ALCIONE BENACCHIO

DEFINIÇÃO DE UMA ARQUITETURA INTEGRADA DE REPOSITÓRIO DE PADRÕES E METADADOS

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre. Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, Universidade Federal do Paraná.
Orientadora: Profa. Dra. Maria Salete Marcon Gomes Vaz

CURITIBA

2008

ALCIONE BENACCHIO

DEFINIÇÃO DE UMA ARQUITETURA INTEGRADA DE REPOSITÓRIO DE PADRÕES E METADADOS

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre. Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, Universidade Federal do Paraná.
Orientadora: Profa. Dra. Maria Salete Marcon Gomes Vaz

CURITIBA

2008

AGRADECIMENTOS

Dizem que Deus responde às nossas preces à sua própria maneira, não à nossa. Acredito que tenha sido por isso que tudo aconteceu de forma tão perfeita. Durante esse mestrado aprendi muito, conheci muitas pessoas, expandi meus horizontes de maneiras que não poderia ter imaginado, e por isso sou grato a Deus, por me dar o que eu preciso e não exatamente o que eu quero.

Agradeço a minha orientadora e amiga Professora Dra. Maria Salete Marcon Gomes Vaz, por ter acreditado e confiado em mim, pela paciência, acompanhamento e diretrizes que levaram ao sucesso deste trabalho. Minha mais profunda e sincera admiração.

Aos membros da banca examinadora, Professor Dr. José Simão de Paula Pinto, Professor Dr. Marcos Sfair Sunye, por sua revisão precisa e suas críticas que contribuíram para enriquecer este trabalho.

Aos grandes amigos que fiz no mestrado, que sem eles teria sido muito mais difícil a adaptação a esse novo ambiente.

Agradeço as Professoras Msc. Andreia de Jesus e Dra. Deborah Ribeiro Carvalho pelo apoio, incentivo e revisão deste trabalho.

Agradeço a minha família pelo amor incondicional, pela compreensão da ausência e pelo intensivo apoio. E principalmente, agradeço a minha esposa Juliana, por aceitar deixar tudo para trás e vir comigo enfrentar mais esse desafio.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

AXIS	*Apache Extensible Interaction System*
BSD	*Berkeley Software Distribution*
CGI	*Common Gateway Interface*
DCMI	*Dublin Core Metadata Initiative*
EAD	*Encoded Archival Description*
FTP	*File Transfer Protocol*
HTML	*HyperText Markup Language*
HTTP	*HyperText Transfer Protocol*
IEC	*International Electrotechnical Commission*
ISAD(G)	*International Standard Archival Description*
ISO	*International Standardization Organization*
JDK	*Java Development Kit*
JSP	*Java Server Pages*
METS	*Metadata Encoding and Transmission Standard*
MDR	*Metadata Registries*
MER	Modelo Entidade Relacionamento
MOF	*Meta Object Facility*
MPEG	*Movie Picture Experts Group*
OMG	*Object Management Group*
PL/pgSQL	*Procedural Language/PostgreSQL Structured Query Language*
RPC	*Remote Procedure Call*
SGBD	Sistema Gerenciador de Banco de Dados
SGML	*Standard Generalized Markup Language*
SMTP	*Simple Mail Transfer Protocol*
SOAP	*Simple Object Access Protocol*

SQL	*Structured Query Language*
UDDI	*Universal Description, Discovery, and Integration*
UML	*Unified Modeling Language*
URL	*Uniform Resource Locator*
XML	*Extensible Markup Language*
XSL	*Extensible Stylesheet Language*
XSLT	*Extensible Stylesheet Language Transformations*
W3C	*World Wide Web Consortium*
WSDL	*Web Service Description Language*

RESUMO

Iniciar o processo de gestão de dados e metadados de uma organização, localizar e compreender os dados disponíveis e o seu contexto, pode ser um trabalho difícil de ser realizado. Metadados são elementos fundamentais para auxiliar nessa tarefa, facilitando os processos de integração, interpretação, organização e localização de informação. Por isso, as organizações procuram adotar padrões de metadados para tratar cada tipo de informação que gerenciam. A adoção de padrões e o registro correto de informação, de acordo com as especificações de cada padrão, são ótimas práticas, mas resolvem apenas parte do problema na gestão de metadados. Ao utilizar vários padrões distintos, começam a surgir os problemas de integração dos dados entre os padrões. Este trabalho apresenta uma especificação de arquitetura de repositório de padrões e metadados, que possibilita a integração dos dados comuns entre padrões distintos e proporciona um repositório onde os sistemas possam acessar e gerenciar os metadados de vários padrões.

ABSTRACT

Start the data and metadata management process from an organization, locate and understand the available data and its context can be a difficulty task to be performed. Metadata are fundamental elements to help in this case, making easy the integration process, interpretation, organization and localization of information. Hence, the organizations seek to use metadata standards to handle each type of information that they manage. The use of standards and the right registry of information, according to specifications of each standards, are great practices, but solve just part of the problem in metadata management. When it uses several distinct standards start to arise the problems of integration of data among standards. This work presents a specification of architecture of standards and metadata repository, which allow an integration of common data among distinct standards and provide a repository where the systems can access and manage the metadata of several standards.

CAPÍTULO 1

INTRODUÇÃO

1.1 Motivação

As organizações têm acumulado uma grande quantidade de informações relacionadas às suas atividades. O que pode ser observado é que, além da quantidade de informações, a diversidade de tipos de dados utilizados também cresceu, tornando comum o uso de imagens, áudios, vídeos, documentos, páginas de internet, entre outros, para registrar informações.

Ao acumular uma grande variedade de tipos de dados, as organizações começam a ter problemas relacionados à localização, descoberta, avaliação, documentação e seleção desses dados. De acordo com Vaz[38], esse problema de gestão pode ser solucionado através do uso de metadados, dados que descrevem dados.

Os metadados auxiliam seus usuários no processo de gestão da informação. O uso de metadados pelas organizações, para descrever seus dados, provê o suporte necessário para localizar, descobrir, selecionar, documentar e avaliar os dados registrados por elas.

Ao utilizar os metadados como parte da solução, deve-se levar em consideração que todos na organização, ao registrar informações sobre cada tipo de dados, necessitam utilizar o mesmo padrão de metadados, ou seja, todos os usuários que registram informações sobre documentos devem utilizar o mesmo padrão.

A utilização de padrões de projeto, de acordo com Gamma et al[12], procura oferecer soluções para problemas recorrentes no desenvolvimento de sistemas. Por tratarem de técnicas conhecidas, testadas e aprovadas por outros desenvolvedores, as arquiteturas e projetos de sistemas tornam-se mais facilmente reutilizáveis.

Esse conceito de padrões de projeto também pode ser aplicado a padrões de metadados. Atualmente, existem diversos padrões de metadados que já foram especificados por várias organizações, sendo que alguns desses padrões serão abordados ao longo deste

trabalho, contudo, é comum as organizações, que utilizam metadados, desenvolverem seus próprios padrões para atendimento às necessidades específicas.

Além dos benefícios citados na utilização de metadados, aliados aos padrões que são amplamente documentados, os padrões de projeto simplificam a atividade de análise, antecipando situações que podem surgir durante a implementação da solução e auxiliando os desenvolvedores a melhor compreender o problema.

Quando uma organização define os padrões de metadados que irá utilizar e inicia o seu processo de gestão, com o tempo percebe a necessidade de novos padrões e, em muitos casos, por falta de um ambiente integrado, começam os problemas de estruturação, gestão de padrões e integração entre padrões distintos de metadados.

A utilização de um repositório de metadados genérico que possibilita a integração e o acesso independente para manipular os dados e os padrões de metadados se faz necessária. Esse repositório resulta na unificação de todo o processo de gestão de metadados e dos padrões e permite a atribuição de um responsável para avaliar e padronizar os metadados, os quais a organização e seus usuários irão utilizar.

Um fator a destacar é que os beneficiários da utilização de repositórios de metadados nem sempre são pessoas. Além das pessoas que os utilizam, através de sua interface de software, ferramentas, aplicações e até mesmo outros repositórios de metadados podem se beneficiar de suas informações.

