2
O método SHDM

O método SHDM, como proposto originalmente em Lima (2003), empregava um modelo orientado a objetos estendido para especificação do esquema conceitual e uma ontologia conceitual para representação desse esquema em um formato implementável na Web Semântica, no caso DAML-OIL (posteriormente OWL) com extensões específicas do método. Quando consideramos o objetivo discutido no capítulo anterior, de empregar o método para construção de aplicações hipermídia como visões navegacionais de dados da Web Semântica, o uso dessa abordagem em muitos casos envolveria um trabalho adicional de adaptação dos dados para a representação específica do método, uma vez que nem todas as abstrações de modelos orientados a objetos podem ser representadas diretamente como ontologias da Web Semântica (uma discussão dessa questão é apresentada mais adiante na seção *2.1-Modelo Conceitual*).

Por esta razão, a versão do método SHDM aqui apresentada é uma evolução da proposta original, empregando diretamente o modelo RDF[8] como a base estrutural do esquema conceitual. Tal mudança permite a utilização de qualquer ontologia definida para a Web Semântica (em OWL ou RDFS) como esquema conceitual da aplicação, sem a necessidade de adaptações. O esquema conceitual e a ontologia conceitual passam a ser uma coisa só.

Outra mudança incorporada nesta nova versão está relacionada à modelagem das operações específicas do domínio da aplicação sendo projetada. Tradicionalmente, em OOHDM tais operações eram modeladas como métodos de classes. Entretanto, a identificação de uma série de fatores indicou a necessidade de substituição dessa abordagem pela incorporação de um modelo para especificação dos processos e operações das aplicações hipermídia. Apesar da definição deste modelo de processos não fazer parte do escopo deste trabalho,

[8] Lembrando que aqui, a expressão "modelo RDF" é utilizada para denotar representações em RDF, de informações sobre domínios de acordo com os padrões RDFS ou OWL.

uma discussão mais detalhada sobre essa questão e as respectivas mudanças no método SHDM são apresentadas na seção *2.3-Operações*.

Estas foram as principais alterações conceituais do método SHDM. Outras alterações que dizem respeito a detalhes da definição da visão navegacional serão exploradas mais adiante. No demais, foi mantido o princípio do desenvolvimento de aplicações hipermídia utilizando uma abordagem iterativa baseada em modelos, e organizada em cinco etapas: Levantamento de Requisitos, Projeto Conceitual, Projeto Navegacional, Projeto de Interface Abstrata e Implementação. Também permaneceram inalterados os principais conceitos oriundos do método OOHDM: (1) existência de um modelo navegacional distinto do modelo conceitual, porém derivado deste segundo um mapeamento explícito; (2) existência de contextos de navegação no modelo navegacional; (3) existência de uma interface abstrata.

O desenvolvimento da aplicação envolve a produção de uma série de artefatos ao longo das etapas citadas anteriormente (listados na Tabela 1), e tipicamente segue o fluxo apresentado a seguir (os números entre chaves correspondem aos números na primeira coluna da tabela):

I. Identificação de atores e tarefas, especificação de cenários e casos de uso {1}, especificação dos diagramas de interação do usuário (UIDs) {2} e validação dos casos de uso e UIDs.
II. Criação da ontologia conceitual {3}, caso não seja empregada uma ontologia já existente.
III. Uma vez definida a ontologia conceitual as instâncias conceituais {4} podem ser geradas.
IV. Especificação do mapeamento navegacional {5}.
V. Especificação do projeto da interface abstrata {7}.
VI. Implementação da aplicação utilizando os artefatos {3}, {4}, {5}, {7} e {8}.

É importante notar que o artefato {8} geralmente é pré-definido, sendo atualizado apenas quando novas tecnologias de interface são introduzidas.

