2
Fundamentos

2.1.
Web Semântica

> A Web Semântica é uma Web de dados. Diariamente utilizamos uma grande quantidade de dados, porém esses dados não são parte da Web. Por exemplo, eu posso ver meu extrato bancário na web, e minhas fotos ou meus compromissos em um calendário. Mas eu posso ver minhas fotos em um calendário para saber o que eu estava fazendo quando as fotografei? Eu posso ver as meus extratos bancários em um calendário? Porque não? Porque nós não temos uma Web de dados. Porque os dados são controlados por aplicações e cada aplicação os mantém para si próprias.
>
> A Web Semântica é uma extensão da Web atual e não uma substituta.
>
> A visão da Web Semântica é estender os princípios da Web de documentos para dados. Dados deveriam ser acessados utilizando a arquitetura geral da web por meio de URIs. Dados deveriam ser relacionados uns com os outros da mesma forma como documentos já são. Isso também significa a criação de um framework comum que permita que dados sejam compartilhados e reusados através das fronteiras das aplicações, corporações e comunidades, para serem processados automaticamente por ferramentas tão bem quanto manualmente, incluindo a criação de possíveis novos relacionamentos entre as peças de dados.
>
> [Herman, 2001]

Muito do cenário descrito na citação acima já é possível atualmente graças as tecnologias que compõe a infraestrutura da Web Semântica. A Web de dados tem se tornado uma realidade a medida que grandes repositórios de dados, de diferentes origens, são interconectados e disponibilizados segundo os padrões criados sobre a arquitetura da Web. As tecnologias da Web Semântica estão cada vez mais acessíveis e por isso, o cenário de uma aplicação que use dados de várias fontes para exibir um conjunto de fotos de pessoas que participaram de vários eventos sobre um calendário ou um mapa, com filtros por eventos relacionados por assunto, instituições, localização etc., pode ser considerado comum atualmente.

Dentre os resultados dos esforços para a criação da Web Semântica podemos destacar o padrão RDF[11], as linguagens para definição de vocabulários RDFS[12] e OWL[13], a linguagem de consultas SPARQL[14], as práticas *Linked Data* e o projeto *Linking Open Data*, que serão vistos nas seções adiante.

2.2. RDF

> O *Resource Description Framework* (RDF) é uma linguagem para a representação de informações na WWW. O RDF é particularmente projetado para representar meta dados sobre recursos na Web, como o título, autor e data de alteração de uma página Web, direitos autorais e licenciamento sobre um documento na Web, o cronograma de disponibilidade de algum recurso compartilhado, ou a descrição das preferências de um usuário da Web para entrega de informação. Entretanto, pela generalização do conceito "recurso da Web", o RDF pode ser usado para representar informações sobre qualquer coisa que possa ser identificada na Web, mesmo quando não pode ser diretamente recuperada pela Web. Exemplos incluem a informação sobre itens disponíveis em uma página de comércio eletrônico (e.g., informações sobre preços, editores e disponibilidade de livros ou CDs) ou a descrição das preferências de um usuário da Web para a entrega de informação.
>
> O RDF foi projetado para situações em que a informação precisa ser processada por aplicações, em vez de simplesmente ser mostrada para pessoas. O RDF fornece um *framework* comum para expressar esta informação de modo que possa ser trocada entre aplicações sem perda de significado. Como o RDF é um *framework* comum, projetistas de aplicações podem aproveitar a disponibilidade de ferramentas comuns para processamento e análise de informações descritas em RDF. A capacidade de troca de informações entre aplicações diferentes significa que as informações podem ser disponibilizadas para outras aplicações que não aquelas para as quais foram originalmente criadas.
>
> [Manola & Miller, 2004]

O RDF é baseado na ideia da identificação de itens de interesse, chamados de recursos, usando identificadores da Web (chamados de *Uniform Resource Identifiers*, ou URIs[15]), e na descrição destes recursos em termos de propriedades e seus valores. Isto permite ao RDF representar declarações simples sobre recursos na forma de triplas <*sujeito*, *propriedade*, *valor*>, onde o URI do recurso de interesse é o *sujeito* desta tripla. Como o valor de uma propriedade pode ser (o URI de) outro recurso, um conjunto de triplas desta forma pode ser entendido

[11] http://www.w3.org/standards/techs/rdf#w3c_all
[12] http://www.w3.org/TR/rdf-schema
[13] http://www.w3.org/TR/owl-guide
[14] http://www.w3.org/TR/rdf-sparql-query
15 http://www.ietf.org/rfc/rfc3986.txt,http://www.w3.org/TR/2010/NOTE-curie-20101216

como um grafo de nós e arcos, representando os recursos, suas propriedades e valores. Para tornar esta discussão mais concreta, o grupo de declarações "existe uma Pessoa identificada por http://www.w3.org/People/EM/contact#me, cujo nome é Eric Miller, seu endereço de e-mail é em@w3.org e seu título é Dr." poderia ser representada pelo grafo apresentado na Figura 2.

Figura 2 – Um grafo RDF que descreve Eric Miller

A Figura 2 ilustra que o RDF usa URIs para identificar:

- Indivíduos, e.g., Eric Miller, identificado por http://www.w3.org/People/EM/contact#me;
- Tipos de coisas, e.g., Pessoa, identificada por http://www.w3.org/2000/10/swap/pim/contact#Person;
- Propriedades destas coisas, e.g., *mailbox* (caixa de correio), identificada por http://www.w3.org/2000/10/swap/pim/contact#mailbox;
- Valores destas propriedades, e.g., mailto:em@w3.org como valor da propriedade *mailbox* (o RDF também usa cadeia de caracteres como "Eric Miller", e valores de outros tipos como inteiros e datas, como valores de propriedades).