A utilização de um repositório de metadados genérico provê aos seus usuários os recursos necessários para descrever quaisquer tipos de dados e proporciona um ambiente no qual a gestão das informações seja feita através dos metadados, proporcionando benefícios para [34]:

- **Localização de Informação**: Reduz o tempo de busca por informação de acordo com o contexto desejado, auxiliando a tomada de decisões.
- **Interpretação de Informação**: Diminui o impacto de avaliação e interpretação da informação, pois descreve os dados que são utilizados.
- **Integração de Dados**: Provê um ambiente, onde os dados estão centralizados. Podem ser utilizados pelos proprietários para análises e ter

uma ampla perspectiva de como essa informação se relaciona e pode ser utilizada.

1.2 Objetivos

Tendo em vista o exposto, este trabalho tem como objetivo geral especificar uma arquitetura de repositório de metadados genérico, que torna possível a integração das funcionalidades de gestão, descrição, reuso e padronização dos dados dentro de uma organização.

Os objetivos específicos são:

1. Definir e implementar um metamodelo genérico para o armazenamento de padrões de metadados.
2. Especificar uma arquitetura de repositório genérico dividida em camadas não acopladas.
3. Desenvolver um ambiente onde os metadados registrados estejam centralizados, com o intuito de integrar todos os padrões utilizados pela organização.
4. Realizar experimentos na arquitetura proposta.

Não faz parte do escopo deste trabalho determinar forma ou tecnologia utilizada para o armazenamento dos padrões e os dados dos metadados.

1.3 Organização

Esta dissertação está estruturada como segue. O **Capítulo 2** apresenta uma revisão bibliográfica sobre a gestão de metadados, abordando suas aplicações, definições gerais sobre metadados, introdução aos padrões de metadados *Dublin Core*, METS e MPEG-7, repositórios de metadados e os problemas comuns no processo de gestão de metadados.

No **Capítulo 3**, é apresentada a Arquitetura Integrada de Repositório de Padrões e Metadados, com sua modelagem, descrição dos atributos das classes, como deve ser o processo de gestão de dados ao utilizar a arquitetura e suas camadas.

O objetivo do **Capítulo 4** é abordar as tecnologias utilizadas para a implementação dos experimentos, bem como suas características e os experimentos para avaliação da arquitetura e sua extensibilidade.

O **Capítulo 5** descreve os trabalhos relacionados, Norma ISO/IEC 11179 e o *Meta Object Facility* (MOF), juntamente com uma análise comparativa com a arquitetura proposta. E, finalmente, no **Capítulo 6**, são apresentadas as conclusões finais e perspectivas de trabalhos futuros.

CAPÍTULO 2

GESTÃO DE METADADOS

2.1 Metadados

Metadados são definidos como "dados que descrevem dados". Podem ser utilizados para descrever objetos ou tornar pública a existência dos mesmos. Eles disponibilizam, descrevem, localizam e auxiliam na compreensão dos dados, transformando-os em conhecimento [38]. Ao ter conhecimento de quais dados estão disponíveis, entender o seu contexto e onde esses estão localizados, informações precisas são obtidas e melhores decisões podem ser tomadas [34].

Os metadados podem descrever diversos tipos de dados como mostra a Figura 2.1 [15]:

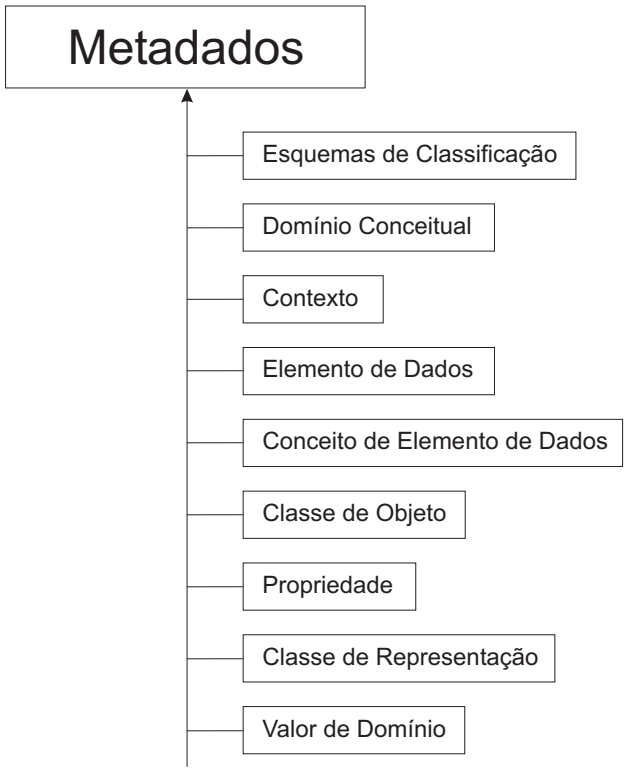

Figura 2.1: Tipos de informação descritos por metadados [15]

- **Esquemas de Classificação**: É a organização de objetos em grupos fundamentados em características que os objetos possuem em comum, como, por exemplo, origem, arranjo, estrutura, aplicação e função.
- **Domínio Conceitual**: É o significado de valor. Podem ser enumerados ou podem ser descritos por uma explicação detalhada. Os metadados carregam sua própria informação e permitem que sejam identificados, nomeados, definidos e, opcionalmente, classificados dentro de um Esquema de Classificação.
- **Contexto**: Ambiente ao qual é aplicado um nome ou definição. Define o objetivo pelos quais os dados têm significado. Podem ser aplicados a diversos tipos, com domínio de negócio, área de informação, sistema de informação, base de dados, arquivo, modelo de dados, ou qualquer outro elemento que necessite ser determinado.
- **Elemento de Dados**: Fundamental para uma organização, é uma unidade de dados para a qual a definição, a identificação, a representação e valores são especificados por meio de um conjunto de atributos.
- **Conceito de Elemento de dados**: É um conceito, podendo ser representado em um formulário de elementos de dados, que descreve independentemente de qualquer representação particular. Pode possuir ou não uma classe ou propriedade e possui uma definição independente da classe de objeto ou da propriedade.
- **Classe de Objeto**: São abstrações, geralmente associadas a idéias do mundo real e possuem significados, propriedades e comportamentos bem definidos.
- **Propriedade**: É uma característica que todas as classes de objetos possuem. Pode ser utilizada para distinguir um objeto de outro e também para definir características únicas.
- **Classe de Representação**: Pode ser considerada o Esquema de Clas-

sificação para representação, fazendo com que um determinado conjunto de classes torne possível diferenciar os elementos em um arquivo.

- **Valor de Domínio**: Um valor de domínio é a representação do valor, mas não tem nenhuma sugestão a respeito dos valores associados nem o significado desses valores.

Ao serem identificados e registrados, os metadados são gerenciados como elementos dentro do repositório, são nomeados e devem possuir pelo menos um Contexto. O Contexto possui um nome e definições podem ser especificadas em uma ou mais linguagens e define o escopo e o significado de cada tipo de metadados, podendo conter informações sobre o domínio de negócio, informações sobre áreas e sistemas, banco de dados, modelagem ou sobre qualquer ambiente determinado pelo proprietário do registro [20].

Um Esquema de Metadados é um conjunto de atributos definido para registrar informações sobre determinado assunto. Através da identificação de problemas no armazenamento e recuperação de informação por falta de padronização, vários esquemas foram criados para atender diferentes propósitos [26].

Os metadados possuem uma vasta aplicação e são utilizados para descrever uma infinidade de informações como imagens, vídeos, atributos, tabelas, bancos de dados, mapas espaciais, documentos, entre outros. As ferramentas que manipulam dados e fazem o mapeamento em ambientes de *Data Warehouse* [39], por exemplo, são baseadas em metadados, que descrevem as informações nele contidos e por eles manipulados, desde dados técnicos até regras de negócio.