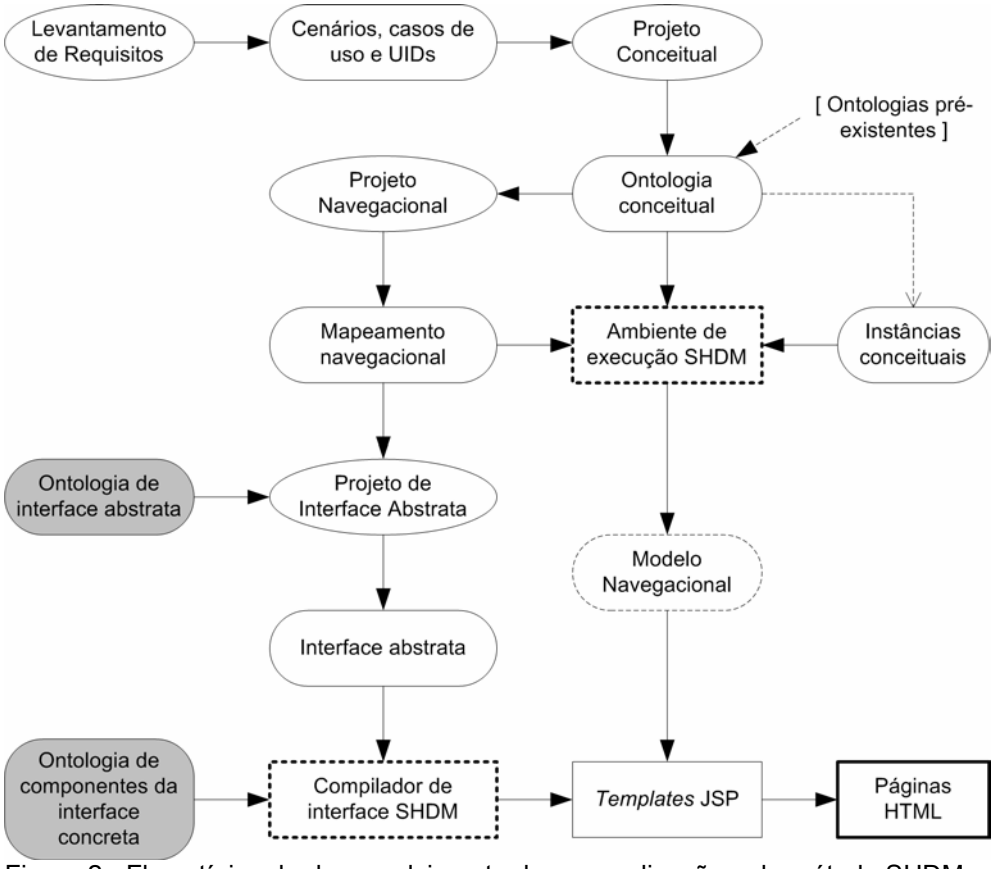

Figura 2 - Fluxo típico de desenvolvimento de uma aplicação pelo método SHDM

	Artefato	Descrição	Etapa
1	Descrição de cenários e casos de uso	Identificação dos atores e tarefas apoiadas pela aplicação.	Levantamento de Requisitos
2	UIDs	Diagramas de interação do usuário	Levantamento de Requisitos
3	Ontologia conceitual	Vocabulário para definição das instâncias conceituais. Pode ser qualquer ontologia da Web Semântica definida em OWL ou RDFS.	Projeto Conceitual
4	Instâncias conceituais	Dados do domínio da aplicação, definidos segundo a ontologia conceitual.	Projeto Conceitual
5	Mapeamento navegacional	Especificação dos mapeamentos de classes navegacionais e elos, e do espaço navegacional (contextos, estruturas de acesso e *landmarks*) Definido com um vocabulário específico do método.	Projeto Navegacional

6	Modelo navegacional	Definição do esquema e instâncias de classes navegacionais. Tipicamente devem ser gerados dinamicamente a partir do modelo conceitual e da especificação do mapeamento navegacional. Sua geração pode ser em tempo de execução ou em uma fase de pré-processamento da aplicação.	Projeto Navegacional
7	Interface abstrata	Definição de elementos da interface abstrata e seus mapeamentos para o modelo navegacional, e para componentes da interface concreta. Definida com um vocabulário específico do método.	Projeto de Interface Abstrata
8	Ontologia de componentes da interface concreta	Definição de possíveis componentes da interface concreta para uso na implementação. Definida com um vocabulário específico do método.	Projeto de Interface Abstrata

Tabela 1 - Artefatos do método SHDM

O presente trabalho está focado nos aspectos relacionados ao uso do modelo de dados RDF para especificação do modelo conceitual, e a criação da visão navegacional da aplicação pela definição do mapeamento navegacional. A etapa de levantamento de requisitos não será abordada e continua seguindo o que foi estabelecido para o método OOHDM por Vilain & Schwabe (2002). A eventual adaptação desta etapa para o método SHDM é objeto de trabalhos futuros. O projeto de interface abstrata também se encontra fora do escopo desse trabalho tendo sido assunto em outra dissertação de mestrado (Moura, 2004).

A seguir é apresentada uma breve análise do modelo conceitual através de uma introdução ao modelo de dados RDF e aos vocabulários RDFS e OWL, seguida de uma apresentação dos principais conceitos e abstrações do modelo navegacional.

2.1
Modelo conceitual

Como mencionado anteriormente, a versão do método SHDM proposta nesse trabalho emprega o modelo de dados do padrão RDF como metamodelo de seu modelo conceitual.A seguir são apresentadas as principais características do modelo de dados RDF e ao término desta seção estão relacionadas as principais distinções entre este modelo e o modelo orientado a objetos empregado anteriormente.

Originalmente o padrão RDF foi concebido para descrição de metadados sobre recursos na WWW, tais como o título, o autor, ou a data da última alteração de uma página *web*. Mas ao se generalizar o conceito de recurso, o modelo de dados RDF pode ser utilizado para representar informações sobre qualquer coisa que possa ser identificada unicamente na *Web* através de um URI, mesmo que esta coisa não esteja disponível na rede (Manola & Miller, 2004).

Em RDF os recursos são descritos em termos de propriedades (que também são identificadas por um URI) e dos valores dessas propriedades. As informações sobre um recurso são declaradas através de sentenças simples, na forma *Sujeito-Predicado-Objeto*, onde o sujeito é um recurso, o predicado uma propriedade e o objeto um valor literal ou outro recurso. Por exemplo, a informação de que *Miguel de Cervantes é o autor de Don Quixote*, pode ser reescrita como:

*"**Don Quixote** tem uma propriedade **autor**, cujo valor é **Miguel de Cervantes**".*

Se o livro Don Quixote for tratado como um recurso, identificado pelo URI `http://www.example.org/DonQuixote`, a propriedade autor for identificada pelo URI `http://purl.org/dc/elements/1.1/creator`, esta informação pode então ser representada por uma sentença RDF no formato *Sujeito-Predicado-Objeto*, onde:

- `http://www.example.org/DonQuixote` é o **sujeito**;
- `http://purl.org/dc/elements/1.1/creator` é o **predicado**;
- "Miguel de Cervantes" é o **objeto**;

Prefixos de *namespaces* podem ser utilizados para simplificar a expressão dos URIs de recursos e propriedades. A tabela a seguir lista os prefixos utilizados nos exemplos apresentados ao longo desta dissertação.

