3
O método SHDM

Proposto originalmente por [Lima, 2003] e evoluído em vários trabalhos posteriores, o método SHDM (*Semantic Hypermedia Design Method*) utiliza uma abordagem baseada em modelos para projetar aplicações hipermídia, considerando os formalismos e primitivas introduzidos pela Web Semântica, em um processo iterativo e incremental de seis etapas: Levantamento de Requisitos, Modelagem do Domínio, Projeto Navegacional, Projeto de Interface, Projeto Comportamental e Implementação.

Neste trabalho, apresentamos algumas mudanças no método SHDM motivadas pelas práticas *Linked Data* [Bizer et al, 2009] e práticas recomendadas pelo projeto *Linking Open Data* (LOD), com o objetivo de torná-lo mais adequado ao desenvolvimento de aplicações hipermídia na Web Semântica.

O método SHDM foi extensamente discutido em trabalhos anteriores, principalmente em [Lima, 2003], [Szundy, 2004] e [Nunes, 2005]. Por isso, será apresentado aqui de forma resumida com destaque para as mudanças introduzidas neste trabalho.

3.1.
Etapas

Cada etapa do método SHDM produz um ou mais modelos que focam em aspectos distintos de uma aplicação hipermídia. Pelo princípio da separação de interesses ("*separation of concerns"*) [DIJKSTRA, 1982], o projetista deve modelar cada aspecto da aplicação separadamente em uma das etapas do método.

Sendo assim, temos a etapa de Levantamento de Requisitos, com ênfase na identificação dos atores e tarefas a serem apoiadas pela aplicação; Modelagem de Domínio, com ênfase nos aspectos do domínio da aplicação, seus dados, meta dados e relacionamentos; Projeto Navegacional, com ênfase nos aspectos de navegação sobre os dados da aplicação; Projeto Comportamental, com ênfase nas regras de negócios e apoio à interação do usuário com a aplicação; e finalmente,

Projeto de Interface, com ênfase nos aspectos visuais e de interação do usuário com a aplicação.

Apesar de apresentadas de maneira sequencial, as seis etapas do método não precisam ser executadas em uma ordem específica e apenas uma única vez, como em um processo em cascata. Estas etapas podem ser executadas de maneira iterativa, incremental e cíclica.

A Tabela 2 é uma atualização da tabela de artefatos gerados em cada etapa do método SHDM, apresentada originalmente por [Szundy, 2004], com destaque para as mudanças recentes.

	Artefato	Descrição	Etapa
1	Descrição dos 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	Ontologias de domínio	Vocabulários para definição das instâncias de domínio (recursos RDF). Pode ser qualquer ontologia da Web Semântica definida em OWL ou RDFS.	Modelagem de Domínio
4	*Namespaces*	Conjunto de *namespaces* (pares de prefixos e URIs) utilizados pela aplicação.	Modelagem de Domínio
5	Repositórios	Repositórios de dados da Web Semântica, geralmente acessados via SPARQL *Endpoints*. A aplicação usa os dados disponíveis nestes repositórios como instâncias do domínio.	Modelagem de Domínio
6	Instâncias do domínio	Dados do domínio da aplicação, definidos segundo a ontologia de domínio. Mais especificamente, recursos RDF.	Modelagem de Domínio
7	Mapeamento navegacional	~~Especificação dos mapeamentos de classes navegacionais e elos dos espaço navegacional~~ Descrição dos atributos navegacionais nas classes de domínio Esquemas de contextos, estruturas de acesso e *landmarks*.	Projeto Navegacional

		Definidos com um vocabulário específico do método.	
8	Modelo Navegacional	~~Definição do esquema de instâncias de classes navegacionais.~~ As instâncias do modelo navegacional são recursos RDF enriquecidos com os atributos navegacionais. ~~Tipicamente devem ser gerados dinamicamente a partir do modelo de domínio 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
9	Modelo de operações	Definição das operações da aplicação que dão semântica as regras de negócio e apoio as tarefas de interação com o usuário. Definidas com um vocabulário específico do método.	Projeto Comportamental
10	Modelo de Interfaces	Definição de elementos da interface abstrata, descrições retóricas e seus mapeamentos para o modelo de operações, incluindo o navegacional, e para componentes da interface concreta. Definida com um vocabulário específico do método	Projeto de Interface
11	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

Tabela 2 – Artefatos do método SHDM

Na tabela acima, os textos sublinhados indicam novos artefatos ou novas inserções nas descrições. Os textos riscados foram deixados propositalmente para indicar elementos que foram removidos. Algumas etapas ou modelos tiveram seu

nome alterado. Cada uma das mudanças indicadas na tabela será vista em detalhes nas seções adiante.

Embora cada artefato seja produzido em uma etapa distinta, estes artefatos podem sofrer mudanças estimuladas por requisitos identificados em outras etapas, que não a sua etapa original.

Por exemplo, uma ontologia de domínio, produzida na etapa de Modelagem de Domínio, pode sofrer uma mudança estimulada pela etapa de Projeto Navegacional, se uma propriedade de domínio é criada na ontologia exclusivamente para dar apoio a uma tarefa identificada durante a etapa de Projeto Navegacional. A primeira vista, pode parecer que esta mudança ocorreu em uma nova iteração da etapa de Modelagem de Domínio. Porém, não deve-se perder o foco do interesse ("*concern*") que motivou tal mudança. No caso, o interesse tem relação com a etapa de Modelagem Navegacional. Neste caso, consideramos que a etapa de Projeto navegacional provocou mudanças no modelo de domínio.

Isso é verdade para todos os outros modelos e etapas do método. Uma vez que todos os modelos são descritos em RDF, é possível que qualquer um deles sofra alterações pela inclusão de novas triplas, motivadas por etapas diferentes de suas etapas originais.

3.2. Levantamento de Requisitos

A etapa de levantamento de requisitos manteve-se conforme a proposta original de [Vilain, 2001], tendo como objetivo principal a identificação dos atores e tarefas a serem apoiadas pela aplicação. Para isso, devem ser produzidas descrições dos cenários e casos de uso, que são apresentadas através de narrativas em formato textual. Além desses, devem ser produzidos ainda os diagramas de interação com o usuário (*User Interaction Diagrams* - UIDs) que servirão de base para uma validação visual do comportamento da aplicação. A Figura 5 apresenta um exemplo deste tipo de diagrama.

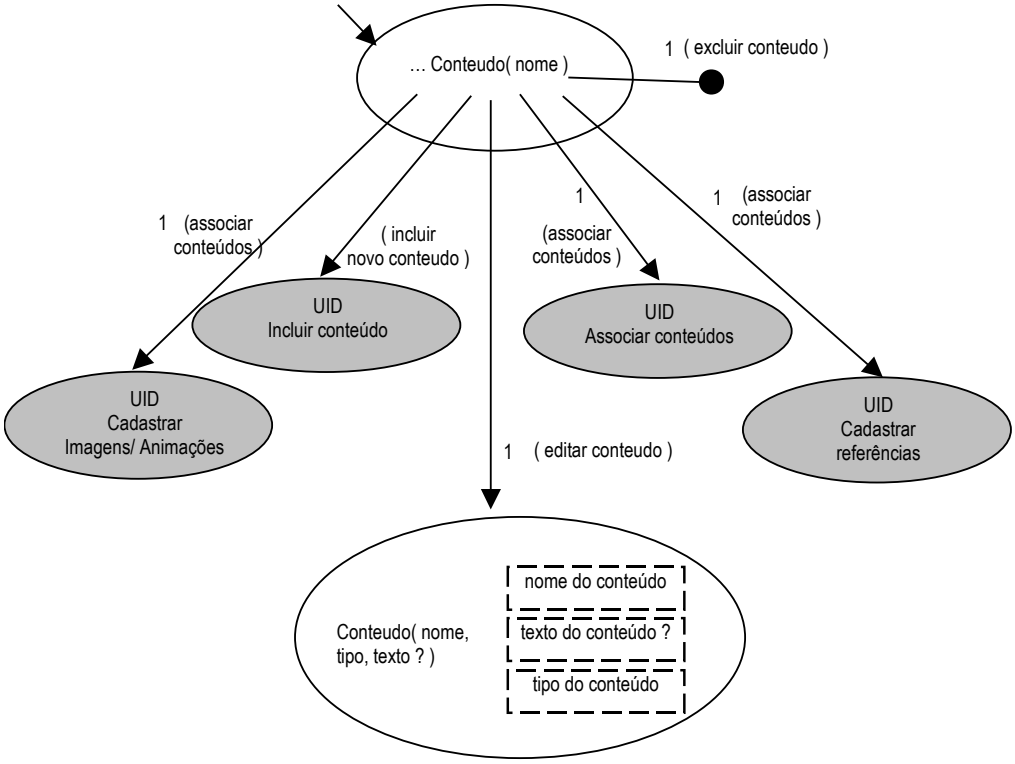


Figura 5 - Exemplo de UID como apresentado por [Leite 01]