Alguns sistemas para a leitura e descrição de metadados específicos, como sistemas de armazenamento e indexação de mapas digitais e gestão eletrônica de documentos arquivísticos, vem sendo desenvolvidos [3].

2.2 Contexto de Metadados

Metadados são constituídos de dados e contexto. A atribuição do contexto aos dados determina a que assunto esses dados estão relacionados. Essa informação auxilia a determinar não apenas a que esses dados estão relacionados como também a relevância dos dados armazenados e como eles devem ser utilizados, como é apresentado na Figura 2.2.

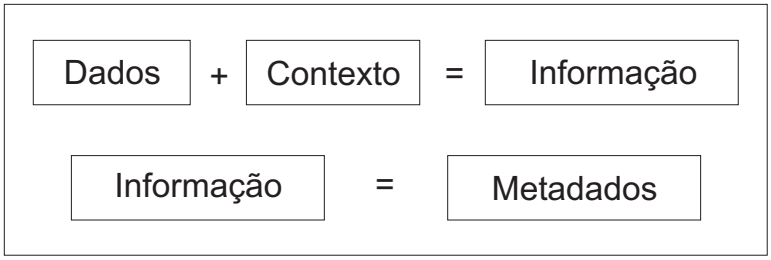

Figura 2.2: Relação entre Dados, Contexto, Informação e Metadados

No contexto organizacional, a gestão de metadados é considerada um aspecto chave para a obtenção dos resultados esperados em relação à busca de informações. Mas as organizações não esperam somente receber os dados corretos dentro de determinado contexto. Esperam também poder levar vantagens com o uso dessas informações e usá-las com conhecimento, como mostra o esquema na Figura 2.3.

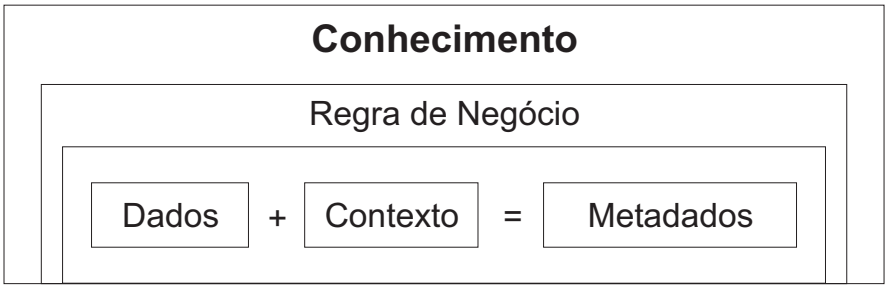

Figura 2.3: Utilizando informações com conhecimento

Utilizar os dados com sabedoria, como demonstrado na Figura 2.4, é um objetivo que demanda um grande esforço para qualquer organização, pois adiciona mais uma camada no controle de suas operações. Essa nova camada adicionada para auxiliar a administração dos dados é responsável pela monitoria e estabelecimento de métricas de avaliação das regras de negócio durante os processos de administração. Ela pode ajudar a

identificar problemas no comportamento das regras e políticas de negócio das organizações.

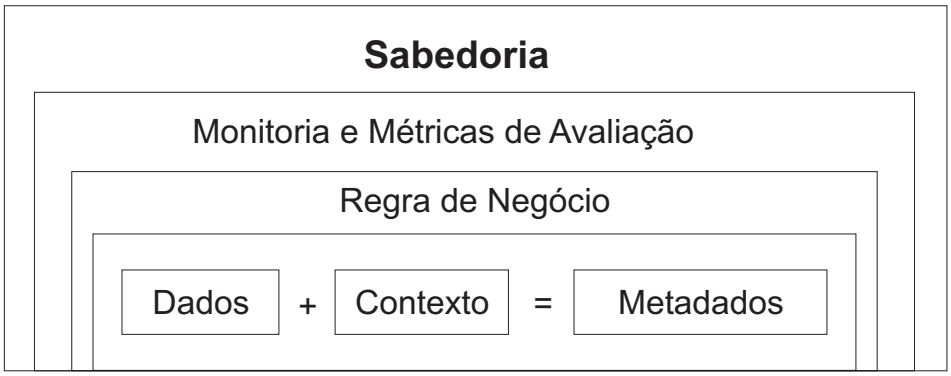

Figura 2.4: Utilizando informações com sabedoria

2.3 Categorias de Metadados

Metadados também são embutidos em arquivos, ou colocados separadamente para fornecer informações sobre os mesmos. Outra aplicação muito difundida está entre os ambientes de *Data Warehouses*, que utilizam metadados para mapear dados de diversas fontes e integrá-las em um único esquema de dados. São utilizados em muitos contextos e duas categorias podem ser destacadas [38]:

- **Metadados Técnicos**: Descrevem dados para aplicações para auxiliar no armazenamento, manipulação e movimentação dos dados, podendo conter informações para Descrição, Classificação, Composição e Histórico das informações por eles armazenadas. São úteis para garantir confiabilidade nos processos de manutenção durante o crescimento dos dados gerenciados.
- **Metadados de Negócio**: Descrevem dados a fim de esclarecer e simplificar o significado de determinado negócio ou processo específico dentro do contexto no qual está inserido.

2.4 Classificação de Metadados

Metadados podem ser classificados de diversas formas e sua classficação está ligada ao tipo de conteúdo ao qual são utilizados para representar, como, por exemplo, para representação de tipos, descrição, classificação, composição, histórico, localização e estatística [38].

- **Descrição de Conteúdo:** Descreve o conteúdo do armazenado. Existem pelo menos três maneiras de extrair essa informação, manual, automática e semi-automática, dependendo do tipo de informação que o dado em questão possui.
- **Classificação de Conteúdo:** São informações suplementares que geralmente são extraídas do conteúdo dos dados, podendo ser extraída de forma manual ou automática.
- **Composição**: São dados lógicos do conteúdo que possuem algum tipo de semântica como parte do conteúdo. Esse tipo de metadados possibilita conhecer determinadas características dos relacionamentos entre os metadados.
- **Histórico**: Descreve informações relativas às modificações e alterações durante o ciclo de vida dos metadados. Esse tipo de informação geralmente é gerada automaticamente.
- **Localização**: São metadados utilizados para localizar informações. Podem conter, por exemplo, o local onde pode ser encontrado determinado arquivo que possua determinada informação ou descrever em que ambiente essa informação está armazenada.
- **Estatístico**: Utilizado para determinar a freqüência de acesso a determinada informação. Esse tipo de metadados é constantemente associado aos metadados do histórico para obter informações de como os dados foram relacionados ao longo do tempo.

Os metadados são utilizados por aplicações, Sistemas de Gerenciamento de Banco de Dados que também utilizam esse recurso em seus dicionários de dados e catálogos.

2.5 Padrões de Metadados

A utilização de padrões já foi vista como forma de limitação entre a comunidade de desenvolvedores. Com o crescimento dos tipos de dados e o volume de informações, os processos de padronização passaram a ser vistos como grandes aliados nos ambientes de gestão de dados [38].

O investimento em padrões facilita a solução de problemas conhecidos, pois, antes de tornar-se um padrão, essas soluções foram amplamente testadas e, após sua eficácia comprovada, disponibilizada à comunidade, trazendo diversos benefícios a seus usuários. Simplifica a atividade de análise, pois geralmente são amplamente documentados, trazendo soluções para problemas que em alguns casos ainda não tinham sido previstos pelos analistas, facilitando a comunicação entre os usuários e proporcionando uniformidade e integração entre soluções [12].

Os padrões estão cada vez mais presentes dentro das organizações. Para atender aos diversos aspectos gerenciados por essas organizações, vários padrões foram desenvolvidos para atender a diferentes áreas, tais como Bibliotecas Digitais, Multimídia, Documentos Arquivísticos, entre outras.

Padrões como *Dublin Core*, METS e MPEG7 são freqüentemente adotados por organizações. Esses padrões, embora sejam utilizados para contextos diferentes, possuem um fator em comum, são muito abrangentes e flexíveis, o que os tornam muito populares.

