3
Viabilizando interoperabilidade através de OnOCs

3.1
Motivação

O capítulo anterior descreveu abordagens para facilitar interoperabilidade entre sistemas de informação. Entre elas estão as federações de bancos de dados, a arquitetura de mediadores, as abordagens baseadas em ontologias e metadados, o uso de repositórios de meta-informações, e tecnologias emergentes como a *Dataweb*.

De maneira geral, para interoperar, os sistemas de informação devem ser capazes de atender os seguintes requisitos:

(R1) Localizar as fontes de objetos.

(R2) Acessar as fontes de objetos.

(R3) Interpretar e processar os objetos contidos nas fontes.

A heterogeneidade das fontes é um dos maiores desafios enfrentados para atender o requisito R3. Para agravar a situação, é crescente o número de fontes de objetos independentes disponíveis na Web: bancos de dados, páginas, documentos, entre outros.

Este capítulo introduz então o conceito de Catálogo de Objetos baseado em Ontologias (*Ontology-based Object Catalog* - OnOC) como uma estratégia para facilitar a interoperabilidade entre fontes de objetos. A abordagem baseia-se numa solução híbrida utilizando tecnologias oriundas de abordagens tradicionais e inovadoras, tais como: catálogos de metadados, servidores de ontologias, mediadores e federações de bancos de dados.

Basicamente, um OnOC armazena uma ontologia de referência, contendo instâncias de referência, bem como os metadados das fontes que desejam interoperar, descritos na forma de ontologias, chamadas de ontologias locais. A interoperabilidade é viabilizada através dos mapeamentos definidos entre as ontologias locais e a ontologia de referência, e dos mapeamentos entre os objetos armazenados nas fontes e as instâncias de referência.

Neste trabalho, usamos o termo “objeto” para denotar qualquer entidade ou conceito do mundo real materializado para ser manipulada por um sistema

computacional. Cada objeto pertence a uma classe e conjuntos de objetos de uma mesma classe possuem propriedades em comum. Por exemplo, Aeroporto e Vôo são classes de objetos. Cada Aeroporto possui um código e um nome, que são atributos comuns dos objetos da classe Aeroporto. O aeroporto “Antônio Carlos Jobim” é um exemplo de objeto da classe Aeroporto. Um objeto de uma classe é chamado de *instância* da classe.

Usamos ainda o termo “fonte de objetos” ou “fonte” para designar um local de armazenamento das classes de objetos. Bancos de dados, arquivos texto, documentos semi-estruturados e páginas Web são exemplos das diferentes tecnologias de armazenamento utilizadas pelas fontes de objetos.

3.2 Metodologia

3.2.1 Descrição

De forma geral, um OnOC é um ambiente composto por ontologias locais, uma ontologia de referência e um conjunto de instâncias de referência. Cada componente possui um papel bem definido no ambiente, que serão discutidos em detalhes nesta seção. Mais precisamente:

- **Ontologia de referência:** descreve as classes e propriedades dos objetos compartilhados na federação. Contém ainda os mapeamentos entre as classes e propriedades equivalentes de cada fonte componente da federação. A ontologia de referência atua como um vocabulário compartilhado e pode ser construída de forma colaborativa, ou seja, quando uma nova fonte de objetos ingressa na federação, novos termos (classes e propriedades) podem ser adicionados à ontologia de referência. A ontologia de referência é armazenada no catálogo.

- **Ontologia local:** descreve as classes e propriedades dos objetos que cada fonte componente da federação deseja compartilhar. As ontologias locais são armazenadas no catálogo. Uma ontologia local representa o esquema de uma fonte componente de uma federação.

- **Instância de referência:** representa, de forma consolidada, uma coleção de objetos, armazenados em fontes distintas, que se referem a um mesmo objeto do mundo real. Cada instância de referência pertence a uma classe da ontologia de referência e participa de mapeamentos que indicam os objetos equivalentes em cada fonte componente.

Para exemplificar a abordagem, considere duas fontes de objetos, utilizadas em sistemas de venda de passagens aéreas de companhias distintas. A Figura 6 (a) representa, em forma tabular, a classe `Aeroporto` e suas propriedades, utilizada pelo sistema da companhia aérea A e a Figura 6 (b) representa a classe `Airport` utilizada pelo sistema da companhia B. Vale salientar que, embora os exemplos sejam fictícios, os dados são reais, evidenciando a ambigüidade existente entre os nomes de aeroportos.

A.Aeroporto

Cod	Nome	Estado
GIG	Galeão, Rio de Janeiro	RJ
SDU	Santos Dumont, Rio de Janeiro	RJ
PHB	Santos Dumont, Parnaíba	PI

(a)

B.Airport

Code	Name	City	Province	Country
GIG	RIO JANEIRO INTL	Rio de Janeiro	Rio de Janeiro	Brazil
SDU	SANTOS DUMONT	Rio de Janeiro	Rio de Janeiro	Brazil
PHB	SANTOS DUMONT	Parnaíba	Piauí	Brazil

(b)

Figura 6 – Exemplo de heterogeneidade dos metadados.

Naturalmente, cada fonte terá sua própria modelagem e até mesmo seus próprios identificadores para os objetos. A classe `Aeroporto` da fonte A, representada na Figura 6 (a), possui três propriedades contendo informações a respeito dos aeroportos. A classe `Airport` da fonte B, representada na Figura 6 (b), possui cinco propriedades para representar cada objeto do tipo `Airport`. Neste exemplo, é visível a heterogeneidade na modelagem dos dados (classes com propriedades distintas). De fato, algumas propriedades de uma classe representam a mesma propriedade em outra classe, porém devido às diferenças sintáticas é evidenciada a heterogeneidade entre os esquemas.

Devido à natureza heterogênea das fontes de objetos, elas podem conter classes estruturalmente heterogêneas, como mostradas na Figura 6. Uma forma de resolver este problema consiste em identificar classes e propriedades equivalentes das diferentes fontes. A Figura 7 ilustra os mapeamentos entre classes e propriedades equivalentes.

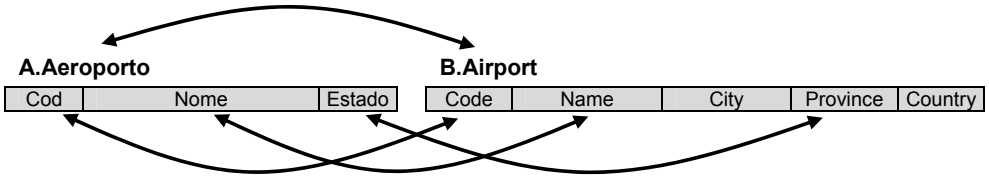

Figura 7 – Mapeamento dos metadados.

Porém, não basta definir mapeamentos apenas entre os metadados (classes e propriedades). Para efetivamente interoperar, ainda é necessário identificar os objetos equivalentes.

Intuitivamente, deve-se selecionar as propriedades que identificam unicamente os objetos equivalentes das diversas fontes. O caso mais simples ocorre quando o identificador dos objetos nas fontes componentes é padrão. Analisando o exemplo da Figura 6, verifica-se que os mesmos códigos são utilizados para identificar os aeroportos em ambas as fontes. Assim é fácil identificar o mesmo aeroporto nas duas fontes.

Porém, em certas situações, pode ser difícil encontrar identificadores padronizados em fontes distintas. Até mesmo dentro de uma única companhia podem existir diferentes identificadores para os mesmos objetos em diferentes contextos. A Figura 8 apresenta um exemplo de heterogeneidade na identificação de objetos. Na classe `Aeroporto` da fonte A, representada na Figura 8 (a), um número seqüencial é usado como identificador dos objetos. Já a classe `Airport` da fonte B, representada na Figura 8 (b), o identificador utilizado é o próprio código do aeroporto.

A.Aeroporto

Cod	Nome	Estado
1	Galeão, Rio de Janeiro	RJ
2	Santos Dumont, Rio de Janeiro	RJ
3	Santos Dumont, Parnaíba	PI

(a)

B.Airport

Code	Name	City	Province	Country
GIG	RIO JANEIRO INTL	Rio de Janeiro	Rio de Janeiro	Brazil
SDU	SANTOS DUMONT	Rio de Janeiro	Rio de Janeiro	Brazil
PHB	SANTOS DUMONT	Parnaíba	Piauí	Brazil

(b)

Figura 8 – Exemplo de heterogeneidade na identificação de objetos.

Neste contexto, torna-se difícil, ou mesmo impossível, mapear os objetos equivalentes de ambas as fontes. Uma possível solução seria criar um mapeamento explícito entre os objetos, como mostra a Figura 9.

Aeroporto_Airport

Cod	Code
1	GIG
2	SDU
3	PHB

Figura 9 – Armazenamento explícito do mapeamento entre objetos equivalentes.

Os mapeamentos entre objetos equivalentes podem ser definidos a partir de uma análise dos metadados das fontes, isto é, da definição das classes e propriedades. Porém, segundo Haas & Carey (2003), a inexistência de metadados é um dos motivos pelos quais as federações de bancos de dados não tiveram sucesso. Além disso, os metadados não garantem a solução completa para a ambigüidade dos termos, pois um mesmo termo pode ser usado em diferentes contextos, em diferentes línguas ou até mesmo de forma errônea. A solução é utilizar artefatos que descrevam as meta-informações das fontes de maneira expressiva, fornecendo meios para descrição das classes, propriedades e dos objetos compartilhados, permitindo ainda a inclusão de informações a respeito da interpretação de seu conteúdo.

Neste contexto, este trabalho propõe a criação de um ambiente que utiliza ontologias para descrição destas meta-informações, bem como das instâncias em comum. As ontologias locais descrevem as classes e propriedades dos objetos a serem compartilhados pelas fontes em uma federação de fontes de objetos. Estas ontologias podem ser vistas como os esquemas das fontes componentes da federação, descritas numa linguagem neutra.