Prefixo	URI do *namespace*
cds	http://example.com/cd#
dc	http://purl.org/dc/elements/1.1/
ex	http://www.example.org/
foaf	http://xmlns.com/foaf/0.1/
nSchema	http://www.tecweb.inf.puc-rio.br/shdm/nSchema#
owl	http://www.w3.org/2002/07/owl#
rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs	http://www.w3.org/2000/01/rdf-schema#
shdm	http://www.tecweb.inf.puc-rio.br/2004/06/shdm-schema#
xsd	http://www.w3.org/2001/XMLSchema#

Tabela 2 - Prefixos dos *namespaces* utilizados na dissertação

Utilizando estes prefixos, a sentença sobre o livro Don Quixote apresentada anteriormente pode ser expressa em uma notação de triplas mais legível, como mostrado abaixo.

```
ex:DonQuixote    dc:creator    "Miguel de Cervantes" .
```

Quadro 2 - Exemplo de declaração de uma sentença RDF

Com a declaração de outras sentenças, mais informação pode ser dada sobre o recurso identificado pelo URI `http://www.example.org/DonQuixote`, como por exemplo, que este recurso é um livro (*ex:Book)*, e que este recurso tem um título (*dc:title)* cujo valor é “Don Quixote”.

```
ex:DonQuixote    rdf:type      ex:Book .
ex:DonQuixote    dc:title      "Don Quixote" .
ex:DonQuixote    dc:creator    "Miguel de Cervantes" .
```

Quadro 3 - Exemplo de declaração de um conjunto de sentenças RDF

Como mencionado no capítulo anterior e exemplificado na Figura 1, qualquer conjunto de triplas RDF pode ser representado como um grafo, com nós e arcos representando os recursos, propriedades e seus valores.

Além do modelo de dados, o padrão RDF também define algumas propriedades e classes para representação de conceitos fundamentais desse modelo. Um exemplo é a propriedade *rdf:type*, usada para determinar o tipo de um recurso. Outras propriedades e classes podem (e devem) ser definidas utilizando-se os vocabulários RDFS ou OWL, criando-se desta forma ontologias para domínios específicos, que por sua vez funcionam como um contrato entre as partes que utilizam informações descritas de acordo com essas ontologias. Além disso, um dos principais benefícios da definição dessas ontologias é o fato de que sua disponibilidade permite a realização de inferências sobre dados descritos segundo as mesmas. Essas ontologias são justamente as que são utilizadas como esquema conceitual das aplicações SHDM, e empregadas para descrição dos dados das instâncias conceituais.

Em RDFS classes correspondem a recursos definidos como sendo do tipo *rdfs:Class*, e propriedades são recursos definidos como sendo do tipo *rdfs:Property*. Tanto classes quanto propriedades podem ser organizadas em hierarquias de generalização/especialização através do uso das propriedades *rdfs:subclassOf* e *rdfs:subpropertyOf* respectivamente. O exemplo abaixo mostra a declaração de duas classes, *ex:Book* e *ex:Publication*, e estabelece que a primeira é subclasse da segunda.

```
ex:Publication    rdf:type          rdfs:Class .
ex:Book           rdf:type          rdfs:Class .
ex:Book           rdfs:subclassOf   ex:Publication .
```

Quadro 4 - Exemplo de declaração de classes em RDFS

RDFS também permite a declaração dos tipos do domínio e contra-domínio das propriedades sendo definidas. Para isso são utilizadas as propriedades

rdfs:domain e *rdfs:range* respectivamente. A declaração do tipo do domínio e contra-domínio de uma propriedade é opcional. Além disso, essas declarações não são restritivas, tendo na verdade o objetivo de permitir inferências sobre os dados descritos com as propriedades definidas.

Um exemplo desse uso é o caso da propriedade *foaf:firstName*, cujo tipo do domínio é uma classe *foaf:Person*. O uso dessa propriedade na descrição do recurso *ex:JohnSmith* não exige que esse recurso seja declarado como sendo do tipo *foaf:Person.* Ao contrário, o modelo RDF define que o processamento dessas informações deve levar à conclusão de que, se *ex:JohnSmith* apresenta a propriedade *foaf:firstName* então *ex:JohnSmith* é um recurso do tipo *foaf:Person*, porque esse é o tipo do domínio de *foaf:firstName.*

Na linguagem OWL recursos que representam classes são definidos como sendo do tipo *owl:Class*. Recursos representando propriedades são definidos como sendo do tipo *owl:DatatypeProperty* ou *owl:ObjectProperty*. Esses dois tipos de propriedades são subclasses de *rdfs:Property*, e a distinção entre os dois se faz quanto ao tipo do valor que podem apresentar. *Datatype properties* só podem ter valores literais, enquanto que *Object properties* só podem ter um recurso como valor. Hierarquia de classes e de propriedades podem ser definidas da mesma forma que em RDFS, assim como os tipos do domínio e contra-domínio de propriedades. Além dessas características a linguagem apresenta um conjunto adicional de classes e propriedades para descrição de restrições e informações que permitem a realização de um maior número de inferências sobre os dados descritos. Como exemplos podemos citar:

- A possibilidade de restrição da cardinalidade de propriedades (*owl:cardinality*, *owl:minCardinality*, *owl:maxCardinality*).
- A declaração de uma propriedade como sendo o inverso de outra (*owl:inverseOf*), transitiva (*owl:TransitiveProperty*), ou simétrica (*owl:SymmetricProperty*).
- A declaração de classes disjuntas (*owl:disjointWith*).
- Identidade de recursos (*owl:sameAs*).

Em função das semelhanças entre o modelo RDF e o modelo orientado a objetos, representamos o esquema conceitual através de diagramas de classe UML

com estereótipos. Esta representação utiliza as mesmas extensões introduzidas por Lima (2003), como a adotada para a representação de hierarquia de propriedades, mas introduz algumas mudanças:

- Os nomes de classes, atributos e relacionamentos são prefixados pelo identificador do *namespace* no qual os mesmos foram definidos.
- O *namespace* de *XML Schema* é considerado padrão para declaração de tipos de atributos, dispensando assim a sua apresentação.

Entretanto as seguintes diferenças entre o modelo RDF e o modelo orientado a objetos devem ser observadas:

- Em RDF, não há garantia de distinção entre propriedades que representam atributos das propriedades que representam relacionamentos entre recursos. Isso só ocorre se o esquema conceitual for definido em OWL, o que pode não ser o caso.
- Em RDF, todas as propriedades são de escopo global, diferentemente do modelo orientado a objetos, onde o escopo de um atributo é a classe que o apresenta.
- A cardinalidade padrão das propriedades RDF é 0..n, enquanto que no modelo orientado a objetos atributos têm cardinalidade padrão 1.
- Em RDF não é obrigatória a declaração do domínio e contra-domínio de uma propriedade, enquanto que no modelo orientado a objetos todo atributo pertence a uma classe e tem um tipo bem definido, e todo relacionamento é definido em função das classes que associa.
- Em RDF não é possível a definição de relacionamentos n-ários, classes de relacionamento, ou relacionamentos de agregação e composição.