2.5.1 *Dublin Core Metadata Initiative* (DCMI)

O Padrão *Dublin Core*, desenvolvido pela *Dublin Core Metadata Initiative* (DCMI) [10], define um grupo de atributos utilizado por autores para descrever seus próprios recursos na *web*. Esse padrão se destaca pela simplicidade, interoperabilidade semântica, consenso internacional, extensibilidade e modularidade de metadados na *web*.

O *Dublin Core* possui quinze atributos, que são descritos na Tabela 2.1.

Atributo	**Descrição**
Título	Nome dado ao recurso
Criador	Entidade responsável por criar o recurso
Assunto	Tópico ou Assunto do recurso
Descrição	Descrição do que se trata o recurso
Publicador	Entidade responsável por fazer o recurso disponível
Colaborador	Entidade responsável por fazer contribuições ao recurso
Data	O ponto ou período de tempo associado ao evento no ciclo de vida do recurso
Tipo	A natureza ou gênero do recurso
Formato	O formato do arquivo ou dimensões do recurso
Identificador	Uma referência única dada ao recurso com um contexto
Fonte	Descrição de um recurso do qual este recurso é derivado
Idioma	Idioma do recurso
Relação	Um recurso relacionado
Abrangência	Aplicação do recurso
Direitos	Informações sobre direitos autorais

Tabela 2.1: Atributos do Padrão *Dublin Core*

No caso do padrão *Dublin Core*, por possuir atributos extremamente genéricos, o mesmo pode ser utilizado nas mais diversas áreas de conhecimento, o que o torna muito versátil [10] e fará parte dos experimentos realizados no Capítulo 4.

2.5.2 *Metadata Encoding and Transmission Standard* (METS)

O *Metadata Encoding and Transmission Standard* (METS) [23] é um padrão para a codificação de metadados descritivos, administrativos e estruturais, utilizados para a gestão e a troca de objetos de repositórios de bibliotecas digitais. Essas bibliotecas de objetos digitais requerem a manutenção de vários tipos de metadados estruturados. Para manter metadados descritivos sobre um livro e garantir que os dados possam ser utilizados sem dúvidas sobre sua precisão, é importante a utilização de metadados técnicos.

O METS é codificado em XML *(Extensible Markup Language)* [9]. Esse padrão possui sete seções, em cada seção um grupo de atributos, sendo: Cabeçalho METS, Metadados Descritivos, Metadados Administrativos, Seção de Arquivos, Mapa Estrutural, Ligações Estruturais, Comportamento. O Padrão METS é um esquema de metadados

mantido pelo *Network Development* and *MARC Standards Office* da Biblioteca do Congresso dos Estados Unidos [23].

2.5.3 Alguns padrões do *Movie Picture Experts Group* (MPEG)

O *Movie Picture Experts Group* (MPEG) [25] é o grupo criado pela ISO/IEC, encarregado do desenvolvimento de padrões para a representação de codificação de dados digitais de áudio e vídeo. Esse grupo produziu vários padrões como o MPEG-1 do qual resultaram em produtos como o Vídeo CD e o MP3, MPEG-2, utilizado na Televisão Digital e no qual o formato de vídeo do DVD é baseado, MPEG-4, que está sendo aplicado para vídeos de alta compressão, como os utilizados por dispositivos móveis e vídeos pela *web*, MPEG-7, utilizado para descrever dados de áudio e vídeo.

O Padrão MPEG-7, que é uma interface de descrição de conteúdo multimídia, foi desenvolvido no intuito de prover um modelo para ser utilizado em repositórios de dados para que possam ser utilizados por pessoas ou para que possa ser recuperado de forma automatizada por aplicações [25].

O MPEG-21 é um framework multimídia aberto, criado para ser utilizado na padronização da construção, distribuição e consumo de elementos multimídia, descrevendo como são esses elementos e como são relacionados uns com os outros. Isso facilita a interoperabilidade, trazendo benefícios aos seus consumidores de conteúdo e provendo a eles o acesso a uma grande variedade de conteúdo.

Como pode ser visto, o grupo MPEG está dedicado à definição de padrões para dados multimídia, seus esforços resultaram em vários padrões, sendo que alguns desses padrões como MPEG-2, MPEG-3 e MPEG-4 são amplamente utilizados pela indústria e fazem parte do nosso cotidiano.

2.6 Mapeamento entre Padrões de Metadados - *Crosswalking*

Um *Crosswalk* provê um mapeamento de elementos de metadados entre um padrão de metadados e outro, ou seja, os mapeamentos relacionam equivalências entre dois ou mais esquemas de metadados [34].

Um fator decisivo que deve ser observado é a precisão na definição dos elementos de dados dos esquemas de metadados. Somente a partir da definição precisa desses elementos é possível que sejam obtidos resultados confiáveis entre padrões distintos [34].

A maioria dos *Crosswalks* desenvolvidos é baseada em padrões de metadados e fazem apenas o mapeamento semântico dos elementos de metadados, como, por exemplo, os *Crosswalks Dublin Core to Encoded Archival Description* (EAD) e *EAD to International Standard Archival Description* (ISAD(G))[24]. O *Crosswalk* deve prover uma conversão entre a especificação dos padrões, incluindo as regras de elementos para elementos no mapeamento, hierarquia e conversão dos conteúdos.

Portanto, um *Crosswalk* completo não deve apenas associar os elementos de dados equivalentes entre um padrão e outro, deve também relacionar as regras de validação desses elementos de dados, para garantir que os dados que serão trocados entre os padrões atendam corretamente as necessidades de seus usuários.

Na Tabela 2.2, é demonstrado um exemplo de associação de alguns dos elementos de dados que compõem o *Crosswalk Dublin Core*, para mapeamento EAD. Nesta tabela está relacionado o cabeçalho principal, inerente a cada padrão.

Dublin Core	**EAD**
Criador	Autor
Colaborador	Autor
Editor	Editor
Direitos	

Tabela 2.2: Exemplo de *Crosswalking - Dublin Core* to EAD[36]

Nesse caso da Tabela 2.2, pode ser visto que os metadados que compõem autoria, do padrão *Dublin Core*, diferencia o autor principal dos colaboradores, classificando-os como Criador e Colaborador. No momento da conversão, o padrão EAD não possui essa

diferenciação e os atributos Criador e Colaborador farão parte do mesmo contexto de Autor.

Pode ser visto também que, no padrão EAD, não possui tipo de metadados para especificação de direitos autorais como o *Dublin Core*, e nesse caso o dado será ignorado no padrão EAD.

2.7 Repositórios de Metadados

Um repositório deve prover funcionalidades para a integração e o acesso independente para manipular os dados e a estrutura dos metadados. Essa integração deve possibilitar que os metadados possam acessar e relacionar os dados entre si.

Para o desenvolvimento de um repositório, devem ser abordados três aspectos em sua arquitetura [34]:

- Base de Dados
- Metamodelos
- Software de Manipulação do Repositório

Como pode ser visto na Figura 2.5, o repositório é um conjunto de aplicações que registra informações sobre cada padrão, acessando a aplicação que possui os metamodelos (esquemas de metadados), que fazem a interface com o meio de armazenamento dos dados.

Com a existência de diversos padrões, vários sistemas de repositórios foram criados. Essa prática força os usuários de metadados a utilizarem vários sistemas diferentes, já que são utilizados para finalidades diferentes.

Vale ressaltar a necessidade de integração entre esses metadados, pois muitas vezes esses dados registrados são relacionados, mas não estão integrados. Nesses casos a integração minimiza o esforço para sincronização e consistência dos dados [11].

Ao serem registrados os metadados e definidos os esquemas de metadados, o repositório deve prover um ambiente no qual os padrões que foram registrados possam manipular os metadados e valores.