O primeiro passo desta metodologia para viabilizar o compartilhamento de objetos é a identificação clara do domínio de aplicação. Por exemplo, se o

objetivo é a criação de um ambiente integrado para venda de passagens aéreas de diversas companhias, então o domínio de aplicação definirá as entidades envolvidas na negociação de venda de passagens, tais como: companhias aéreas, vôos ofertados, assentos, etc.

Neste contexto, cada fonte precisa descrever as classes de objetos que deseja compartilhar na federação sob forma de uma ontologia local. De forma geral, a ontologia incluirá termos para descrição de objetos, classes e propriedades.

Cada fonte disponibiliza um conjunto de objetos para ser compartilhado com os outros usuários da federação. Os vôos oferecidos pelas companhias aéreas são exemplos de objetos. Cada objeto compartilhado pertence a uma classe. Por exemplo, o vôo ”5528” é um objeto que pertence à classe `Vôo`, assim como o objeto “GIG” pertence à classe `Aeroporto`. Além disso, cada classe de objetos compartilhados possui propriedades que descrevem suas características. Por exemplo, os objetos da classe `Vôo` possuem as seguintes propriedades: aeroporto/cidade/estado/país de origem e de destino, número do vôo, aeronave, preço, etc. Além de representarem os atributos das classes, as propriedades podem definir relacionamentos entre objetos. Por exemplo, a propriedade que define o aeroporto de origem relaciona um objeto da classe `Vôo` com um objeto da classe `Aeroporto`.

Além das ontologias locais, o ambiente utiliza uma ontologia de referência que é criada com base nos termos comuns compartilhados pelas fontes componentes da federação. A ontologia de referência pode ser definida a priori, por uma autoridade central da federação, ou ser construída de forma colaborativa, ou seja, quando uma nova fonte de objetos ingressa na federação, novos termos podem ser adicionados à ontologia de referência.

A Figura 10 mostra de forma abstrata as ontologias O_A e O_B que descrevem respectivamente os termos das fontes de objetos A e B do exemplo corrente. As setas bidirecionais indicam os mapeamentos definidos entre as ontologias locais e a ontologia de referência (O_R).

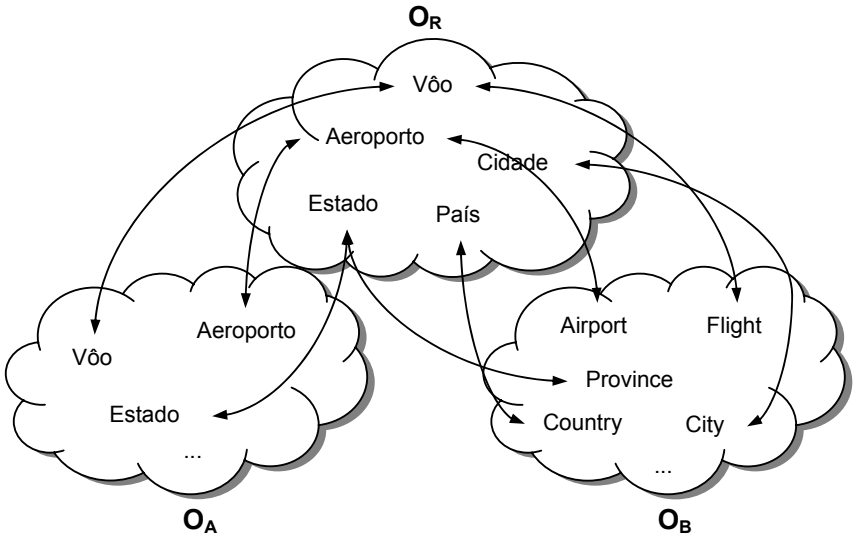

Figura 10 – Representação abstrata das ontologias e mapeamentos entre classes.

A ontologia de referência mantém os termos básicos necessários para atender às necessidades dos usuários, servindo de ponto focal para consultas e acesso aos objetos compartilhados na federação. Além disso, cada termo da ontologia de referência possuirá mapeamentos identificando os termos equivalentes nas ontologias locais que representam as fontes componentes.

Como proposta de solução para a heterogeneidade dos objetos compartilhados na federação, o ambiente inclui um conjunto de objetos de referência, de agora em diante chamado instâncias de referência. Uma instância de referência representa, de forma consolidada, uma coleção de objetos, armazenados em fontes distintas, que se referem a um mesmo objeto do mundo real. Cada instância de referência pertence a uma classe da ontologia de referência e participa de mapeamentos que indicam os objetos equivalentes em cada fonte componente. Deste modo, é possível descobrir os objetos equivalentes das fontes distintas através dos mapeamentos definidos nas instâncias de referência.

A Figura 11 ilustra esquematicamente o uso das instâncias de referência (I_R) e mapeamentos para os objetos de duas fontes componentes (I_A e I_B). As setas bidirecionais indicam os mapeamentos definidos entre os objetos das fontes e as instâncias de referência.

Para facilitar o entendimento dos mapeamentos, nas Figuras 10 e 11 as instâncias de referência são representadas externamente à ontologia de referência. Porém, as instâncias de referência fazem parte da ontologia de referência e, portanto, serão suprimidas das próximas figuras deste trabalho.

Portanto, entende-se ontologia de referência com um conjunto de descrições das classes e propriedades de referência, juntamente com as instâncias de referência e os mapeamentos.

As ontologias locais e a ontologia de referência (incluindo as instâncias de referência) mantidas por este ambiente são armazenadas em um repositório de ontologias (Figura 12), o qual oferece meios para consultá-las.

A seção seguinte descreve a criação das ontologias e dos mapeamentos.

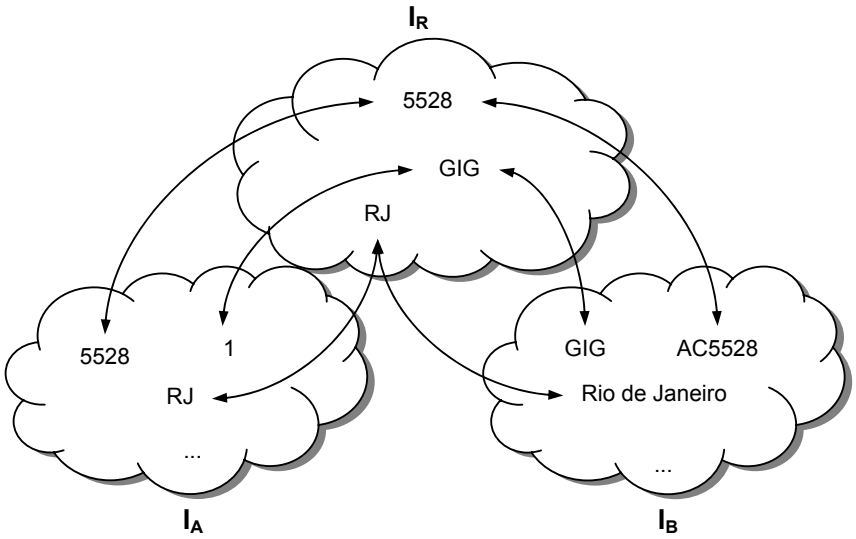

Figura 11 – Representação abstrata das instâncias e mapeamentos.

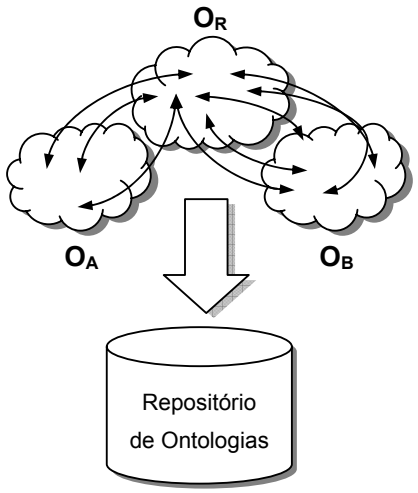

Figura 12 – Repositório de ontologias.

3.2.2
Criação das ontologias e mapeamentos

3.2.2.1
Composição básica das ontologias

Esta seção apresenta os componentes básicos para descrição das ontologias do ambiente em OWL (*Web Ontology Language*). OWL é uma linguagem para descrição de recursos na Web com suporte à descrição de classes, propriedades e objetos. Além disso, permite relacionar estes elementos e impor restrições à sua interpretação (Bechhofer et al., 2004; Smith et al., 2004).

No contexto deste trabalho, a linguagem OWL pode ser entendida como um modelo de dados canônico, padronizando o formato para descrição das fontes componentes da federação.

O primeiro passo para definição da ontologia é a criação do documento com a extensão OWL. Este documento conterá as descrições das classes, propriedades e objetos descritos na linguagem OWL. Este documento da ontologia, bem como os termos especificados nela, são recursos na Web. Para identificar recursos na Web são utilizados identificadores que seguem um sistema uniforme de identificação e são chamados de *Uniform Resource Identifiers*, ou apenas, URIs (Berners-Lee et al., 1998). Um exemplo de URI para uma ontologia é mostrado abaixo:

```
http://localhost/minhaOntologia.owl
```

As URNs (*Universal Resource Name*), um caso particular de URIs, também podem ser usadas para identificar as ontologias e seus termos. Uma URN é um identificador único utilizado para recursos persistentes que não informa a sua localização física. Uma URN define um nome e endereço para objetos persistentes acessíveis através da Web. Basicamente uma URN consiste de três campos separados por dois pontos: "urn" seguido do identificador do namespace e o nome do recurso. A seguir é mostrado um exemplo de uma URN:

```
urn:localhost:minhaOntologia
```

Após criar o documento da ontologia, deve-se inicializar a tag `rdf:RDF`, como pode ser visto na Figura 13. Todas as definições ficarão dentro desta tag.

```
<rdf:RDF>
...
</rdf:RDF>
```

Figura 13 – Tag delimitadora de uma ontologia em OWL..