Em decorrência dessas diferenças, a representação de uma ontologia conceitual como um diagrama de classes no estilo UML deve ser encarada como um exemplo do uso esperado das propriedades e classes da ontologia e não como

uma definição precisa do esquema conceitual. A figura a seguir mostra um exemplo do que poderia ser o esquema conceitual das informações sobre um livro, como as apresentadas no Quadro 1.

Figura 3 - Exemplo de representação de um esquema conceitual como um diagrama de classes no estilo UML

2.2
Modelo navegacional

O modelo navegacional corresponde a uma visão sobre os dados do modelo conceitual, definindo quais informações poderão ser acessadas pelos usuários da aplicação e como estas informações poderão ser exploradas. Esta visão é especificada de acordo com o perfil dos usuários e das tarefas a terem suporte da aplicação, podendo para um mesmo modelo conceitual existir um modelo navegacional distinto para diferentes tipos de usuários e tarefas.

A especificação do modelo navegacional é dividida em dois esquemas, cada qual abordando aspectos diferentes do modelo. O Esquema de Classes Navegacionais está relacionado com a definição de QUAIS informações serão apresentadas, incluindo a definição de classes e relacionamentos navegacionais (elos). O Esquema de Contextos Navegacionais trata da questão de COMO será a navegação pelas informações apresentadas, incluindo a definição de contextos navegacionais e estruturas de acesso.

A definição do modelo navegacional se dá pela especificação de um mapeamento navegacional que estabelece uma relação entre os dados do modelo conceitual e o modelo navegacional. A declaração desse mapeamento é expressa em RDF utilizando-se um vocabulário específico do método. Diagramas de classes navegacionais, diagramas de contextos, tabelas de mapeamento de classes navegacionais, cartões de declaração de contextos e cartões de declaração de estruturas de acesso são utilizados como formas auxiliares mais convenientes para apresentação da informação codificada em RDF. Os detalhes relativos à declaração do mapeamento navegacional são abordados no próximo capítulo.

2.2.1
Classes navegacionais

Classes navegacionais definem quais dados do modelo conceitual poderão ser acessados pelos usuários da aplicação.

Um dos desafios na apresentação de dados de um grafo RDF é identificar quais das informações associadas a um recurso devem ser exibidas. Considere, por exemplo, o grafo mostrado na Figura 1, que contém informações sobre um recurso correspondente ao livro Don Quixote. Qual informação, nesse grafo, uma aplicação que exibe os dados de um livro deve considerar como sendo informação sobre o livro? Apenas o título, ou o título e o nome do autor, por exemplo? O método SHDM trata dessa questão por meio da abstração de classes navegacionais, permitindo tratar arranjos de recursos, propriedades e valores de propriedades, como unidades de informação mais significativas para a tarefa suportada pela aplicação. No caso do exemplo citado, pode ser definida uma classe navegacional *Book* (Livro), com atributos *title* (titulo) e *author* (autor). Dessa forma fica estabelecido explicitamente que as informações sobre um livro incluem seu título e o nome de seu autor.

Toda classe navegacional é especificada em função de uma classe conceitual, que é dita ser a sua classe base. Esta especificação é feita através da declaração de um mapeamento que estabelece quais recursos, instâncias da classe base, serão tratados como instâncias da classe navegacional, e quais são os atributos da classe navegacional, oriundos de atributos da classe base ou de classes conceituais relacionadas. Este modelo de mapeamento se reflete nas duas visões que se pode ter de uma classe navegacional. Na primeira, ela é vista como uma classe RDF, e dessa forma utilizada na classificação de recursos. Na segunda, ela aparece como uma classe de um modelo orientado a objetos, apresentando atributos com valores de tipos específicos.

O valor do atributo de uma classe navegacional pode ser: (1) um valor simples (qualquer valor dos tipos de *XML Schema*); (2) uma lista; (3) uma âncora; (4) um índice. As características e detalhes da declaração de cada um destes tipos de atributos são apresentados no próximo capítulo.

Em termos práticos, cada atributo corresponde a um padrão de recuperação de dados no grafo RDF a partir do recurso que é a instância da classe

navegacional. A combinação desses padrões forma uma máscara, que quando aplicada sobre um recurso, instância da classe navegacional, resulta na obtenção de um subgrafo do grafo RDF de instâncias conceituais. Este subgrafo representa um nó navegacional, equivalente a uma instância da classe navegacional na visão orientada a objetos. A figura a seguir exemplifica a aplicação da máscara de uma classe navegacional sobre um grafo RDF para identificação de diversos nós de uma classe.