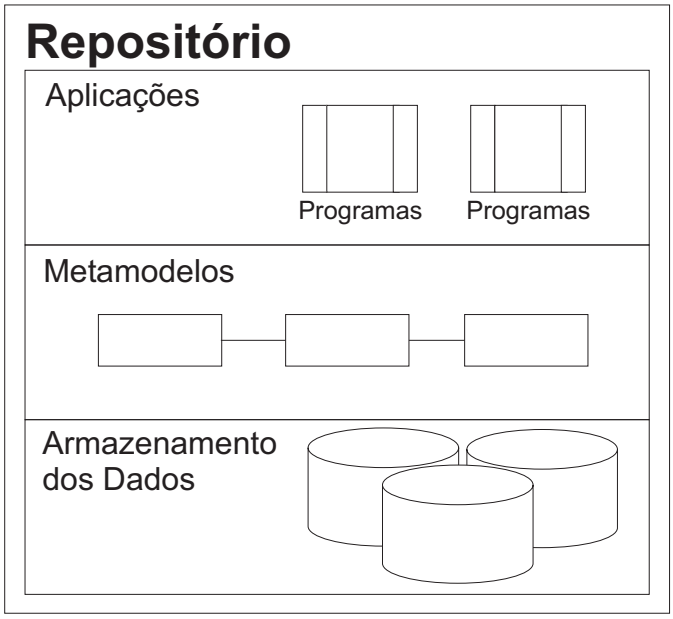

Figura 2.5: Arquitetura de um Repositório de Metadados [34]

Isso contribui com o trabalho de gestão de metadados e auxilia o Administrador a decidir quais metadados devem ser coletados, mantidos e onde novos metadados propostos pela comunidade que utiliza os repositórios estarão integrados e reutilizados.

Sistemas de registros de metadados devem ser a base de um ambiente de gestão de metadados, provendo as funcionalidades para localização, atualização e utilização dos dados que os metadados descrevem [34].

2.8 Problemas comuns na Gestão de Metadados

Ao iniciar um processo de gestão de metadados, é preciso definir quais metadados serão coletados e mantidos, mas o sistema de registro de metadados deve permitir que os metadados armazenados por ele possam receber manutenções através do tempo [38]. Contudo, sempre surgem problemas nesse processo:

- **Variedade de formas**: Metadados podem possuir muitas formas, contendo diferentes maneiras de tratar atributos, funções, modelos entre outros.
- **Novos metadados**: O ambiente deve tornar possível a adição de novos esquemas de metadados, para atender a diferentes usuários.

- **Tipos diferentes de metadados**: Metadados podem ser criados, a todo momento, para atender a diferentes propósitos, mas deve haver o cuidado para que os metadados criados não possuam a mesma semântica que outros existentes, gerando duplicidade de metadados para o mesmo propósito.
- **Usuários de metadados**: Quando existe um número muito grande de usuários, os repositórios de metadados tornam-se muito grandes, e as informações acabam não sendo conhecidas pelos usuários.
- **Vocabulários de metadados**: O modelo deve prover um formato uniforme para facilitar a gestão dos metadados pelos usuários.

2.9 Considerações Finais

Ao longo deste capítulo, foram apresentados os principais conceitos relativos à gestão de metadados, enfatizando a flexibilidade dos mesmos para descrever diversos tipos de dados. Destacou-se também que os metadados estão presentes em vários tipos de aplicações e podem ser armazenados de diversas formas.

Para ressaltar a importância da padronização de dados dentro das organizações, foram descritos os padrões *Dublin Core*, METS e alguns dos padrões definidos pelo grupo MPEG. Também foram abordados os conceitos de repositório de metadados e os problemas comuns no seu processo de gestão. Esses problemas de gestão, citados na Seção 2.8, podem ser resolvidos e principalmente prevenidos com a utilização de um ambiente adequado de gestão, que provê a integração dos metadados desde a sua composição.

CAPÍTULO 3

ARQUITETURA INTEGRADA DE REPOSITÓRIO DE PADRÕES E METADADOS

Com o crescente interesse das organizações em padronizar o uso de metadados, diversos padrões foram criados para atender diversos contextos, e vários sistemas de registro de metadados foram criados no intuito de atender os requisitos de cada padrão.

Essa atividade de desenvolvimento, voltada para o padrão e não para os metadados, faz os usuários de metadados utilizarem vários sistemas diferentes para registrar seus metadados. Esse problema afeta os desenvolvedores, que acabam criando várias estruturas de sistema para se adequarem a cada padrão que seus usuários necessitam.

Além desses problemas, existe a necessidade de integração entre os dados de metadados entre padrões distintos, pois muitas vezes esses dados de alguma forma estão relacionados. Com estruturas diferentes e modelagens fortemente atreladas aos dados, relacionar as informações entre padrões pode ser uma tarefa difícil de ser realizada.

A integração dos dados pode beneficiar os usuários de várias formas, como no contexto de negócio, podendo auxiliar no relacionamento dos dados e ajudar a compreender novas situações criadas por esse relacionamento, mas pode ser utilizada no contexto de tarefas técnicas, como minimizar o esforço para sincronização e consistência dos dados [11].

Em um repositório baseado na especificação proposta, além do acesso aos dados que os metadados descrevem, os metadados podem ser gerenciados, descritos, localizados e atualizados.

A arquitetura, a partir de agora chamada de Metapadrão, provê recursos para o armazenamento do significado dos dados e como esses dados são representados e descritos. A partir dessas descrições, pode ser compreendido como esses dados estão contextualizados e principalmente garantir que um tipo de metadados criado seja apenas reutilizado e não

duplicado.

3.1 Arquitetura

O Metapadrão provê um ambiente no qual os padrões de metadados registrados possam ser utilizados como esquemas de metadados, para que os dados sejam armazenados em um repositório de metadados.

Dentro do Repositório de Padrões de metadados são armazenados os padrões requisitados pelos Usuários, analisados pelo Administrador. Após, o Administrador define os atributos e registra os padrões de metadados que serão disponibilizados no repositório de padrões.

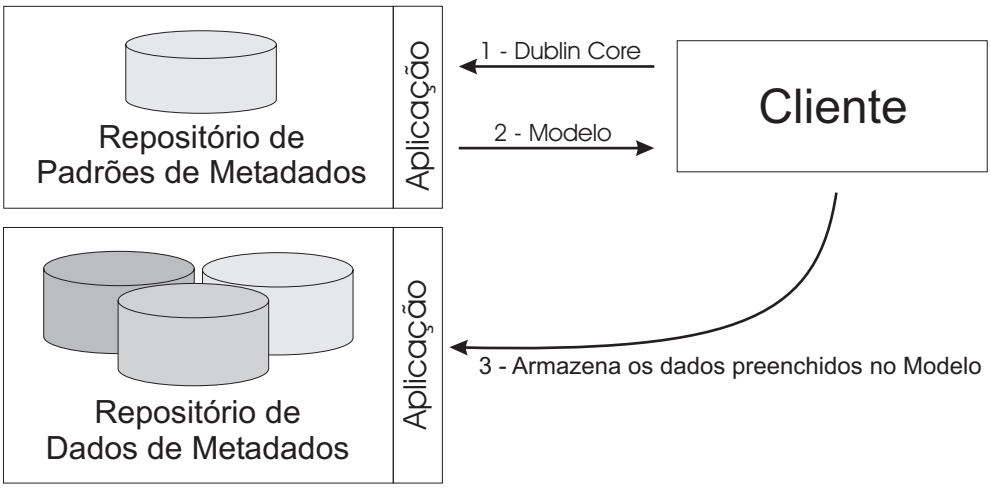

Figura 3.1: Arquitetura de Aplicações com o Metapadrão

Essa divisão na arquitetura de modelagem e armazenamento mostrado na Figura 3.1 contribui com o trabalho de gestão de metadados, pois auxilia o Administrador a definir ou redefinir quais metadados devem ser coletados e mantidos e quais novos metadados devem ser criados. Esses novos metadados estarão integrados e podem ser reutilizados sem a preocupação dos dados que foram gerados anteriormente.