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

3.3. Modelagem de Domínio

Na etapa de Modelagem de Domínio são descritos todos os itens de informação do domínio da aplicação e os seus relacionamentos, resultando em um modelo de domínio. Mais especificamente, emprega-se diretamente o modelo RDF como base estrutural do modelo de domínio e, dessa maneira, é possível que se utilize qualquer ontologia definida para a Web Semântica (em OWL ou RDFS) como esquema conceitual da aplicação, sem a necessidade de adaptações [Szundy, 2004].

Esta etapa, que originalmente era chamada de Modelagem Conceitual, passou a ser chamada de Modelagem de Domínio porque, conforme demonstrado por [Santos, 2010, p. 47], descreve os dados reais do domínio de uma aplicação compostos por recursos, classes e propriedades RDF, representando completamente sua semântica.

O modelo de domínio é composto, primeiramente por um conjunto de ontologias de domínio, que descrevem classes e propriedades RDF e seus

relacionamentos, e instâncias de recursos RDF descritos de acordo com essas ontologias.

No presente trabalho, apresentamos algumas recomendações para a criação e reutilização das ontologias de domínio em conformidade com as práticas *Linked Data*. Além disso, foram introduzidos como primitivas do modelo de domínio os *namespaces* e os repositórios de dados da Web Semântica ou *datasets*, devido ao entendimento de que são componentes importantes do domínio de aplicações na Web Semântica.

3.3.1. Projeto de Ontologias de Domínio

> Ontologias de domínio capturam o conhecimento sobre um domínio em particular, por exemplo, o conhecimento sobre área farmacêutica ou sobre gráficas. Essas ontologias fornecem uma descrição detalhada dos conceitos de um domínio restrito.
>
> Antoniou & van Harmelen, 2008

> Uma ontologia de domínio (*domain ontology* ou *domain-specific ontology*) modela um domínio específico, ou parte do mundo. Ela representa os significados dos termos aplicados ao domínio em questão. Por exemplo, a palavra "carta" pode ter distintos significados. Uma ontologia sobre o domínio do jogo de pôquer poderia modelar seu significado como uma "carta de um jogo", enquanto uma ontologia de um serviço postal poderia modelar seu significado como um "texto a ser entregue a um destinatário".
>
> Wikipedia

O Projeto de Ontologias de Domínio é a parte da etapa de Modelagem de Domínio que tem como objetivo a seleção, extensão e criação, nesta ordem, das ontologias de domínio que descrevem os itens de informação do domínio da aplicação.

Diferentemente da modelagem clássica orientada a objetos, onde não há a preocupação com a reutilização de dados e meta dados, na modelagem de aplicações para a Web Semântica a reutilização é vista como uma boa prática e é encorajada. Dentre as recomendações das práticas *Linked Data*, está a reutilização de termos presentes nos vocabulários bem conhecidos pela comunidade da Web Semântica [Bizer et al., 2008].

Existem várias metodologias para a construção de ontologias, entre as quais podemos citar Cyc[47], On-To-Knowledge[48] e METHONTOLOGY[49]. Todas apresentadas em detalhes por [Gomez-Perez et al., 2003] que pode ser utilizado como referência para o assunto, uma vez que não está no escopo deste trabalho a discussão sobre cada uma das metodologias disponíveis para a construção de ontologias.

Embora o método SHDM não recomende uma metodologia específica para o projeto de uma ontologia, deixando o projetista livre para escolher a metodologia que preferir ou não usar nenhuma, um conjunto de passos guias para a construção das ontologias de domínio da aplicação, com ênfase na reutilização, passou a ser recomendado a partir deste trabalho. A lista a seguir descreve esses passos:

1. A partir dos atores e tarefas identificados na etapa de Levantamento de requisitos, identificar as entidades e relacionamentos do domínio da aplicação. Neste passo, pode ser feito o uso de um esquema conceitual como um diagrama de classes no estilo UML, conforme a proposta original para modelagem conceitual do SHDM [Lima, 2003]. Porém, a produção deste esquema não é obrigatória e deve ser encarada como uma ferramenta de auxílio no entendimento do domínio, com a consciência das suas diferenças em relação ao modelo RDF;
2. Procurar por ontologias RDFS ou OWL bem conhecidas da comunidade e selecionar aquelas mais adequadas ao domínio de aplicação identificado;
3. Nos casos em que as ontologias encontradas atendam parcialmente aos requisitos identificados, estender essas ontologias com classes, propriedades e relacionamentos próprios;
4. Quando não forem encontradas ontologias adequadas ao domínio identificado, mesmo que parcialmente, projetar uma nova ontologia com suas próprias classes, propriedades e relacionamentos de acordo com o que foi identificado no passo 1. Antes de criar um novo termo, verificar a existência de termos isolados em outras ontologias, que possam

[47] http://techwiki.openstructs.org/index.php/Cyc_Mapping_Methodology
[48] http://semanticweb.org/wiki/OTK_methodology
[49] http://semanticweb.org/wiki/METHONTOLOGY

participar da ontologia própria. É uma boa prática que nova a ontologia seja projetada com a visão de que será reutilizada pela comunidade.

A Figura 6 ilustra uma ontologia de domínio, gerada a partir de um esquema conceitual de classes no estilo UML, que utiliza termos de ontologias conhecidas da comunidade da Web Semântica e termos próprios.

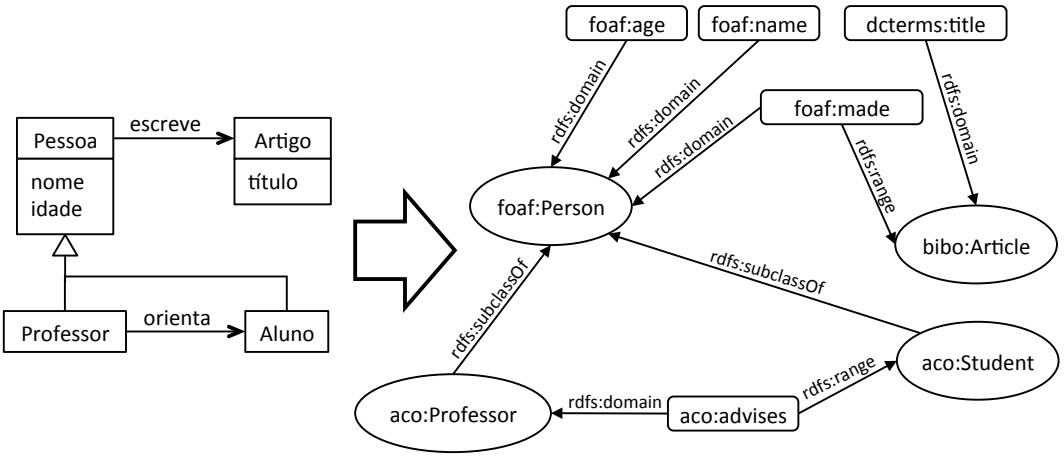


Figura 6 – Ontologia de domínio gerada a partir de um esquema conceitual de classes

Para a construção da ontologia ilustrada na figura acima, foram identificados os conceitos "Pessoa", "Professor", "Aluno" e "Artigo" e os relacionamentos "cria" e "orienta". A lista a seguir detalha o mapeamento de cada conceito para as ontologias de domínio usadas pela aplicação:

- Do vocabulário FOAF (*Friend of a Friend*)[50], utilizado para descrever pessoas e seus relacionamentos, foi usada a classe foaf:Person para o conceito "Pessoa" e as propriedades foaf:name, foaf:age e foaf:made respectivamente para "nome", "idade" e "escreve";
- Da ontologia BIBO (*Bibliographic Ontology*)[51], para descrição de produção bibliográfica, foi usada a classe bibo:Article para o conceito "Artigo";
- Para o atributo "título" do conceito "Artigo" foi usada a propriedade dcterms:title do vocabulário Dublin Core[52], para domínios de propósito geral;

[50] http://www.foaf-project.org
[51] http://bibliontology.com
[52] http://www.dublincore.org

- Os conceitos “Professor” e “Aluno” ilustram os casos em que não foram encontrados termos adequados nas ontologias disponíveis. Neste caso, foi criada a ontologia própria chamada “Academic Ontology”. Os conceitos “Professor” e “Aluno” foram mapeados para as classes aco:Professor e aco:Student e o relacionamento “orienta” para a propriedade aco:advises. Foram usados termos em inglês seguindo a visão de que esses termos poderiam ser reutilizados pela comunidade.

