



UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

**Integração entre redes heterogêneas de ativos digitais:
modelo híbrido de metadados baseado em meta-
modelos dinâmicos**

DIEGO ERNESTO ROSA PESSOA

João Pessoa
2012



UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

**Integração entre redes heterogêneas de ativos digitais:
modelo híbrido de metadados baseado em meta-
modelos dinâmicos**

Diego Ernesto Rosa Pessoa

Dissertação submetida ao Programa de Pós-Graduação em Informática da
Universidade Federal da Paraíba, como parte dos requisitos necessários
para obtenção do grau de mestre em Informática.

Área de Concentração: Ciência da Computação

Linha de Pesquisa: Computação Distribuída

Dênio Mariz Timóteo de Sousa
(Orientador)

João Pessoa, Paraíba, Brasil
Agosto/2012

P475i Pessoa, Diego Ernesto Rosa.

Integração entre redes heterogêneas de ativos digitais: modelo híbrido de metadados baseado em meta-modelos dinâmicos / Diego Ernesto Rosa Pessoa.-- João Pessoa, 2012.

162f. : il.

Orientador: Dênio Mariz Timóteo de Sousa

Dissertação (Mestrado) – UFPB/CCEN

1. Informática. 2. Ciência da Computação. 3. *Integração de sistemas.* 4. *Interoperabilidade de metadados.* 5. *Gerenciamento de ativos digitais.*

UFPB/BC

CDU: 004(043)

Resumo

O crescimento da produção de ativos digitais dentro de ambientes corporativos torna o gerenciamento de ativos digitais uma tarefa cada vez mais complexa, visto que é preciso fazer uso de metadados bem definidos, de maneira que seja possível localizar e utilizar facilmente ativos provenientes de diferentes repositórios. Entretanto, obter um modelo padronizado, que atenda às peculiaridades de todos os envolvidos, é normalmente uma tarefa altamente complexa, devido às heterogeneidades existentes entre os modelos. Adiciona-se ainda o fato de que as soluções existentes se especializam em resolver uma parte do problema, deixando uma lacuna nas demais fases do processo de integração. Sendo assim, para o desenvolvimento dessa dissertação, foi proposto, implementado e avaliado um framework de serviços que visa contemplar a maior parte do fluxo de trabalho do processo de integração entre sistemas de gerenciamento de ativos digitais. Esse framework visa reunir e complementar soluções já existentes, disponibilizando os recursos necessários para apoiar a identificação das correlações entre elementos dos modelos de metadados e fornecendo uma arquitetura baseada em componentes responsáveis pela geração dinâmica de meta-modelos, que por sua vez irão agregar o resultado da análise das correlações e servirão como uma linguagem unificada, possibilitando a interoperabilidade entre os repositórios de ativos digitais, independentemente do padrão de metadados utilizado por cada um. A validação do trabalho foi realizada através da execução de um experimento utilizando uma prova de conceito aplicada a um cenário envolvendo modelos heterogêneos de sistemas de gerenciamento de ativos digitais reais. O objetivo do experimento foi realizar uma avaliação da qualidade do mapeamento fornecido automaticamente. Como resultado foi obtido um índice de acerto de até 98% dentro do universo dos elementos que obtiveram ao menos uma sugestão de correlação, o que ocorreu para 66% dos elementos dos modelos.

Palavras-chave: Interoperabilidade de metadados, Integração de sistemas, Gerenciamento de ativos digitais.

Abstract

The increased production of digital assets in enterprise environments makes digital asset management an increasingly complex task, since we must make use of well-defined metadata, so that you can easily find and use assets from different repositories. However, obtaining a standardized model that cover the peculiarities of all involved, is usually a highly complex task due to heterogeneity between the models. In addition, is considered the fact that existing solutions specialize in solving part of the problem, leaving a gap in the other phases of the integration process. Thus, for the development of this dissertation was proposed, implemented and evaluated a service framework for metadata interoperability that aims to encompass most of the workflow of digital asset management integration process. This framework aims to bring together and complement existing solutions, providing support to semi-automatic identification of correlations between elements of metadata models and a component-based architecture responsible for the dynamic generation of meta-models, which in turn will aggregate the result of the correlation analysis and serve as a unified language, enabling interoperability between digital assets repositories, regardless the metadata standard used by each one. Lastly, the work validation was conducted by performing an experiment using a proof of concept applied to a scenario involving heterogeneous models of digital asset management real systems. The objective of experiment was perform an assessment of quality of the automatically mapping obtained. The result was accuracy up to 98% within the universe of elements that had at least an correlation suggestion, which occurred in 66% of the elements of the models.

Keyword: Metadata interoperability, System integration, Digital Asset Management.

Agradecimentos

Primeiramente a Deus, pela saúde, paz e fé, que me deu forças para lutar pelos meus sonhos, proporcionando a realização de mais uma conquista em minha vida.

À minha mãe, Sonely Pessoa, pelo carinho, apoio, tolerância, incentivo e fé em mim. Estando ao meu lado tanto para comemorar as conquistas como para apoiar nos momentos difíceis.

Ao meu pai, Joaquin Rosa, pelo exemplo de vida, força, perseverança, coragem e luta, e pelas sábias lições de esperança e confiança, que me ajudaram a amadurecer e a buscar a realização dos meus sonhos.

Ao meu orientador Dênio Mariz, pela paciência e orientação durante todo o processo de construção deste trabalho.

Aos membros da banca, por participarem e opinarem neste trabalho dando sua honrosa contribuição.

Aos amigos e companheiros de trabalho, por todo tempo que passamos juntos, seja em momentos de diversão ou em momentos sérios.

Enfim, a todos que contribuíram de alguma forma para a realização deste trabalho. Muito obrigado!

Conteúdo

Resumo.....	3
Abstract.....	5
Agradecimentos	6
Conteúdo	7
Lista de Símbolos.....	9
Lista de Figuras.....	10
Lista de Tabelas.....	12
1 Introdução	14
1.1 Motivações	15
1.1.1 P1: Compatibilidade com formatos de armazenamento de dados ou interfaces de acesso	17
1.1.2 P2: Integração de modelos de metadados heterogêneos.....	18
1.1.3 P3: Localização transparente e descoberta.....	19
1.2 Objetivos	21
1.2.1 Objetivo Geral	21
1.2.2 Objetivos Específicos	22
1.3 Metodologia.....	22
1.4 Estrutura da Dissertação	23
2 Fundamentação teórica	24
2.1 Inter-redes.....	24
2.1.1 Definição de Inter-rede	24
2.1.2 Organização de uma inter-rede	25
2.2 Gerenciamento de ativos digitais.....	26
2.2.1 Ativos digitais	27
2.2.2 Sistemas de gerenciamento de ativos digitais	27
2.3 Metadados	30
2.3.1 Padrões de metadados	30
2.3.2 Classificação dos metadados	33
2.3.3 Representação de metadados em sistemas de informações.....	36
2.3.4 Linguagens de definição de esquemas.....	40
2.3.5 Distinção entre modelos e meta-modelos.....	41
3 Interoperabilidade de metadados em ambientes heterogêneos.....	43
3.1 Conceitos ligados à interoperabilidade de metadados.....	43
3.1.1 Níveis de interoperabilidade	44
3.2 Heterogeneidades que dificultam a interoperabilidade	46
3.2.1 Heterogeneidades estruturais.....	47
3.2.2 Heterogeneidades semânticas	50
3.3 Nível de integração necessário para a interoperabilidade.....	51
3.4 Pesquisas sobre interoperabilidade	53
3.4.1 Ontologias e Interoperabilidade	53
3.4.2 Classificação de elementos contidos em modelos de interoperabilidade	56

4 Trabalhos Relacionados	57
4.1 Propostas de Arquitetura de Integração	57
4.1.1 Uso de um modelo global	57
4.1.2 Uso de múltiplos modelos.....	59
4.1.3 Uso de modelos combinados	59
4.1.4 Quadro comparativo	60
4.2 Sistemas de correspondência de metadados	62
4.2.1 MuMle (Multi-level Metadata Mapping System)	63
5 Um Framework de serviços para Integração de bases de dados heterogêneas....	74
5.1 Visão geral do Dynamo Framework.....	75
5.1.1 Requisitos de software	77
5.1.2 Ferramentas e tecnologias	79
5.1.3 Arquitetura de componentes.....	81
5.2 Arquitetura do modelo de interoperabilidade baseado na abordagem de meta-modelos dinâmicos	93
5.3 Etapas para a interoperabilidade de metadados entre bases de dados heterogêneas ..	96
5.3.1 Obtenção dos modelos que serão integrados.....	97
5.3.2 Correspondência de linguagens	100
5.3.3 Correspondência de elementos dos modelos	101
5.3.4 Confirmação dos mapeamentos	104
5.3.5 Construção do meta-modelo	107
5.3.6 Operações sobre os meta-modelos.....	110
6 Avaliação Experimental.....	115
6.1 Prova de conceito.....	115
6.1.1 Módulo de Interface de administração	115
6.1.2 Módulo de Controle	117
6.2 Experimento de validação	118
6.2.1 Plano do Experimento	118
6.2.2 Execução do Experimento.....	122
6.2.3 Análise do Experimento.....	139
7 Considerações Finais.....	141
7.1 Contribuições do trabalho em relação ao estado da arte.....	141
7.2 Cenários de aplicação do presente trabalho.....	143
7.3 Perspectiva para trabalhos futuros	144
Bibliografia	145
Anexos	152
Anexo A – RDF/RDFS.....	153
Anexo B – Sintaxe do meta-modelo dinâmico numa representação em RDFS	155
a) BBC (TV-Anytime) e TV Brasil (PBCore)	155
b) Cinemateca e BBC (TV-Anytime)	158
c) Cinemateca e TV Brasil (PBCore)	160

Lista de Símbolos

XML : Extensible Markup Language

RDFS : Resource Description Framework Schema

OWL : Web Ontology Language

DAM : Digital Asset Management

SQL-DDL : Structured Query Language - Data Definition Language

ER : Entity-relationship model

UML : Unified Modeling Language

MOF : Meta-Object Facility

XMLS : XML Schema

EAI : Enterprise Application Integration

ERP : Enterprise Resource Planning

DOI : Digital Object Identifier

Lista de Figuras

Figura 1. Crescimento da receita dos sistemas de gerenciamento de ativos digitais nos últimos anos em milhões de dólares (GISTICS)	16
Figura 2. Número de indústrias que mais utilizam sistemas de gerenciamento de ativos digitais.....	16
Figura 3. Um registro de metadados seguindo o padrão PBCore2.0	19
Figura 4. Um registro de metadados seguindo o padrão TV-Anytime	19
Figura 5. Mapeamento das similaridades dos elementos presentes nos modelos PBCore 2.0 e TV-Anytime	19
Figura 6. Ilustração de um cenário prático contendo um exemplo de intercâmbio de conteúdos em sistemas de gerenciamento de ativos digitais distintos	21
Figura 7. Exemplo de comunicação entre nós dentro de um domínio (a); estrutura padrão de um nó(b).	26
Figura 8. Exemplo de inter-rede contendo dois domínios distintos	26
Figura 9. Arquitetura básica de um sistema de gerenciamento de ativos digitais (AUSTERBERRY, 2006).....	29
Figura 10. Visão geral dos blocos de construção de metadados	35
Figura 11. Ilustração da Meta-Object Facility (OMG, 2006a).....	39
Figura 12. Heterogeneidades de nos níveis de modelo e instância (HASLHOFER e KLAS, 2010).....	46
Figura 13. Formas de aplicação de ontologias	55
Figura 14. Ilustração do processo de projeção (AMIR, S. et al., 2011).....	64
Figura 15. Exemplo da representação do grafo.....	65
Figura 16. Fragmento exemplo em RDF	66
Figura 17. Etapas do processo de correspondência de metadados realizado pelo MuMle (S. Amir et al., 2010)	67
Figura 18. Representação gráfica dos casos de uso do Dynamo Framework.....	77
Figura 19. Arquitetura de implementação do Dynamo Framework.....	79
Figura 20. Visão Geral de Arquitetura do Dynamo Framework	81
Figura 21. Visão geral das classes presentes em cada módulo dentro componente Meta Model Processor.....	82
Figura 22. Classes envolvidas no módulo de controle.....	83

Figura 23. Classes que compõem o módulo de tradução.....	86
Figura 24. Classes que compõem o módulo de extensão	88
Figura 25. Diagrama de sequência para a confirmação das similaridades (a); Diagrama de sequência para os processos de coleta de informações dos domínios que serão integrados e cálculo das similaridades (a);.....	90
Figura 26. Arquitetura da camada conceitual do meta-modelo	93
Figura 27. Níveis de modelagem	95
Figura 28. Etapas do fluxo de execução do Dynamo Framework e os componentes utilizados em cada uma	97
Figura 29. Modelo de classes para representar o grafo processado pelo Dynamo Framework.....	99
Figura 30. Exemplo de um grafo para representar um determinado esquema de metadados	102
Figura 31. Estrutura detalhada de um nó de propriedade	103
Figura 32. Representação dos mapeamentos entre elementos e propriedades dos esquemas de metadados.....	104
Figura 33. Exemplo da relação de equivalência entre elementos de dois modelos	105
Figura 34. Exemplo da relação de inclusão entre elementos de dois modelos	106
Figura 35. Exemplo da relação de exclusão entre elementos de dois modelos	106
Figura 36. Exemplo da relação de exclusão entre elementos de dois modelos	107
Figura 37. Tela do editor de ontologias Neon Toolkit, mostrando as classes para a ontologia construída a partir do meta-modelo	108
Figura 38. Representação das hierarquias definidas entre as classes.....	110
Figura 39. Exemplo de uma expressão de consulta SPARQL	111
Figura 40. Exemplo de uma expressão de consulta SPARQL com parâmetros.....	111
Figura 41. Exemplo de conversão de consultas entre dois modelos distintos (BBC e TV Brasil) levando em consideração o meta-modelo gerado pelo Dynamo Framework.....	112
Figura 42. Tela da interface de administração do M-MAP.....	117
Figura 43. Padrão de URLs para acessar o M-MAP.....	117
Figura 44. Exemplo do registro de metadados de um conteúdo da TV Brasil	119
Figura 45. Exemplo do registro de metadados de um conteúdo do sistema da BBC	120
Figura 46. Exemplo do registro de metadados de um conteúdo da Cinemateca	121
Figura 47. Diagrama de atividades descrevendo as etapas da execução do experimento.....	122

Lista de Tabelas

Tabela 1. Seleção de linguagens de definição de esquema e suas características (HASLHOFER e KLAS, 2010)	41
Tabela 2. Quadro resumo dos projetos de interoperabilidade citados.....	61
Tabela 3. Regras de mapeamento de linguagens	65
Tabela 4. Regras de mapeamento de linguagens	66
Tabela 5. Descrição detalhada dos casos de uso essenciais do Dynamo Framework	78
Tabela 6. Relação entre os requisitos e os problemas a serem solucionados	79
Tabela 7. Métodos do Dynamo Framework utilizados para a etapa de definição dos modelos que farão parte da integração	99
Tabela 8. Seleção de trabalhos que abordam o mapeamento no nível de linguagem (M2)	101
Tabela 9. Descrição das classes do meta-modelo	109
Tabela 10. Métodos do Dynamo Framework que abstraem a manipulação das consultas sobre o meta-modelo RDF.....	112
Tabela 11. Controladores de Retaguarda do M-MAP.....	118
Tabela 12. Elementos processados do esquema da BBC baseado no padrão de metadados TV-Antyime	123
Tabela 13. Elementos processados do esquema da TV Brasil baseado no padrão de metadados PBCore	124
Tabela 14. Elementos processados do esquema da Cinemateca.....	125
Tabela 15. Total de elementos contendo ao menos uma correlação sugerida a cada par de modelos	126
Tabela 16. Resultado dos mapeamentos sugeridos do esquema da BBC (TV-Anytime) para a TV Brasil (PBCore)	127
Tabela 17. Resultado dos mapeamentos sugeridos a partir do esquema de metadados da TV Brasil (PBCore) para o da BBC(TV-Antyime).....	128
Tabela 18. Resultado dos mapeamentos sugeridos do esquema da TV Brasil (PBCore) para a Cinemateca	129
Tabela 19. Resultado dos mapeamentos sugeridos do esquema da BBC (TV-Anytime) para a Cinemateca.....	130

Tabela 20. Resultado dos mapeamentos sugeridos do esquema da Cinemateca para a BBC (TV-Anytime)	131
Tabela 21. Resultado dos mapeamentos sugeridos do esquema da Cinemateca para a TV Brasil (PBCore).....	132
Tabela 22. Exemplo de atribuição das relações de equivalência no mapeamento dos modelos	134
Tabela 23. Exemplo de atribuição da relação de inclusão no mapeamento dos modelos.	134
Tabela 24. Exemplo de atribuição da relação de exclusão no mapeamento dos modelos	135
Tabela 25. Exemplo de atribuição da relação de sobreposição no mapeamento dos modelos	135
Tabela 26. Análise do nível de acerto das sugestões de mapeamento entre os esquemas	136
Tabela 27. Definição dos modelos de origem e destino para a construção do meta-modelo	136
Tabela 28. Elementos usados para a construção do meta-modelo a partir dos esquemas da BBC (TV-Anytime) e TV Brasil (PBCore)	137
Tabela 29. Elementos usados para a construção do meta-modelo a partir dos esquemas da Cinemateca e BBC (TV-Anytime)	137
Tabela 30. Elementos usados para a construção do meta-modelo a partir dos esquemas da Cinemateca e TV Brasil (PBCore)	138

1

Introdução

A disseminação do uso do computador e da Internet em alta velocidade, juntamente com o barateamento de recursos de processamento e armazenamento, contribuíram para que a produção de conteúdos dentro de ambientes corporativos se tornasse cada vez mais baseada em arquivos digitais. As instituições passaram a utilizar arquivos digitais em vários departamentos, seja através da produção ou do consumo. À medida que esses conteúdos se propagam dentro de uma instituição, ou até mesmo fora dela, surgem novas necessidades, como a agregação de metadados (dados sobre os dados) adicionais para facilitar as buscas, o controle da distribuição e a atribuição de direitos autorais. A incorporação desses metadados faz com que as informações armazenadas deixem de ter o valor de simples arquivos digitais para assumirem o valor de patrimônio (ativos) em uma organização, facilitando inclusive o seu reuso. Os sistemas cujo papel é realizar a administração e exploração desses ativos são os sistemas de gerenciamento de ativos digitais (*Digital Asset Management - DAM*) (AUSTERBERRY, 2006).

Ativos digitais têm se tornado bens reutilizáveis e muitas empresas já fazem parcerias (comerciais ou não) para cessão de uso de conteúdos digitais (filmes, imagens, áudio, documentos), como é o caso de algumas redes de TV que disponibilizam seu acervo público para parceiros comerciais em regime B2B (*business-to-business*).

Devido a necessidades particulares e à existência de vários padrões, muitas vezes os metadados são organizados de forma distinta em diferentes instituições, tanto em relação à estrutura como à semântica, que acaba fazendo com que os DAM de instituições diferentes se isolem entre si, no sentido de que seus metadados são incompatíveis e dificultam buscas, compartilhamento, integração e reuso de conteúdos. Nesse cenário, para que dois sistemas de gerenciamento de ativos digitais distintos possam realizar operações em comum seria necessário um grande esforço manual, a fim de mapear as semelhanças entre os modelos.

Visando facilitar esse processo, algumas soluções foram propostas. Porém, as encontradas não realizavam uma especialização num cenário específico (como o gerenciamento de ativos digitais) e o foco era apenas num ponto do processo de integração (como a qualificação das similaridades entre em dois modelos).

Este trabalho propõe um framework de serviços para integração de metadados entre redes de sistemas de gerenciamento de ativos digitais. A proposta visa prover, através da utilização de meta-modelos dinâmicos (construídos sob-demanda com base nas compatibilidades entre os modelos), recursos necessários para contemplar três importantes problemas do processo de interoperabilidade: compatibilidade com diferentes formatos de armazenamento de dados ou interfaces de acesso (P1); integração de modelos de metadados heterogêneos (P2); e localização transparente (P3). Por ter se tornado um padrão bem aceito para arquiteturas descentralizadas, foi adotada a arquitetura orientada a serviços (SOA) para integrar cada fonte de metadados como um serviço independente. Como forma de demonstrar a aplicabilidade do framework proposto, é descrita uma solução semiautomática para contemplar as principais etapas do processo de interoperabilidade. Como resultante da utilização do framework proposto, espera-se integrar trabalhos já existentes na literatura, a fim de fornecer uma solução semiautomática (há participação de um usuário humano durante o processo) que possa contemplar os requisitos para que se possa obter a interoperabilidade.

1.1 Motivações

Para atingir efetivamente a interoperabilidade de metadados é preciso definir o escopo de atuação. Para este trabalho, o domínio de aplicação escolhido foi o gerenciamento de ativos digitais. Sendo assim, serão considerados modelos de metadados de sistemas desse tipo. Essa escolha foi motivada pelo crescimento da utilização dos sistemas de gerenciamento de ativos digitais em instituições de vários ramos, motivado pela rapidez e redução de custos para recuperação, edição, distribuição e reutilização de ativos.

A Figura 1 mostra o crescimento da receita obtida nos sistemas de gerenciamento de ativos digitais nos últimos anos. Destaca-se o crescimento contínuo a cada ano e que no ano de 2011 a barreira de um bilhão de dólares em receitas foi ultrapassada.

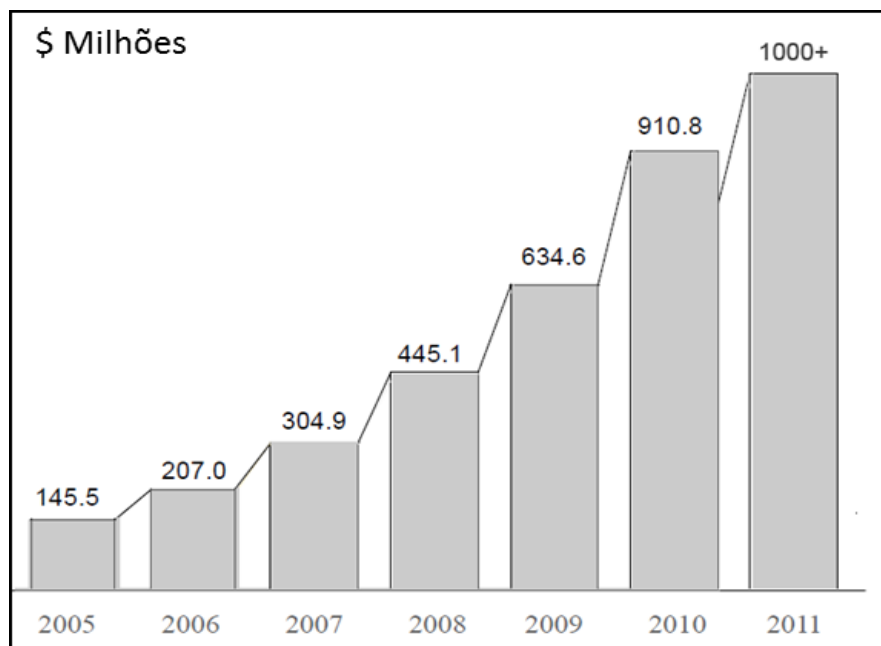


Figura 1. Crescimento da receita dos sistemas de gerenciamento de ativos digitais nos últimos anos em milhões de dólares (GISTICS)

A Figura 1 descreve as indústrias que mais utilizam sistemas de gerenciamento de ativos digitais. Observa-se que o uso atinge vários ramos de negócio, o que significa que uma grande variedade de indústrias poderão se beneficiar do resultado deste trabalho.

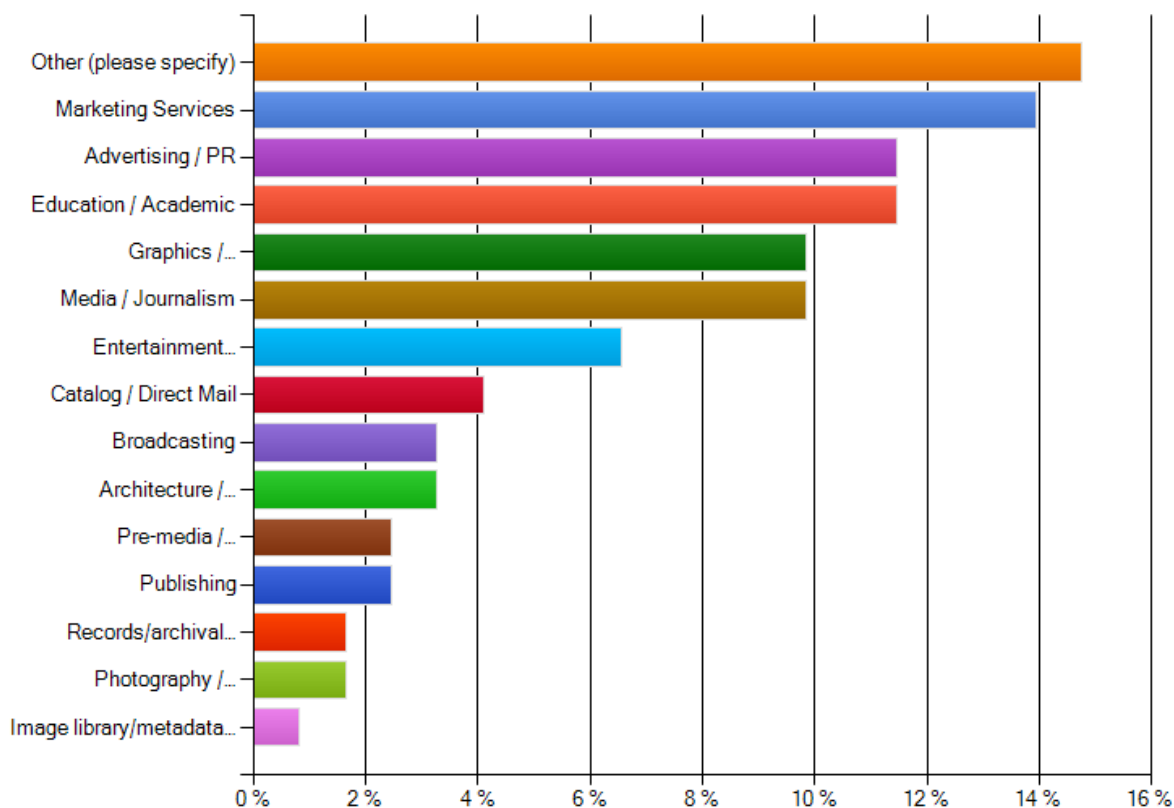


Figura 2. Número de indústrias que mais utilizam sistemas de gerenciamento de ativos digitais (GISTICS)

Em relação aos modelos de metadados, há diferenças quando se comparam modelos de instituições distintas por vários motivos. Do ponto de vista das instituições, há uma forte resistência para manter os sistemas já existentes e o formato dos metadados armazenados neles, na maioria das vezes motivadas por razões técnicas ou econômicas. Ainda, por fatores legais (e.g. gerenciamento de direitos autorais digitais), muitas instituições não pretendem exportar ou adaptar os seus sistemas às exigências de sistemas externos e, por isso, não buscam padrões interoperáveis ainda na concepção do modelo ou adotam soluções proprietárias (AUSTERBERRY, 2006). Por último, a existência de diferentes padrões e a possibilidade estendê-los também possibilita a diversidade de metadados tanto em estrutura quanto em semântica (HASLHOFER e KLAS, 2010).

Para integrar diferentes modelos de metadados, é necessário construir uma solução um nível acima das arquiteturas existentes, ao invés de tentar buscar a integração entre os diferentes modelos diretamente. A vantagem dessa abordagem é possibilitar a autonomia e independência de cada sistema, visto que não será preciso realizar nenhuma adaptação na arquitetura ou modelo de dados para que a integração seja possível.

A seguir são descritos os problemas que devem ser resolvidos por essa solução. Embora a maioria deles tenha a sua origem em problemas bem conhecidos de integração de dados padronizados (HASLHOFER e KLAS, 2010), aqui são investigados mais profundamente sob a perspectiva dos sistemas de gerenciamento de ativos digitais.

1.1.1 P1: Compatibilidade com formatos de armazenamento de dados ou interfaces de acesso

Um sistema de gerenciamento de ativos digitais pode armazenar metadados bem estruturados em um banco de dados relacional, manter descrições semi-estruturadas de metadados como documentos XML ou HTML, ou até mesmo armazenar metadados de forma completamente desestruturada, como por exemplo, em arquivos de texto no sistema de arquivos. Mesmo no cenário em que duas fontes de dados mantêm metadados estruturados, os modelos de dados adotados ainda podem diferir em sua utilização, devido às diferenças dos formatos dos dados. Dependendo das propriedades estruturais dos modelos, as fontes de dados podem ser acessadas através de interfaces diferentes, incluindo navegadores Web, linguagens de consulta estruturais (e.g. SQL), consultas não estruturadas (e.g. pesquisas em texto), ou interfaces de um domínio específico, tais como protocolos de intercâmbio de metadados (e.g. protocolo OAI-PMH (LAGOZE, 2002)).

Tendo em vista que este trabalho tem o objetivo de construir um framework de serviços que seja usado por aplicativos de alto nível para acessar metadados originados de diversas fontes, fornecer uma interface de consultas com base apenas em texto é insuficiente. O intercâmbio entre aplicações requer uniformidade sobre semântica, sintaxe e estrutura. Por isso, é preferível que as interfaces de acesso ofereçam uma linguagem de consulta mais expressiva, pelo menos semi-estruturada. Dentre propostas existentes está o RDF (Resource Description Framework) (W3C, 2004a), uma infraestrutura que possibilita a codificação, o intercâmbio e a reutilização de metadados. Trata-se de uma linguagem baseada em XML que impõe limitações estruturais à expressões semânticas, possibilitando a publicação de vocabulários legíveis para pessoas e computadores, além de proporcionar a reutilização e a distribuição dos metadados.

1.1.2 P2: Integração de modelos de metadados heterogêneos

No domínio de sistemas de gerenciamento de ativos digitais, modelos de metadados (*metadata schema*) definem a semântica dos metadados. Muitas instituições usam modelos padronizados como o Dublin Core (DUBLIN CORE, 2010) e o TV-Anytime (MCPARLAND, 2002), por exemplo, e outras aplicam modelos próprios e adaptados para suas necessidades específicas. Quando há mais de um modelo de metadados envolvido, é provável que ocorram conflitos semânticos. A abordagem comum para lidar com esse problema baseia-se na especificação de mapeamentos operacionais entre elementos dos modelos. Uma vez que a correspondência (semi-) automática é outro tema de pesquisas existentes, assumiu-se que as ferramentas e soluções de mapeamento estão disponíveis e foram referenciados trabalhos recentes (e. g. (AMIR, S. et al, 2011)) abordando este assunto. De fato, no domínio de sistemas de gerenciamento de ativos digitais, tais mapeamentos estão disponíveis pelo menos para os modelos padronizados (HASLHOFER e KLAS, 2010). A seguir são ilustradas as heterogeneidades semânticas entre dois registros de metadados de um conteúdo, correspondentes a modelos distintos.

A Figura 3 representa o registro de metadados sobre o documentário “Pelé Eterno”, usando elementos do padrão de metadados PBCore 2.0 (PBCORE, 2011), uma extensão do Dublin Core que adiciona elementos específicos para conteúdos audiovisuais.

Elemento	Valor
pbCoreIdentifier	123456789
pbCoreTitle	Pelé Eterno
pbCoreDescription	Conta a trajetória de Edson Arantes do Nascimento
pbCoreCreator	role=diretor creator=Anibal Massaini Neto

Figura 3. Um registro de metadados seguindo o padrão PBCore2.0

O outro registro descrito na Figura 4 armazena a mesma informação usando o padrão TV-Anytime. Nesse caso, o campo com a identificação “BasicDescription.Title.text” é usado para representar o título.

Elemento	Valor
ContentReference.CRID	crid://cinemateca.com.br/123456789
BasicDescription.Title	text=Pelé Eterno
BasicDescription.Synopsis	text=Conta a trajetória de Edson Arantes do Nascimento, o Pelé
BasicDescription.CreditsList	CreditsItem = { role=diretor PersonName={Anibal Massaini Neto} }

Figura 4. Um registro de metadados seguindo o padrão TV-Anytime

A Figura 5 mostra o resultado do mapeamento realizado entre os registros representados nos modelos PBCore e TV-Anytime descritos anteriormente.

Elemento PBCore	Elemento TV-Anytime
pbCoreIdentifier	ContentReference.CRID
pbCoreTitle	BasicDescription.Title.text
pbCoreDescription	BasicDescription.Synopsis.text
pbCoreCreator	BasicDescription.CreditList.creditsItem

Figura 5. Mapeamento das similaridades dos elementos presentes nos modelos PBCore 2.0 e TV-Anytime

Os sistemas clientes geralmente usam um determinado esquema de metadados para a formulação de suas consultas (*queries*). Se, por exemplo, uma aplicação fornece uma interface gráfica de consultas para TV-Anytime, todas as consultas irão conter elementos do modelo TV-Anytime. Logo, deve existir um componente na arquitetura responsável por traduzir as consultas que foram formuladas baseadas no esquema selecionado em consultas que estejam relacionadas às outras fontes específicas.

1.1.3 P3: Localização transparente e descoberta

Os metadados em si não precisam necessariamente ser replicados para outros sistemas e o acesso deve ser realizado mesmo que os metadados sejam mantidos nos sistemas das instituições que os originaram. Entretanto, para um cliente consultar vários sistemas, a localização, ou seja, a obtenção das informações necessárias para que os sistemas possam ser acessados (e. g. protocolo, endereço IP e porta), deve ser realizada

de forma transparente ao usuário. Assim, como pré-requisito técnico, podemos assumir que cada fonte de dados pode ser acessada através de um protocolo da Internet (e.g. HTTP).

O acesso transparente às fontes de dados descentralizadas exige a descoberta e a recuperação de fontes relevantes sem a intervenção do sistema cliente. Fatores como velocidade de conexão ou as capacidades técnicas (largura de banda, processamento) de uma fonte de dados podem afetar o tempo de resposta da consulta. O sistema de integração deve levar esses fatores em consideração no momento da formulação das consultas.

1.2 Contextualização

Como forma de ilustrar o contexto do trabalho, é descrito a seguir um cenário de utilização para o trabalho proposto, delimitando seu escopo e identificando em que ponto ocorrerá a sua contribuição.

Considerando um cenário composto por três instituições que lidam com gerenciamento de ativos digitais (TV Brasil, BBC e Cinemateca). Destaca-se que nesse cenário cada instituição possui uma organização distinta para o seu acervo de ativos: enquanto a TV Brasil utiliza como base o padrão de metadados PBCore (PBCORE, 2011), a BBC adota o padrão TV-Anytime (MCPARLAND, 2002) e a Cinemateca utiliza um modelo proprietário, ou seja, não segue como base nenhum padrão existente.

No ambiente citado, há a necessidade do intercâmbio de ativos digitais entre as instituições. Porém, como os modelos de dados dos sistemas adotados por cada instituição são incompatíveis, inicialmente esse intercâmbio não poderá ser realizado, até que alguma espécie de mapeamento entre os modelos seja realizada. É exatamente nesse ponto que a contribuição do framework de serviços proposto neste trabalho se encaixa.

A Figura 6 exibe um detalhamento maior dos modelos de dados de cada instituição e das operações realizadas. Neste caso, o filme cujo título é “Pele Eterna”, que pertence inicialmente à Cinemateca, foi compartilhado com a BBC, para tal, deve ser realizado o mapeamento entre os modelos da Cinemateca e BBC. O mesmo ocorre quando a TV Brasil faz uma pesquisa pelo mesmo conteúdo na BBC. Uma vez realizado o mapeamento (com o apoio dos serviços disponibilizados pelo framework), torna-se possível fazer consultas entre bases com modelos de metadados heterogêneos.

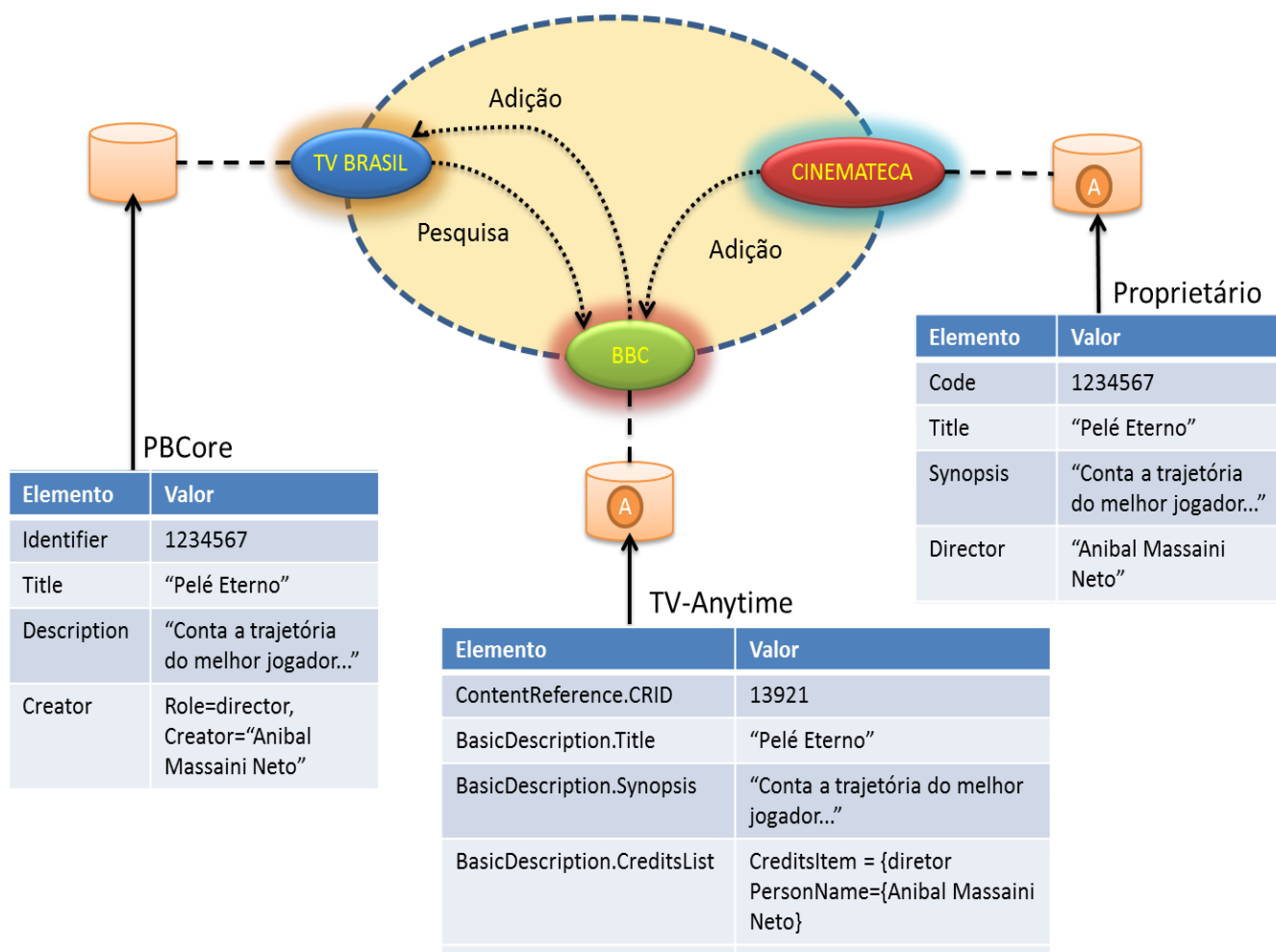


Figura 6. Ilustração de um cenário prático contendo um exemplo de intercâmbio de conteúdos em sistemas de gerenciamento de ativos digitais distintos

A contribuição do presente trabalho no cenário descrito seria inicialmente possibilitar apoio ao usuário sugerindo um mapeamento entre os modelos, e após a consolidação do mapeamento, realizar a conversão de consultas (leitura e escrita) entre as bases. Sendo assim, a ideia geral é encapsular os serviços e o processo para se alcançar a interoperabilidade num framework de serviços, que possa ser utilizado facilmente para construir aplicações cujo foco seja a integração entre bases de sistemas distintos.

1.3 Objetivos

1.3.1 Objetivo Geral

Propor um framework de serviços para possibilitar a interoperabilidade, no nível de metadados, entre bases de dados heterogêneas em redes de sistemas de gerenciamento de ativos digitais.

1.3.2 Objetivos Específicos

Para atingir o objetivo geral foram elencados os seguintes objetivos específicos:

- Avaliar as soluções existentes em nível de interoperabilidade, elencando quais poderiam ser reutilizadas para montar uma solução unificada que atenda a maior parte do fluxo de trabalho do processo de integração.
- Construir um framework de serviços que possua componentes capazes de fornecer apoio para a solução de três importantes problemas existentes no processo de integração: compatibilidade com formatos de armazenamento de dados ou interfaces de acesso; integração de modelos de metadados heterogêneo e localização transparente e descoberta.
- Validar o framework proposto através do desenvolvimento de uma prova de conceito aplicada ao contexto de gerenciamento de ativos digitais.

1.4 Metodologia

- A1: Investigar as principais características de redes heterogêneas de ativos digitais, visando chegar a um entendimento das questões de representação da informação que permeiam os problemas de integração de diferentes bases de dados;
- A2: Apresentar as principais iniciativas e trabalhos relacionados na área, além de tentativas de se obter classificação para os componentes de um projeto de interoperabilidade;
- A3: Construir um framework que possua componentes capazes de fornecer recursos que apoiem as principais etapas do processo de integração de bases heterogêneas;
- A4: Propor um modelo de interoperabilidade baseado na geração dinâmica de meta-modelos que possibilite o armazenamento das bases de dados integradas;
- A5: Desenvolver uma prova de conceito e avaliá-la através de um estudo de caso aplicado ao contexto de gerenciamento de ativos digitais, contendo repositórios com características distintas.

1.5 Estrutura da Dissertação

Este trabalho está dividido em 7 capítulos. Após a introdução, segue o Capítulo 2 que aborda a Fundamentação Teórica com os conceitos básicos das principais áreas que envolvem este trabalho: Inter-redes, o Gerenciamento de ativos digitais e Metadados.

No Capítulo 3 são apresentadas as principais características de redes heterogêneas de ativos digitais, com foco nos problemas de interoperabilidade de sistemas, expondo as heterogeneidades, identificando os tipos, dificuldades e o nível de integração necessário.

O capítulo 4 exibe uma descrição das principais pesquisas relacionadas a este trabalho.

No capítulo 5 é apresentada a proposta para integração entre sistemas heterogêneos de gerenciamento de ativos digitais através da construção de um framework.

No capítulo 6 é realizada a validação do trabalho, no qual é descrito o cenário de validação, as características do estudo de caso desenvolvido e a realização dos experimentos juntamente da exposição dos resultados obtidos.

O capítulo 7 contém as considerações finais e sugestões para trabalhos futuros.

2

Fundamentação teórica

Neste capítulo é apresentada a fundamentação teórica necessária ao entendimento desse trabalho: conceitos relacionados a modelos de interconexão de redes, gerenciamento de ativos digitais e metadados. A fundamentação levantada tem por objetivo subsidiar a definição da estratégia de interoperabilidade de metadados entre redes heterogêneas de gerenciamento de ativos digitais.

2.1 Inter-redes

2.1.1 Definição de Inter-rede

Uma inter-rede de computadores é uma interconexão entre várias redes. A Internet, do ponto de vista do usuário, é vista como uma única inter-rede. Porém, cada rede que constitui a inter-rede possui uma identidade única e mecanismos internos de comunicação. Assim, as redes que participam de uma inter-rede são chamadas de sub-redes (STALLINGS, 2002).

A interconexão entre sub-redes distintas é feita através de Sistemas Intermediários (Intermediate Systems – IS) (ISO/IEC, 1988), como os roteadores, que possuem como funções principais (STALLINGS, 2002):

- Prover uma ligação (link) entre redes;
- Prover roteamento e entrega de dados entre os processos dos sistemas finais localizados em diferentes redes;
- Prover as funções de ligação, roteamento e entrega de maneira que não sejam necessárias modificações na arquitetura de rede de nenhuma sub-rede conectada à inter-rede.

Essa última função implica que um Sistema Intermediário deve acomodar uma série de diferenças entre as redes, como:

- Endereçamento: redes podem possuir esquemas de endereçamento diferentes, por isso deve haver um endereçamento global;
- Tamanho dos dados: os tamanhos dos pacotes de dados de uma rede para a outra podem diferenciar em tamanho e por isso deve haver a possibilidade de fragmentação dos pacotes de dados para permitir a comunicação entre as redes;
- Interface: As redes podem apresentar hardware e software diferentes, assim, um roteador deve ser independente a essas diferenças;
- Confiabilidade: a operação de um roteador não deve se basear na premissa de que a rede é confiável.

Em suma, a função dos Sistemas Intermediários é prover conexão ignorando os detalhes sobre como os dados são repassados entre redes diferentes. Neste caso, o roteador possui interfaces que sabem se comunicar com cada protocolo de rede envolvida na interconexão. Além disso, a comunicação entre as redes é feita através de um protocolo padrão para transmissão de dados (e.g. Internet Protocol – IP) que funciona independentemente de qual a tecnologia de software e hardware presentes em cada sub-rede.

2.1.2 Organização de uma inter-rede

Uma das abordagens de organização para inter-redes é a divisão em domínios de forma independente. Nesse contexto, um domínio é uma sub-rede ou um conjunto de sub-redes, administradas por uma mesma instituição. Cada domínio é formado por um conjunto de nós (servidores e roteadores) e enlaces (responsáveis por interligar os nós). Dentro de um domínio, a comunicação entre nós ocorre através de um protocolo de roteamento inter-nós (ou intra-domínio), cujo papel é prover rotas entre os nós de origem e destino dentro de um domínio específico, como mostra a Figura 7.

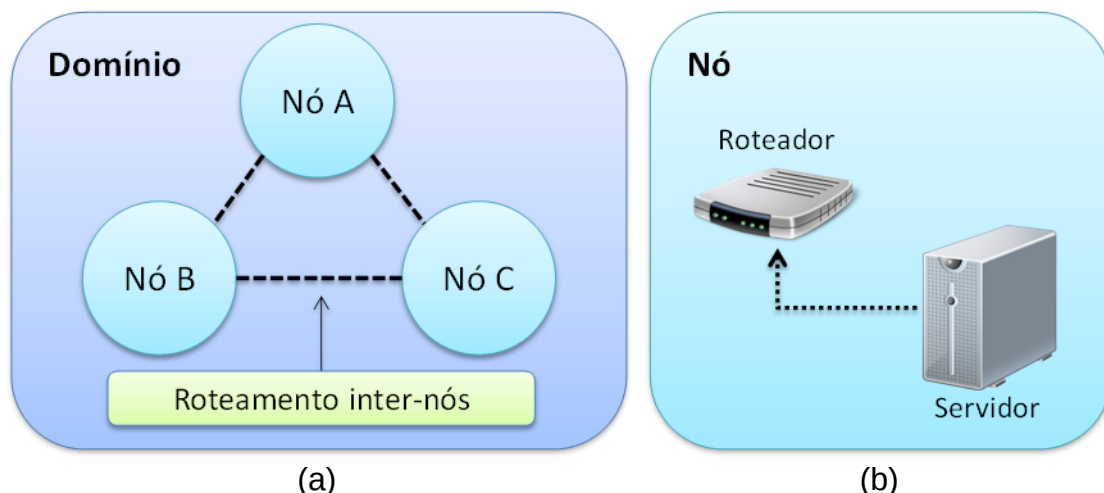


Figura 7. Exemplo de comunicação entre nós dentro de um domínio (a); estrutura padrão de um nó(b).

Podem-se considerar dois domínios como “vizinhos” quando há um ou mais enlaces entre nós de ambos os domínios (ver Figura 8). Isso possibilita a integração entre nós pertencentes a esses domínios. Para tal, é utilizado um protocolo de roteamento inter-domínios.