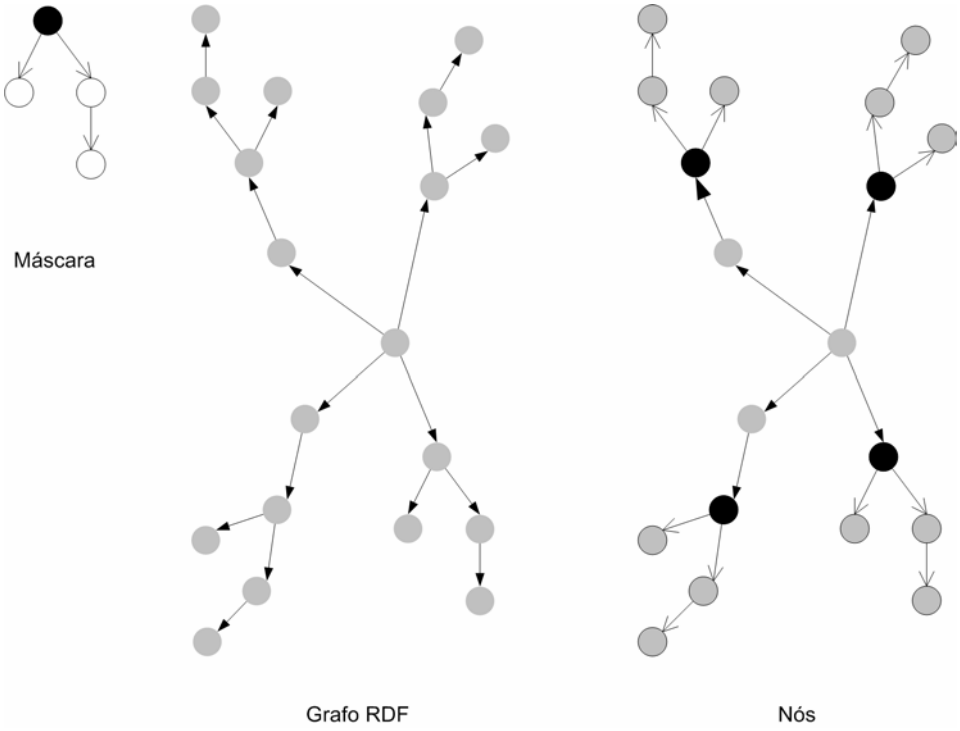

Figura 4 - Exemplo da aplicação da máscara de uma classe navegacional em um grafo RDF

As classes navegacionais também podem ser organizadas em hierarquias de generalização/especialização. A hierarquia definida deve ser uma visão da hierarquia de classes conceituais, o que quer dizer que uma classe navegacional só pode especializar uma outra classe navegacional, se a classe base da primeira também for uma especialização da classe base da segunda.

A tabela a seguir corresponde à declaração do mapeamento de uma classe navegacional *Book* sobre o esquema conceitual da Figura 3.

Navigational class: Book	
Baseclass: ex:Book	
Instances filter: -	
Subclass of: -	
Attributes	
name	**Source**
title	dc:title
author	dc:creator ; foaf:name

Tabela 3 - Exemplo de tabela de mapeamento de classe navegacional

A figura a seguir mostra a representação gráfica da classe navegacional definida pelo mapeamento da Tabela 3.

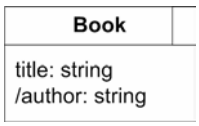

Figura 5 - Exemplo de representação gráfica da classe navegacional *Book*

Nesta figura, pode ser observada uma alteração na representação gráfica de uma classe navegacional introduzida na nova versão do método: Atributos que não são derivados diretamente de atributos da respectiva classe conceitual base têm os nomes precedidos por uma barra "/". Este é o caso do atributo *author* cujo valor é mapeado do atributo *foaf:name* da classe conceitual relacionada *foaf:Person*. Esta representação é válida tanto para atributos simples quanto para atributos mapeados como listas, âncoras ou índices. Os detalhes sobre os tipos de mapeamento de atributos são apresentados na seção *3.1.2-Definição de atributos*.

2.2.2 Classes em contexto

Uma classe em contexto funciona como um padrão decorador (Gamma et al., 1995) de uma classe navegacional, acrescentando atributos de forma que um nó dessa classe possa apresentar informações diferentes de acordo com o contexto em que está sendo exibido. Seu uso só é necessário quando se deseja que um nó tenha propriedades particulares dependendo do papel (*"role"*) desempenhado; isto

tipicamente se traduz em possuir atributos diferentes em algum contexto específico.

2.2.3 Elos

Elos são as ligações entre nós, e entre nós e estruturas de acesso, e definem os possíveis caminhos de navegação da aplicação. Os elos de uma aplicação SHDM podem ser definidos de três formas diferentes: (1) mapeamento de propriedades conceituais; (2) declaração de âncoras em estruturas de acesso ou classes navegacionais; (3) definição de padrões de navegação interna para contextos.

A definição de elos através do mapeamento de propriedades conceituais é o equivalente ao que se tinha na versão anterior do método como mapeamento de relacionamentos conceituais. A diferença básica é que neste caso o relacionamento conceitual é representado por uma propriedade RDF ligando dois recursos. Uma nova funcionalidade introduzida neste trabalho é a possibilidade de um elo ser definido pelo mapeamento de um caminho de propriedades conceituais ligando dois recursos, o que permite que na visão navegacional este caminho seja visto com um elo direto entre estes recursos. Considere um modelo conceitual com as classes *Aluno*, *Departamento*, *Centro* e *Universidade* e que defina as relações de que um aluno é formado em um departamento, que pertence a um centro, que por sua vez é parte de uma universidade. No mapeamento navegacional, o caminho de propriedades (*formado*, *pertence*, *parte*) que liga instâncias de *Aluno* e *Universidade* passando por instâncias de *Departamento* e *Centro* pode ser mapeado como um elo *estuda* ligando diretamente as instâncias de *Aluno* e *Universidade*, como mostra a Figura 6. Sob esta ótica, o tradicional mapeamento um para um entre um relacionamento conceitual e um elo pode ser tratado como o mapeamento de um caminho formado por apenas uma propriedade conceitual.

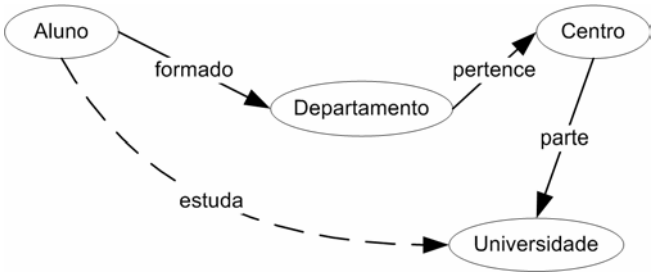

Figura 6 - Exemplo de mapeamento de um caminho de propriedades como um elo

Além da funcionalidade de um elo mapear um caminho de propriedades, também foi introduzida a opção de que este mapeamento seja feito no sentido inverso do caminho mapeado. No caso do exemplo apresentado anteriormente o mesmo caminho (*formado*, *pertence*, *parte*) poderia então ser mapeado não como um elo levando de *Aluno* para *Universidade*, mas como um elo levando de *Universidade* para *Aluno*.