Em virtude da Web fornecer um ambiente onde os recursos encontram-se distribuídos, a linguagem OWL viabiliza a utilização das fontes de informação de forma distribuída. Por exemplo, uma classe definida em uma ontologia pode ser estendida em outra ontologia. Além disso, podemos criar uma ontologia usando termos já definidos em outros vocabulários (ontologias). Para isso, não é necessário redefini-los ou copiá-los da fonte, basta referenciá-los usando o identificador do termo na Web.

Para usar termos definidos em outras ontologias, é preciso indicar quais ontologias estão sendo usadas através de um conjunto de declarações de XML *namespaces* no cabeçalho do arquivo OWL dentro da tag `rdf:RDF`.

Os namespaces permitem a definição de prefixos para os URIs, simplificando a identificação dos termos. Os endereços declarados para os prefixos dos namespaces fornecem meios de interpretar as URIs absolutas dos termos utilizados na ontologia. Além disso, o uso de namespaces torna a apresentação da ontologia mais legível, substituindo os URIs dos termos pelos prefixos declarados no cabeçalho do documento.

```
<rdf:RDF
    xmlns        = "http://localhost/minhaOntologia#"
    xmlns:minha  = "http://localhost/minhaOntologia#"
    xml:base     = "http://localhost/minhaOntologia#"
    xmlns:owl    = "http://www.w3.org/2002/07/owl#"
    xmlns:rdf    = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs   = "http://www.w3.org/2000/01/rdf-schema#"
    xmlns:xsd    = "http://www.w3.org/2001/XMLSchema#">
...
<owl:Class rdf:ID="Aeroporto"/>
...
</rdf:RDF>
```

Figura 14 – Exemplo de declaração dos namespaces de uma ontologia em OWL.

A Figura 14 mostra como fazer a declaração dos namespaces de uma ontologia na linguagem OWL. Brevemente, temos:

- A primeira cláusula `xmlns` identifica o namespace da ontologia, identificando o endereço padrão que deve ser utilizado para interpretação da URI dos termos definidos na ontologia, é a URI padrão de identificação do documento. Assim, qualquer termo que apareça sem prefixo de namespace deve ser interpretado prefixando o endereço padrão contido nesta declaração.
- A segunda declaração define um prefixo para o namespace padrão, caso algum termo tenha sido definido usando o prefixo.
- A terceira cláusula `xml:base`, identifica a URI que deve ser utilizada para interpretação das URIs relativas presentes na ontologia.
- A quarta declaração, usando a cláusula `xmlns:owl` declara a URI que deve ser usada na interpretação dos termos antecedidos pelo prefixo `owl`. Por exemplo, o termo `owl:Class` é uma cláusula da linguagem OWL para descrição de classes. Ela é definida, originalmente, no documento de especificação da linguagem cuja localização foi definida através do prefixo do namespace `owl`. A URI utilizada será `http://www.w3.org/2002/07/owl#Class`. Este namespace é declarado para permitir a interpretação do vocabulário da linguagem OWL que está definido no endereço `http://www.w3.org/2002/07/owl`.
- Os demais namespaces servem para referenciar outros vocabulários utilizados no documento da ontologia que está sendo definido. Um exemplo disso é a própria linguagem OWL que utiliza alguns termos dependentes de construtores definidos em outros vocabulários, tais como: RDF, RDFS e XML *Schema*. Por isso, os namespaces que referenciam os vocabulários de cada uma destas linguagens também precisam ser declarados, bem como qualquer outro vocabulário utilizado para construção da ontologia.

A Tabela 3 mostra os namespaces mais utilizados, suas URIs e uma breve descrição de seu conteúdo.

Tabela 3 – Namespaces freqüentemente utilizados em ontologias.

NS	URI	Descrição
owl	http://www.w3.org/2002/07/owl#	Vocabulário da linguagem OWL
rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#	Vocabulário da linguagem RDF
rdfs	http://www.w3.org/2000/01/rdf-schema#	Vocabulário da linguagem RDF Schema
xsd	http://www.w3.org/2001/XMLSchema#	Vocabulário da linguagem XML Schema
dc	http://purl.org/dc/elements/1.1/	Dublin Core - vocabulário para descrição de documentos eletrônicos.

A utilização de URIs e namespaces introduzem a técnica ideal para identificação de objetos para compartilhamento entre fontes distribuídas. Vale observar que cada termo (classes, propriedades e instâncias) da ontologia possui uma URI para identificá-lo. A URI carrega informação suficiente para acessar e recuperar dados a respeito do termo. Por exemplo, a URI `http://localhost/aeroportosBrasileiros.owl#GIG` identifica a instância GIG da ontologia de aeroportos brasileiros. Para acessar dados a respeito deste aeroporto ou referenciá-lo em outra fonte basta utilizar essa URI.

No cenário ideal, vislumbra-se organizações responsáveis por classes de objetos e pela padronização dos identificadores (URIs) destes objetos. Um exemplo disso é uma organização mundial responsável pela padronização dos URIs que identificam aeroportos. Este padrão seria compartilhado entre todas as companhias aéreas. O resultado seria que, nos bancos de dados das companhias, ao invés de existirem diferentes identificadores para aeroportos, existam referências aos identificadores através das URIs padronizadas evitando a ambigüidade e permitindo o compartilhamento das informações.

Após a declaração dos namespaces são definidos os metadados do documento usando a tag `owl:Ontology`. Esta cláusula permite descrever meta-informações a respeito do documento tais como: autor, data de criação, versão, comentários, e outras informações. A Figura 15 mostra um exemplo do uso da cláusula `owl:Ontology` usando termos do vocabulário do RDF Schema e do Dublin Core, representados, respectivamente, pelos namespaces `rdfs` e `dc`.

```
...
<owl:Ontology rdf:about="">
    <rdfs:comment>Exemplo de ontologia em OWL</rdfs:comment>
    <dc:creator>Daniela Brauner</dc:creator>
    <dc:date>2004-12-12</dc:date>
...
</owl:Ontology>
...
```

Figura 15 – Exemplo de definição dos metadados do documento OWL.

O atributo `rdf:about` fornece uma referência para a ontologia. Quando vazio, a referência adotada para a ontologia é a URI base definida pela cláusula `xml:base` na declaração dos namespaces.

Por conseguinte, criado o arquivo e inicializadas as definições em OWL da ontologia, partimos para a descrição dos termos (classes, propriedades e objetos).

Uma classe em OWL é descrita utilizando a cláusula `owl:Class`. A classe possui propriedades e é formada por uma coleções de objetos. Os objetos pertencentes a uma classe são chamados de *instâncias* da classe. Em OWL, uma propriedade é uma relação binária, classificada em dois tipos de acordo com as entidades envolvidas:

- *Datatype Property*: relaciona instâncias a valores de dados. Estes valores podem ser literais RDF ou instâncias de tipos de dados do XML Schema;
- *Object Property*: relaciona instâncias de duas classes.

As próximas seções mostram em mais detalhes os construtores da linguagem OWL através de exemplos para construção da ontologia de referência e das ontologias locais.

3.2.2.2 Criação da ontologia de referência

O objetivo de um OnOC é permitir o compartilhamento de objetos em uma federação de fontes de objetos. A base para viabilizar o compartilhamento é a criação de um vocabulário comum, facilitando a comunicação entre os membros da federação. Este vocabulário é a *ontologia de referência*, o componente central do ambiente proposto. Através dela, é possível descobrir os termos que as fontes de objetos componentes da federação possuem em comum.

A ontologia de referência é composta por metadados (classes e propriedades) e instâncias de referência. Basicamente, para compartilhar objetos na federação, a nova fonte de objetos precisa mapear sua ontologia local à ontologia de referência do ambiente, identificando as classes, propriedades e objetos que se relacionam com os termos da ontologia de referência.

Os mapeamentos são definidos através de propriedades, que serão abordadas em detalhes mais adiante nesta dissertação. Para auxiliar no mapeamento dos termos, é importante incluir comentários em cada termo definido na ontologia.

A ontologia de referência pode atuar como um padrão para novas fontes de objetos que estão sendo projetadas. Conseqüentemente, fontes com esquemas que já seguem a ontologia de referência da federação serão interoperáveis.

Para o ambiente proposto, a ontologia de referência pode ser definida a priori ou de forma colaborativa. Ambas abordagens serão descritas a seguir.

3.2.2.2.1 Definição a priori

A ontologia de referência pode ser definida a priori, ou seja, antes da criação da federação. Nesta abordagem um usuário com papel de administrador da federação deve carregar a ontologia de referência no momento da implantação do ambiente de gerenciamento da federação.

Sugere-se que a ontologia seja criada de forma similar a um padrão. Um padrão é definido por uma comunidade de especialistas no domínio. Exemplos de comunidades e organizações padronizadoras são: ISO, OpenGIS, OASIS e W3C. Na definição da ontologia de referência da federação é importante a concordância sobre os termos utilizados nela em uma comunidade de especialistas. Isto porque ela atuará como um vocabulário compartilhado da federação. Se não houver concordância, a federação corre o risco de não possuir fontes de objetos cadastradas.

Para criar a ontologia de referência sugere-se um estudo detalhado das necessidades da comunidade, de forma a capturar os termos mais usados do domínio de aplicação. A idéia é criar uma ontologia expressiva o suficiente para suprir as necessidades da comunidade, contendo um conjunto de classes, propriedades e objetos de uso comum entre diversos usuários do domínio de aplicação.

Para a criação da ontologia pode-se utilizar uma metodologia de desenvolvimento de ontologias dentre as inúmeras já propostas na literatura. Brauner et al. (2003) faz a avaliação de um processo de desenvolvimento de ontologias incluindo um breve resumo de outras técnicas propostas.

Como explicado anteriormente, no cabeçalho de um documento de ontologia, são declarados os namespaces dos vocabulários utilizados. No cabeçalho da ontologia de referência, devem ser adicionados os namespaces para as ontologias locais gerenciadas pelo catálogo da federação. Isto porque os termos definidos nas ontologias locais serão mapeados para os termos da ontologia de referência usando as URIs originais das ontologias locais.