O RDF também fornece uma sintaxe baseada em XML (chamada RDF/XML[16]) para a representação destes grafos. O Quadro 1 é um trecho de RDF na notação RDF/XML que corresponde ao grafo apresentado na Figura 2.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:contact="http://www.w3.org/2000/10/swap/pim/contact#">

  <contact:Person rdf:about="http://www.w3.org/People/EM/contact#me">
    <contact:fullName>Eric Miller</contact:fullName>
    <contact:mailbox rdf:resource="mailto:em@w3.org"/>
    <contact:personalTitle>Dr.</contact:personalTitle>
  </contact:Person>
</rdf:RDF>
```

Quadro 1 – RDF/XML descrevendo Eric Miller

Como HTML, RDF/XML é processável por máquinas e, usando URIs, pode interligar itens de informação através da Web. Entretanto, diferentemente do hipertexto convencional, URIs RDF podem se referir a qualquer coisa identificável, incluindo coisas que podem não ser diretamente recuperadas na Web (como a pessoa Eric Miller). O resultado é que além de descrever coisas como páginas Web, o RDF pode descrever carros, negócios, pessoas, notícias, eventos etc. As propriedades RDF possuem URIs, para identificar precisamente os relacionamentos que existem entre os itens interligados.

Além da notação RDF/XML, um grafo RDF pode ser representado em outras notações a saber: N3[17], NTriples[18] e Turtle[19].

Devido à natureza distribuída da Web, há uma necessidade de se garantir a unicidade de URIs, que são cunhados por uma infinidade de agentes na Web. Uma forma de particionar o espaço de identificadores é através da definição de *namespaces*, que servem de prefixo para as URIs. Tipicamente, um *namespace* está associado a um agente (pessoa ou organização) que gerencia os URIs prefixados com este *namespace*. Prefixos de *namespaces* podem ser utilizados para simplificar a expressão dos URIs de recursos e propriedades. A tabela a seguir lista alguns prefixos de *namespaces* bastante utilizados pela comunidade da Web Semântica.

[16] http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210
[17] http://www.w3.org/DesignIssues/Notation3
[18] http://www.w3.org/2001/sw/RDFCore/ntriples
[19] http://www.w3.org/TeamSubmission/turtle

Prefixo	URI do *namespace*
rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs	http://www.w3.org/2000/01/rdf-schema#
owl	http://www.w3.org/2002/07/owl#
foaf	http://xmlns.com/foaf/0.1/
dc	http://purl.org/dc/elements/1.1/
xsd	http://www.w3.org/2001/XMLSchema#
contact	http://www.w3.org/2000/10/swap/pim/contact#

Tabela 1 – Exemplos de prefixos de *namespaces*

No exemplo do Quadro 1 foi utilizado o prefixo "contact" para o URI http://www.w3.org/2000/10/swap/pim/contact#, simplificando as declarações sobre as classes e propriedades que utilizam este *namespace*.

Para mais informações sobre o RDF consultar [Manola & Miller, 2004].

2.3. RDF Schema

O RDF fornece um meio para expressar declarações simples sobre recursos, usando propriedades nomeadas e valores. Entretanto, as comunidades de usuários do RDF também precisam de meios para definir os vocabulários (termos) que pretendem usar nestas declarações, especificamente, para indicar que estão descrevendo tipos ou classes específicos de recursos, e propriedades específicas na descrição destes recursos.

O RDF em si não fornece meios para a definição de classes e propriedades de um domínio específico. Em vez disso, estas classes e propriedades são descritas como um vocabulário RDF, usando extensões do RDF fornecidas pela Linguagem de Descrição de Vocabulários RDF ou RDF Schema (RDFS) [Brickley & Guha, 2004].

O RDFS não fornece um vocabulário de classes de um domínio específico. Em vez disso, fornece as facilidades necessárias para descrever estas classes e propriedades, e para indicar quais classes e propriedades espera-se que sejam usadas em conjunto (por exemplo, para dizer que a propriedade foaf:name pode ser usada para descrever uma foaf:Person). Em outras palavras, o RDFS fornece um sistema de tipos para RDF. Este sistema de tipos é semelhante, em alguns

aspectos, aos sistemas tipos de linguagens de programação orientadas a objetos, como Java. Por exemplo, o RDFS permite que recursos sejam definidos como instâncias de uma ou mais classes. Também permite que as classes sejam organizadas hierarquicamente; por exemplo a classe ex:Cachorro pode ser definida como subclasse de ex:Mamifero que é subclasse de ex:Animal, significando que qualquer recursos que é da classe ex:Cachorro também é, implicitamente, da classe ex:Animal. Porém, classes e propriedades RDF são em alguns aspectos muito diferentes dos tipos de linguagens de programação. As descrições de classes e propriedades RDF não criam uma restrições sobre as informações, mas sim fornecem informações adicionais sobre os recursos que descrevem.

A Figura 3 apresenta um grafo RDF que representa uma hierarquia de classes de veículos descritas com vocabulário RDFS.

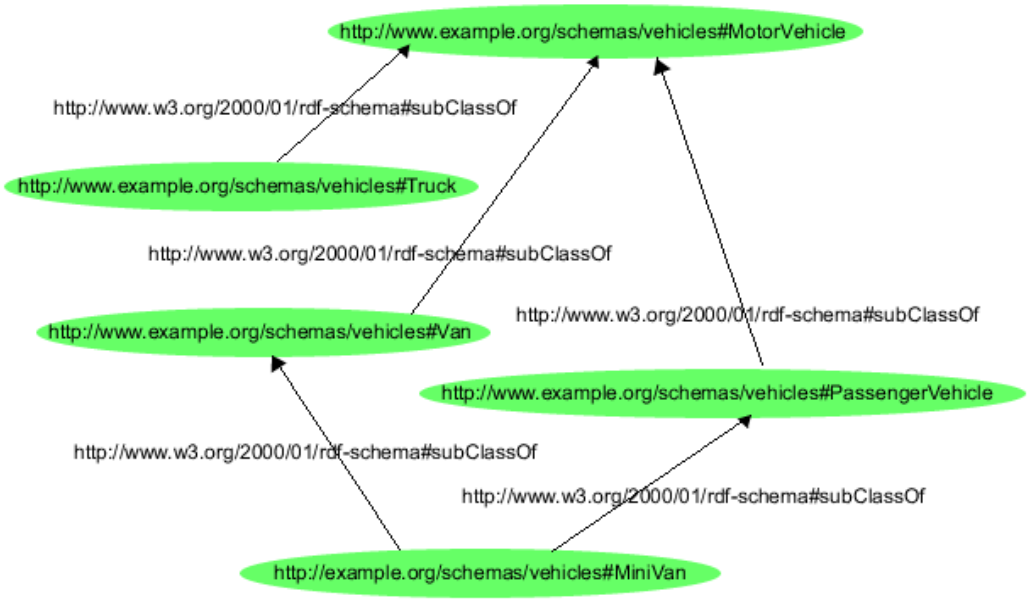

Figura 3 – Hierarquia de classes de veículos em RDFS

2.4. OWL

A OWL (*Web Ontology Language*) é uma linguagem para definir e instanciar ontologias na Web [Smith et al., 2004]. Ontologia é um termo emprestado da filosofia que se refere à ciência de descrever os tipos das entidades no mundo e como elas são relacionadas. Uma ontologia OWL pode incluir descrições de classes, propriedades e suas instâncias. Dada uma ontologia, a

semântica formal da OWL especifica como derivar suas consequências lógicas, por exemplo, os fatos não literalmente presentes na ontologia, mas decorrentes da sua semântica. Essas decorrências podem ser baseadas em um documento único ou vários documentos distribuídos que foram combinados utilizando os mecanismos de definição da OWL.

Em relação ao RDFS, OWL é considerada uma linguagem mais expressiva, pois é capaz de expressar vários tipos de relacionamentos entre classes e propriedades impossíveis de se expressar em RDFS. A OWL pode representar vários tipos de relacionamentos, tais como transitividade, simetria, relação inversa, etc.; restrições de cardinalidade e presença de valores em propriedades; equivalência entre classes e propriedades; classes derivadas de operações sobre conjuntos como interseção, união e complemento; classes enumeradas, classes disjuntas etc.

Recentemente a linguagem OWL2[20] passou a ser uma recomendação do W3C, introduzindo várias melhorias. Porém a primeira versão da OWL ainda é muito presente nas descrições dos dados disponíveis na Web Semântica.

2.5. SPARQL

SPARQL é um protocolo [Kendall et al., 2008] e um linguagem de consultas para RDF [Prud'hommeaux & Seaborne, 2008].

A linguagem de consultas SPARQL está para o RDF assim como a linguagem SQL está para os bancos de dados relacionais. A SPARQL é uma linguagem de consultas específica para grafos RDF. O Quadro 2 apresenta uma expressão de consulta em linguagem SPARQL, que deve retornar o título do livro identificado pelo URI *http://example.org/book/book1*.

