4
O Projeto Conceitual SHDM

O Projeto Conceitual é a etapa responsável pela elaboração dos seguintes artefatos:

- Esquema Conceitual SHDM;
- Ontologia Conceitual SHDM;
- Instâncias.

O Esquema Conceitual SHDM é um modelo de classes e relacionamentos a respeito de um domínio específico. Este artefato é útil para expressar abstrações do domínio original e serve como ponto de partida para criação das ontologias que serão efetivamente implementadas.

A Ontologia Conceitual SHDM é uma definição do domínio em um formato implementável na Web Semântica, utilizando linguagens de ontologias para Web, como DAML+OIL, RDF(S) e RDF. As instâncias representam os dados propriamente ditos e suas definições devem ser válidas de acordo com a ontologia conceitual.

Nas subseções seguintes apresentaremos as primitivas de modelagem conceitual, seus significados semânticos e seus mapeamentos para as linguagens da Web Semântica mencionadas.

4.1.
Esquema Conceitual

O método SHDM utiliza classes e relacionamentos orientados a objetos representados na linguagem UML para modelar um domínio específico de uma aplicação Web. Alguns estereótipos são utilizados para permitir a especificação de detalhes adicionais pertinentes a aplicações Web.

Quando comparamos o modelo orientado a objetos (OO) com o modelo RDF, podemos afirmar que conceitos de classes e subclasses (relações de especialização e generalização) podem ser modelados de forma semelhante. Entretanto, existe uma diferença significativa quando modelamos atributos OO e associações OO para em seguida representá-los no modelo RDF. Estas duas abstrações OO são modeladas através de propriedades RDF indistintamente, ou

seja, em modelos RDF não existe distinção entre uma propriedade que descreve uma classe (atributo) e uma propriedade que descreve uma relação de associação com outra classe. Além disto, propriedades RDF podem ser especializadas através de relações de subjugamento (*subsumption)*[26], permitindo a criação de subrelacionamentos. Nosso Esquema Conceitual tira proveito destas novas características conforme descrito a seguir.

4.1.1. Subclasse

No esquema conceitual SHDM representamos relacionamentos de especialização/generalização (herança) através da notação UML tradicional. Com isto definimos graficamente subclasses em um domínio. Este conceito de subclasse também existe no modelo de dados do RDF e esta similaridade entre os modelos permite um mapeamento simples, conforme descrito a seguir.

Notação:

Figura 9 - Notação de subclasse (como na UML)

[26] Hierarquias de subjugamento ("subsumption"), taxonomias e generaliza o: A relação de subjugamento pode ser entendida como uma relação de implica o conectando conceitos mais específicos a conceitos mais gerais, em taxonomias conceituais. Em termos formais, o subjugamento define um reticulado ("lattice"), um tipo de ordenação parcial, que pode ser representado por um grafo direcionado acíclico. Os grafos hierárquicos definidos pelo subjugamento no precisam ser rvores, podendo ser tipos mais gerais de grafos, nos quais os ns filho são re-entrantes, i.e., um n pode ter mais de um pai. No entanto, comum o reticulado definido pelo subjugamento possuir uma estrutura nuclear em rvore, com a superposição de uma ou mais rvores, ou de estruturas de classificação cruzadas. A rela o de subjugação pode ser vista como uma relação de generalização, no sentido de que o subjugador ("subsumer") expressa uma generaliza o sobre o subjugado ("subsumed"). [http://coral.lili.uni-bielefeld.de/DATR/inherlexweb/node7.html]

Exemplo:

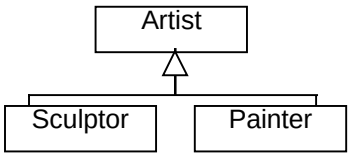

Figura 10 - Exemplo de subclasse

Semântica:

Este é o exemplo mais simples do esquema de classes e é equivalente ao conhecido relacionamento de especialização/generalização, onde as classes *Scuptor* e *Painter* são subclasses de *Artist* e herdam seus atributos e métodos. Apresentamos este exemplo simples apenas para introduzir a notação equivalente em DAML+OIL.