Em seguida, na descrição da tag `owl:Ontology` são declarados os metadados da ontologia de referência. A Figura 16 mostra um exemplo de um documento da ontologia de referência contendo apenas as declarações dos namespaces e os metadados da ontologia.


```
<rdf:RDF
    xmlns           = "http://localhost/OnOC/OR.owl#"
    xmlns:OR        = "http://localhost/OnOC/OR.owl#"
    xml:base        = "http://localhost/OnOC/OR.owl#"
    xmlns:owl       = "http://www.w3.org/2002/07/owl#"
    xmlns:rdf       = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs      = "http://www.w3.org/2000/01/rdf-schema#"
    xmlns:xsd       = "http://www.w3.org/2001/XMLSchema#"
    xmlns:fonte_01  = "http://localhost/Fonte_01.owl#">
<owl:Ontology rdf:about="">
    <rdfs:comment>Exemplo de ontologia de referência</rdfs:comment>
    <dc:creator>Daniela Brauner</dc:creator>
    <dc:date>2004-12-12</dc:date>
    ...
</owl:Ontology>
...
<!-- Definições de classes, propriedades e objetos -->
...
</rdf:RDF>
```

Figura 16 – Exemplo de definição dos metadados da ontologia de referência.

Nesta abordagem, as atualizações da ontologia de referência são feitas apenas pelo administrador do ambiente. Ao surgir a necessidade de inclusão de

um novo termo somente o administrador terá permissão de adicioná-lo à ontologia de referência.

3.2.2.2.2 Construção colaborativa

Outra abordagem para criação da ontologia de referência é a construção de forma colaborativa.

Nesta abordagem, cada fonte componente da federação colabora com um ou mais termos (classes, propriedades e objetos de referência) para criação da ontologia de referência. Os membros da federação definem as regras da federação para atualização da ontologia de referência.

São considerados termos reprovados aqueles que levam a inconsistências na ontologia de referência. Neste caso, ou as novas assertivas (que definem os novos termos) devem ser rejeitadas, ou a ontologia de referência revista.

Quando novos termos passam a fazer parte da ontologia de referência o administrador da federação deve dar permissão de acesso para atualizações nestes termos da ontologia aos usuários responsáveis por cada novo termo.

3.2.2.3 Criação da ontologia local

Esta seção introduz a descrição de uma ontologia local na linguagem OWL para representar uma fonte componente da federação onde os objetos são representados em um banco de dados relacional. Nos limitaremos à descrição das transformações de esquemas no modelo relacional para ontologias em OWL por caracterizar-se a situação mais comum. Porém, tais transformações podem ser facilmente adaptadas para esquemas que seguem outros modelos de dados.

Para fazer parte da federação, cada fonte precisa informar sua ontologia local, contendo a descrição de classes e propriedades da coleção de objetos que deseja compartilhar.

Ao criar o documento OWL indica-se usar o nome original da fonte de dados para identificá-lo. Assim, garante-se a identificação da fonte através do fragmento da URI da ontologia dado pelo nome do documento.

Para mapear um esquema relacional em uma ontologia descrita na linguagem OWL, são propostas algumas regras de transformação e nomenclatura. De fato, tais regras introduzem uma padronização na transformação entre o esquema relacional e uma ontologia descrita em OWL.

A Figura 17 mostra as tabelas do banco de dados relacional utilizado nos exemplos seguintes para demonstração das transformações. Note que os atributos cujos nomes aparecem sublinhados representam as chaves primárias das tabelas. Já o símbolo "#" denota que a restrição de chave estrangeira do atributo.

Aeroporto

Cod	Nome	#Estado
GIG	Galeão, Rio de Janeiro	RJ
SDU	Santos Dumont, Rio de Janeiro	RJ
PHB	Santos Dumont, Parnaiba	PI

(a)

Estado

Cod	Nome
RJ	Rio de Janeiro
PI	Piauí

(b)

Figura 17 – Exemplo de banco de dados relacional.

Descrição de tabelas

As tabelas de um banco de dados relacional são descritas como classes na ontologia em OWL. As classes da ontologia herdam o mesmo nome das tabela que representam. Neste trabalho, os nomes das classes são escritos com a primeira letra maiúscula e as demais minúsculas. A Figura 18 mostra a maneira de descrever classes na linguagem OWL. O exemplo representa as tabelas `Aeroporto` e `Estado` do exemplo corrente (vide Figura 17).

```
<owl:Class rdf:ID="Aeroporto">
   ...
</owl:Class>

<owl:Class rdf:ID="Estado">
   ...
</owl:Class>
```

(a)

```
<owl:Class rdf:ID="Aeroporto"/>

<owl:Class rdf:ID="Estado"/>
```

(b)

Figura 18 – Exemplo de descrição de uma classe em OWL.

Descrição de atributos

Os atributos das tabelas do esquema relacional são transformados em propriedades na ontologia. Neste trabalho, os nomes atribuídos às propriedades são formados a partir da concatenação do nome da classe correspondente à

tabela do banco de dados que representa, do símbolo separador ponto final (“.”) e do nome original do atributo da tabela. Vale salientar a importância de manter o nome da classe junto ao nome do atributo de forma a evitar conflitos de nomes de atributos iguais presentes em tabelas diferentes de um mesmo banco de dados. Os nomes das propriedades são escritos com todas as suas letras minúsculas. Dependendo da restrição do atributo no banco de dados utiliza-se um tipo de propriedade em OWL para descrevê-lo.

Os atributos sem restrição de chave estrangeira são descritos em OWL utilizando a cláusula `owl:DatatypeProperty`. O domínio da propriedade é a classe que representa a tabela de origem do atributo em questão e a imagem é o tipo de dados equivalente ao do atributo. Mais precisamente, os tipos de dados utilizados no modelo relacional são mapeados para tipos de dados especificados no XML *Schema*. Alguns mapeamentos entre os tipos de dados são mostrados na Tabela 4, seguindo a especificação do SQL/XML (SQL/XML, 2004). O SQL/XML é um padrão ISO que oferece suporte ao uso de XML no contexto de um sistema de banco de dados com suporte à linguagem SQL.

Tabela 4 – Mapeamento entre os tipos de dados do SQL para o XML *Schema*.

SQL	XML *Schema*[26]
CHAR	xsd:string
VARCHAR	xsd:string
DECIMAL	xsd:decimal
NUMERIC	xsd:decimal
INTEGER	xsd:integer
SMALLINT	xsd:integer
BIGINT	xsd:integer
FLOAT	xsd:float
REAL	xsd:float
DOUBLE	xsd:double
BOOLEAN	xsd:boolean
DATE	xsd:date
TIME WITHOUT TIME ZONE	xsd:time
TIME WITH TIME ZONE	xsd:time
TIMESTAMP WITHOUT TIME ZONE	xsd:dateTime
TIMESTAMP WITH TIME ZONE	xsd:dateTime

Os atributos com restrição de chave primária também são descritos utilizando a cláusula `owl:DatatypeProperty`, porém com a adição da cláusula

[26] xsd é o namespace do XML *Schema* - http://www.w3.org/2001/XMLSchema#

`owl:InverseFunctionalProperty` em sua descrição. Um exemplo de definição usando a propriedade `owl:InverseFunctionalProperty` pode ser visto na Figura 19 na propriedade `aeroporto.cod` da classe `Aeroporto`.

Em OWL esta cláusula designa a característica funcional inversa de uma propriedade onde, para um mesmo valor de imagem, pode-se afirmar que as entidades participantes do domínio serão iguais, caracterizando a restrição de chave primária de um atributo do modelo relacional.

Se uma propriedade P é caracterizada como funcional inversa, então

$$\text{para todo } x, y \text{ e } z,\ P(x,z) \wedge P(y,z) \Rightarrow x = y$$

Um exemplo da aplicação desta regra pode ser visto a seguir, onde duas instâncias da ontologia de referência possuem "SDU" como valor da propriedade `aeroporto.cod`, que é uma propriedade funcional inversa. A implicação lógica da sentença leva a igualdade das duas instâncias.

$$aeroporto.cod(SDU, \text{“SDU”}) \wedge$$
$$aeroporto.cod(Air\_2, \text{“SDU”})$$
$$\Rightarrow sameAs(SDU, Air\_2)$$

Atributos que representam uma chave primária composta não podem ser expressos em OWL. Por exemplo, para expressar o significado de uma chave primária composta por duas propriedades P e Q, para os objetos x e y, estas propriedades deveriam ter o mesmo valor cada, como mostra a regra abaixo:

$$P(x,z) \wedge P(y,z) \wedge Q(x,w) \wedge Q(y,w) \Rightarrow x = y$$

Para representar esta característica de um esquema relacional é preciso utilizar uma linguagem de especificação de regras como, por exemplo, SWRL (*Semantic Web Rule Language*) (Horrocks et al., 2004). No contexto deste trabalho usaremos SWRL somente para especificação dos mapeamentos, que são descritos mais adiante na seção 3.2.2.4.

Os atributos com restrição de chave estrangeira são descritos utilizando a cláusula `owl:ObjectProperty`. O domínio da propriedade é a classe que representa a tabela na qual o atributo está definido. A imagem da propriedade é

a classe que representa a tabela cuja chave primária é utilizada como chave estrangeira.