```
SELECT ?title
WHERE
{
  <http://example.org/book/book1> <http://purl.org/dc/elements/1.1/title>
?title .
}
```

Quadro 2 – Expressão de consulta em linguagem SPARQL

[20] http://www.w3.org/TR/owl2-overview

Dado um grafo RDF que contenha a tripla a seguir:

```
<http://example.org/book/book1>
<http://purl.org/dc/elements/1.1/title> "SPARQL Tutorial" .
```

o resultado da execução da expressão de consulta do Quadro 2 seria o apresentado na tabela abaixo:

Title
"SPARQL Tutorial"

Para detalhes da sintaxe da linguagem SPARQL, consulte [Prud'hommeaux & Seaborne, 2008].

O protocolo SPARQL contém uma *interface*, chamada de "SparqlQuery", que contém uma operação "query" e é descrita abstratamente em termos de um *web service* que implementa sua *interface*, tipos, falhas e operações.

Um *Sparql Endpoint* é um serviço que implementa o protocolo SPARQL. Ele permite ao usuário (humano ou máquina) fazer uma consulta a uma base de conhecimento usando a linguagem SPARQL. O resultado é retornado em um formato processável por máquina, por exemplo, um arquivo RDF. Sendo assim, podemos fornecer para um navegador RDF um URI, que referencia um arquivo RDF diretamente, ou o URI de um *Sparql Endpoint*, que demanda uma consulta SPARQL e retorna um conjunto de triplas RDF. Um exemplo prático de uso de um *Sparql Endpoint* para explorar a Web Semântica é o projeto LOD que disponibiliza um conjunto de repositórios RDF interconectados, que podem ser acessados por meio de diversos *Sparql Endpoints*.

A linguagem SPARQL 1.1[21] está em desenvolvimento e inclui várias melhorias, entre elas, as extensões para consultas em bases federadas[22] que são de especial interesse para o presente trabalho. Estas extensões permitem que uma mesma consulta seja disparada para vários repositórios de dados e tenha os seus resultados combinados em uma única resposta. O Quadro 3 apresenta um exemplo de expressão de consulta em SPARQL 1.1 que utiliza as extensões de federação.

[21] http://www.w3.org/TR/sparql11-query
[22] http://www.w3.org/2009/sparql/docs/fed/service.xml

```
SELECT ?name ?projectName
WHERE
{
  SERVICE <http://people.example/sparql> {
    ?people <http://xmlns.com/foaf/0.1/name>   ?name .
    ?people <http://example.org/project/worksIn>   ?project .      }
  SERVICE <http://project.example/sparql> {
    ?project  <http://example.org/project/hasTitle>       ?projectName  .
      }
}
```

Quadro 3 - Expressão de consulta em SPARQL 1.1 com extensões de federação

A seguir apresentamos um exemplo do resultado da execução da expressão de consulta apresentada no Quadro 3.

Triplas no *Sparql endpoint* <http://people.example/sparql>:

```
<http://example.org/people/people15>
<http://xmlns.com/foaf/0.1/name>
"Mike Jackson" .

<http://example.org/people/people15>
<http://example.org/project/worksIn>
<http://example.org/project/project1> .
```

Triplas no *Sparql endpoint* <http://project.example/sparql>:

```
<http://example.org/project/project1>
<http://example.org/project/hasTitle>
"Querying remote RDF Data" .