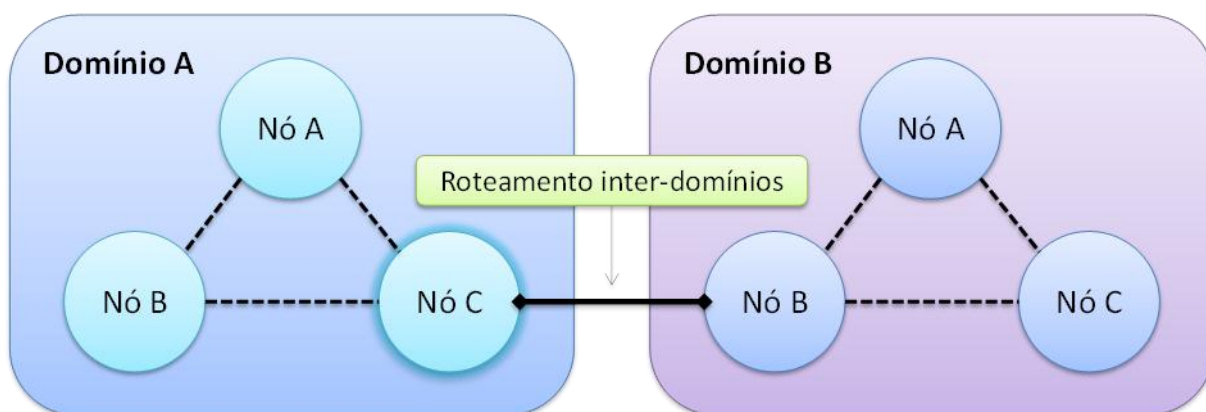


Figura 8. Exemplo de inter-rede contendo dois domínios distintos

2.2 Gerenciamento de ativos digitais

Como a produção de conteúdos dentro de ambientes corporativos tem se tornado cada vez mais baseada em arquivos digitais, a tendência é que as instituições passem a utilizar arquivos digitais em vários departamentos, seja produzindo ou consumindo. Porém, à medida que os ativos se propagam dentro de uma instituição, ou até fora dela, surgem novas necessidades, como a atribuição de restrições de uso, direitos autorais, enriquecimento dos metadados visando facilitar a busca, controle de distribuição e etc. Isso faz com que eles deixem de ter o valor de simples arquivos digitais para se tornarem um patrimônio de sua instituição de origem.

O papel desempenhado pelos sistemas de gerenciamento de ativos digitais (*Digital Asset Management* - DAM) é justamente realizar a administração e a exploração desses ativos (AUSTERBERRY, 2006).

2.2.1 Ativos digitais

Para que um conteúdo possa ser tratado como um ativo digital, é necessário que ele contenha informações sobre os direitos de uso de propriedade intelectual (AUSTERBERRY, 2006). Essa ligação entre conteúdo e os direitos de uso é realizada pelo DAM, através da adição de metadados adicionais ao conteúdo.

É importante ressaltar que o gerenciamento de ativos vai muito além do gerenciamento de direitos, que forma apenas uma pequena, mas vital parte da solução total. Outra característica importante do DAM é o provimento de um framework para possibilitar a comercialização ou venda de ativos digitais. Afinal, como qualquer patrimônio, o ativo digital também possui seu valor. Portanto, para que o acesso ao ativo seja estabelecido, além de possuir os direitos para sua utilização, o requerente precisará arcar com o seu valor, de acordo com o que for definido pela instituição que o provê.

2.2.2 Sistemas de gerenciamento de ativos digitais

Especificamente, o DAM fornece recursos para administrar e explorar ativos digitais dentro de grandes repositórios. Um sistema de gerenciamento de ativos digitais provê recursos e ferramentas para autores, editores e usuários finais das mídias.

Ativos digitais podem ser armazenados em diferentes formatos: áudio, vídeo, gráficos, imagens, documentos de texto, etc.; sendo possível inclusive realizar associações entre os ativos. Em relação à distribuição, os ativos podem ser vendidos, alugados ou até mesmo trocados.

Em um sistema de gerenciamento de ativos digitais encontramos, entre outros, os seguintes recursos (AUSTERBERRY, 2006):

- Armazenamento e arquivamento: utiliza sistemas de armazenamento transparente que permite crescimento e recuperação da base de dados de maneira eficiente;
- Múltiplos formatos e versões: deve ser possível armazenar vários formatos de conteúdo digital (vídeo, áudio, foto, livros, etc.) e suas versões;

- Catalogação e indexação: utiliza os metadados dos ativos digitais (autor, título, duração, entre outros) para catalogar e indexar os arquivos, o que facilita o gerenciamento;
- Busca: capacidade de buscar um conteúdo, tanto pelos seus metadados como por palavras-chave;
- Identificação Única: ao inserir um ativo, um identificador único é gerado, independente do ativo armazenado e permitindo a localização do local armazenado;
- Publicação: capacidade para publicação e compartilhamento do conteúdo armazenado, seus metadados e direitos;
- Direitos autorais: definem restrições de uso e direitos de propriedade intelectual;
- Contratos: gerenciamento dos termos que regem a distribuição e a forma de utilização dos ativos pelas redes participantes, de acordo com os direitos autorais definidos para cada conteúdo.

O gerenciamento de ativos é feito através de uma interface de gerenciamento que disponibiliza visualizações, buscas e a recuperação dos ativos digitais. Essa interface pode ser baseada na *Web* (acessada através de um navegador) ou *Desktop* (acessada a partir do computador do cliente).

O acesso baseado na *Web* abre a oportunidade para a visualização e autoria de conteúdos a partir de uma área de trabalho remota. Esse acesso pode ser liberado para qualquer ponto conectado à Internet ou pode ser restringido à rede local de uma instituição ou a uma rede privada. Já as interfaces *Desktop* possuem uma melhor integração com o sistema operacional e demais aplicativos que rodam na máquina do cliente, o que pode dar uma agilidade maior em cenários onde os conteúdos devem ser importados diretamente de outros programas que estão instalados na máquina do cliente, como por exemplo, *softwares* de edição de mídias.

A arquitetura de um Gerenciador de Ativos Digitais é apresentada na Figura 9. Toda a arquitetura é conectada através de uma rede de comunicação, e contém serviços específicos de armazenamento de conteúdo (*content repository*), armazenamento de metadados (*metadata*), lógica de negócios (*business logic*), proxies de aplicação (*media proxies*) e a camada de apresentação (*presentation layer*).

A camada de interface do usuário (*content user*), que é ilustrada na Figura 9 como baseada na *Web* (*web browser client*), interage com a camada de apresentação (*presentation layer*) recebendo ou enviando solicitações (*query*). Já a comunicação com os componentes de proxies de aplicação (*media proxies server*) e repositório de conteúdo (*content repository*) ocorre de forma unilateral, onde a camada de interface do usuário apenas recebe dados.

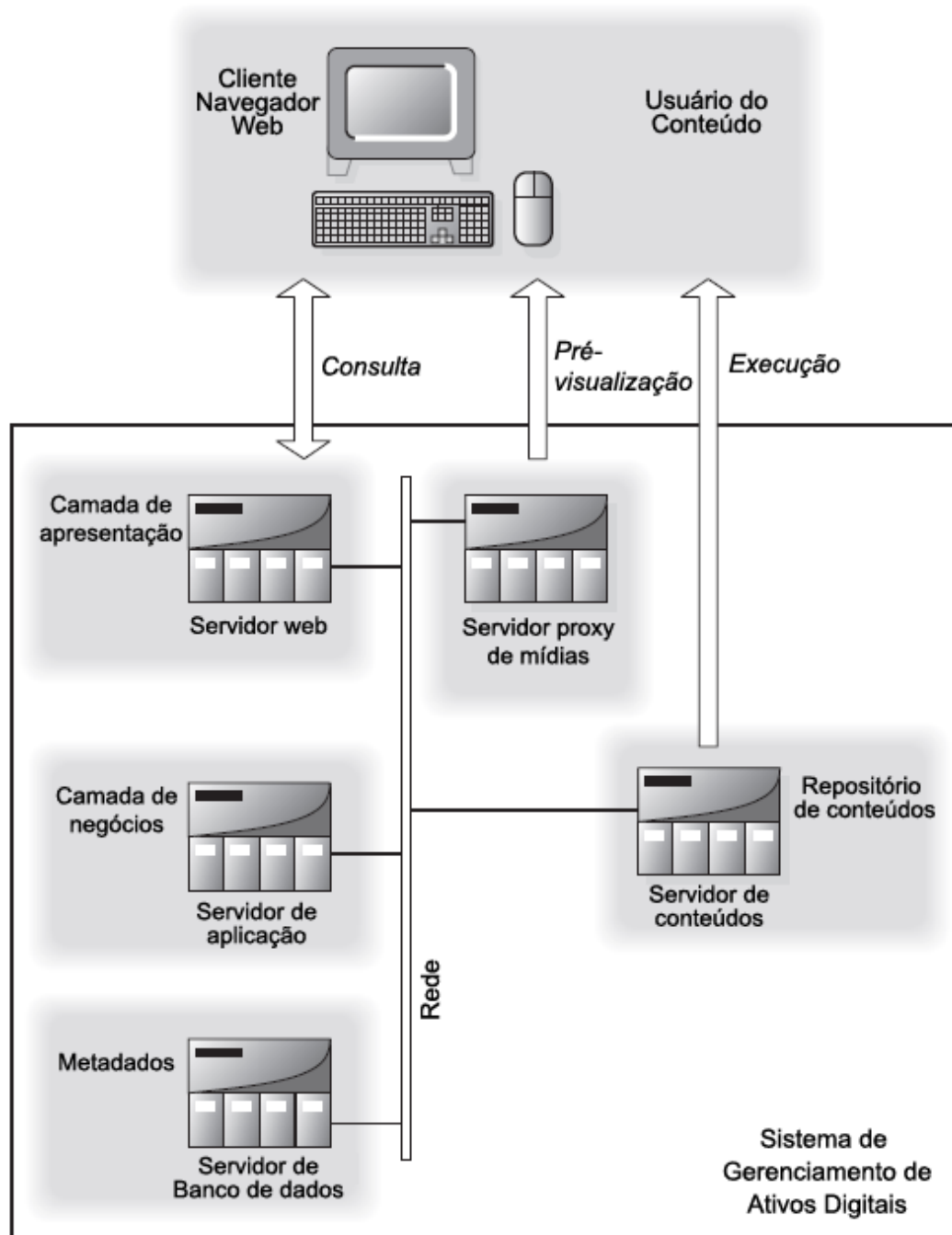


Figura 9. Arquitetura básica de um sistema de gerenciamento de ativos digitais (AUSTERBERRY, 2006)

2.3 Metadados

Metadados são informações descritivas sobre uma determinada fonte de dados, cujo uso é voltado a organizar, identificar, representar, localizar, interoperar e gerenciar esses dados. (BLANKEN, 2007).

No contexto do gerenciamento de ativos digitais, os metadados possuem o importante papel de descrever os ativos digitais. A descrição pode ocorrer em vários níveis, contendo desde as informações específicas sobre a mídia armazenada (e.g. formato, codec, resolução) até os direitos de uso do ativo.

2.3.1 Padrões de metadados

Muito da complexidade de um ativo digital vem do fato de que um conteúdo de mídia pode variar de tamanho, dificultando a transmissão e a busca; ou pode ser armazenado em um formato compactado, dificultando a extração da informação. Para trabalhar com essas complexidades inerentes aos conteúdos digitais, diversos padrões de metadados foram definidos, como MPEG-7, TV-Anytime e Dublin Core Metadata Initiative. Tais padrões são descritos a seguir.

2.3.1.1 MPEG-7

O MPEG-7 começou a ser desenvolvido em 1996 pelo Moving Picture Experts Group-MPEG, com o objetivo de ser um padrão para representação codificada de dados digitais de áudio e vídeo. O MPEG é aprovado pela International Organization for Standardization - ISO, sendo que os padrões anteriores (MPEG-1, MPEG-2 e MPEG-4) tinham como foco a codificação e compactação dos conteúdos (VICENTE, 2005). Já o MPEG-7 é formalmente conhecido como Multimedia Content Description Interface, ou interface para descrição de conteúdos multimídia, desenvolvido para que possa ser usado em grandes bases de material multimídia.

O objetivo do MPEG-7 é padronizar a descrição de conteúdo multimídia, permitindo a interoperabilidade entre sistemas e aplicações em geração, gerenciamento, distribuição e consumo de conteúdo audiovisual (CHANG, SIKORA e PURL, 2001).

O MPEG-7 é uma ferramenta que desde sua criação vem se consolidando, sendo utilizado nas práticas de descrição de conteúdo de objetos multimídia em sistemas de gerenciamento e recuperação da informação que servem a diferentes grupos de usuários,

e que têm apresentado resultados práticos satisfatórios quanto à eficiência do sistema (KAPELA et al, 2010).

O MPEG-7 possui uma estrutura que provê ferramentas de descrição (*Description Tools*), representadas por elementos de metadados. Esses elementos são empregados “para criar descrições que serão utilizadas por ferramentas com funções para pesquisar, filtrar e navegar de forma eficiente em conteúdos multimídia” (CHELLA, 2004). É possível utilizar as ferramentas de descrição para arquivos digitais ou analógicos, imagem em movimento ou foto, som, vídeo, ou seja, elas independem dos formatos dos arquivos. A utilização do formato acontece em sistemas com diferentes objetivos e estruturas, como acervos especializados (e.g. Sistemas de Gerenciamento de Ativos Digitais).

Devido à complexidade desse padrão, apenas alguns protótipos foram desenvolvidos em caráter científico e ainda existem poucas aplicações comerciais de criação, visualização e intercâmbio baseados no MPEG-7. A implantação do MPEG-7 exige que os responsáveis em gerenciar as informações recebam algum tipo de treinamento, e tenham algum conhecimento da linguagem XML. Esse processo gera um custo relacionado ao tempo de implantação e treinamento que precede a aplicação propriamente dita do modelo.

Em suma, o MPEG-7 oferece a vantagem de possibilitar a descrição de diversos aspectos dos objetos multimídia de forma bastante completa. Nesse aspecto, acredita-se que as complexidades e custos associados de implantação e utilização do MPEG-7 sejam compensadas pela garantia de qualidade de gerenciamento, interoperabilidade, armazenamento e recuperação de informações que esse modelo proporciona.

2.3.1.2 TV-Anytime

O TV-Anytime (MCPARLAND, 2002) é um padrão desenvolvido pelo TV-Anytime Forum (<http://www.tv-anytime.org>), uma associação global de organizações que desenvolvem especificações audiovisuais e outros serviços baseados no mercado de armazenamento em alta quantidade de conteúdos digitais em plataformas consumidoras.

Seu objetivo é adicionar às especificações gerais desses sistemas dois conceitos-chaves, chamados:

- Metadados padronizados
- Referência de conteúdo e resolução da localização

Os metadados no TV-Anytime são utilizados para descrever informações sobre programas de TV, programas de rádio e outros conteúdos do gênero. Tais informações consistem do que é fornecido por usuários humanos (e.g. título, gênero, etc.) e/ou do que é fornecido por máquina (tipo de áudio, razão de aspecto, etc.). O principal uso dos metadados do TV-Anytime é atrair usuários para os conteúdos e proporcionar que os usuários recebam informações detalhadas dos que lhe forem de interesse. Através da adoção de metadados num formato padronizado, o usuário torna-se apto a encontrar um novo conteúdo facilmente e utilizá-lo em outros repositórios que adotem o mesmo padrão.

A especificação da referência do conteúdo apresenta um jeito padronizado de encontrar (fazer referência para) um pedaço de um conteúdo. Nesse contexto, "conteúdo" pode referenciar um programa de TV individual, ou um largo grupo de programas que estão relacionados de alguma forma. A cada uma dessas entidades é atribuída um Identificador de Referência do Conteúdo (Content Reference Identifier - CRID). Pelo uso do CRID, é possível um usuário capturar o conteúdo pelo resultado do processo que realiza a resolução da localização. Este processo possibilita que um dispositivo capture um programa ou suas repetições, independentemente de qual canal o veiculou ou do horário que foi transmitido.

Em resumo, o TV-Anytime é um padrão aberto para descrição de um conteúdo e de como adquiri-lo. As ideias do Identificador de Referência do Conteúdo (CRID) e do fornecimento de metadados padronizados são ferramentas poderosas para o armazenamento e a localização eficiente de conteúdos. A expectativa do padrão é possibilitar, através das ideias expressas na sua especificação, que os usuários encontrem facilmente conteúdos que eles estejam interessados e possam armazenar em seus próprios repositórios ou até mesmo compartilhar com outros, desde que utilizem o mesmo padrão de metadados.

2.3.1.3 Public Broadcasting Metadata Dictionary Project (PBCore)

Organizado como um conjunto de campos especificados que podem ser usados em aplicações de banco de dados, o PBCore (PBCORE, 2011) é utilizado como um modelo de dados para os meios de catalogação e sistemas de gestão de ativos digitais. Como um esquema, ele permite o intercâmbio de dados entre acervos de mídia, sistemas e organizações.

O padrão PBCore baseia-se no padrão Dublin Core, com a adição de elementos úteis para mídias digitais. Desde o seu lançamento inicial em 2005, PBCore tem sido adotado por muitos usuários em meios de comunicação públicos, bem como acervos de filmes, instituições acadêmicas e outros acervos audiovisuais.

O PBCore encontra-se na versão 2.0, cujo lançamento foi realizado em janeiro de 2011. Nessa versão, os usuários avançados são capazes de tirar proveito dos novos atributos opcionais dentro dos elementos, que pode especificar fontes, taxonomias e partes de uma mídia. Com essas melhorias, o PBCore possui a pretensão de se tornar muito mais poderoso como um formato para aplicações web semânticas que envolvem meios de comunicação.

Plataformas em todo o sistema de comunicação pública utilizam diferentes bases de dados e ferramentas para gerenciar mídia e metadados: sistemas de trânsito, servidores de playout (transmissão de vídeo), sistemas de gerenciamento de ativos digitais, etc. O PBCore fornece uma maneira de agregar e normalizar os metadados a partir de diferentes sistemas para tornar possível a fácil localização, o compartilhamento e a reutilização em diferentes plataformas.

Muitos sistemas e ferramentas utilizadas em portais de mídia online e repositórios digitais confiáveis estão adotando o PBCore como um padrão de troca de metadados. Com o lançamento do PBCore 2.0, as oportunidades de intercâmbio de metadados e conteúdo com esses sistemas vai crescer. Ao adotar uma abordagem baseada no PBCore para gerenciar ativos de mídia, significa a possibilidade de fácil integração com outros sistemas que utilizam o mesmo padrão.

2.3.2 Classificação dos metadados

Metadados podem ser classificados quanto as suas funções como descritivos, estruturais, administrativos e de direitos autorais (NISO, 2004; JOHNSTON, 2004). Quanto seu nível de abstração semântica, são classificados como metadados de baixo-nível ou alto nível (WESTERMANN, 2003). Metadados de baixo nível têm menor valor para os usuários finais do que os de alto nível, que são semanticamente ricos, pois descrevem entidades semânticas como numa narrativa de mundo real, contendo objetos, eventos, conceitos, estados, lugares e tempo, juntamente de seus atributos e relações (BENITEZ et al., 2001). Por isso, a qualidade semântica e a expressividade dos metadados são essenciais para a recuperação efetiva de ativos digitais.

A construção dos metadados consiste principalmente nos elementos de alto-nível, que contêm informações semanticamente ricas como, por exemplo, o nome de pessoas ou a descrição de um ativo. Enquanto os elementos de baixo-nível consistem em metadados mais técnicos como, por exemplo, a largura de uma imagem ou a quantidade de quadros por segundo de um vídeo.

Existem algumas características comuns que podem ser observadas na construção de metadados: cada descrição é construída a partir de um conjunto de elementos descritivos (e.g.: título, gênero) e seus respectivos valores (e.g. “O auto da compadecida, “Comédia Romântica”). Tais elementos são definidos como parte de um esquema de metadados (*metadata schema*), que pode ser padronizado, como é o caso do PBCore e TV-Anytime, citados na Seção 2.3.1, ou proprietário, termo usado para designar um modelo de metadados que não segue nenhum padrão previamente definido. A especificação dos elementos citados anteriormente pode ser realizada utilizando uma determinada linguagem. Para XML, a linguagem utilizada normalmente é XML Schema (W3C, 2006), para o caso de um banco de dados relacional, um esquema é representado na forma de tabelas usando SQL-DDL (ISO/IEC JTC 1/SC 32, 2003).

Baseado nas características descritas podem-se identificar três principais blocos presentes na construção dos metadados: a) os valores dos elementos descritos, representados como instância de metadados (*metadata instance*); b) as definições de elementos, representadas como esquema de metadados (*metadata schema*); e c) a linguagem para definição dos esquemas de metadados (*schema definition language*). A Figura 10 ilustra esses três blocos e suas respectivas dependências. Em seguida cada bloco é detalhado.

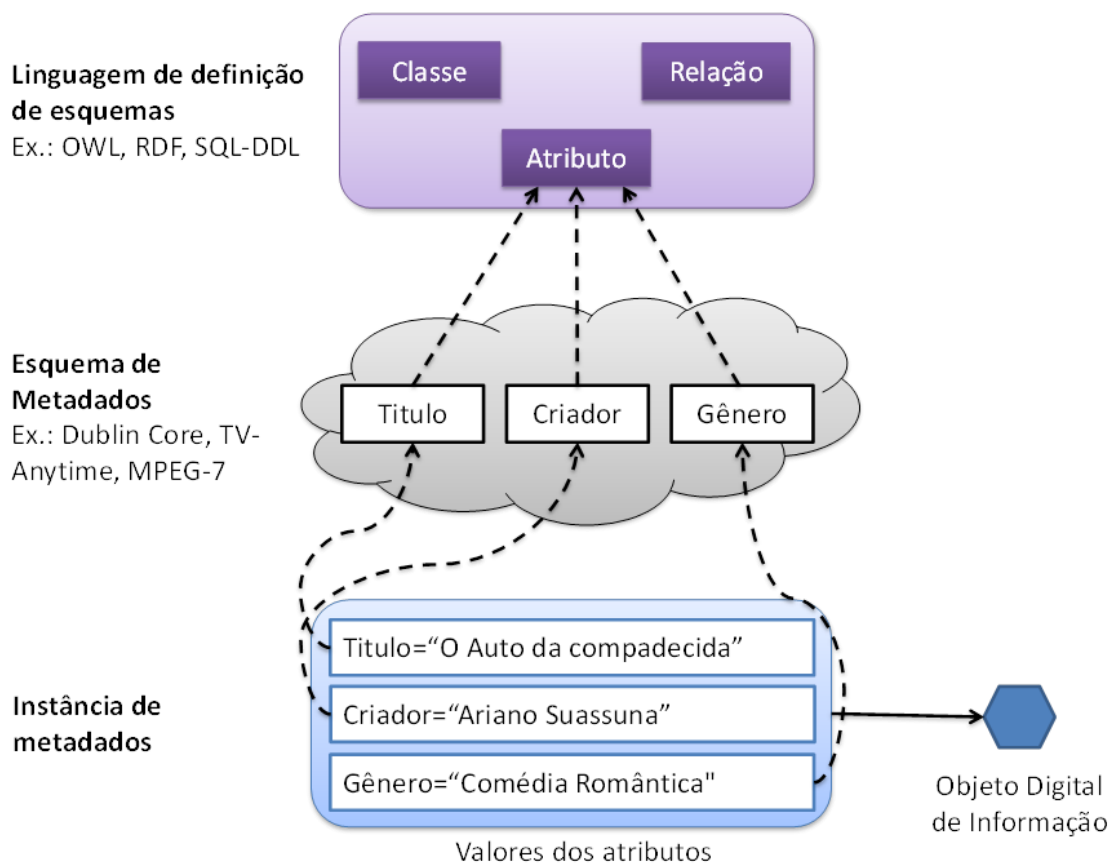


Figura 10. Visão geral dos blocos de construção de metadados

2.3.2.1 Linguagem de definição do esquema de metadados

Um metadado de domínio específico é expresso dentro de uma estrutura padrão, cuja construção é feita através de uma linguagem de definição de esquema. Cada linguagem provê um conjunto de elementos primitivos (ex.: classes, atributos, relações etc), que devem ser constituídos não apenas de construções sintáticas, mas também de definições semânticas, de forma a possibilitar que as linguagens sejam compreendidas por máquina.

A definição semântica do termo *linguagem* já implica que, para que seja possível estabelecer uma comunicação com outras, deve existir um acordo no significado dos elementos primitivos. Esse é o caso para as linguagens de definição de esquemas: normalmente, define-se uma linguagem padrão ou ao menos algum tipo de consenso em relação ao significado dos elementos primitivos. Exemplos de linguagens utilizadas para especificar esquemas são: XML Schema (W3C, 2006), SQL-DDL (ISO/IEC JTC 1/SC 32, 2003), RDF Vocabulary Description Language Schema (RDFS) (W3C, 2004a), Web Ontology Language (OWL) (W3C, 2004b) e Unified Modeling Language (UML) (OMG, 2007b).

2.3.2.2 Esquema de metadados

O segundo bloco de construção de metadados trata da definição dos elementos, denotado como esquema de metadados. Um esquema de metadados trata-se basicamente de um conjunto de elementos com uma definição semântica precisa, que possam ser opcionalmente conectados por alguma estrutura (RAHM e BERNSTEIN, 2001). A semântica de um esquema é definida pelo significado de seus elementos. Um esquema de metadados é utilizado para definir os nomes dos elementos de um modelo, juntamente de seus significados, e opcionalmente de regras que definem como os valores do conteúdo devem ser formulados.

É importante ressaltar que o termo “esquema de metadados”, que teve origem no domínio de banco de dados, é nomeado de forma diferente em outros contextos ou comunidades. Por exemplo, no campo de organizações de conhecimento e informação, tende-se a usar o termo “vocabulário de metadados” para o que chamamos de esquema de metadados. Essas comunidades tendem a considerar metadados mais de um ponto de vista linguístico do que técnico. Apesar destas diferentes percepções, acredita-se que o termo “esquema de metadados” é apropriado para os propósitos desse trabalho, visto que é considerada a interoperabilidade de metadados principalmente de uma perspectiva técnica.

2.3.2.3 Instância de metadados

O próximo bloco de construção de metadados diz respeito ao conjunto de valores atribuídos aos metadados, denominado instâncias de metadados. As combinações de valores e elementos formam a descrição de algum objeto de informação. Como fora mencionado anteriormente, as regras para descrição de metadados são impostas de acordo com esquema relacionado. Se há total correlação entre o que foi descrito e o esquema, pode-se dizer que a instância de metadados corresponde ao esquema.

2.3.3 Representação de metadados em sistemas de informações

Metadados são construídos a partir de objetos de informação que são projetados para serem persistidos, recuperados e intercambiados entre sistemas que lidam com informações. A representação dos metadados nesses sistemas é definida através de modelos de informação divididos em vários níveis. Num sistema de informação típico, podem-se identificar ao menos quatro níveis: físico, lógico, programação/representação e conceitual (HASLHOFER e KLAS, 2010).

No nível mais baixo, o nível físico, os metadados consistem de bits e bytes que são representados na memória, escritos nos discos e transmitidos pela rede. Os componentes do sistema que trabalham nesse nível estão preocupados principalmente com a otimização da alocação de memória, com a atribuição de registros a partições do sistema de arquivos e com a construção de índices para uma recuperação eficiente. Desse modo, o modelo de informação nesse nível compreende conceitos como formatos, arquivos, registros e índices.

O nível lógico funciona como uma abstração para o nível físico, de forma que torna-se possível omitir os metadados de mais baixo nível dos desenvolvedores de aplicações. Como exemplo de aplicação presente no nível lógico e lida diretamente com o nível físico estão os Sistemas de Gerenciamento de Banco de Dados (SGBDs). Nesse caso, as instâncias de metadados são descritas como dados contidos em tabelas. Assim, além de prover a transparência necessária para o nível físico, o nível lógico organiza a eficiência dos metadados (e.g., via normalização) e introduz recursos importantes para os desenvolvedores de aplicações, como garantia da consistência de dados, controle de concorrência e transações.

No nível de programação/representação, os elementos que fazem parte do esquema conceitual são manifestados ou apresentados das seguintes formas: a) esquemas de metadados como código numa determinada linguagem, refletindo o domínio da aplicação através de classes ou tipos, enquanto a descrição dos metadados se dá através do preenchimento de objetos em tempo de execução; ou b) elementos do modelo de metadados e seus conteúdos codificados usando uma linguagem de marcação como XML. Em ambos os casos, as descrições dos metadados devem se adaptar a determinados modelos de documentos e esquemas de metadados, de acordo com a linguagem de programação de modelagem (e.g. XML Schema, DTD) escolhida. A seleção dessa linguagem deve levar em consideração a compatibilidade com a tecnologia utilizada para a programação/representação dos elementos (e.g. XML, Java).

Um esquema de metadados no nível conceitual assemelha-se a entidades do mundo real com suas propriedades e relacionamentos. Linguagens comuns para a criação de modelos conceituais são a ER (Entity-Relationship Model) ou a UML (Unified Modeling Language) (OMG, 2007b).

2.3.3.1 Perspectiva dos níveis de modelos de metadados

Todos os níveis de representação de metadados citados anteriormente têm em comum o fato de que os elementos descritivos de um determinado esquema, isto é, os elementos e seus relacionamentos, são implementados na forma de um modelo de dados. Tal modelo encapsula os elementos definidos de um esquema, os representa semanticamente de maneira formal, e provê uma sintaxe para serializar as descrições de metadados. A interpretação semântica de um modelo de metadados é dada através do mapeamento dos elementos do modelo às entidades correspondentes em um determinado domínio de aplicação. A sintaxe de um esquema de metadados (elementos e regras que indicam como os valores devem ser atribuídos aos elementos) é definida usando uma determinada notação, que pode ser simbólica (e.g. palavras ou fórmulas), textual (e.g. sentenças descritas em uma linguagem natural, ou seja, falada ou escrita por humanos) ou gráfica (e.g. diagramas).

As linguagens de definição de esquemas estão presentes em cada nível e são usadas para expressar modelos de informação. O núcleo de cada linguagem é seu meta-modelo - também chamado de meta-modelo de metadados. O meta-modelo contempla a sintaxe abstrata da linguagem (e.g. definição de elementos e atributos), juntamente das notações concretas e semânticas (e.g. restrições de tamanho, padrões ou tipos de dados toleráveis). Sendo assim, pode-se considerar que a interpretação de uma linguagem se dá através do mapeamento dos elementos definidos no meta-modelo (sintaxe abstrata) para os elementos primitivos de uma linguagem (sintaxe concreta). As definições semânticas do modelo podem ser representadas em vários graus de formalidade, desde a descrição em linguagem natural a linguagens formais ou matemáticas. Especialmente do ponto de vista de interoperabilidade, definições rígidas e semânticas possibilitam uma interpretação consistente que podem facilitar a integração com outros sistemas (SEIDEWITZ, 2003).

Modelos de metadados e meta-modelos estão organizados em níveis diferentes, mas encontram-se alinhados aos níveis de informação mencionados anteriormente. No nível mais baixo podem-se encontrar as descrições de metadados, que são instâncias válidas de um modelo de metadados (ex.: Classes Java, modelo UML, relações de bancos de dados) que refletem os elementos de um determinado esquema de metadados. O modelo de metadados em si é uma instância válida de um meta-modelo de metadados, sendo por sua vez, parte de uma determinada linguagem de definição de esquema. Devido a essa abstração, é possível criar meta-modelos que representam

metadados (ex.: instâncias de metadados de um modelo UML também podem ser representadas como instâncias de um meta-modelo UML).

A especificação MOF - *Meta Object Facility* - (OMG, 2006a) oferece uma definição para esses níveis distintos: M0 é o nível mais baixo, o nível das instâncias de metadados (e.g. definição de valores dos atributos de um modelo), M1 contém a estrutura do modelo para uma aplicação particular, ou seja, os esquemas de metadados (e.g. definição dos campos que compõem o modelo) são modelos M1. Linguagens de modelagem residem no nível M2, sua sintaxe abstrata ou meta-modelo pode ser considerada como um mecanismo de modelagem particular (e.g. a definição de um atributo primitivo da linguagem). No nível mais alto, M3, podem ser encontradas as linguagens de modelagem universais na qual os sistemas de modelagem são especificados (e.g. estruturas básicas, tipos primitivos). A Figura 11 ilustra os quatro níveis de abstração da Meta-Object Facility (MOF).

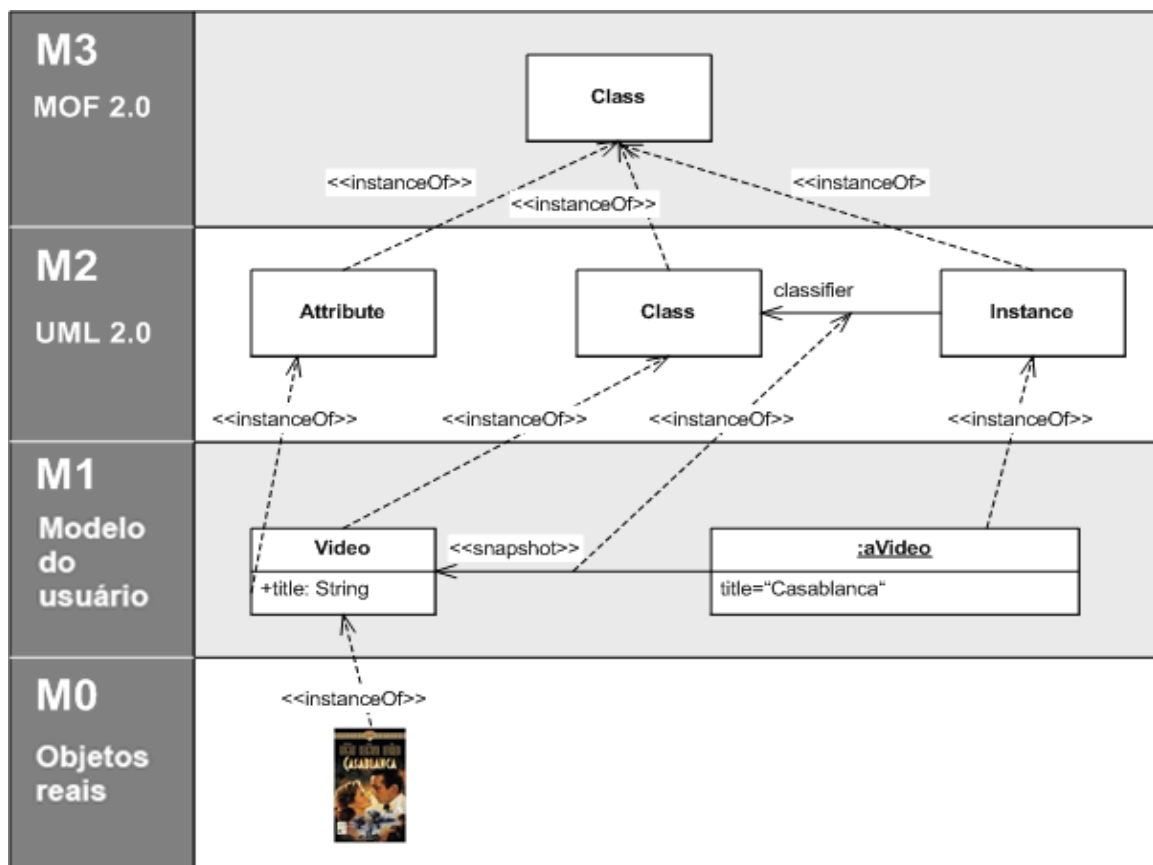


Figura 11. Ilustração da Meta-Object Facility (OMG, 2006a)

No que diz respeito aos níveis de abstração citados anteriormente, um ponto que pode ser questionado é como a sintaxe abstrata de linguagem de modelagem no nível M3 (meta-meta-modelo) é expressa. No geral, há duas opções de definição de um meta-modelo: usar um mecanismo diferente de modelagem (ex.: gramáticas livres-de-contexto

para linguagens de programação, semântica formal ou um conjunto explícito de axiomas¹ e regras de deduções para linguagens baseadas em lógica descritiva² ou usar sua própria linguagem de modelagem de forma recursiva. Nesse último caso, os elementos de um meta-modelo são expressos na mesma linguagem no qual o meta-modelo foi descrito (e.g. utilizar XML Schema para descrever o esquema da própria linguagem XML Schema). Isso é chamado de *meta-modelo reflexivo* (SEIDEWITZ, 2003), visto que os elementos de uma linguagem de modelagem podem ser descritos usando um conjunto mínimo de elementos da mesma linguagem.

2.3.4 Linguagens de definição de esquemas

A Tabela 1 mostra uma visão geral de um conjunto selecionado de linguagens de definição de esquemas que podem ser atribuídas ao nível M2. Para cada linguagem, é descrita a sua sintaxe concreta, ou seja, como os elementos da linguagem são representados.

Além disso, expõe-se como a semântica é definida para cada linguagem, assumindo que as opções disponíveis são: linguagem natural, semântica formal (i.e., definições matemáticas precisas), implementações de referência e suítes de teste. Finalmente, descrevem-se as representações internas de uma linguagem de modelagem: seu meta-modelo ou sintaxe abstrata. É considerado também o tipo do meta-modelo (e.g., orientado a objetos ou baseado em conexões semânticas) e qual a categoria de construção do modelo (estrutural ou comportamental) suportada. Adicionalmente, é fornecida uma visão geral dos elementos principais e um esboço se o modelo pode ser definido reflexivamente.

A Linguagem de Modelagem Unificada (*Unified Modeling Language* – UML) (OMG, 2007b) e o modelo Entidade-Relacionamento (ER) (CHEN, 1976) são usados para a modelagem conceitual de sistemas de software, portanto empregam uma notação sintática gráfica. RDFS (W3C, 2004a), e OWL (W3C, 2004b) são linguagens conceituais que suportam a modelagem de esquemas de metadados em termos de conexões semânticas, que requerem que sua representação interna seja baseada em grafos.

¹ Princípio evidente, que não precisa ser demonstrado; máxima, sentença; norma admitida como princípio.

² Lógicas descritivas são linguagens de representação do conhecimento construídas especialmente para expressar conhecimento através de conceitos e hierarquias.

Java é uma representação para a parte de programação do nível de programação/representação. Sua sintaxe e semântica são definidas por Gosling et al. (GOSLING, 2005); desde a versão 6 do Java, a sua API também compreende um meta-modelo (JAVA COMMUNITY PROCESS, 2006; BRACHA e UNGAR, 2004) que reflete os comportamentos primitivos e estruturais da linguagem Java. Outra linguagem de modelagem para a definição de esquemas de metadados é XML Schema. Sua semântica em linguagem natural, tal como seu meta-modelo hierárquico (sintaxe abstrata) são definidos pela W3C (W3C, 2006).

Tabela 1. Seleção de linguagens de definição de esquema e suas características (HASLHOFER e KLAS, 2010)

		Conceitual				Programação / Representação		Lógica	
		UML	ER	RDF/S	OWL	Java SE	XML Schema	SQL-DDL	Lógica Descritiva
Sintaxe Abstrata (meta-modelo)	Tipo de modelo	Orientada a objetos	Grafos dedicados	Grafos rotulados baseados em conexões semânticas	Grafos rotulados baseados em conexões semânticas	Orientado a Objetos	Hierárquico	Modelo relacional	Lógico
	Construção do modelo	Estrutural e comportamental	Estrutural	Estrutural	Estrutural	Estrutural e comportamental	Estrutural	Estrutural (comportamental)	Estrutural
	Principais elementos de modelagem	Classes, Atividades, Iteração, Casos de uso, Estados, Implantação e Componentes.	Conjunto de entidades, atributos e relacionamentos	Grafo, Propriedade, Literal, Recurso, Classe, Tipo de Dado	Ontologia, Classe, Tipo de Dado, Objeto, Propriedade, Restrição	Classe, Interface, Campo, Método, Anotação, Enum, Pacote	Tipos de dados, Tipo Simples, Tipo Complexo, Elemento, Atributo	Relação, Atributo, Esquema, Tabela, Domínio	Frases, Termos, Sentenças
	Definição reflexiva	Sim (via nível M3)	Não	Sim	Sim	Não	Sim	Não	Não
Sintaxe concreta		Notação gráfica	Notação gráfica	RDF/XML	RDF/XML	Sintaxe Java	XML	Sintaxe SQL	Sintaxe de Lógica Descritiva
Semântica		Linguagem natural	Formato	Formato	Formato	Implementação natural da linguagem	Linguagem natural	Formato	Formato

2.3.5 Distinção entre modelos e meta-modelos

Para esclarecer a diferença entre modelos e meta-modelos, citados até aqui quase que indistintamente, algumas considerações devem ser feitas. De modo geral, um meta-modelo apresenta as definições e os relacionamentos para um modelo, descrevendo-o. Em geral a meta-modelagem é baseada em quatro camadas (OMG, 2006a):

- A camada de objetos (do usuário) ou M0, que consiste da informação que se deseja descrever (normalmente chamada de “dado”);

- A camada de modelo ou M1, que consiste de metadados que descrevem as informações que compõem o modelo de dados;
- A camada de meta-modelo ou M2, que consiste da descrição da estrutura e da semântica dos metadados. Um meta-metadado (metadado que descreve outro metadado) é agregado aos meta-modelos. Pode-se imaginar um meta-modelo como uma “linguagem” para descrever diferentes tipos de modelos;
- A camada meta-meta-modelo ou M3, que é composta de descrições da estrutura e semânticas de meta-meta-dados. Em outras palavras, é uma linguagem para definir diferentes tipos de metadados.

Um meta-modelo é uma definição precisa das construções e regras necessárias para criar modelos semânticos. Um meta-modelo é sempre construído para um propósito em particular e não é razoável tentar entendê-lo sem conhecer o objetivo para o qual foi criado.

Pelo menos em teoria, é possível se ter “meta-meta-meta-modelos” (ou M4, M5...) e assim por diante. Isso não é necessário na maioria dos casos, mas abre a possibilidade de descrição da estrutura do conhecimento até o nível desejado.

Para uniformizar a utilização do termo no contexto desse trabalho, adota-se a seguinte definição: um meta-modelo é um modelo que descreve outros modelos a partir de regras necessárias para especificar sua semântica. O objetivo do meta-modelo é descrever os componentes de um modelo e seus relacionamentos.

Este trabalho desenvolve um framework de serviços que utiliza, para o gerenciamento dos modelos integrados, uma abordagem de construção dinâmica de meta-modelos, na qual pretende lidar com o significado dos metadados disponíveis para um sistema e com seus relacionamentos, objetivando a interoperabilidade entre fontes distintas. Os meta-modelos serão apresentados em sua forma final na sintaxe RDF/RDFS (W3C, 2004a).

3

Interoperabilidade de metadados em ambientes heterogêneos

Quando é tratada a interoperabilidade de sistemas em ambientes digitais diversificados, pode-se lidar com diferentes perspectivas (sistêmica, sintática, estrutural, semântica). A complexidade do assunto envolve muito mais do que formatos diferentes de dados, como se pode imaginar numa abordagem inicial. O formato dos dados se inclui como parte do problema de incompatibilidade de metadados, mas forma apenas uma parcela do total que é necessário tratar.

Este capítulo apresenta uma visão geral sobre o conceito de interoperabilidade de metadados, com foco na sua aplicação em ambientes heterogêneos. Na Seção 3.1 são discutidos conceitos ligados à interoperabilidade de metadados, como a definição e os tipos existentes. Na Seção 3.2 são apresentadas heterogeneidades que dificultam a interoperabilidade. Na Seção 3.3 é exposto como identificar até que nível de heterogeneidade deve ser considerado. Finalmente, na Seção 3.4.2 são citadas as pesquisas existentes nesta área e com base nelas é apresentada uma classificação dos elementos importantes para um modelo que vise a interoperabilidade.

3.1 Conceitos ligados à interoperabilidade de metadados

Quando se busca interoperabilidade deve-se levar em consideração que há uma grande diversidade de características que se manifestam em vários níveis no ambiente de interação entre sistemas. Para que um sistema consiga se integrar a outro de características distintas, é preciso detectar as diferenças e de alguma forma contorná-las.

Considerando uma perspectiva técnica, as diversidades entre sistemas são bastante amplas. Podem-se encontrar diferentes arquiteturas, modelos, plataformas, linguagens de programação, padrões de armazenamento de dados, formatos de arquivos e mídias de armazenamento (HASLHOFER e KLAS, 2010). Essas características acrescentam

complexidade ao processo de interoperabilidade, visto que os dados armazenados podem ser de variados tipos, como vídeo, áudio, imagem e texto. Além de que a estrutura dos dados armazenados pode seguir diferentes modelos ou paradigmas (orientado a objetos ou estruturado, como os bancos de dados; semiestruturados, como documentos XML; e não estruturados, como textos simples). Uma característica importante é que os dados armazenados representam informações que devem estar num formato adequado para a leitura humana, entretanto, muitas vezes a interpretação semântica dessas informações não consegue ser realizada facilmente por softwares. Isso torna ainda mais complexa a automatização do processo.

O grau de incompatibilidade se torna ainda maior quando se tratam de sistemas de diferentes organizações. O uso de várias linguagens de programação, onde algumas estão em desuso (e.g. COBOL, FORTRAN), os diferentes sistemas operacionais (e.g. Windows, Linux, MacOS) e as diferentes características que possibilitam a conectividade em rede, são algumas razões para a incompatibilidade.

3.1.1 Níveis de interoperabilidade

No contexto de sistemas de informações, a interoperabilidade literalmente denota à habilidade de um sistema utilizar ou trabalhar com partes de outros sistemas. Na literatura podem-se encontrar definições similares: Baker et al. (BAKER et al., 2002) elencam vários pontos de vista de interoperabilidade, como o potencial dos metadados para atravessar fronteiras entre sistemas em diversos contextos. Outros autores definem interoperabilidade como “a troca de metadados entre dois ou mais sistemas com ou sem perda mínima de informação e sem nenhum esforço especial em qualquer sistema” (NISO, 2004) ou como a “habilidade para aplicar uma sintaxe única de consulta sobre informações expressas em múltiplos formatos descritivos”. (HUNTER E LAGOZE, 2001).

A noção de interoperabilidade ainda pode ser dividida: para Baker et al. (BAKER et al., 2002), obter a interoperabilidade é um problema para ser resolvido em três níveis principais:

1. Nível de transporte e troca (ex.: protocolos);
2. Nível de representação de metadados (ex.: ligação sintática, linguagem de codificação);

3. Nível de metadados com seus atributos (ex.: estrutura definida no esquema de metadados) e valores (ex.: vocabulários controlados).

Sheth (SHETH, 1998) citado por Ouksel (OUKSEL e SHETH, 1999), apresenta quatro classes principais de preocupações com interoperabilidade:

1. Interoperabilidade de sistemas: lidar com heterogeneidades do sistema, como plataformas incompatíveis;
2. Interoperabilidade sintática: lidar com os aspectos de representação dos dados em nível de leitura por máquina;
3. Interoperabilidade estrutural: lidar com estruturas e modelos de dados;
4. Interoperabilidade semântica: interpretar o significado semântico das informações armazenadas;

Tolk (TOLK, 2006) propõe outra visão, que consiste em seis níveis:

1. Interoperabilidade Técnica: comunicação estabelecida através da infraestrutura;
2. Interoperabilidade sintática: estrutura comum para intercâmbio de informação;
3. Interoperabilidade semântica: modelo de informação comum;
4. Interoperabilidade programática: sensibilidade ao contexto;
5. Interoperabilidade dinâmica: habilidade de compreender mudanças de estado;
6. Interoperabilidade conceitual: modelo totalmente especificado, mas independente de implementação.

Baseado no significado literal, nas definições da literatura e considerando as classificações de metadados descritas na Seção 2.3.2, pode-se definir interoperabilidade de metadados como uma propriedade qualitativa dos objetos de informação, que permite que as informações sejam usadas e compartilhadas de forma homogênea em diferentes sistemas e aplicativos.

Com essa definição, é possível distinguir claramente a concepção da interoperabilidade de metadados, que busca resolver os problemas no nível da representação de informação, envolvendo desde a infraestrutura de comunicação e hardware até a incompatibilidade de plataformas de software.

Soluções para os níveis de interoperabilidade citados requerem abordagens específicas para cada nível que se procura atingir. Assim, torna-se importante esclarecer que tipos de interoperabilidade se procuram resolver, quais problemas se pretende solucionar e quais níveis de interoperabilidade são necessários no contexto em estudo. Tal assunto será abordado nas próximas seções.

3.2 Heterogeneidades que dificultam a interoperabilidade

As heterogeneidades que devem ser eliminadas para prover interoperabilidade de metadados já foram identificadas em pesquisas recentes. A Figura 12 mostra uma classificação das heterogeneidades predominantes mencionadas na literatura, considerando uma perspectiva baseada num modelo central (HASLHOFER e KLAS, 2010).