A Figura 19 mostra a representação na linguagem OWL das classes e propriedades que representam as tabelas e os atributos do exemplo corrente (vide Figura 17).

```
...
<owl:Class rdf:ID="Aeroporto"/>
<owl:Class rdf:ID="Estado"/>
<owl:DatatypeProperty rdf:ID="aeroporto.cod">
    <rdf:type rdf:resource="owl:InverseFunctionalProperty"/>
    <rdfs:domain rdf:resource="#Aeroporto"/>
    <rdfs:range rdf:resource="xsd:string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="aeroporto.nome">
    <rdfs:domain rdf:resource="#Aeroporto"/>
    <rdfs:range rdf:resource="xsd:string"/>
</owl:DatatypeProperty>
<owl:ObjectProperty rdf:ID="aeroporto.estado">
    <rdfs:domain rdf:resource="#Aeroporto"/>
    <rdfs:range rdf:resource="#Estado"/>
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:ID="estado.cod">
    <rdf:type rdf:resource="owl:FunctionalProperty"/>
    <rdfs:domain rdf:resource="#Estado"/>
    <rdfs:range rdf:resource="xsd:string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="estado.nome">
    <rdfs:domain rdf:resource="#Estado"/>
    <rdfs:range rdf:resource="xsd:string"/>
</owl:DatatypeProperty>
...
```

Figura 19 – Exemplo de representação em OWL de uma tabela e seus atributos.

Descrição de objetos

Os registros das tabelas (objetos) que serão compartilhados na federação são descritos na ontologia como instâncias das classes previamente definidas. Todas as instâncias definidas na ontologia devem possuir um identificador único, definido através do atributo `rdf:ID`. A URI base da ontologia concatenada ao identificador da instância formam a URI de identificação da instância. Cada tupla

de cada tabela de cada fonte de dados será identificada por uma URI. A questão, portanto, é como gerar esta URI.

Uma primeira alternativa é a identificação da instância assumir o valor definido para o atributo chave primária do objeto na fonte. Um exemplo utilizando esta abordagem é a URI `http://localhost/Fonte_01.owl#GIG` que identifica a instância GIG da ontologia que representa uma fonte de dados. O objeto que representa o aeroporto do Galeão é identificado na fonte através da chave primária GIG. Ao ser transformado para a ontologia, a instância é identificada com o valor da chave primária.

Por não explicitar na URI o nome da classe do objeto, esta abordagem não permite a definição de todos os registros da fonte. Como as instâncias ficam todas no mesmo namespace da ontologia, quando ocorrerem valores iguais para atributos chave de tabelas diferentes não será possível representá-los, pois eles terão a mesma URI. A Figura 20 mostra um exemplo utilizando esta abordagem, onde duas instâncias de classes distintas na mesma ontologia utilizam o mesmo identificador. Desta forma a ontologia gerada estaria inconsistente pois existem duas instâncias com a mesma URI, uma instância da classe Estado e outra da classe Funcionário.

```
...
<Estado rdf:ID="1">
    <estado.cod>RJ</estado.cod>
    <estado.nome>Rio de Janeiro</estado.nome>
</Estado>
<Funcionario rdf:ID="1">
    <funcionario.cod>1</funcionario.cod>
    <funcionario.nome>Maria Silva</funcionario.nome>
</Funcionario>
...
```

Figura 20 – Exemplo de identificação usando o identificador original do objeto.

Uma segunda abordagem sugerida é a concatenação do nome da tabela ao valor do atributo identificador. Desta forma, o identificador da instância é formado a partir da concatenação do nome da classe (tabela) com o(s) valor(es) do(s) atributo(s) que compõe(m) a chave primária.

Caso de fato a chave primária seja composta, os valores dos atributos que compõem a chave primária são concatenados para identificar a instância que

representa a tupla da fonte. Os valores dos atributos são concatenados usando como separador o símbolo "_".

Por exemplo, considerando que a chave primária de `Aeroporto` seja composta apenas pelo atributo `aeroporto.cod`, então o objeto GIG seria identificado pela URI `http://localhost/Fonte_01.owl#Aeroporto.GIG`.

A Figura 21 mostra a representação em OWL dos objetos do banco de dados do exemplo utilizado neste capítulo, adotando a abordagem descrita anteriormente para geração dos identificadores das instâncias.

```
...
<Estado rdf:ID="Estado.RJ">
    <estado.cod>RJ</estado.cod>
    <estado.nome>Rio de Janeiro</estado.nome>
</Estado>
<Aeroporto rdf:ID="Aeroporto.GIG">
    <aeroporto.cod>GIG</aeroporto.cod>
    <aeroporto.nome>Galeão, Rio de Janeiro</aeroporto.nome>
    <aeroporto.estado rdf:resource="#Estado.RJ"/>
</Aeroporto>
...
```

Figura 21 – Exemplo de identificação concatenando o nome da tabela ao identificador original do objeto.

Uma terceira abordagem consiste na utilização de vários namespaces para referenciar os esquemas e as instâncias da ontologia separadamente. Esta abordagem é indicada para eliminar o risco de identificações duplicadas dentro da mesma ontologia. Isto é, para fontes onde são utilizados os mesmos valores para identificação de registros diferentes em tabelas distintas, é possível garantir a criação de URIs únicas através desta abordagem.

Nesta abordagem, a ontologia representando o esquema da fonte de dados fica armazenada no namespace contendo a identificação *esquema*. Cada tabela da fonte tem uma ontologia representando suas tuplas (objetos). Cada tabela recebe um namespace contendo o nome da própria tabela. As instâncias de cada classe são descritas juntamente com a ontologia que representa a tabela. Assim, poderia ser utilizado o valor do atributo identificador do registro como identificador da instância nas ontologias locais, ou seja, o valor do atributo chave primária, concatenado a URI base de cada ontologia.

A Figura 22 mostra duas ontologias como exemplo desta abordagem. A Figura 22 (a) mostra a ontologia que representa as instâncias da classe `Estado` da ontologia da `Fonte_01`. Já a Figura 22 (b) mostra a ontologia contendo as definições das instâncias da classe `Aeroporto` onde a ontologia de estados é referenciada utilizando o namespace identificado pelo prefixo `estado`.

Nesta abordagem, na ontologia mostrada na Figura 22 (b), o objeto `GIG` da classe `Aeroporto` da fonte de objetos `Fonte_01` é representado pela URI: `http://localhost/Fonte_01/Aeroporto/Aeroporto.owl#GIG`.

```
<rdf:RDF
xmlns:base = "http://localhost/Fonte_01/Estado/Estado.owl#"
xmlns:Fonte_01 = "http://localhost/Fonte_01/esquema/Fonte_01.owl#">
...
<Fonte_01:Estado rdf:ID="RJ">
    <Fonte_01:estado.cod>RJ</Fonte_01:estado.cod>
    <Fonte_01:estado.nome>Rio de Janeiro</Fonte_01:estado.nome>
</Fonte_01:Estado>
...
```

(a)

```
<rdf:RDF
...
xmlns:base = "http://localhost/Fonte_01/Aeroporto/Aeroporto.owl#"
xmlns:Fonte_01 = "http://localhost/Fonte_01/esquema/Fonte_01.owl#"
xmlns:Estado = "http://localhost/Fonte_01/Estado/Estado.owl#">
...
<Fonte_01:Aeroporto rdf:ID="GIG">
    <Fonte_01:aeroporto.cod>GIG</Fonte_01:aeroporto.cod>
    <Fonte_01:aeroporto.nome>Galeão, Rio de Janeiro
    </Fonte_01:aeroporto.nome>
    <Fonte_01:aeroporto.estado rdf:resource="Estado:RJ"/>
</Fonte_01:Aeroporto>
...
```

(b)

Figura 22 – Exemplo de identificação utilizando uma estrutura de diretórios.

Uma outra abordagem para geração de URIs para as instâncias é a utilização de um algoritmo para geração de chaves únicas. Desta forma, todas as instâncias poderiam ser mantidas dentro do mesmo namespace da ontologia.

As regras descritas nesta seção são a base para transformações de esquemas relacionais em OWL. Vale salientar que mapear uma fonte de objetos

para uma ontologia em OWL não é o mesmo que gerar um documento XML contendo os dados de um banco de dados. No caso discutido nesta seção, a questão de nomeação de tuplas por URIs é mais sofisticado. A diferença está na necessidade de identificação única de cada termo definido na ontologia, onde cada instância representa um recurso identificado pela sua URI.

3.2.2.4 Definição dos mapeamentos

Os mapeamentos definidos entre as ontologias locais e de ontologia de referência são criados para auxiliar o compartilhamento de objetos na federação, viabilizando a localização das fontes que possuem informações relevantes para atender às consultas requisitadas ao ambiente, entre outros casos de uso.

No caso do ambiente utilizar a abordagem de construção da ontologia de referência a priori, estes mapeamentos são definidos pelos usuários responsáveis por cada fonte componente no momento em que esta fonte passa a fazer parte da federação. No caso do ambiente utilizar a abordagem de construção colaborativa da ontologia de referência, este mapeamento é definido no momento em que um novo termo passa a fazer parte da ontologia.

O primeiro passo é verificar a possibilidade de definir um mapeamento entre os objetos da fonte e os objetos de referência. Para fazer parte da federação, o usuário responsável pela fonte deverá cadastrá-la no ambiente. O ambiente fornecerá ao usuário meios de definir os relacionamentos para cada um de seus objetos com a ontologia de referência. A quantidade de objetos e falta de similaridade na descrição dos objetos indicarão a viabilidade de realizar ou não estes mapeamentos manualmente ou via regras de inferência.

A falta de similaridade na descrição dos objetos acontece quando os objetos não possuem nenhuma propriedade equivalente, por exemplo, nomes ou identificadores iguais que possam vir a ser comparados. Por outro lado, se os objetos apresentam características comparáveis, então eles possuem similaridade na descrição dos objetos, permitindo o mapeamento manual ou a criação de regras que infiram os mapeamentos. Um exemplo de objetos com descrições similares é mostrado pela regra abaixo:

$$B{:}Airport(x) \wedge B{:}airport.code(x,z) \wedge$$
$$OR{:}Aeroporto(y) \wedge OR{:}aeroporto.cod(y,z)$$
$$\Rightarrow sameAs(x,y)$$

A regra de mapeamento descrita acima diz que um objeto **x** da classe `Airport` da ontologia **B** que possua **z** como valor da propriedade `airport.code` será igual a um objeto **y** da classe `Aeroporto` da ontologia de referência (OR) que também possua **z** como valor da propriedade `aeroporto.cod`.