<http://example.org/project/project2>
<http://example.org/project/hasTitle>
"Querying multiple SPARQL endpoints" .
```

Dadas as triplas acima, o resultado da execução da expressão de consulta do Quadro 3 seria o apresentado na tabela abaixo:

Name	projectName
“Mike Jackson”	“Querying remote RDF Data”

2.6. Linked Data

O termo *Linked Data* [Bizer et. al., 2007] refere-se a um conjunto de regras, princípios e melhores práticas para se publicar e interligar dados estruturados na Web com o objetivo de facilitar a exploração desses dados [Berners-Lee, 2007]. Os princípios *Linked Data* são:

- Usar URIs (*Uniform Resource Identifiers*) para nomear qualquer coisa, desde conceitos tangíveis como pessoas, frutas, carros etc.; a conceitos abstratos como amor, amizade, guerra, estatísticas etc.;
- Usar URIs de modo que seja possível encontrar estes nomes na Web, ou seja, sempre que houver um acesso ao URI (via protocolo HTTP), alguma informação deve ser retornada;
- O acesso a um URI deve ser fornecer informação útil nos padrões da Web Semântica (RDF, RDFS, OWL etc.);
- Devem ser incluídos *links* (elos) para outros URIs de modo que seja possível descobrir informações adicionais.

A ideia básica por trás das práticas *Linked Data* é aplicar a arquitetura original da Web [Jacobs & Walsh, 2004] – calcadas no uso de URIs, no protocolo HTTP, na sua natureza distribuída etc. – no compartilhamento de dados estruturados em uma escala global. A lista a seguir descreve os elementos da arquitetura da Web nos quais se baseiam as práticas *Linked Data*:

- **Recursos** – Na terminologia da arquitetura da Web, todos os itens de interesse são chamados de recursos. Os recursos são as "coisas" cujas propriedades e relacionamentos se deseja descrever. Cabe ressaltar que recursos podem denotar "coisas" abstratas;
- **Identificadores de recursos** – Recursos são identificados por *Uniform Resource Identifiers (URIs)* que, no contexto das práticas *Linked Data*, estão restritos aos HTTP URIs (evitando-se os outros esquemas de URIs como URNs[23] e DOIs[24]). Os URIs fornecem uma maneira simples de criar nomes globalmente únicos sem gestão centralizada e funcionam não apenas como nomes mas também como meio de acesso as informações sobre um recurso na Web;
- **Representações** – Uma representação é uma cadeia de *bytes* em um certo formato como HTML, RDF/XML ou JPEG. Por exemplo, uma fatura é um recurso de informação e pode ser representado como uma

[23] URN (*Uniform Resource Name*) é um URI que usa o esquema URN e não implica na disponibilidade do recurso identificado. Exemplo: urn:isbn:0451450523

[24] DOI (*Digital Object Identifier*) é uma cadeia de caracteres usada para identificar univocamente documentos eletrônicos ou outros objetos. Exemplo: doi:10.1000/182

página HTML, um arquivo PDF ou um documento RDF. Um mesmo recurso pode ter várias representações em diferentes formatos, qualidades ou idiomas;

- **Dereferenciamento de HTTP URIs** – O dereferenciamento é o processo de acessar um URI na Web para obter a informação sobre o recurso referenciado.
- **Negociação de conteúdo** – É um passo da comunicação via protocolo HTTP em que o cliente envia cabeçalhos com cada requisição indicando que tipo de representação prefere receber. e.g. text/html, application/rdf+xml. De posse desta informação, cabe ao servidor entregar a representação no formato que possuir, seguindo a ordem de preferências indicada no cabeçalho da requisição. Em alguns casos, uma resposta válida é um redirecionamento para um segundo URI que se refere à representação preferida.

2.6.1. *Linked Data* e o modelo de dados RDF

Os dados publicados na Web segundo as práticas *Linked Data* representam informações sobre recursos por meio do RDF (*Resource Description Framework*). O RDF fornece um modelo de dados que é extremamente simples, mas estritamente adaptado para a arquitetura da Web.

Os principais benefícios do uso do modelo de dados RDF no contexto das práticas *Linked Data* são:

- Os clientes podem encontrar qualquer URI em um grafo RDF através da Web e obter informação adicional;
- Informações de diferentes origens se mesclam naturalmente;
- O modelo de dados permite a criação de links RDF entre dados de diferentes origens;
- O modelo de dados permite a representação de informações de diferentes esquemas em um modelo único;

- Combinados com linguagens de esquemas como RDFS ou OWL, o modelo de dados permite a representação de dados fortemente estruturados tão bem quanto dados semi-estruturados.

No contexto das práticas *Linked Data*, algumas funcionalidades do modelo RDF devem ser evitadas:

- O uso de nós anônimos ("blank nodes"), pois não é possível definir *links* RDF externos para nós anônimos e a mesclagem de dados de diferentes origens é muito difícil quando nós anônimos são usados;
- O uso de reificações[25], pois a semântica da reificação não é clara.
- O uso de coleções e contêineres RDF [Antoniou & Van Harmelen, 2008], pois não trabalham bem com a linguagem SPARQL e pode-se obter resultados equivalentes com o uso de múltiplas triplas contendo o mesmo predicado.

2.6.2. Criação de URIs

Recursos são nomeados com URIs. Ao se publicar dados é importante que se dedique algum esforço na escolha de bons URIs para os recursos [Sauermann et al., 2008]. Embora os URIs devam ser bons nomes de modo que outros tenham confiança em usá-los como *links* para seus próprios dados, deve haver uma infraestrutura técnica para que os URIs sejam dereferenciáveis, o que pode impor alguns limites no seu uso. A seguir, são listadas algumas recomendações para a escolha de URIs:

- Deve-se usar HTTP URIs para tudo. O esquema http:// é o único completamente suportado pelas ferramentas e infraestrutura atuais;

[25] Em RDF é possível fazer declarações sobre declarações. Por exemplo, dada a tripla <ex:a> <ex:b> <ex:c> ., é possível criar um grafo que "fale" sobre essa tripla:
_:xxx rdf:type rdf:Statement .
_:xxx rdf:subject <ex:a> .
_:xxx rdf:predicate <ex:b> .
_:xxx rdf:object <ex:c> .
Chamamos este grafo de reificação, que geralmente é usada para descrever confiança sobre outras declarações.

- Os URIs devem ser definidos em um HTTP *namespace* sob controle de quem está publicando os dados. Não se deve definir URIs em *namespaces* de terceiros;
- Nomes curtos são melhores. Exemplo: http://dbpedia.org/resource/Berlin é melhor que http://www4.wiwiss.fu-berlin.de:2020/demos/dbpedia/cgi-bin/resources.php?id=Berlin;
- Deve-se tentar manter os URIs estáveis e persistentes. Alterar os URIs podem quebrar *links* previamente estabelecidos, por isso é aconselhável dedicar alguma reflexão extra para eles, numa fase inicial.