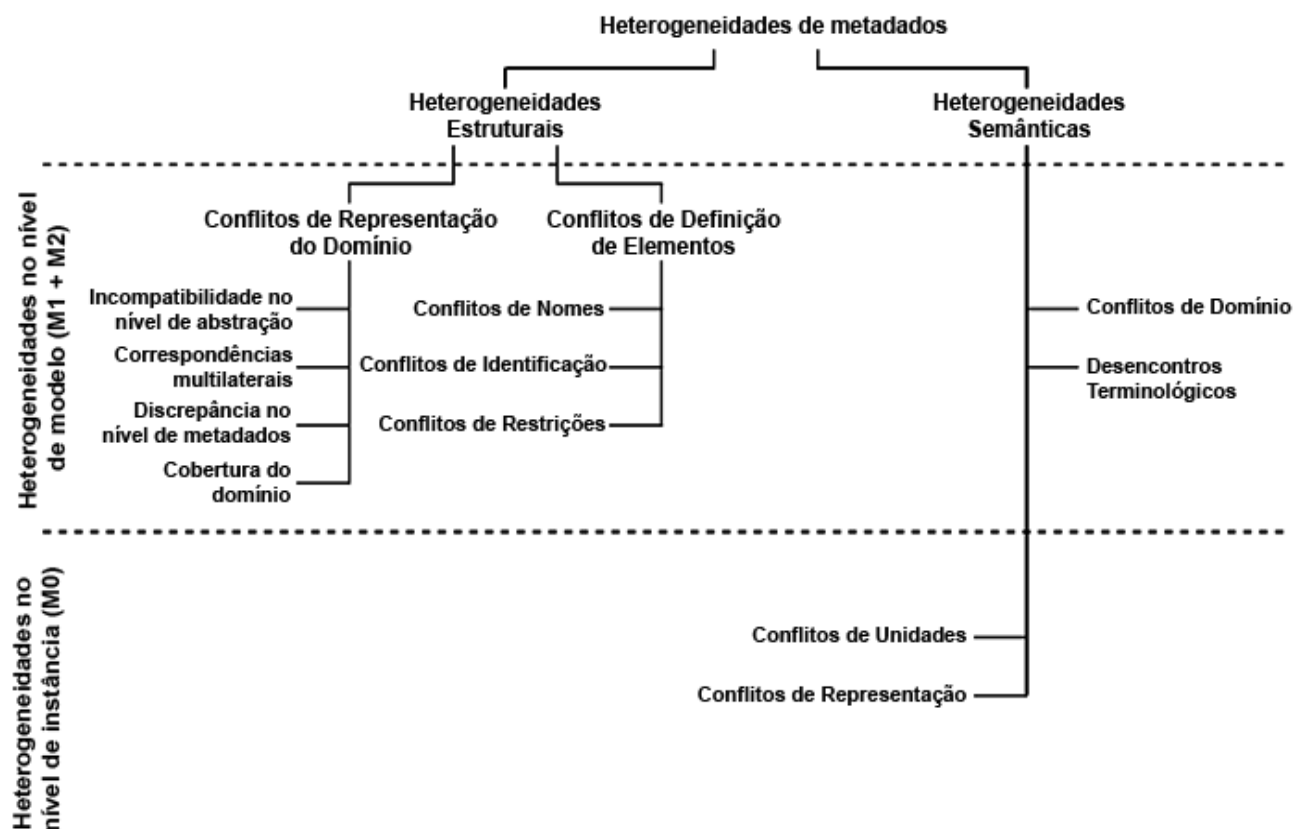


Figura 12. Heterogeneidades de nos níveis de modelo e instância (HASLHOFER e KLAS, 2010)

Destaca-se que há uma relação de instanciamento entre instâncias de metadados, esquemas de metadados e linguagens de definição de esquemas, ou seja, metadados são instâncias de modelos de metadados e modelos de metadados são instâncias dos meta-modelos de metadados. Generalizando esses relacionamentos, a Figura 12 distingue as heterogeneidades entre o nível de modelo e o nível de instância, como uma primeira

divisão de classificação. Na segunda divisão, que contempla as heterogeneidades em nível de instância, apresentam-se duas classes: estrutural, causada pela distinção de propriedades que forma a estrutura dos modelos; e semântica, que ocorre por causa dos conflitos na interpretação do significado dos elementos do modelo ou de valores dos conteúdos nos diversos contextos de interpretação.

3.2.1 Heterogeneidades estruturais

Heterogeneidades estruturais no nível de modelo ocorrem graças às incompatibilidades presentes na forma de representar a estrutura dos modelos. Um modelo principal consiste de seus elementos atômicos (ex.: entidades, atributos, relações) e uma combinação ou conjunto desses elementos formam uma determinada estrutura para representar um domínio de interesse particular.

Assim sendo, podem-se agrupar as heterogeneidades estruturais de acordo com as ocorrências em dois níveis distintos: nos conflitos de definição de elementos, que a ocorrem a partir das definições dos elementos do modelo (e.g. nome, identificação, restrições), e nos conflitos de representação de domínio, que ocorrem porque elementos específicos de um determinado domínio podem ser representados de várias maneiras diferentes. A seguir serão analisados os dois grupos de heterogeneidades estruturais.

3.2.1.1 Incompatibilidade no nível de abstração

A incompatibilidade no nível de abstração diz respeito ao grupo de conflitos no nível de representação do domínio da aplicação. Aparece quando a mesma entidade de mundo real é organizada em diferentes hierarquias ou é agregada de forma diferente em elementos do modelo. Um exemplo para esse tipo de conflito no nível M2 é a capacidade de definir atributos e relações em várias linguagens: enquanto o modelo ER (que utiliza os termos *attribute* e *relation*) e OWL (que utiliza os termos *datatypeProperty* e *objectProperty*), definem elementos primitivos para ambas as características (atributo e relacionamento), XML Schema (que usa o elemento *attribute*) e Java (que usa o elemento *field*) agrupam essas características num único primitivo. Incompatibilidades no nível de abstração em M1, o nível dos modelos de metadados, ocorrem, por exemplo, quando um modelo de metadados atribui o criador de um recurso digital a uma simples entidade “Criador”, como é o caso do padrão de metadados Dublin Core, enquanto outros modelos como o TV-Anytime e também o MPEG-7 fazem distinção entre “Pessoas” e “Organizações”, ambos sendo uma especialização do conceito “Criador”.

3.2.1.2 Correspondências multilaterais

As correspondências multilaterais tratam-se de outro conflito de representação do domínio da aplicação. Resulta diretamente das incompatibilidades em nível de abstração mencionadas anteriormente. Em cada nível, um elemento em um modelo pode corresponder a múltiplos elementos em outro modelo, e vice-versa. Por exemplo, pode haver correspondência entre o campo “Autor” do Dublin Core e os campos “Nome” e “Sobrenome” do TV-Anytime. Isto ocorre porque na descrição de metadados do padrão TV-Anytime, esses elementos (nome e sobrenome) podem ser sendo usados no contexto de “créditos de autoria da mídia”, assumindo então a representação do “Autor”.

3.2.1.3 Discrepância no nível de metadados

As discrepâncias no nível de metadados são conflitos de representação de domínio que ocorrem porque determinado elemento do modelo não tem qualquer correspondência direta com os elementos do outro modelo. Isso, entretanto, não significa necessariamente que o outro modelo não poderá capturar a mesma informação sobre um determinado domínio. Conceitos do mundo real representados como elementos em um modelo (e.g. “autor” como atributo) podem ser modelados de maneira diferente em outro modelo (e.g. “autor” como entidade) ou até serem capturados como conteúdo de um elemento do modelo em nível de instância (e.g. parte do campo descrição contém o nome do autor).

3.2.1.4 Cobertura do domínio

O problema de cobertura do domínio ocorre quando não há correspondências entre os elementos existentes no modelo. Isso acontece quando um conceito do mundo real refletido em um modelo foi deixado de fora no outro modelo, embora ambos os modelos tenham sido projetados para o mesmo domínio semântico. Por exemplo, num modelo o campo “informações” de uma imagem descreve quem está na foto, enquanto em outro modelo o mesmo campo “informações” descreve o tamanho e a resolução da imagem.

O fato da nomenclatura do atributo ser similar, mas a informação contida ser semanticamente distinta é o que diferencia a o problema da cobertura do domínio da discrepância no nível de metadados, visto que nela os elementos possuem estruturas diferentes (atributo ou entidade), mas mantêm o mesmo valor semântico.

3.2.1.5 Conflitos de nomes

Os conflitos de nomes são conflitos que ocorrem quando elementos do modelo descrevem a mesma entidade, mas possuem nomes diferentes. No nível de as linguagens de definição de esquema (M2), meta-modelos distintos podem atribuir nomes diferentes aos conceitos primitivos que são utilizados na linguagem. No UML, por exemplo, define-se o conceito primitivo *Classe*, enquanto no modelo ER (Entidade-relacionamento) usa-se *Entidades* para representar o mesmo tipo de conceito do mundo real. Da mesma forma, no nível M1, modelos distintos de metadados podem-se atribuir nomes diferentes a elementos que representam os mesmos conceitos. Por exemplo, num dado modelo, a descrição de um elemento pode ser representada pelo rótulo de "informações", em um segundo modelo por "sinopse" e em um terceiro por "descrição".

3.2.1.6 Conflitos de Identificação

Os conflitos de identificação tratam-se de um caso especial de conflitos de nomes que envolvem a identificação única das entidades do modelo. No nível M2, dependendo da linguagem usada para definir o modelo, as entidades são identificadas apenas pelo seu nome (e.g. "entidade" no modelo ER e "tabela" no modelo SQL-DDL) ou por algum identificador único (ex.: "id" no XMLSchema ou OWL). Conflitos de identificação também podem aparecer no nível M1, onde podem existir, por exemplo, modelos cujos elementos possuem identificação única através de *namespaces* e outros no qual os elementos, mais especificamente suas tabelas e colunas, são identificadas apenas por nomes simples, aumentando o risco de ocorrência de ambiguidade.

3.2.1.7 Conflitos de restrições

Conflitos de restrições são os conflitos que ocorrem porque modelos distintos proveem diferentes possibilidades de definição de restrições. Um exemplo para uma restrição no nível M2 é a habilidade de um esquema importar características de outros esquemas (e.g. herança de tipos). Isso é possível em linguagens como XML Schema ou OWL, mas não em ER. Chaves primárias incompatíveis, referências conflitantes (e.g. atributos similares fazendo referência para elementos diferentes) ou restrições de domínio (e.g. unicidade, obrigatoriedade) podem causar o mesmo tipo de conflito no nível de esquemas de metadados M1.

3.2.2 Heterogeneidades semânticas

Heterogeneidades semânticas no nível de modelo são conflitos que ocorrem por causa das diferenças na semântica dos modelos. Lembrando que a semântica de um modelo é definida pela semântica de domínio e o mapeamento das entidades do domínio para elementos do modelo. O domínio semântico provê o significado para cada elemento do modelo e pode conter expressões de linguagem, no caso das linguagens de definição de esquema, ou entidades do mundo real no caso de modelos de metadados.

3.2.2.1 Conflitos de domínio

Os conflitos de domínio ocorrem quando os domínios se sobrepõem, são sobrepostos, ou são agregados a outros, ou quando são incompatíveis. Um exemplo desse tipo de conflito no nível M2 é o nível de abrangência da expressividade das linguagens; com linguagens que possuem conjunto variado de recursos, torna-se possível modelar coisas que não são expressas com outras linguagens menos poderosas. Com OWL, por exemplo, é possível expressar que duas classes são equivalentes ou que uma classe é a união de duas outras classes. Em outras linguagens como XML Schema ou Java não há construções nas linguagens para indicar tais relacionamentos. Obviamente, conflitos de domínio podem ocorrer entre os modelos de metadados no nível M1. Por exemplo, se um modelo reflete o domínio de pagamento eletrônico e outro de conteúdos multimídia, é improvável que haja qualquer correspondência significativa entre esses modelos.

3.2.2.2 Desencontros terminológicos

Os desencontros terminológicos tratam-se de outro tipo de heterogeneidade que ocorre em dois níveis: conflitos de sinônimos, quando o mesmo conceito é mapeado em elementos com nomes diferentes e conflitos de homônimos, quando conceitos diferentes são mapeados em elementos com o mesmo nome.

Um exemplo de conflito de sinônimo seria o uso de termos distintos para expressar o mesmo conceito semântico, como o uso dos termos distintos “Autor” e “Criador” para expressar o mesmo significado.

Já um exemplo de conflito de homônimo no nível de linguagem seria o conceito de polimorfismo ou sobrecarga presente em linguagens orientadas a objeto como Java. Essas linguagens suportam funções polimórficas nas quais os seus parâmetros podem conter vários tipos. Os tipos podem ser organizados numa hierarquia de subtipos e as assinaturas

de métodos podem ser sobrecarregadas, ou seja, o mesmo método quando utilizado pode gerar comportamentos semanticamente diferentes. Em Java, por exemplo, num cenário contendo uma classe abstrata denominada “Operação”, com um método com a seguinte assinatura: “calcular(int valor1, int valor2)”, e contendo também as classes “Soma” e “Subtração” como especialização para “Operação”, ao executar o método “calcular” cuja assinatura está presente na classe abstrata, podemos ter uma operação de soma ou subtração de acordo com a subclasse que foi instanciada.

3.2.2.3 Conflitos de unidades

Heterogeneidade semântica que ocorre no nível de instância de metadados (M0), quando diferentes sistemas de medida são usados para mensurar informações associadas ao conteúdo. Por exemplo, as dimensões de uma determinada imagem podem ser representadas em pixels por um modelo e em centímetros por outro. Mesmo se as imagens possuísem semanticamente a mesma dimensão, os valores representados seriam diferentes.

3.2.2.4 Conflitos de representação

São conflitos que resultam do uso de diferentes formatos para representação dos valores das informações associadas aos conteúdos. Por exemplo, dois valores de data, que semanticamente são iguais, podem ser representados de formas diferentes em cada sistema (e.g. data=01.01.2012 ou data=2012/01/01).

3.3 Nível de integração necessário para a interoperabilidade

De modo geral, os problemas que envolvem interoperabilidade de metadados entre sistemas heterogêneos se relacionam diretamente à grande diversidade de sistemas, à semântica distinta entre aplicações, às diversas granularidades possíveis de se encontrar numa fonte de informação, ao tamanho das aplicações e à evolução constante dos sistemas.

Mais especificamente, as heterogeneidades têm origem, principalmente, na diversidade semântica. As informações estão armazenadas em diferentes formatos, esquemas e vocabulários. Além da maneira pela qual a informação é acessada.

Em projetos de integração, nem sempre se faz uso completo de uma fonte de informação. Esta é a origem dos problemas relativos às granulometrias³ de informações possíveis ao se realizar a integração com uma nova fonte. Componentes internos (por exemplo, partes de bancos de dados) devem ser extraídos e integrados para se obter a informação necessária.

Finalmente, o tamanho dos sistemas atuais e a constante evolução das fontes de informação adicionam mais complexidade ao processo: torna-se necessário que muitas fontes sejam consultadas, interpretadas e coordenadas ao se propor um sistema integrado. Além disso, é obrigatório que isso seja feito de forma dinâmica, em função da constante evolução dos sistemas e das fontes de dados que os alimentam.

O objetivo de um projeto de integração, ou seja, os resultados que se pretende obter, os propósitos e os fatores externos (e.g. recursos e tempo) que limitam esse projeto, são os maiores determinantes do nível de integração necessário. Mas, pode-se afirmar que o problema de integrar sistemas diversos se apresenta em várias dimensões, não estando restrita apenas à recuperação da informação. Para se conseguir uma integração efetiva devem-se executar tarefas relacionadas à recuperação, interpretação, manipulação dos dados e compartilhamento. Só assim um projeto de integração poderá agregar valor ao trabalho das pessoas, integrando fontes internas e externas, de diferentes formatos, com diferentes formas de acesso e em dimensões distintas de tempo e espaço.

Ao se procurar entender que nível de integração é necessário e quais são as possibilidades para se estudar a interoperabilidade, pode-se também utilizar algum tipo de classificação, com a finalidade de reduzir a complexidade do problema e facilitar sua compreensão. Algumas propostas são citadas na literatura para tal classificação e serão apresentadas na seção seguinte, com o objetivo de possibilitar melhor entendimento dos componentes que são comuns em um projeto de integração. Com esta classificação será possível confrontar duas ou mais soluções de interoperabilidade, e, através de métricas, escolher a mais viável.

³ Refere-se a tamanho básico das unidades que podem ser manipuladas; habilidade de conceitualizar o mundo em diferentes níveis de complexidade.

3.4 Pesquisas sobre interoperabilidade

As abordagens que buscam interoperabilidade são variadas e podem também ser consideradas técnicas combinadas. Existem pesquisas sobre a interoperabilidade, provenientes de diferentes comunidades acadêmicas. A Seção 4 (Trabalhos Relacionados) destaca alguns destes trabalhos, realizados por pesquisadores da área de bancos de dados, de bibliotecas digitais, de recuperação da informação, dentre outros. Em cada uma dessas áreas há abordagens diversificadas para a solução do problema.

Com a popularidade dos sistemas abertos e aplicativos que trabalham fora dos limites da organização, tornou-se relevante o estudo de uma camada de software que execute a integração dos componentes distintos e oculte a heterogeneidade inerente aos sistemas. Essa camada é o *middleware*, que faz a intermediação entre os aplicativos e o sistema operacional. A interoperabilidade nas empresas, que consiste na integração de processos através de sistemas de informação é estudada por uma nova disciplina, denominada *Enterprise Application Integration* (EAI).

3.4.1 Ontologias e Interoperabilidade

3.4.1.1 Definição de ontologias

No domínio da computação, pode-se entender ontologia inicialmente como a definição de categorias para as coisas que existem em um mesmo domínio. Ontologia é um “catálogo de tipos de coisas”, as quais assume-se a existência em um domínio de interesse na perspectiva de uma pessoa que usa uma determinada linguagem (SOWA, 1999).

Entretanto, mesmo considerando apenas o domínio da computação, são diversos os conceitos apresentados para o termo e existem muitas contradições. Uma das definições mais conhecidas para ontologias no campo da Inteligência Artificial é a de Gruber (1995) citado por Corazzon (2002, p. 1):

Uma ontologia é uma especificação explícita de uma conceitualização. O termo é tomado da filosofia, onde uma ontologia é uma consideração sistemática da existência. Para os sistemas da inteligência artificial, o que existe é o que pode ser representado. [...] Dessa forma, no contexto da inteligência artificial, pode-se descrever a ontologia de um programa definindo um grupo de termos representacionais. Em tal ontologia, definições associam nomes de entidades no universo do discurso (por ex. classes, relações, funções ou outros objetos, com textos legíveis para pessoas, descrevendo o que os nomes significam, e, os axiomas formais que restringem a interpretação e o uso adequado desses termos) [...].

Esta definição é questionada por Guarino e Giaretta (1998), citado por Corazzon (2002, p. 1), que após examinar algumas possíveis interpretações para o termo ontologia, afirmam:

[...] um ponto inicial nesse esforço de tornar claro o termo, será uma análise cuidadosa da interpretação adotada por Gruber. O principal problema com tal interpretação é que ela é baseada na noção conceitualização, a qual não corresponde a nossa intuição. [...] Uma conceitualização é um grupo de relações extensionais descrevendo um 'estado das coisas' particular, enquanto a noção que temos em mente é uma relação intensional, nomeando algo como uma rede conceitual a qual se superpõe a vários possíveis 'estados das coisas'.

Borst (1997, p. 12) apresenta outra definição considerada mais simples e completa: “Uma ontologia é uma especificação formal e explícita de uma conceitualização compartilhada”. Nessa definição, “formal” significa legível para computadores; “especificação explícita” diz respeito a conceitos, propriedades, relações, funções, restrições, axiomas que são explicitamente definidos; “compartilhado” quer dizer conhecimento consensual; e, “conceitualização” diz respeito a um modelo abstrato de algum fenômeno do mundo real. Além disso, a ontologia também deve definir as regras que regulam a combinação entre os termos e as relações. As ontologias e os relacionamentos entre os termos são criados por especialistas naquele domínio. Os usuários formulam consultas usando os conceitos especificados na ontologia e o sistema trata da heterogeneidade e da distribuição pelos repositórios. Em última instância, uma ontologia define uma “linguagem” (conjunto de termos) que será utilizada para formular consultas.

No contexto desse trabalho, não se faz opção por uma definição específica, dentre as diversas apresentadas, nem se pretende discutir aqui qual seria a melhor e mais completa. O termo é utilizado ao longo do trabalho no âmbito de sistemas automatizados.

A ideia é que técnicas de organização como o uso de ontologias possam auxiliar na obtenção da interoperabilidade. Em seguida, a Seção 3.4.1.2 apresenta algumas características das ontologias no âmbito da interoperabilidade de sistemas. No Capítulo 4, a Seção 4.1 apresenta aplicações que objetivam a interoperabilidade e utilizam as ontologias para tal.

3.4.1.2 Ontologias no âmbito da interoperabilidade de sistemas

Em função da necessidade de interoperar, inerente aos ambientes atuais que lidam com informação, muitos esquemas de metadados têm sido propostos nos últimos anos. Os metadados podem ser organizados em ontologias que classificam suas categorias e possibilitam mais informações sobre um determinado domínio.

Além de outras utilizações, as ontologias são uma abordagem possível para se tratar problemas de integração. Segundo Pacheco e Kern (PACHECO E KERN, 2001, p.58), “[...] a construção de ontologias comuns tem sido proposta como abordagem promissora para a interoperabilidade de sistemas”.

Muitas aplicações que envolvem ontologias tratam a questão da interoperabilidade e estão presentes em diversos campos do conhecimento. Um esquema gráfico que identifica formas de aplicação é apresentado na Figura 13.

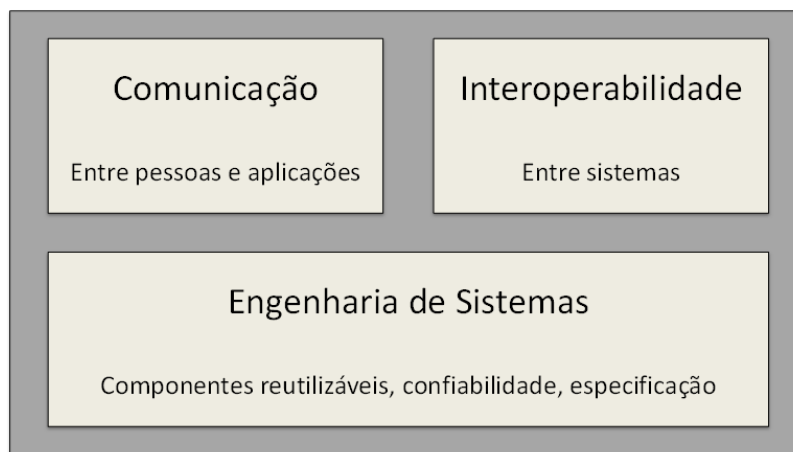


Figura 13. Formas de aplicação de ontologias

Conforme a Figura 13, uma das formas de utilizar ontologias é na interoperabilidade entre sistemas. Esses sistemas podem consultar diversas fontes de dados e sua integração será mais eficiente através da organização, formalização e tradução entre os diferentes significados ou conceitos presentes. Gómez-Pérez (GOMES-PERÉZ, 2001) corrobora com essa ideia ao afirmar que as ontologias podem ser utilizadas na comunicação entre pessoas, organizações e na interoperabilidade entre sistemas (tradução de métodos de modelagem, paradigmas, linguagens e ferramentas de software).

Aplicações que utilizam ontologias são encontradas em diversos campos do conhecimento. As aplicações relativas à integração de fontes de dados que utilizam ontologias, de maior interesse para este trabalho, são tratadas com mais detalhes na seção seguinte.

Ontologias são um tópico relevante para estudos de interoperabilidade, visto que, determinar conceitos e relações em domínios específicos de conhecimento é parte importante de um projeto de integração entre fontes heterogêneas. O desenvolvimento de ontologias, bem como a possibilidade de combiná-las, para alcançar níveis de integração desejados, é atualmente uma importante atividade de pesquisa interdisciplinar, que integra

a Ciência da Computação e a Ciência da Informação. Ambientes de informação distribuídos em redes são cada vez mais dominantes na maioria das atividades relacionadas à informação para negócios, para serviço público e para uso pessoal. A integração semântica nestes contextos pode ser conseguida através de ferramentas que utilizam ontologias para organizar o conhecimento.

3.4.2 Classificação de elementos contidos em modelos de interoperabilidade

Com o objetivo de minimizar a complexidade do problema, alguns autores propõem estruturas de classificação que tentam facilitar a compreensão das variáveis envolvidas. Além disso, a existência de uma classificação pode proporcionar uniformidade às diversas abordagens. Apresentam-se em seguida algumas tentativas de realizar tal classificação obtida na literatura.

Problemas de interoperabilidade rapidamente detectados podem se tornar parte do conjunto de decisões ainda na etapa de projeto, resultando em uma implementação de sistemas com menos problemas. Segundo Keshav, Os componentes principais de um projeto de integração são (Keshav et al., 1999):

- *Translator*: converte dados e funções entre os diferentes formatos dos componentes, sem alterar o conteúdo da informação;
- *Controller*: coordena e intermedia o movimento da informação entre os componentes usando processos pré-definidos de tomada de decisão;
- *Extender*: adiciona novas características e funcionalidades, proporcionando melhorias às capacidades de um componente;
- Elementos híbridos: “*translator-controller*”, o “*translator-controller-extender*” e o “*controller-extender*”.

Para se que possa ter uma ideia da variedade de possibilidades, de teorias e projetos que buscam a integração de fontes de dados, algumas propostas de diferentes comunidades científicas são apresentadas na seção seguinte.

4

Trabalhos Relacionados

A seguir são expostas algumas pesquisas que abordam propostas de arquiteturas de integração, como também as abordagens existentes para tratar de correspondência de metadados.

4.1 Propostas de Arquitetura de Integração

As propostas de arquitetura para integração de metadados encontradas na literatura baseiam-se na construção de modelos genéricos resultantes das fontes de dados a serem integradas, mas empregando diferentes estratégias. Por exemplo, o BUSTER (STUCKENSCHMIDT; WACHE, 2000) utiliza um modelo global para descrever a semântica das fontes de informação, construindo um vocabulário compartilhado. O OBSERVER (MENA et al., 1996) lida com a agregação de múltiplos modelos, no formato de ontologias, ficando responsável pela complexa tarefa de manter a consistência e eficiência dos modelos agregados, sem a necessidade de geração de um modelo genérico. Já o DWQ (CALVANESE, 2001) utiliza uma abordagem que incorpora as estratégias descritas anteriormente, logo os modelos das fontes de dados participantes da integração podem ser agregados a um modelo genérico ou podem-se utilizar modelos distintos com os respectivos mapeamentos.

4.1.1 Uso de um modelo global

4.1.1.1 BUSTER

O BUSTER (STUCKENSCHMIDT; WACHE, 2000) faz uso de uma abordagem híbrida para descrever a semântica das fontes de informação. A semântica de cada fonte é descrita por sua própria ontologia, a qual uma é construída a partir de um vocabulário global compartilhado. O vocabulário compartilhado é uma ontologia (global), que define as faixas de valor de atributos e seus conceitos.

No BUSTER, o vocabulário compartilhado é uma ontologia (global), que define as faixas de valor de atributos e seus conceitos (WACHE et al., 2001). Uma ontologia-fonte é um refinamento (parcial) da ontologia geral que restringe a faixa de valores de alguns atributos. Como as ontologias-fonte usam apenas o vocabulário da ontologia geral, as duas permanecem comparáveis.

O sistema BUSTER parte do princípio que a análise e transformação do contexto do conhecimento é necessária para alcançar interoperabilidade em nível semântico. Aborda o problema de preservar o significado de atributos únicos no banco de dados, visto que terminologias diferentes podem ser usadas (STUCKENSCHMIDT; WACHE, 2000).

Regras de integração orientam a correlação das fontes de dados para uma visão integrada fornecida pelo componente mediador. As regras se baseiam em uma descrição de objetos de cada fonte de dados. Assim, cada objeto, tabela de banco relacional ou um simples número, será “encapsulado” por um objeto padrão. Este objeto é um predicado, conforme definido na lógica, com campos que contem variáveis e descrevem instâncias encontradas no banco de dados.

4.1.1.2 KRAFT

O KRAFT (WACHE et al., 2001) é uma abordagem linguagens usada em sistemas de integração baseados em ontologias. O KFRAFT adota um processo conhecido como fusão do conhecimento⁴, onde é possível combinar o conhecimento de diversas fontes de maneira altamente dinâmica.

Para combinar informações, as instâncias de dados devem usar o conhecimento associado ao contexto. Este tipo de conhecimento pode ser expresso na forma de restrições. Para representar as restrições e instâncias de um grupo de definições, é necessária uma linguagem comum, ou seja, uma ontologia compartilhada;

⁴ Processamento de informações que trata da associação, correlação e combinação de dados de uma ou mais fontes, com o objetivo de obter uma avaliação mais completa e precisa da situação. O processo é caracterizado pelo refinamento contínuo das estimativas e pela avaliação da necessidade de fontes adicionais ou modificação do próprio processo, para se alcançar resultados mais satisfatórios.

4.1.2 Uso de múltiplos modelos

4.1.2.1 OBSERVER

No OBSERVER (Mena et al.,1996) o conteúdo das fontes de dados é descrito por conceitos presentes em ontologias que representam um domínio do conhecimento. Para Mena (Mena et al.,1996), o problema diz respeito ao compartilhamento do vocabulário, ou seja, lidar com diferentes termos ou conceitos para descrever informações similares.

O OBSERVER permite a representação de ontologias por relacionamentos “inter-ontologias” e correlaciona os termos entre elas. As consultas do usuário são reescritas usando esses relacionamentos, o que resulta em traduções entre as ontologias.

Em geral, existem diferenças entre as ontologias, pois são desenvolvidas de forma independente. Ontologias diferentes, com vocabulários diferentes podem atender melhor às necessidades dos usuários.

O OBSERVER tem a capacidade de lidar com múltiplas ontologias, mas fica também responsável pela tarefa (muitas vezes complexa) de manter a consistência e eficiência. As ontologias devem estar interligadas por relacionamentos inter-ontologias e organizadas em grupos que correspondem a áreas do conhecimento. Como alguns grupos são mais gerais que outros, podem ser organizadas hierarquias.

4.1.3 Uso de modelos combinados

4.1.3.1 COIN

O Context Interchange Project (COIN) (BRESSAN, 1997) lida com a integração de fontes de dados distribuídas e heterogêneas, adotando uma arquitetura de mediadores cujo principal componente é o denominado *MINT*, um mediador que proporciona consultas a múltiplas fontes e tem a capacidade de detectar e resolver conflitos semânticos. *Wrappers* fazem a ligação física e lógica entre as fontes de dados e o mediador.

O sistema COIN adota uma abordagem híbrida, ou seja, cada fonte de dados é descrita por sua própria ontologia. Para que as ontologias locais sejam comparáveis, elas são construídas a partir de um vocabulário global compartilhado. Este vocabulário contém termos básicos (primitivos), os quais são combinados em ontologias locais para descrever semânticas mais complexas

Os componentes da arquitetura executam três grupos diferentes de processos:

- Processos-cliente: proporciona interação entre os pedidos ao banco de dados e um mediador;
- Processos-servidor: permitem consultas a semi-estruturas na web e gateways do BD, que conectam fisicamente os bancos a rede;
- Processos-mediadores: componentes que fornecem serviços coletivos de mediação.

4.1.3.2 Data Warehouse Quality (DWQ)

O Data Warehouse Quality (DWQ) (CALVANESE, 2001) é o único dos sistemas citados em que as três abordagens são implementadas.

O DWQ assume que cada fonte (sistema a ser integrado) é uma coleção de tabelas relacionais e cada uma delas é descrita em termos de sua ontologia com o auxílio de consultas. Uma consulta global e sua decomposição em sub-consultas é relacionada a conceitos ontológicos. Se as sub-consultas estão contidas na consulta global, elas estão corretas.

4.1.4 Quadro comparativo

As abordagens citadas possuem o objetivo de garantir a interoperabilidade entre sistemas distintos, mas muitas vezes exige-se a realização de adaptações nas aplicações que desejam utilizá-las. A arquitetura BUSTER é um exemplo, é definido um vocabulário global e o sistema cliente que desejar se integrar precisa adequar o seu modelo àquele vocabulário. Já o OBSERVER não obriga os sistemas clientes a adotarem um modelo pré-definido, porém, por não utilizar um mapeamento num modelo genérico, a realização de consultas torna-se um processo bastante custoso, visto que a cada solicitação deverá ser feito o mapeamento entre os modelos. Por último, o DWQ possui a vantagem dar flexibilidade à utilização (seguindo um modelo genérico ou mantendo os modelos originais e realizando o mapeamento), porém, ainda encaixa-se nos mesmos problemas das propostas citadas.

Além disso, as arquiteturas propostas não focam num domínio da aplicação específico. O que pode acabar dificultando a integração quando sistemas de domínios de conhecimento totalmente distintos buscam a integração.

Tomando como base os desafios citados, a proposta deste trabalho mantém o seu foco especificamente nos sistemas de gerenciamento de ativos digitais e usa uma abordagem que não obriga a adoção por parte dos sistemas clientes de um modelo já existente, mas também mantém um meta-modelo que é construído e atualizado dinamicamente, de forma a manter a compatibilidade contínua após o mapeamento dos modelos.

A Tabela 2 contém um resumo dos projetos para integração baseados em ontologias apresentados:

Tabela 2. Quadro resumo dos projetos de interoperabilidade citados

Projeto	Princípio de funcionamento	Papel da ontologia	Representação do meta-modelo	Comentários
BUSTER	Regras de interação orientam a correlação entre as fontes em função de uma visão integrada produzida pelo mediador.	Ontologia global representa um vocabulário compartilhado que define faixas de atributos e conceitos.	Lógica descritiva (OWL)	Ontologia representa uma cópia da estrutura (por ex. esquema de banco de dados) e a define melhor adicionando conceitos.
OBSERVER	Relações inter-ontologias usam princípios lingüísticos	Ontologias diferentes representam, de forma independente, as fontes de dados.	Lógica descritiva (CLASSIC)	O vocabulário é definido por um repositório compartilhado (IRM), que armazena as relações inter-ontologias.
KFRAFT	Arquitetura de mediadores e wrappers (representados por agentes) executam "fusão do conhecimento"	Ontologia compartilhada caracteriza totalmente a área do conhecimento referente ao problema.	Linguagem frame-based	Fusão de conhecimento é um Processo baseado em restrições.
COIN	Arquitetura de wrappers e mediadores	A semântica de cada fonte é descrita por sua própria ontologia, a qual é construída a partir de um vocabulário comum compartilhado.	Linguagem frame-based (F-Logic)	Solução de conflitos semânticos é feita por um componente mediador especializado.
DWQ	Um modelo conceitual (em um nível superior) descreve as relações entre os modelos fonte e um modelo genérico.	Análise de conteúdo e descrição do processo de integração.	Lógica descritiva (extensão DLR)	Aceita abordagem por um modelo global, por vários modelos ou por um modelo híbrida.

Como foi visto, a modelagem de semântica de metadados através do uso de meta-modelos não é uma ideia nova (WACHE et al., 2001). Entretanto, em contraste às outras, este trabalho não propõe uma abordagem de meta-modelo único, onde um modelo global provê um vocabulário compartilhado para as fontes. A ideia do trabalho atual é deixar o sistema cliente livre para utilizar o seu modelo de forma independente para a formulação das consultas. Uma vez que os modelos adotados em várias instituições podem ser diferentes uns dos outros, contamos com mapeamentos entre os elementos dos esquemas, gerando como resultado um meta-modelo comum gerado em cada integração entre duas fontes distintas. O meta-modelo contém uma mescla dos elementos de ambas as fontes, sendo o mapeamento dos elementos relacionados realizado de forma semiautomática, ou seja, com o apoio do usuário.

4.2 Sistemas de correspondência de metadados

Considerando uma arquitetura que aborde o problema de heterogeneidade de metadados, são necessárias ferramentas ou mecanismos para que analisem as correspondências dos modelos para que se possa atingir a interoperabilidade. Estas ferramentas e mecanismos devem resolver a heterogeneidade semântica e estrutural dos esquemas e alinhar os termos presentes nos modelos. Várias técnicas de mapeamento de esquemas foram propostas ao longo da última década. Entre elas podemos citar: Similarity Flooding (MELNIK, 2002), Cupid (MADHAVAN et al., 2001), S-Match (GIUNCHIGLIA et al., 2004) e MuMle (AMIR, S. et al., 2011).

A desvantagem do método Similarity Flooding descrito em (MELNIK, 2002) é que ele é baseado na ideia da propagação de similaridades. O conceito básico por trás do algoritmo é que as adjacências entre elementos contribuem para essa propagação. Assim, o algoritmo poderá executar de forma inesperada nos casos em que as informações de adjacência não forem preservadas nos modelos. Já o S-Match, descrito em (GIUNCHIGLIA et al., 2004) é limitado às estruturas baseadas numa representação em árvores e não consideram propriedades adicionais dos elementos ou regras.

Algumas dessas abordagens fazem a análise da informação estrutural e semântica, mas muitas vezes deixam de considerar fatores importantes, como por exemplo, a comparação da equivalência de propriedades e as diferenças estruturais entre as classes de elementos. Assim, torna-se difícil detectar a correspondência entre modelos que contenha muitos tipos complexos (que contenham um conjunto de outros tipos). Além

disso, a principal desvantagem destes métodos é a forma de utilizar a informação estrutural. Por exemplo, os autores do Cupid (MADHAVAN et al., 2001) consideram que a maior parte do conteúdo relevante está representada nas folhas (elemento de mais baixo nível de uma estrutura composta de metadados) e que as folhas têm variações menores entre os esquemas do que as estruturas internas. Assim, a similaridade entre nós baseia-se na semelhança de seu conjunto de folhas. Isso nem sempre é verdade, visto que podemos encontrar conceitos equivalentes ocorrendo em estruturas completamente diferentes. Como exemplo temos o atributo “title”, que no PBCore é um campo de texto simples denominado “pbcoreTitle”, mas no TV-Anytime é representado como o atributo “text” do elemento complexo “Title” que faz parte do elemento complexo “BasicDescription”. No caso, a comparação entre os elementos folhas “title” e “text” não traria um bom grau de correlação.

O MuMle (Multi-level Metadata Mapping System) (AMIR, S. et al., 2011), por sua vez, tomou como base as abordagens citadas para construir uma ferramenta simplificada visando alcançar a interoperabilidade atingindo dois níveis (conceitual e lógico). O MuMle pode ser utilizado para integrar vários esquemas heterogêneos, representados através de linguagens diferentes, combinando vários elementos estruturais e semânticos disponíveis nos esquemas para detectar as correlações entre os termos de metadados. Além disso, a abordagem preenche a lacuna da análise estrutural existentes nos sistemas de correspondência mencionados anteriormente. A seguir é detalhado o funcionamento do sistema MuMle.

4.2.1 MuMle (Multi-level Metadata Mapping System)

O MuMle é um sistema de correspondência de metadados que pode ser utilizado para auxiliar desenvolvedores no processo de integração, através da detecção de correspondências entre elementos de esquemas de metadados. O sistema trabalha tanto no nível do modelo de dados (esquema), como no nível da linguagem de definição de esquemas. Para aumentar o grau de acerto das correlações, o MuMle utiliza uma combinação de vários tipos de informações semânticas e estruturais.

Esta seção descreve as duas partes principais do MuMle: projeção e correspondência. Na etapa de projeção procura-se alcançar a interoperabilidade no nível da linguagem de definição de esquemas. Na etapa de correspondência o objetivo é

encontrar as correlações entre os elementos de metadados, usando vários tipos de informações (e.g. semântica, estrutural).

4.2.1.1 *Projeção*

Na fase de projeção são propostas regras de transformação que permitem a projeção de diferentes linguagens no mesmo espaço de representação. Este espaço é um modelo abstrato que serve como base para representar conceitualmente vários tipos de esquemas independentemente da linguagem utilizada para descrevê-los. Os esquemas são modelados como grafos rotulados e as informações semânticas e estruturais são salvas para uso na etapa de correspondência de elementos do modelo.

A Figura 14 ilustra o processo de projeção. O exemplo cita três linguagens de descrição (XML Schema, RDF Schema e OWL). Elas foram escolhidas por possuírem um grande volume de utilização e por apresentarem uma heterogeneidade significativa (uma descrição estrutural em alto nível para a XML Schema e uma alta expressividade semântica para RDF Schema e OWL).

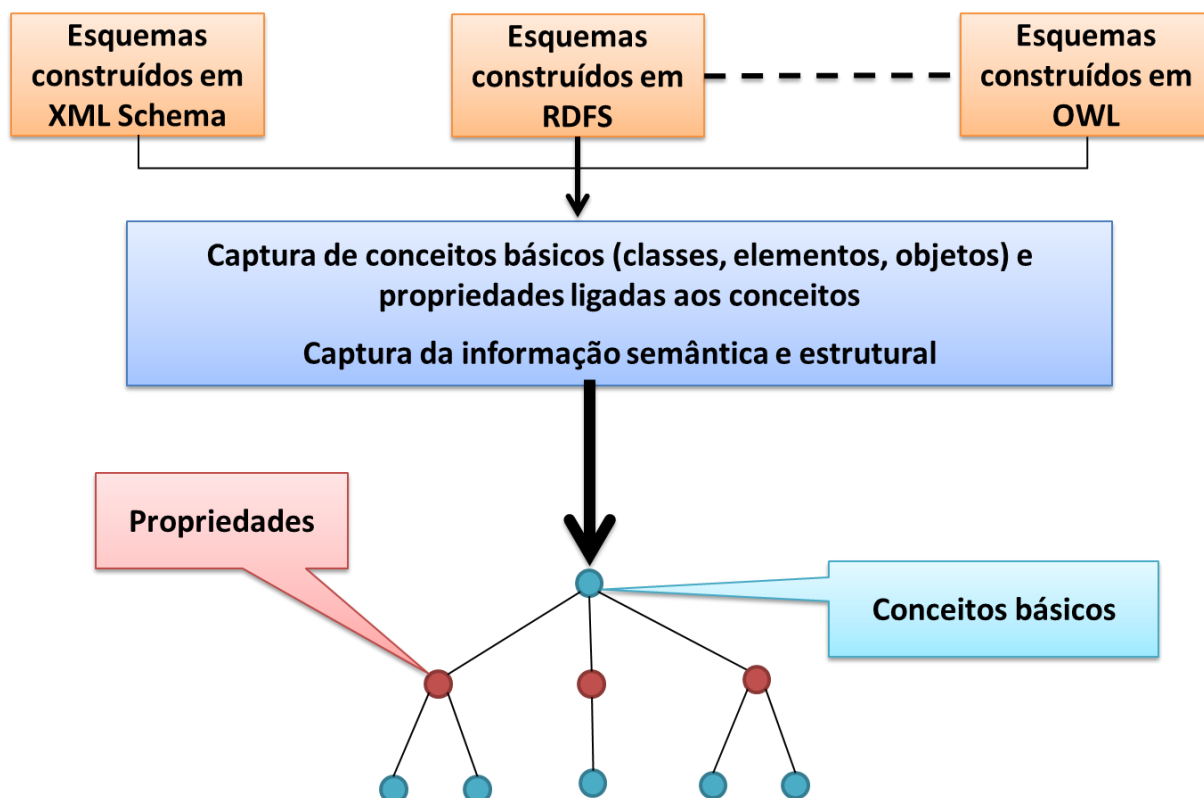


Figura 14. Ilustração do processo de projeção (AMIR, S. et al., 2011)

1) Modelagem de esquemas em grafos: Devido a alta heterogeneidade das linguagens de definição de esquemas, não é possível encontrar um modelo de

representação que suporte as características de todos os esquemas. O MuMle modela estes esquemas como grafos rotulados representando do esquema somente conceitos básicos e propriedades ligadas aos conceitos. Estes dois elementos (conceitos básicos e propriedades), são informações comuns e básicas em várias linguagens. Para construir o grafo, a descrição dos conceitos presentes nas linguagens devem ser obtidos. Por exemplo, todas as classes (rdf:class) e propriedades (rdf:property) no RDF Schema precisam ser representadas por nós. A Tabela 3 mostra os conceitos básicos e as propriedades para três linguagens diferentes. Considerando as linguagens baseadas em classificação de taxonomia (e.g. XML Schema), foi usada a ideia proposta por Miletic et al. (Miletic et al., 2007) para capturar a semântica de propriedades implícitas. Logo, o nome do nó de propriedade é a combinação dos nós pai e filho.

A classificação dos nós ocorre na definição de dois tipos: nós normais e nós de propriedade. Os nós normais correspondem aos conceitos básicos (e.g., xsd:complexType, rdfs:class, xsd:element, etc.) e nós de propriedade correspondem a propriedades ligadas aos conceitos (e.g., rdf:property, owl:ObjectProperty, etc.). A Figura 15 ilustra um exemplo de um grafo que representa o meta-modelo RDF descrito na Figura 16.

Tabela 3. Regras de mapeamento de linguagens

Tipo do nó	XSD	RDFS	OWL
Nó normal	xsd:element xsd:attribute xsd:attributeGroup xsd:group	rdfs:class	owl:class
Nó de propriedade	parentName childName	e rdf:property	owl:objectProperty

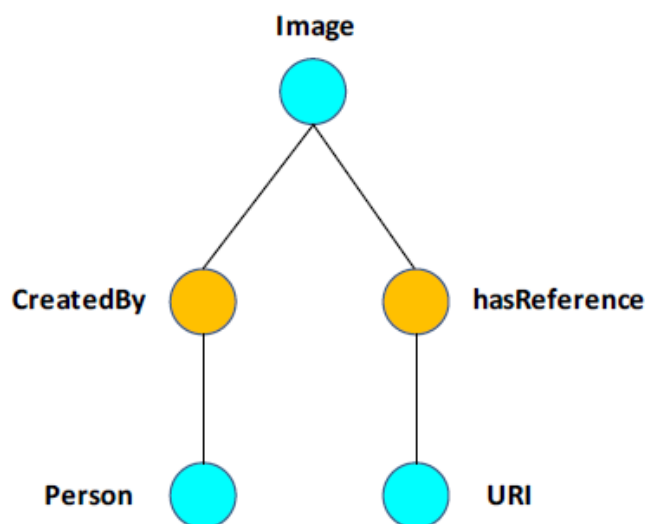


Figura 15. Exemplo da representação do grafo

```

<rdf:RDF>
-----
<rdfs:Class rdf:about="Image">
<rdfs:subClassOf rdf:resource="&abs;Object"/>
</rdfs:Class>
<rdf:Property rdf:about="hasReference">
<rdfs:domain rdf:resource="Image"/>
<rdfs:range rdf:resource="URI"/>
</rdf:Property>
<rdf:Property rdf:about="createdBy">
<rdfs:range rdf:resource="Image"/>
<rdfs:domain rdf:resource="Person"/>
</rdf:Property>
-----
</rdf:RDF>

```

Figura 16. Fragmento exemplo em RDF

2) Informações estruturais e semânticas: Tipos diferentes de informação estrutural e semântica podem ser especificados com diferentes linguagens de definição de esquema. Contudo, nem todas as informações serão necessárias para a etapa de correspondência. Logo, utiliza-se apenas a parte da informação que pode ajudar a detectar correlações entre elementos dos modelos de metadados. A comparação de tipos de dados, por exemplo, é ignorada porque pode-se não encontrar um nó n_i que se correlacione com outro nó n_j , mas um deles corresponde a um nó folha (tipo simples) e o outro é um nó complexo (tipo complexo). Além disso, a informação dos tipos de dados não está disponível em todas as linguagens de definição de esquema (ex.: todos os tipos simples são `rdfs:literal` para RDF Schema). As cardinalidades de elementos também são outro exemplo de informação ignorada. A Tabela 3 mostra as informações estruturais e semânticas utilizadas no processo de correspondência.

Tabela 4. Regras de mapeamento de linguagens

XSD	<code>xsd:restriction</code> , <code>xsd:abstraction</code> , <code>xsd:extension</code> , <code>xsd:substitutionGroup</code>
RDFS	<code>rdfs:seeAlso</code> , <code>rdfs:isDefinedBy</code> , <code>rdfs:subClassOf</code>
OWL	<code>owl:unionOf</code> , <code>owl:complementOf</code> , <code>owl:intersectionOf</code> , <code>owl:TransitiveProperty</code> , <code>owl:someValuesFrom</code> , <code>owl:equivalentClass</code> , <code>owl:disjointWith</code> , <code>rdfs:subClassOf</code> <code>owl:distinctMembers</code> , <code>owl:differentFrom</code> <code>owl:AllDifferent</code> , <code>owl:sameAs</code> , <code>owl:equivalentProperty</code>

4.2.1.2 Correspondência

Nesta seção é descrita em detalhes a etapa de correspondência. Conforme pode ser visto na Figura 17, esta etapa é composta de três partes principais: pré-processamento, computação das similaridades linguísticas e computação das similaridades estruturais. São recebidos como entrada dois esquemas representados na forma de grafos rotulados, como também as informações semânticas e estruturais obtidas a partir da etapa de projeção. Em seguida, todos os nós irrelevantes em ambos os esquemas serão eliminados e os úteis normalizados. Após a etapa de pré-processamento, é realizado o cálculo da similaridade linguística entre os nós de ambos os esquemas através da exploração da semântica de sua correspondência de nomes e comentários, bem como da informação semântica obtida da etapa de projeção. Por fim, é computada a similaridade estrutural e os mapeamentos corretos são selecionados de acordo com os coeficientes de similaridade linguística e estrutural.