Se a quantidade de objetos for pequena e não existirem descrições similares, o usuário deve optar pelo mapeamento manual, pois seria difícil e até mesmo impossível a definição de regras.

Porém, se a quantidade de objetos for grande, o usuário pode especificar regras para definição de seus mapeamentos de forma automática. Para isso, o módulo de inferência do ambiente interpreta as regras inferindo os mapeamentos entre os objetos da fonte e os objetos de referência.

Os mapeamentos entre objetos podem ser definidos usando a propriedade `owl:sameAs` que declara a igualdade entre duas instâncias. A Figura 23 mostra um exemplo usando a propriedade `owl:sameAs` para relacionar a instância de referência SDU aos objetos das fontes A e B.

```
<Aeroporto rdf:ID="SDU" />
    ...
    <owl:sameAs rdf:resource="A:2" />
    <owl:sameAs rdf:resource="B:SDU" />
</Aeroporto>
```

Figura 23 – Exemplo de descrição de igualdade entre instâncias em OWL.

Este relacionamento é útil no momento que o cliente faz uma consulta solicitando informações a respeito do aeroporto identificado por SDU. Além da resposta incluir as informações contidas na ontologia de referência, conterá também as URIs originais dos objetos equivalentes nas ontologias A e B indicando que estas URIs possuem mais informações a respeito deles.

Após definir os mapeamentos entre os objetos, o segundo passo é criar os relacionamentos entre as classes e propriedades da ontologia local com as classes e propriedades da ontologia de referência. Para isso, são utilizadas propriedades definidas na linguagem OWL, tais como: `owl:equivalentClass` e `rdfs:subClassOf`.

A propriedade `owl:equivalentClass` é usada para indicar a equivalência entre duas classes de ontologias distintas. A propriedade indica que ambas as classes possuem precisamente o mesmo conjunto de instâncias.

A Figura 24 mostra um exemplo usando o construtor `owl:equivalentClass` relacionando classes com namespaces diferentes.

```
<owl:Class rdf:ID="A:Aeroporto" />
    <owl:equivalentClass rdf:resource="B:Airport" />
</owl:Class>
```

Figura 24 – Exemplo de descrição de equivalência entre classes em OWL.

A propriedade `rdfs:subClassOf` representa o construtor taxonômico fundamental da linguagem RDFS (RDF *Schema*), também usado na linguagem OWL. Esta propriedade relaciona uma classe mais específica com outra mais geral: se **x** é subclasse de **y**, então toda instância de **x** também é instância de **y**. Além disso, a relação é transitiva: se **x** é subclasse de **y** e **y** subclasse de **z**, então **x** é subclasse de **z**. Um exemplo do uso do construtor `rdfs:subClassOf` pode ser visto na Figura 25.

```
...
<owl:Class rdf:ID="Voo" />
<owl:Class rdf:ID="VooInternacional">
    <rdfs:subClassOf rdf:resource="#Voo" />
    ...
</owl:Class>
...
```

Figura 25 – Exemplo de descrição de uma subclasse em OWL.

Um exemplo prático de uso do relacionamento de subclasse é numa consulta por vôos quando o ambiente estiver atuando como mediador para as fontes federadas. Ao receber a consulta, o ambiente verifica na ontologia de referência quais as fontes possuem informações sobre vôos. Ao encontrar uma classe que é subclasse de vôo o ambiente inclui suas instâncias na resposta à consulta, sabendo que esta classe possui objetos que também são vôos.

Além de definir relacionamentos com classes nomeadas na ontologia, as propriedades `rdfs:subClassOf` e `owl:equivalentClass` podem ser usadas para definir relacionamentos com classes definidas por expressões.

Estas expressões definem características que os objetos que são instâncias desta classe devem atender. Em OWL é possível definir restrições nas imagens das propriedades como mostra o exemplo a seguir.

A classe `VooNacional` é definida através de uma expressão. A expressão define que a classe `VooNacional` é equivalente à classe das instâncias de vôos cujas origem e destino são em aeroportos brasileiros. Para isso, usamos Lógica de Descrição (Baader et al., 2003) que é facilmente traduzida para OWL para definir a seguinte expressão:

VooNacional ≡
∃ voo.origem (∃ aeroporto.estado (∃ estado.pais {Brasil})) ∩
∃ voo.destino (∃ aeroporto.estado (∃ estado.pais {Brasil}))

Porém, existe uma grande diferença no uso das propriedades `rdfs:subClassOf` e `owl:equivalentClass` para definição das restrições de classes. Usando `rdfs:subClassOf` os vôos nacionais podem não incluir todos aqueles que têm origem e destino em aeroportos brasileiros. Porém, usando `owl:equivalentClass`, um vôo é uma instância da classe `VooNacional` se, e somente se, o vôo tiver origem e destino num aeroporto brasileiro. As implicações podem ser vistas na Tabela 5.

Tabela 5 – Implicações das propriedades rdfs:subClassOf e owl:equivalentClass.

Propriedade	Implicação
rdfs:subClassOf	VooNacional(?v) ⇒ Voo(?v) ∧ ∃?a (Aeroporto(?a) ∧ voo.origem(?v,?a) ∧ ∃?x (Estado(?x) ∧ aeroporto.estado(?a, ?x) ∧ estado.pais(?x, Brasil))) ∧ ∃?b (Aeroporto(?b) ∧ voo.destino(?v,?b) ∧ ∃?w Estado(?w) ∧ aeroporto.estado(?b, ?w) ∧ estado.pais(?w, Brasil)))
owl:equivalentClass	VooNacional(?v) ⇔ Voo(?v) ∧ ∃?a (Aeroporto(?a) ∧ voo.origem(?v,?a) ∧ ∃?x (Estado(?x) ∧ aeroporto.estado(?a, ?x) ∧ estado.pais(?x, Brasil))) ∧ ∃?b (Aeroporto(?b) ∧ voo.destino(?v,?b) ∧ ∃?w Estado(?w) ∧ aeroporto.estado(?b, ?w) ∧ estado.pais(?w, Brasil)))

Caso não seja possível definir os mapeamentos entre os objetos usando as cláusulas da linguagem OWL, são definidas regras para expressar os relacionamentos que se deseja. Neste trabalho as regras de mapeamento são descritas na linguagem SWRL (*Semantic Web Rule Language*) devido à sua proximidade com a linguagem OWL. A SWRL é uma linguagem para especificação de regras baseada na combinação das sub-linguagens OWL-DL e

OWL Lite com a sub-linguagem Unary/Binary Datalog RuleML (Horrocks et al., 2004).

As regras em SWRL possuem a forma de uma implicação lógica com um antecedente (corpo) e um conseqüente (cabeça). Para facilitar o entendimento os exemplos são mostrados usando uma sintaxe simples, na seguinte forma:

$$antecedente \Rightarrow conseqüente$$

Tanto o antecedente quanto o conseqüente podem ser conjunções de átomos na forma $a_1 \wedge ... \wedge a_2$. As variáveis são apresentadas antecedidas por um ponto de interrogação.

Uma regra em SWRL é interpretada da seguinte maneira: se as condições especificadas no antecedente forem verdadeiras, então as condições especificadas no conseqüente também serão verdadeiras.

A regra abaixo indica que um vôo com origem e destino em aeroportos localizados no mesmo país é um vôo nacional. A Figura 26 mostra a mesma regra descrita na sintaxe RDF/XML de SWRL.

$$Voo(?x) \wedge Aeroporto(?y) \wedge Aeroporto(?z) \wedge$$
$$Estado(?v) \wedge Estado(?u) \wedge Pais(?w) \wedge$$
$$voo.origem(?x,?y) \wedge voo.destino(?x,?z) \wedge$$
$$aeroporto.estado(?y,?v) \wedge aeroporto.estado(?z,?u) \wedge$$
$$estado.pais(?v,?w) \wedge estado.pais(?u,?w)$$
$$\Rightarrow VooNacional(?x)$$

```
...
<swrl:Variable rdf:ID="u"/>
<swrl:Variable rdf:ID="v"/>
<swrl:Variable rdf:ID="w"/>
<swrl:Variable rdf:ID="x"/>
<swrl:Variable rdf:ID="y"/>
<swrl:Variable rdf:ID="z"/>
<ruleml:Imp>
    <ruleml:_body rdf:parseType="Collection">
        <swrl:ClassAtom>
            <swrl:classPredicate rdf:resource="Aeroporto" />
            <swrl:argument1 rdf:resource="#y"/>
        </swrl:ClassAtom>
        <swrl:ClassAtom>
```

```
            <swrl:classPredicate rdf:resource="Aeroporto" />
            <swrl:argument1 rdf:resource="#z"/>
        </swrl:ClassAtom>
        <swrl:ClassAtom>
            <swrl:classPredicate rdf:resource="Estado" />
            <swrl:argument1 rdf:resource="#v"/>
        </swrl:ClassAtom>
        <swrl:ClassAtom>
            <swrl:classPredicate rdf:resource="Estado" />
            <swrl:argument1 rdf:resource="#u"/>
        </swrl:ClassAtom>
        <swrl:ClassAtom>
            <swrl:classPredicate rdf:resource="Pais" />
            <swrl:argument1 rdf:resource="#w"/>
        </swrl:ClassAtom>
        <swrl:ClassAtom>
            <swrl:classPredicate rdf:resource="Voo" />
            <swrl:argument1 rdf:resource="#x"/>
        </swrl:ClassAtom>
        <swrl:IndividualPropertyAtom>
            <swrl:propertyPredicate rdf:resource="voo.origem" />
            <swrl:argument1 rdf:resource="#x"/>
            <swrl:argument2 rdf:resource="#y"/>
        </swrl:IndividualPropertyAtom>
        <swrl:IndividualPropertyAtom>
            <swrl:propertyPredicate rdf:resource="voo.destino" />
            <swrl:argument1 rdf:resource="#x"/>
            <swrl:argument2 rdf:resource="#z"/>
        </swrl:IndividualPropertyAtom>
        <swrl:IndividualPropertyAtom>
            <swrl:propertyPredicate rdf:resource="aeroporto.estado" />
            <swrl:argument1 rdf:resource="#y"/>
            <swrl:argument2 rdf:resource="#v"/>
        </swrl:IndividualPropertyAtom>
        <swrl:IndividualPropertyAtom>
            <swrl:propertyPredicate rdf:resource="aeroporto.estado" />
            <swrl:argument1 rdf:resource="#z"/>
            <swrl:argument2 rdf:resource="#u"/>
        </swrl:IndividualPropertyAtom>
        <swrl:IndividualPropertyAtom>
            <swrl:propertyPredicate rdf:resource="estado.pais" />
            <swrl:argument1 rdf:resource="#v"/>
            <swrl:argument2 rdf:resource="#w"/>
        </swrl:IndividualPropertyAtom>
        <swrl:IndividualPropertyAtom>
```

```
            <swrl:propertyPredicate rdf:resource="estado.pais" />
            <swrl:argument1 rdf:resource="#u"/>
            <swrl:argument2 rdf:resource="#w"/>
        </swrl:IndividualPropertyAtom>
    </ruleml:_body>
    <ruleml:_head rdf:parseType="Collection">
        <swrl:ClassAtom>
            <swrl:classPredicate rdf:resource="VooNacional" />
            <swrl:argument1 rdf:resource="#x"/>
        </swrl:ClassAtom>
    </ruleml:_head>
</ruleml:imp>
...
```