- Os URIs escolhidos podem ser limitados pelo ambiente técnico do proprietário dos dados. Se um servidor é chamado demo.serverpool.wiwiss.example.org e não há opção de se obter outro nome de domínio, todos URIs serão iniciados por http://demo.serverpool.wiwiss.example.org/. Se não for possível executar o servidor na porta 80, então todas os URIs poderão ser iniciadas por http://demo.serverpool.wiwiss.example.org:2020/. Quando possível, deve-se limpar estes URIs pela adição de algumas regras de reescrita de URIs configuráveis nas diretivas do servidor Web.
- Frequentemente será necessário usar algum tipo de chave primária como parte do URI para garantir que cada um é único. Quando possível deve-se usar uma chave primária significativa do domínio utilizado. Por exemplo, quando se trata de livros, usar o número ISBN com parte do URI é melhor do que usar uma chave primária interna de uma tabela de um banco de dados relacional.

2.6.3. Que vocabulários usar para representar informações?

A fim de tornar tão fácil quanto possível o processamento dos dados publicados pelos aplicativos clientes, deve-se reutilizar os termos dos vocabulários bem conhecidos da comunidade sempre que possível. A definição de novos termos próprios deve ocorrer somente quando não for possível encontrar termos adequados aos requisitos originais nos vocabulários existentes. Um conjunto de vocábulos conhecidos evoluiu na comunidade da Web Semântica.

Recomenda-se verificar se os dados podem ser representados usando termos desses vocabulários antes de definir eventuais novos termos:

Friend-of-a-Friend (FOAF)[26] – vocabulário para descrever pessoas;

Dublin Core (DC)[27] – define meta dados genéricos;

Semantically-Interlinked Online Communities (SIOC)[28] – vocabulário para representar comunidades *online*;

Description of a Project (DOAP)[29] – vocabulário para descrever projetos;

Simple Knowledge Organization System (SKOS)[30] – vocabulário para representar taxonomias e conhecimento fracamente estruturado;

MusicOntology[31] – fornece termos para descrever artistas, álbuns e músicas;

Review Vocabulary[32] – vocabulário para representar revisões;

Creative Commons (CC)[33] – vocabulário para descrever termos de licenças.

Uma lista mais extensa de vocabulários[34] bem conhecidos é mantida pelo projeto *Linking Open Data*.

É uma prática comum mesclar termos de diferentes vocabulários. É especialmente recomendado que se use as propriedades rdfs:label e foaf:depiction sempre que possível pois são termos bem suportados pelas aplicações clientes, geralmente utilizados na representação visual do recurso.

Se for necessário utilizar URIs para localizações geográficas, áreas de pesquisa, tópicos gerais, artistas, livros ou CDs, deve-se considerar o uso de URIs dos repositórios de dados disponibilizados pelo projeto *Linking Open Data*, por exemplo *Geonames*[35], *DBpedia*[36], *Musicbrainz*[37], *dbtune*[38] ou *RDF Book*

[26] http://xmlns.com/foaf/0.1
[27] http://dublincore.org/documents/dcmes-xml
[28] http://sioc-project.org
[29] http://usefulinc.com/doap
[30] http://www.w3.org/2004/02/skos
[31] http://musicontology.com
[32] http://purl.org/stuff/rev#
[33] http://creativecommons.org/ns
[34] http://esw.w3.org/TaskForces/CommunityProjects/LinkingOpenData/CommonVocabularies
[35] http://www.geonames.org/ontology
[36] http://dbpedia.org
[37] http://musicbrainz.org
[38] http://dbtune.org

Mashup[39]. Os dois principais benefícios de se usar URIs destas fontes de dados são:

1. Os URIs são dereferenciáveis, significando que a descrição do conceito pode ser recuperada a partir da Web. Por exemplo, usar o URI do *DBPedia* http://dbpedia.org/page/Doom para identificar o jogo de computador "Doom" fornece uma extensa descrição do jogo, incluindo resumos em dez diferentes idiomas e várias classificações;
2. Os recursos denotados pelos URIs já estão ligados aos URIs de outras fontes de dados. Por exemplo, é possível navegar a partir do URI http://dbpedia.org/resource/Berlin para dados sobre Berlim fornecidos pelo *Geonames* e *EuroStat*[40]. Portanto, ao utilizar URIs dos conceitos destas fontes de dados, é possível interligar estes dados com uma rede rica e de rápido crescimento de outras fontes de dados.

Uma lista mais extensa de repositórios de dados com URIs dereferenciáveis é mantida pelo projeto LOD no endereço http://ckan.net/group/lodcloud. Estes repositórios fazem parte da nuvem LOD apresentada na Figura 1.

Bons exemplos de como os termos de diferentes vocabulários bem conhecidos são mesclados em um documento e como URIs de conceitos existentes são reaproveitados são fornecidos pelos perfis FOAF de Tim Berners-Lee[41] e Ivan Herman[42].

Quando não for possível encontrar bons vocabulários que cubram as classes e propriedades necessárias, deve-se definir termos próprios. Definir novos termos não é difícil. Classes e propriedades RDF são recursos próprios, identificados por URIs e publicados na Web. Por isso, tudo o que foi dito sobre a publicação de dados de acordo com as práticas *Linked Data* aplica-se também aos novos termos. Os vocabulários são definidos usando RDFS e OWL.

A seguir são listadas alguns passos guias na definição de novos vocabulários:

[39] http://www4.wiwiss.fu-berlin.de/bizer/bookmashup/index.html
[40] http://www4.wiwiss.fu-berlin.de/eurostat
[41] http://www.w3.org/People/Berners-Lee/card
[42] http://www.ivan-herman.net/foaf.rdf

- **Não defina novos vocabulários do zero**, mas complemente os que já existem com termos adicionais (em seu *namespace*) para representar seus dados.
- **Forneça informações para humanos e máquinas**. e.g. sempre crie um rótulo para cada termo usando a propriedade rdfs:label e forneça rdfs:comments para cada termo inventado.
- **URIs dos termos dereferenciáveis**. É essencial que os URIs dos termos sejam dereferenciáveis de forma que os clientes possam encontrar a definição de um termo.
- **Faça uso de termos de terceiros**. O uso de termos de terceiros ou o mapeamento entre eles ajuda a promover o nível de troca de dados na Web de Dados da mesma forma que links de hipertextos faz para a tradicional web de documentos.
- **Expresse toda informação importante explicitamente**. Por exemplo, expresse todos os *ranges* e *domains* explicitamente. Não deixe informações importantes de fora.
- **Não crie modelos rígidos e frágeis**; deixe alguma flexibilidade para a expansão do modelo. Se você não sabe exatamente o que está fazendo, use RDFS para definir seus vocabulários. Um exemplo de vocabulário bem definido é a Bibliographic Ontology (BIBO) [43], que foi usado em uma das aplicações de exemplo apresentadas no capítulo 5 desta dissertação.

O Quadro 4 é um exemplo que contém a definição de uma classe e uma propriedade seguindo as regras descritas acima. Este exemplo usa a notação Turtle[44]. As declarações de *namespaces* foram omitidas.

