegação circular, similar a navegação seqüencial, com a diferença que o primeiro e último nós do contexto estão ligados como vizinhos diretos; navegação por índice, onde o auxílio de um índice associado ao contexto fornece acesso aos nós que fazem parte deste. Apesar do tipo de navegação em um contexto ser propriedade do contexto navegacional, na prática, a forma de navegação é geralmente definida pela interface da aplicação e pode ser composta por mais de um tipo simultaneamente.

3.4.6. Classes em Contexto

Uma classe em contexto é a aplicação do padrão de projeto de decorador, definido em [Gamma et al, 1995], sobre uma classe navegacional, quando os nós desta classe são acessados em determinado contexto navegacional. Um classe em contexto, portanto, poderá definir novos atributos navegacionais que existirão apenas quando um nó desta classe for acessado em um contexto navegacional específico, como também suprimir determinados atributos que forem redundantes naquele contexto. Por exemplo, se estivermos navegando em um contexto definido pela expressão "alunos orientados pelo Prof. Schwabe" e a classe "Aluno" possuir como atributo "orientador" que apresenta o nome de seu orientador, seria razoável que este atributo fosse suprimido já que a definição do

contexto já apresenta seu valor e que será constante em todos os nós que o formam.

Apesar de ser uma primitiva do modelo navegacional prevista nos métodos OOHDM e SHDM, na prática, geralmente esta propriedade de visibilidade dos atributos de um nó quando acessados sob determinado contexto navegacional que caracteriza uma classe em contexto é definida apenas no projeto de interface abstrata da aplicação. No ambiente HyperDE, esta primitiva é suportada através de visões customizadas definidas no projeto de interface.

3.4.7. Estruturas de Acesso (Índices)

Estruturas de acesso ou índices, são definidos a partir de um conjunto de âncoras que podem apontar para outros índices ou para nós em um determinado contexto navegacional. Índices podem aparecer de forma independente, quando geralmente são utilizados como pontos de entrada para contextos navegacionais, ou como atributo de um nó em contexto.

Um índice pode ser mais facilmente entendido como uma estrutura de dados bidimensional, análogo a uma tabela, onde uma dimensão (as colunas de uma tabela), é definida por atributos do índice, que podem ser de tipo simples ou âncora com alvo para outro índice ou nó em contexto. Para que o índice seja de real utilidade navegacional, pelo menos um desses atributos deverá ser de tipo âncora. Na outra dimensão (as linhas de uma tabela) estão os elementos do índice, formados pelos valores dos atributos (células de uma tabela). Os elementos de um índice são definidos de 4 formas e essa forma determina o tipo de índice:

- **Índice derivado de consulta:** este tipo de índice, assim como um contexto navegacional, tem seus elementos definidos por uma expressão de consulta. No entanto, diferente de expressão de consulta de contexto, a expressão de consulta de um índice não necessariamente deverá ter como resultado um conjunto de nós, mas qualquer conjunto de dados arbitrário. Um exemplo de expressão de consulta para um índice pode ser: “o nome dos professores do departamento de informática, o nome da principal área de pesquisa aos quais cada um está associado e a contagem de alunos orientados

por cada professor". Note que neste exemplo, o conjunto de dados é formado pelos valores dos nomes de professores e áreas de pesquisa e uma contagem. A definição de atributos deste índice poderá então fazer uso desses dados para a criação de âncoras para índices ou nós em contextos.

- **Índice derivado de contexto:** este tipo de índice é similar ao derivado por consulta, no entanto a expressão de consulta deste índice é exatamente igual à expressão de consulta do contexto navegacional associado e tem como resultado o mesmo conjunto de nós definidos para este contexto. Geralmente, mas não necessariamente, este tipo de índice é utilizado como ponto de entrada para os nós no contexto associado.
- **Índice arbitrário:** os elementos deste tipo de índice são definidos arbitrariamente, um a um. Como cada elemento é definido explicitamente, os atributos para cada elemento podem variar de elemento para elemento, no entanto, isto não é muito comum. O uso deste tipo de índice é geralmente evitado pela sua complexidade e esforço exigido em sua manutenção. O ambiente HyperDE em sua versão atual não dá suporte a este tipo de índice.
- **Índice facetado:** os elementos deste índice são baseados em uma composição de facetas, que veremos em seção mais adiante sobre navegação facetada. Como o ambiente HyperDE em sua atual versão não dá suporte a navegação facetada, este índice também não é tratado pelo ambiente.