Finalmente considere um carro como descrito na Figura 7(a). Pode ser interessante no modelo navegacional, ao se mapear o relacionamento *possui* especializá-lo em função das subclasses de *Parte*, como ilustrado na Figura 7(b). Este tipo de mapeamento oferece a oportunidade de refinamento da semântica da relação representada pela propriedade mapeada e é mais uma funcionalidade adicionada ao mapeamento de elos navegacionais.

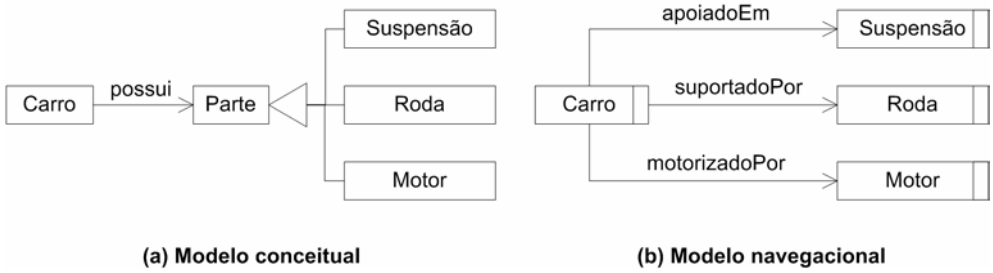

Figura 7 - Exemplo do mapeamento de elos especializando as relações representadas por propriedades conceituais

É importante observar que o mapeamento de um elo que especializa uma propriedade é diferente da definição de uma subpropriedade RDF. O conceito de subpropriedade estabelece que: Se uma propriedade *P* é subpropriedade de uma propriedade *Q*, então todos os recursos associados através de *P* também estarão associados através de *Q*. Já este tipo de mapeamento funciona no sentido inverso. No caso do elo *motorizadoPor* no exemplo da Figura 7(b), o mapeamento define

que se um recurso **carro1,** instância de *Carro,* estiver associado pela propriedade *possui* a um recurso **motor1,** instância de *Motor* – subclasse de *Parte* – (i.e., **carro1** *possui* **motor1**), então também será verdade que **carro1** será associado à **motor1** através do elo *motorizadoPor*. Mais ainda, esse mapeamento pode estar especializando um caminho de propriedades (como no exemplo da Figura 6), o que não é possível de se obter com a definição de uma sub-propriedade.

2.2.4 Contextos navegacionais

Contextos navegacionais são conjuntos de nós, e um nó sempre é manipulado dentro de um contexto específico. Um nó pode fazer parte de mais de um contexto, e um contexto pode apresentar nós de tipos diferentes, mas em geral o que se encontra são contextos com nós de um único tipo. Alguns exemplos de contextos navegacionais são: todos os professores; alunos com matrícula anterior a 2004; CDs gravados por um artista; CDs ou músicas com os dizeres "*Electric Ladyland*" no título; etc.

A definição de quais nós fazem parte de um contexto pode ser feita com base em um critério de seleção declarado na especificação do contexto, ou através da seleção de um valor em uma composição de facetas (veja a seção *2.2.6-Navegação facetada* para mais detalhes). Em ambos os casos, os nós são selecionados em função da presença de alguma característica comum, como ser de um tipo específico, ou apresentar um determinado valor para um atributo ou elo.

A definição de um contexto também incluiu a determinação de como seus nós podem ser acessados uns a partir dos outros. Isso é feito através da especificação de um padrão de navegação interna, cujo objetivo é permitir exploração de todos os elementos de um contexto, a partir de qualquer um deles. Existem quatro tipos de navegação intracontextual que podem ser aplicados diretamente, ou combinados, para definir o padrão de navegação interna de um contexto. Tais tipos são:

- **Navegação livre –** garante o acesso direto entre os nós do contexto. Em termos práticos o uso da navegação livre implica a recuperação de todos os demais nós do contexto junto com o nó selecionado,

cabendo à interface da aplicação utilizar esses dados, para dar ao usuário acesso direto aos nós.

- **Navegação circular** – os nós do contexto formam um círculo, e elos implícitos são criados entre os nós para navegação. Para cada nó são criados elos para o nó anterior e o próximo nó. A navegação em qualquer sentido levará de volta ao nó de partida após passar uma vez por todos os demais nós.
- **Navegação seqüencial** – os nós do contexto formam uma seqüência, e elos implícitos são criados entre os nós para navegação. Para cada nó são criados elos para o primeiro e o último nó da seqüência. Para cada nó, exceto o primeiro, é criado um elo para o nó anterior. Para cada nó, exceto o último, é criado um elo para o próximo nó. Partindo-se de qualquer nó, é possível navegar em um dos dois sentidos até se alcançar o primeiro ou o último nó da seqüência (dependendo do sentido seguido).
- **Navegação por índice -** é realizada com o auxílio de um índice associado ao contexto. Para cada nó do contexto é criado um elo para o índice associado. Requer a declaração do índice associado, que deve ser obrigatoriamente um índice derivado de contexto baseado no contexto em questão.

Exceto por combinações com os tipos de navegação circular e seqüencial simultaneamente, qualquer combinação de tipos de navegação pode formar o padrão de navegação interna de um contexto.