Mapeamento para DAML+OIL:

```
<daml:Class rdf:ID="Sculptor">
     <rdfs:subClassOf rdf:resource="#Artist"/>
</daml:Class>
```

4.1.2. Subrelacionamento

A modelagem de subrelacionamentos é representada com uma nova notação: uma linha pontilhada com uma seta, semelhante à notação de especialização de classes. A diferença está na linha pontilhada conectando super-relacionamentos com subrelacionamentos. Esta inovação mostra-se útil principalmente quando precisamos explorar super-relacionamentos sem precisar conhecer “a priori” todos os subrelacionamentos existentes.

Notação:

Figura 11 - Notação de Subrelacionamento

Exemplo:

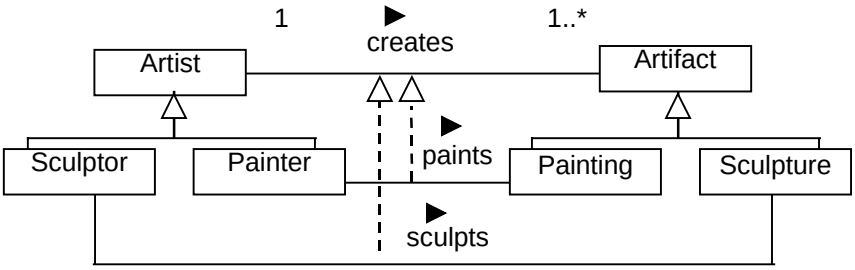

Figura 12 - Exemplo de Subrelacionamento

Semântica:

Utilizamos esta notação para representar a especialização de um relacionamento. No exemplo da Figura 12, o relacionamento *creates* é especializado em dois relacionamentos: *paints* e *sculpts*. Com o uso desta notação podemos especificar um relacionamento de generalização entre associações e permitir ao projetista fazer referência a uma associação mais geral (*creates*), sempre que for útil e apropriado. Outra vantagem desta abordagem é que poderemos fazer consultas em todas as instâncias de *creates* incluindo as instâncias dos subrelacionamentos (se desejado) .

A linguagem de consulta que escolhemos (RQL) tem capacidade de explorar estas novas especializações/generalizações entre relacionamentos. Poderemos consultar o esquema e obter os nomes de todos os subrelacionamentos de *creates* independente da quantidade de subníveis. (ex: *subPropertyOf(creates)).* Ou se desejarmos, poderemos obter apenas os subrelacionamentos diretos sem transitividade (ex: ^*subPropertyOf(creates)).* Em relação às instâncias, quando efetuarmos consulta utilizando o super-relacionamento, as instâncias dos subrelacionamentos também serão retornadas. Caso queiramos recuperar apenas as instâncias de um subrelacionamento, bastará usar seu nome.

Mapeamento para DAML+OIL[27]:

```
<rdf:Property rdf:ID="paints">
      <rdfs:domain rdf:resource="#Painting"/>
      <rdfs:range rdf:resource="#Painter"/>
      <rdfs:subPropertyOf rdf:resource="#creates"/>
</rdf:Property>
```

4.1.3. Multiplicidade de Atributo

Conforme [OMG 1999], a multiplicidade de atributos é o número possível de valores que uma instância pode conter, ou seja, significa a quantidade de vezes que um atributo ocorre nas instâncias de uma determinada classe. No caso comum em que a multiplicidade é 1..1, o atributo necessariamente existe com exatamente um valor. A multiplicidade de atributos é particularmente útil para representar dados semi-estruturados, pois permite ao projetista acomodar múltiplas ocorrências (ou nenhuma ocorrência) de um atributo.

Notação*: email[0..*]: xsd:string*

Exemplo:

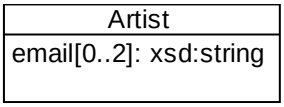

Figura 13 - Exemplo de Multiplicidade de Atributo

Semântica:

No exemplo da Figura 13, a classe *Artist* possui um atributo *email* com multiplicidade [0..2]. Esta faixa de valores foi escolhida para exemplificar instâncias de artistas que podem tanto não possuir *email* quanto possuir um ou dois *emails*.