Ainda no Projeto de Ontologias de Domínio são selecionados os *namespaces* que serão utilizados pela aplicação. Quando se reutiliza ontologias conhecidas da comunidade da Web Semântica, alguns *namespaces* passam a fazer parte do modelo de domínio naturalmente, sem intervenção direta do projetista, uma vez que estas ontologias geralmente especificam os seus *namespaces* explicitamente. É importante que o projetista esteja consciente dos *namespaces* que fazem parte das ontologias obtida de terceiros.

Para criar novos *namespaces*, a única prática recomendada é que se evite criar prefixos já utilizados pela comunidade para não gerar conflitos de *namespaces*, quando um mesmo prefixo está relacionado a mais de um URI. É importante ressaltar que isto não é uma restrição, mas apenas uma recomendação. O serviço http://prefix.cc auxilia o projetista na procura por prefixos já utilizados e também na resolução de conflitos de *namespaces* por meio de votação.

3.3.2. Seleção de repositórios de dados (*Datasets)*

Devido ao projeto *Linking Open Data*, uma grande quantidade de repositórios de dados, em vários domínios de informação está disponível para uso. O projeto DBPedia, disponibiliza grande parte das informações contidas na Wikipedia nos formatos da Web Semântica. Da mesma forma, outras bases de dados importantes estão disponíveis, dentre as quais podemos citar: Geonames com informações geográficas, MuzicBrainz com dados sobre músicas, artistas,

álbums etc, LinkedMDB sobre filmes, atores, diretores etc, e várias outras que participam da nuvem *LOD* (*"LOD cloud"*)[53].

É natural que as aplicações na Web Semântica façam uso dos dados disponíveis nestes repositórios. Por isso, a partir deste trabalho, o SHDM passa a apoiar a utilização destes dados, como parte complementar das instâncias do domínio da aplicação. Dessa maneira, além de ser possível a construção de uma aplicação hipermídia cujo modelo de domínio combine dados locais e dados remotos, o cenários passam a incluir uma nova classe da aplicações, que são aquelas cujo objetivo exclusivo é fornecer navegabilidade às bases de dados já existentes na Web Semântica. Por exemplo, é possível construir uma aplicação hipermídia com uma navegação particular (p.ex., sobre carros esportivos) utilizando os dados da base DBPedia, diferente da navegação padrão sobre os mesmos dados oferecida pela Wikipedia.

A partir do presente trabalho, uma nova tarefa do projetista de aplicações com o SHDM é selecionar os repositórios de dados que deseja utilizar para compor as instâncias do domínio da aplicação. Não há um método específico para a seleção desses repositórios. Normalmente eles são escolhidos em função das ontologias usadas para descrever os dados mantidos, que devem ser as mesmas selecionadas para a aplicação, e de sua disponibilidade.

Uma vez selecionados os repositórios de dados, deve-se manter as descrições sobre estes repositórios de acordo com o vocabulário voiD[54]. O Quadro 6 é um exemplo de descrição do repositório DBPedia com o vocabulário voiD na notação RDF Turtle.

```
:DBPedia a void:Dataset ;
       dcterms:title "DBPedia" ;
       dcterms:description "RDF data extracted from Wikipedia" ;
       foaf:homepage <http://dbpedia.org/> ;
       void:sparqlEndpoint <http://dbpedia.org/sparql> .
```

Quadro 6 – Exemplo de descrição de repositório de dados com voiD na notação Turtle

Essa descrição deve conter pelo menos um endereço de acesso aos dados do repositório, que geralmente é o endereço de um SPARQL *Endpoint*. Os identificadores dos repositórios selecionados serão utilizados posteriormente nos

[53] http://richard.cyganiak.de/2007/10/lod
[54] http://vocab.deri.ie/void/guide

esquemas de contextos, para indicar que os recursos nos quais se baseiam os nós do contexto, devem ser obtidos dos repositórios indicados. Um exemplo do uso de repositórios de dados em um esquema de contexto será visto mais adiante na seção 3.4.

Na etapa de implementação, a aplicação deve ser implementada de maneira a combinar os dados provenientes dos repositórios da Web Semântica com os dados locais a partir das definições dos esquemas de contextos.

3.4. Projeto Navegacional

O objetivo da etapa de Projeto Navegacional é produzir o modelo navegacional da aplicação.

> 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.
>
> [Szundy, 2004]

Em aplicações hipermídia, a existência de visões navegacionais sobre os dados de um mesmo modelo de domínio se justifica porque esses dados podem ser acessados de diversas formas diferentes para apoiar tarefas e usuários distintos.

> Consideremos, por exemplo, um sistema de software baseado em hipermídia para um departamento acadêmico. Este sistema incluiria: informações gerais sobre o departamento, professores, cursos, projetos de pesquisa, orçamentos, informações sobre financiamentos, publicações, etc., e deve ser composto por várias aplicações, suportando diversos tipos de usuários, cada qual com diferentes necessidades de informação. Visitantes, estudantes e instituições de financiamento são alguns tipos possíveis de usuários. É claro que os visitantes que acessarem informações sobre projetos e cursos entenderão o domínio de forma distinta de um gerente de uma instituição de financiamento que deseje investigar projetos de pesquisa para decidir sobre a concessão de apoio financeiro ao departamento.
> O modelo conceitual funcionará como um repositório compartilhado de modelagem, a partir do qual construiremos diferentes visões (na verdade, visões navegacionais) do domínio do problema; cada uma delas constituirá um tipo distinto de aplicação hipermídia.
>
> [Rossi, 1996]

Embora o modelo navegacional tenha mantido a sua essência, uma visão navegacional sobre os dados do modelo de domínio, neste trabalho apresentamos algumas mudanças que o tornaram mais simples e consistente com o modelo de dados RDF.

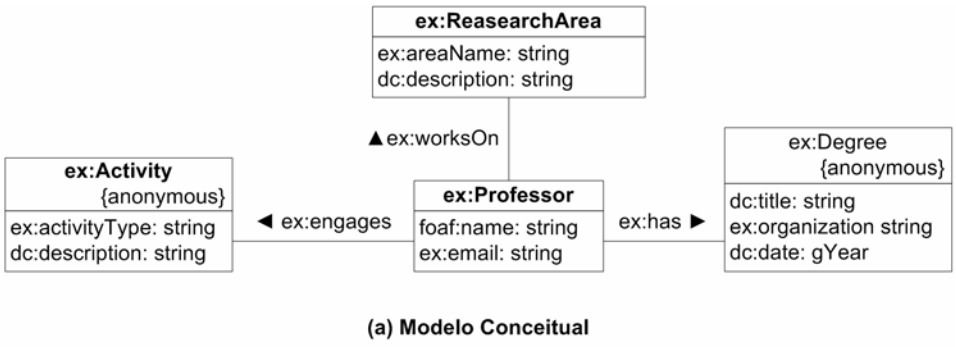
3.4.1. Mudanças no modelo navegacional

Até o presente trabalho, uma das tarefas do Projeto Navegacional era produzir um mapeamento das classes navegacionais sobre as classes do modelo de domínio, a partir do qual é gerado um novo grafo de instâncias de dados navegacionais, distinto do grafo de instâncias de dados de domínio original. Este novo grafo apresenta novas instâncias de dados com novos atributos e elos com o objetivo de atender os requisitos das tarefas de navegação. A Tabela 3 apresenta os mapeamentos produzidos nas tarefas de mapeamento de classes navegacionais:

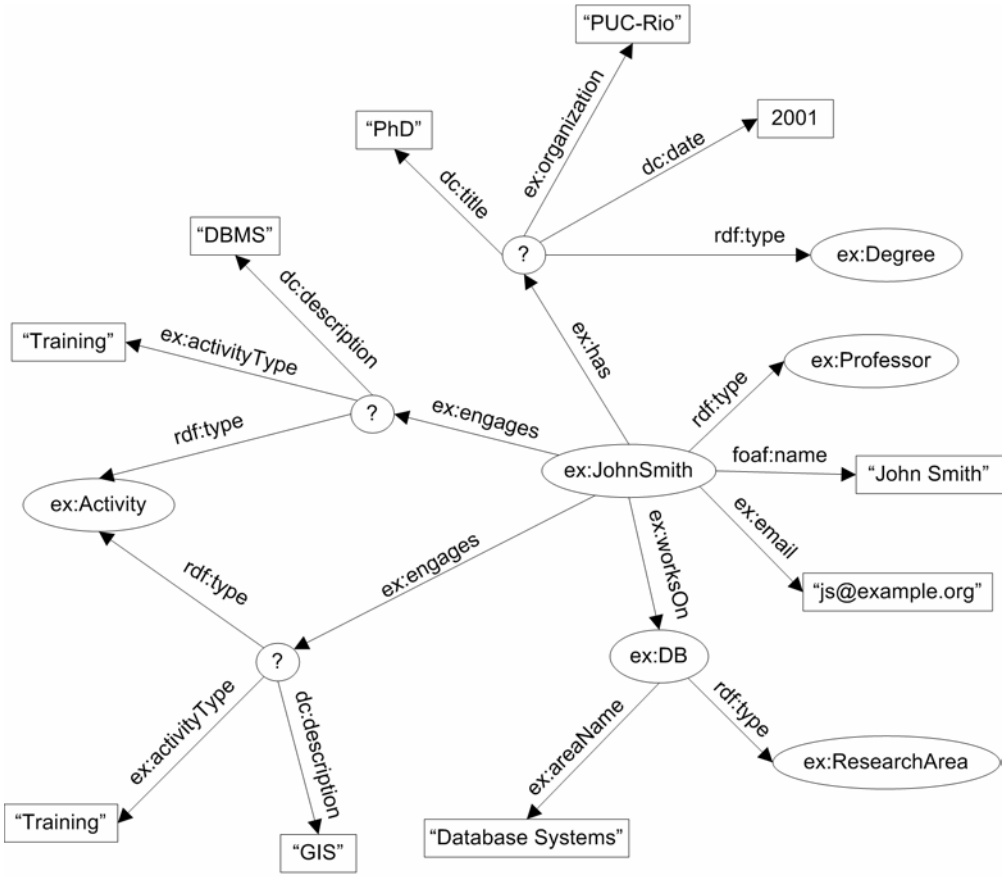
	Descrição
1.	O mapeamento direto de uma classe base de domínio em uma classe navegacional, significando que o grafo de dados navegacionais seria formado pelas instâncias das classes navegacionais, construídas a partir das classes base de domínio;
2.	Filtros no mapeamento entre uma classe navegacional e sua classe base. Por exemplo, no mapeamento de uma classe base de domínio “Pessoa” na classe navegacional “MenorDeIdade”, adicionar o filtro “< 18” na propriedade “idade”, para retornar somente instâncias de “Pessoa” com o valor da propriedade “idade” menor que o inteiro 18;
3.	O mapeamento direto de atributos das classes de domínio em atributos simples das instâncias navegacionais;
4.	O mapeamento dos relacionamentos entre as classes de domínio em elos navegacionais entre as classes navegacionais. Por exemplo, criar o elo navegacional inverso “filho” a partir do relacionamento de domínio “pai”;
5.	O mapeamento de propriedades das classes de domínio e elos navegacionais em atributos navegacionais (âncoras e índices).

Tabela 3 – Mapeamentos produzidos no mapeamento de classes navegacionais

As figuras abaixo, apresentadas originalmente em [Szundy, 2004], ilustram como era a construção do grafo de instâncias navegacionais a partir de um grafo RDF de domínio. A Figura 8(b) apresenta o grafo de instâncias navegacionais resultante do mapeamento das instâncias de domínio (antigas instâncias conceituais) da Figura 7(b). A Figura 8(a) representa o esquema de classes navegacionais especificado sobre o esquema conceitual da Figura 7 (a) segundo o qual as instâncias conceituais foram definidas.



(a) Modelo Conceitual



(b) Instâncias Conceituais

Figura 7 – Exemplo de um grafo RDF de instâncias de domínio

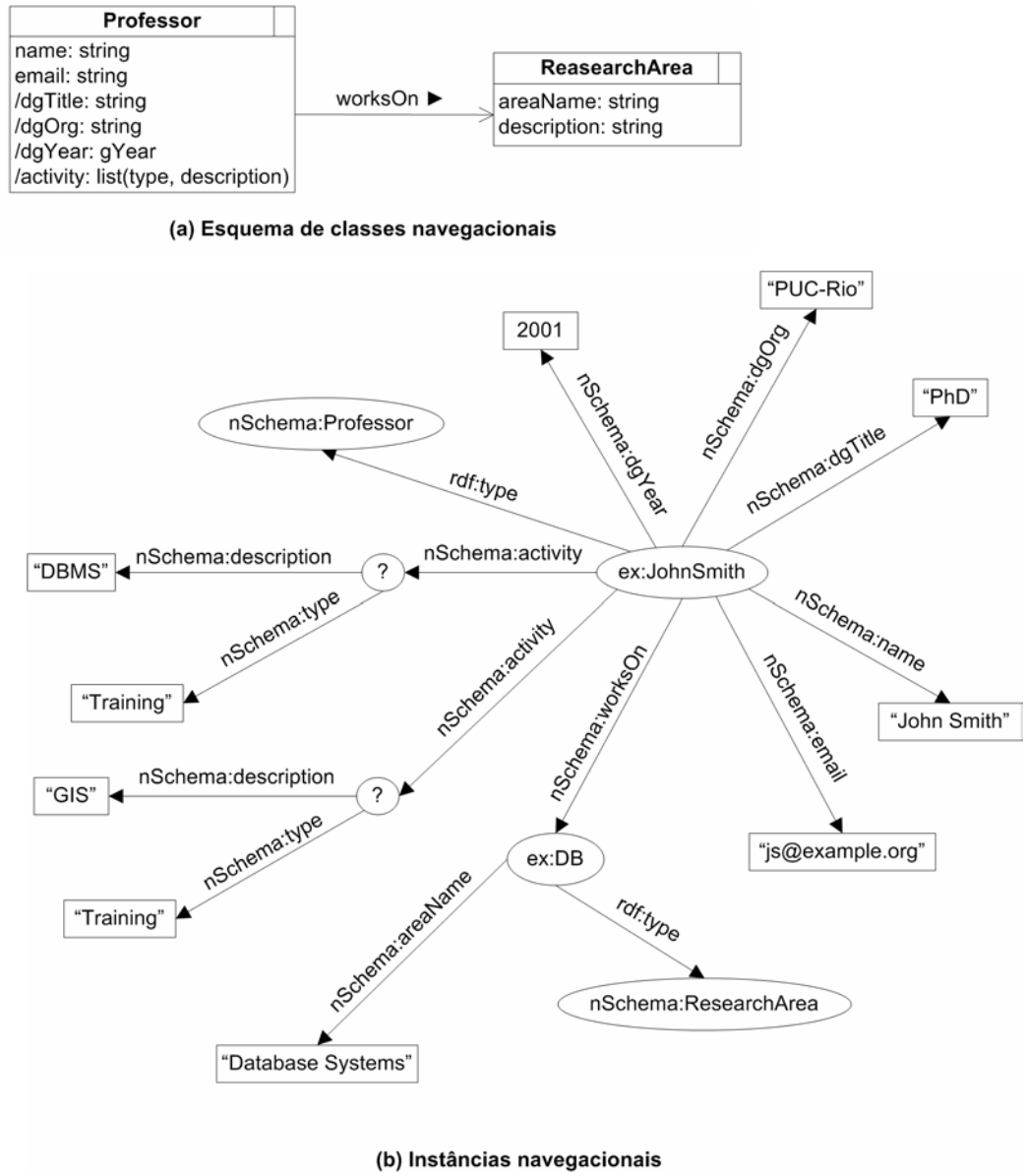


Figura 8 – Exemplo de um grafo RDF de instâncias navegacionais

Como é possível notar nas figuras acima, as transformações sobre as instâncias do domínio geraram um novo grafo fazendo com que essas instâncias perdessem muitas de suas características originais. Ainda que o objetivo original de gerar uma nova visão sobre as instâncias de domínio tenha sido alcançado com sucesso, é possível atingir o mesmo objetivo com menos transformações no grafo original devido as características do modelo de dados RDF.