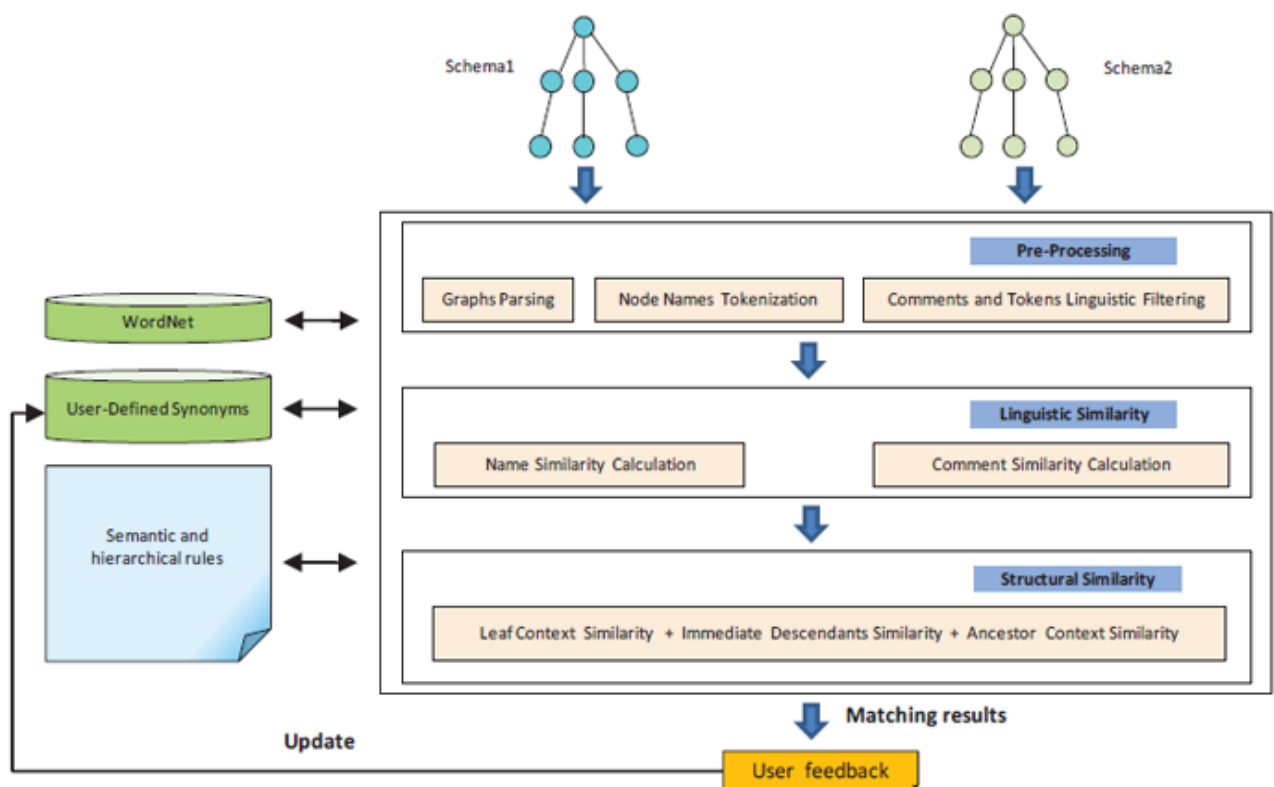


Figura 17. Etapas do processo de correspondência de metadados realizado pelo MuMle (S. Amir et al., 2010)

4.2.1.2.1 Pré-processamento

Esta etapa é iniciada pela análise de todas as entidades envolvidas no processo de correspondência, incluindo nós normais e nós de propriedade, como também comentários (i.e.: descrição textual disponível em elementos `xsd:documentation`, `rdfs:comment`, etc) correspondentes a essas entidades. Em seguida os nomes dos nós e os comentários são normalizados a fim de se tornarem úteis para a etapa de computação das similaridades linguísticas.

a) Nome dos nós: Normalmente cada entidade no grafo é modelada por um nó que contém um nome. O nome do nó é representado por uma cadeia de caracteres (String), sem conter espaços, que pode ser uma palavra, um termo ou uma expressão (uma combinação de palavras). A fim de calcular as similaridades entre os nomes de nós, uma etapa de normalização é necessária. Logo, cada nome de entidade é quebrado em um conjunto de tokens M através de um tokenizador (conjunto de critérios para transformar a palavra em um conjunto de tokens) configurável, considerando pontuação, letras maiúsculas, símbolos especiais e dígitos, e.g. `MediaRegionLocator` torna-se (`Media`, `Region`, `Locator`). Assim que a etapa de tokenização é concluída, os tokens são lematizados, isto é, eles são analisados morfologicamente a fim de que sejam encontradas todas as suas possíveis formas básicas. Deste modo, por exemplo, "Locations" está associado com a sua forma singular, "Location". Um dicionário definido pelo usuário também é usado para encontrar acrônimos e abreviações, e.g., "ID" torna-se "Identifier".

b) Comentários: como mencionado anteriormente, os comentários presentes nos modelos são usados como outra informação semântica. Para possibilitar uma maior relevância semântica, os comentários precisam ser filtrados linguisticamente através da eliminação de palavras que contenham pouca informação útil, como artigos, proposições e conjunções.

4.2.1.2.2 Computação da similaridade linguística

Esta fase concentra-se na computação da similaridade linguística entre cada um dos elementos do par de grafos construídos a partir dos esquemas de metadados mapeados. O resultado dessa fase é a construção de uma matriz contendo as similaridades linguísticas. Para identificar o significado e os sinônimos de cada elemento é realizada uma consulta ao serviço de banco de dados léxico WordNet (WORDNET, 2012). Além dos nomes dos nós, os comentários também são usados como uma segunda fonte semântica para o processo de correspondência de metadados. Nos comentários é considerada apenas a informação

mais pertinente, excluindo-se termos que não tenham valor semântico. O grau de similaridade linguístico é extraído a partir da soma de ambas as similaridades (nome e comentários). Desta forma, as informações estruturais e semânticas obtidas na etapa de projeção são usadas para deduzir outros mapeamentos.

a) Correspondência de nomes

O propósito desta fase é encontrar uma correspondência inicial através do cálculo da distância de similaridade dos nomes de todos os nós pertencentes aos pares de esquemas a serem mapeados. Cada nó é representado por um conjunto de tokens, por ser uma estrutura mais rica semanticamente do que a linguagem natural. Começa-se então pela explicitação dos significados de cada token usando o Wordnet (WORDNET, 2012), que consegue encontrar vários sinônimos para um termo específico. Isso ajuda a resolver problemas de conflitos terminológicos que ocorrem quando padrões de metadados são desenvolvidos por comunidades diferentes, o que poderia ocasionar a descrição de uma mesma informação, porém fazendo uso de termos diferentes.

O conjunto contendo os tokens e o agrupamento dos sinônimos pode ser representado pela seguinte fórmula:

$$M'_i = M_i \cup \{m_k | \exists m_j \in M_i \cap m_k \in \text{synset}(m_j)\} \quad (1)$$

Cada nó representado por um conjunto de tokens M_i possuirá, após a consulta ao Wordnet, um conjunto de sinônimos (*synset*) para cada token m_i . M'_i é o resultado final que traz o reagrupamento de todo o conjunto de sinônimos retornados após a construção de M_i .

Uma vez que a etapa de explicitação dos tokens foi realizada, são computadas as similaridades entre todos pares de nós pertencentes aos dois esquemas mapeados. Para fazer isso, é calculado o S_{name} para cada par de nós (n_1, n_2) usando a métrica de Jaro-Winkler (JW) (BILENKO, 2003) entre cada token $M_i \in M'_i$ e todos os tokens $m_j \in M_2$ (e vice-versa). A distância de Jaro-Winkler é dada por:

$$JW(m_i, m_j) = \left(\frac{r}{|m_i|} + \frac{r}{|m_j|} - \frac{r-t}{r} \right) \quad (2)$$

Onde r é o número de caracteres correspondentes, t é o número de transposições e $(||m_i|, |m_j|)$ são a quantidade de termos (*strings*) correspondentes aos tokens (m_i, m_j) .

Em seguida, é obtida a pontuação (MJV) de cada token m_i :

$$MJW(m_i, M') = \max_{m_j \in M'} JW(m_i, m_j) \quad (3)$$

Finalmente, a média das melhores similaridades é calculada para que se possa obter a similaridade de nomes entre os nós:

$$S_{name}(n_1, n_2) = \frac{\sum_{m_i \in M_1} MJW(m_i, M'_2) + \sum_{m_j \in M_2} MJW(m_j, M'_1)}{|M_1| + |M_2|} \quad (4)$$

b) Correspondência de comentários

Nesta etapa é aplicada a técnica TF/IDF (AMIR, S. et al., 2010) a fim de calcular as similaridades entre comentários. Para isso, todos os comentários nos dois esquemas a serem mapeados são considerados como documentos. Cada nó é representado por um vetor $v = (w_1, w_2, \dots, w_p)$ cujas coordenadas são os resultados do TF/IDF. Mais detalhes sobre essa técnica podem ser obtidos em (AMIR, S. et al., 2010). Assim, a similaridade $S_{comment}$ entre dois nós (n_i, n_j) é a distância entre os vetores correspondentes aos comentários.

$$S_{comment}(n_1, n_2) = \frac{\sum_{k=1}^p w_{ik} w_{jk}}{\sqrt{\sum_{k=1}^p (w_{ik})^2 * \sum_{k=1}^p (w_{jk})^2}} \quad (5)$$

O resultado do processo acima é uma matriz de similaridade linguística $lSim$, onde:

$$lSim(n_1, n_2) = \mu_1 * S_{name}(n_i, n_j) + \mu_2 * S_{comment}(n_i, n_j) \quad (6)$$

c) Informações estruturais e semânticas:

Informações estruturais e semânticas obtidas do processo de projeção são usadas no MuMle para detectar outros mapeamentos candidatos, especialmente os mapeamentos de tipos complexos. A seguir são apresentados alguns exemplos de tais aspectos e como eles são usadas para deduzir outros mapeamentos candidatos.

- **Aspectos de generalização:** Relacionamentos de generalização entre dois tipos indica que um tipo é um subconjunto do outro (e.g., *xsd:extension*, *rdfs:subClassOf*, *xsd:abstraction*, etc.). Esta informação pode ajudar efetivamente o algoritmo de correspondência a inferir correlações complexas. Por exemplo, considerando que o atributo *ms:Agent* é definido no esquema a ser mapeado usando o padrão MPEG-7, este atributo é linguisticamente mapeado para o elemento complexo *mpeg7:AgentType* (o resultado do *lSim* aponta o maior limiar de correlação para este elemento). Neste contexto, a união de dois elementos de tipo complexo *mpeg7:PersonType* e *mpeg7:OrganizationType* também são considerados candidatos a mapeamento para *ms:Agent*. Isso acontece porque *mpeg7:OrganizationType* e *mpeg7:PersonType* são extensão do *mpeg7:AgentType*. Outras propriedades estruturais também podem ser consideradas (e.g., *owl:disjointWith*, *owl:unionOf*, etc.)
- **Aspectos semânticos:** Considerando que n_i e n_j são dois nós linguisticamente correlatos, e n_k é outro nó que corresponde a uma classe que tem uma propriedade de equivalência para n_i (e.g., *owl:equivalentProperty*, *owl:equivalentClass*, *xsd:substitutionGroup*, etc.). Esta informação possibilita a dedução que n_k é outro candidato a mapeamento para n_j . Outras propriedades semânticas também podem ser usadas (e.g., *owl:someAs*, *owl:differentFrom*, etc.)

O uso as informações estruturais e semânticas ajudam a descobrir mais mapeamentos mas também pode aumentar o número de falsos positivos, que podem ser eliminados na etapa de computação da similaridade estrutural, descrita na seção seguinte.

4.2.1.2.3 Computação da similaridade estrutural

A computação da similaridade linguística pode prover vários nós candidatos. Também pode haver várias correspondências candidatas que diferem em estrutura, mas têm um alto grau de similaridade. Por exemplo, o padrão TV-Antyime possui os tipos complexos “Title” e “Genre” (que descrevem o título e o gênero de uma mídia), ambos possuem um atributo descritivo denominado “Text”, que poderiam acabar sendo apontados como candidatos a mapeamento pela similaridade linguística. Entretanto, o significado dos atributos “Text” do elemento “Title” e “Text” do elemento “Genre” não são compatíveis,

resultando então num falso positivo. Para contornar esse tipo de situação, o MuMle realiza a computação da similaridade estrutural. O algoritmo de correspondência estrutural é baseado no contexto atual de um determinado nó. São considerados três tipos de níveis, dependendo da posição do nó grafo: antecessor, descendente imediato e folha. Logo, O contexto de um nó é a combinação dos três níveis. Desta forma, além de “Text”, também é considerado o seu nó antecessor, no caso “Genre” ou “Title”, no momento da comparação. Maiores detalhes sobre como o MuMle realiza a computação das similaridades estruturais nos três níveis podem ser obtidos em (AMIR, S. et al., 2011).

4.2.1.2.4 Representação da medição de similaridades

Após a realização das etapas descritas anteriormente, é obtido um coeficiente de similaridade, que é calculado através da soma do grau de similaridade obtido nas etapas de computação da similaridade linguística e estrutural.

Os elementos de maior similaridade serão agrupados e o retorno é enviado de volta para o sistema cliente, que irá realizar a interação com o usuário, que por sua vez irá confirmar os mapeamentos candidatos ou realizar algum ajuste e a partir dali ficar responsável por gerenciar o modelo obtido como resultado, de forma que ele sirva como uma linguagem comum para os sistemas que utilizam os esquemas de metadados integrados no processo.

O papel do MuMle no Dynamo Framework se restringe à computação das similaridades em mais baixo nível (no nível de nós de grafos construídos a partir de dois modelos de metadados). O Dynamo Framework faz o papel de sistema cliente para o MuMle e ao mesmo tempo disponibiliza uma API que oferece uma abstração para os sistemas de correspondência, admitindo até a possibilidade da utilização de outros sistemas (além do MuMle) de forma simultânea, ou a criação de uma base de dados local que armazene os dados provenientes de integrações anteriores. Além de dar suporte às fases anterior à computação das similaridades, o Dynamo Framework também age na fase que vem após esse processo, fornecendo uma interface de acesso para que seja possível a interação com usuários visando a confirmação das similaridades, e posteriormente o armazenamento do modelo genérico proveniente do processo de integração, disponibilizando operações que podem ser feitas em cima desse modelo. A participação do MuMle nas etapas do processo de interoperabilidade de metadados considerado pelo Dynamo Framework é descrita em detalhes na Seção 5.3.

4.3 Comparações com o trabalho proposto

A seguir é realizada a comparação entre os trabalhos relacionados e o Dynamo Framework, o framework de serviços proposto neste trabalho. São avaliados aspectos como o escopo em relação ao processo de interoperabilidade, o nível de abstração e facilidade de adequação em projetos.

Como o Dynamo Framework possui uma abrangência que contempla várias partes do processo de interoperabilidade entre bases distintas é necessário delimitar um escopo de comparação. Os trabalhos relacionados citados concentram o foco em uma determinada fase do processo. Logo, para uma análise mais eficiente, será considerada a parte do Dynamo Framework que mais se adequa ao objeto da comparação.

Primeiramente, ao fazer a comparação com as propostas de arquitetura de integração existentes na literatura, considera-se uma parte do Dynamo Framework que é o modelo de integração adotado, que consiste na estratégia utilizada para armazenar os meta-modelos resultantes de cada mapeamento. A Tabela 5 descreve a comparação entre os modelos adotados por cada trabalho.

Tabela 5. Comparação entre as arquiteturas de integração presentes na literatura e o Dynamo Framework

Trabalhos relacionados	Estratégia	Dynamo Framework
BUSTER, KFRAFT	Uso de um modelo global que servirá como base para todos os participantes	Utiliza um meta-modelo global para representar cada integração entre pares de modelos.
OBSERVER	Uso de múltiplos modelos independentes. O mapeamento é feito a cada integração, não há modelo genérico.	Cada modelo integrado é independente, dispensando a adequação a um modelo genérico. Entre cada par é gerado um meta-modelo global.
COIN, DWQ	Uso de modelos combinados.	Utiliza a combinação das estratégias de modelos múltiplos independentes e uso de modelo global.

Já em relação aos sistemas de correspondência de metadados, a diferença principal diz respeito à delimitação do escopo. Enquanto o Dynamo Framework oferece um aparato de serviços que apoiam as várias fases do processo de interoperabilidade (ver Seção 5.3), os sistemas de correspondência de metadados concentram-se basicamente no cálculo das correlações entre elementos de modelos distintos. Inclusive, o Dynamo Framework utiliza esse tipo de sistema como parte dos seus serviços para auxiliar na execução da etapa de correspondência dos elementos dos modelos.

5

Um Framework de serviços para Integração de bases de dados heterogêneas

Segundo Haslofer e Klas (HASLHOFER e KLAS, 2010), a construção de uma solução para a interoperabilidade de metadados torna-se uma atividade cada vez mais complexa, dependendo do grau de heterogeneidade das bases que se deseja integrar. Dadas essas dificuldades, muitas vezes a construção de uma solução automática acaba sendo preterida, de forma que o processo de integração seja realizado utilizando um mapeamento manual.

No entanto, o mapeamento manual tende a ser um processo lento e ineficiente, principalmente quando aplicado a modelos de metadados de larga escala (ex.: MPEG-7, TV-Anytime, PBCore, etc.), visto que seria necessário um grande esforço humano para analisar as similaridades entre cada um atributos dos elementos presentes nos modelos, o que faz com que o processo além de demorado, muitas vezes nem traga um resultado satisfatório, dependendo do critério utilizado para identificar as correlações.

Outro problema que deve ser levado em consideração é que, como já foi citado nas seções anteriores, as heterogeneidades dos modelos de metadados estão presentes em dois níveis: esquema e linguagem de definição de esquema. As heterogeneidades estruturais e semânticas no nível de esquema ocorrem quando a mesma informação é representada de forma distinta em esquemas diferentes. As heterogeneidades de linguagens de definição de esquema ocorrem devido à estrutura substancial e discrepâncias semânticas das linguagens. Por exemplo, muitas descrições estruturais construídas em XML Schema não podem ser expressas usando RDF Schema (RDFS) e muitas descrições do RDFS não são suportadas pelo XML Schema.

Como foi visto no Capítulo 4, existe uma grande diversidade de arquiteturas que agregam modelos conceituais cujo objetivo é a busca pela interoperabilidade em bases de dados heterogêneas. Entretanto, esses trabalhos limitam-se a descrever um modelo conceitual, de forma que é pouco detalhada tecnicamente a estrutura dos seus componentes, o que dificulta uma possível implementação. Ainda no Capítulo 4 foram citadas também algumas propostas de sistemas de correspondências de metadados. Entretanto, apenas o uso de tais sistemas não é suficiente para tornar duas bases de dados heterogêneas compatíveis, visto que o processo de integração vai bem mais além do que a obtenção do grau de similaridades entre propriedades dos modelos de dados.

Visando fornecer um maior suporte ao processo de interoperabilidade de bases de dados heterogêneas como um todo, a proposta deste trabalho é a construção de um *framework*, denominado Dynamo Framework, cujo objetivo é disponibilizar recursos essenciais para várias etapas do processo, como o processamento dos esquemas das bases que se deseja integrar, mesmo que representados em linguagens diferentes (e.g. XML Schema, RDFSchema, SQL-DDL); a realização de um mapeamento automático inicial baseado nas correlações semânticas e estruturais dos modelos em ambos os níveis (esquema e linguagem de definição de esquema) e a disponibilização de recursos para o usuário realizar algum ajuste no mapeamento antes de efetuar a confirmação. Agrega-se ainda ao *framework* a implementação de um modelo de interoperabilidade baseado em um conjunto de modelos genéricos (meta-modelos), construídos e mantidos dinamicamente a partir do mapeamento das bases integradas.

O capítulo se estrutura da seguinte forma: na Seção 5.1 é apresentada uma visão geral do Dynamo Framework, incluindo a descrição geral do funcionamento, os requisitos de software, ferramentas e tecnologias utilizadas e a sua arquitetura de componentes. Na Seção 5.2 é descrita a arquitetura do modelo de interoperabilidade baseado na abordagem de meta-modelos dinâmicos, que é utilizado pelo *framework* para armazenar o resultado da integração entre os modelos. A Seção 5.3 apresenta as etapas de utilização do *framework* para atingir a interoperabilidade entre bases de dados heterogêneas.

5.1 Visão geral do Dynamo Framework

Segundo Sayão (SAYÃO, 2001), as estratégias consideradas mais comuns em projetos que visam a integração de sistemas são: mapeamento (adaptação de cada elemento de um modelo em comparação com outro), redução (apenas uma parte dos

modelos é considerada) e pragmatismo (utilizar elementos distintos para cumprir as mesmas funções, orientados pelos mesmos objetivos).

O Dynamo Framework consiste de um conjunto de componentes que objetiva eliminar ou reduzir conflitos semânticos, através da detecção de similaridades de elementos e da posterior construção de um meta-modelo comum, de mais alto nível, integrando modelos de dados heterogêneos. A estratégia para integração utilizada pelo framework proposto trata-se de uma combinação das estratégias citadas por Sayão (SAYÃO, 2001). É adotado o mapeamento, visto que são comparados e adaptados elementos de cada modelo; e o pragmatismo, por ocorrer a correlação de elementos com funções similares, mesmo que contenham estruturas distintas.

A ideia do Dynamo Framework é possibilitar um conjunto de recursos que podem ser agregados facilmente por qualquer sistema que possua uma base de dados e queira realizar a integração com outras, mesmo que essas bases possuam uma estrutura totalmente diferente da sua. Esses recursos são acessados via uma API (*Application Programming Interface* ou Interface de Programação de Aplicativos), que contém métodos cujas responsabilidades são: 1) Processar os meta-modelos (esquemas de metadados) das bases de dados que se pretende integrar, carregando-os numa representação única e legível para o *framework*, independentemente do formato de entrada (e.g. XML Schema, RDFSchema, SQL-DDL); 2) Realizar o mapeamento dos elementos correlatos dos meta-modelos de forma semiautomática (o usuário ou sistema cliente participa confirmando o mapeamento sugerido) e 3) Armazenar o meta-modelo resultante da integração de forma a possibilitar o acesso posterior através da execução de operações entre as bases de dados integradas.

Nas motivações deste trabalho foram citados três principais problemas para a obtenção da integração entre sistemas heterogêneos destacados por Haslofer e Klas (HASLHOFER e KLAS, 2010), são eles: P1) compatibilidade com formatos de armazenamento de dados e interfaces de acesso; P2) Integração de modelos de metadados heterogêneos e P3) Localização transparente e descoberta. Visando solucioná-los, o Dynamo Framework foi construído com base em requisitos de software que buscam contemplar a resolução dos problemas citados.

5.1.1 Requisitos de software

O diagrama de casos de uso, expresso em UML (*Unified Modeling Language*) (OMG, 2006c), expressa os requisitos funcionais do framework na forma de casos de uso. Segundo o RUP (*Rational Unified Process*) (AHMAD e JOCHEN, 2008), para cada requisito funcional tem-se um caso de uso. Na Figura 18 é exibida a representação gráfica em UML dos casos de uso essenciais do Dynamo Framework. Logo após é realizada uma descrição detalhada de cada um dos casos de uso, e em seguida é feita uma comparação de cada um dos requisitos destacados com os três problemas citados anteriormente.

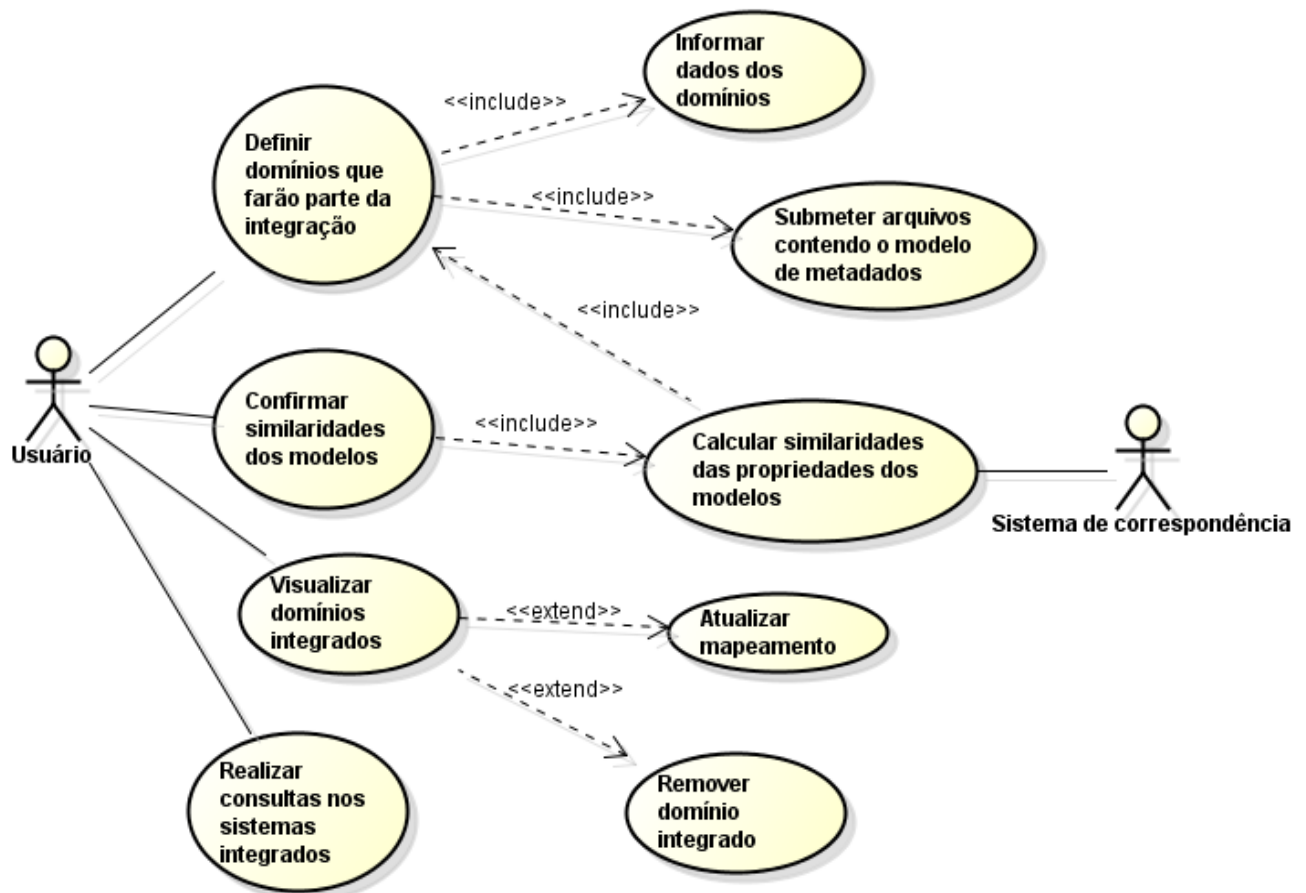


Figura 18. Representação gráfica dos casos de uso do Dynamo Framework

A Tabela 6 apresenta uma descrição detalhada dos casos de uso essenciais do sistema mostrados no diagrama da Figura 18, juntamente da apresentação de suas condições de entrada (pré-condições) e saída (pós-condições).

Tabela 6. Descrição detalhada dos casos de uso essenciais do Dynamo Framework

Casos de uso	Descrição	Entradas e pré-condições	Saídas e pós-condições
[CDU-01] Definir domínios que farão parte da integração	Como pré-requisito para iniciar a integração entre domínios, o usuário deve definir quais os domínios que participarão do processo, fornecendo as informações referentes a cada um deles.	• Devem ser informados os dados dos domínios que serão integrados (Identificador, nome e certificado digital dos nós mestres de cada domínio) • Devem ser enviados os arquivos contendo os modelos de metadados de cada um dos domínios.	• O Dynamo Framework processa as informações e faz a conversão para um formato compatível com o sistema de correspondência.
[CDU-02] Calcular similaridades das propriedades do modelo	De posse das informações acerca da estrutura do modelo de dados de cada domínio a ser integrado, o sistema de correspondência irá realizar um processamento interno e retornar o coeficiente de similaridade entre cada um dos elementos dos modelos.	• O usuário deve fornecer informações sobre cada um dos domínios, como a identificação e o modelo de metadados. • Para possibilitar a compatibilidade com os sistemas de correspondência, os arquivos dos modelos devem ser processados e representados numa estrutura de dados de conjuntos.	• O resultado da análise das correlações dos elementos dos modelos é construído na forma de conjuntos chave-valor, contendo os elementos correlacionados entre os modelos, juntamente com o coeficiente de correlação, representado através de número contido no intervalo [0,1]
[CDU-03] Confirmar similaridades dos modelos	Após receber do sistema de correspondência os coeficientes de correlação, o usuário deverá ter acesso a esses dados e poderá selecionar quais elementos sugeridos pelo sistema de correspondência são os mais adequados aos domínios que participam da integração.	• O usuário deve possuir acesso aos coeficientes de correlação gerados pelo sistema de correspondência.	• Após a confirmação do usuário, o meta-modelo genérico será construído e armazenado, juntamente com o mapeamento dos modelos dos domínios para o meta-modelo criado, possibilitando a execução de consultas posteriores.
[CDU-04] Visualizar domínios integrados	O usuário deve ter acesso às informações dos domínios cuja integração já foi realizada. Podendo opcionalmente alterar ou refazer algum mapeamento realizado anteriormente.	• Para o caso de alteração ou remoção, deve ser informado o par de modelos de metadados alvos da ação	• O usuário receberá uma mensagem confirmando a execução da operação.
[CDU-05] Realizar consultas nos sistemas integrados	O usuário poderá solicitar consultas às bases de dados integradas. Tais consultas podem ter a característica de apenas leitura (pesquisa) ou escrita (adição/atualização/remoção) de dados.	• O usuário deve informar o domínio destino da consulta, como também a ação a ser realizada, através da montagem da consulta com seus respectivos parâmetros.	• No caso de uma consulta de pesquisa, o retorno será o conjunto de informações, construído no formato compatível com o meta-modelo genérico. Já no caso de uma solicitação de escrita o retorno será uma mensagem de confirmação da realização da ação.

A Tabela 7 descreve a relação entre os problemas citados por Haslofer e Klas (HASLHOFER e KLAS, 2010) e os requisitos propostos.

Tabela 7. Relação entre os requisitos e os problemas a serem solucionados

Requisito	Problemas
[CDU-01] Definir domínios que farão parte da integração	P1, P2
[CDU-02] Calcular similaridades das propriedades do modelo	P2
[CDU-03] Confirmar similaridades dos modelos	P2
[CDU-04] Visualizar domínios integrados	P1, P3
[CDU-05] Realizar consultas nos sistemas integrados	P1, P2, P3

5.1.2 Ferramentas e tecnologias

O Dynamo Framework foi implementado sobre a plataforma Java Enterprise Edition (Java EE), utilizando componentes da especificação EJB (Enterprise Java Beans) 3.0. A Figura 19 apresenta a pilha de componentes de software que compõe a arquitetura geral do Dynamo Framework.

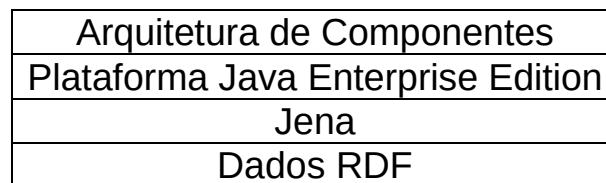


Figura 19. Arquitetura de implementação do Dynamo Framework

5.1.2.1 Java Enterprise Edition (Java EE)

Java é uma linguagem de programação e uma plataforma de computação lançada pela primeira vez pela Sun Microsystems em 1995. É a tecnologia que capacita muitos programas da mais alta qualidade, como utilitários, jogos e aplicativos corporativos, entre muitos outros, por exemplo. O Java é executado em mais de 850 milhões de computadores pessoais e em bilhões de dispositivos em todo o mundo, inclusive telefones celulares e dispositivos de televisão (ORACLE, 2012).

Java Platform, Enterprise Edition (ou Java EE, ou EE, ou em português Java Edição Empresarial) é uma plataforma de programação para servidores na linguagem de programação Java. A Plataforma Java (Enterprise Edition) difere-se da Plataforma Java Standard Edition (Java SE) pela adição de bibliotecas que fornecem funcionalidade para implementar software Java distribuído, tolerante a falhas e multi-camada, baseada amplamente em componentes modulares executando em um servidor de aplicações. A plataforma Java EE é considerada um padrão de desenvolvimento já que o fornecedor de software nesta plataforma deve seguir determinadas regras se quiser declarar os seus

produtos como compatíveis com Java EE. Devido a essas características a plataforma é utilizada principalmente para o desenvolvimento de aplicações corporativas.

5.1.2.2 Dados RDF

Todos os artefatos gerados durante a integração de um novo domínio são persistidos em uma base de dados RDF, gerada de acordo com o meta-modelo resultante do mapeamento entre os modelos de dados de cada domínio a ser integrado. No meta-modelo são expressos conceitos como classes, propriedades e relacionamentos do modelo unificado a ser descrito.

Fazendo uso dos dados RDF, o Dynamo Framework é capaz de realizar acesso a bases de dados RDF de domínios externos, que implementam o protocolo SPARQL. Possibilitando inclusive o roteamento de requisições a propriedades de meta-modelos advindos de mapeamentos com domínios externos (terceiros). No caso de um roteamento, os metadados dos modelos das aplicações integradas ficam todos na base de dados RDF local e as consultas são realizadas remotamente.

5.1.2.3 Jena

Jena (APACHE, 2012) é um framework Java que provê um ambiente programático para manipulação de RDF, RDFS e OWL, SPARQL. O Jena é um projeto de código aberto mantido pela incubadora de projetos da Apache Foundation.

O framework Jena inclui uma API RDF, que permite leitura e escrita de RDF nas notações RDF/XML; além do armazenamento em memória ou persistente (arquivos, base de dados nativa e base de dados relacionais) e uma máquina de consulta SPARQL.

Na estrutura interna do Dynamo Framework, o framework Jena cumpre o papel do ambiente RDF que compõe a camada de armazenamento do meta-modelo e as consultas sobre ele realizadas.

A base de dados RDF local do Dynamo Framework pode ser persistida em quaisquer das opções de armazenamento do Jena, porém foi utilizada a base de dados padrão, que usa o modo de armazenamento em sistema de arquivos.

Há outros frameworks RDF disponíveis no mercado, sendo que, entre os critérios de escolha, foi dada prioridade aos frameworks gratuitos que pudessem ser distribuídos junto com o Dynamo Framework. Além do Jena, podemos citar opções como o Sesame (ADUNA, 2011) e o Redland (BECKETT, 2001). Além do ser gratuito, a escolha pelo JENA

se deu pelo fato do mesmo possuir funcionalidades que atendiam requisitos importantes para resolver os problemas de interoperabilidade descritos ao longo deste trabalho.

5.1.3 Arquitetura de componentes

O framework possui um conjunto de componentes responsáveis por gerenciar e armazenar as informações sobre os modelos de dados integrados, na forma de meta-modelos, que serão construídos dinamicamente a partir do mapeamento das similaridades entre os modelos e armazenados na base de modelos integrados. A Figura 20 mostra essa base de modelos integrados, juntamente dos demais componentes que compõem essa camada, cuja descrição será feita logo em seguida.

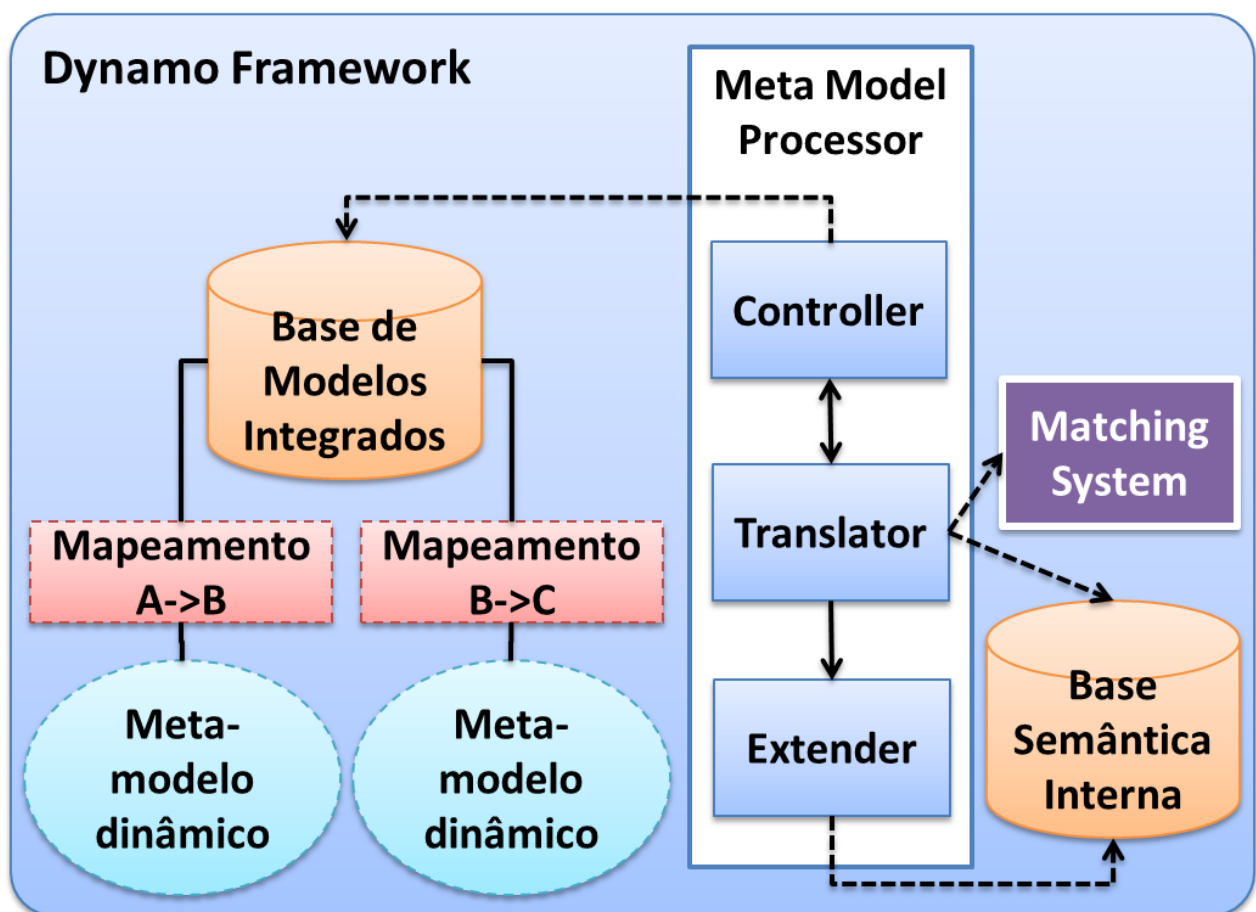


Figura 20. Visão Geral de Arquitetura do Dynamo Framework

5.1.3.1 Meta Model Processor (Processador de modelos de metadados)

Conjunto de componentes responsáveis por fazer o processamento dos modelos de metadados. Possui a função tanto de receber, como de prover informações para a interface de administração. No momento da integração com uma nova base de dados, recebe a descrição do modelo (esquema de metadados) utilizado pelas bases que serão integradas e através dos seus componentes *Controller*, *Translator* e *Extender*, consegue detectar

inicialmente as correlações e gerar dinamicamente, em um segundo momento (após o feedback do usuário em relação às correspondências) o meta-modelo genérico, resultado da integração, que será armazenado para uso posterior.

A Figura 21 mostra uma visão geral das classes utilizadas na implementação dos componentes do Meta Model Processor. Em seguida esses componentes são detalhados.

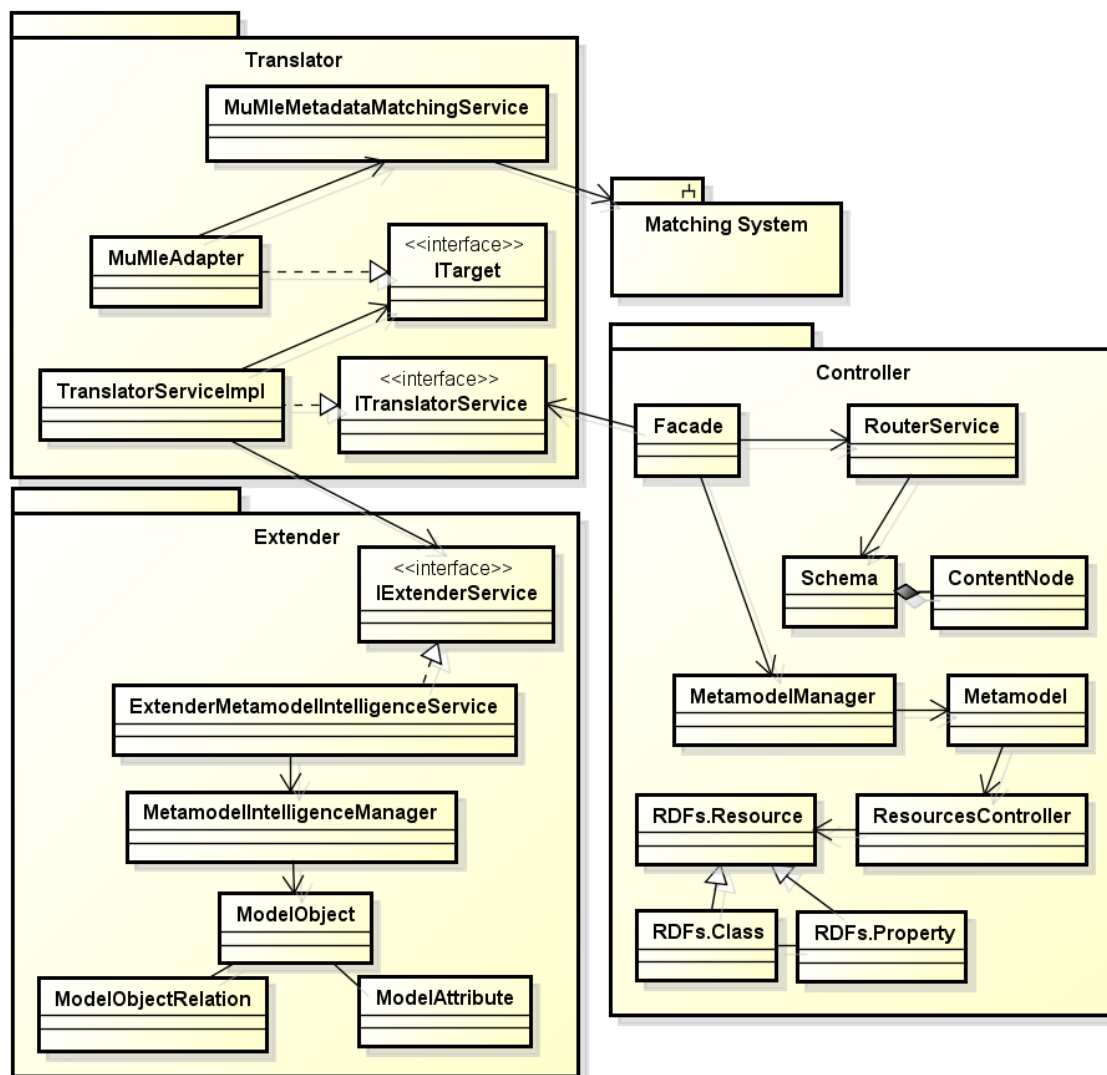


Figura 21. Visão geral das classes presentes em cada módulo dentro componente Meta Model Processor

5.1.3.2 Controller (Módulo de Controle)

É a porta de entrada do Dynamo Framework para os serviços externos que desejam utilizar os recursos do *framework*. No processo de integração sua função é receber do usuário, via uma interface proveniente de um sistema externo, as informações sobre o esquema de metadados do domínio que será integrado, e em seguida repassá-las para o *Translator*. Após o *Translator* processar as informações, o *Controller* devolve para a interface o esquema de metadados, agrupando os elementos relacionados e

acrescentando um coeficiente de correlação em cada um deles. Após a fase de integração, a função do *Controller* é realizar operações de leitura ou escrita na base que contém o meta-modelo genérico.

Para a implementação do módulo de controle, foi utilizado como base o padrão de projeto Fachada (*Facade*) (GAMBLE, 1999). A fachada coordena as requisições e provê uma interface simples para acessar subsistemas complexos. Isso permite a construção de interfaces customizadas sem ter de se preocupar com a compatibilidade em relação aos componentes de mais baixo nível. A Figura 22 apresenta as classes que compõem o módulo de controle. Em seguida é explicada a função de cada uma.

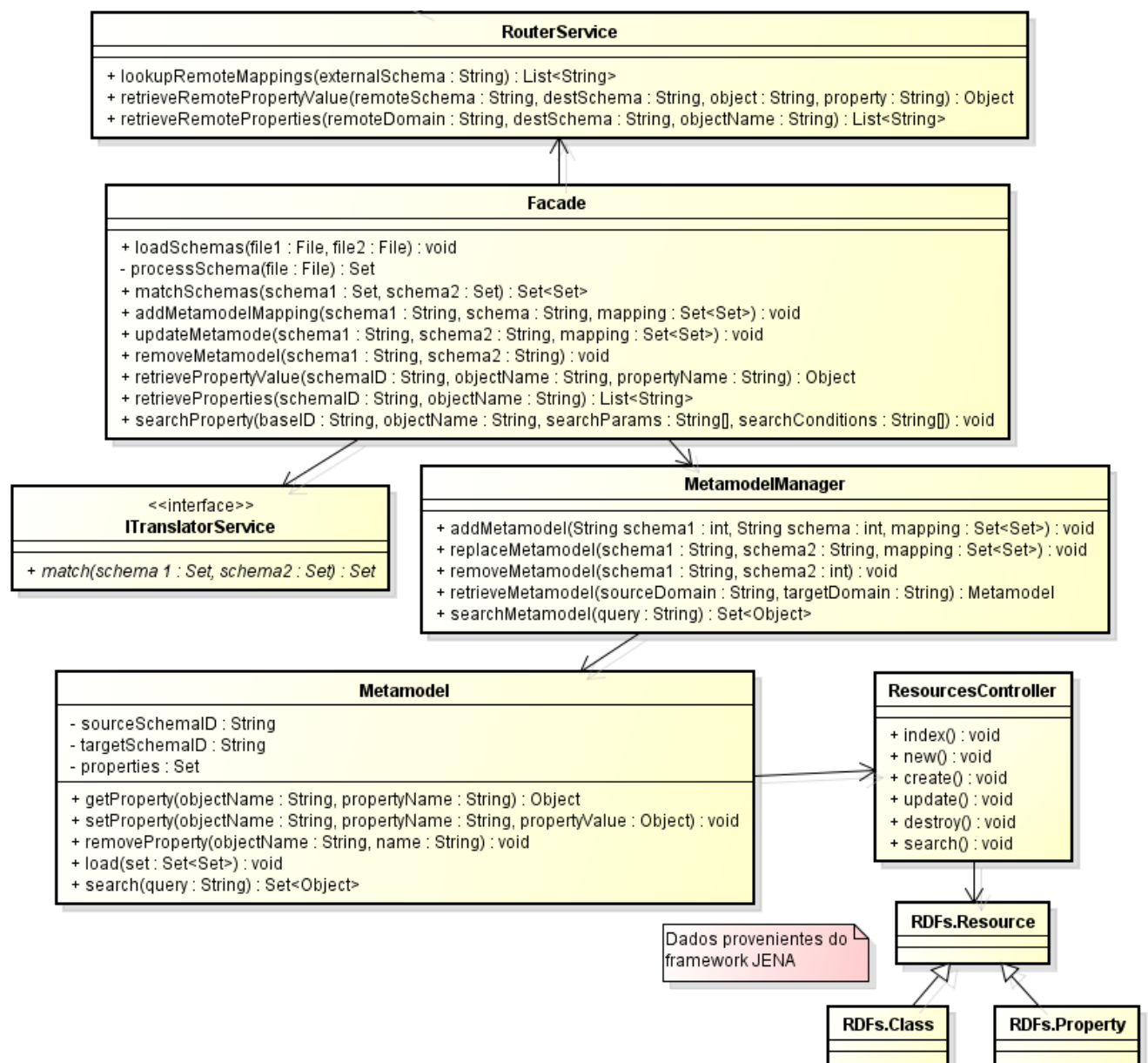


Figura 22. Classes envolvidas no módulo de controle

A classe “RouterService” representa o serviço de roteamento, dando suporte a integração de bases de terceiros, através da reutilização de um mapeamento feito com outra base de dados. Por exemplo, em um cenário hipotético contendo três bases de dados (TV Brasil, BBC e Cinemateca), a integração da TV Brasil com a BBC resultará num meta-modelo contendo o mapeamento do modelo de dados das duas instituições, da mesma forma teremos o mesmo com um mapeamento entre BBC e Cinemateca. Entretanto, caso a TV Brasil queira se integrar à Cinemateca, ao invés de realizar um novo processo de integração entre as duas instituições, um caminho alternativo seria reaproveitar o mapeamento realizado anteriormente para a BBC, nesse caso, a TV Brasil solicitaria acesso à Cinemateca via BBC, fazendo uso dos recursos de roteamento.