Figura 26 – Exemplo de regra em SWRL para restrição.

Por conseguinte, a expressão abaixo mostra a definição de um mapeamento de igualdade entre uma instância **x** de uma ontologia local **A** e uma instância **y** da ontologia de referência. Ambas as ontologias, local e de referência, são identificadas pelos seus namespaces: *localOntologyA* e *referenceOntology*, respectivamente. A regra afirma a igualdade através da cláusula *owl:sameAs* entre as instâncias da classe *Airport* da ontologia local **A** que possuírem o valor do atributo *airport.code* igual ao valor do atributo *aeroporto.cod* de uma instância de referência da classe *Aeroporto* da ontologia de referência.

$$localOntologyA:Airport(?x) \wedge$$
$$referenceOntology:Aeroporto(?y) \wedge$$
$$localOntologyA:airport.code(?z) \wedge$$
$$referenceOntology:aeroporto.cod(?z)$$
$$\Rightarrow owl:sameAs(?x, ?y)$$

Um exemplo da regra em SWRL para definição do mapeamento definido acima pode ser vista na Figura 27.

```
...
<swrl:Variable rdf:ID="x"/>
<swrl:Variable rdf:ID="y"/>
<swrl:Variable rdf:ID="z"/>
<ruleml:Imp>
    <ruleml:_body rdf:parseType="Collection">
```

```
        <swrl:ClassAtom>
            <swrl:classPredicate rdf:resource="localOntologyA:airport"/>
            <swrl:argument1 rdf:resource="#x"/>
        </swrl:ClassAtom>
        <swrl:ClassAtom>
            <swrl:classPredicate
                        rdf:resource="referenceOntology:aeroporto"/>
            <swrl:argument1 rdf:resource="#y"/>
        </swrl:ClassAtom>
        <swrl:IndividualPropertyAtom>
            <swrl:propertyPredicate
                       rdf:resource="localOntologyA:airport.code" />
            <swrl:argument1 rdf:resource="#x"/>
            <swrl:argument2 rdf:resource="#z"/>
        </swrl:IndividualPropertyAtom>
        <swrl:IndividualPropertyAtom>
            <swrl:propertyPredicate
                     rdf:resource="referenceOntology:aeroporto.cod" />
            <swrl:argument1 rdf:resource="#y"/>
            <swrl:argument2 rdf:resource="#z"/>
        </swrl:IndividualPropertyAtom>
    </ruleml:_body>
    <ruleml:_head rdf:parseType="Collection">
        <swrl:ClassAtom>
            <swrl:classPredicate rdf:resource="owl:sameAs" />
            <swrl:argument1 rdf:resource="#x"/>
            <swrl:argument2 rdf:resource="#y"/>
        </swrl:ClassAtom>
    </ruleml:_head>
</ruleml:imp>
...
```

Figura 27 – Exemplo de regra em SWRL para mapeamento.

3.2.3 Criação da federação

Esta seção descreve como a federação é criada usando o ambiente proposto neste trabalho.

A federação será gerenciada pelo catálogo de objetos baseado em ontologias (*Ontology-based Object Catalog* - OnOC), discutido em detalhes na próxima seção. O catálogo atua como ponto focal da federação e deve ser mantido por um usuário administrador do ambiente. O administrador precisa

tomar algumas decisões antes da criação da federação. A primeira delas é quanto ao tipo de estratégia que será usada para criação da ontologia de referência de forma a configurar as permissões do ambiente, conforme discutido anteriormente.

Para fazer parte da federação a nova fonte de objetos precisa seguir o protocolo de participação do OnOC. O primeiro passo consiste em registrar a nova fonte no OnOC. Para isso, é preciso que o usuário responsável pela fonte cadastre-se no ambiente e forneça a ontologia que descreve os objetos que deseja compartilhar. Esta ontologia será armazenada localmente ao OnOC.

O segundo passo consiste na definição dos mapeamentos indicando os relacionamentos existentes entre os objetos da fonte e as instâncias de referência do OnOC. Caso não seja possível definir estes mapeamentos, o responsável pela fonte precisa definir regras para que o ambiente possa inferir estes mapeamentos.

O terceiro passo consiste na definição dos mapeamentos entre a ontologia local e a ontologia de referência. Estes mapeamentos são definidos com base nos relacionamentos entre as instâncias das classes.

Para entender melhor o funcionamento do ambiente, a próxima seção introduz o conceito do catálogo de objetos baseado em ontologias.

3.3
Catálogo de Objetos baseado em Ontologias (OnOC)

3.3.1
Conceito

Este trabalho visa o compartilhamento de objetos em uma federação de fontes de objetos heterogêneas. A metodologia introduzida na seção anterior indica como criar as ontologias que dão suporte ao ambiente introduzido nesta seção.

O catálogo de objetos baseado em ontologias (*Ontology-based Object Catalog* - OnOC) atua como ponto focal de uma federação de fontes de objetos independentes, fornecendo um ambiente com serviços de consolidação para os objetos representados nas diversas fontes componentes da federação, e serviços de acesso para consulta a objetos e metadados das fontes componentes.

O catálogo mantém e gerencia as ontologias e o acesso aos objetos da federação, fornecendo serviços aos usuários para acesso aos metadados e aos

objetos disponíveis nas fontes de objetos componentes. Além disso, mantém os mapeamentos necessários para identificar os objetos e os metadados (classes e propriedades) equivalentes entre as fontes componentes. Assim, entre outras situações de uso, o catálogo atua como um mediador generalizado para a federação de fontes de objetos, fornecendo serviços de acesso e busca aos objetos e metadados federados.

Como pode ser visto na Figura 28, o OnOC atende requisições provenientes de seus clientes. Os clientes de OnOCs podem ser classificados em três tipos distintos, de acordo com sua atuação no ambiente: aplicações clientes, aplicações participantes e usuário administrador.

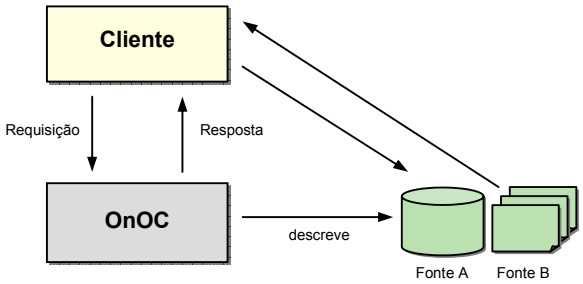

Figura 28 – Diagrama de interação entre o OnOC, o cliente e as fontes de dados.

São chamadas de *aplicações clientes* as aplicações que requisitam consultas ao catálogo. Exemplos destas aplicações podem ser tanto aquelas que fornecem interface com usuários humanos para manipular as informações do catálogo quanto agentes de software que fazem uso do catálogo para padronizar o vocabulário de comunicação entre eles. Entre outras, consultas a metadados e consultas a objetos compartilhados na federação são exemplos de consultas realizadas pelas aplicações clientes.

São chamados de *aplicações participantes* os clientes que possuem uma fonte de objetos cadastrada no ambiente para compartilhar objetos na federação. Uma aplicação participante é responsável por uma ou mais fonte de objetos componente da federação.

Além disso, existe o cliente chamado *usuário administrador* ou apenas *administrador*. O administrador é o cliente responsável por gerenciar o catálogo. Entre suas tarefas estão gerenciar as permissões dos demais usuários no acesso às ontologias mantidas pelo catálogo e criar e atualizar a ontologia de referência.

Como foi visto anteriormente, um OnOC pode ser acessado por agentes de software. Agentes são definidos como entidades computacionais autônomas

desenvolvidas para resolver problemas e capazes de operar efetivamente em ambientes abertos e dinâmicos (Luck et al., 2003). Na maioria das vezes, os agentes são implantados em ambientes para interagir e cooperar com outros agentes (incluindo tanto pessoas quanto software). Estes ambientes são conhecidos como sistemas multi-agentes. O modelo de comunicação de agentes definido pela FIPA (2000) especifica que, para se comunicar, dois agentes precisam compartilhar uma ontologia comum no domínio de discurso que desejam conversar. Desta forma, é possível garantir a mesma interpretação para os termos utilizados nas mensagens trocadas pelos agentes.

Berger & Kessler (2003) argumentam que, em sistemas multi-agentes, o uso de ontologias pode ir além da capacidade de acessar e processar o conteúdo semântico destas informações. A forma mais popular de comunicação entre agentes é através do uso de ontologias explicitamente especificadas e compartilhadas. Assim, dois agentes podem se comunicar utilizando um conhecimento comum compartilhado a respeito do domínio em que atuam.