```
# Definition of the class "Lover"

<http://sites.wiwiss.fu-berlin.de/suhl/bizer/pub/LoveVocabulary#Lover>
    rdf:type rdfs:Class ;
    rdfs:label "Lover"@en ;
    rdfs:label "Liebender"@de ;
    rdfs:comment "A person who loves somebody."@en ;
    rdfs:comment "Eine Person die Jemanden liebt."@de ;
    rdfs:subClassOf foaf:Person .
```

[43] http://bibliontology.com
[44] http://www.w3.org/TeamSubmission/turtle

```
# Definition of the property "loves"

<http://sites.wiwiss.fu-berlin.de/suhl/bizer/pub/LoveVocabublary#loves>
    rdf:type rdf:Property ;
    rdfs:label "loves"@en ;
    rdfs:label "liebt"@de ;
    rdfs:comment "Relation between a lover and a loved person."@en ;
    rdfs:comment "Beziehung zwischen einem Liebenden und einer geliebten
Person."@de ;
    rdfs:subPropertyOf foaf:knows ;
    rdfs:domain                                    <http://sites.wiwiss.fu-
berlin.de/suhl/bizer/pub/LoveVocabulary#Lover> ;
    rdfs:range foaf:Person .
```

Quadro 4 - Exemplo de definição de uma classe e uma propriedade

É possível observar que os termos são definidos em um *namespace* que é controlado por Chris Bizer e eles estão relacionados ao vocabulário FOAF usando os *links* rdfs:subPropertyOf e rdfs:subClassOf.

2.6.4. O que deve ser retornado como descrição RDF de um URI?

Assumindo que todos os dados são expressos em triplas RDF: que triplas deveriam participar da representação RDF que é retornada em resposta à dereferenciação de um URI que identifica um recurso?

1. **A descrição**: a representação RDF deve incluir todas as triplas de seu conjunto de dados que possuem o URI do recurso como sujeito. Esta é a descrição imediata do recurso;
2. ***Backlinks***: todas as triplas que possuem o URI do recurso como objeto;
3. **Descrições relacionadas**: pode-se incluir qualquer informação adicional sobre recursos relacionados que podem ser de interesse. e.g. autores e livros;
4. **Meta dados**: a representação pode conter qualquer meta dado a ser anexado ao dado publicado. e.g. um URI que identifica o autor e a informação de licença;
5. **Sintaxe**: deve-se fornecer pelo menos descrições em RDF/XML, que é uma sintaxe oficial para RDF. Pode-se fornecer sintaxes adicionais como Turtle (application/x-turtle);

O Quadro 5 apresenta parte da descrição RDF do recurso http://dbpedia.org/data/Alec_Empire na notação Turtle.

```
# Metadata and Licensing Information

<http://dbpedia.org/data/Alec_Empire>
    rdfs:label "RDF description of Alec Empire" ;
    rdf:type foaf:Document ;
    dc:publisher <http://dbpedia.org/resource/DBpedia> ;
    dc:date "2007-07-13"^^xsd:date ;
    dc:rights
        <http://en.wikipedia.org/wiki/WP:GFDL> .

# The description

<http://dbpedia.org/resource/Alec_Empire>
    foaf:name "Empire, Alec" ;
    rdf:type foaf:Person ;
    rdf:type <http://dbpedia.org/class/yago/musician> ;
    rdfs:comment
        "Alec Empire (born May 2, 1972) is a German musician who is
..."@en ;
    rdfs:comment
        "Alec Empire (eigentlich Alexander Wilke) ist ein deutscher
Musiker. ..."@de ;
    dbpedia:genre <http://dbpedia.org/resource/Techno> ;
    dbpedia:associatedActs
<http://dbpedia.org/resource/Atari_Teenage_Riot> ;
    foaf:page <http://en.wikipedia.org/wiki/Alec_Empire> ;
    foaf:page <http://dbpedia.org/page/Alec_Empire> ;
    rdfs:isDefinedBy <http://dbpedia.org/data/Alec_Empire> ;
    owl:sameAs  <http://zitgist.com/music/artist/d71ba53b-23b0-4870-a429-
cce6f345763b> .