[27] Vale ressaltar que já atualizamos nossos mapeamentos para refletir a decisão do consórcio W3C em retirar as definições DAML+OIL que fossem redundantes em relação a RDF(S), conforme [van Harmelen et al. 2003]. Neste exemplo de mapeamento, não utilizamos *daml:Property* e sim *rdf:Property*.

Mapeamento para DAML+OIL:

```
<daml:Restriction>
      <daml:onProperty rdf:resource="#email"/>
      <daml:minCardinality>0</daml:minCardinality>
      <daml:maxCardinality>2</daml:maxCardinality>
</daml:Restriction>
```

4.1.4. Tipos de Dados utilizando XML Schema

Ao definir os tipos de dados de cada atributo, utilizamos os tipos pré-definidos em XML Schema Part 2: Datatypes [Biron & Malhotra 2001]. Esta decisão aparentemente simples merece dois comentários: (1) Esta recomendação XML Schema foi um avanço por si, pois anteriormente todos os tipos de dados XML eram apenas literais; (2) Somente na segunda versão de DAML+OIL o suporte aos tipos de dados XML foi oferecido.

Notação: *creationDate: xsd:date*

Exemplo:

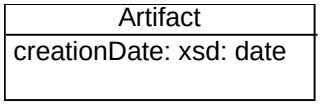

Figura 14 - Exemplo de Tipos de Dados utilizando XML Schema

Semântica:

Esta notação permite explorar a definição dos mais de 20 tipos de dados definidos na recomendação como date, integer, nonNegativeInteger, etc.

Mapeamento para DAML+OIL:

```
<daml:DatatypeProperty rdf:ID="creationDate">
      <daml:range
rdf:resource="http://www.w3.org/2000/10/XMLSchema#date"/>
      <daml:domain rdf:resource="#Artifact"/>
</daml:DatatypeProperty>
```

4.1.5.
Enumeração de Atributo

A enumeração permite definir exaustivamente os elementos que podem ocorrer como valores de um atributo em uma instância. Nenhum outro valor é permitido.

Notação: *type: [electronic, physical]*

Exemplo:

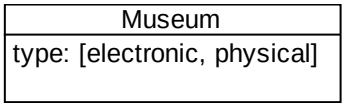

Figura 15 - Exemplo de Enumeração de Atributo

Semântica:

Esta notação é útil sempre que necessitarmos especificar todos os possíveis valores que um atributo pode receber quando as instâncias de classe forem criadas. Esta notação na verdade representa um domínio enumerado para um tipo de dados que será usado por um atributo. Na Figura 15 são apresentados dois valores possíveis para o tipo de um museu: eletrônico e físico.

Definimos uma heurística de mapeamento de atributos enumerados SHDM para DAML+OIL: o nome da classe DAML+OIL que representará a coleção de valores será sempre XKind onde X é o nome da classe UML. (OBS: o atributo *parseType="daml:collection"* é uma extensão ao RDF e indica que os seus subelementos devem ser tratados como uma unidade (uma única coleção)).

Mapeamento para DAML+OIL:

```
<daml:Class rdf:about="MuseumKind">
     <daml:oneOf parseType="daml:collection">
          <art:MuseumKind rdf:about="electronic"/>
          <art:MuseumKind rdf:about="physical"/>
     </daml:oneOf>
</daml:Class>
```

4.1.6.
Classe Inferida

Em DAML+OIL é possível criar definições de classes “intensionais” através do uso de expressões booleanas ou através de condições (necessárias ou

necessárias e suficientes) para classificação em taxonomias. Como estas classes “intensionais” podem conduzir a modelos extremamente vastos e complexos, o projetista da aplicação pode fazer uso desta notação para distingui-las das classes declaradas explicitamente como subclasses.

Esta inovação do método SHDM é útil, portanto, sempre que desejarmos definir classes intensionalmente, sem conhecer “a priori” todas as possibilidades. Por exemplo, “Produto em promoção” que corresponde a qualquer produto (ou subclasse de Produto) cujo atributo “desconto” é diferente de 0.

Outra utilidade da classe inferida é quando estamos fazendo uma engenharia reversa de um domínio, ou seja, quando já existe uma ontologia definida e queremos reusá-la. Neste caso, é necessário modelar tudo que foi previamente definido, por exemplo, em DAML+OIL e representar as classes em diagramas da UML.