Como o modelo RDF é um modelo de dados em grafo, é muito simples adicionar novas informações sobre um nó do grafo criando elos entre este nó e outros nós do grafo, sem se preocupar com restrições de esquema, comuns nos modelos de dados relacional e orientado a objetos. Especificamente para o modelo

RDF, isso significa adicionar novas triplas no grafo RDF com o recurso em questão na posição do sujeito das triplas. O mesmo é verdade para os meta dados. Como uma classe RDFS/OWL também é um recurso RDF, adicionar informações sobre uma classe é o mesmo que adicionar informações a qualquer recurso RDF. A seguir será mostrado como o modelo navegacional foi simplificado pela aplicação das propriedades intrínsecas do modelo RDF.

A partir deste trabalho, as tarefas de mapeamentos de classes navegacionais, descritas na Tabela 3, foram removidas da etapa de Projeto Navegacional. Essa mudança se deu pelo entendimento de que é mais natural, no modelo RDF, enriquecer as instâncias do modelo de domínio com atributos navegacionais, mantendo o grafo original acrescido dos novos atributos, e assim apoiar as tarefas identificadas na etapa de Projeto Navegacional. As outras tarefas do Projeto Navegacional foram mantidas e a tarefa de definição de atributos navegacionais, que antes era executada no mapeamento de classes navegacionais, manteve-se agora por meio do enriquecimento das classes de domínio com descrições de atributos navegacionais.

Sendo assim, a etapa de Projeto navegacional passou a ser composta das seguintes tarefas:

1. Especificação do esquema de contextos
    a. Definição dos contextos
    b. Definição das estruturas de acesso
    c. Definição dos *landmarks*
2. Definição dos atributos navegacionais
3. Especificação das classes em contexto
4. Especificação da navegação facetada

As tarefas 3 e 4 da lista acima permaneceram inalteradas de acordo com a definição apresentada por [Szundy, 2004], por isso não serão apresentadas aqui. Apesar de inalteradas as tarefas 1.a, 1.b e 1.c, as definições de contexto e estruturas de acesso passaram a aceitar expressões na linguagem de consulta SPARQL para as especificações das regras de seleção em vez da linguagem RDQL apresentada por [Szundy, 2004]. A tarefa 2 foi significativamente modificada e será vista na seção 3.4.1.2. A ontologia do modelo navegacional

utilizada para especificar as primitivas das tarefas acima também sofreu alterações e está descrita no apêndice A.

Com a remoção dos mapeamentos de classes navegacionais o esquema de classes navegacionais deixou de ser necessário e, da mesma forma, o conceito de classe navegacional. Apesar disso, foi mantido o conceito de nó navegacional, entendido agora como uma instância do domínio (recurso RDF) acessado a partir de um contexto navegacional.

Os contextos navegacionais, definidos na especificação do esquema de contextos, passam a acessar diretamente as instâncias do domínio, e não um novo grafo construído a partir dessas instâncias. Durante a navegação, o grafo resultante da aplicação do modelo navegacional possui as seguintes características:

1. É um sub-grafo do domínio composto pelos nós (recursos RDF) que atendem as regras de seleção definidas no contexto acessado.
2. O sub-grafo também inclui todos os elos (propriedades RDF) ligados aos nós do contexto, assim como a outra ponta destes elos (valores literais e recursos RDF);
3. Os nós do sub-grafo que são incluídos num contexto apresentam novos elos ligados aos atributos navegacionais, definidos pelas descrições dos atributos navegacionais sobre as classes do domínio.

3.4.1.1. Especificação do esquema de contextos

> 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.
>
> [Szundy, 2004]

Na especificação do esquema de contextos são definidos os contextos navegacionais em que os nós serão acessados, as estrutura de acesso e os *landmarks* da aplicação. A Figura 9 apresenta a representação gráfica de um esquema de contextos navegacionais que manteve a mesma notação apresentada por [Lima, 2003].

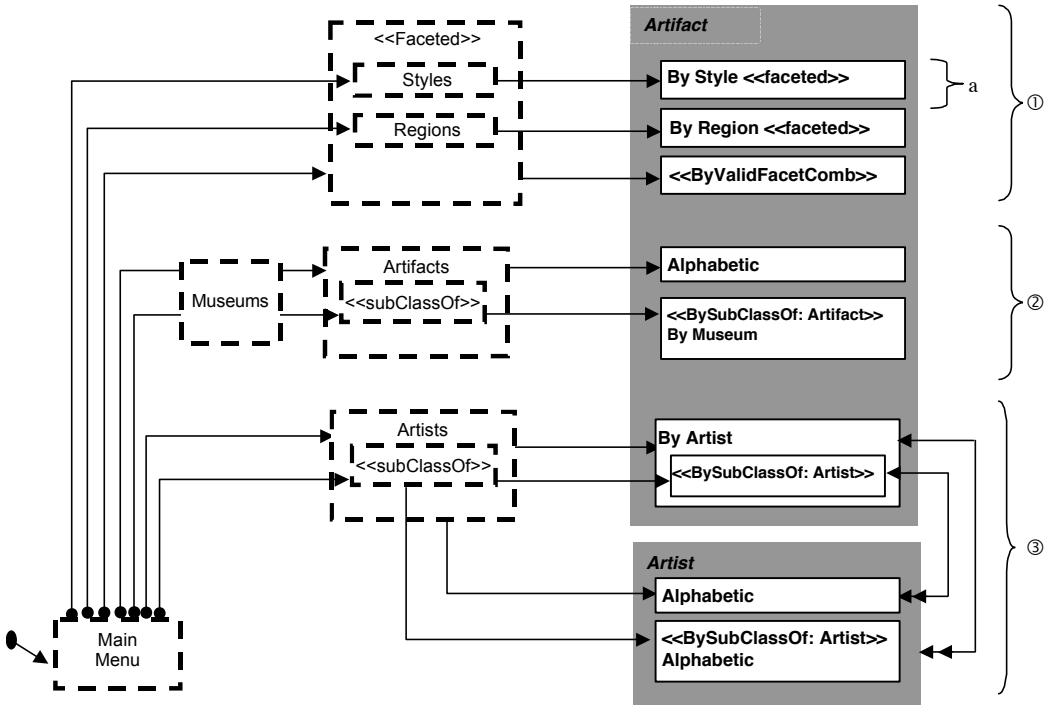


Figura 9 – Esquema de Contextos Navegacionais SHDM da Ontologia de Artes

Para detalhes sobre esta notação, ver [Lima, 2003].

A tarefa de especificação do esquema de contextos sofreu duas mudanças significativas, apresentadas a seguir.