Os metadados registrados devem possuir as seguintes especificações:

- Identificador único para cada elemento
- Versão do padrão utilizado na sua descrição

- Contexto dos metadados
- Definição de metadados obrigatórios, opcionais ou baseados em condições
- Um tipo de metadados pode ocorrer muitas vezes
- Metadados relacionados hierarquicamente
- Tipos de valores de metadados

A padronização na descrição de dados, o entendimento claro dos dados por meio dos elementos organizacionais, o reuso dos dados através do tempo, espaço e aplicações, a padronização dos dados dentro de uma organização e a gestão e reuso dos componentes de dados desvinculando a modelagem dos dados armazenados são outros benefícios providos pelo Metapadrão.

3.2 Aplicação do Metapadrão

O Metapadrão armazena todos os elementos de dados (Atributos) que podem ser utilizados na descrição dos padrões de metadados. Um elemento de dados pode ser a junção de vários elementos, também possui uma associação de contexto, e se o dado relativo ao elemento de dados é obrigatório, opcional ou condicional.

No que diz respeito a extensibilidade, o Metapadrão pode ser exemplificado da seguinte forma: Os Padrões *Dublin Core* e MPEG-7 serão registrados no Repositório de Padrões. O primeiro passo é registrar os atributos de cada padrão. Pode ser observado que tanto o Padrão *Dublin Core* quanto o MPEG-7 possuem o atributo *Creator*, esse atributo é relacionado ao contexto Autoria.

Nesse caso, o Administrador irá criar apenas um atributo *Creator* que será compartilhado entre os dois padrões, possibilitando o reuso, localização e a recuperação de informações relativas ao mesmo contexto entre padrões distintos.

O Administrador também poderá atribuir uma legenda ao atributo *Creator* para cada padrão no qual ele é utilizado, tornando mais compreensível para os usuários do padrão.

Nesse caso, encontrar todos os metadados criados por determinado autor poderá ser um processo integrado, já que tanto no Padrão *Dublin Core* quanto no MPEG-7 foram utilizadas a mesma estrutura de armazenamento de autores, como pode ser visto na Figura 3.2.

	Dublin Core	**MPEG-7**
Author	Creator	Creator

Figura 3.2: Atributo *Creator* utilizado pelos padrões *Dublin Core* e MPEG-7

Ao ter todos os atributos registrados, o Administrador identifica que o Atributo *Creator* descreve o nome e o sobrenome do autor do metadado no mesmo atributo.

Através de análises, nos dados que gerencia o Administrador, conclui-se que o Atributo *Creator* ou qualquer outro atributo que esteja relacionado ao contexto autoria do tipo de metadados deve registrar separadamente os dados do *name, last name, email*, como demonstrado na Figura 3.3.

	Dublin Core			**MPEG-7**		
Author Context	Creator			Creator		
	name	last name	email	name	last name	email

Figura 3.3: Atributo *Creator* estendido com novos atributos

Ao fazer essa alteração, o Administrador não somente redefine o contexto de autoria entre os padrões, tornando dessa forma esses dados homogêneos, como também tem a possibilidade de estender todos os padrões que irá utilizar.

3.3 Modelo Conceitual do Metapadrão

O modelo, descrito na Figura 3.4, é uma abstração do modelo conceitual dos dados que serão utilizados pelo Metapadrão para armazenar padrões de metadados. Essa especificação foi elaborada utilizando a Metodologia UML de diagrama de classes[21].

A partir de um modelo com essas características, deverão ser realizadas as consultas, e os dados deverão ser utilizados como modelos para o armazenamento do conteúdo dos metadados em outro meio de armazenamento.

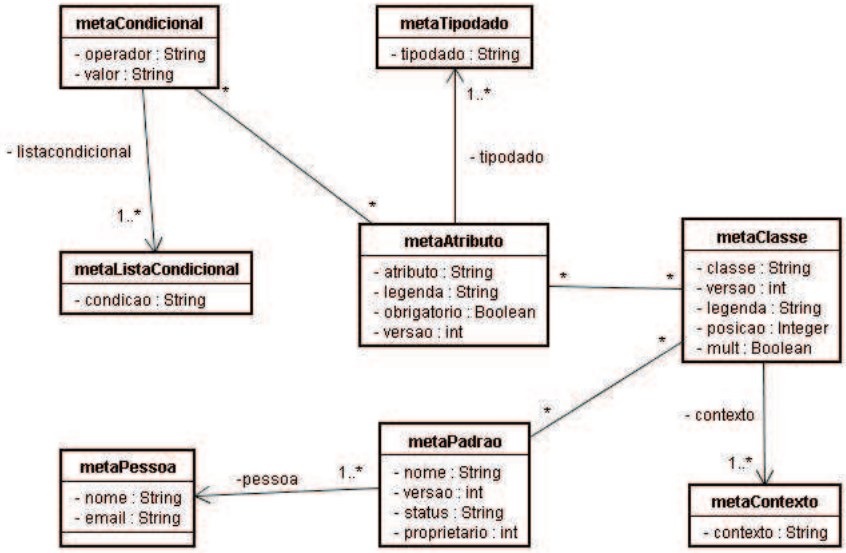

Figura 3.4: Diagrama de classes do Metapadrão

As classes apresentadas, no diagrama de classes na Figura 3.4, contêm atributos que representam os dados utilizados para armazenar os padrões no repositório. Na seqüência, a descrição das classes e seus atributos é apresentada:

metaPadrão: Possui atributos para registrar os padrões disponíveis no repositório.

- nome: Nome do padrão registrado.
- versao: Versão do padrão registrado, um padrão pode possuir várias versões. Esta informação é importante, pois é através da versão que deverão ser notificadas as atualizações em aplicações Clientes.
- *status*: Situação do padrão. Sugerido os *status* disponível, indisponível, obsoleto.

metaPessoa: Classe responsável por registrar os dados referentes ao responsável pela criação, administração e alterações no padrão.

- nome: Nome da pessoa responsável pelo padrão.
- email: Contato com o responsável pelo padrão.

metaClasse: Cada padrão possui um conjunto de classes que o compõe, e uma classe é a composição de um ou mais atributos, forma utilizada para reutilizar os metadados já registrados como atributos.

- classe: Nome da classe que poderá ser utilizada por qualquer padrão.
- legenda: Descreve a função da classe de forma legível. É ela que deve ser apresentada ao usuário pela interface Cliente no momento de preencher os dados.
- versao: Uma classe também pode ser atualizada, e assim sendo criada uma nova versão.
- posicao: Define a prioridade de exibição seqüencial de um atributo.
- mult: Informação booleana que indica se uma classe pode ser duplicada.

metaContexto: Indica o contexto ou forma de classificação das informações registradas em Classes. Informação útil para descobrir quais classes podem representar informações referentes ao mesmo assunto em diferentes padrões.

- contexto: Descrição do contexto.

metaAtributo: Na classe Atributo são inseridas as informações que irão representar atributos para as Classes dos Padrões.

- atributo: Descrição do atributo.
- legenda: Descreve o atributo de forma legível para que seja apresentado ao usuário na aplicação Cliente.
- obrigatorio: Define se um atributo é obrigatório ou não.
- versao: Um atributo pode ser atualizado, e assim sendo criada uma nova versão.

metaTipoDado: Ao inserir novos atributos, esses são vinculados a um tipo de dado que estão descritos nessa classe.

- tipodado: Descreve os tipos de dados disponíveis para os Atributos.

metaCondicional: Aos Atributos podem ser adicionadas as regras que testam valores condicionais.

- operador: Tipo de operador que deve ser comparado com o valor do atributo e o valor.
- valor: Valor a ser comparado na checagem condicional.

metaListaCondicional: Registra as situações nas quais as condições registradas na classe Condicional devem ser testadas.

- condicao: Descrição da condição.

As classes devem ser instanciadas considerando alguns critérios. Na entidade Padrão são especificados os padrões, cada atributo do padrão será descrito como uma classe, possuindo atributos nos quais serão definidas as informações, conforme especificado nos atributos por obrigatório, opcional ou condicional. Caso seja necessário utilizar alguma regra para preenchimento do atributo de propriedade, deverá ser especificada a condição na entidade condicional.