As classes definidas em DAML+OIL explicitamente como subclasses são representadas no esquema conceitual SHDM sem este estereótipo. Entretanto, aquelas classes que dependem de condições “booleanas” de restrição, para serem classificadas corretamente como subclasses, devem ser modeladas com o estereótipo. Desta forma, posteriormente, o projetista pode validar mais facilmente sua modelagem com o uso de uma máquina de inferência.

Notação: estereótipo da UML chamado <<inferred>>

Exemplo:

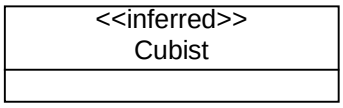

Figura 16 - Exemplo de Classe Inferida

Semântica:

Com esta notação podemos destacar no modelo as classes cuja classificação não foi definida através de declarações explícitas, ou seja, classes que serão subclasses de outras através de avaliação de uma restrição, por exemplo, uma expressão “booleana” de conjunção/interseção.

Mapeamento para DAML+OIL:

```
<daml:Class rdf:ID="Cubist">
     <daml:intersectionOf rdf:parseType="daml:collection">
          <rdfs:Class rdf:about="#Painter"/>
          <daml:Restriction rdf:about="#Cubism-
Restriction"/>
     </daml:intersectionOf>
</daml:Class>
<daml:Restriction rdf:ID="Cubism-Restriction">
     <daml:onProperty rdf:resource="#style"/>
     <daml:hasValue rdf:resource="#Cubism"/>
</daml:Restriction>
```

Neste trecho acima definimos a classe *Cubist*. As instâncias que satisfizerem as condições definidas serão classificadas como instâncias desta classe.

Este trecho DAML+OIL especifica que a classe *Cubist* é uma subclasse inferida de *Painter*, pois é a interseção da classe *Painter* com a restrição que define o seu único estilo possível de pintura como *Cubism*. Esta é uma definição de classe intensional.

Abaixo apresentamos um exemplo que define a classe *Cubist* explicitamente como uma subclasse da classe *Painter*. Para a classe *Cubist* existe uma restrição nos valores da propriedade estilo, sendo permitido apenas o estilo *Cubism*.

```
<daml:Class rdf:ID="Cubist">
     <daml:Restriction>
          <daml:onProperty rdf:resource="#style"/>
          <daml:hasValue rdf:resource="#Cubism"/>
     </daml:Restriction>
</daml:Class>
<daml:Class rdf:about="#Cubist">
     <rdfs:subClassOf rdf:resource="#Painter"/>
</daml:Class>
```

Após modelar as classes inferidas, o projetista da aplicação pode fazer uso de máquinas de inferência com suporte a DAML+OIL para validar as hierarquias de especialização/generalização.

4.1.7. Estereótipo “ArbitraryClassHierarchy”

Além das classes inferidas, em nosso metamodelo definimos outro estereótipo para representar hierarquias de classes com profundidade arbitrária,

chamada *arbitraryClassHierarchy*. No entanto, precisaremos aguardar a especificação da linguagem OWL para detalhar a definição do metamodelo SHDM, uma vez que DAML+OIL não oferece suporte para definição de meta-modelos.

Notação: <<arbitraryClassHierarchy>>

Exemplo:

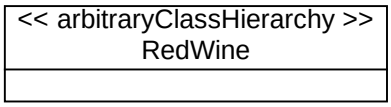

Figura 17 - Exemplo de hierarquia de profundidade arbitrária

Semântica:

Esta notação também pode ser utilizada ao realizarmos uma “engenharia reversa”. Ela pode ser usada para substituir uma grande quantidade de classes que participam de uma mesma hierarquia de especialização. No caso, a classe que recebe o estereótipo é aquela que representa a generalização. A sua utilidade pode ser observada quando o modelo fica muito extenso e queremos representá-lo de modo mais conciso, sem explicitar todos os nomes de classes que não possuem significados adicionais.