A linguagem SPARQL é uma recomendação do W3C para consultas em base de dados RDF [Prud'hommeaux & Seaborne, 2008]. Além disso, muitas das bases de dados RDF do projeto *Linking Open Data* implementam esta linguagem. Por isso, o método SHDM passou aceitar, a partir deste trabalho, a linguagem SPARQL para a especificação das regras de seleção dos contextos navegacionais. O Quadro 7 apresenta a especificação de um contexto navegacional com a regra de seleção descrita em linguagem SPARQL.

```
:PersonAlpha a shdm:Context ;
  rdfs:context_name  "PersonAlpha" ;
  rdfs:context_query "SELECT DISTINCT ?s
                      WHERE { ?s rdf:type foaf:Person;
                              ?s foaf:name ?name. }
                      ORDER BY ?name".
```

Quadro 7 - Especificação de um contexto com regra de seleção em linguagem SPARQL

Conforme apresentado na seção 3.3.2, o modelo de domínio passou a contar com repositórios de dados externos a aplicação (*datasets*) para complementar as instâncias de domínio. Por isso, os contextos navegacionais também devem fazer uso destes *datasets* na especificação de suas regras de seleção. A especificação dos *datasets* é feita por meio do uso das extensões de federação[55] da linguagem SPARQL 1.1[56], ainda em desenvolvimento, porém com implementações disponíveis em alguns *frameworks* RDF do mercado. O Quadro 8 apresenta uma expressão de consulta em SPARQL 1.1 com extensões de federação de dados.

```
SELECT DISTINCT ?s
WHERE {
   SERVICE <http://dbpedia.org/sparql>
   { ?s rdf:type foaf:Person . }

   SERVICE <http://lod.openlinksw.com/sparql>
   { ?s rdf:type foaf:Person . }
}
```

Quadro 8 – Expressão de consulta em SPARQL 1.1 com extensões de federação

As definições das estruturas de acesso, landmarks e classes em contexto, não sofreram mudanças significativas e continuam conforme apresentado por [Szundy, 2004].

3.4.1.2. Definição dos atributos navegacionais

Para enriquecer as instâncias de domínio com atributos navegacionais, o projetista deve adicionar descrições sobre os atributos navegacionais diretamente nas classes de domínio. O Quadro 9 apresenta a descrição de um atributo navegacional do tipo "âncora para contexto" definido para a classe de domínio foaf:Person em notação RDF Turtle.

```
foaf:Person a owl:Class ;
  rdfs:subClassOf foaf:Agent;
  rdfs:label "Person" ;
  rdfs:comment "A person.";
  shdm:navigation_attributes [
    a shdm:ContextAnchorAttribute;
    shdm:navigation_attribute_name "link";
    shdm:context_anchor_label_expression "foaf:name";
    shdm:context_anchor_target_node_expression "self";
    shdm:context_anchor_target_context :PersonAlpha;
  ].
```

Quadro 9 – Descrição de um atributo navegacional "âncora para contexto" em uma classe de domínio

[55] http://www.w3.org/TR/2010/WD-sparql11-federated-query-20100601
[56] http://www.w3.org/TR/sparql11-query

A Figura 10 apresenta um grafo RDF em que as instâncias de domínio foram enriquecidas com os atributos navegacionais seguindo a especificação apresentada no Quadro 9.

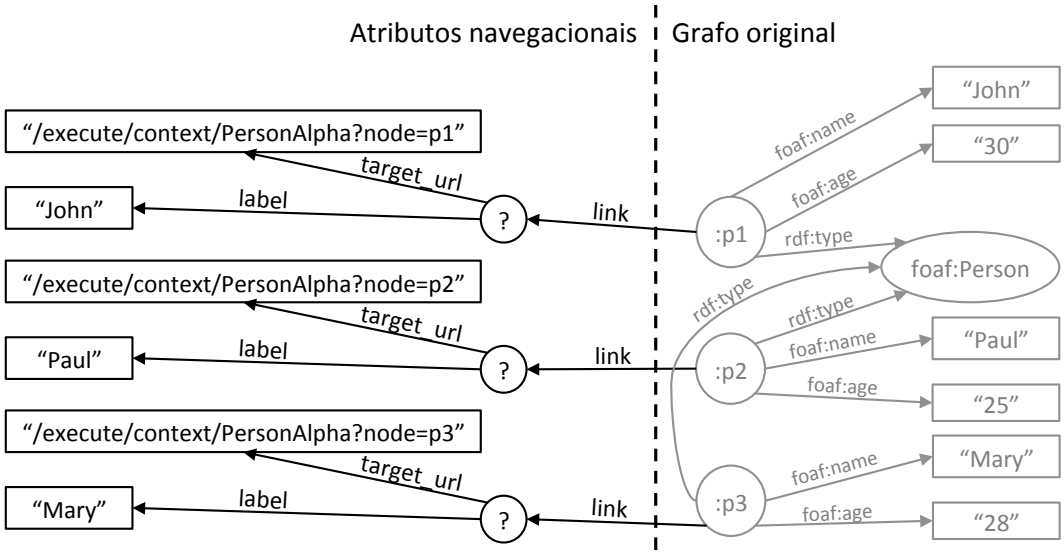


Figura 10 – Grafo RDF enriquecido com atributos navegacionais

Na figura acima é possível observar, no lado direito, que os nós e elos do grafo original foram mantidos intactos. Nos elementos que compõem a parte do grafo gerada pelas definições do modelo navegacional (lado esquerdo), é possível notar a presença dos valores definidos na descrição do atributo navegacional apresentado no Quadro 9:

- O valor da propriedade shdm:navigation_attribute_name definiu o identificador do elo "link" entre os recursos do domínio e o recurso que representa a âncora navegacional;
- O valor da propriedade shdm:context_anchor_label_expression, que representa a propriedade foaf:name, gerou o valor de cada propriedade "label" da âncora navegacional;
- O valor da propriedade target_url da âncora navegacional é computada a partir dos valores das propriedades shdm:context_anchor_target_node_expression e shdm:context_anchor_target_context da definição do atributo navegacional.

É importante não confundir a descrição de um atributo navegacional, apresentada no Quadro 9, com os atributos navegacionais de um nó navegacional, representado pelos nós "?" da Figura 10. O primeiro é definido pelo projetista durante a elaboração do modelo navegacional. O segundo é gerado automaticamente pela aplicação durante a navegação do usuário.

As descrições de atributos navegacionais para as estruturas de acesso (índices) são similares à descrição para classes do domínio apresentada no Quadro 9, com a diferença de que o recurso que recebe o atributo shdm:navigation_atributes é do tipo shdm:Index. As entradas de índice são compostas por instâncias do domínio enriquecidas com os atributos navegacionais, assim como os nós navegacionais.

3.4.1.3. Impactos das mudanças no modelo navegacional

Sem os mapeamentos de classes navegacionais, o modelo navegacional passou a ser mais simples e conforme com o modelo RDF. A aplicação do modelo navegacional sobre o grafo de instâncias do domínio não remove triplas do grafo original e adiciona novos elos aos nós originais, constituindo em uma transformação mais simples do que a geração de um grafo completamente novo. Entretanto, o mapeamento de classes navegacionais apoiava alguns requisitos que precisam continuar sendo apoiados com a nova abordagem. A seguir serão apresentadas as diferenças na abordagem atual em relação à anterior, numeradas de acordo com cada mapeamento descrito na Tabela 3:

1. O mapeamento entre classes de domínio e classes navegacionais passou a ser desnecessário, visto que os contextos navegacionais acessam diretamente as instâncias do domínio;
2. Os filtros passam a ser descritos diretamente nas regras de seleção dos contextos navegacionais. Também é possível criar filtros similares estendendo o modelo de domínio com classes OWL descritas a partir de restrições de propriedades [Smith, Welty & McGuinness, 2004]. Neste caso, mesmo que a mudança seja no modelo de domínio, o requisito pode ter sido identificado no modelo navegacional;

3. O atributo navegacional simples deixou de existir como primitiva do modelo navegacional, já que as propriedades dos nós do domínio podem ser acessadas diretamente;
4. Os elos navegacionais também deixaram de existir como primitiva do modelo navegacional. Como os elos eram usados para construir os atributos navegacionais do tipo âncora ou índice, na nova abordagem, continua sendo possível definir tais atributos diretamente a partir das propriedades dos nós do domínio. Ainda assim, se for necessário criar novas propriedades baseadas nas propriedades do grafo original, é possível estender o modelo de domínio através das inferências lógicas das propriedades OWL (owl:TransitiveProperty e owl:SymmetricProperty).
5. A definição dos atributos navegacionais se dá pelo enriquecimento das classes do modelo de domínio com descrições de atributos navegacionais, em vez de uma regra de mapeamento.