É obrigatória a especificação de um contexto para cada classe. As classes podem ser utilizadas por qualquer padrão, desde que o contexto dessa classe seja o contexto especificado pela norma para o padrão que se pretende descrever.

3.4 As camadas do Metapadrão

O foco deste trabalho é definir a arquitetura, como os dados dos padrões devem ser acessados pelas aplicações Clientes e como devem ser registrados em um repositório de metadados. Dessa forma, detalhes de implementação não fazem parte deste trabalho, tampouco a definição de tecnologias que devem ser utilizadas no seu desenvolvimento, com exceção do XML que é a base para a troca de dados.

No Capítulo 4, são apresentadas as tecnologias utilizadas na implementação dos experimentos baseados na arquitetura do Metapadrão, que servem de sugestão para o seu desenvolvimento.

3.4.1 Repositório de Padrões

O Repositório de Padrões deve ser um conjunto de aplicações, compostas por um banco de dados, no qual a modelagem descrita deve ser implementada e uma aplicação que auxilie o Administrador de metadados a gerenciar os padrões.

3.4.1.1 Funcionalidades - Aplicação para Administração de Padrões

A aplicação para registro de padrões no repositório deve ser capaz de manipular o banco de dados, possuindo todas as funcionalidades para inserção e atualização de dados dos padrões.

Definição de Padrões: Após estudo sobre o padrão de metadados ou definição de um novo modelo de metadados que será utilizado pela organização, a aplicação deve ser capaz de manipular esse padrão e registrar o proprietário desse padrão.

Copiar Padrões: A aplicação deve prover uma funcionalidade de copiar todos os dados presentes em um padrão, com a função de facilitar para o Administrador a preservação dos padrões existentes e criar novas versões de padrões de metadados baseados em versões anteriores.

Repositório de Padrões: Ao inserir um padrão ou definição de um novo modelo, o Administrador deve procurar utilizar os atributos e condições já existentes no Repositório de Padrões, caso não existam e após análise julgar necessário deve então adicionar os novos metadados.

Exclusão de um Padrão: Um padrão nunca deve ser excluído do Repositório, exceto quando não existem metadados registrados baseados nesse padrão. Quando um novo padrão for criado e um padrão não for mais necessário, esse padrão terá seu *Status* modificado, e a aplicação que faz a interface com as aplicações Clientes, que provê a consulta ao banco de dados, não irá mais entregar esse padrão, mas a informação de seu formato continua presente no banco de dados, podendo ser associado aos dados já registrados no Repositório de Metadados.

Interface com Aplicação Cliente: A aplicação utilizada pelo Administrador pode prover um serviço de interface com as aplicações Clientes que requisitam os padrões de metadados, ou poderão ser criadas aplicações específicas para isso.

Publicação dos Padrões Disponíveis: Através de página *web* ou qualquer outro meio de divulgação disponível, a aplicação deve possibilitar apresentar os padrões disponíveis no repositório para os usuários de metadados e exibir um formato XML deste padrão para facilitar aos desenvolvedores a criação de aplicações Clientes.

Funcionalidades para Notificações: Deve prover um meio para notificar aos Clientes que utilizam os padrões de metadados que uma nova versão de um padrão de metadados foi criada e que suas aplicações devem ser testadas ou atualizadas caso não estejam preparadas para funcionar com o novo padrão de metadados, antes que determinado padrão se torne Obsoleto.

3.4.1.2 Processo de Registro dos Padrões

O Administrador é o responsável por registrar e atualizar modelos de metadados solicitados pelos usuários, além de avaliar e transformar esse modelo em padrão de metadados que deve ser utilizado por todos os usuários que necessitam registrar metadados desse contexto. As solicitações de novos padrões deverão ser feitas ao Administrador com uma descrição detalhada da justificativa da adoção desse novo padrão.

Uma análise do padrão é realizada, comparando os atributos utilizados por esse novo padrão e os atributos já existentes no repositório. Também deve ser considerada, de acordo com a justificativa, a busca por um padrão já existente e que possa ser utilizado como está ou apenas estendido. Por isso a importância de uma análise minuciosa, que deve também procurar reutilizar todos os metadados já existentes no repositório, o que irá facilitar a busca de dados semelhantes entre padrões distintos, criando um mapeamento dos atributos entre modelos distintos já no momento da criação de um novo padrão.

Durante a criação de um novo padrão, duas informações muito importantes serão registradas no repositório de padrões: *Status* e Versão.

No atributo *Status*, o novo padrão será registrado como **Indisponível** até que esse padrão esteja completamente registrado. O atributo de Versão deve ser registrado com o valor inicial que irá depender do sistema de versionamento utilizado. Existem diversos modelos de sistemas de versionamento, e nessa especificação não é definido qual modelo deve ser utilizado, ficando a cargo do Administrador escolher o sistema de versionamento que melhor se adequará à sua organização.

Assim que todas as informações necessárias como Classes, Atributos, Condicionais estiverem registradas para o padrão, o *Status* deve ser atualizado para **Disponível**, esse ciclo de vida pode ser visto na Figura 3.5.

O processo de registro de padrões, como mostra a Figura 3.5, é um procedimento realizado por um trabalho conjunto entre o Usuário e o Administrador. O Usuário demonstra o que precisa e o Administrador procura atender essas necessidades vinculando-as a de outros usuários.

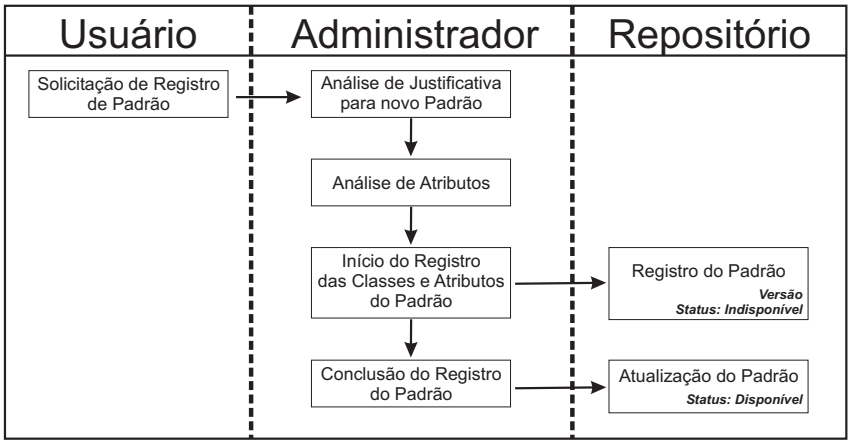

Figura 3.5: Ciclo de vida do registro de padrões

3.4.1.3 Versionamento

Existem diversos sistemas de versão utilizados em aplicações. Neste trabalho, não é indicado um sistema de versionamento específico, apenas que deve ser escolhido um sistema conforme a conveniência de cada organização.

A versão do padrão é um ponto muito importante para o Repositório de Padrões, pois, sempre que um padrão for requerido, será localizada a última versão disponível de um padrão.

Com o passar do tempo, padrões de metadados podem evoluir, informações adicionais podem ser requeridas, e informações que não estão disponíveis ou de baixa relevância para as organizações podem ser retiradas.

Mesmo assim, o formato de um padrão não deve se perder. Por isso o sistema de versionamento para padrões é tão importante, pois, sempre que um padrão evolui, ele ganha nova versão e metadados ou aplicações Clientes que foram preparadas para utilizar aquela versão devem continuar funcionando até que sejam atualizadas.

Assim que um padrão ganha uma nova versão, os usuários desse padrão devem ser notificados, por isso é proposto que todos os usuários de determinado padrão tenham um cadastro de uso. Esse cadastro não tem relação com o sistema de autenticação que não é descrito neste trabalho, mas pode ser utilizado por ferramentas para fazerem a autenticação de usuários autorizados a acessar os dados de determinados padrões.

Essa alteração não diz respeito apenas às aplicações Clientes do repositório, mas

também podem ser aplicadas aos metadados já criados utilizando a versão anterior, dando aos usuários do repositório uma forma para atualizar os metadados já criados.