Este modelo de navegação intracontextual deixa evidente que os elementos do contexto são explorados em uma determinada ordem. Em princípio essa ordem é indeterminada, mas é possível a especificação de uma ordenação baseada nos valores dos atributos dos nós. Para um único contexto pode ser definido mais de um critério de ordenação, entretanto apenas um dos critérios definidos é utilizado na ordenação dos elementos num dado momento. O critério empregado é informado pelo usuário da aplicação. Dentre os critérios definidos para o contexto, um deles pode ser especificado como sendo o critério de ordenação padrão, utilizado quando o usuário não informar o critério a ser empregado.

2.2.5
Estruturas de acesso

Estruturas de acesso, também chamadas de índices, representam conjuntos de âncoras que apontam para outras estruturas de acesso ou nós dentro de contextos específicos. Tipicamente são empregadas para fornecer pontos iniciais de navegação (aparecendo como menus da aplicação), ou permitir o acesso a partir de um nó a diversos outros nós.

As informações apresentadas em uma estrutura de acesso são organizadas como entradas de dados, onde cada entrada contém uma série de atributos simples ou âncoras (espera-se que toda entrada tenha pelo menos uma âncora). Opcionalmente uma entrada pode apresentar uma lista de entradas subordinadas, permitindo dessa forma a construção de índices hierárquicos. De forma análoga aos contextos, as entradas de uma estrutura de acesso podem ser ordenadas com base nos valores de seus atributos.

O método SHDM permite a definição de quatro tipos de índices que se distinguem em função da origem de seus dados.

- **Índice derivado de consulta** – seus os dados são recuperados a partir de uma consulta feita diretamente sobre o grafo RDF de instâncias navegacionais. Os resultados dessa consulta são formatados segundo uma estrutura pré-definida de atributos para gerar entradas do índice. Índices desse tipo possuem apenas um nível de entradas e todas as entradas apresentam a mesma estrutura (mesmo conjunto de atributos), por isso esses índices também são classificados como índices uniformes.
- **Índice derivado de contexto** – índices desse tipo também são classificados como índices uniformes, ou seja, possuem apenas um nível de entradas e todas as entradas apresentam a mesma estrutura. A diferença para um índice derivado de consulta, é que aqui os valores usados para gerar as entradas são os dados de um contexto, e não os resultados de uma consulta. Cada entrada nesse índice leva a um elemento do contexto correspondente. Equivale a definição da visão de um contexto como um índice para seus elementos.

- **Índice arbitrário** – nesse tipo de índice todas as entradas são declaradas explicitamente na especificação do índice, podendo ser organizadas em uma estrutura hierárquica e apresentar atributos distintos. Além disso, cada entrada deve ter uma identificação única.
- **Índice facetado** – este tipo de índice é baseado em uma composição de facetas (veja seção *2.2.6-Navegação facetada*) e suas entradas são definidas em função dos valores selecionados nessa composição de facetas.

2.2.6 Navegação facetada

A exploração de uma coleção de dados por navegação é caracterizada tipicamente pela aplicação sucessiva de critérios de seleção, que por sua vez correspondem a particionamentos sucessivos da coleção em questão. Considere o exemplo de uma aplicação que implementa um catálogo de obras de arte, na qual para localizar uma obra o usuário escolhe o país, em seguida o período, depois o tipo da obra e finalmente uma obra específica. A Figura 8 apresenta o exemplo de uma navegação nessa aplicação, e a Figura 9 mostra o reflexo dessa navegação no particionamento sucessivo do conjunto de obras de arte.

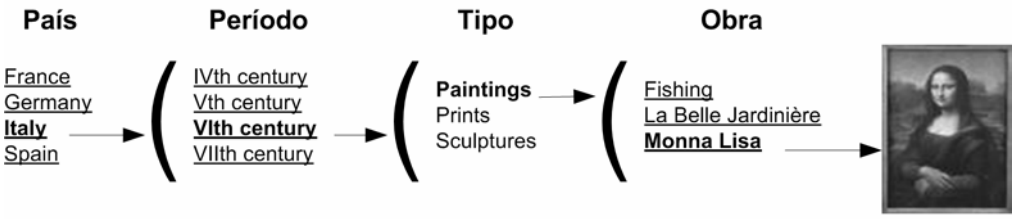

Figura 8 - Exemplo de navegação em busca de uma obra de arte

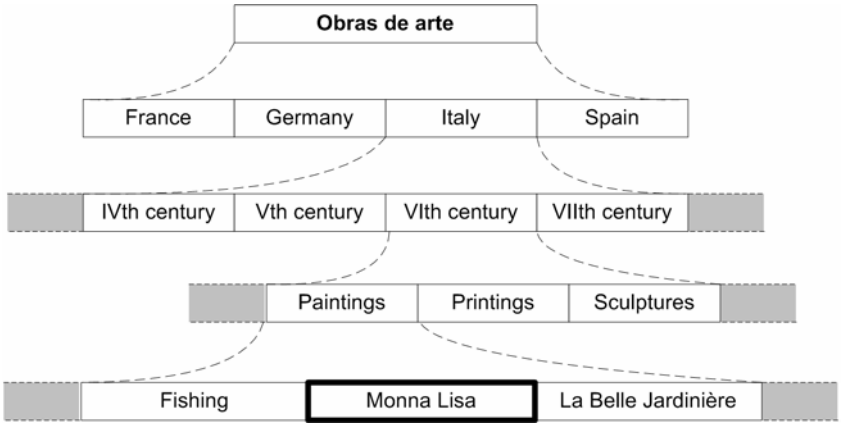

Figura 9 - Exemplo do particionamento do conjunto de obras de arte

Hearst et al. (2002) e Yee et al. (2003) defendem as vantagens desta forma de busca por uma informação através da exploração por navegação quando comparada, por exemplo, com alternativas como busca por palavra-chave, principalmente quando o conjunto de informações exploradas é vasto e a princípio tem-se apenas uma vaga noção daquilo que é buscado. E justamente esta "vaga noção" requer que a aplicação permita a escolha de diferentes caminhos para exploração, pois cada usuário pode empregar um raciocínio diferente na sua busca. No exemplo anterior o caminho usado foi *País>Período>Tipo de obra*, mas para alguns usuários poderia ser mais natural procurar por *Artista>Tipo de obra*, ou *Tipo de obra>País>Período*.