Com a nova abordagem, perdeu-se a possibilidade de ocultar as propriedades dos recursos RDF do modelo de domínio para os modelos que acessam o modelo navegacional. Porém isso não é considerado um problema grande, devido a natureza aberta dos dados da Web Semântica. Ainda assim, continua sendo possível ocultar informações do usuário na especificação das interfaces, na etapa de Projeto de Interface.

3.5. Projeto Comportamental

Na etapa de Projeto Comportamental, proposta originalmente para o método SHDM por [Santos, 2010], é produzido o modelo de operações, que é um modelo geral para especificação da lógica de negócio de aplicações. A primitiva básica deste modelo é a operação cujo principal papel é causar transformações no estado da aplicação.

No presente trabalho, o modelo de operações sofreu pequenas mudanças em seu vocabulário para torná-lo mais simples, mas mantendo a mesma semântica da proposta original.

O diagrama de classes apresentado na Figura 11 ilustra as classes e propriedades que compõem o vocabulário do modelo de operações.

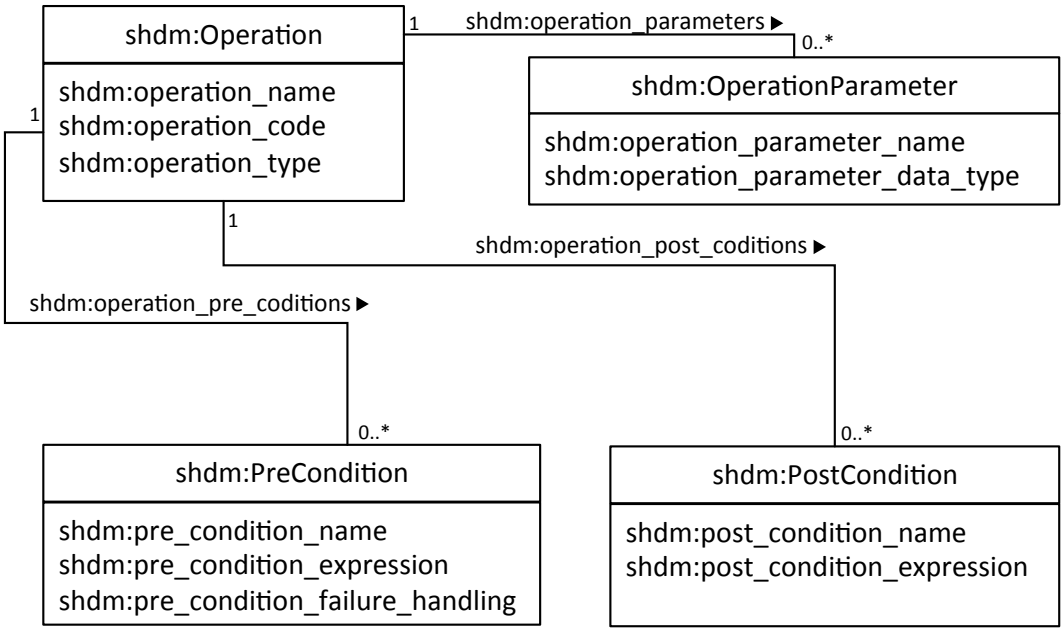


Figura 11 – Vocabulário do modelo de operações

As operações, instâncias da classe shdm:Operation, adicionam comportamentos à aplicação. A propriedade shdm:operation_code contém a definição de um procedimento a ser executado. A propriedade shdm:operation_type indica se a operação é interna ou externa. As operações internas descrevem as regras de negócio do domínio da aplicação e não podem ser invocadas por agentes externos (usuários ou sistemas). As operações externas podem ser invocadas por agentes externos e dão apoio a interação do usuário com a aplicação.

Uma operação pode ter parâmetros de entrada, instâncias de shdm:OperationParameter, que funcionam de forma similar a parâmetros de métodos de linguagens de programação. Uma operação também pode ter pré-condições e pós-condições. As pré-condições, instâncias de shdm:PreCondition, são avaliadas antes do código da operação e em caso de falha, um código de tratamento de falha é executado. As pós-condições, instâncias de shdm:PostConditions, são avaliadas após a execução do código da operação e retornam um erro em caso de falha, mas sem desfazer as mudanças geradas pela execução do código da operação.

O Quadro 10 apresenta uma especificação completa em RDF de uma operação do SHDM que representa a operação matemática de divisão entre números inteiros. A operação espera por dois parâmetros de entrada (“a” e “b”) que devem ser do tipo inteiro e possui uma pré-condição para evitar a divisão por zero.

```
:division a shdm:Operation ;
  shdm:operation_name "division";
  shdm:operation_code "a / b";
  shdm:operation_type "internal";
  shdm:operation_parameters [
     a shdm:OperationParameter;
     shdm:operation_parameter_name "a";
     shdm:operation_parameter_data_type "Integer";
  ];
  shdm:operation_parameters [
     a shdm:OperationParameter;
     shdm:operation_parameter_name "b";
     shdm:operation_parameter_data_type "Integer";
  ];
  shdm:operation_pre_conditions [
     a shdm:PreCondition;
     shdm:pre_condition_name "zero_division_error";
     shdm:pre_condition_expression "b != 0";
     shdm:pre_condition_failure_handling "return 'Zero division error!'"
  ].
```

Quadro 10 – Descrição de uma operação em RDF

A linguagem utilizada para a definição do código da operação depende do ambiente de implementação da aplicação. O método SHDM não força o uso de nenhuma linguagem específica. Normalmente utiliza-se a mesma linguagem de programação de propósito geral utilizada na implementação da aplicação. Contudo, é possível utilizar linguagens mais especializadas, como linguagens de workflow ou de descrição de processos de negócios, ou mesmo usar uma linguagem específica de domínio (*Domain Specific Language* - DSL) [Van Deursen et al., 2000] própria, desde que a arquitetura de execução da aplicação implemente a semântica dessas linguagens. Por exemplo, a biblioteca Java[57] BPELJ[58] é um interpretador para a linguagem BPEL(*Business Process Execution Language*)[59], para descrição de processos de negócio. Uma aplicação modelada com o SHDM e implementada em linguagem Java, poderia utilizar a linguagem BPEL para a definição dos códigos das operações. No ambiente Synth,

[57] http://www.java.com
[58] http://www.ibm.com/developerworks/library/specification/ws-bpelj
[59] http://www.bpelsource.com/bpel_info/spec.html

apresentado no capítulo 4, os códigos das operações são descritos em linguagem Ruby[60].

É importante que a linguagem utilizada seja capaz de expressar a semântica de acesso às primitivas dos modelos da aplicação, ou seja, a linguagem deve conhecer os conceitos de classes, propriedades e recursos do modelo de domínio em RDF; contextos, nós, índices, entradas de índices e atributos navegacionais do modelo navegacional; interface abstrata, descrição retórica, *widgets* concretos e efeitos do modelo de interfaces e, finalmente, as primitivas do próprio modelo de operações: operações, parâmetros, pré-condições e pós-condições. Como todos os modelos do SHDM são descritos em RDF, basta que linguagem possa expressar a semântica de acesso aos dados em RDF, a mesma do modelo de domínio, para poder expressar a semântica de acesso a qualquer das primitivas dos outros modelos.

3.5.1. Comportamentos pré-definidos

Quando um usuário executa uma tarefa de navegação, ou seja, ativa uma âncora navegacional para percorrer de um nó origem até um nó destino, vários processamentos ocorrem na aplicação:

1. Um conjunto de nós do contexto acessado fica disponível ao usuário;
2. Todos os atributos do nó destino também ficam disponíveis;
3. Os atributos do nó origem podem ficar indisponíveis;
4. Alguns nós do contexto do nó origem podem ficar indisponíveis se o contexto destino for diferente;
5. O nó origem pode ou não ficar indisponível dependendo do contexto do nó destino.

Os processamentos listados acima descrevem a semântica de navegação, que é independente do domínio da aplicação, ou seja, são sempre os mesmos para qualquer modelo de domínio. Esses processamentos podem ser expressos em um conjunto de passos executados pela aplicação que descrevem um comportamento. Sendo assim, a navegação pode ser considerada um comportamento pré-definido

[60] http://www.ruby-lang.org

da aplicação e, consequentemente, pode ser descrita pelo modelo de operações. O mesmo é válido para qualquer modelo do SHDM cuja semântica possa ser descrita em passos processáveis pela aplicação.