3.4.8. Âncoras

Âncoras são objetos navegacionais que têm como função representar um caminho navegacional entre duas estruturas de acesso, entre dois nós em contexto e entre estruturas de acesso e nós em contexto. Âncoras, portanto, podem ser definidas como atributos de índices ou classes navegacionais, que representam a origem do caminho navegacional e têm como alvo um índice ou nó em contexto, representando o destino do caminho navegacional. O tipo de uma âncora é definido por seu alvo, portanto, existem âncoras para índices e âncoras para nós

em contexto. Âncoras para nós em contexto definidas como atributos de classes navegacionais também devem estar associadas a um elo da classe navegacional sob o qual o atributo foi definido e cujo valor do elo é usado para definir o nó de destino no contexto alvo.

Note que a definição de âncoras neste modelo é desacoplada de sua forma de visualização na aplicação, que poderá ser representada de diferentes maneiras pela interface com o usuário. O importante é que as âncoras representadas na interface serão remeterão às âncoras definida no modelo navegacional.

3.4.9. Landmarks

O método SHDM introduziu a primitiva de *landmark*. Um landmark define uma âncora que pode ser acessada de qualquer parte da aplicação. Esta âncora pode ser do tipo âncora para índice ou âncora para nó em contexto. Uma aplicação pode definir um número de arbitrário de landmarks. Landmarks também são geralmente utilizados (mas não necessariamente) para definir o ponto de entrada inicial da aplicação, ou seja, a visão sobre o alvo da âncora de um landmark (índice ou nó em contexto) é a primeira coisa apresentada ao usuário da aplicação quando este inicia seu uso.

3.4.10. Nós

Nós são os objetos navegacionais de uma aplicação OOHDM/SHDM. Nós portanto são necessariamente instâncias de uma classe navegacional definida no modelo navegacional da aplicação. Os nós de uma aplicação só podem ser acessados pelo usuário se inseridos dentro de um contexto navegacional. O projeto de interface da aplicação define a forma de exibição (visão) obtida pelo usuário sobre esses nós, que pode variar de contexto para contexto para um mesmo nó, como visto na definição de classes em contexto.

3.4.11. Navegação Facetada

Navegação facetada é um tipo de navegação introduzido pelo método SHDM por [Lima, 2003] e é definido por um conjunto de contextos navegacionais e estruturas de acesso baseados em um grupo de facetas.

Para ilustrar a necessidade deste novo conceito, antes de descrevê-lo, vejamos um exemplo apresentado por [Szundy, 2004]: uma classe navegacional "ObraDeArte" possui os atributos "período", "país" e "tipo". Suponha a tarefa de encontrar determinada obra de arte na aplicação, através da navegação em índices, cujo caminho definirá o critério de seleção do contexto navegacional sob o qual os nós do tipo "ObraDeArte" serão apresentados, ou seja, podemos inicialmente acessar o índice de períodos, onde encontramos os valores "século IV, século V, século VI". Selecionando um destes períodos, navegaríamos para o índice de países, já filtrado pela seleção anterior do período, onde veríamos os valores "França, Itália, Brasil". Selecionando um destes países, navegaríamos para o índice de tipos, já filtrado pelas seleções anteriores de período e país, onde veríamos os valores "pinturas, esculturas, retratos". Finalmente então, escolhendo um destes tipos veríamos o índice de obras de arte, baseado no contexto contendo os nós do tipo "ObraDeArte" cujo critério de seleção corresponde as escolhas de período, país e tipo, feitas pelo caminho de navegação realizado.

A primeira vista, esta tarefa é perfeitamente suportada pelas primitivas já definidas anteriormente, através da simples definição de índices e contextos navegacionais. No entanto, imagine que se desejasse que a de navegação entre os 3 primeiros índices, de períodos, países e tipos, pudesse ser realizada em qualquer ordem, dentre as 9 combinações possíveis, ou seja, seria possível navegar na ordem Períodos > Tipos > Países, ou Tipos > Países > Períodos, ou Países > Períodos > Tipos, etc. Para possibilitar a navegação em qualquer ordem, teríamos que definir 27 índices distintos, 9 índices para cada passo da navegação. Adicionando mais um índice nesta combinação, por exemplo, "Artistas", seriam necessários a definição de 64 novos índices. Teoricamente possível, mas na prática tornaria o modelo navegacional demasiadamente complexo.

O conceito de navegação facetada permite definir de forma concisa e flexível este tipo de navegação feito por particionamentos sucessivos. Uma faceta é definida por um critério de seleção dentro do universo de nós disponíveis na aplicação. Índices e contexto navegacionais facetados são definidos por um conjunto de facetas, que pode ser compostas em qualquer ordem, dinamicamente gerando os conjuntos de nós que irão compor os contextos ou índices subseqüentes. No exemplo anterior, portanto, bastaria definir apenas um índice facetado composto pelas facetas Período, Tipo, País e Artista, para que de forma

dinâmica, qualquer índice possível fossem gerado em tempo de execução na aplicação, baseado na seqüência de escolhas do usuário durante sua navegação.

O ambiente HyperDE em sua versão atual não dá suporte a navegação facetada, e portanto, não permite a definição de índices ou contextos facetados.