A classe “Facade” (Fachada) é o ponto de entrada do componente *Controller*, de tal forma que é a única que recebe requisições diretamente dos serviços externos que porventura façam uso do *framework*. Sua responsabilidade é fornecer uma interface de acesso centralizada, de maneira que os componentes internos do *Controller* possam ser modificados, sem demandar alterações no módulo de interface de administração. Na fachada estão concentradas chamadas para os demais serviços do *Controller*.

A interface “ITranslatorService” é o ponto de acesso ao componente *Translator*. Sua estrutura, componentes com os quais se comunica e aplicabilidade são expostos na seção 5.1.3.3.

A classe “MetamodelManager” possui a responsabilidade de gerenciar os Meta-modelos na forma de objetos (classe “Metamodel”), possuindo métodos para recuperar, adicionar, atualizar e remover propriedades armazenadas no meta-modelo genérico.

A classe “Metamodel” faz acesso direto às propriedades do meta-modelo genérico. Sendo possível carregar os dados armazenados no formato RDF e a partir disso recuperá-los ou alterá-los.

A classe “ResourcesController” interage diretamente com os dados armazenados em RDF, fazendo uso das APIs disponibilizadas pelo *Framework Jena*. Essa classe realiza a leitura dos arquivos e repassa as informações em forma de objeto para o “Metamodel”, como também recebe dados do “Metamodel” e escreve no arquivo RDF correspondente.

5.1.3.3 *Translator (Módulo de Tradução)*

Responsável por receber as informações do esquema de metadados do *Controller* e processar as informações utilizando algum mecanismo de correspondência de metadados. O mecanismo de correspondência utilizado foi o MuMle (Multi-level Metadata Mapping System) (S. AMIR et al., 2011), devido à sua eficácia, visto o alto nível de acerto. Sua função é receber como informações do esquema de metadados das bases de dados que serão integradas. A saída fornecida são os coeficientes de correlação entre cada elemento dos modelos.

Após receber os resultados do mecanismo de correspondência, o *Translator* repassará para o *Extender*, cuja responsabilidade será armazenar as informações mapeadas numa base local de conhecimento, para que possam ser utilizadas em operações futuras.

O *Translator* foi implementado utilizando como base os padrões de projeto *Bridge* e *Adapter* (GAMBLE, 1999). O *Bridge* possibilita a conversão de dados de uma forma para outra em componentes, sem afetar os seus componentes clientes, permitindo assim que sejam realizadas modificações nos componentes de mais baixo nível de forma independente. Nesse caso, o *Bridge* é utilizado para dar compatibilidade à realização do cálculo das correspondências tanto através de um sistema externo, como através do acesso às informações internas armazenadas no serviço do *Extender*, de forma que possíveis mudanças nos componentes de acesso interno ou externo não afetarão os componentes que fazem acesso ao *Translator*. Já o *Adapter* permite que classes com interfaces incompatíveis possam interagir. Como o *Translator* faz o papel de um sistema cliente do sistema de correspondência (componente externo), o padrão *Adapter* é usado para mapear a interface do sistema de correspondência externo, de forma que ela se torne compatível com a interface de acesso do *Translator*.

A Figura 23 mostra as classes envolvidas no módulo de tradução. Em seguida são detalhadas as funções de cada uma.

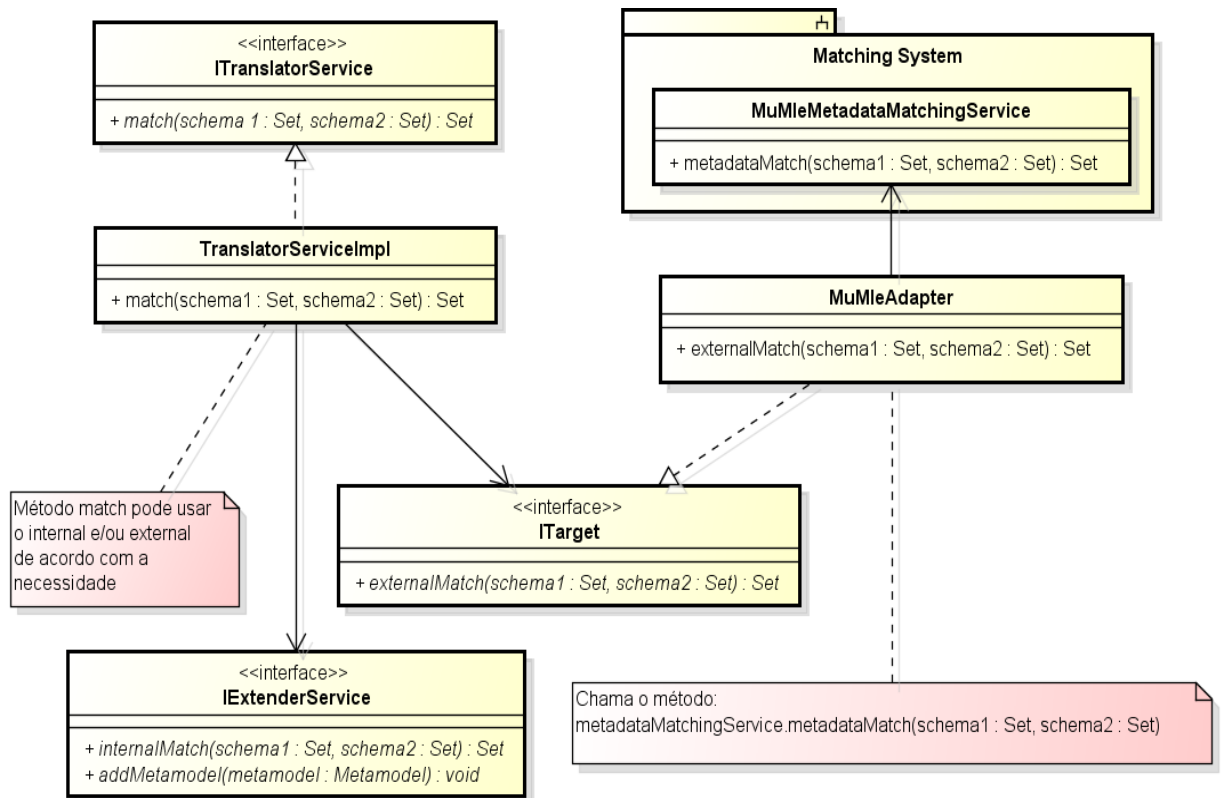


Figura 23. Classes que compõem o módulo de tradução

A classe “TranslatorServiceImpl” segue a interface “ITranslatorService” e serve como ponto de entrada para o módulo *Translator*, cuja principal função é realizar o mapeamento automático das propriedades de dois modelos de dados passados. O “TranslatorServiceImpl” recebe os dados já processados pelo *Controller*, numa estrutura de dados de conjunto e a partir desse momento realiza o processo do mapeamento em duas etapas: (1) Passa o conjunto de dados contendo as informações dos modelos que serão integrados para o sistema de correspondência (acesso externo) e (2) passa as mesmas informações para o componente *Extender*, através do acesso à interface “IExtenderService”, que fará a consulta baseada nos dados obtidos de integrações anteriores. Desta forma, para possibilitar o acesso às duas fontes (externa e interna), a classe “TranslatorServiceImpl” possui uma referência para a interface “IExtenderService”, como também para a interface “ITarget”, que é a interface compatível com o Dynamo Framework e será para ela que “MuMleAdapter” fará a adaptação da interface fornecida pelo sistema de correspondência externo, no caso, o MuMle.

A classe “MuMleAdapter” implementa a interface “ITarget”, de forma que os dados por ela retornados serão compatíveis com o “TranslatorServiceImpl”. Da mesma forma, o “MuMleAdapter” possui uma referência para o serviço de correspondência de Metadados MuMIE, e realizará toda a conversão de dados necessária, traduzindo as informações do

formato presente no MuMle para o que seja compatível com o Dynamo Framework. Uma vantagem da adoção do padrão de projeto *Adapter* nesse caso, é que torna possível a utilização de outros sistemas de correspondência de metadados, sem realizar nenhuma alteração nos componentes internos do *Translator*, sendo necessária apenas a criação de outras classes “adaptadoras”.

Por fim, a interface “*IExtenderService*” é o ponto de entrada para o módulo *Extender*, que retornará as sugestões de correlação de elementos, só que baseado no histórico de integrações realizadas anteriormente. A descrição das classes que implementam essa interface é realizada na seção 5.1.3.4.

5.1.3.4 Extender (Módulo de Extensão)

Responsável por realizar operações que irão ampliar o domínio de conhecimento a cada nova integração que é realizada. O *Extender* armazena as informações que foram mapeadas pelo *Translator* numa base de conhecimento, de forma que elas possam ser utilizadas em um momento posterior, caso domínios similares venham a ser integrados, dispensando um novo acesso ao mecanismo de correspondência.

A implementação do módulo *Extender* envolve basicamente a criação de classes para acessar a base de inteligência do meta-modelo e retornar essas informações num formato compatível com a interface “*IExtenderService*” que é fornecida ao componente *Translator*. A Figura 24 apresenta as classes que compõem o módulo de extensão. Em seguida são detalhadas as funções de cada uma.

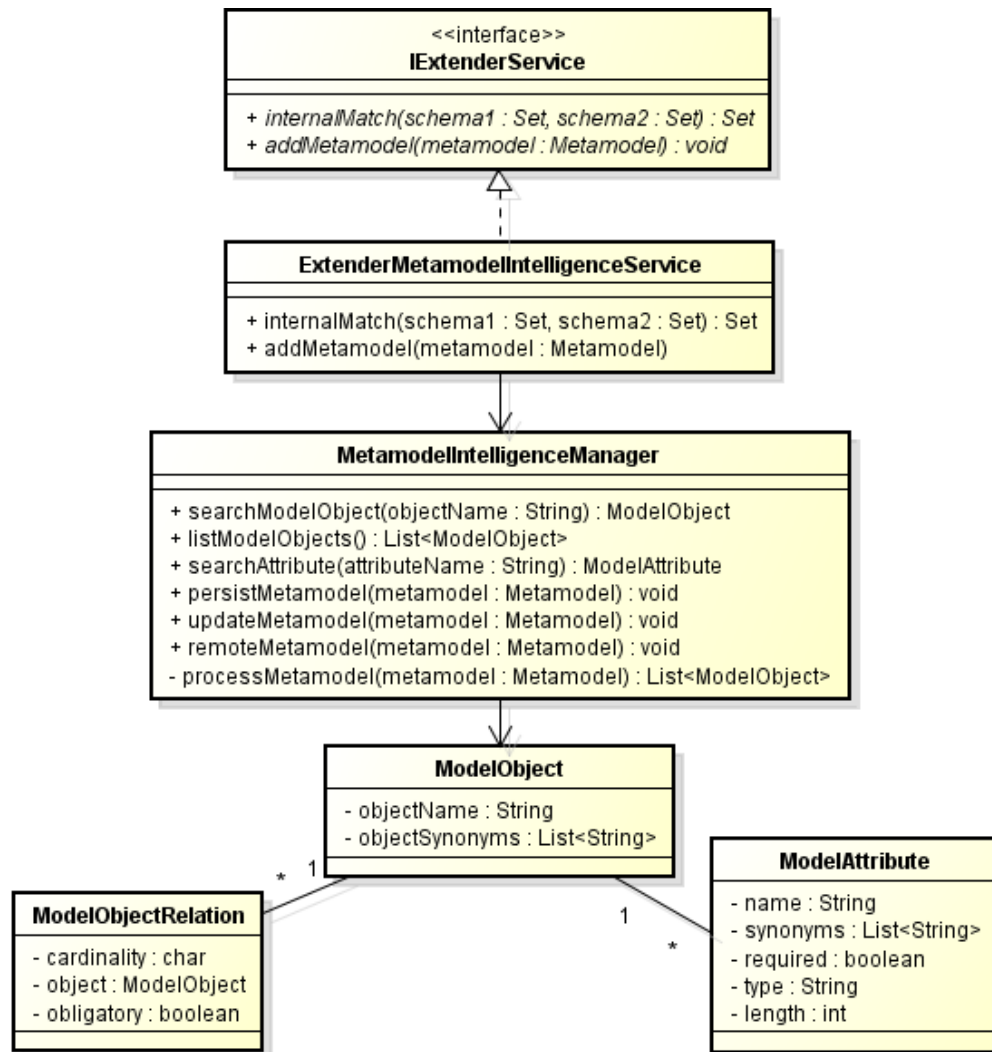


Figura 24. Classes que compõem o módulo de extensão

A classe “ExtenderMetamodelIntelligenceService” implementa a interface “IExtenderService”, que é o ponto de acesso ao módulo *Extender*. Possui duas funções básicas: (1) Realizar o mapeamento de dois modelos fornecidos, baseando-se na base de inteligência; (2) Adicionar informações à base de inteligência, passando um meta-modelo. O acesso à essa base é feito por intermédio do “MetamodelIntelligenceManager”.

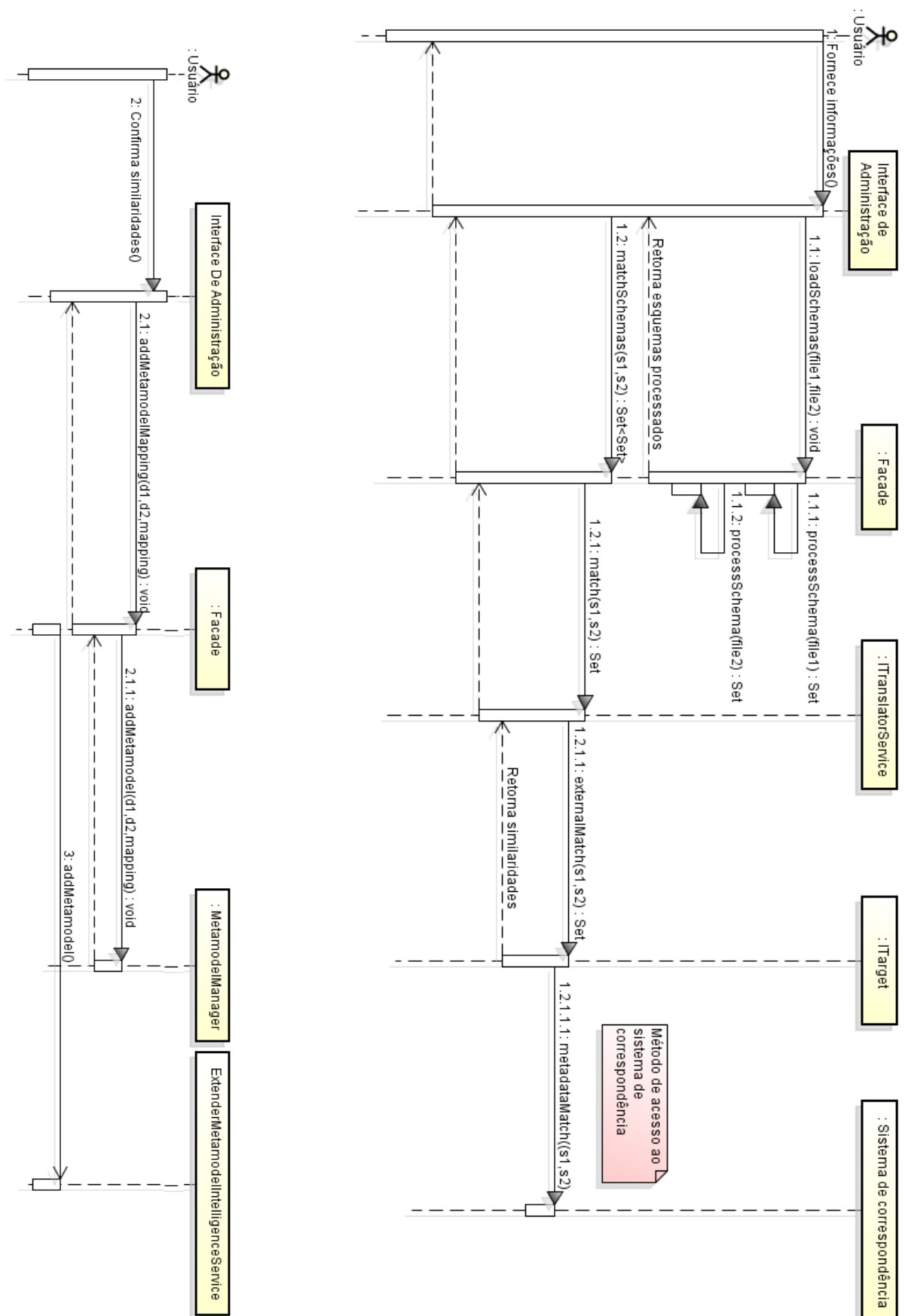
A classe “MetamodelIntelligenceManager” controla toda a base de inteligência construída de acordo com os mapeamentos realizados anteriormente. É através dessa classe que é possível manipular os dados da base (inserir, atualizar, remover), ou até mesmo passar conjuntos de elementos dos modelos que estão sendo integrados e obter a informação sobre a correlação dos objetos e propriedades.

A classe “ModelObject” é o tipo de dado que armazena a informação de um objeto de um dado mapeamento. Trazendo o nome do objeto e quais nomes de objeto seriam similares. Essa classe contém uma lista de “ModelAttribute” que representa cada atributo

do objeto, trazendo informações como o nome, lista de sinônimos, obrigatoriedade, tipo (integer, string, double etc) e tamanho (número de caracteres). A classe “ModelObject” também possui uma lista de “ModelObjectRelation”, que representa as relações do objeto em questão, trazendo informações como o objeto que é relacionado, a cardinalidade do relacionamento e a obrigatoriedade.

5.1.3.5 Colaboração entre os módulos

O diagrama de sequência apresentado na Figura 25 ilustra a sequência de colaboração entre os módulos em um caso normal de navegação na aplicação.



(a) (a)

Figura 25. Diagrama de sequência para a confirmação das similaridades (a); Diagrama de sequência para os processos de coleta de informações dos domínios que serão integrados e cálculo das similaridades (a);

As interações entre o usuário e a aplicação sempre ocorrem através do módulo de interface de administração, por meio do envio de requisições do protocolo HTTP à aplicação. Já as operações que envolvem acessos externos (e.g. sistema de correspondência) são invocadas na forma de Web Services.

A seguir é descrita passo-a-passo a sequência de colaboração ilustrada na Figura 25:

Passo 1: Fornecer informações das bases que serão integradas – São fornecidas informações sobre as bases de dados que serão integradas, tais como identificador, nome, descrição, e o mais importante, esquema de metadados, que é representado através de um arquivo construído com base numa linguagem de definição de esquema de metadados (e.g. XML Schema, RDFS).

Passo 1.1: loadSchemas(file1, file2) – Os dados são recebidos via o componente “Facade”, que é parte integrante do módulo de Controle (*Controller*), que por sua vez processa os esquemas de cada domínio em questão (etapas 1.1.1 e 1.1.2 do diagrama), a fim de transformar a representação de uma linguagem de definição de esquema para uma estrutura de dados de conjuntos em Java (*Set*). O resultado do processamento é retornado para a interface de administração, que por sua vez exibe os elementos dos modelos processados para o usuário.

Passo 1.2: matchSchemas(schema1, schema2) - Para dar início ao processo de análise das correlações dos modelos de metadados, a interface repassa os esquemas processados para a classe “Facade”, que por sua vez aciona a classe que implementa a interface “ITranslatorService”.

Passo 1.2.1: match(schema1, schema2) - O componente da interface “ITranslatorService” possui a responsabilidade de fazer o papel de mediador entre a fachada (Facade) que interage com a interface do usuário e as fontes que fornecerão os coeficientes de correlação (um ou mais sistemas de correspondência ou consulta interna à base de conhecimento do componente *Extender*).

Passo 1.2.1.1: externalMatch(schema1, schema2) – Considerando um cenário com o acesso a um sistema de correspondência externo, é realizada uma chamada ao método `externalMatch` da classe que implementa a interface “ITarget”. Essa classe é considerada a classe adaptadora, cuja responsabilidade é converter a representação dos esquemas de metadados para um formato compreensível pelo sistema

de correspondência em questão e realizar a chamada para o método no sistema de correspondência que irá retornar os coeficientes de similaridade entre os modelos (passo 1.2.1.1.1 do diagrama). Os dados obtidos são enviados de volta para a Interface de Administração, que os apresenta ao usuário, aguardando a confirmação das similaridades (passo 2).

Passo 2: Confirmar similaridades – De posse das informações fornecidas pelo sistema de correspondência, caberá ao usuário, via interface de administração, confirmar se as relações sugeridas estão de acordo com o modelo de negócios das bases a serem integradas, podendo fazer ajustes no mapeamento caso necessário.

Passo 2.1: addMetamodelMapping(schema1, schema2, mapping) – Após a confirmação do usuário, a interface de administração recebe os dados e invoca o método `addMetamodelMapping(schema1, schema2, mapping)`, da classe “Facade”. São passados via parâmetro as informações informadas dos dois domínios que serão integrados e o mapeamento das similaridades.

Passo 2.1.1: addMetamodel(schema1, schema, mapping) – Ao receber os dados da “Facade”, a classe “MetamodelManager” os converte de uma representação de estrutura em dados em Java para RDF. Em seguida são acionados os componentes do Framework *Jena*, que irá persistir o meta-modelo construído com base no mapeamento dos domínios. Após a conclusão do processo, é enviado o retorno para a interface de administração, que exibe a mensagem de confirmação da execução do processo para o usuário.

Passo 3: addMetamodel() – Logo após o meta-modelo ser persistido, o mapeamento proveniente da análise dos dois modelos é armazenado na base de inteligência do componente *Extender*, permanecendo presente mesmo que num momento futuro a integração entre os domínios que deram origem a esse mapeamento seja desfeita (usuário solicite a remoção do meta-modelo). Esse mapeamento que é armazenado no *Extender* trata-se das similaridades semânticas e estruturais encontradas entre os elementos do modelo. Essa base poderá ser utilizada em integrações posteriores, como forma de complementar o mapeamento das similaridades.

5.2 Arquitetura do modelo de interoperabilidade baseado na abordagem de meta-modelos dinâmicos

A partir da pesquisa em trabalhos relacionados, a Seção 4.1 apresentou diversas propostas de arquiteturas para integração. Uma adaptação combinada dessas abordagens foi realizada para dar base ao modelo de interoperabilidade utilizado pelo Dynamo Framework.

O modelo de interoperabilidade adotado pelo Dynamo Framework faz uso da abordagem de combinação das bases de dados integradas em um modelo genérico, da mesma forma que o COIN e o DWQ (vide Seção 4.1.3). No entanto, é realizada uma adaptação que é a adoção de um modelo genérico para a integração entre cada par de bases de dados, resultando em múltiplos modelos genéricos, abordagem semelhante à proposta pelo OBSERVER (vide Seção 4.1.3).

O resultado do processo de integração é armazenado na Base de Modelos Integrados (aquí nomeada de BMI), que é constituída pelo conjunto de Modelos Genéricos, daqui em diante chamados “MG”, que correspondem ao resultado do mapeamento entre dois modelos de bases de dados distintas, daqui em diante chamado “MBs” (para melhor entendimento ver Figura 26).

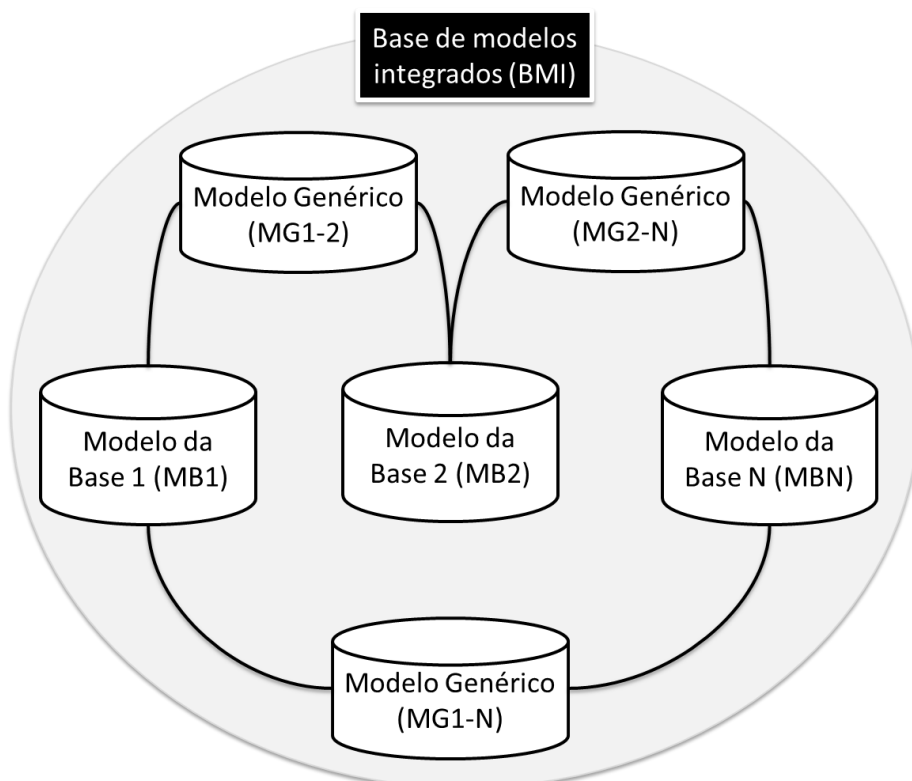


Figura 26. Arquitetura da camada conceitual do meta-modelo

Após a realização das etapas para o mapeamento entre duas bases dados (mais detalhes na Seção 5.3), é construído o Modelo Genérico (MG). Cada MG corresponde a uma visão unificada dos Modelos das Bases (MBs). O MG consiste de uma representação global contendo os conceitos e relações que são de interesse para ambas as bases. No caso em estudo, o interesse são informações advindas de sistemas de gerenciamento de ativos digitais. Cada MB corresponde a uma base disponível, ou pelo menos parte dela, visto que não é necessário que uma base seja totalmente conceitualizada (pode-se fazer uma pré-seleção da parte da base que será usada na integração ou pode-se realizar uma abordagem incremental).

A Base de Modelos Integrados (BMI) contém o conjunto de MGs, após sua interação com os MBs. A BMI pode não conter uma representação completa de todas as bases de dados (e.g. abordagem incremental), mas representa uma descrição consolidada dos conceitos e relações importantes até o momento. Tal descrição está sujeita a mudanças à medida que novos MBs são integrados ou alterados.

Além dos modelos, a BMI também contém a especificação das interdependências entre elementos de diferentes MBs e, entre os MBs e o MG. O processo de integração deve mapear as relações entre os modelos (MB para MB e MBs para MG) e o mapeamento do modelo genérico é realizado a cada vez que uma nova base é incluída (sob-demanda).

Apesar de utilizar um mapeamento par a par, é possível obter o mapeamento entre bases que não estejam integradas diretamente, através do recurso de roteamento presente no Dynamo Framework. Dessa maneira, mesmo que o modelo MB1 citado na Figura 26 não tivesse sido mapeado para um Modelo N (MG1-N), como existe um MG mapeando MB1 e MB2, e o MB2 é mapeado junto ao MBN (MG2-N), o MB1 poderia realizar a integração com MBN através do roteamento por MB2. A desvantagem em utilizar essa abordagem é que o MB1 só visualizaria os elementos de MBN que foram considerados pelo MB2.

Considerando os níveis de modelagem de metadados (vide Seção 2.3.2), os MGs fazem o mapeamento no nível de modelo dos MBs, resultando em um meta-modelo compatível com ambos MBs. Por exemplo, se no MB1 o atributo que representa o título de um elemento é definido por “title” e no MB2 o mesmo atributo é definido por “name”, o meta-modelo do MG irá conter a informação que o atributo “title” do MB1 é compatível com “name” do MB2. A Figura 27 mostra os níveis de modelagem presentes em cada MB.

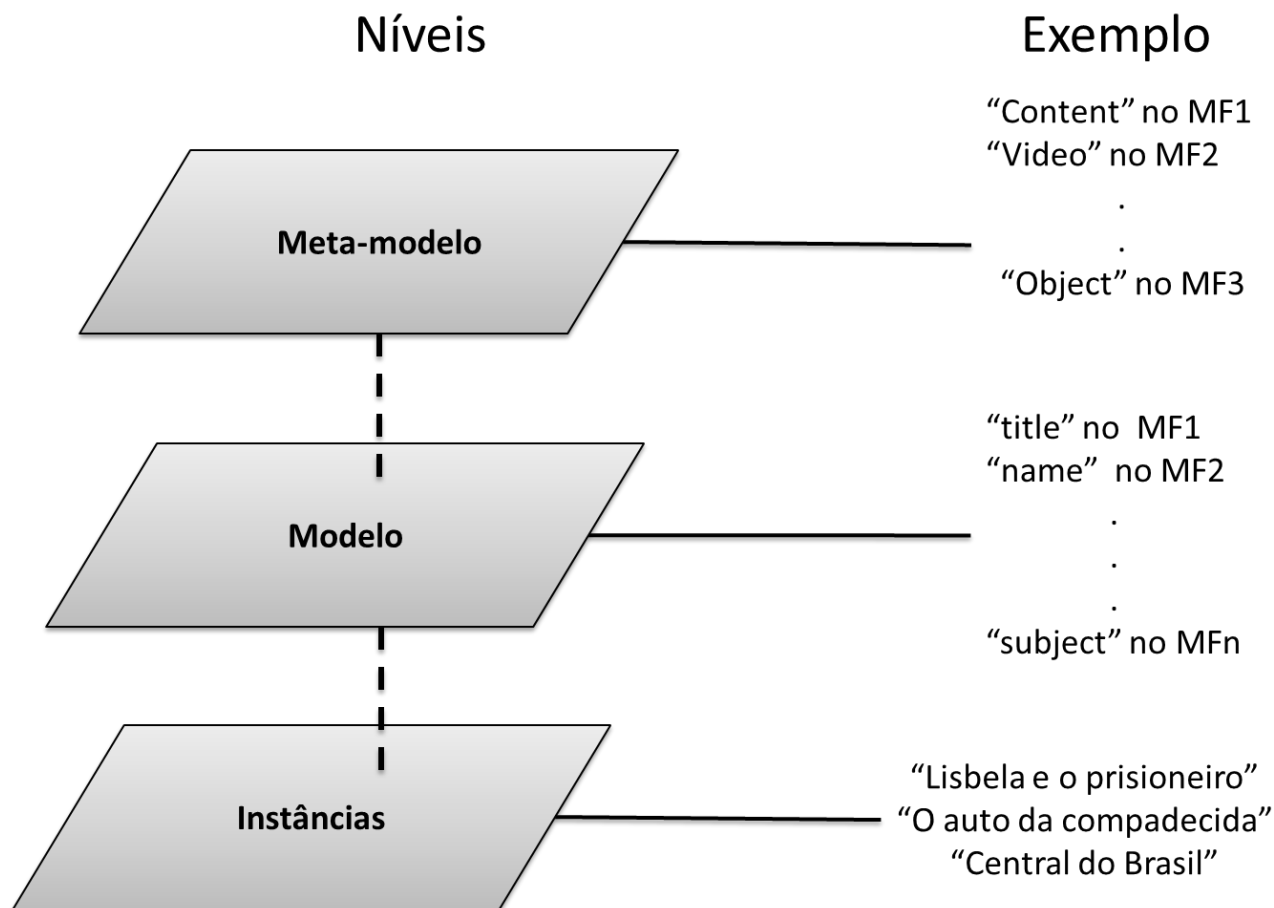


Figura 27. Níveis de modelagem

Em resumo, a Base de Modelos Integrados (BMI) contém o resultado final da integração entre diversas bases de dados heterogêneas. É a BMI que fornecerá para o Dynamo Framework as informações provenientes das bases integradas.

Ressalta-se que a BMI é parte integrante do Dynamo Framework, e que somente a BMI isolada não seria suficiente para alcançar a interoperabilidade entre bases distintas, sua responsabilidade é o armazenamento do mapeamento das bases integradas. Os demais componentes do Dynamo Framework possuem um papel fundamental em todo o processo, desde a interface de entrada para receber os modelos e possibilitar a participação do usuário, a detecção de correlações entre elementos e a integração dos demais componentes do *framework* com a própria Base de Modelos Integrados.

5.3 Etapas para a interoperabilidade de metadados entre bases de dados heterogêneas

As subseções anteriores apresentaram uma visão geral do funcionamento e da arquitetura de componentes, além do modelo de interoperabilidade adotado pelo Dynamo Framework. Esta subseção apresenta a sequência de etapas utilizada para alcançar a interoperabilidade de metadados entre bases de dados heterogêneas, descrevendo a função dos componentes do Dynamo Framework em cada uma delas.

Essas etapas foram definidas com base no fluxo de execução do Dynamo Framework para realizar o processo de interoperabilidade de metadados entre sistemas de gerenciamento de ativos digitais, são elas:

1. Definição das bases de dados que farão parte da integração: o framework recebe, através do componente Controller, os arquivos contendo os esquemas de metadados das bases de dados que serão integradas. Em seguida,
2. Projeção e transformação: ocorre a comparação de elementos pelo sistema de correspondência de metadados, no nível de linguagem (M2). Baseado no resultado, as representações originais, que podem estar em linguagens diferentes, são transformadas numa estrutura única compatível com o *framework*.
3. Correspondência de elementos dos modelos: é invocado o sistema de correspondência de metadados para realizar a comparação de cada um dos elementos dos esquemas de metadados no nível de modelo (M1). O resultado traz a lista de elementos correlacionados, juntamente dos respectivos coeficientes de similaridade.
4. Confirmação dos mapeamentos sugeridos: ocorre a participação do usuário, através de interface gráfica existente numa aplicação externa, para validar o mapeamento sugerido na etapa de correspondência.
5. Construção e representação dos meta-modelos: é construído efetivamente o meta-modelo genérico contendo as informações do mapeamento após a confirmação do usuário.
6. Operações sobre os meta-modelos: após a construção do meta-modelo, torna-se possível o acesso entre as bases integradas. Assim, operações de leitura ou escrita podem ser executadas.

A Figura 28 ilustra a ligação entre as etapas citadas, mostrando quais componentes do Dynamo Framework são utilizados em cada uma delas. Em seguida cada etapa é descrita de forma detalhada.

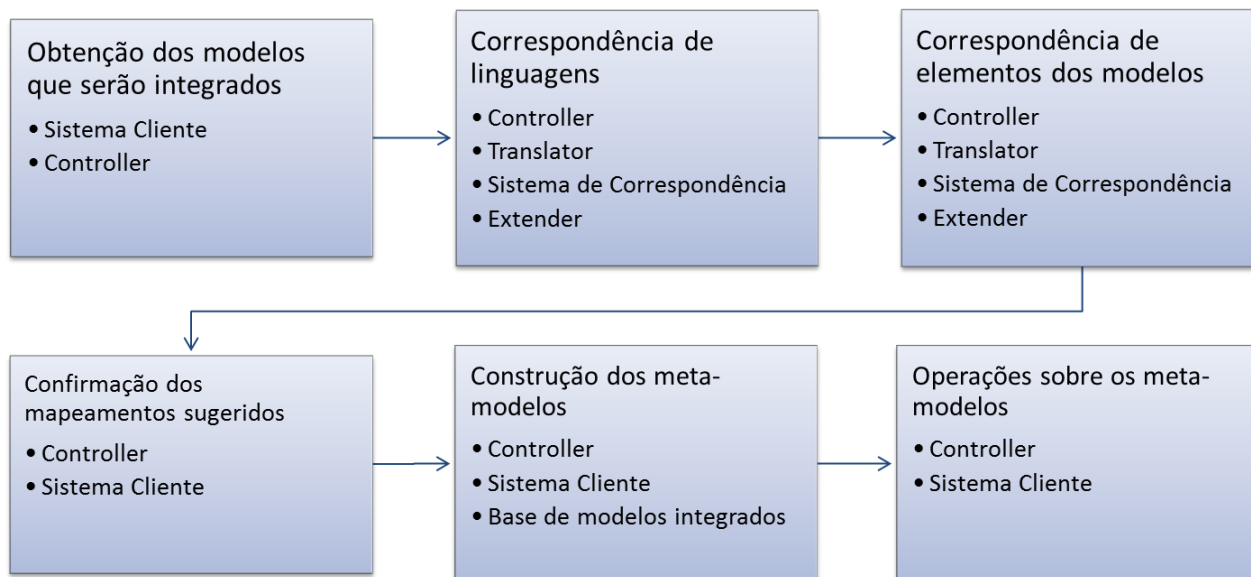


Figura 28. Etapas do fluxo de execução do Dynamo Framework e os componentes utilizados em cada uma

5.3.1 Obtenção dos modelos que serão integrados

A etapa inicial consiste do recebimento dos modelos de dados (esquemas de metadados) das bases de dados que serão integradas. O Dynamo Framework trabalha com a comparação de dois modelos por vez. Assim, recebe-se como entrada um par de esquemas de metadados, correspondente ao modelo de cada uma das bases. O formato de representação das informações dos esquemas de metadados é um arquivo de texto escrito em uma linguagem de definição de esquemas (e.g. XML Schema, RDFSchema, OWL, etc.). O par de arquivos é enviado pelo sistema cliente (sistema que está fazendo uso do framework) como parâmetro de entrada ao Dynamo Framework, que irá processá-los e dar seguimento ao processo.

Uma vez carregados pelo Dynamo Framework, os esquemas são mapeados em vários níveis, desde a linguagem de definição do esquema, até as propriedades dos elementos.

Num cenário hipotético de integração entre dois sistemas de gerenciamento de ativos digitais, o ponto a ser analisado são os esquemas de metadados utilizados pelos sistemas. Caso ambos usem esquemas de metadados similares, o processo de integração

torna-se mais simples, no qual serão relacionados os elementos definidos em cada um e passados para um meta-modelo genérico. Entretanto, no caso de cada sistema usar um esquema diferente de metadados (o que acontece geralmente), é necessário utilizar algum mecanismo adicional para mapear os elementos correlatos e em seguida gerar um modelo que sirva de linguagem comum para ambos.

Considerando o segundo caso (esquemas diferentes), após receber a representação de cada esquema de metadados, também se faz necessário levantar a linguagem que foi utilizada para descrevê-los. Por exemplo, um sistema pode usar um esquema de metadados representado em XML Schema e outro pode ter o esquema representado em RDFSchema. Nesse caso, para identificar as similaridades entre elementos, seria realizado inicialmente o mapeamento em mais alto nível, procurando identificar as informações semânticas e estruturais das linguagens.

Após o levantamento das informações semânticas e estruturais, elas são agrupadas numa estrutura de grafos rotulados. A cada nível do grafo, são mapeadas informações comuns de cada linguagem. Por exemplo, no nível um podem-se mapear as classes, no nível dois as propriedades e assim por diante. De forma que, ao ser realizada a comparação entre dois grafos gerados a partir de dois modelos de metadados diferentes, seja possível interpretar os elementos que possuam o mesmo conceito semântico. Esses grafos são usados como elementos de entrada para a etapa de correspondência de metadados.

A interface de comunicação com o usuário, onde serão enviados os arquivos contendo as informações dos esquemas de metadados, fica a cargo do sistema cliente. O Dynamo Framework recebe os dados através da sua API Java que será acessada a partir do sistema cliente. O componente do Dynamo Framework que recebe os dados de entrada dessa primeira etapa é o “*Controller*” (vide subseção 5.1.3.2), mais especificamente a classe “*Façade*” através do método “*loadSchemas*”. A Tabela 3 descreve os dois métodos que são utilizados durante essa primeira etapa, detalhando os parâmetros de entrada e saída. Outro método citado é “*processSchema*”, um método privado que é chamado internamente pelo método “*loadSchemas*” e realiza o processamento individual de cada arquivo submetido. A Figura 29 mostra um diagrama de classes UML que mostra como a estrutura de grafos rotulados é representada no formato de objetos (classes) no Dynamo Framework.

Tabela 8. Métodos do Dynamo Framework utilizados para a etapa de definição dos modelos que farão parte da integração

Método	Classe	Visibilidade	Entrada	Saída
loadSchemas	Façade	Público	Arquivos contendo os modelos de dados dos sistemas que serão integrados	Par de estruturas processadas em uma representação em Java
proccessSchema	Façade	Privado	Arquivo contendo um a descrição de um esquema de metadados	Representação em Java contendo os elementos do esquema

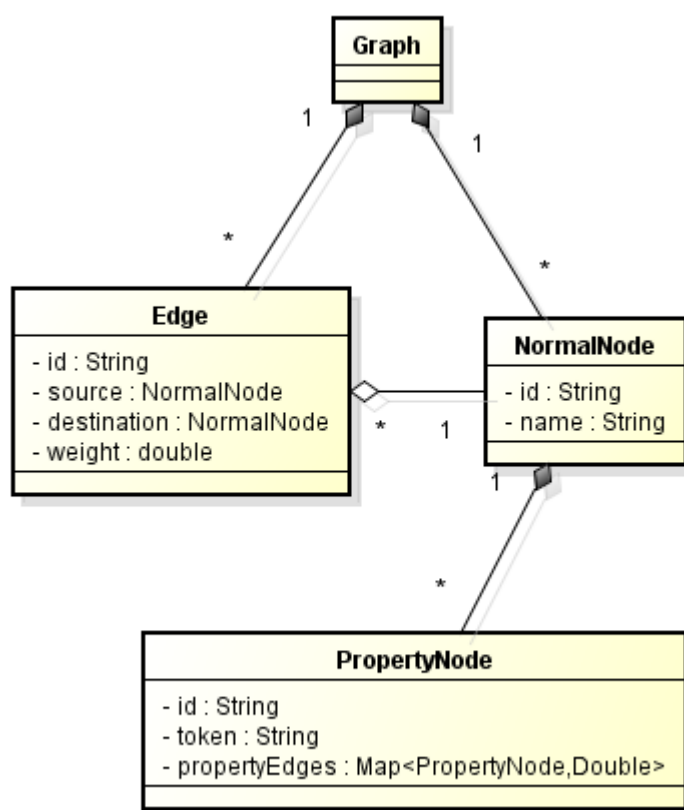


Figura 29. Modelo de classes para representar o grafo processado pelo Dynamo Framework

O grafo é representado pelo objeto denominado “Graph”, que é basicamente formado por um conjunto de nós e arestas, que descrevem as relações entre nós, inclusive definindo um valor de peso, que receberá o coeficiente de correspondência dos nós. Os nós foram divididos em dois tipos: nós normais e nós de propriedade, seguindo a definição utilizada pelo MuMle (vide Seção 4.2.1). Assim, um nó normal se relaciona com outros nós normais, mas também possui relação com vários nós de propriedade. Cada nó de

propriedade possui referência para um conjunto de outros nós do mesmo tipo, que ocorre quando há correspondência entre as propriedades. Da mesma forma, cada relação entre nó de propriedade contém um peso, que recebe como valor o coeficiente de correlação dos nós de propriedade.

Essa estrutura de grafos será utilizada tanto para realizar a correspondência no nível de linguagem (subseção 5.3.2), como para realizar a correspondência no nível de modelo (subseção 5.3.4). No primeiro caso (comparação no nível de linguagem) o grafo vai conter nós contendo informações sobre os elementos da linguagem que o esquema de metadados foi definido. Já no segundo caso (comparação no nível de modelo), o grafo passará a conter nós referenciando os elementos do esquema. Ou seja, a mesma estrutura é utilizada posteriormente, mas em perspectivas distintas.

5.3.2 Correspondência de linguagens

Tanto na fase de correspondência de linguagens, como na fase de correspondência de elementos do modelo, há a presença do MuMle (AMIR, S. et al., 2011), com a incumbência de realizar o cálculo das similaridades entre os elementos presentes. Sendo que na fase de correspondência de linguagens a comparação é feita no nível de linguagem de definição de esquemas, permitindo a compatibilidade do sistema com várias destas linguagens. Enquanto na fase de correspondência de elementos do modelo, as similaridades são buscadas no nível do modelo de dados. A escolha do MuMle foi motivada pela sua estratégia de cálculo dos coeficientes de correlação, que leva em consideração vários tipos de informações (linguística, semântica e estrutural), de forma a possibilitar um maior acerto das correspondências.

A utilização do MuMle tem o objetivo de tornar o processo de interoperabilidade semi-automático (“semi” porque está prevista a participação do usuário no fim do processo). Contudo, se consideradas as definições de interoperabilidade de Haslofer e Klas (HASLHOFER e KLAS, 2010), onde os autores definem interoperabilidade como a *“habilidade de intercâmbio de metadados entre dois ou mais sistemas sem ou com a mínima perda de informação e sem qualquer ajuste especial em ambos sistemas”*, pode-se dizer que foi alcançada a automatização do processo em caso de um alto grau de acerto de correspondências entre dois modelos, visto que a participação do usuário no mapeamento de elementos seria nula ou mínima.

No caso específico desta etapa de obtenção da correspondência de linguagens, se os esquemas de metadados fornecidos forem construídos usando a mesma linguagem, não será necessário realizar nenhum mapeamento nesse nível. É então considerada a linguagem que foi utilizada por ambos e inicia-se o processamento dos modelos, realizando a leitura dos elementos e com base nisso a construção da representação de grafo (vide seção 5.3.1).

O Dynamo Framework pode armazenar na sua base de conhecimento, ligada ao componente interno Extender, os mapeamentos pré-definidos entre várias linguagens de definição de esquemas. Haslofer e Klas (HASLHOFER e KLAS, 2010) destacam trabalhos que propõem soluções de mapeamento para várias linguagens, como UML, RDFs, OWL, XMLSchema e SQL-DDL (vide Tabela 9). Fazendo uso desse recurso, consegue-se atingir o mapeamento automático no nível de linguagem, evitando que seja necessária a participação do usuário para confirmar a relação entre elementos das linguagens, o que reduz a complexidade do processo para o usuário e o mantém com o foco no seu interesse principal, o mapeamento dos modelos de dados.

Tabela 9. Seleção de trabalhos que abordam o mapeamento no nível de linguagem (M2)

	UML (Orientado a objeto)	RDF/S, OWL (Ligações semânticas)	XML Schema (Hierárquico)	SQL-DDL (Relacional)
UML (Orientado a objeto)		Gasevic et al. 2004	OMG 2007a	Ambler 2003
RDF/S, OWL (Ligações semânticas)	OMG 2006b		-	Motik et al. 2007
XML Schema (Hierárquico)	Bernauer et al. 2004	Lethi e Frankhauser 2004		Atay et al. 2007
SQL-DDL (Relacional)	Fong 1997	Bizer e Seaborne 2004	Lee et al. 2003	

5.3.3 Correspondência de elementos dos modelos

Na etapa de correspondência de elementos dos modelos são comparados os elementos definidos nos esquemas de metadados. Baseado na etapa anterior, que encontrou uma linguagem comum para a representação dos esquemas fornecidos, é realizada a análise de cada elemento e suas respectivas propriedades, construindo a estrutura de grafo que será enviada ao sistema de correspondência de metadados, no caso o MuMle.