Os comportamentos pré-definidos são instâncias de operações, independentes do modelo de domínio da aplicação, que são descritas com o vocabulário do modelo de operações utilizado para dar semântica aos outros modelos do SHDM. Sendo assim, as semânticas do modelo navegacional, modelo de interface e de qualquer novo modelo que venha a fazer parte do método poderiam ser descritas pelo modelo de operações, se puderem ser processadas pelo interpretador do modelo de operações.

No ambiente Synth, a semântica de navegação foi implementada como instâncias do vocabulário do modelo de operações. O Quadro 11 apresenta a especificação da operação pré-definida de navegação em contexto do Synth, que utiliza métodos e variáveis do *framework* sobre o qual o ambiente foi desenvolvido. O trecho destacado em negrito, ilustra o acesso à uma primitiva do modelo navegacional: o contexto navegacional.

```
:context a shdm:Operation ;
  shdm:operation_name "context";
  shdm:operation_code "
context_id = params.delete(:id)
node_id    = params.delete(:node)
context    = SHDM::Context.find(context_id)

@context   = context.new(params) #instance a context with user parameters
@current_node   =@context.nodes.first if node_id.nil? or node_id.blank?
@current_node ||=@context.nodes.select{|node| node.uri == node_id}.first
@current_node ||=NodeDecorator.new(RDFS::Resource.new(node_id), @context)

render :text => render_context(@context, @current_node)
";

  shdm:operation_type "external".
```

Quadro 11 – Especificação do código da operação de navegação em contexto do Synth

A lista a seguir descreve em linguagem natural a especificação da operação apresentada no quadro acima:

1. Extrai os valores “id” (identificador do contexto navegacional) e “node” (identificador do nó navegacional) dos parâmetros (variável “params”) recebidos na chamada à operação de navegação (i.e. quando o usuário clica no link);

2. Recupera a representação da especificação completa do contexto navegacional na base de dados da aplicação;
3. Instancia a versão de *runtime* do contexto navegacional com os parâmetros recebidos durante a navegação, para gerar o conjunto de nós navegacionais do contexto de acordo com esses parâmetros;
4. Seleciona a versão de *runtime* do nó navegacional corrente entre os nós do contexto a partir do identificador do nó obtido no passo 1.
5. Finalmente, delega para o módulo de interface a geração da representação visual (interface gráfica) do nó navegacional acessado no contexto indicado.

3.6. Projeto de Interfaces

O objetivo da etapa de Projeto de Interfaces é construir o Modelo de Interfaces da aplicação, que representa os elementos da aplicação percebidos pelo usuário e as interações do usuário com esses elementos. O modelo de interfaces para o SHDM foi proposto originalmente por [Moura, 2004] e estendido por [Luna, 2009] com formalismos para expressar o funcionamento das interfaces presentes nos sites da Web 2.0, conhecidas como interfaces RIA (*Rich Internet Application*) [Bozzon et al., 2006].

No presente trabalho, o Modelo de Interfaces não sofreu nenhuma alteração, por isso, será apresentado nesta seção de forma resumida.

O Modelo de Interfaces se baseia na ideia de que é possível separar a "essência" de uma interface de sua apresentação. Isto é possível pela decomposição da especificação da interface em um Modelo de Interface Abstrata e um Modelo de Interface Concreta. A Figura 12 apresenta o meta modelo de Interface Abstrata do SHDM.

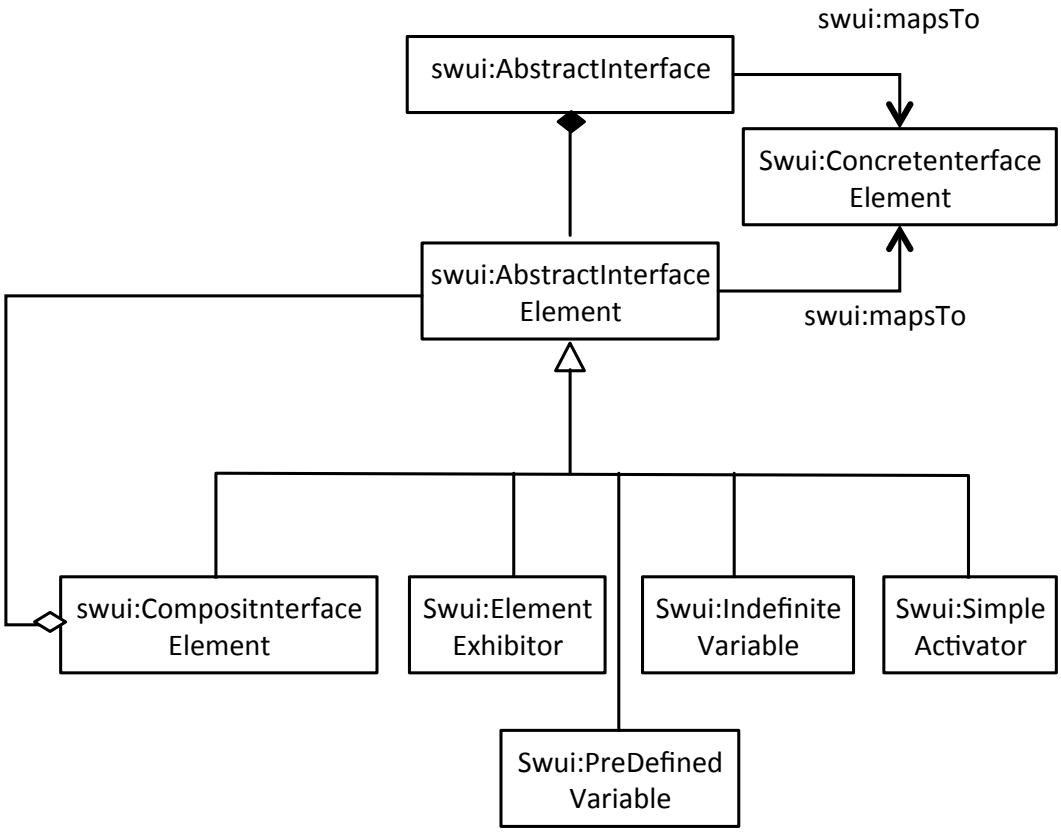


Figura 12 – Meta modelo de Interface Abstrata

A Interface Abstrata foca nos papéis desempenhados por cada *widget* de interface na troca de informações entre a aplicação e o mundo exterior, incluindo o usuário. Esta interface é abstrata no sentido de que ela não captura como a interface será apresentada visualmente ou qualquer informação que dependa do ambiente de execução. A Interface Concreta é responsável por capturar detalhes de apresentação e execução da interface.

Em resumo, no meta modelo de Interface Abstrata, uma interface abstrata é a composição de elementos de interface abstrata (*widgets*). Estes elementos podem ser um *ElementExhibitor*, que é capaz de apresentar valores; um *IndefiniteVariable*, que é capaz de capturar uma entrada textual arbitrária; um *DefinedVariable*, que é capaz de capturar valores de entrada (um ou vários) a partir de um conjunto conhecido de alternativas; ou um *SimpleActivator*, que é capaz de reagir a um evento externo e sinalizá-lo para a aplicação.

A partir de uma Interface Abstrata, uma especificação de mapeamento feita pelo projetista determina como cada *widget* será mapeado em um ou mais elementos de Interface Concreta ou Operações.

Observe que o mapeamento em Operações unifica o acesso ao Modelo de Domínio, caso a operação utilizada seja uma operação CRUD (Create, Read, Update e Delete) [Kilov, 1990] sobre um elemento do domínio; ao Modelo Navegacional, caso a operação seja de Navegação; e ao Modelo Operações em qualquer outro caso. Esta estratégia permite uma separação clara entre a navegação hipertextual e outras operações do domínio. Quando um evento (i.e., clique do usuário) é capturado pela interface (via um *widget SimpleActivator*), o mapeamento para operações determinará que operações serão executadas, independente de se tratar de uma operação de navegação ou outro tipo de operação.

Para mais detalhes sobre o Modelo de Interfaces, em particular sobre seu uso com Interfaces RIA, consulte [Luna, 2009].