Mapeamento para DAML+OIL:

o mapeamento é a declaração de todas as classes envolvidas

4.1.8. Relacionamento Inverso

A modelagem de relacionamentos inversos é representada com uma nova notação: duas setas e os nomes dos relacionamentos separados por uma barra. O primeiro nome de relacionamento é obrigatoriamente o nome correspondente a seta que aponta para o lado esquerdo. Como estamos definindo modelos para serem implementados, esta especificação de relacionamentos inversos será útil para elaborar consultas.

Notação:

Figura 18 - Notação de Relacionamento Inverso

Exemplo:

Figura 19 - Exemplo de Relacionamento Inverso

Semântica:

O primeiro nome de relacionamento é obrigatoriamente o nome correspondente à seta que aponta para o lado esquerdo. Ou seja, na Figura 19 estamos modelando o relacionamento de associação onde um Artifact é criado por um Artist e este Artist cria um ou mais Artifact.

Mapeamento para DAML+OIL:

```
<rdf:Property rdf:ID="isCreatedBy">
      <daml:inverseOf rdf:resource="#creates"/>
</rdf:Property>
```

4.2. Exemplo resumido da Ontologia Conceitual

O pequeno exemplo de Esquema Conceitual que aparece na Figura 12 ao ser mapeado para uma Ontologia Conceitual SHDM pode gerar um representação RDF/XML conforme a Figura 20 abaixo.

```
        <rdf:Description rdf:about="http://www.icom.com/schema.rdf#Painter">
                <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-
schema#Class"/>
                <rdfs:subClassOf
rdf:resource="http://www.icom.com/schema.rdf#Artist"/>
                <rdfs:label xml:lang="en">Painter</rdfs:label>
                <rdfs:label xml:lang="nl">Schilder</rdfs:label>
        </rdf:Description>
        <rdf:Description rdf:about="http://www.icom.com/schema.rdf#Painting">
                <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-
schema#Class"/>
                <rdfs:subClassOf
rdf:resource="http://www.icom.com/schema.rdf#Artifact"/>
                <rdfs:label xml:lang="en">Painting</rdfs:label>
                <rdfs:label xml:lang="nl">Schilderij</rdfs:label>
        </rdf:Description>
        <rdf:Description rdf:about="http://www.icom.com/schema.rdf#paints">
                <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-
ns#Property"/>
                <rdfs:subPropertyOf
rdf:resource="http://www.icom.com/schema.rdf#creates"/>
                <rdfs:domain
rdf:resource="http://www.icom.com/schema.rdf#Painter"/>
                <rdfs:range
rdf:resource="http://www.icom.com/schema.rdf#Painting"/>
                <rdfs:label xml:lang="en">paints</rdfs:label>
                <rdfs:label xml:lang="nl">schildert</rdfs:label>
        </rdf:Description>
```

Figura 20 – Trecho da Ontologia Conceitual de Artes

4.3. Instâncias

A Figura 21 apresenta algumas instâncias deste mesmo exemplo.

```
        <rdf:Description rdf:about="http://www.european-
history.com/jpg/guernica03.jpg">
                <rdf:type rdf:resource="http://www.icom.com/schema.rdf#Painting"/>
                <exhibited xmlns="http://www.icom.com/schema.rdf#"
rdf:resource="http://www.museum.es/"/>
                <technique xmlns="http://www.icom.com/schema.rdf#"
xml:lang="en">oil on canvas</technique>
        </rdf:Description>
        <rdf:Description rdf:about="http://www.european-history.com/picasso.html">
                <rdf:type rdf:resource="http://www.icom.com/schema.rdf#Cubist"/>
                <paints xmlns="http://www.icom.com/schema.rdf#"
rdf:resource="http://www.european-history.com/jpg/guernica03.jpg"/>
                <paints xmlns="http://www.icom.com/schema.rdf#"
rdf:resource="http://www.museum.es/woman.qti"/>
                <first_name xmlns="http://www.icom.com/schema.rdf#"
xml:lang="en">Pablo</first_name>
                <last_name xmlns="http://www.icom.com/schema.rdf#"
xml:lang="en">Picasso</last_name>
        </rdf:Description>
```

Figura 21 – Trecho das instâncias da Ontologia Conceitual de Artes

Pode-se observar que o vocabulário do modelo conceitual é essencialmente DAML+OIL.