Apesar do uso das abstrações de estruturas de acesso e contextos navegacionais permitir a modelagem desses diversos caminhos, Lima (2003) mostrou como essa abordagem resultaria em modelos navegacionais muito complexos e introduziu as abstrações de estruturas de acesso e contextos facetados, demonstrando como estas permitem a modelagem de diversos caminhos de exploração em uma aplicação de forma concisa e flexível.

O modelo de navegação facetada do método SHDM se baseia na definição de contextos e estruturas de acesso com base em um grupo de facetas.

Uma faceta corresponde a um critério de classificação do universo de objetos sendo organizado, e é definida em função das características desses objetos. A seleção de um valor para uma faceta resulta na seleção do conjunto de objetos que apresentam o respectivo valor para a característica associada à faceta. No caso do método SHDM toda faceta é definida em função de um atributo

simples de uma classe navegacional, ou de um elo partindo desta classe para outra classe navegacional.

Um grupo de facetas reúne facetas definidas para uma mesma classe de objetos, e o contexto baseado nesse grupo apresenta apenas instâncias dessa classe. A definição dos elementos do contexto é feita com base em uma seleção dinâmica composicional, pois é o resultado da combinação, em qualquer ordem, de sucessivas seleções de valores nas facetas do grupo feitas pelo usuário durante a execução da aplicação.

Aplicando este modelo ao exemplo anterior de busca por obras de artes, um grupo de facetas é especificado, contendo facetas definidas sobre os valores do período de criação, o país de origem, o autor e o tipo de uma obra. Finalmente com base neste grupo, um contexto e um índice facetado são especificados.

Os possíveis valores de uma faceta podem ser recuperados dos valores existentes nas instâncias dos objetos classificados ou declarados explicitamente, podendo até ser organizados em uma hierarquia arbitrária. Para o caso da faceta de país de origem de uma obra de arte, por exemplo, pode ser criada uma hierarquia, que num primeiro nível apresenta continentes e num segundo nível apresenta países. Nesse caso, a seleção de um continente corresponde à seleção de todas as obras de arte cujo país de origem é um dos países que estiverem subordinados ao continente selecionado. A figura a seguir mostra um exemplo de como os valores dessa faceta seriam exibidos em uma aplicação, com e sem a definição de uma hierarquia.

Figura 10 - Exemplo de organização hierárquica dos valores de uma faceta de obras de artes pelo país de origem

É importante observar que nesse exemplo uma obra de arte apresenta apenas a informação do país de origem, mas a declaração da hierarquia sobre os valores desse atributo permite um refinamento adicional do modelo navegacional.

O método SHDM também permite a definição de uma faceta em função da hierarquia de subclasses da classe para a qual a faceta é definida. Este recurso permite a criação de índices de hierarquia de classes.

No caso geral, entretanto, algumas das combinações de valores das facetas de um grupo podem ser inválidas, não sendo aplicáveis a nenhum nó que possa existir na aplicação. Para evitar que estes casos resultem em navegações que levam o usuário a conjuntos vazios, as combinações inválidas podem ser declaradas explicitamente na definição de um grupo de faceta de forma que não sejam apresentadas em momento algum para o usuário.

2.2.7 Landmarks

Quando uma estrutura de acesso ou um nó em um contexto podem ser acessados diretamente de qualquer ponto da aplicação, o método SHDM permite a definição de um *landmark*, que é um elo o para elemento em questão. O uso de um *landmark*, que é declarado com uma âncora, simplifica não apenas a apresentação do diagrama de contextos, como também as declarações de classes navegacionais e estruturas de acesso, que caso contrário teriam todas que incluir a definição de uma mesma âncora apontando para o elemento acessado através do *landmark*. Com o *landmark*, essa âncora é definida uma única vez para todo o modelo navegacional.

2.3 Operações

Tanto em OOHM quanto em SHDM, operações de navegação são intrínsecas ao modelo da aplicação e operações específicas do domínio da aplicação são modeladas explicitamente.

Como mencionado anteriormente, em OOHDM as operações específicas do domínio da aplicação eram modeladas como métodos de classes, e este aspecto não foi abordado na primeira versão do SHDM. Esta abordagem foi abandonada

no presente trabalho. Essa decisão foi motivada, primeiramente, pelo fato de RDF ser apenas um modelo para descrição de dados, e sua adoção como a base estrutural do modelo conceitual SHDM não permite mais a definição de métodos de classes, uma vez que esse conceito não existe em RDF.

Além disso, constatou-se que a modelagem das operações de aplicações hipermídia pode ser uma questão bem mais complexa do que o tratamento dispensado pelo método até então assumia. Existem, por exemplo, operações que atuam sobre dados do modelo conceitual e outras que atuam sobre dados do modelo navegacional, operações que são realizadas por outros sistemas, tendo efeito sobre a aplicação de forma indireta, e ainda aquelas que fazem parte ou desencadeiam processos que também resultam em navegação. Em todos esses casos, a abordagem de modelar as operações como métodos de classes do modelo conceitual ou navegacional mostrou-se insuficiente.

Tais questões indicaram a necessidade da definição de modelo para especificação dos processos e operações das aplicações hipermídia. Entretanto, a definição deste modelo e como ele se enquadra no fluxo de desenvolvimento do método SHDM são questões que estão fora do escopo desta dissertação.

Mesmo assim, um primeiro passo nesse sentido já foi dado com a definição de um vocabulário para declaração de assinaturas de operações. Dessa forma, como fazia antes no OOHDM, é possível definir quais são as operações da aplicação. A diferença é que ao invés de especificar essas operações como métodos de uma classe, elas serão definidas como entidades independentes. E até que o modelo para especificação de processos e operações seja definido, foi estabelecido que a especificação das assinaturas das operações será feita em conjunto à especificação do mapeamento navegacional.