O grafo é formado por nós normais e nós de propriedade. Os nós de propriedade têm o seu nome formado pela concatenação do nome do nó predecessor (nó de entidade) e o seu próprio nome. Na etapa atual são adicionados tokens a cada nó, e a cada token é atribuído um conjunto de sinônimos (*synset*). Os tokens são os elementos mais relevantes extraídos do nome do nó, dado algum conjunto de separadores. Por exemplo, em um nó cujo nome é “role_description”, seus respectivos *tokens* serão “role” e “description”. Para cada token é atribuído um conjunto de sinônimos, obtidos a partir de uma consulta pelo nome do token ao serviço de correspondência de metadados. Mais detalhes sobre a estrutura dos nós normais e nós de propriedade podem ser vistos na Seção 4.2.1.

A Figura 30 mostra o grafo $G(A)$, que ilustra o exemplo de uma estrutura de grafo levando em consideração um modelo de dados simples contendo apenas três entidades: “Usuário”, “Endereço” e “Papel”. A entidade “Usuário” possui o atributo “login” e uma referência para a entidade “Endereço” e uma referência para a entidade “Papel”. A entidade “Endereço” possui os atributos “rua”, “bairro” e “número”. A entidade “Papel” possui os atributos “id” e “descrição”. Naturalmente as entidades são representadas no grafo como nós de entidade, enquanto as propriedades são representadas como nós de propriedade.

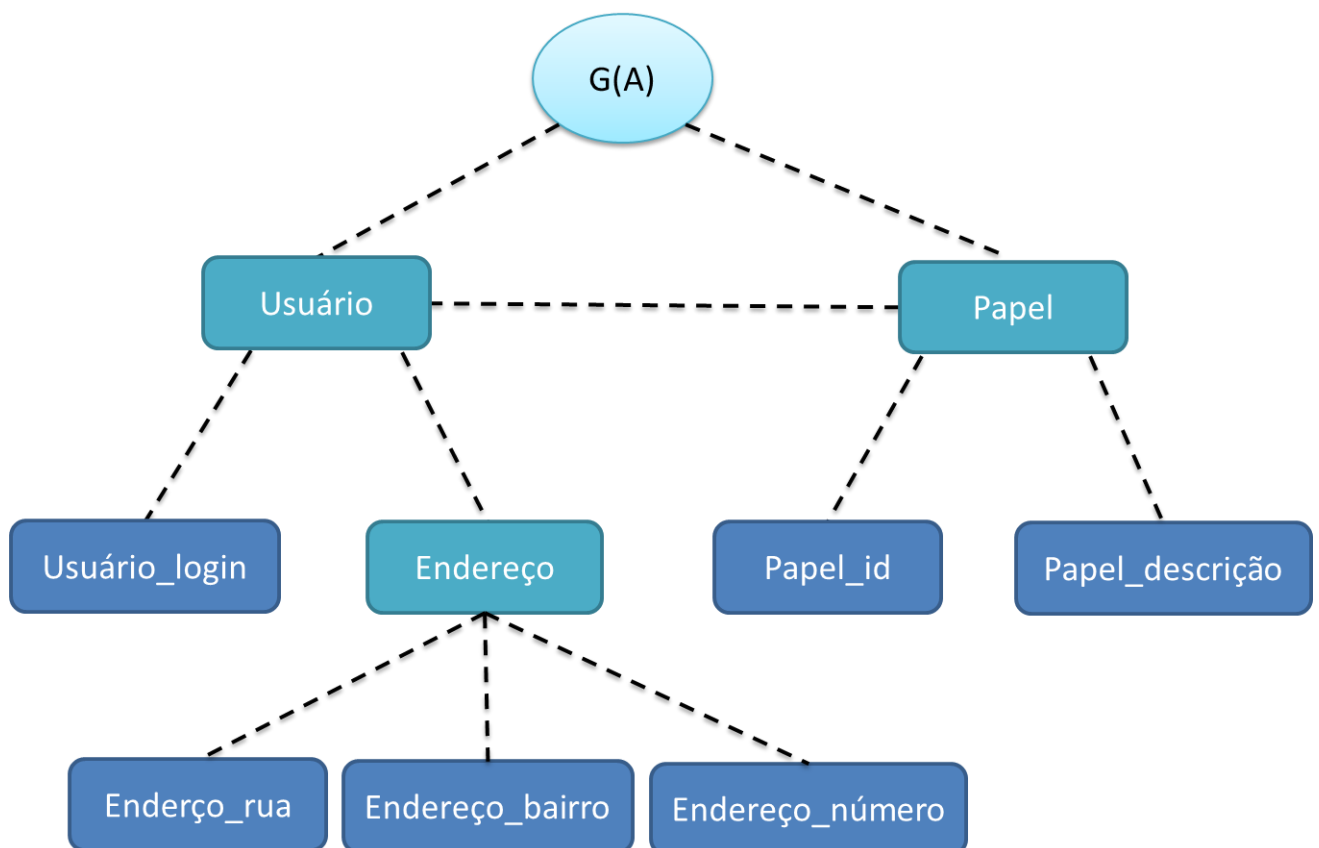


Figura 30. Exemplo de um grafo para representar um determinado esquema de metadados

Ao receber o grafo contendo as informações dos elementos do esquema de metadados, o MuMle fica responsável por comparar cada nó de propriedade e obter seus respectivos sinônimos, junto à base lexical Wordnet.

Após a obtenção dos sinônimos, o resultado é uma estrutura no formato da que é exemplificada na Figura 31. No caso, foram encontrados três sinônimos relevantes para o token “description”: “label”, “name” e “detailing”, enquanto que para o token “role” foram encontrados os sinônimos “function” e “responsibility”.

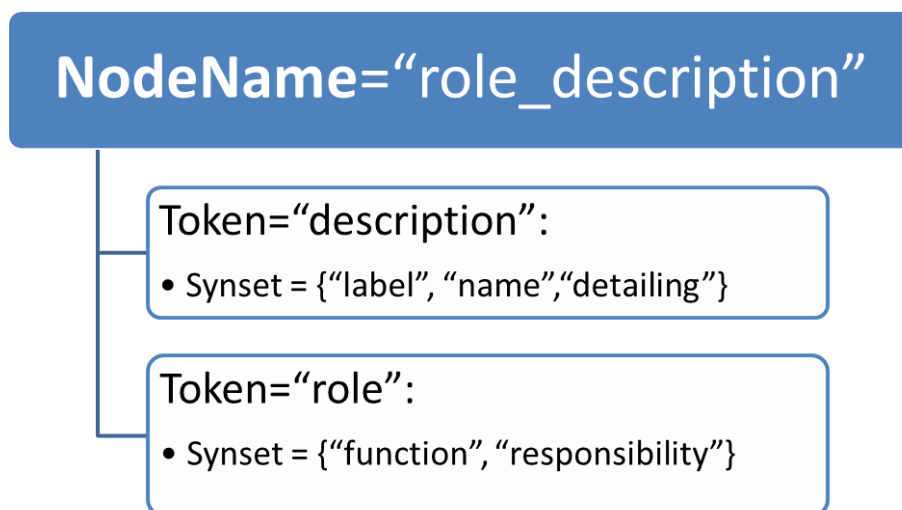


Figura 31. Estrutura detalhada de um nó de propriedade

Como o sistema MuMle utiliza como fonte de consulta de sinônimos o Wordnet, que só interpreta termos em inglês, o Dynamo Framework restringe o suporte inicialmente à modelos de dados cujos elementos foram descritos na língua inglesa, o que ocorre na maioria dos modelos de metadados padronizados (e.g. Dublin Core, PBCore, TV-Anytime, MPEG-7 , etc). Entretanto, a arquitetura de componentes do Dynamo Framework possibilita que sejam adicionados posteriormente outros sistemas de correspondências que interpretem termos em outros idiomas.

Após a definição dos tokens e obtenção dos seus respectivos sinônimos, é realizada a comparação entre os elementos dos dois grafos gerados a partir do par de esquemas de metadados fornecido como entrada para a primeira etapa. A comparação é feita considerando a combinação entre cada um dos tokens, onde será extraído o índice máximo de correlação obtido entre os tokens das propriedades comparadas. Como resultado desse procedimento, a estrutura de grafos utilizada pelo Dynamo Framework irá receber as informações dos nós relacionados e do valor do coeficiente de correspondência de cada relação, construindo uma representação como a apresentada na Figura 32.

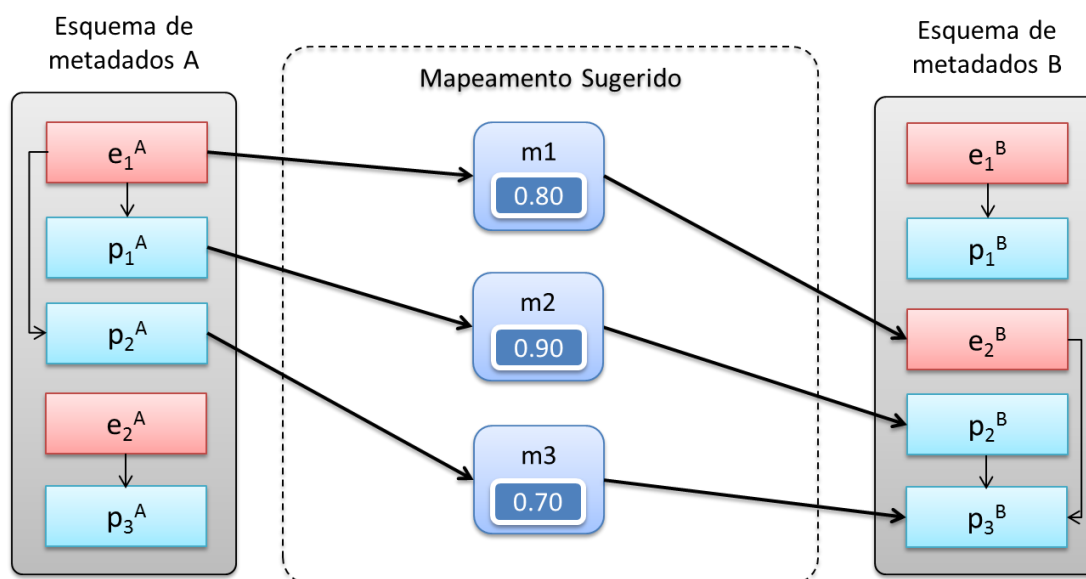


Figura 32. Representação dos mapeamentos entre elementos e propriedades dos esquemas de metadados

A Figura 32 descreve um exemplo de mapeamento entre elementos e propriedades entre os dois esquemas de metadados comparados. Cada mapeamento possui um coeficiente de correspondência, valor numérico entre 0 e 1 obtido após a comparação semântica e estrutural dos elementos dos modelos. As fórmulas que detalham como o MuMle realiza essas comparações e obtém esse valor numérico foram apresentadas na Seção 4.2.1. Caso o índice máximo obtido seja relativamente baixo, isso implicará que as propriedades dos sistemas possuem pouca compatibilidade e demandará uma maior atenção por parte do usuário no momento de definir que os elementos são correlatos.

De posse dos mapeamentos sugeridos, contendo a informação dos elementos correspondentes e seus respectivos coeficientes, são completados os dados apresentados na estrutura de classes vista na primeira etapa (subseção 5.3.1).

5.3.4 Confirmação dos mapeamentos

Com a informação de todos os mapeamentos sugeridos entre os elementos dos modelos representados na forma de objetos. Ocorre o envio dessa representação ao sistema cliente, que ficará responsável por interpretá-la e apresentar ao usuário a lista contendo dos mapeamentos sugeridos. O usuário participa do processo atribuindo a cada um dos mapeamentos uma das seguintes relações: equivalência (confirmação que as propriedades são correspondentes), inclusão (uma propriedade corresponde à concatenação de outras), exclusão (não há correspondência e a propriedade mapeada não será considerada) e sobreposição (não há correspondência, mas a propriedade é

importante e precisa ser agregada ao modelo genérico). A seguir cada uma dessas modalidades é exemplificada.

5.3.4.1 Relação de equivalência

A relação de equivalência ocorre quando a interpretação do usuário indica que os elementos comparados, mesmo que possuam uma nomenclatura diferente, contêm o mesmo significado semântico, confirmando a equivalência entre ambos. A propriedade “author” (autor) num esquema de metadados não-padronizado quando comparada à propriedade “creator” (criador) do esquema de metadados Dublin Core é um exemplo desse tipo de relação. A Figura 34 ilustra o exemplo citado.

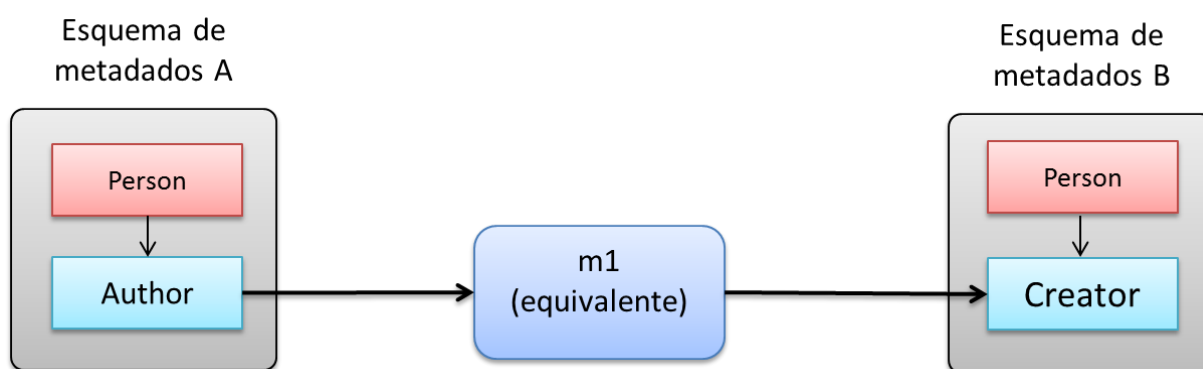


Figura 33. Exemplo da relação de equivalência entre elementos de dois modelos

5.3.4.2 Relação de inclusão

A relação de inclusão acontece quando a interpretação de um elemento contém uma interpretação de outro elemento. No contexto do exemplo visto na Figura 34, a interpretação da propriedade “creator” do Dublin Core inclui a interpretação dos elementos “GivenName” e “FamilyName” do TV-Anytime, porque esses elementos descrevem uma pessoa com o papel de autor.

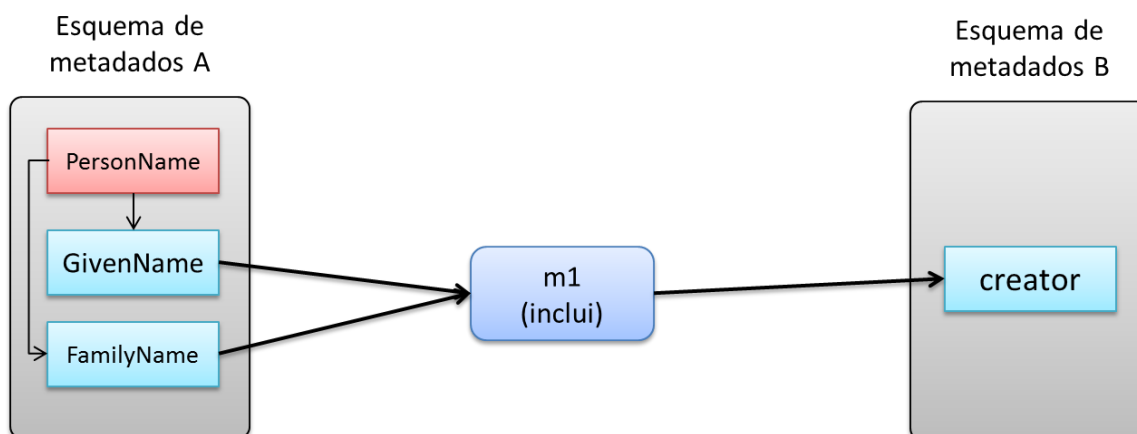


Figura 34. Exemplo da relação de inclusão entre elementos de dois modelos

5.3.4.3 Relação de exclusão

A relação de exclusão ocorre quando os elementos que foram indicados no mapeamento possuem significados distintos. Por exemplo, a interpretação da propriedade “rights” num esquema baseado no Dublin Core e da propriedade “birthday” num esquema não padronizado remete à exclusão desse mapeamento sugerido entre ambos, conforme ilustra a Figura 35.

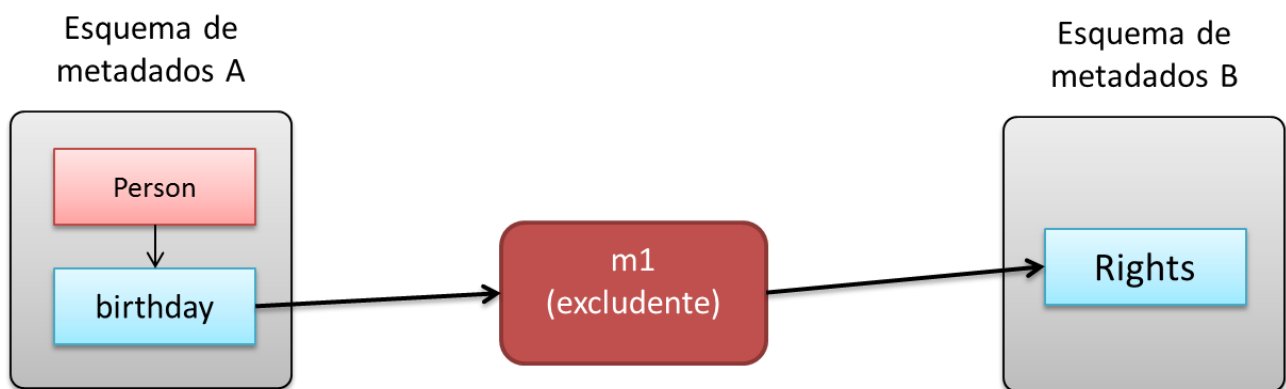


Figura 35. Exemplo da relação de exclusão entre elementos de dois modelos

5.4.3.4 Relação de sobreposição

A relação de sobreposição acontece quando dois elementos são correlacionados mas não é possível excluir um dos dois para a representação no modelo genérico. Por exemplo, os elementos “description” (descrição), “synopsis” (sinopse) e “info” (informações) são exemplos de elementos com interpretações de sobreposição. A propriedade “description” normalmente provê informações similares às fornecidas pela propriedade “synopsis”, o mesmo ocorre quando se compara a propriedade “synopsis” com “info”. Logo, no modelo genérico seriam admitidas as três propriedades, conforme mostra a Figura 36.

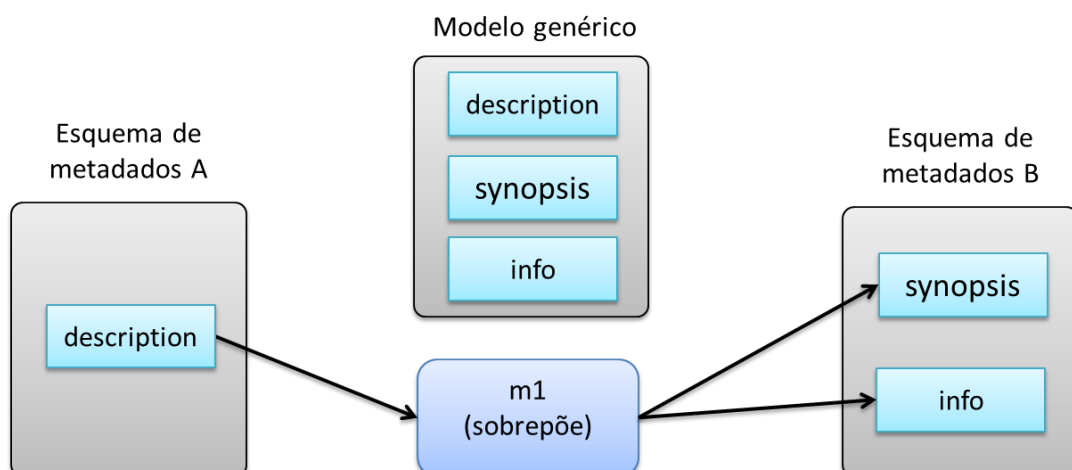


Figura 36. Exemplo da relação de exclusão entre elementos de dois modelos

Após a finalização dos ajustes por parte do usuário, o Dynamo Framework passa para etapa de construção do meta-modelo genérico, que irá construir um modelo único formado pelas informações mapeadas dos dois esquemas de metadados.

5.3.5 Construção do meta-modelo

A construção do meta-modelo inicia-se a partir da confirmação, pelo usuário, das propriedades correlacionadas entre os esquemas que estão sendo mapeados. Nesta etapa o meta-modelo será armazenado na base de modelos integrados. Possibilitando assim, um mecanismo de acesso comum.

5.3.5.2 Esquema de representação do meta-modelo

Após a confirmação de todas as correspondências entre os modelos que estão sendo integrados, obtêm-se como resultado o meta-modelo gerado inicialmente em formato RDFs.

Ressalta-se que o RDFs é um formato flexível, de modo que já existem adaptadores capazes de converter a representação em RDFs para XSchema, OWL, SQL-DD ou outros esquemas. Algumas dessas ferramentas de adaptação são citadas por Haslhofer e Kles (Haslhofer e Kles, 2010).

O motivo para uso do RDFS está na necessidade de representar conhecimento adicional sobre o recurso (ou seja, no uso de metadados mais expressivos). O RDFS torna possível verificar se as propriedades aplicadas a um recurso fazem sentido e se os valores utilizados em conjunto com as propriedades são possíveis. Pode-se verificar a integridade dos dados, a validade de declarações, criar uma hierarquia de tipo definindo-se classes e subclasses, utilizar recursos de herança (comuns nas linguagens orientadas a objeto).

Um esquema pode ser especificado na sintaxe RDF, e partir daí, transportado e armazenado em outra forma e em outra sintaxe, pois o que importa nesse caso é o modelo que suporta a sintaxe e não a sintaxe em si. Os meta-modelos deste trabalho são construídos em RDFSchema, não significando que essa sintaxe seja obrigatória.

O RDF permite a apresentação textual e a apresentação gráfica, que mostra mais claramente os recursos e propriedades do meta-modelo, bem como suas inter-relações. As marcações RDFSchema presentes baseiam-se em Nebel (1998), que possibilita algumas extensões.

Utilizando-se das considerações sobre a definição do meta-modelo genérico feitas na seção anterior, representa-se o modelo em RDFSchema⁵, gerado a partir da linguagem OWL. O meta-modelo exemplificado nesta seção foi gerado para o mapeamento de um esquema baseado no padrão de metadados PBCore para um outro baseado no padrão de metadados Dublin Core. Para a representação gráfica do meta-modelo (Figura 37) utiliza-se a ferramenta *Neon Toolkit*. O meta-modelo é descrito em seguida. A sintaxe completa de um meta-modelo em RDFSchema é apresentada no Anexo B.

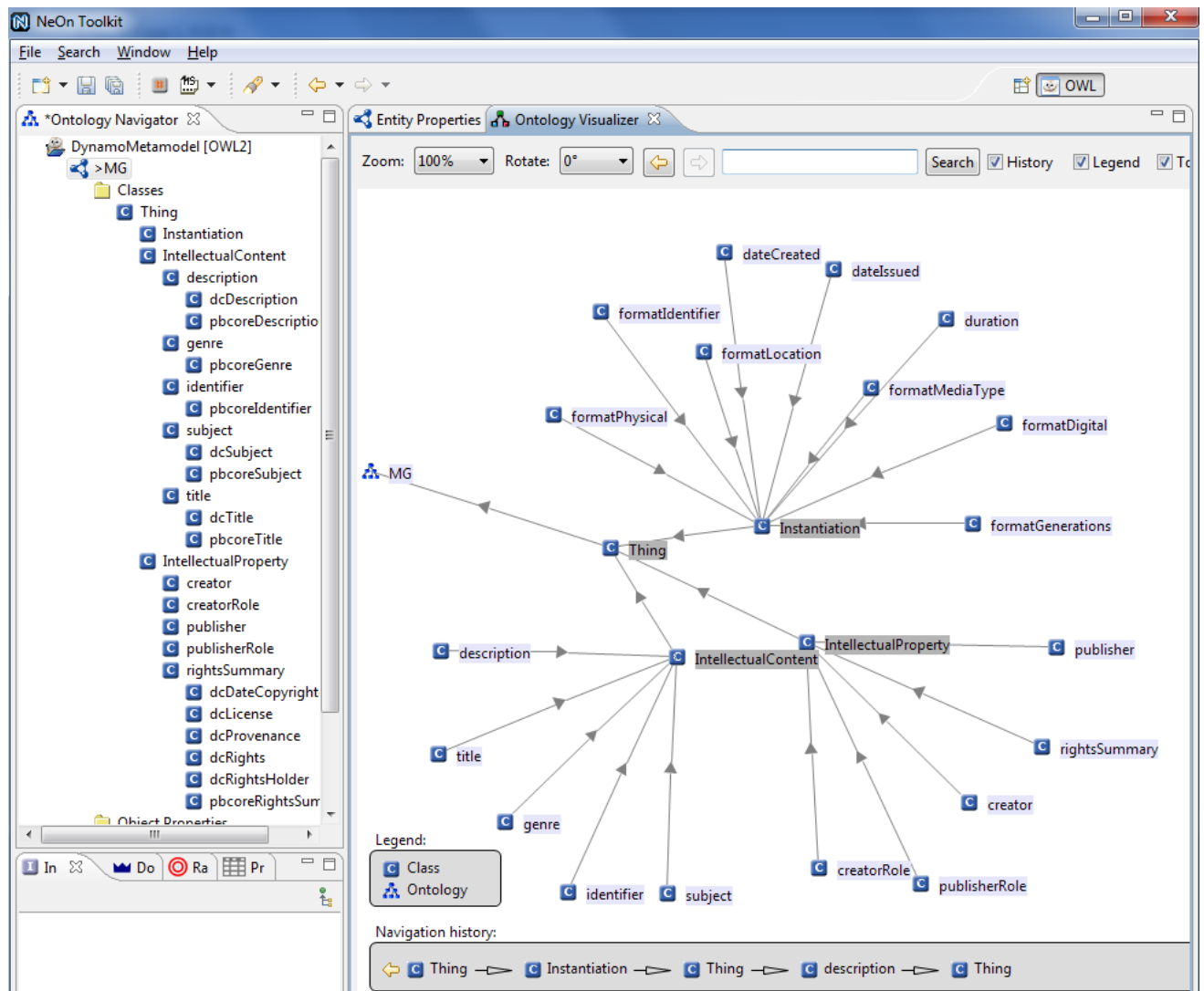


Figura 37. Tela do editor de ontologias Neon Toolkit, mostrando as classes para a ontologia construída a partir do meta-modelo

⁵ Alguns conceitos básicos do RDF/RDFSchema são apresentados no Anexo A

5.3.5.3 Definição das classes

A Tabela 10 descreve as classes principais presentes no meta-modelo gerado para o exemplo em questão.

Tabela 10. Descrição das classes do meta-modelo

Classe	Subclasses	Descrição
MG	IntellectualContent, IntellectualProperty, Instantiation	Classe que contém o modelo genérico
IntellectualContent	description, genre, identifier, subject, title	Propriedades básicas do conteúdo de mídia
IntellectualProperty	creator, creatorRole, publisher, publisherRole, rightsSummary	Propriedades relacionadas à restrições de direitos autorais e propriedade intelectual
Instantiation	dateCreated, dateIssued, duration, formatDigital, formatGenerations, formatIdentifier, formatLocation, formatMediaType, formatPhysical	Informações técnicas do conteúdo de mídia

5.3.5.4 Definições de namespaces:

1) Meta-modelo genérico:

<http://ci.ufpb.br/Dynamo/metamodel/MG#>

2) Classe IntellectualContent:

<http://ci.ufpb.br/Dynamo/metamodel/MG#IntellectualContent>

3) Classe IntellectualProperty:

<http://ci.ufpb.br/Dynamo/metamodel/MG#IntellectualProperty>

4) Classe Instantiation:

Namespace 4: < http://ci.ufpb.br/Dynamo/metamodel/MG#Instantiation

5.3.5.5 Hierarquia de classes

A Figura 38 apresenta um fragmento das hierarquias definidas no meta-modelo. As classes que estão ligadas diretamente às classes principais (IntellectualContent, IntellectualProperty e Instantiation) representam as propriedades do modelo genérico. Essas propriedades que representam os atributos do modelo genérico possuem uma ou mais subclasses, que indicam qual mapeamento entre elementos dos esquemas deu origem à essa propriedade no modelo genérico. Por exemplo, na Figura 38 é possível observar a classe “title” ligada à “IntellectualContent”, como subclasses de “title” estão presentes “dcTitle” (title no esquema Dublin Core) e “pbCoreTitle” (title no esquema

PBCore), indicando os elementos que deram origem à classe genérica “title”. Outro exemplo é a classe “rightsSummary”, subclasse de “IntellectualProperty”. Nessa classe, há referência para cinco atributos do Dublin Core: “dateCopyrighted”, “license”, “provenance”, “rights” e “rightsHolder” e só um do PBCore: “rightsSummary”, visto que o primeiro divide a informação em várias propriedades enquanto o segundo agrupa todas em uma só.

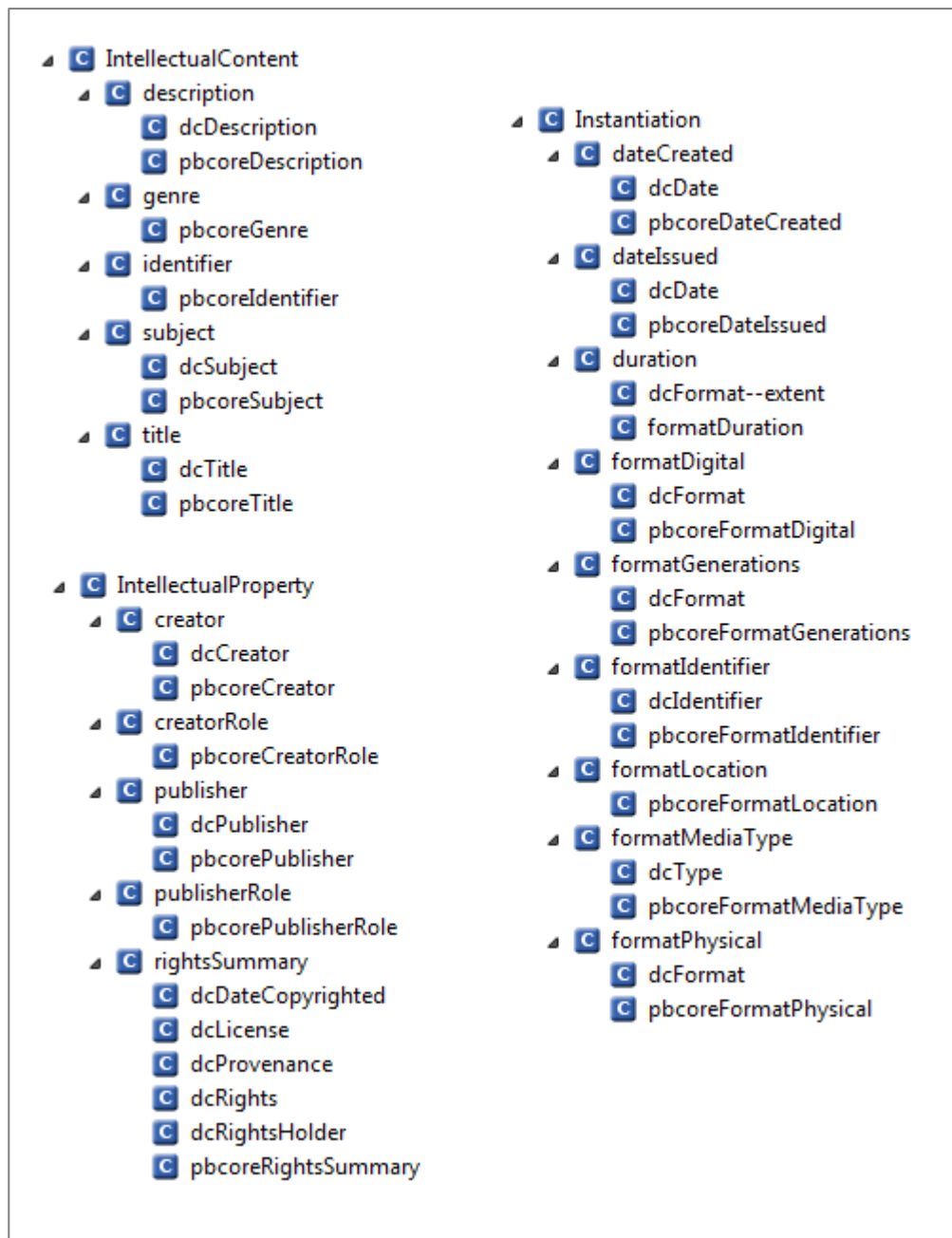


Figura 38. Representação das hierarquias definidas entre as classes

5.3.6 Operações sobre os meta-modelos

As operações sobre os meta-modelos são realizadas com base em consultas sobre o meta-modelo, escritas na linguagem SPARQL (W3C, 2008). Essas consultas devem

sempre retornar um conjunto de recursos RDF. A seguir serão abordadas as opções de sintaxe das expressões de consultas utilizadas pelo Dynamo Framework.

5.3.6.1 Expressões de consulta em SPARQL

O uso de expressões de consultas na linguagem SPARQL deve seguir a sintaxe aceita pela máquina de consultas ARQ⁶, parte do framework Jena, que é uma das implementações de maior conformidade com a especificação⁷ da linguagem SPARQL definida pelo W3C e é utilizado pelo Dynamo Framework. A máquina de consultas ARQ também implementa algumas das novas funcionalidades da versão 1.1 da linguagem SPARQL, em particular, a consulta em bases de dados SPARQL federadas⁸. A Figura 40 ilustra um exemplo de uma expressão de consulta na linguagem SPARQL.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX mg: <http://ci.ufpb.br/Dynamo/metamodel/MG#>

SELECT DISTINCT ?s WHERE { ?s rdf:type mg:IntellectualContent. }
```

Figura 39. Exemplo de uma expressão de consulta SPARQL

Esta expressão de consulta deve retornar todos os recursos do tipo `mg:IntellectualContent`.

O trecho `SELECT DISTINCT ?s` no início da expressão garante que o resultado será um conjunto de recursos RDF e é um padrão comum entre as expressões de consultas de contexto na linguagem SPARQL.

Quando é necessário combinar a expressão de consulta com o valor de algum parâmetro, deve-se interpolar a cadeia de caracteres da expressão com o valor do parâmetro.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX mg: <http://ci.ufpb.br/Dynamo/metamodel/MG#>
PREFIX content:
<http://ci.ufpb.br/Dynamo/metamodel/MG#IntellectualContent>

SELECT DISTINCT ?s WHERE { ?s rdf:type mg:IntellectualContent.
                           ?s content:creator . }
```

Figura 40. Exemplo de uma expressão de consulta SPARQL com parâmetros

⁶ <http://jena.sourceforge.net/ARQ>

⁷ <http://www.w3.org/TR/rdf-sparql-query>

⁸ <http://www.w3.org/TR/sparql11-query>

A Figura 40 ilustra um exemplo de expressão de consulta em SPARQL à propriedade de nome “creator”.

Como o Dynamo Framework utiliza o Jena Framework agregado à sua arquitetura de componentes, consegue-se fornecer aos sistemas clientes uma camada de abstração para as consultas SPARQL. A Figura 41 mostra um exemplo do processo de conversão de consultas, considerando os modelos de duas instituições distintas (BBC e TV Brasil), utilizando o meta-modelo construído e gerenciado pelo Dynamo Framework.

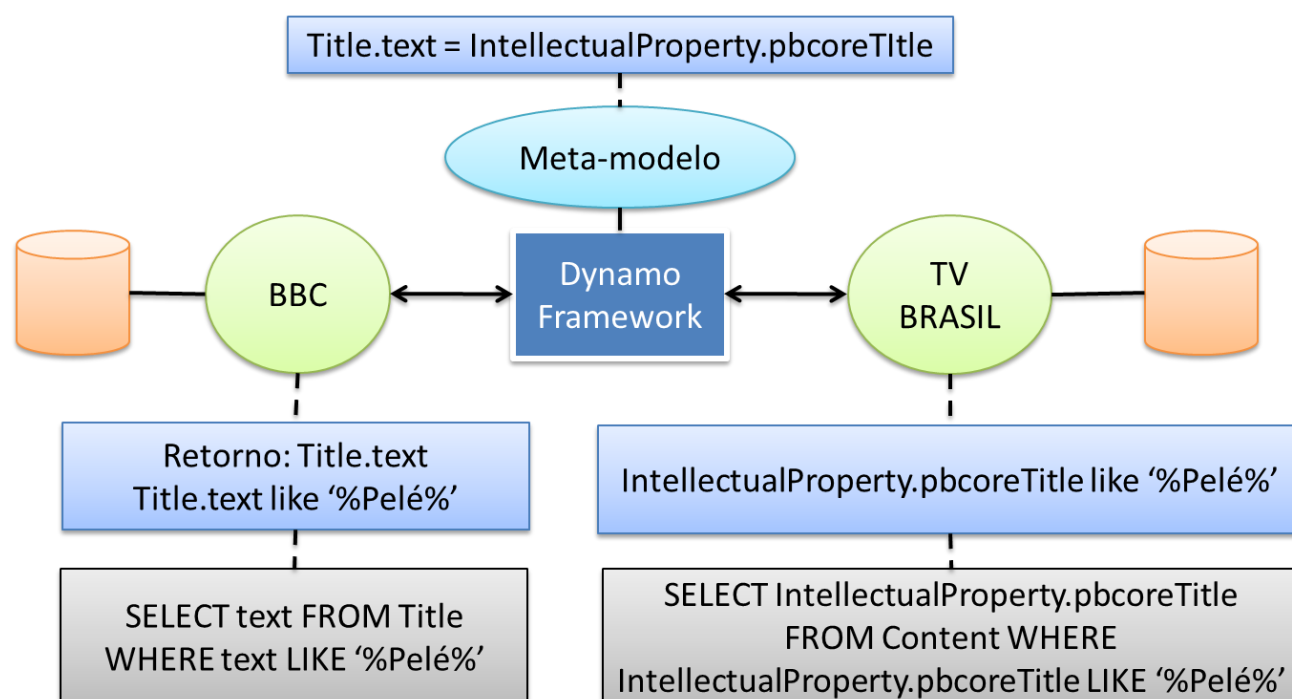


Figura 41. Exemplo de conversão de consultas entre dois modelos distintos (BBC e TV Brasil) levando em consideração o meta-modelo gerado pelo Dynamo Framework

A interação dos sistemas clientes sobre o meta-modelo acontecerá via API Java disponibilizada pelo Dynamo Framework, através do acesso aos métodos descritos na Tabela 11.

Tabela 11. Métodos do Dynamo Framework que abstraem a manipulação das consultas sobre o meta-modelo RDF

Método	Classe	Entrada	Saída
retrievePropertyValue	Façade	Identificação do meta-modelo, nome do elemento que contém a propriedade e nome da propriedade	Valor da da propriedade
retrieveProperties	Façade	Identificação do meta-modelo, nome do elemento	Lista com os nomes das propriedades

addMetamodelMapping	Façade	Identificação do par de esquemas, conjunto contendo os mapeamentos	Construção do meta-modelo RDF.
updateMetamodel	Façade	Identificação do meta-modelo, conjunto contendo os mapeamentos	Atualização do meta-modelo RDF.
removeMetamodel	Façade	Identificação do meta-modelo	Remoção do meta-modelo RDF.

Desta forma o Dynamo Framework consegue simplificar a realização de consultas sobre o meta-modelo, evitando que os sistemas clientes necessitem construir consultas SPARQL e interpretar o resultado obtido em RDFs, no caso, o resultado é representado numa estrutura de objetos, conforme foi descrito na Seção 5.1.3.

5.3.6.2 Busca e descoberta

Como já foi citado, o domínio de aplicação que o Dynamo Framework se adequa inicialmente é o de sistemas de gerenciamento de ativos digitais, cujos elementos principais são os próprios ativos digitais. Para realizar o processo de busca sobre as bases integradas é necessário definir um identificador único para cada ativo inserido. Existem diversas soluções para definição de identificadores, das quais a DOI (Digital Object Identifier) se mostrou mais adequada para a representação de objetos neste trabalho. Em um framework DOI, é possível persistir a identificação, gerenciar propriedades intelectuais do conteúdo, gerenciar metadados, facilitar o comércio eletrônico e automatizar o gerenciamento de conteúdo (DOI, 2010).

Um DOI pode apontar para documentos, imagens, áudio, vídeo, contratos, ou seja, virtualmente qualquer coisa que uma URL pode apontar, o sistema de identificação DOI pode gerenciar (DOI, 2010).

Um identificador DOI consiste de duas partes, um prefixo designado por um Gerente de Diretório (Directory Manager), e um sufixo, que identifica um item em particular. Podem existir vários Gerentes de Diretórios, eventualmente um em cada país ou para cada nicho da indústria (indústria literária, fotos, música, software, etc.). Os primeiros dois caracteres do prefixo designam o Gerente de Diretório, por exemplo, em 10.1001 o Gerente de Diretório é 10. A segunda parte do prefixo, neste caso 1001, define a organização que adquiriu o prefixo DOI. O sufixo é seguido por uma "/" e logo depois, temos o prefixo, uma sequência única de caracteres que é emitida pela organização, representando um objeto digital específico.

O DOI também pode ser utilizado como uma ferramenta adicional para detectar similaridades entre modelos de metadados distintos. No caso, quando dois conteúdos de mesmo DOI forem encontrados, as similaridades no nível de instância indicariam que dois atributos são correlatos. Por exemplo, um conteúdo cujo título representado por um campo denominado “Title” for exatamente igual a um campo de outro modelo com o mesmo valor mas de nome “Name”, por possuírem valores idênticos, será assumida uma sugestão de mapeamento entre os atributos “Title” e “Name”. A implementação deste recurso não foi adotada em um primeiro momento como critério para a detecção de correlações pelo Dynamo Framework, mas a arquitetura do framework permite que em trabalhos futuros sejam implementados sistemas de correspondência que levem também em consideração o nível de instância de metadados, fazendo a identificação de similaridades de atributos em conteúdos através do DOI.

5.3.6.3 Intercâmbio de dados

O compartilhamento de ativos digitais entre os domínios segue o conceito similar ao modelo de interconexão proposto para a Internet (STALLINGS, 2002). Na perspectiva de integração do Dynamo Framework é possível agrupar grupo de bases de dados em redes, logo, pode-se utilizar o processo de integração definido pelo framework para possibilitar a comunicação inter-redes.

Ao agrupar várias bases no formato de rede é necessário especificar qual será a base de dado padrão (default) para comunicação, pois quando outra rede requisitar uma pesquisa ou troca de algum conteúdo, a comunicação deve ser feita a partir da base padrão.

Para o tráfego de conteúdo em redes distintas também é importante a identificação única dos conteúdos armazenados. Para isso, cada rede da inter-rede recebe um prefixo de identificação único.

Como não é foco deste trabalho, não foi definido um mecanismo para utilização de políticas para o intercâmbio de dados entre redes distintas, porém deixa-se em aberto a possibilidade de trabalhos futuros agregarem as políticas ao meta-modelo genérico.

6

Avaliação Experimental

Neste capítulo será realizada uma avaliação experimental da arquitetura proposta. O capítulo divide-se em duas seções principais. Na seção 6.1 é apresentada a prova de conceito construída a fim de validar o Dynamo Framework. Na Seção 6.2 é detalhada a execução do experimento de validação, contemplando as etapas do Dynamo Framework para alcançar a interoperabilidade em três bases de dados de sistemas de gerenciamento de ativos digitais com modelos de metadados distintos.

6.1 Prova de conceito

A título de validação do Dynamo Framework proposto no Capítulo 5, foi implementado um sistema denominado M-MAP, que fará o papel de “Sistema Cliente”, ou seja, um exemplo de sistema que agrega o Dynamo Framework à sua estrutura com a função de apoiar a realização das principais etapas do processo de interoperabilidade entre bases de dados distintas. A seguir é realizado o detalhamento técnico do M-MAP, apresentando os componentes da interface do usuário e como ocorre a comunicação com o Dynamo Framework.

O M-MAP é composto de um módulo de interface de administração, composto de uma interface Web por onde o usuário interage com o sistema e um módulo de controle, que recebe as requisições do usuário e as processa, com o auxílio do Dynamo Framework. O software possui uma organização modular, de forma a possibilitar o acesso aos módulos do Dynamo Framework, através de uma composição de um ou mais componentes da camada de controle.

6.1.1 Módulo de Interface de administração

O ambiente do M-MAP dispõe de uma interface gráfica de formulários que é acessada por meio de um navegador web e permite a comunicação do usuário com o M-MAP e consequentemente com o Dynamo Framework, através da execução de operações necessárias para a o processo de interoperabilidade entre bases de dados distintas.

A interface irá prover, durante a etapa de confirmação dos mapeamentos, um espaço para o usuário confirmar a relação dos elementos existentes nos modelos de metadados que serão integrados. Para auxiliar essa tarefa, a interface receberá do Dynamo Framework as potenciais relações e os respectivos coeficientes de correspondência.

Após a realização de todas as etapas do processo de interoperabilidade, a interface será a responsável por fornecer visões de gerenciamento, pesquisa e intercâmbio dos ativos digitais que estão contidos em bases diferentes.

A interface gráfica do ambiente M-MAP, mostrada na Figura 42, foi organizada em dois níveis de navegação. No primeiro nível é exibido um menu horizontal de itens relacionados a todo o processo. Sendo assim, temos a opção "Novo Mapeamento", que leva às etapas do processo de integração com um novo domínio e "Domínios Mapeados", que possibilita a visualização das informações dos domínios que foram integrados anteriormente. O item "Início" leva à tela inicial do ambiente e não está relacionada a nenhuma etapa do processo, da mesma forma que o item "Sair", que leva o usuário à tela de autenticação.

Ao clicar em um item do menu horizontal de primeiro nível, um menu vertical de segundo nível será exibido, apresentando itens que representam as etapas relacionadas à opção selecionada. Por exemplo, ao clicar no item de primeiro nível "Novo Mapeamento", será exibido o menu de segundo nível apresentando os itens "1) Definição dos modelos", "2) Processamento dos esquemas", "3) Confirmação das similaridades", "4) Armazenamento", todos relacionados às etapas para a realização da integração entre dois domínios. A Figura 42 apresenta a tela do M-MAP para a configuração de um novo mapeamento, na qual é possível observar os itens do menu horizontal de primeiro nível e os itens do menu vertical de segundo nível.



Figura 42. Tela da interface de administração do M-MAP

6.1.2 Módulo de Controle

Por possuir uma interface Web, o ambiente de administração do M-MAP pode ser utilizado remotamente, bastando que se acesse a URL do serviço em um navegador Web. As URLs de acesso ao ambiente do M-MAP seguem o padrão de URLs do *framework* PrettyFaces (OCPSOFT, 2009), apresentado na Figura 43.

```
http://[endereço_do_servidor]/[controlador_de_retaguarda]
```

Figura 43. Padrão de URLs para acessar o M-MAP

O M-MAP possui vários controladores de retaguarda para o ambiente de configuração do processo de interoperabilidade, cuja sequência de execução foi descrita de forma detalhada na Seção 5.3.

Tabela 12 apresenta os endereços dos controladores de retaguarda do M-MAP, organizados por etapa do processo de interoperabilidade.

Tabela 12. Controladores de Retaguarda do M-MAP

Etapa	Controladores
Fornecer informações dos domínios que serão integrados	domainIntegrationForm, addDomainsAction
Invocar o processo de análise dos modelos e cálculo dos coeficientes de correlação	matchSchemasAction
Confirmar similaridades dos modelos	consolidateSimilaritiesAction
Visualizar informações de um domínio integrado	domainInformationView
Listar propriedades do meta-modelo obtido da integração de dois domínios	metamodelPropertiesListView
Alterar propriedades de um meta-modelo existente	metamodelPropertiesForm, saveMetamodelPropertiesAction
Remover um meta-modelo existente	metamodelRemoveConfirmForm, metamodelRemoveAction

6.2 Experimento de validação

Utilizando o sistema desenvolvido como uma prova de conceito para validar a aplicação do Dynamo Framework no processo de interoperabilidade de metadados, realizaram-se experimentos considerando o cenário de uma inter-rede contendo três conjuntos de sistemas de gerenciamento de ativos digitais que utilizam modelos de metadados distintos. A apresentação detalhada dos experimentos é realizada a seguir, dividida em três subções: plano do experimento, execução do experimento e análise do experimento.

6.2.1 Plano do Experimento

O ambiente de validação toma como base um experimento contemplando a integração de três redes de sistemas de gerenciamento de ativos digitais de instituições distintas. A seguir é feita uma breve descrição dos sistemas, destacando o modelo de metadados utilizado por cada um.