# Backlinks

<http://dbpedia.org/resource/60_Second_Wipeout>
    dbpedia:producer <http://dbpedia.org/resource/Alec_Empire> .
<http://dbpedia.org/resource/Limited_Editions_1990-1994>
    dbpedia:artist <http://dbpedia.org/resource/Alec_Empire> .
```

Quadro 5 – Exemplo de descrição de recurso RDF

2.6.5. Como gerar *links* RDF para outras fontes de dados?

Links RDF permitem que navegadores *Linked Data* e rastreadores naveguem pelas fontes de dados para descobrir dados adicionais. O domínio da aplicação determinará quais propriedades RDF são usadas como predicados. Por exemplo, propriedades comumente utilizadas no domínio de descrição de pessoas são foaf:knows, foaf:based_near e foaf:topic_interest. Exemplos combinando essas propriedades com valores da *DBPedia*, *DBLP bibliography*[45] e *RDF Book Marshup* são encontradas nos perfis FOAF de Tim Berners-Lee e Ivan Herman.

É um prática comum usar a propriedade owl:sameAs para afirmar que outra fonte de dados também fornece informações sobre um recurso específico. Um *link*

[45] http://dblp.l3s.de/d2r

owl:sameAs indica que dois URIs se referem à mesma coisa. Portanto, owl:sameAs é usado para mapear entre diferentes URIs. Exemplos de uso de owl:sameAs para indicar que dois URIs falam sobre a mesma coisa também são encontrados no perfil FOAF de Tim Berners-Lee que afirma que http://www.w3.org/People/Berners-Lee/card#i identifica o mesmo recurso que http://www4.wiwiss.fu-berlin.de/bookmashup/persons/Tim+Berners-Lee and http://www4.wiwiss.fu-berlin.de/dblp/resource/person/100007. No entanto, é necessário exercer cautela no uso da propriedade owl:sameAs, pois ela implica na identidade *total* dos recursos relacionados, o que nem sempre é verdade.

Links RDF podem ser criados manualmente ou gerados automaticamente por meio de algoritmos. A segunda abordagem é mais comumente utilizada para grandes conjuntos de dados [Bizer et al., 2008].

2.7. Aplicações na Web Semântica

Desde o surgimento do RDF muitas aplicações passaram a fazer o seu uso como modelo de dados e com isso tirar proveito das vantagens oferecidas por esse modelo, como a independência de esquema, poucas restrições de conformidade, fácil distribuição etc. Contudo, não é toda aplicação que utiliza o RDF como modelo de dados que deve ser considerada uma aplicação na Web Semântica. O uso do RDF é apenas um primeiro passo para que possa ser possível para uma aplicação utilizar os dados distribuídos na nuvem LOD. No contexto desta dissertação, são consideradas aplicações na Web Semântica aquelas que, além de fazerem uso do modelo de dados RDF, também utilizam os dados distribuídos na nuvem LOD para apoiar as suas tarefas. O site da BBC para a cobertura da Copa do Mundo de Futebol em 2010[46] é um exemplo de aplicação na Web Semântica, pois utiliza extensamente os dados da nuvem LOD para apoiar suas tarefas.

Um fator a se considerar é que, antes de qualquer coisa, as aplicações na Web Semântica são aplicações. Por isso, a construção destas aplicações deve considerar os mesmos aspectos que são importantes para qualquer outra aplicação. Uma aplicação deve apoiar as tarefas de um determinado usuário em um

[46] http://news.bbc.co.uk/sport2/hi/football/world_cup_2010/default.stm

determinado contexto. O projetista deve modelar cada interesse ou aspecto da aplicação para que as tarefas identificadas sejam apoiadas com sucesso. Sendo assim, são gerados modelos que descrevem o domínio, os comportamentos, as interfaces e outros aspectos da aplicação.

No caso das aplicações na Web Semântica, devido ao modelo RDF permitir que se descreva livremente dados e meta dados estruturados, é possível representar os próprios modelos da aplicação em RDF. Com isso, as aplicações na Web Semântica podem ser descritas por modelos processáveis em RDF. O que significa que, além de ser possível processar os dados do domínio da aplicação, também é possível processar os dados e meta dados que descrevem a própria aplicação. Isso torna possível, entre outras coisas, a criação de ferramentas como o Synth – apresentado neste trabalho – que recebe como entrada modelos que descrevem vários aspectos de uma aplicação representados em RDF e produzem como saída uma aplicação executável resultante da interpretação destes modelos.

2.7.1. Aplicações Corporativas na Web Semântica

Segundo [Steele, 2006; Bohn & Short, 2009], as informações críticas para o sucesso do negócio das corporações são difíceis de se manter, integrar, recuperar e usar. Os sistemas de informações corporativos manipulam uma grande quantidade de dados e por isso precisam oferecer escalabilidade efetiva. A arquitetura da Web tem se mostrado uma abordagem bem sucedida para a integração de sistemas complexos, entre eles os sistemas corporativos. Com isso, tem surgido uma classe de aplicações conhecidas como aplicações corporativas na Web Semântica (*Enterprise Semantic Web Applications*), que são aquelas aplicações que utilizam as tecnologias da Web Semântica na sua construção, dados distribuídos na Web, vocabulários conhecidos da comunidade, porém parte dos seus dados, URIs e vocabulários são controlados por uma corporação.

Esta classe de aplicações têm, além das necessidades comuns as outras aplicações na Web Semântica, de tratar de questões pertinentes ao mercado corporativo como segurança dos dados privados, confiabilidade das informações, disponibilidade e recuperação de falhas, etc.

2.8. Métodos para construção de aplicações hipermídia

Esta seção apresenta resumidamente alguns métodos para construção de aplicações hipermídia. Cabe ressaltar que até o presente trabalho, nenhum dos métodos disponíveis atualmente levavam em consideração questões importantes para aplicações na Web Semântica desenvolvidas segundo as práticas *Linked Data*, como o uso dos dados distribuídos na nuvem LOD ou a ênfase na reutilização de vocabulários conhecidos da comunidade. Estes métodos, no máximo, mencionam o uso do RDF como base de dados da aplicação.

2.8.1. OOHDM e SHDM

O OOHDM (*Object Oriented Hypermedia Design Method*) é um método orientado a objetos para design de aplicações *web* [Rossi et al., 2003], onde a aplicação é modelada em cinco etapas distintas: Levantamento de Requisitos, Modelagem Conceitual, Modelagem Navegacional, Projeto de Interface Abstrata e Implementação [Schwabe et al., 1999]. O processo se dá de forma iterativa e incremental seguindo o desenvolvimento de cada etapa e produzindo novos modelos ou enriquecendo os já existentes.