Neste contexto, um OnOC pode ser utilizado para auxiliar a comunicação entre agentes, tornando a ontologia de referência do catálogo o vocabulário compartilhado da comunidade de agentes. Cada agente terá sua ontologia local cadastrada e mapeada no catálogo, representando o vocabulário do agente. Ao trocar mensagens, os agentes podem usar o OnOC de duas formas distintas.

A primeira é quando um agente recebe uma mensagem num vocabulário desconhecido e precisa traduzi-la. Neste contexto, o agente receptor traduz a mensagem com o auxílio da ontologia de referência do catálogo. Isto é, a partir dos termos utilizados na mensagem, o agente descobre o termo equivalente da ontologia de referência e, em seguida, o termo equivalente na sua ontologia local.

A segunda forma consiste em traduzir a mensagem antes de enviá-la. Neste contexto, o agente emissor busca na ontologia de referência os termos equivalentes aos termos que ele utilizará na mensagem. Assim, o agente receptor só precisa traduzir os termos da ontologia de referência para a sua ontologia local.

O OnOC é similar ao conceito de gazetteer (Atkinson & Fitzke, 2002). Originalmente, um gazetteer é um dicionário de locais geográficos. Cada instância de um serviço de gazetteer cobre tipicamente uma região limitada, como um país, e tem um vocabulário associado de identificadores dos geo-objetos. Um gazetteer fornece operações para localizar e recuperar descrições de geo-objetos incluindo suas localizações espaciais.

Como uma generalização de um gazetteer, um OnOC armazenará instâncias de referência com atributos básicos (identificadores padrão e alternativo, nomes e descrições) cobrindo o domínio de aplicação em questão. O OnOC fornecerá serviços para recuperação destes objetos com base nesses atributos.

Outro conceito similar ao OnOC é o serviço de catálogo de metadados (*Metadata Catalog Service* - MCS) desenvolvido no contexto da arquitetura de Grid Computacional (Singh et al., 2003). Em particular, Tuchinda et al. (2004) descreve Artemis, um sistema de planejamento e formulação de consultas que viabiliza aos usuários consultar catálogos de metadados num Grid. O Artemis inclui um mediador de consultas baseado em técnicas de planejamento que atualiza dinamicamente seu modelo de domínio, e um sistema de formulação de consultas baseado em ontologia que auxilia os usuários a criar e refinar suas consultas interativamente, baseado no conteúdo dos repositórios.

3.3.2 Situações de uso

Por ser um catálogo de objetos, um OnOC pode ser usado em vários contextos onde seja necessária a catalogação dos itens armazenados em fontes de dados distintas.

Para as situações de uso descritas nesta seção, considere o mesmo domínio de aplicação do sistema integrado de venda de passagens aéreas utilizado nos exemplos da seção 3.2.

3.3.2.1 Catálogo de metadados

Um OnOC pode ser utilizado como catálogo de metadados, fornecendo serviços para localização de esquemas e de fontes de objetos que possuam informações a respeito de determinadas classes e propriedades.

Considere um usuário interessado em descobrir quais fontes de objetos possuem informações a respeito de Aeroportos. Com base na ontologia de referência, o usuário solicita ao OnOC a descrição da classe “Aeroporto”. O OnOC busca a descrição da classe solicitada incluindo as definições das classes relacionadas à classe Aeroporto presentes na sua ontologia de referência. Esta descrição contém as propriedades da classe Aeroporto bem como as referências

para as ontologias locais que possuem informações que estejam relacionadas a aeroportos, seja este relacionamento de equivalência, subclasse ou outro.

Ainda neste contexto, o catálogo de metadados também pode ser usado como inspiração para modelagem de dados de um novo esquema.

3.3.2.2
Catálogo de instâncias de referência

Similar ao conceito de gazetteer, um OnOC pode atuar como catálogo de instâncias de referência. Neste caso, o OnOC deve fornecer acesso apenas a um conjunto mínimo de informação sobre os objetos da federação representados pelas instâncias de referência.

Considere um usuário interessado em obter as informações mínimas a respeito de Aeroportos brasileiros. Ao atuar como um catálogo de instâncias, através de uma consulta às instâncias de referência da classe Aeroporto localizados no Brasil, o OnOC retorna as informações básicas sobre os aeroportos que possui em seu conjunto de instâncias de referência.

3.3.2.3
Mediador para acesso a objetos distribuídos

O OnOC pode ser usado como mediador para acesso a objetos armazenados em fontes fisicamente distribuídas. Seu objetivo é processar consultas diretamente nas fontes, utilizando a ontologia de referência da federação para localizar as ontologias locais das fontes que possuem objetos que atendem a consulta do cliente.

Considere um usuário que deseja descobrir informações a respeito do aeroporto cuja identificação é dada sigla “GIG”.

Para isso, o usuário faz uma requisição ao catálogo passando as seguintes informações: “GIG” como identificador do objeto e “Aeroporto” como sua classe. O catálogo busca nas instâncias de referência o objeto da classe “Aeroporto” cujo identificador seja “GIG”. A resposta desta busca é o objeto com seus valores de propriedades, definidos na ontologia de referência do catálogo, incluindo as informações contidas nas fontes que possuem mais informações a respeito dele.

Atuando como mediador o catálogo consulta fontes distribuídas em busca de informações que atendam às consultas dos clientes. Isto é possível através dos mapeamentos entre as ontologias locais e a ontologia de referência que

estão no catálogo. Os mapeamentos precisam ser bidirecionais permitindo que uma consulta enviada ao catálogo possa ser traduzida numa consulta à ontologia local da fonte.

3.3.3 Requisitos

Esta seção descreve os requisitos de um OnOC de forma a atender o seu propósito cobrindo as situações de uso descritas na seção anterior.

Entre os requisitos de alto-nível assumimos que um OnOC deve:

(R1) **oferecer serviços de cadastramento de clientes:** um OnOC deve permitir o cadastro de seus clientes. Estes clientes podem ser de três tipos: aplicações clientes, aplicações participantes e usuário administrador. As aplicações clientes são aquelas que farão apenas consultas ao catálogo. As aplicações participantes são aqueles clientes que compartilham objetos no ambiente. O usuário administrador pode ser um ou mais usuários que administram o OnOC.

(R2) **oferecer serviços de gerência de permissões de acesso:** um OnOC deve permitir ao administrador definir as permissões de acesso das aplicações clientes e das aplicações participantes às ontologias locais e à ontologia de referência. Caso a ontologia de referência seja criada a priori, as aplicações participantes terão permissão de escrita e leitura para as suas ontologias locais e apenas a permissão de leitura na ontologia de referência, Caso a ontologia de referência seja criada de forma colaborativa, algumas aplicações participantes terão permissão de escrita. Os demais clientes terão apenas permissão de leitura em todas as ontologias de um OnOC.

(R3) **oferecer serviço de autenticação e controle de acesso:** um OnOC deve autenticar as aplicações clientes, as aplicações participantes e o usuário administrador, e controlar o acesso aos seus recursos de acordo com as permissões de cada cliente.

(R4) **oferecer serviços de gerência de ontologias:** para viabilizar o compartilhamento de objetos entre as fontes componentes da

federação, um OnOC deve oferecer meios para gerência de metadados e instâncias de referência, incluindo armazenamento, consulta e atualização. Um OnOC deve disponibilizar meios das aplicações participantes incluirem e atualizarem suas ontologias locais, bem como definirem o mapeamento entre a ontologia local e a ontologia de referência. Um OnOC deve oferecer ainda meios do administrador incluir e atualizar a ontologia de referência, bem como dar a permissão de acesso aos usuários para incluir termos na ontologia de referência se a federação optar pela construção colaborativa da ontologia de referência.

(R5) oferecer serviços de consulta aos metadados: um OnOC deve permitir consultas às ontologias locais (metadados das fontes) e à ontologia de referência. Estes serviços devem viabilizar consultas por classes de uma ontologia, propriedades de uma classe, e descrição de todas as classes e propriedades de uma ontologia. Além disso, um OnOC deve ser capaz de informar as fontes que possuem informações a respeito de uma classe.

(R6) oferecer serviços de consulta às instâncias de referência: um OnOC de permitir consultas às instâncias definidas na ontologia de referência. Assim, as aplicações (clientes ou participantes) podem obter as informações básicas a respeito dos objetos compartilhados. Além disso, deve ser capaz de informar a fonte componente da federação que também possui informações a respeito da(s) instância(s) consultada(s).

(R7) oferecer serviços de consulta aos objetos das fontes: quando estiver atuando como mediador, um OnOC deve fornecer serviços para consulta aos objetos mantidos pelas aplicações participantes, ou seja, às fontes federadas. As consultas enviadas ao OnOC devem ser decompostas e enviadas às fontes responsáveis pelos objetos requisitados. Os resultados destas consultas retornam ao catálogo e devem ser unificados para entregar um único resultado ao cliente.

Para facilitar o entendimento do relacionamento entre os requisitos listados nesta seção e as situações de uso apresentadas na seção anterior, vale salientar a relação entre eles.

Os requisitos (R1), (R2) e (R3) são necessários para viabilizar o cadastramento, acesso e permissão dos clientes de um OnOC. Estes requisitos, juntamente ao requisito (R4) para gerenciamento das ontologias do catálogo, são essenciais em qualquer situação de uso do catálogo. O requisito (R5) é necessário para que o OnOC possa atuar como um catálogo de metadados, como descrito na seção 3.3.2.1. Já os requisitos (R5) e (R6) são necessários para que o OnOC possa atuar como um catálogo de instâncias de referência, como descrito na seção 3.3.2.2. O (R7) só é necessário para que o OnOC atue como um mediador para acesso aos objetos distribuídos, como descrito na seção 3.3.2.3.