6.2.1.1 Instituição 1: Serviço de intercâmbio de conteúdos da TV Brasil

O primeiro caso considerado é o sistema de intercâmbio de conteúdos digitais da TV Brasil. O sistema utiliza uma arquitetura descentralizada na qual cada emissora de TV associada possui o seu próprio sistema de gerenciamento de ativos digitais independente, mas forma uma rede com a possibilidade de efetuar buscas e compartilhamento conteúdos entre os parceiros. Cada associada é um “nó” e os nós conectados formam um “domínio”.

Dentro de um domínio, a comunicação entre os nós ocorre através de um protocolo de roteamento inter-nós (ou intra-domínio), cujo papel é prover rotas entre os nós de origem e destino. A formação do domínio TV Brasil permite, portanto, o intercâmbio dos programas produzidos através da rede, o que representa uma grande economia em relação ao intercâmbio utilizando outros meios (e.g. transmissão via satélite).

Em um cenário considerando mais de um domínio, consideram-se domínios como “vizinhos” quando há uma ou mais ligações entre nós de ambos os domínios. Isso possibilita a integração com domínios externos. Para tal, é utilizado um protocolo de roteamento inter-domínios.

O padrão de metadados adotado para construir modelo de dados do sistema de gerenciamento de ativos digitais da TV Brasil foi o PBCore. A Figura 44 descreve um exemplo do registro de metadados de um conteúdo do sistema.

```
<PBCoreDescriptionDocument "...">
  <pbcoreIdentifier>
    <identifier>TVBRASIL.EBC-DF.1234455</identifier>
  </pbcoreIdentifier>
  <pbcoreTitle>
    <title>Café com a Presidente</title>
  </pbcoreTitle>
  <pbcoreDescription>
    <description>Programa de rádio da presidente.
  </description>
  </pbcoreDescription>
  <pbcoreGenre>
    <genre>Politics; Nature; Environment</genre>
  </pbcoreGenre>
  <pbcoreCreator>
    <creator>SECOM/PR</creator>
    <creatorRole>Producer</creatorRole>
  </pbcoreCreator>
</PBCoreDescriptionDocument>
```

Figura 44. Exemplo do registro de metadados de um conteúdo da TV Brasil

A compatibilidade dos repositórios que fazem parte do domínio TV Brasil (comunicação intra-domínio) é total, visto que usam a mesma infraestrutura técnica e o mesmo padrão de metadados. Entretanto, quando se faz necessária a integração com instituições externas (comunicação inter-domínios), que possuam uma organização diferente do seu modelo de dados, o acesso uniforme se torna impossível, como será mostrado nos exemplos a seguir, dado que seus metadados não são interoperáveis. Para que essa integração seja possível, o M-MAP irá receber como parâmetro de entrada o esquema de metadados de um dos nós que represente o domínio TV Brasil, mais

especificamente o nó que contém o modelo de dados padrão que se aplique a todo o domínio. No âmbito da TV Brasil, quem faz esse papel é o nó denominado “nó mestre”.

6.2.1.2 Instituição 2: Serviço TV-Anytime da BBC

A BBC oferece um serviço de informação da programação on-line para a sua TV e rádio. Via Web Services SOAP (SOAP, 2007), clientes podem requisitar vários detalhes sobre o conteúdo, incluindo informações básicas da programação como o título ou sinopse de um programa. O padrão TV-Anytime foi escolhido pela BBC para representar as informações dos programas. Sua principal ideia é descrever o conteúdo multimídia de emissoras de TV de forma que um usuário ou um agente em nome do usuário possa entender quais conteúdos estão disponíveis para aquisição (MCPARLAND, 2003). Para representar metadados audiovisuais, o padrão TV-Anytime engloba um grande número de elementos definidos pelo padrão MPEG-7 (ISO/IEC, 2007).

A Figura 45 mostra um exemplo da descrição de metadados no padrão TV-Anytime de um ativo digital da BBC.

```
<ProgramInformation programId="crid://bbc.co.uk/123456789">
  <BasicDescription>
    <Title>
      Olympic torch relay heads through
      Royal Wootton Bassett
    </Title>
    <Synopsis>A teenager with one leg was cheered wildly as he
carried the Olympic torch.
    </Synopsis>
    <Genre>
      <Name>Sports</Name>
    </Genre>
    <CreditsList>
      <CreditsItem role="mpe7:cs:RoleCS:2001:AUTHOR">
        <PersonName>
          <mpeg7:GivenName>John</mpeg7:GivenName>
          <mpeg7:FamilyName>Doe</mpeg7:FamilyName>
        </PersonName>
      </CreditsItem>
    </CreditsList>
    <CreationCoordinates>
      <CreationLocation>us</CreationLocation>
    </CreationCoordinates>
  </BasicDescription>
</ProgramInformation>
```

Figura 45. Exemplo do registro de metadados de um conteúdo do sistema da BBC

Como parte da validação do presente trabalho, foi construída uma base de dados contendo a mesma estrutura do serviço da BBC e o seu esquema de metadados foi

importado como parâmetro de entrada do processo de interoperabilidade via a interface do sistema M-MAP.

6.2.1.3 Instituição 3: Banco de conteúdos culturais da Cinemateca

O Banco de Conteúdos Culturais é resultado de uma iniciativa do Ministério da Cultura em conjunto com o Ministério da Ciência e Tecnologia, realizado pela Cinemateca Brasileira em parceria com o Centro Técnico Audiovisual, cujo objetivo é disponibilizar as coleções digitalizadas de fotografias e cartazes de filmes brasileiros do acervo da Cinemateca, além de reportagens de telejornais diários da TV Tupi e registros fílmicos brasileiros.

Devido às especificidades dos conteúdos existentes na base da Cinemateca, os responsáveis pela construção do modelo optaram por não seguir nenhum padrão de metadados existente. A Figura 46 ilustra como é descrito um conteúdo armazenado no Banco de Conteúdos Culturais da Cinemateca.

```
<movieDescription>
  <title>COPA ROCA : PRIMEIRO JOGO</title>
  <code>007761</code>
  <givenTitle>COPA ROCA - PRIMEIRO JOGO -
    ARGENTINA X BRASIL - 8 DE JANEIRO DE 1939
  </givenTitle>
  <productionYear>1939</productionYear>
  <city>Rio de Janeiro</city>
  <state>DF</state>
  <direction>Mauro, Humberto</direction>
  <production>INCE - Instituto Nacional
    de Cinema Educativo</production>
  <photography>Maurio, Humberto</photography>
  <synopsis>Pela Copa Roca, a partida de futebol entre Brasil e
Argentina, no Estádio São Januário. Grande presença de público no
estádio.
    Final: Brasil 0 X 3 Argentina, com gols de Moreno, Garcia e
Massantonio.
  </synopsis>
  <duration>122m</duration>
  <genre>Documentário</genre>
  <subject>Futebol</subject>
  <geographicTerms>Rio de Janeiro</geographicTerms>
  <categories>Sonoro / Não Ficção</categories>
</movieDescription>
```

Figura 46. Exemplo do registro de metadados de um conteúdo da Cinemateca

Como forma de validação do presente trabalho, foi realizada a importação do esquema de metadados da cinemateca, através do envio do arquivo contendo a descrição do conjunto de elementos do esquema, pela interface do M-MAP.

6.2.2 Execução do Experimento

Para realizar o experimento buscando a integração entre duas ou mais bases heterogêneas foram seguidas algumas etapas. Tais etapas são descritas na Figura 47, ilustrando também como o processo ocorreu levando em consideração os sistemas das três instituições citadas anteriormente.

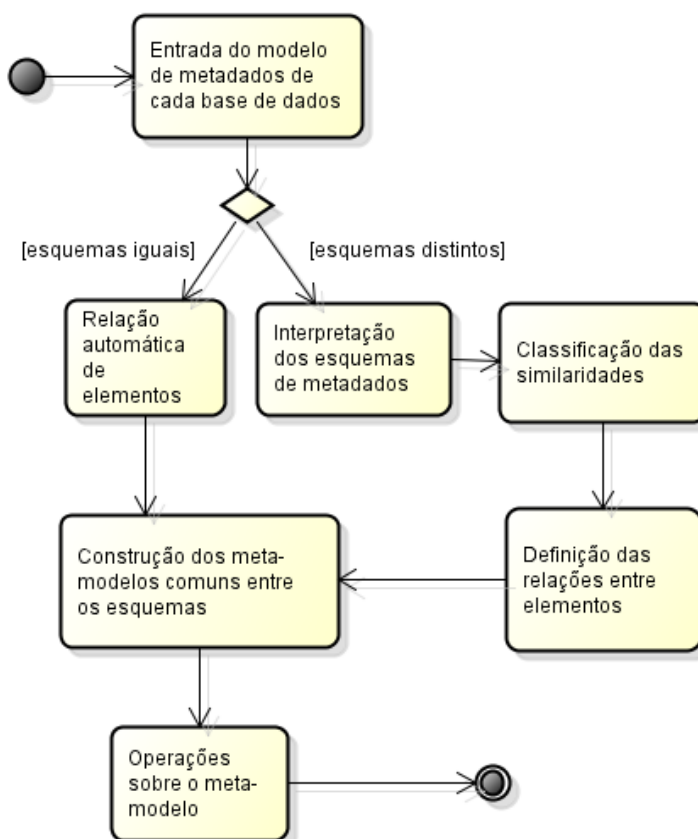


Figura 47. Diagrama de atividades descrevendo as etapas da execução do experimento

6.2.2.1 Entrada do modelo de metadados de cada domínio

O primeiro passo é definir os esquemas de metadados que serão integrados, par a par. Por exemplo, considerando as instituições citadas, seleciona-se o esquema da TV Brasil para se integrar com os da BBC e Cinemateca, um de cada vez. Através do formato do arquivo fornecido é detectada a linguagem de definição de esquemas utilizada, tornando o arquivo passível de interpretação pelo Dynamo Framework. Neste estudo de caso em particular, os esquemas de metadados foram fornecidos na linguagem XML Schema (W3C, 2006). Os arquivos contendo as informações dos esquemas são enviados via interface de administração do sistema M-MAP, no qual internamente repassará as informações para o Dynamo Framework, que ficará responsável por realizar a interpretação dos dados descritos nos arquivos.

6.2.2.2 Interpretação dos esquemas de metadados

Quando recebe os esquemas dos sistemas que serão integrados, o M-MAP aciona o componente Controller do Dynamo Framework, que por sua vez realiza uma análise dos arquivos e constrói uma estrutura de objetos Java contendo a representação dos elementos dos esquemas.

Devido à alta heterogeneidade das linguagens de definição de esquemas, não é possível encontrar um modelo de representação que suporte todas as características dos esquemas. Foi utilizada então a abordagem proposta por S. Amir et al. (S. Amir et al., 2011) na qual os esquemas são modelados como gráficos rotulados, representando somente conceitos básicos do esquema e propriedades ligadas entre si.

A representação de grafos contém os elementos referentes aos modelos que serão integrados, divididos em dois níveis. No primeiro nível estão os elementos que definem as classes (nível macro), enquanto que no segundo nível estão os atributos referentes às classes (nível micro). Essa divisão é feita visando otimizar a análise da correlação, visto que é possível inicialmente relacionar apenas as classes para só em um segundo momento levar em consideração os seus atributo.

Sendo assim, após o Dynamo Framework construir a estrutura de grafos contendo os elementos dos modelos, é feita a explicitação dos tokens e dos respectivos sinônimos (conforme foi descrito na Seção 5.3), visando aumentar o número de mapeamentos sugeridos.

A seguir são apresentados os esquemas de metadados considerados no estudo de caso após o processamento inicial, onde são definidos os tokens e são obtidos os sinônimos. A Tabela 13 mostra os elementos do esquema da base de dados da BBC, baseado no padrão de metadados TV-Anytime. A Tabela 14 exhibe os elementos do modelo de dados da TV Brasil, baseado no padrão PBCore. Já a Tabela 15 contém a descrição dos elementos do modelo de dados da Cinemateca, que não segue nenhum padrão pré-definido.

Tabela 13. Elementos processados do esquema da BBC baseado no padrão de metadados TV-Anytime

Elemento de origem	Tokens	Sinônimos
BasicDescription		
PromotionalInformation	Promotional, information	-
ParentalGuidance	Parental, guidance	control
Language	Language	idiom

Caption Language	Caption, language	Idiom, subtitle
Sign Language	Sign, language	signature
CreditsList	Credit	Right, authority
AwardsList	Award	-
RelatedMaterial	Related, material	-
ProductionDate	Production, date	creation, conception
ProductionLocation	Production, location	Point, localization, region
DepictedCoordinates	Depicted, coordinates	Describe, location
CreationCoordinates	Creation, coordinates	Production, conception, location
Duration	Duration	-
ContentReference		
authority	Author	Owner, creator
data	Data	File, bytes
CRID	Id, identifier	Identification
Title		
Text	Title, Text	Name
Synopsis		
Text	Synopsis, text	Description
Language	Synopsis, language	Description, Idiom
Genre		
Value	Genre, value	Category, classification

Tabela 14. Elementos processados do esquema da TV Brasil baseado no padrão de metadados PBCore

Elemento de origem	Tokens	Sinônimos
IntellectualContent		
pbcoreAssetType	Asset, type	Media, format
pbcoreAssetDate	Asset, date	Media, creation
pbcoreIdentifier	Id, identifier	Identification
pbcoreTitle	Title	Name
pbcoreSubject	Subject	Topic, theme
pbcoreDescription	Description	Synopsis, information
pbcoreGenre	Genre	Category, classification
pbcoreRelationType	Relation	Association
pbCoreRelationCoverage	Coverage	Depth
IntellectualProperty		
pbcoreRelationCreator	Creator	Author, owner
pbcoreRelationCreatorRole	Role, creator	Function, privilege

pbcorePublisher	publisher	-
pbcorePublisherRole	Role, publisher	Function, privilege
pbcoreRightSummary	Right, summary	-
Instantiation		
instantiationIdentifier	Instance, identifier, id	identification
instantiationDate	Instantiation, date	Media, timestamp
InstantiationDimensions	Dimensions	Size
instantiationPhysical	Physical, Format	-
instantiationDigital	Digital, Format	-
instantiationStandard	Standard	Criterion, control
instantiationLocation	Location	Place, point
instantiationMediaType	Media, type	-
instantiationGenerations	Generations	-
instantiationFileSize	File, size	Length
instantiationTimeStart	Time, start	Begin
instantiationDuration	Duration	Length
instantiationDataRate	data, rate	Speed
InstantiationColors	Color	-
InstantiationTrackers	Tracker	-
instantiationLanguage	Language	Idiom
InstantiationRights	Rights	-

Tabela 15. Elementos processados do esquema da Cinemateca

Elemento de origem	Tokens	Sinônimos
MovieDescription		
Title	Title	Name
GivenTitle	Given, Title	Attributed
Code	Code	Identifier
productionYear	Production, year	Production, creation, date
City	City	Location, place
State	State	Location, place
Director	Director	Direction
Producer	Producer	Production
Photography	Photo	Image
Synopsis	Synopsis	Information, description
Duration	Duration	Size
Genre	Genre	Category

Subject	Subject	Theme
Geographicterms	Geographic, terms	Location
Categories	Category, categories	Classification

Após a explicitação dos tokens e a obtenção dos sinônimos, o Dynamo Framework repassa essas informações para o componente Translator, que ficará responsável por acessar o sistema de correspondência (*Matching System*), e indicar o nível de correlação entre cada um dos elementos dos dois modelos em questão.

6.2.2.3 Classificação das similaridades

Para o desenvolvimento do Dynamo Framework foi utilizado como sistema de correspondência o MuMle (S.AMIR et al., 2011). O MuMle calcula os coeficientes de correlação com base nas similaridades linguísticas e estruturais. Para a similaridade linguística, são feitas comparações entre nomes de classes, elementos e comentários, sempre utilizando como guia a base de dados do Wordnet (WORDNET, 2012) para a detecção de sinônimos. Como a similaridade linguística pode prover vários candidatos, também é realizada a análise das similaridades em nível estrutural, visto que também pode acontecer de componentes que diferem em sua estrutura possuírem um alto grau de similaridade. O coeficiente de similaridade é calculado através da soma do grau de similaridade obtido após a computação das similaridades linguísticas e estruturais. Quando pronta, a lista com os elementos contendo os elementos passíveis de correlação é retornada para o Dynamo Framework.

Após a execução desta etapa é obtido um total de seis mapeamentos, resultantes da análise da correlação entre as combinações possíveis dos três esquemas de metadados citados anteriormente. A seguir são apresentados os resultados de cada um dos mapeamentos e também é realizada uma breve análise, indicando quais mapeamentos realmente indicam uma correlação real (correlação válida), parcial ou quando simplesmente não há nenhuma correlação entre os elementos sugeridos.

Foram consideradas apenas as correlações de elementos julgadas mais relevantes, ou seja, as que obtiveram o valor do coeficiente de correlação maior que 0,75 (sendo o intervalo de 0 a 1). Com isso, devido ao ponto de corte em 0,75, nem todos os elementos possuem alguma correlação sugerida. A Tabela 16 mostra quantos elementos aparecem com ao menos uma propriedade correlacionada em cada uma das combinações.

Tabela 16. Total de elementos contendo ao menos uma correlação sugerida a cada par de modelos

	BBC (TV-Anytime)	TV Brasil (PBCore)	Cinemateca
BBC (TV-Anytime)		12 de 21 (57%)	12 de 21 (57%)
TV Brasil (PBCore)	17 de 31 (54%)		19 de 31 (61%)
Cinemateca	13 de 15 (86%)	13 de 15 (86%)	

Para o cenário atual, obteve-se ao menos uma sugestão de correlação para em média 66% das propriedades dos modelos citados. Este valor pode variar de acordo com o ponto de corte (definido em 0,75 para este estudo de caso) ou com o número de sinônimos considerados. Caso o ponto de corte seja demasiadamente baixo ou o número de sinônimos considerados seja grande, pode ocorrer um aumento no número de falsos-positivos, podendo prejudicar a verificação do mapeamento por parte do usuário ou induzi-lo ao erro, por sugerir muitas correlações para elementos sem nada em comum. A definição do ponto de corte é realizada através da passagem de um parâmetro na chamada do método que fornece os coeficientes de correlação. No caso, o sistema cliente pode definir um ponto de corte distinto a cada chamada, inclusive comparando os resultados até obter um resultado que se adeque melhor às suas expectativas.

A seguir são apresentados os mapeamentos obtidos após a etapa de classificação de similaridades considerando as seis possibilidades de combinações entre modelos. Organiza-se da seguinte forma: elementos do modelo de origem na primeira coluna, sugestões de similaridades no modelo de destino na segunda coluna, os coeficientes de correlação entre os elementos na terceira e a classificação da correlação na quarta. No caso da classificação, trata-se de uma análise pessoal sobre a relevância da correlação levando em consideração a próxima fase de confirmação dos mapeamentos (ver subseção 5.3.4). São consideradas *válidas* as correlações aptas a serem mapeadas pelo usuário como relação de equivalência, sobreposição ou inclusão; *parcial* as que se encaixem como relações de sobreposição ou inclusão; e *sem correlação* as que sejam classificadas como relação de exclusão, ou seja, as que provavelmente serão ignoradas nas fases posteriores.

a) BBC (TV-Anytime) para TV Brasil (PBCore)

Tabela 17. Resultado dos mapeamentos sugeridos do esquema da BBC (TV-Anytime) para a TV Brasil (PBCore)

Elemento de origem	Elemento similar	Coeficientes	Classificação
BasicDescription			
Language	Instantiation.instantiationLanguage	0,92	Correlação válida
ProductionDate	Instantiation.instantiationDate	0,99	Correlação válida
	IntellectualProperty.pbcoreRelationType	0,88	Sem correlação
	IntellectualProperty.pbcoreRelationCreatorRole	0,88	Sem correlação
	IntellectualProperty.pbcoreRelationCreator	0,84	Sem correlação
	Instantiation.instantiationDataRate	0,83	Sem correlação

	Instantiation.instantiationLocation	0,80	Sem correlação
	Instantiation.instantiationDuration	0,80	Sem correlação
ProductionLocation	Instantiation.instantiationLocation	0,96	Correlação válida
CreationCoordinates	IntellectualContent.pbcoreRelationType	0,80	Sem correlação
	IntellectualProperty.pbcoreRelationType	0,91	Sem correlação
	IntellectualProperty.pbcoreRelationCreator	0,86	Correlação parcial
	IntellectualProperty.pbcoreRelationCreatorRole	0,83	Correlação parcial
	Instantiation.instantiationLocation	0,83	Sem correlação
Duration	Instantiation.instantiationDuration	0,83	Sem correlação
	Instantiation.instantiationDuration	0,91	Correlação válida
	IntellectualProperty.pbcoreRelationType	0,83	Sem correlação
Title	Instantiation.instantiationDuration	0,76	Sem correlação
	Instantiation.instantiationDuration	0,91	Correlação válida
	IntellectualProperty.pbcoreRelationType	0,83	Sem correlação
Text	Instantiation.instantiationDuration	0,76	Sem correlação
	Instantiation.instantiationDuration	0,91	Correlação válida
	IntellectualProperty.pbcoreRelationType	0,83	Sem correlação
Synopsis			
Text	IntellectualProperty.pbcoreDescription	0,96	Correlação válida
Language	Instantiation.instantiationLanguage	0,96	Correlação válida
	IntellectualProperty.pbcoreDescription	0,79	Sem correlação
Genre			
Value	IntellectualContent.pbcoreGenre	0,99	Correlação válida
	Instantiation.instantiationGenerations	0,75	Sem correlação
ContentReference			
Authority	IntellectualProperty.pbcoreRelationCreator	0,99	Correlação válida
	IntellectualProperty.pbcoreRelationCreatorRole	0,88	Correlação parcial
	IntellectualProperty.pbcoreRelationType	0,88	Sem correlação
	Instantiation.instantiationLocation	0,80	Sem correlação
	Instantiation.instantiationDuration	0,80	Sem correlação
	IntellectualProperty.pbcoreRightsSummary	0,77	Sem correlação
	Instantiation.instantiationGeneration	0,75	Sem correlação
Data	Instantiation.instantiationDataRate	0,99	Correlação parcial
	Instantiation.instantiationDate	0,83	Sem correlação
CRID	Instantiation.instantiationIdentifier	0,99	Correlação válida
	IntellectualProperty.pbcoreIdentifier	0,99	Correlação válida

b) TV Brasil (PBCore) para BBC(TV-Anytime)

Tabela 18. Resultado dos mapeamentos sugeridos a partir do esquema de metadados da TV Brasil (PBCore) para o da BBC(TV-Anytime)

Elemento de origem	Elemento similar	Coeficientes	Classificação
IntellectualContent			
pbcoreIdentifier	ContentReference.CRID	0,99	Correlação válida
pbcoreTitle	Title.text	0,99	Correlação válida
pbcoreDescription	Synopsis.text	0,96	Correlação válida
	Synopsis.language	0,78	Sem correlação
pbcoreGenre	Genre.value	0,99	Correlação válida

pbcoreRelationType	BasicDescription.creationCoordinates	0,91	Sem correlação
	BasicDescription.productionDate	0,88	Sem correlação
	ContentReference.authority	0,88	Sem correlação
	BasicDescription.duration	0,83	Sem correlação
	BasicDescription.productionLocation	0,80	Sem correlação
IntellectualProperty			
pbcoreRelationCreator	ContentReference.authority	0,99	Correlação válida
	BasicDescription.creationCoordinates	0,86	Correlação parcial
	BasicDescription.productionDate	0,84	Sem correlação
pbcoreRelationCreatorRole	ContentReference.authority	0,88	Correlação parcial
	BasicDescription.productionDate	0,88	Sem correlação
	BasicDescription.creationCoordinates	0,83	Correlação parcial
pbcoreRightsSummary	ContentReference.authority	0,77	Sem correlação
	ContentReference.productionDate	0,77	Sem correlação
Instantiation			
instantiationIdentifier	ContentReference.CRID	0,99	Correlação válida
instantiationDate	BasicDescription.productionDate	0,99	Correlação válida
	ContentReference.data	0,83	Sem correlação
instantiationLocation	BasicDescription.productionLocation	0,96	Correlação válida
	BasicDescription.creationCoordinates	0,83	Correlação válida
	ContentReference.authority	0,80	Sem correlação
	ContentReference.productiondate	0,80	Sem correlação
instantiationGenerations	BasicDescription.creationCoordinates	0,78	Sem correlação
	BasicDescription.duration	0,76	Sem correlação
	BasicDescription.productionDate	0,75	Sem correlação
	Genre.value	0,75	Sem correlação
	ContentReference.authority	0,75	Sem correlação
InstantiationFileSize	Title.text	0,78	Sem correlação
InstantiationTimeStart	Title.text	0,78	Sem correlação
instantiationDuration	BasicDescription.duration	0,91	Correlação válida
instantiationDataRate	ContentReference.data	0,99	Correlação parcial
	BasicDescription.ProductionDate	0,83	Sem correlação
	ContentReference.authority	0,78	Sem correlação
instantiationLanguage	Synopsis.language	0,96	Correlação válida
	BasicDescription.language	0,91	Correlação válida

c) TV Brasil (PBCore) para Cinemateca

Tabela 19. Resultado dos mapeamentos sugeridos do esquema da TV Brasil (PBCore) para a Cinemateca

Elemento de origem	Elemento similar	Coeficientes	Classificação
IntellectualContent			
pbcoreIdentifier	MovieDescription.code	0,96	Correlação válida
pbcoreTitle	MovieDescription.text	0,99	Correlação válida
	MovieDescription.givenTitle	0,96	Correlação válida
pbcoreSubject	MovieDescription.subject	0,99	Correlação válida
pbcoreDescription	MovieDescription.synopsis	0,97	Correlação válida
	MovieDescription.director	0,86	Sem correlação

pbcoreGenre	MovieDescription.genre	0,99	Correlação válida
pbcoreRelationType	MovieDescription.duration	0,83	Sem correlação
	MovieDescription.state	0,80	Sem correlação
	MovieDescription.director	0,80	Sem correlação
	MovieDescription.city	0,80	Sem correlação
	MovieDescription.geographicTerms	0,80	Sem correlação
	MovieDescription.synopsis	0,77	Sem correlação
IntellectualProperty			
pbcoreRelationCreator	MovieDescription.categories	0,81	Sem correlação
	MovieDescription.productionYear	0,78	Sem correlação
	MovieDescription.genre	0,78	Sem correlação
	MovieDescription.director	0,75	Correlação válida
pbcoreRelationCreatorRole	MovieDescription.genre	0,83	Sem correlação
	MovieDescription.director	0,8	Correlação parcial
	MovieDescription.productionYear	0,80	Sem correlação
	MovieDescription.categories	0,78	Sem correlação
pbcoreRightsSummary	MovieDescription.synopsis	0,80	Sem correlação
	MovieDescription.director	0,75	Correlação parcial
Instantiation			
instantiationIdentifier	MovieDescription.code	0,99	Correlação válida
	MovieDescription.state	0,76	Sem correlação
instantiationDate	MovieDescription.productionYear	0,93	Correlação válida
instantiationPhysical	MovieDescription.synopsis	0,86	Sem correlação
instantiationDigital	MovieDescription.synopsis	0,86	Sem correlação
instantiationLocation	MovieDescription.geographicTerms	0,96	Correlação válida
	MovieDescription.city	0,96	Correlação válida
	MovieDescription.state	0,96	Correlação válida
	MovieDescription.producer	0,78	Sem correlação
instantiationGenerations	MovieDescription.duration	0,76	Sem correlação
	MovieDescription.genre	0,76	Sem correlação
InstantiationFileSize	MovieDescription.title	0,78	Sem correlação
InstantiationTimeStart	MovieDescription.state	0,82	Sem correlação
	MovieDescription.title	0,78	Sem correlação
instantiationDuration	MovieDescription.duration	0,99	Correlação válida
	MovieDescription.director	0,80	Sem correlação
instantiationDataRate	MovieDescription.productionYear	0,86	Sem correlação

d) BBC (TV-Anytime) para Cinemateca

Tabela 20. Resultado dos mapeamentos sugeridos do esquema da BBC (TV-Anytime) para a Cinemateca

Elemento de origem	Elemento similar	Coeficientes	Classificação
BasicDescription			
CreditsList	MovieDescription.director	0,76	Correlação válida
	MovieDescription.producer	0,76	Correlação válida
ProductionDate	MovieDescription.producer	0,96	Sem correlação
	MovieDescription.productionYear	0,93	Correlação válida
	MovieDescription.state	0,85	Sem correlação
	MovieDescription.city	0,85	Sem correlação
	MovieDescription.geographicTerms	0,85	Sem correlação

	MovieDescription.director	0,80	Sem correlação
	MovieDescription.duration	0,80	Sem correlação
	MovieDescription.photography	0,76	Sem correlação
ProductionLocation	MovieDescription.geographicTerms	0,99	Correlação válida
	MovieDescription.city	0,99	Correlação válida
	MovieDescription.state	0,99	Correlação válida
	MovieDescription.producer	0,93	Sem correlação
	MovieDescription.productionYear	0,80	Sem correlação
	MovieDescription.director	0,77	Sem correlação
	MovieDescription.duration	0,83	Sem correlação
CreationCoordinates	MovieDescription.producer	0,81	Correlação parcial
	MovieDescription.director	0,81	Correlação parcial
	MovieDescription.city	0,80	Sem correlação
	MovieDescription.productionYear	0,80	Sem correlação
	MovieDescription.state	0,80	Sem correlação
	MovieDescription.duration	0,94	Correlação válida
Duration	MovieDescription.duration	0,94	Correlação válida
Title			
Text	MovieDescription.title	0,99	Correlação válida
	MovieDescription.givenTitle	0,94	Sem correlação
Synopsis			
Text	MovieDescription.synopsis	0,99	Correlação válida
Language	MovieDescription.synopsis	0,82	Sem correlação
Genre			
Value	MovieDescription.genre	0,99	Correlação válida
ContentReference			
Authority	MovieDescription.city	0,85	Correlação válida
	MovieDescription.geographicTerms	0,85	Correlação parcial
	MovieDescription.state	0,85	Sem correlação
	MovieDescription.productionYear	0,82	Sem correlação
	MovieDescription.duration	0,80	Sem correlação
	MovieDescription.director	0,80	Sem correlação
	MovieDescription.synopsis	0,76	Sem correlação
	MovieDescription.genre	0,75	Sem correlação
Data	MovieDescription.productionYear	0,78	Sem correlação
	MovieDescription.geographicTerms	0,77	Sem correlação
CRID	MovieDescription.code	0,99	Correlação válida

e) Cinemateca para BBC (TV-Anytime)

Tabela 21. Resultado dos mapeamentos sugeridos do esquema da Cinemateca para a BBC (TV-Anytime)

Elemento de origem	Elemento similar	Coeficientes	Classificação
MovieDescription			
Title	Title.text	0,99	Correlação válida
Code	ContentReference.CRID	0,99	Correlação válida
GivenTitle	Title.text	0,94	Correlação válida
ProductionYear	BasicDescription.productionDate	0,93	Correlação válida
	ContentReference.authority	0,82	Sem correlação

	BasicDescription.creationCoordinates	0,80	Sem correlação
	BasicDescription.productLocation	0,80	Sem correlação
	ContentReference.data	0,78	Sem correlação
City	BasicDescription.productLocation	0,99	Correlação válida
	ContentReference.authority	0,85	Sem correlação
	BasicDescription.productionDate	0,85	Sem correlação
	BasicDescription.creationCoordinates	0,80	Sem correlação
State	BasicDescription.productLocation	0,99	Correlação válida
	ContentReference.authority	0,85	Sem correlação
	BasicDescription.productionDate	0,85	Sem correlação
	BasicDescription.creationCoordinates	0,80	Sem correlação
Director	BasicDescription.creationCoordinates	0,81	Sem correlação
	ContentReference.authority	0,80	Correlação parcial
	BasicDescription.productionDate	0,80	Sem correlação
	BasicDescription.produtionLocation	0,77	Sem correlação
	BasicDescription.creditsList	0,76	Correlação parcial
Producer	BasicDescription.productionDate	0,96	Sem correlação
	BasicDescription.productLocation	0,93	Sem correlação
	BasicDescription.creationCoordinates	0,81	Sem correlação
	BasicDescription.creditList	0,76	Correlação parcial
Photography	BasicDescription.productionDate	0,76	Sem correlação
Synopsis	Synopsis.text	0,99	Correlação válida
	Synopsis.language	0,82	Sem correlação
	ContentReference.authority	0,76	Sem correlação
	BasicDescription.productionDate	0,76	Sem correlação
Duration	BasicDescription.duration	0,99	Correlação válida
	BasicDescription.creationCoordinates	0,83	Sem correlação
	ContentReferecen.authority	0,80	Sem correlação
	BasicDescription.productiondate	0,80	Sem correlação
Genre	Genre.value	0,99	Correlação válida
Geographicterms	BasicDescription.creationCoordinates	0,99	Correlação válida
	BasicDescription.productionLocation	0,99	Correlação válida
	ContentReference.authority	0,85	Sem correlação
	BasicDescription.productionDate	0,85	Sem correlação
	ContentReference.data	0,77	Sem correlação

f) Cinemateca para TV Brasil (PBCore)

Tabela 22. Resultado dos mapeamentos sugeridos do esquema da Cinemateca para a TV Brasil (PBCore)

Elemento de origem	Elemento similar	Coeficientes	Classificação
MovieDescription			
Title	IntellectualContent.pbcoreTitle	0,99	Correlação válida
	Instantiation.instantiationTimeStart	0,78	Sem correlação
	Instantiation.instantiationFileSize	0,78	Sem correlação
Code	Instantiation.instantiationidentifier	0,99	Correlação válida
	IntellectualContent.pbcoreIdentifier	0,96	Correlação válida

GlvenTitle	IntellectualContent.pbcoreTitle	0,94	Correlação válida
productionYear	Instantiation.instantiationDate	0,93	Correlação válida
	Instantiation.instantiationDataRate	0,86	Sem correlação
	IntellectualProperty.pbcoreRelationCreatorRole	0,80	Sem correlação
	IntellectualProperty.pbcoreRelationCreator	0,78	Sem correlação
City	Instantiation.instantiationLocation	0,96	Correlação válida
	IntellectualContent.pbcoreRelationType	0,80	Sem correlação
State	Instantiation.instantiationLocation	0,96	Correlação válida
	Instantiation.instantiationTimeStart	0,82	Sem correlação
	IntellectualContent.pbcoreRelationType	0,80	Sem correlação
	Instantiation.instantiationIdentifier	0,76	Sem correlação
Director	IntellectualContent.pbcoreDescription	0,86	Sem correlação
	IntellectualProperty.pbcoreRelationCreatorRole	0,80	Correlação parcial
	Instantiation.instantiationDuration	0,80	Sem correlação
	IntellectualContent.pbcoreRelationType	0,80	Sem correlação
	IntellectualProperty.pbcoreRightsSummary	0,75	Sem correlação
	IntellectualProperty.pbcoreRelationCreator	0,75	Correlação válida
Producer	Instantiation.instantiationLocation	0,78	Sem correlação
Synopsis	IntellectualContent.pbcoreDescription	0,97	Correlação válida
	Instantiation.instantiationPhysical	0,86	Sem correlação
	Instantiation.instantiationDigital	0,86	Sem correlação
	IntellectualProperty.pbcoreRightsSummary	0,80	Sem correlação
	IntellectualContent.pbcoreRelationType	0,77	Sem correlação
Duration	id=Instantiation.instantiationDuration	0,99	Correlação válida
	id=IntellectualContent.pbcoreRelationType	0,83	Sem correlação
	Instantiation.instantiationGenerations	0,76	Sem correlação
Genre	IntellectualContent.pbcoreGenre	0,99	Correlação válida
	IntellectualProperty.pbcoreRelationCreatorRole	0,83	Sem correlação
	IntellectualProperty.pbcoreRelationCreator	0,78	Sem correlação
	Instantiation.instantiationGenerations	0,75	Sem correlação
Geographicterms	Instantiation.instantiationLocation	0,96	Correlação válida
	IntellectualContent.pbcoreRelationType	0,80	Sem correlação
Categories	IntellectualProperty.pbcoreRelationCreator	0,81	Sem correlação
	IntellectualProperty.pbcoreRelationCreatorRole	0,78	Sem correlação

6.2.2.4 Definição das relações entre elementos

Após receber o cálculo dos coeficientes gerados pelo sistema de correspondência, o Dynamo Framework repassa o resultado para o sistema cliente, que por sua vez ficará responsável por exibir a informação na interface de administração. Assim, o usuário consegue visualizar os resultados da análise e confirmar as relações entre elementos. Na Seção 5.3.4 foram citados os quatro tipos de relações possíveis (equivalência, inclusão,

exclusão e sobreposição), além disso, há o suporte às funções pré-definidas (e.g. split, concat) para realizar algum tratamento no valor da propriedade, no caso de haver a necessidade de capturar, por exemplo, apenas parte da informação (e.g. ano em uma data).

a) Relação de Equivalência

As relações de equivalência são atribuídas quando um elemento de origem possui correspondência com apenas um elemento do modelo de destino, como exemplo temos o campo do TV-Anytime “Title.text” que corresponde ao “IntellectualProperty.pbcoreTitle” do PBCore. Nesse caso, também é possível usar uma função adicional para selecionar parte da informação disponível na propriedade, por exemplo, a propriedade “MovieDescription.ProductionYear” da Cinemateca está relacionada à propriedade “BasicDescription.productionDate” do TV-Anytime, porém, a propriedade da cinemateca considera apenas o ano, para tal atribui-se uma função adicional (split passando como parâmetro o divisor de data “/”) e recuperando o primeiro elemento da divisão, que corresponde ao ano, no padrão de datas americano, que é o padrão para várias bases de dados. A Tabela 23 mostra um exemplo da atribuição de relações de equivalência no estudo de caso atual.

Tabela 23. Exemplo de atribuição das relações de equivalência no mapeamento dos modelos

Elemento de origem	Elemento de destino	Relação	Função adicional
Title.text	IntellectualProperty.pbcoreTitle	equivalência	-
MovieDescription.ProductionYear	BasicDescription.productionDate	equivalência	split("/") [0]

b) Relação de Inclusão

As relações de inclusão aparecem quando um dado elemento de origem se relaciona à 2 ou mais elementos de destino, através de operações como a concatenação. Como exemplo usando o estudo de caso atual, por exemplo, temos o elemento “ContentReference.CRID” do TV-Anytime que possui similaridade para a concatenação entre os campos IntellectualProperty.pbCoreIdentifier e instantiation.instantiationIdentifier, visto que o CRID deve possuir uma identificação única que represente tanto as informações em relação ao ativo digital, como as informações da mídia atribuída a ele. A Tabela 24 mostra a relação citada como exemplo.

Tabela 24. Exemplo de atribuição da relação de inclusão no mapeamento dos modelos

Elemento de origem	Elemento de destino
--------------------	---------------------

ContentReference.CRID	IntellectualProperty.pbcoreIdentifier
	Instantiation.instantiationIdentifier

c) Relação de Exclusão

As relações de exclusão ocorrem quando dois elementos que foram sugeridos como correlatos não possuem efetivamente uma relação comum. Nesse caso, atribui-se a relação de exclusão e o elemento não será considerado no meta-modelo comum. A Tabela 25 mostra um exemplo de sugestão de correlação que resulta numa relação de exclusão: temos a propriedade “ContentReference.data” do TV-Anytime como origem e a propriedade “Instantiation.instantiationDataRate” como destino. Como “data” representa os dados em si (conjunto de bytes) e “datarate” a taxa de transmissão dos bytes, não há uma efetiva relação entre as propriedades, logo, é atribuída a relação de exclusão para esse mapeamento.

Tabela 25. Exemplo de atribuição da relação de exclusão no mapeamento dos modelos

Elemento de origem	Elemento de destino
ContentReference.data	Instantiation.instantiationDataRate

d) Relação de Sobreposição

As relações de sobreposição aparecem quando um elemento possui uma correlação fraca ou até nenhuma correlação, mas é importante para ser considerado no modelo genérico. Por exemplo, a Tabela 26 mostra o mapeamento do TV-Anytime para o elemento “IntellectualProperty.pbcoreRelationCreatorRole” do PBCore que define o papel do autor (“authority” no TV-Anytime). Entretanto, o TV-Anytime não possui o campo “authorityRole”, que é criado após a explicitação da relação de sobreposição com o elemento de destino.

Tabela 26. Exemplo de atribuição da relação de sobreposição no mapeamento dos modelos

Elemento de origem	Elemento de destino
ContentReference.authorityRole	IntellectualProperty.pbcoreRelationCreatorRole

6.2.2.5 Construção do meta-modelo contendo o mapeamento genérico entre os esquemas

Considerando as correlações sugeridas, para medir o nível de acerto foi realizada a seguinte métrica: dentre os mapeamentos sugeridos, em quantos a primeira correlação (a de mais alto coeficiente de correspondência) aparece como correlação válida? A Tabela 27

traz esses números, calculando a porcentagem em relação ao total de propriedades com correlações sugeridas.

Tabela 27. Análise do nível de acerto das sugestões de mapeamento entre os esquemas

	BBC (TV-Anytime)	TV Brasil (PBCore)	Cinemateca
BBC (TV-Anytime)		83,3%	58,3%
TV Brasil (PBCore)	58,8%		47,3%
Cinemateca	76,9%	76,9%	

A Tabela 27 mostra que o mapeamento mais eficiente foi o que partiu do esquema da BBC (TV-Anytime) para o da TV Brasil (PBCore), trazendo em 83,3% das vezes uma correlação válida como primeira opção apresentada a cada elemento. Já o mapeamento menos eficiente foi o da TV Brasil (PBCore) para a Cinemateca, onde apenas em 47,3% das vezes a primeira opção representava uma correlação válida. Como a Cinemateca possui um modelo de dados que não segue nenhum padrão e utiliza elementos criados para suprir basicamente as necessidades da instituição, fica comprovado que pode não ser muito eficiente o mapeamento de um padrão existente para um padrão proprietário. Entretanto, ao analisar o mapeamento utilizando como referência o modelo da Cinemateca, há uma grande melhora, visto que tanto o TV-Anytime e o PBCore contemplam boa parte das necessidades da Cinemateca, apesar do inverso não acontecer.

A análise do nível de acerto e a comparação de quem se enquadra melhor como modelo de origem ou destino é importante para definir como os meta-modelos serão construídos. Ressalta-se que cada meta-modelo será único para cada par de esquemas de metadados integrado e terá o papel de servir como meio de acesso comum a ambos. Visando sempre buscar uma maior possibilidade de acerto, foram selecionados os modelos de origem e destino baseados no nível de acerto das sugestões de mapeamento; assim, para a construção do meta-modelo, foi considerada a ordem descrita na Tabela 28.

Tabela 28. Definição dos modelos de origem e destino para a construção do meta-modelo

Bases consideradas	Origem	Destino	Acerto obtido nas sugestões de mapeamento
BBC e TV Brasil	BBC (TV-Anytime)	TV Brasil (PBCore)	83,3%
TV Brasil e Cinemateca	Cinemateca	TV Brasil (PBCore)	76,9%
BBC e Cinemateca	Cinemateca	BBC (TV-Anytime)	69,2%

Após a atribuição das relações por parte do usuário, constrói-se um meta-modelo comum (meta-modelo) para cada par de esquemas considerados no estudo de caso. A Tabela 29 mostra os elementos do meta-modelo a partir do mapeamento entre BBC (TV-

Anytime e TV Brasil (PBCore). A Tabela 30 mostra os elementos do meta-modelo baseado no mapeamento entre Cinemateca e TV Brasil (PBCore). Já a Tabela 31 mostra os elementos do meta-modelo construído a partir do mapeamento entre Cinemateca e BBC (TV-Anytime). Para cada caso, apresenta-se o nome que do elemento que fará parte do meta-modelo, quais os elementos relacionados no esquema mapeado e a classificação da relação atribuída após a etapa de definição das relações entre elementos.

a) Elementos do meta-modelo: BBC (TV-Anytime) e TV Brasil (PBCore)

Tabela 29. Elementos usados para a construção do meta-modelo a partir dos esquemas da BBC (TV-Anytime) e TV Brasil (PBCore)

Nome do elemento	Elemento relacionado	Classificação de Relação
Content		
BasicDescription		
Language	Instantiation.instantiationLanguage	equivalência
ProductionDate	Instantiation.instantiationDate	equivalência
ProductionLocation	Instantiation.instantiationLocation	equivalência
CreationCoordinates	IntellectualProperty.pbcoreRelationType	exclusão
Duration	Instantiation.instantiationDuration	equivalência
Title		
Text	IntellectualProperty.pbcoreTitle	equivalência
Synopsis		
Text	IntellectualProperty.pbcoreDescription	equivalência
Language	Instantiation.instantiationLanguage	equivalência
Genre		
Value	IntellectualContent.pbcoreGenre	equivalência
ContentReference		
Authority	IntellectualProperty.pbcoreRelationCreator	equivalência
AuthorityRole	IntellectualProperty.pbcoreRelationCreatorRole	sobreposição
Data	Instantiation.instantiationDataRate	exclusão
CRID	IntellectualProperty.pbcoreIdentifier	inclusão
	Instantiation.instantiationIdentifier	

b) Elementos do meta-modelo: Cinemateca e BBC (TV-Anytime)

Tabela 30. Elementos usados para a construção do meta-modelo a partir dos esquemas da Cinemateca e BBC (TV-Anytime)

Nome do elemento	Elemento relacionado	Classificação da relação	Função adicional
MovieDescription			
Title	Title.text	equivalência	
Code	ContentReference.CRID	equivalência	

GivenTitle	Title.text	equivalência	
productionYear	BasicDescription.productionDate	equivalência	split("/") - pos: 0
City	BasicDescription.productLocation	equivalência	split(",") - pos: 3
State	BasicDescription.productLocation	equivalência	split(",") - pos: 2
Director	BasicDescription.creditsList	equivalência	get("director")
Producer	BasicDescription.creditsList	equivalência	get("producer")
Photography	BasicDescription.productionDate	exclusão	
Synopsis	Synopsis.text	equivalência	
Duration	BasicDescription.duration		
Genre	Genre.value		
GeographicTerms	BasicDescription.creationCoordinates	inclusão	
	BasicDescription.productionLocation		

c) Elemento do meta-modelo: Cinemateca e TV Brasil (PBCore)

Tabela 31. Elementos usados para a construção do meta-modelo a partir dos esquemas da Cinemateca e TV Brasil (PBCore)

Nome do elemento	Elemento relacionado	Classificação da relação	Função adicional
MovieDescription			
Title	IntellectualContent.pbcoreTitle	equivalência	
Code	IntellectualContent.pbcoreIdentifier	inclusão	
	Instantiation.instantiationIdentifier		
GlvenTitle	IntellectualContent.pbcoreTitle	equivalência	
productionYear	Instantiation.instantiationDate	equivalência	split("/") - pos: 0
City	Instantiation.instantiationLocation	equivalência	split(",") - pos: 2
State	Instantiation.instantiationLocation	equivalência	split(",") - pos: 3
Director	IntellectualProperty.pbcoreRelationCreator	equivalência	
Synopsis	IntellectualContent.pbcoreDescription	equivalência	
Duration	Instantiation.instantiationDuration	equivalência	
Genre	IntellectualContent.pbcoreGenre	equivalência	
Geographicterms	Instantiation.instantiationLocation	equivalência	
Categories	IntellectualProperty.pbcoreRelationCreator	exclusão	

A geração do meta-modelo inicia-se a partir da confirmação, pelo usuário, das propriedades correlacionadas entre os esquemas que estão sendo mapeados. Após essa etapa, deve ser selecionado também qual será o modelo de representação do meta-modelo (e.g. RDFS ou SQL-DDL). Em seguida ocorrerá o armazenamento num ponto central de cada domínio, possibilitando, assim, uma linguagem de acesso comum entre os domínios.

Para o estudo de caso em questão, foi utilizada a linguagem RDF Schema para armazenar o meta-modelo. Para a realização de consultas foi adotada a linguagem SPARQL, visto que é uma linguagem que opera sobre modelos de dados construídos em RDF, tornando-se possível a realização de consultas posteriores no meta-modelo gerado.

O código RDF gerado para a representação dos três meta-modelos oriundos do estudo de caso em questão podem ser visualizados no Anexo B deste trabalho. A sintaxe de consultas em SPARQL funciona conforme foi explicitado na Seção 5.3.6.

6.2.3 Análise do Experimento

Como resultado do processo de interoperabilidade entre os modelos de dados das bases descritas, foram construídos três meta-modelos, um para cada combinação de bases de dados: TV Brasil x BBC, TV Brasil x Cinemateca e BBC x Cinemateca. O armazenamento dos meta-modelos foi realizado na base de redes integradas no Dynamo Framework, que contém a referência para os três modelos de dados, podendo ser acessado por qualquer sistema que utilize uma das três bases, onde o sistema referido fará o papel realizado pelo M-MAP neste estudo de caso.