O SHDM (*Semantic Hypermedia Design Method*) [Lima, 2003] é uma evolução do OOHDM, e manteve, portanto, os seus fundamentos, além de acrescentar no método os formalismos e primitivas introduzidos pela *Web* Semântica. No capítulo 3 do presente trabalho, serão apresentadas as mudanças recentes no método SHDM que o tornaram mais consistente com as práticas *Linked Data*.

2.8.2. UWE e WebML

A metodologia UWE (*UML-based Web Engineering*) [Koch e Kraus, 2002] é baseada no enfoque orientado a objetos e interativo da *Unified Modeling Language* [UML, 2009] para especificação de aplicações *web*. O principal foco do UWE é a sistematização, personalização e geração semiautomática de aplicações *web*.
O processo de autoria do UWE é composto de quatro passos, sendo estes a Análise de Requisitos, o Design Conceitual, o Design Navegacional e o Design de Apresentação, que produzem os seguintes artefatos [Koch et al., 2001]: modelo de caso de uso, modelo conceitual, modelo do espaço navegacional e modelo da

estrutura navegacional, e modelo de apresentação. Estes modelos são refinados em sucessivas iterações no processo de desenvolvimento do UWE.

WebML (*Web Modeling Language*) [Ceri et al., 2000] é um modelo para design de aplicações suportado por uma ferramenta denominada WebRatio [Ceri et al., 2000]. Possui três modelos básicos: Modelo de Dados (correspondente ao modelo conceitual do OOHDM), Modelo de Hipertexto (análogo ao modelo navegacional do OOHDM) e um Modelo de Apresentação.
O Modelo de Dados usa uma notação simplificada do modelo de Entidade-Relacionamento, no qual as entidades são representadas como retângulos e conectadas por relacionamentos binários, representados por linhas retas com o nome do relacionamento.

Os principais componentes do Modelo de Hipertexto do WebML são as *views*, áreas, páginas, unidades, operações, *links* e variáveis de sessão/aplicação. Uma *view* é um grafo de páginas, possivelmente agrupado em áreas, permitindo aos usuários de um determinado grupo que realizem atividades específicas. As páginas contêm unidades conectadas por *links*, que representam partes atômicas de informação a serem publicadas. As unidades são elementos que representam a informação descrita no modelo de dados. É no modelo de hipertexto, ou modelo navegacional, que a navegação de *links* entre as páginas e entre as unidades é definida.

O método WebML permite que operações de atualização sejam especificadas para compor aplicações *web*. As operações básicas de atualização são: criação, modificação e exclusão de instâncias de uma entidade. É permitida também a criação ou exclusão de instâncias de um relacionamento. Algumas operações podem enviar e-mail ou invocar *web services*. Como operações não disponibilizam dados para o usuário, elas não são incluídas em páginas.

No Modelo de Apresentação são especificadas o posicionamento das unidades nas páginas e a aparência gráfica da aplicação.

[Santos, 2010]

2.8.3. OOWS

O OOWS (*Object Oriented Web Solution*) [Pastor et al., 2003] é um método que captura os requisitos de uma aplicação Web por meio de modelos conceituais para obter, automaticamente, protótipos operacionais a partir destes modelos. O desenvolvimento segundo o OOWS possui duas etapas principais: modelagem conceitual e desenvolvimento da solução.

Na modelagem conceitual é obtida a especificação do sistema usando modelos conceituais. Esta etapa é dividida em três sub-etapas:

1. Levantamento de requisitos funcionais (*functional requirements elicitation*) – Técnicas baseadas em casos de uso e cenários são aplicadas para a construção de um esquema conceitual;

2. Modelagem conceitual clássica (*classic conceptual modelling*) – Usando modelos estruturais, funcionais e dinâmicos, o sistema é estruturado e os comportamentos são capturados;
3. Modelagem de navegação e apresentação – O modelo navegacional é construído para modelar requisitos navegacionais do diagrama de classes. Uma vez que o modelo navegacional é construído, os requisitos de apresentação são especificados usando o modelo de apresentação que é fortemente baseado no modelo navegacional.

Na etapa de desenvolvimento da solução é determinada a plataforma alvo e é selecionado um estilo de arquitetura específico. Então, um conjunto de correspondências entre as primitivas de abstração e os elementos que implementam cada camada do estilo de arquitetura são aplicados de modo a se obter automaticamente o sistema final. Um estilo de arquitetura em camadas utilizado no OOWS utiliza as camadas de apresentação, aplicação e persistência.

2.8.4. Hera

O propósito do Hera (http://wwwis.win.tue.nl/~hera) é suportar o projeto de aplicações que fornecem estruturas Web baseadas em navegação (apresentações hipermídia) sobre dados semanticamente estruturados, em RDF, de forma personalizada e adaptativa [Rossi et al., 2008]. A abordagem é centrada em modelos que representam aspectos essenciais do projeto de aplicações. A

Figura 4 apresenta uma visão geral dos modelos do Hera.

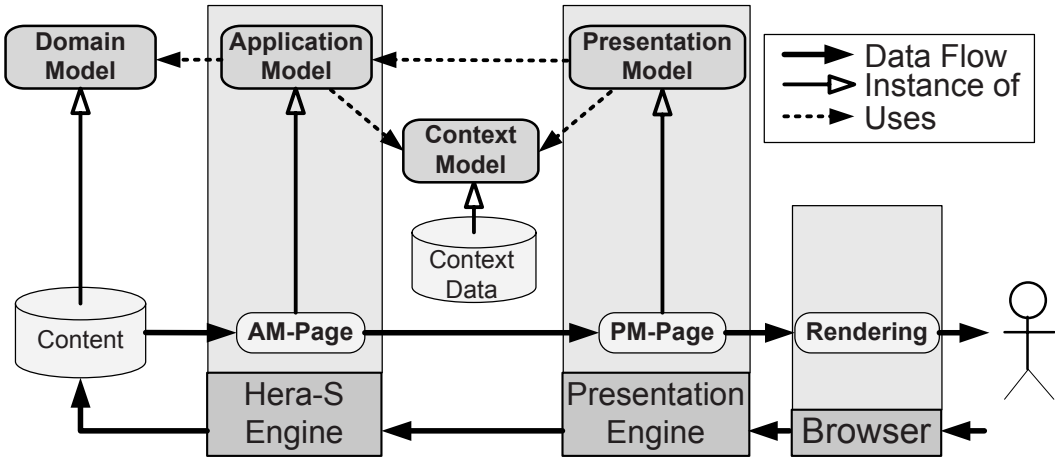

Figura 4 – Modelos do Hera

O ponto inicial da modelagem com o Hera é o modelo de domínio (*Domain Model* – DM) que descreve a estrutura dos dados de conteúdo. Baseado no modelo de domínio, o projetista deve criar o modelo da aplicação (*Application Model* – AM) que descreve a estrutura navegacional sobre o conteúdo. O Hera permite a personalização e adaptação dinâmicas do conteúdo. Para este propósito, dados de contexto são mantidos (sob controle da aplicação) em um modelo de contexto (*Context Model* – CM). Estes dados de contexto são tipicamente atualizados com base nas interações do usuário. Com base no DM e CM, o AM serve como uma receita que prescreve como o conteúdo é transformado em uma estrutura de navegação. A instanciação do AM com conteúdos concretos resulta nas páginas AM (*Application Model Pages* – AMP), que podem ser vistas como páginas que contém conteúdo para ser apresentado e primitivas de navegação (baseadas nas relações semânticas do DM) que o usuário pode usar para navegar para outras AMPs e então "mover" semanticamente para uma parte diferente do conteúdo. Uma AMP não é diretamente adequada para um navegador web, mas pode ser transformada em uma apresentação adequada pelo gerador de apresentação, i.e., uma máquina que executa uma especificação, por exemplo, um modelo de apresentação (*Presentation Model* – PM) do projeto de apresentação concreta em termos de leiaute e outros detalhes de apresentação. Em resumo, o AM especifica a construção da estrutura navegacional sobre o conteúdo, enquanto a fase de apresentação especificada pela PM é responsável pela transformação desta estrutura em elementos que se adaptam à situação concreta de navegação.