O Dynamo Framework, com auxílio do sistema de correspondência de metadados, ofereceu ao usuário em média 66% de sugestões de mapeamentos, mas ressalta-se que, após a fase de confirmação por parte do usuário, apenas uma propriedade foi inserida como relação de sobreposição, ou seja, os 34% de propriedades que foram apontadas como não-relacionadas pelo Dynamo Framework realmente não possuíam nenhuma propriedade correlacionada no outro modelo, mesmo após avaliação do usuário, possibilitando então um acerto efetivo de 98%.

Dentro desse total de propriedades sugeridas, o nível de acerto logo no primeiro elemento sugerido (o de mais alta correlação), foi de 76,4%, o que contribuiu significativamente para a diminuição do trabalho manual do usuário, possibilitando que fossem realizados pequenos ajustes para a construção do meta-modelo unificado.

A análise das correspondências testando as seis combinações possíveis entre os esquemas considerados possibilitou a percepção de qual ordem deveria ser seguida (qual deveria ser o modelo de origem e qual deveria ser o modelo de destino), desta forma, pôde-se obter um maior número de correspondências entre os modelos.

Com a existência dos meta-modelos na base de redes integradas, tornou-se possível a execução de consultas interoperáveis entre as bases, utilizando os meta-modelos construídos como linguagem comum de acesso, que é o principal objetivo do Dynamo Framework.

Apesar de o Dynamo Framework ter descartado com 98% de eficiência os elementos não-correlacionados e indicado com 76,4% de acerto os elementos similares, ainda há a dependência por parte do usuário para validar os mapeamentos. Ressalta-se que o usuário possui autonomia para alterar à vontade os mapeamentos sugeridos, portanto, seu papel é fundamental para que a interoperabilidade de metadados seja alcançada. Dessa forma, recomenda-se que o usuário que opere o processo de integração conheça bem os modelos de dados das bases que estão sendo integradas, a ponto de saber identificar acertadamente as equivalências entre propriedades.

Outro fator a destacar sobre o experimento é a variação dos resultados de acordo com a alteração de algumas variáveis, como o coeficiente de correlação que servirá como ponto de corte e o número de sinônimos considerados. Entretanto, deve-se avaliar com cuidado a alteração destes valores, visto que quanto menor valor definido como ponto de corte, mais aparecerão elementos sugeridos para o usuário analisar e quanto maior a quantidade de sinônimos considerados, maiores serão as chances do surgimento de falsos positivos. Uma análise mais detalhada sobre quais seriam os valores ideais para essas variáveis teria que levar em consideração um estudo sobre os modelos em questão e um experimento que realizasse várias combinações a fim de descobrir com que valores seria obtido o melhor resultado.

No estudo de caso em questão, simulou-se um ambiente de integração inicial, considerando que não houve integrações anteriores. Desta forma, após as combinações de mapeamentos realizadas no estudo de caso, o componente Extender do Dynamo Framework é acionado, a fim de armazenar o histórico de interações por parte do usuário em cima dos modelos, ajudando a criar uma inteligência própria no framework, que poderá possibilitar um maior número de acertos à medida que são realizados novos mapeamentos, fazendo com que o sistema aprenda com os mapeamentos realizados a cada fase.

7

Considerações Finais

Este trabalho apresentou o Dynamo Framework, um framework de serviços focado no apoio ao processo de interoperabilidade de metadados entre bases de dados heterogêneas pertencentes ao domínio de gerenciamento de ativos digitais.

O Dynamo Framework foi validado através da implementação de uma prova de conceito, chamada M-MAP, possibilitando a realização de um estudo de caso prático utilizando bases de dados heterogêneas baseadas em modelos reais. Os testes práticos realizados mostram que o framework foi capaz de fornecer apoio durante o processo de interoperabilidade, viabilizando o estabelecimento de uma linguagem comum para consultas interoperáveis. A utilização do Dynamo Framework facilita, portanto, a integração de diferentes redes de sistemas de gerenciamento de ativos digitais e a consequente criação da chamada comunicação inter-redes. As conexões e mapeamentos com outros sistemas são cadastrados e administrados pelo framework.

7.1 Contribuições do trabalho em relação ao estado da arte

Os requisitos do Dynamo framework foram definidos de acordo com três principais problemas no âmbito da integração entre sistemas heterogêneos citados por Haslofer e Klas (HASLHOFER e KLAS, 2010): Compatibilidade com formatos de armazenamento de dados e interfaces de acesso (P1); Integração de modelos de metadados heterogêneos (P2) e Localização transparente e descoberta (P3). Após a explicitação da proposta e a obtenção do resultado dos experimentos, descreve-se a seguir como o Dynamo Framework tratou cada um dos problemas e em quais pontos a resolução foi alcançada.

P1) Compatibilidade com formatos de armazenamento de dados e interfaces de acesso

A resolução da compatibilidade com formatos de armazenamento de dados se deu principalmente através da etapa de correspondência de linguagens (ver subseção 5.3.2). Nesta etapa foram analisadas as linguagens no qual os modelos de metadados estão representados e foi realizado o mapeamento nesse nível, possibilitando a utilização (via componente Extender do Dynamo Framework), de mapeamentos já existentes entre as linguagens de definição de esquema mais utilizadas atualmente. Logo, tornou-se possível a

compatibilidade com os modelos, mesmo que representados em arquivos com formatos e linguagens diferentes, como xsd (XML Schema), rdf (RDFs) ou sql (SQL-DDL).

Sobre a compatibilidade com interfaces de acesso, após construir e armazenar os meta-modelos, uma API Java é disponibilizada pelo Dynamo Framework para que se possa fazer uso dos seus recursos, oferecendo uma interface comum que pode ser utilizada por qualquer sistema baseado em Java. Como perspectiva futura pode-se considerar a implementação de serviços web (web services) para fornecer a compatibilidade dos métodos do Dynamo Framework com outras linguagens de programação além de Java.

P2) Integração de modelos de metadados heterogêneos

A resolução do problema da integração de modelos de metadados heterogêneos passa pelas principais etapas de execução do fluxo de trabalho do Dynamo Framework, descritas nas subseções da seção 5.3. Foram realizados inicialmente pré-processamentos visando enriquecer cada propriedade do modelo, incluindo além do nome, palavras-chave extraídas de comentários e sinônimos obtidos a partir de bases léxicas existentes. Em seguida foram mapeados os elementos que possuem alguma similaridade, juntamente com o coeficiente de correlação, que indica o nível de proximidade entre os elementos sugeridos. A etapa seguinte contou a contribuição do usuário, que analisou os mapeamentos e classificou as relações sugeridas pelo sistema. Por fim, foi gerado um meta-modelo que serviu como linguagem de acesso comum e possibilitou a consulta por atributos para a realização de operações entre as bases integradas.

Desta forma, pode-se dizer que houve uma boa contribuição para a resolução deste problema. Apesar da dependência da confirmação por parte de um usuário humano, o Dynamo Framework possibilita um conjunto de recursos que oferecem apoio ao usuário durante as várias etapas do processo, possibilitando uma considerável diminuição do esforço, gerando economia de recursos e tempo.

Além disso, a arquitetura no qual foi construído o framework possui módulos de inteligência aptos a armazenar os mapeamentos já feitos. Logo, caso dois modelos de dados fornecidos já tenham sido mapeados anteriormente, o framework oferecerá o mapeamento considerando os ajustes conhecidos, possibilitando assim um processo praticamente automático. Logo, a expectativa é que com o amadurecimento do uso, possa-se aumentar ainda mais a eficiência do framework para auxiliar na resolução do problema da interoperabilidade de metadados de forma a exigir menor participação do usuário.

P3) Localização transparente e descoberta

O Dynamo Framework não faz acesso à base de dados por vias externas (e.g. consultas pela rede), a ideia é possibilitar que ele seja agregado facilmente a qualquer aplicação que queira se integrar com outras, fazendo o papel de um componente integrador independente, que possuirá uma base customizada para cada sistema cliente que o utilizar. Desta forma, a informação sobre como acessar as bases de destino fica a cargo do sistema cliente, sendo o Dynamo Framework o responsável por possibilitar a construção de um meio para o acesso comum entre as bases, através do mapeamento dos modelos de dados.

Após o mapeamento entre pares de modelos de dados, são construídos os meta-modelos, e partir disso torna-se possível a solicitação de uma consulta de busca por elementos pertencentes às bases integradas. Como as bases agora falam uma mesma linguagem em relação à metadados pode-se solicitar o valor de um conjunto de propriedades, através de qualquer interface de acesso disponível pelo sistema a que a consulta se destina (e.g. query SQL, web services, etc). Dessa forma consegue-se obter informações da base de destino interagindo diretamente com o meta-modelo, que fará o papel de intermediador, converterá as consultas baseando-se no mapeamento realizado e retornará as informações no formato compatível com cada base.

Para garantir a eficiência na localização de conteúdos levando em consideração a integração de várias bases de dados, é fortemente recomendável a adoção do DOI (Digital Object Identifier) (DOI, 2010) como propriedade de identificação de cada conteúdo de mídia. Visto que, com isso, seria possível identificar um determinado conteúdo quando ele encontra-se presente em várias bases distintas. No caso, todas as referências apontariam para um mesmo identificador DOI.

7.2 Cenários de aplicação do presente trabalho

Quaisquer ambientes que sejam compostos por diversas fontes heterogêneas podem utilizar o Dynamo Framework como solução de apoio à integração. Como ideia de aplicação prática da solução apresentada aqui, imagina-se uma possível participação em projetos de gestão de ativos digitais em redes de TV. Como esses projetos visam uma maior disseminação de conteúdos, com informações distribuídas entre várias emissoras diferentes, pode-se inferir a importância de técnicas que possibilitem a integração de dados

de diferentes redes de TV, incluindo a sua apresentação aos usuários finais em uma visão única e integrada.

Outra possibilidade é pensar na ideia de integração entre ambientes aplicados à integração entre ontologias. O objetivo seria, dentro de um mesmo domínio do conhecimento, uniformizar os conceitos semanticamente iguais, independente das peculiaridades de cada ontologia individual. Esse processo não seria feito manualmente, de comum acordo com os construtores de cada ontologia, mas automaticamente à medida das necessidades em cada domínio do conhecimento.

7.3 Perspectiva para trabalhos futuros

Além dos trabalhos complementares citados, que devem abordar questões ainda pendentes relacionadas a interoperabilidade de metadados em redes heterogêneas, existem ainda ideias para trabalhos futuros, mais abrangentes, que tomariam como ponto de partida o que foi desenvolvido aqui.

São deixadas como sugestões de trabalhos futuros a implementação de um sistema de correspondência integrado a uma base léxica que identifique sinônimos de termos em outros idiomas, visando expandir o alcance do mecanismo; a definição e o controle das políticas de integração entre redes distintas; e o gerenciamento de contratos digitais envolvendo as instituições integradas.

Adicionalmente, sugere-se o aprofundamento e a integração de conceitos envolvendo gerenciamento de bancos de dados, lógica, ontologias e inteligência artificial, inseridas nos âmbito da Ciência da Computação e Ciência da Informação, possibilitando a projetos de interoperabilidade a capacidade de incrementar estratégias de gestão das informações e conhecimento, de forma que se possibilite a aquisição de conhecimento, a organização e, em seguida a extensão, no sentido de integrar, e, finalmente, de disponibilizar um ambiente de integração colaborativo.

Bibliografia

- ADUNA B. (2011), System documentation for Sesame 2.x. Disponível em <<http://www.openrdf.org/doc/sesame2/system>>
- AHMAD, K; JOCHEN, K. (2008). IBM Rational Unified Process Reference and Certification Guide: Solution Designer (RUP).
- AMBLER, S. (2003). Agile Database Techniques: Effective Strategies for the Agile Software Developer. Wiley, New York.
- AMIR, S.; BILASCO I. M.; DANISMAN T.; SAYAD I. E.; DJERABA C. (2010). Multimedia metadata mapping: Towards helping developers in their integration task. MoMM, páginas 13–19.
- AMIR, S.; BENNABAS Y.; BILASCO I. M.; DJERABA C. (2011). MuMle: A New System for Multimedia Metadata Interoperability, ACM ICMR (MIR).
- APACHE. Jena Project. (2012). Disponível em <<http://jena.apache.org/documentation>>
- AUSTERBERRY. D. (2006). Digital Asset Management. 2. ed. Oxford: Elsevier. 352p.
- ATAY, M., CHEBOTKO, A., LIU, D., LU, S., AND FOTOUHI, F. (2007). Efficient schema-based XML to-relational data mapping. Inform. Syst. 32, 3, 458–476.
- BAKER, T.; BLANCHI, C.; BRICKLEY, D.; DUVAL, E.; HEERY, R.; JOHNSTON, P.; KALINICHENKO, L.; NEUROTH, H.; SUGIMOTO, S. (2002). Principles of metadata registries. DELOS Network of Excellence on Digital Libraries.
- BECKETT, D. (2001). The Design and Implementation of the Redland RDF Application Framework. WWW10 Conference.
- BENITEZ, A. B.; ZHONG, D.; CHANG, S.-F.; AND SMITH, J. R. (2001). MPEG-7 MDS content description tools and applications. 9th International Conference on Computer Analysis of Images and Patterns (CAIP'01). Springer, Berlin, 2001, 41–52.
- BERNAUER, M., KAPPEL, G., AND KRAMLER, G. (2004). Representing XML schema in UML—A comparison of approaches. In Web Engineering, N. Koch et al. eds., Lecture Notes in Computer Sciences, vol. 3140. Springer, Berlin, 440–444.

- BERNERS-LEE, T. Notation 3. (1998). Design note. World Wide Web Consortium (W3C). Disponível em: <<http://www.w3.org/DesignIssues/Notation3>>. Acesso em Abril 2011.
- BILENKO, M., MOONEY, R. J., COHEN, W. W., P. RAVIKUMAR, D., e FIENBERG, S. E. (2003). "Adaptive name matching in information integration," IEEE Intelligent Systems, vol. 18, no. 5, pp. 16–23.
- BIZER, C. AND SEABORNE, A. (2004). D2RQ—Treating non-RDF databases as virtual RDF graphs. In Proceedings of the 3rd International Semantic Web Conference (ISWC2004). <http://www.wiwiss.fu-berlin.de/suhl/bizer/D2RQ/>.
- BLANKEN, H. (2007). Multimedia Retrieval. 1. ed. Nova Iorque: Springer.
- BOWMAN, C. M. et al. (2011). The Harvest Information Discovery and Access System. Disponível em: <<http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.21.1842>>. Acesso em: Junho 2011.
- BRACHA, G. AND UNGAR, D. (2004). Mirrors: Design principles for meta-level facilities of object-oriented programming languages. In Proceedings of the 19th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA'04), ACM, New York, 331–344.
- CALVANESE, D.; GIACOMO, G.; LENZERINI, M. (2001). Description logics for information integration. In Computational Logic: From Logic Programming into the Future, Lecture Notes in Computer Science. Springer-Verlag.
- CHANG, S.-F.; SIKORA, T.; PURL, A. (2001). Overview of MPEG-7 Standard. IEEE Transactions on Circuits and Systems for Video Technology, v. 11, n. 6, p. 688-695, Junho 2001.
- CHELLA, M. T. (2004). Sistema para classificação e recuperação de conteúdo multimídia baseado no padrão MPEG-7. UNICAMP: São Paulo, 2004. Disponível em: <<http://www.nied.unicamp.br/~siros/doc/2232.pdf>>
- CHEN, P. P.-S. (1976). The entity-relationship model—toward a unified view of data. ACM Trans. Datab.Syst. 1, 1, 9–36. 1976.
- CORAZZON, R. (2002). What is ontology? Disponível em: <http://www.formalontology.it/section_4.htm>.

- CORCHO, O.; FERNÁNDEZ-LÓPEZ, M.; GOMEZ-PÉREZ, A. (2001). Ontoweb, Technical Roadmap V 1.3. Disponível em:
<<http://babage.dia.fi.upm.es/ontoweb/wp1/OntoRoadMap/documents/OntoWeb-D1.3.pdf>>. Acesso em: Maio 2011.
- DUBLIN CORE. DMCI: Dublin Core Metadata Initiative. (2010). Disponível em:
<<http://www.dublincore.org/documents/dcmi-terms>>
- FONG, J. (1997). Converting relational to object-oriented databases. SIGMOD Rec. 26, 1, 53–58.
- GAMBLE, R.; STIGER, P.; PLANT, R. (1999). Rule-Based systems Formalized within a Software Architectural Style, Journal of Knowledge Based Systems, forthcoming.
- GRUBER, T. (1996). What is an Ontology? Disponível em: <<http://www-ksl.stanford.edu/kst/what-is-an-ontology.html>>. Acesso em: 14 set. 2002.
- GASEVIC, D., DJURIC, D., DEVEDZIC, V., AND DAMJANOVI, V. (2004). Converting UML to OWL ontologies. In Proceedings of the 13th International World Wide Web Conference on Alternate Track Papers and Posters (WWW Alt.'04). ACM, New York, 488–489.
- GIUNCHIGLIA, F.; SHVAIKO, P.; YATSKEVICH, M. (2004). S-match: an algorithm and an implementation of semantic matching. In ESWS, pages 61–75.
- GOSLING, J., JOY, B., STEELE, G., AND BRACHA, G. (2005). The Java Language Specification (3rd Ed.). Addison-Wesley, Reading, MA.
- GUARINO, N.; GIARETTA, P. (1995). Ontologies and KBs, towards a terminological clarification. Disponível em:
<<http://www.ladseb.pd.cnr.it/infor/Ontology/Papers/KBKS95.pdf>>.
- HASLHOFER, B.; KLAS, W. (2010). A survey of techniques for achieving metadata interoperability. ACM Comput. Surv., 42(2), 2010.
- HAUSENBLAS M. (2010). Multimedia vocabularies on the semantic web, jul 2005.
- HAUSENBLAS M. (2010). Multimedia vocabularies on the semantic web. DOI. DOI System.
- HUNTER, J.; LAGOZE, C. (2011). Combining RDF and XML schemas to enhance interoperability between metadata application profiles. In Proceedings of the 10th

- International Conference on the World Wide Web (WWW'01), ACM, New York, 2011, 457–466.
- ISO/IEC JTC 1/SC 29. (2007). MPEG-7 multimedia content description interface. ISO 15938-1-11:2002-2007. International Standardization Organization (ISO).
- ISO/IEC JTC 1/SC 32. (2003). SQL - ISO/IEC 9075-1:2003. International Standardization Organization (ISO).
- JAVA COMMUNITY PROCESS. (2006). JSR 269: Pluggable annotation processing API. <http://jcp.org/en/jsr/detail?id=269>.
- JOHNSTON, P. (2004). Minerva—Technical guidelines for digital cultural content creation programs. Tech.Rep., UKOLN, University of Bath.
- KAPELA, R; et al. (2010). Real-time visual content description system based on MPEG-7 descriptors. Multimedia Tools Appl, mar. 2010. Disponível em: <http://www.springerlink.com/content/0x6j6768l7332463/fulltext.pdf>.
- KESHAV, R. et al. Architecture Integration Elements. (1999). Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.24.80> Acesso em: Maio 2011.
- LAGOZE, Carl; Herbert Van de Sompel, Michael Nelson and Simeon Warner. (2002). "OAI-PMH Implementation Guidelines: Specification and XML Schema for the OAI Identifier Format". Disponível em: <http://www.openarchives.org/OAI/2.0/guidelines-oai-identifier.htm>.
- LEE, D., MANI, M., AND CHU, W. W. (2003). Schema conversion methods between xml and relational models. In Knowledge Transformation for the Semantic Web, IOS. Press, 1–17.
- LETHI, P. AND FRANKHAUSER, P. (2004). XML data integration with OWL: Experiences and Challenges. In Proceedings of the Symposium on Applications and the Internet (SAINT'04). IEEE, Los Alamitos, CA, 160–170.
- MADHAVAN, J.; BERNSTEIN, P. A.; RAHM, E. (2001). Generic schema matching with cupid. In VLDB, pages 49–58, 2001.
- MCPARLAND, A. (2002). TV-Anytime: Using all that extra data. Tech. rep., BBC. Disponível em: <http://www.bbc.co.uk/rd/pubs/whp/whp-pdf-files/WHP050.pdf>.

- MELNIK, S.; GARCIA-MOLINA, H.; RAHM, E. (2002). Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In ICDE, pages 117–128, 2002.
- MENA, E. et al. (2006). OBSERVER: An Approach for Query Processing in Global Information Systems based on Interoperation across Pre-existing Ontologies. Disponível em: <<http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.101.8349>>. Acesso em: Junho 2011.
- MILETIC I.; VUJASINOVIC M.; IVEZIC N.; MARJANOVIC Z. (2007). Enabling semantic mediation for business applications: Xml-rdf, rdf-xml and xsd-rdfs transformations. IESA, páginas 483–494, 2007.
- MOTIK, B., HORROCKS, I., AND SATTLER, U. (2007). Bridging the gap between OWL and relational databases. In Proceedings of the 16th International Conference on the World Wide Web (WWW'07). ACM, New York, 807–816.
- NISO. (2004). Understanding metadata. National Information Standards Organization (NISO). Disponível em: <<http://www.niso.org/standards/resources/UnderstandingMetadata.pdf>>. Acesso em: Abril 2011.
- OCPSOFT. (2009). PrettyFaces Rewrite. Disponível em <<http://ocpssoft.org/prettyfaces/docs>>.
- OMG. (2006a). Meta Object Facility (MOF) core specification—Version 2.0. Object Management Group (OMG). Disponível em: <<http://www.omg.org/cgi-bin/apps/doc?formal/06-01-01.pdf>>.
- OMG. (2006b). Ontology definition metamodel specification (ODM). Object Management Group (OMG). Disponível em: <<http://www.omg.org/docs/ptc/06-10-11.pdf>>.
- OMG. (2006c). UML 2.0: Infrastructure specification. Object Management Group (OMG). Disponível em: <<http://www.omg.org/docs/ptc/03-09-15.pdf>>.
- OMG. (2007a). MOF 2.0 / XMI Mapping Specification, V2.1.1. Object Management Group (OMG).
- OMG. (2007b). Unified Modelling Language (UML). Object Management Group (OMG). Disponível em: <<http://www.uml.org/>>.

- ORACLE. (2012). O que é Java e por que é necessária. Disponível em:
<http://www.java.com/pt_BR/download/faq/whatis_java.xml>
- OUKSEL, A. M.; SHETH, A. (1999). Semantic interoperability in global information systems. SIGMOD Rec. 28, 1, 5–12, 1999.
- PACHECO, R. C. S.; KERN, V. M. (2001). Uma ontologia comum para a integração de bases de informações e conhecimento sobre ciência e tecnologia. Disponível em:
<<http://revista.ibict.br/index.php/ciinf/article/viewArticle/197> >. Acesso em: Maio 2011.
- PBCORE. (2011). Public Broadcasting Metadata Dictionary Project 2.0. Disponível em
<<http://pbcore.org/documentation>>
- RAHM, E.; BERNSTEIN, P. A. (2001). A survey of approaches to automatic schema matching. VLDB J. 10, 4, 334–350, 2001.
- SAYÃO, L. F. (2001). Modelos teóricos em Ciência da Informação – abstração e método científico. Disponível em:
<<http://revista.ibict.br/index.php/ciinf/article/viewArticle/228>>. Acesso em: Junho 2011.
- SEIDWITZ, E. (2003). What models mean. IEEE Softw. 20, 5, 26–32, 2003.
- SHETH, A.; KLAS, W. (1998) Multimedia Data Management: Using Metadata to Integrate and Apply Digital Media. McGraw-Hill, New York, 1998.
- SOAP. (2007). Simple Object Access Protocol, World Wide Web Consortium. Disponível em: <<http://www.w3.org/TR/2007/REC-soap12-part0-20070427>>
- SOWA, J. F. (1999). Building, sharing and merging ontologies. Disponível em:
<<http://www.jfsowa.com/ontology/ontoshar.htm>>.
- STALLINGS, W. (2002). High-Speed Networks and Internets: Performance and Quality of Service. 2. ed. New Jersey: Prentice Hall. 2002.
- STUCKENSCHMIDT, H.; WACHE, H. (2000) Context modeling and transformation for semantic interoperability. Disponível em:
<<http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.38.625>>.

- VICENTE, P. J. V. (2005). El estándar MPEG-7. Revista de Ingeniería Informática del CIIRM, Murcia (Espanha), n.3, p. 1-5, 2005. Disponível em: <http://www.cii-murcia.es/informas/jul05/articulos/El_estandar_MPEG-7.pdf>
- WACHE, H. et al. (2001). Ontology-based integration of information – a survey of existing approaches. Disponível em: <<http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.12.8073>>. Acesso em: Junho 2011.
- WESTERMANN, U.; KLAS, W. (2003). An analysis of XML database solutions for the management of MPEG-7 media descriptions. ACM Comput. Surv. 35, 4, 2003, 331–373.
- W3C. (2004a). RDF vocabulary description language 1.0: RDF schema. W3C semantic web activity. RDF Core Working Group. Disponível em: <<http://www.w3.org/TR/rdf-schema/>>.
- W3C. (2004b). Web Ontology Language (OWL). W3C semantic web activity. Web Ontology Working Group. Disponível em: <<http://www.w3.org/2004/OWL/>>.
- W3C. (2006). XML schema 1.1 Part 1: Structure. W3C XML Core Working Group. Disponível em: <<http://www.w3.org/TR/xmlschema11-1/>>.
- W3C. (2008). SPARQL Query Language for RDF. RDF Data Access Working Group. Disponível em: <<http://www.w3.org/TR/rdf-sparql-query/>>.
- WIEDERHOLD, G. (1999). Mediation to deal with heterogeneous data sources. Disponível em: <<http://www-db.stanford.edu/pub/gio/1999/Interopdocfigs.html>>.
- WORDNET. (2012). <<http://wordnet.princeton.edu>>, acessado em 10-jun-2012.
- TOLK, A. (2006). What comes after the semantic web: PADS implications for the dynamic web. In Proceedings of the 20th Workshop on Principles of Advanced and Distributed Simulation (PADS'06), IEEE, Los Alamitos, CA, 55, 2006.

Anexos

Anexo A – RDF/RDFS

O intercâmbio entre aplicações requer uniformidade sobre semântica, sintaxe e estrutura. As comunidades definem diferentes semânticas para seus metadados, de acordo com suas necessidades. A sintaxe, que consiste do arranjo sistemático de elementos de dados, facilita o intercâmbio e o uso de metadados entre diversas aplicações. Sobre a estrutura, pode-se imaginar uma limitação formal sobre a sintaxe, que proporcione uma representação consistente da semântica. Os metadados, ou são parte do próprio dado, ou estão separados dele, ou podem ser dados sobre outros metadados, configurando o que se chama de meta-meta-dados.

Dentre as novas propostas está o RDF, uma infraestrutura que possibilita a codificação, o intercâmbio e a reutilização de metadados. Trata-se de uma aplicação XML que impõe limitações estruturais à expressão semântica. Possibilita a publicação de vocabulários legíveis para pessoas e computadores, além de proporcionar a reutilização e a distribuição dos metadados .

Como infra-estrutura, o RDF permite a interoperabilidade de metadados e permite a criação de mecanismos para convenções comuns de semântica, sintaxe e estrutura. O RDF utiliza o XML como sintaxe o que proporciona independência de plataforma, é legível para pessoas e possibilita representação de estruturas complexas.

O modelo de dados RDF proporciona uma forma para a descrição de “recursos”, os quais têm “propriedades” (atributos ou características). No contexto do RDF, um “recurso” é qualquer objeto identificável de forma única por uma URI (pode ser um conjunto de páginas da Web). As propriedades associadas aos recursos são chamadas “propriedades-tipo” e expressam os relacionamentos dos “valores” associados aos recursos. Os valores podem ser “atômicos” (strings, números, etc) ou outros recursos, os quais, por sua vez, podem ter suas propriedades. Às propriedades do recurso são chamadas de “descrição”.

Com a crescente necessidade de metadados para integração digital, o RDF não pode atender mais a todas expectativas em relação à semântica. Segundo DODDS et al.(2001), o RDF em conjunto com XML será suficiente para obter a representação desejada em muitos tipos de aplicações, mas , em alguns casos, será necessário maior poder de expressão.

Para estender o uso do RDF em situações que demandam maior número de metadados para a identificação dos dados, desenvolveu-se o RDFS (RDF Schema). A

especificação RDFS é escrita em RDF e permite modelagem de um maior número de primitivas, proporciona informações sobre a interpretação das declarações do modelo RDF, permite verificar a integridade dos dados e estabelecer uma hierarquia de tipos, tais como classes, subclasses, relacionamentos, restrições, dentre outras. Pode-se necessitar de mais informações sobre um recurso, como por exemplo, o que ele representa: no caso de uma propriedade que representa um autor, pode ser necessário que o valor associado faça referência a uma pessoa, e não a um carro ou a uma casa; se uma propriedade representa um aniversário, é preciso verificar que o valor é uma data, e não um número ou um tipo de animal (DODDS et al., 2001, p. 167).

O predicado de uma afirmação é identificado por uma URI e os objetos por strings. Além de efetuar a validação, o RDFS pode aplicar certas restrições ao uso de propriedades. O RDFS possibilita verificar se fazem sentido as propriedades aplicadas a um recurso e os valores atribuídos a estas propriedades.

Para maiores detalhes, consultar a especificação completa do RDFS, que contém todos os conceitos, expressões e operadores válidos⁹. O esquema do RDFS definido em RDF também pode ser consultado¹⁰.

9 Disponível em: <<http://www.w3.org/TR/2000/CR-rdf-schema-20000327>>.

10 Disponível em: <<http://www.w3.org/2000/01/rdf-schema>>.

Anexo B – Sintaxe do meta-modelo dinâmico numa representação em RDFS

Apresenta-se em seguida a descrição dos elementos dos meta-modelos construídos a partir do estudo de caso descrito no Capítulo 6. Os sinais “#” que aparecem ao lado do nome dos representam diferentes espaços de nomes (namespaces), ou seja, diferentes URIs-Uniform Resource Identifier para um recurso. Os namespaces estão definidos como #1 para a raiz, < http://ci.ufpb.br/Dynamo/metamodel/> (meta-modelo), #2 para < http://ci.ufpb.br/Dynamo/metamodel/MG-[base1-base2] > (modelo genérico), #3 para <http://ci.ufpb.br/Dynamo/metamodel/MG-[base1-base2]/MF> (propriedade do modelo fonte considerado em uma integração). Com a descrição podem-se compreender as relações entre as classes dos modelos.

Na sintaxe abaixo estão descritas as relações entre conceitos do meta-modelo em RDFS (RDF Schema). Considera-se que MG é o modelo-genérico, construído com base no mapeamento a partir de combinações entre os três esquemas das bases das instituições consideradas: BBC (TV-Anytime), TV Brasil (PBCore) e Cinemateca. Apresenta-se a seguir a descrição das classes referenciando as propriedades, juntamente dos seus respectivos namespaces.

a) BBC (TV-Anytime) e TV Brasil (PBCore)

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns="http://ci.ufpb.br/Dynamo/metamodel/MG-BBC-TVBRASIL#"
  xml:base="http://ci.ufpb.br/Dynamo/metamodel/MG-BBC-TVBRASIL"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:swrl="http://www.w3.org/2003/11/swrl#"
  xmlns:swrlx="http://www.w3.org/2003/11/swrlx#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:owl2="http://www.w3.org/2006/12/owl2#">
  <owl:Ontology rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-BBC-TVBRASIL"/>
    <owl:Class rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-BBC-TVBRASIL#BasicDescription">
      <rdfs:subClassOf rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-BBC-TVBRASIL#Content"/>
    </owl:Class>
    <owl:Class rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-BBC-TVBRASIL#BasicDescription.Duration">
      <rdfs:subClassOf rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-BBC-TVBRASIL#BasicDescription"/>
    </owl:Class>
    <owl:Class rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-BBC-
```

```

TVBRASIL#BasicDescription.Language">
  <rdfs:subClassOf rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-
BBC-TVBRASIL#BasicDescription"/>
</owl:Class>
<owl:Class rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-BBC-
TVBRASIL#BasicDescription.ProductionDate">
  <rdfs:subClassOf rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-
BBC-TVBRASIL#BasicDescription"/>
</owl:Class>
<owl:Class rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-BBC-
TVBRASIL#BasicDescription.ProductionLocation">
  <rdfs:subClassOf rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-
BBC-TVBRASIL#BasicDescription"/>
</owl:Class>
<owl:Class rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-BBC-
TVBRASIL#Content">
  <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
</owl:Class>
<owl:Class rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-BBC-
TVBRASIL#ContentReference">
  <rdfs:subClassOf rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-
BBC-TVBRASIL#Content"/>
</owl:Class>
<owl:Class rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-BBC-
TVBRASIL#ContentReference.CRID">
  <rdfs:subClassOf rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-
BBC-TVBRASIL#ContentReference"/>
</owl:Class>
<owl:Class rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-BBC-
TVBRASIL#ContentReference.authority">
  <rdfs:subClassOf rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-
BBC-TVBRASIL#ContentReference"/>
</owl:Class>
<owl:Class rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-BBC-
TVBRASIL#ContentReference.authorityRole">
  <rdfs:subClassOf rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-
BBC-TVBRASIL#ContentReference"/>
</owl:Class>
<owl:Class rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-BBC-
TVBRASIL#Genre">
  <rdfs:subClassOf rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-
BBC-TVBRASIL#Content"/>
</owl:Class>
<owl:Class rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-BBC-
TVBRASIL#Genre.value">
  <rdfs:subClassOf rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-
BBC-TVBRASIL#Genre"/>
</owl:Class>
<owl:Class rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-BBC-
TVBRASIL#Instantiation.instantiationDate">
  <rdfs:subClassOf rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-
BBC-TVBRASIL#BasicDescription.ProductionDate"/>
</owl:Class>
<owl:Class rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-BBC-
TVBRASIL#Instantiation.instantiationDuration">
  <rdfs:subClassOf rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-
BBC-TVBRASIL#BasicDescription.Duration"/>
</owl:Class>
<owl:Class rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-BBC-
TVBRASIL#Instantiation.instantiationIdentifier">
  <rdfs:subClassOf rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-

```

```

BBC-TVBRASIL#ContentReference.CRID"/>
  </owl:Class>
  <owl:Class rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-BBC-
TVBRASIL#Instantiation.instantiationLanguage">
    <rdfs:subClassOf rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-
BBC-TVBRASIL#BasicDescription.Language"/>
    <rdfs:subClassOf rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-
BBC-TVBRASIL#Synopsis.language"/>
  </owl:Class>
  <owl:Class rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-BBC-
TVBRASIL#Instantiation.instantiationLocation">
    <rdfs:subClassOf rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-
BBC-TVBRASIL#BasicDescription.ProductionLocation"/>
  </owl:Class>
  <owl:Class rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-BBC-
TVBRASIL#IntellectualContent.pbcoreGenre">
    <rdfs:subClassOf rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-
BBC-TVBRASIL#Genre.value"/>
  </owl:Class>
  <owl:Class rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-BBC-
TVBRASIL#IntellectualProperty.pbcoreDescription">
    <rdfs:subClassOf rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-
BBC-TVBRASIL#Synopsis.text"/>
  </owl:Class>
  <owl:Class rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-BBC-
TVBRASIL#IntellectualProperty.pbcoreIdentifier">
    <rdfs:subClassOf rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-
BBC-TVBRASIL#ContentReference.CRID"/>
  </owl:Class>
  <owl:Class rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-BBC-
TVBRASIL#IntellectualProperty.pbcoreRelationCreator">
    <rdfs:subClassOf rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-
BBC-TVBRASIL#ContentReference.authority"/>
  </owl:Class>
  <owl:Class rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-BBC-
TVBRASIL#IntellectualProperty.pbcoreRelationCreatorRole">
    <rdfs:subClassOf rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-
BBC-TVBRASIL#ContentReference.authorityRole"/>
  </owl:Class>
  <owl:Class rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-BBC-
TVBRASIL#IntellectualProperty.pbcoreTitle">
    <rdfs:subClassOf rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-
BBC-TVBRASIL#Title.text"/>
  </owl:Class>
  <owl:Class rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-BBC-
TVBRASIL#Synopsis">
    <rdfs:subClassOf rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-
BBC-TVBRASIL#Content"/>
  </owl:Class>
  <owl:Class rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-BBC-
TVBRASIL#Synopsis.language">
    <rdfs:subClassOf rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-
BBC-TVBRASIL#Synopsis"/>
  </owl:Class>
  <owl:Class rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-BBC-
TVBRASIL#Synopsis.text">
    <rdfs:subClassOf rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-
BBC-TVBRASIL#Synopsis"/>
  </owl:Class>
  <owl:Class rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-BBC-
TVBRASIL#Title">

```

```

        <rdfs:subClassOf rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-
BBC-TVBRASIL#Content"/>
    </owl:Class>
    <owl:Class rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-BBC-
TVBRASIL#Title.text">
        <rdfs:subClassOf rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-
BBC-TVBRASIL#Title"/>
    </owl:Class>
</rdf:RDF>

```

b) Cinemateca e BBC (TV-Anytime)

```

<?xml version="1.0"?>
<rdf:RDF xmlns="http://ci.ufpb.br/Dynamo/metamodel/MG-Cinemateca-BBC#"
  xml:base="http://ci.ufpb.br/Dynamo/metamodel/MG-Cinemateca-BBC"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:swrl="http://www.w3.org/2003/11/swrl#"
  xmlns:swrlx="http://www.w3.org/2003/11/swrlx#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:owl2="http://www.w3.org/2006/12/owl2#">
  <owl:Ontology rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-Cinemateca-
BBC"/>
    <owl:Class rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-Cinemateca-
BBC#BasicDescription.creationCoordinates">
        <rdfs:subClassOf rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-
Cinemateca-BBC#GeographicTerms"/>
    </owl:Class>
    <owl:Class rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-Cinemateca-
BBC#BasicDescription.creditsList">
        <rdfs:subClassOf rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-
Cinemateca-BBC#Director"/>
    </owl:Class>
    <owl:Class rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-Cinemateca-
BBC#BasicDescription.duration">
        <rdfs:subClassOf rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-
Cinemateca-BBC#Duration"/>
    </owl:Class>
    <owl:Class rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-Cinemateca-
BBC#BasicDescription.productLocation">
        <rdfs:subClassOf rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-
Cinemateca-BBC#City"/>
        <rdfs:subClassOf rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-
Cinemateca-BBC#State"/>
    </owl:Class>
    <owl:Class rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-Cinemateca-
BBC#BasicDescription.productionDate">
        <rdfs:subClassOf rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-
Cinemateca-BBC#ProductionYear"/>
    </owl:Class>
    <owl:Class rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-Cinemateca-
BBC#BasicDescription.productionLocation">
        <rdfs:subClassOf rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-
Cinemateca-BBC#GeographicTerms"/>
    </owl:Class>
    <owl:Class rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-Cinemateca-
BBC#City">

```

```

        <rdfs:subClassOf rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-
Cinematca-BBC#MovieDescription"/>
        <additionalFunction rdf:datatype="http://ci.ufpb.br/Dynamo/metamodel/MG-
Cinematca-BBC#text">split("&quot;;&quot;") [3]</additionalFunction>
    </owl:Class>
    <owl:Class rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-Cinematca-
BBC#Code">
        <rdfs:subClassOf rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-
Cinematca-BBC#MovieDescription"/>
    </owl:Class>
    <owl:Class rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-Cinematca-
BBC#ContentReference.CRID">
        <rdfs:subClassOf rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-
Cinematca-BBC#Code"/>
    </owl:Class>
    <owl:Class rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-Cinematca-
BBC#Director">
        <rdfs:subClassOf rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-
Cinematca-BBC#MovieDescription"/>
        <additionalFunction rdf:datatype="http://ci.ufpb.br/Dynamo/metamodel/MG-
Cinematca-BBC#text">get("&quot;director&quot;")</additionalFunction>
    </owl:Class>
    <owl:Class rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-Cinematca-
BBC#Duration">
        <rdfs:subClassOf rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-
Cinematca-BBC#MovieDescription"/>
    </owl:Class>
    <owl:Class rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-Cinematca-
BBC#Genre">
        <rdfs:subClassOf rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-
Cinematca-BBC#MovieDescription"/>
    </owl:Class>
    <owl:Class rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-Cinematca-
BBC#Genre.value">
        <rdfs:subClassOf rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-
Cinematca-BBC#Genre"/>
    </owl:Class>
    <owl:Class rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-Cinematca-
BBC#GeographicTerms">
        <rdfs:subClassOf rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-
Cinematca-BBC#MovieDescription"/>
    </owl:Class>
    <owl:Class rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-Cinematca-
BBC#GivenTitle">
        <rdfs:subClassOf rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-
Cinematca-BBC#MovieDescription"/>
    </owl:Class>
    <owl:Class rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-Cinematca-
BBC#MovieDescription"/>
    <owl:Class rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-Cinematca-
BBC#Producer">
        <rdfs:subClassOf rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-
Cinematca-BBC#MovieDescription"/>
        <additionalFunction rdf:datatype="http://ci.ufpb.br/Dynamo/metamodel/MG-
Cinematca-BBC#text">get("&quot;producer&quot;")</additionalFunction>
    </owl:Class>
    <owl:Class rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-Cinematca-
BBC#ProductionYear">
        <rdfs:subClassOf rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-
Cinematca-BBC#MovieDescription"/>
        <additionalFunction

```

```

rdf:datatype="&rdf;text">split("&quot;/&quot;")[0]</additionalFunction>
</owl:Class>
<owl:Class rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-Cinematca-
BBC#State">
  <rdfs:subClassOf rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-
Cinematca-BBC#MovieDescription"/>
  <additionalFunction rdf:datatype="http://ci.ufpb.br/Dynamo/metamodel/MG-
Cinematca-BBC#text">split("&quot;;&quot;")[2]</additionalFunction>
</owl:Class>
<owl:Class rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-Cinematca-
BBC#Synopsis">
  <rdfs:subClassOf rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-
Cinematca-BBC#MovieDescription"/>
</owl:Class>
<owl:Class rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-Cinematca-
BBC#Synopsis.text">
  <rdfs:subClassOf rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-
Cinematca-BBC#Synopsis"/>
</owl:Class>
<owl:Class rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-Cinematca-
BBC#Title">
  <rdfs:subClassOf rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-
Cinematca-BBC#MovieDescription"/>
</owl:Class>
<owl:Class rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-Cinematca-
BBC#Title.text">
  <rdfs:subClassOf rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-
Cinematca-BBC#GivenTitle"/>
  <rdfs:subClassOf rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-
Cinematca-BBC#Title"/>
</owl:Class>
</rdf:RDF>

```

c) Cinematca e TV Brasil (PBCore)

```

<?xml version="1.0"?>
<rdf:RDF xmlns="http://ci.ufpb.br/Dynamo/metamodel/MG-Cinematca-TVBrasil#"
  xml:base="http://ci.ufpb.br/Dynamo/metamodel/MG-Cinematca-TVBrasil"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:swrl="http://www.w3.org/2003/11/swrl#"
  xmlns:swrlx="http://www.w3.org/2003/11/swrlx#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:owl2="http://www.w3.org/2006/12/owl2#">
  <owl:Ontology rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-Cinematca-
TVBrasil"/>
  <owl:Class rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-Cinematca-
TVBrasil#City">
    <rdfs:subClassOf rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-
Cinematca-TVBrasil#MovieDescription"/>
    <additionalFunction rdf:datatype="http://ci.ufpb.br/Dynamo/metamodel/MG-
Cinematca-TVBrasil#text">split("&quot;;&quot;")[2]</additionalFunction>
  </owl:Class>
  <owl:Class rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-Cinematca-
TVBrasil#Code">
    <rdfs:subClassOf rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-
Cinematca-TVBrasil#MovieDescription"/>
  </owl:Class>
  <owl:Class rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-Cinematca-

```

```

TVBrasil#Director">
    <rdfs:subClassOf      rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-
Cinematca-TVBrasil#MovieDescription"/>
    </owl:Class>
    <owl:Class      rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-Cinematca-
TVBrasil#Duration">
        <rdfs:subClassOf      rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-
Cinematca-TVBrasil#MovieDescription"/>
        </owl:Class>
        <owl:Class      rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-Cinematca-
TVBrasil#Genre">
            <rdfs:subClassOf      rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-
Cinematca-TVBrasil#MovieDescription"/>
            </owl:Class>
            <owl:Class      rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-Cinematca-
TVBrasil#GeographicTerms">
                <rdfs:subClassOf      rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-
Cinematca-TVBrasil#MovieDescription"/>
                </owl:Class>
                <owl:Class      rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-Cinematca-
TVBrasil#GivenTitle">
                    <rdfs:subClassOf      rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-
Cinematca-TVBrasil#MovieDescription"/>
                    </owl:Class>
                    <owl:Class      rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-Cinematca-
TVBrasil#Instantiation.instantiationDate">
                        <rdfs:subClassOf      rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-
Cinematca-TVBrasil#ProductionYear"/>
                        </owl:Class>
                        <owl:Class      rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-Cinematca-
TVBrasil#Instantiation.instantiationDuration">
                            <rdfs:subClassOf      rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-
Cinematca-TVBrasil#Duration"/>
                            </owl:Class>
                            <owl:Class      rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-Cinematca-
TVBrasil#Instantiation.instantiationLocation">
                                <rdfs:subClassOf      rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-
Cinematca-TVBrasil#City"/>
                                <rdfs:subClassOf      rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-
Cinematca-TVBrasil#GeographicTerms"/>
                                <rdfs:subClassOf      rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-
Cinematca-TVBrasil#State"/>
                                </owl:Class>
                                <owl:Class      rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-Cinematca-
TVBrasil#Instantiation.instantiationIdentifier">
                                    <rdfs:subClassOf      rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-
Cinematca-TVBrasil#Code"/>
                                    </owl:Class>
                                    <owl:Class      rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-Cinematca-
TVBrasil#IntellectualContent.pbcoreDescription">
                                        <rdfs:subClassOf      rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-
Cinematca-TVBrasil#Synopsis"/>
                                        </owl:Class>
                                        <owl:Class      rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-Cinematca-
TVBrasil#IntellectualContent.pbcoreGenre">
                                            <rdfs:subClassOf      rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-
Cinematca-TVBrasil#Genre"/>
                                            </owl:Class>
                                            <owl:Class      rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-Cinematca-
TVBrasil#IntellectualContent.pbcoreIdentifier">
                                                <rdfs:subClassOf      rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-

```

```

Cinematca-TVBrasil#Code"/>
</owl:Class>
<owl:Class      rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-Cinematca-
TVBrasil#IntellectualContent.pbcoreTitle">
  <rdfs:subClassOf      rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-
Cinematca-TVBrasil#GivenTitle"/>
  <rdfs:subClassOf      rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-
Cinematca-TVBrasil#Title"/>
</owl:Class>
<owl:Class      rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-Cinematca-
TVBrasil#IntellectualProperty.pbcoreRelationCreator">
  <rdfs:subClassOf      rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-
Cinematca-TVBrasil#Director"/>
</owl:Class>
<owl:Class      rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-Cinematca-
TVBrasil#MovieDescription"/>
<owl:Class      rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-Cinematca-
TVBrasil#ProductionYear">
  <rdfs:subClassOf      rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-
Cinematca-TVBrasil#MovieDescription"/>
  <additionalFunction  rdf:datatype="http://ci.ufpb.br/Dynamo/metamodel/MG-
Cinematca-TVBrasil#text">split(&quot;/&quot;)[0]</additionalFunction>
</owl:Class>
<owl:Class      rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-Cinematca-
TVBrasil#State">
  <rdfs:subClassOf      rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-
Cinematca-TVBrasil#MovieDescription"/>
  <additionalFunction  rdf:datatype="http://ci.ufpb.br/Dynamo/metamodel/MG-
Cinematca-TVBrasil#text">split(&quot;;&quot;)[3]</additionalFunction>
</owl:Class>
<owl:Class      rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-Cinematca-
TVBrasil#Synopsis">
  <rdfs:subClassOf      rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-
Cinematca-TVBrasil#MovieDescription"/>
</owl:Class>
<owl:Class      rdf:about="http://ci.ufpb.br/Dynamo/metamodel/MG-Cinematca-
TVBrasil#Title">
  <rdfs:subClassOf      rdf:resource="http://ci.ufpb.br/Dynamo/metamodel/MG-
Cinematca-TVBrasil#MovieDescription"/>
</owl:Class>
</rdf:RDF>

```