3.4.1.4 Proprietário

O proprietário do padrão é o Usuário que solicitou a criação de um padrão de metadados ao Administador. O proprietário faz parte de uma lista de cadastro de pessoas que utilizam padrões, mas outros usuários, além de poder utilizar os padrões, podem solicitar alterações no modelo ao Administrador e não apenas o Proprietário.

Para modificações em um padrão podemos ver na Figura 3.6.

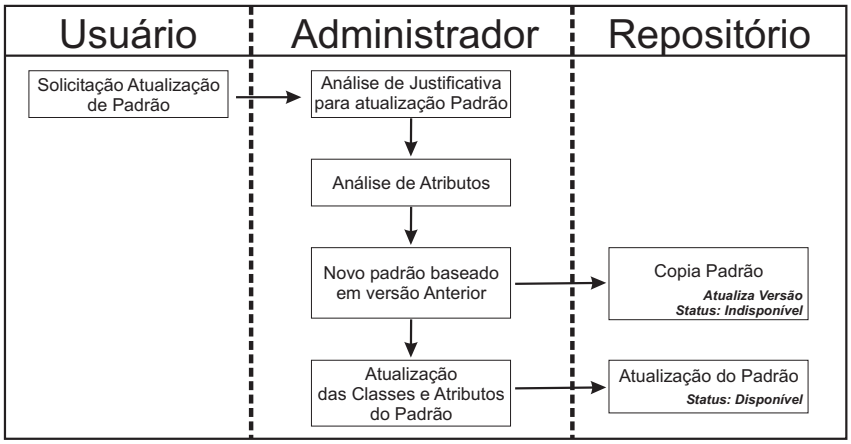

Figura 3.6: Ciclo de vida de atualização de padrões

Em um processo de atualização de padrões, um grupo de usuários pode ter a necessidade de criar um novo padrão baseado em outro, e o proprietário do padrão original concorda que a nova versão é mais adequada. Nesse caso, o padrão anterior não deve mais ser utilizado e deve receber o *status* de Obsoleto. Quando um padrão recebe o *status* de Obsoleto, ele ainda pode ser carregado pelas aplicações Cliente, mas tem seu uso desencorajado, sendo uma forma de notificar os seus usuários a buscarem a versão mais recente.

Outra situação, que pode ocorrer nas atualizações, é que as alterações propostas não estejam de acordo com as necessidades do proprietário. Então, esse grupo de usuários e o Administrador podem concordar que existe a necessidade da criação de um novo padrão baseado no padrão que está sendo utilizado. Nesse caso, o Administrador irá criar um

novo padrão herdando as características do padrão em questão.

3.4.2 Aplicação Cliente

A aplicação Cliente é a responsável pela interface direta com o usuário. Ela deve ser capaz de consultar a aplicação que controla o Repositório de Padrões, passando a informação do padrão desejado, ler esses dados escritos em XML e disponibilizar um ambiente para o preenchimento dessas informações pelo usuário.

Essa aplicação pode ser criada de diversas formas e para diversas finalidades, ou seja, a aplicação pode ser um Cliente Genérico ou específico para um padrão, pode ser um Cliente para consulta de metadados já registrados e até mesmo aplicações ou algoritmos de extração de metadados. Os tipos de Clientes sugeridos estão abaixo descritos:

Cliente Genérico: Deve ser capaz de ler todas as informações XML retornadas pelo Repositório de Padrões, aplicando todas as condições de atributos, para qualquer padrão de metadados. Deve também possuir flexibilidade para informar qual padrão deve montar e possuir funcionalidades para descobrir quais padrões de metadados estão disponíveis no Repositório de Padrões.

Cliente Específico: Criado para atender um determinado padrão e determinado grupo de usuários. Pode possuir detalhes mais específicos sobre o padrão ao qual implementa.

Cliente de Consulta de Metadados: Aplicação criada para consultar dados no Repositório de Metadados. Possui como finalidade principal a localização e integração dos dados registrados entre padrões distintos e tornar possível que metadados de mesmo contexto possam ser localizados entre vários padrões distintos de metadados.

Algoritmos de Extração de Metadados: Podem ser programas que fazem a análise automática ou semi-automática de dados

multimídia e registram esses dados no Repositório de Metadados conforme o padrão especificado no Repositório de Padrões.

Não faz parte do escopo deste trabalho definir o sistema de autenticação ou validação de usuários dos padrões ou dos Clientes.

3.4.3 Repositório de Metadados

O Repositório de Metadados deve ser um local de armazenamento centralizado, capaz de armazenar os dados XML registrados pelas aplicações Cliente, possuindo uma aplicação que faça a interface com a aplicação Cliente e o sistema de armazenamento.

A aplicação que faz a interface com a aplicação Cliente irá fornecer as funcionalidades para Consulta e Notificação.

Consulta: Deve possibilitar que todos os dados registrados no Repositório de Metadados possam ser lidos e comparados, podendo ser capaz de consultar metadados criados a partir de padrões de metadados distintos.

Notificação: Possuir os mecanismos necessários para localizar metadados de padrões distintos, com funcionalidades para notificar o usuário de eventuais problemas em seus metadados e para atualização de versão nos dados dos metadados.

A tecnologia utilizada para registrar os metadados não faz parte do escopo deste trabalho, ficando a critério do usuário definir por algum sistema de registro de dados em arquivos ou utilizar um Sistema Gerenciador de Banco de Dados que dê suporte à manipulação de dados XML.

3.4.3.1 Ciclo de Vida dos Metadados

Um ponto muito importante a ser considerado nesse repositório de metadados é a relação entre os metadados registrados no Repositório de Metadados e o padrão que está registrado no Repositório de Padrões, conforme mostrado na Figura 3.7.

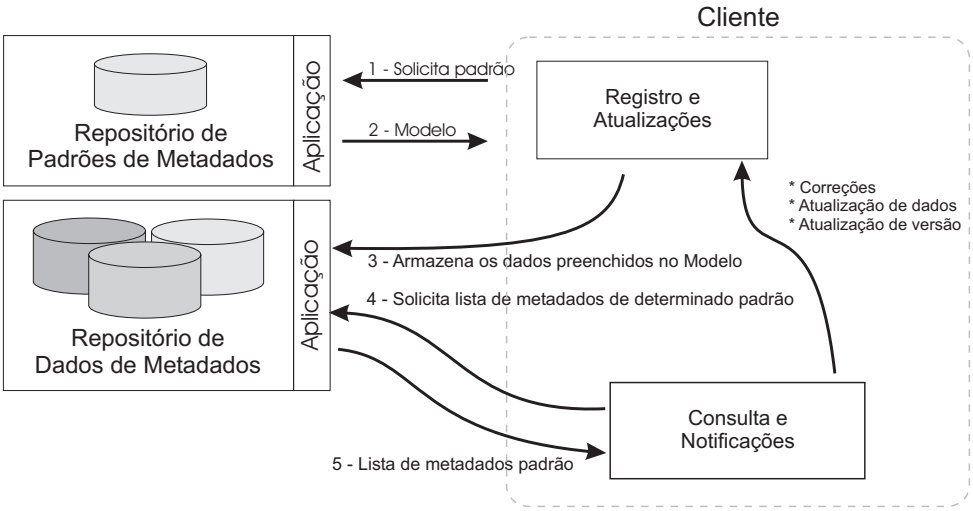

Figura 3.7: Ciclo de vida dos metadados

Uma vez que os padrões e os metadados estão versionados e sua ligação não é fortemente dependente, o padrão pode evoluir de versão e os metadados registrados no repositório não, tornando-se um processo opcional. Através do recurso de versionamento, esses metadados criados em determinadas versões podem ser identificados e, se assim for conveniente, podem ser atualizados pelo seu proprietário.

Portanto, o trabalho propõe que os metadados sejam registrados, versionados, consultados. Os proprietários dos metadados devem ser notificados quando houver alterações nos padrões e os metadados devem ser atualizados quando necessário.

3.4.3.2 Dados do Proprietário

Além do XML dos metadados, o Repositório de Metadados deve possuir informações básicas sobre os Proprietários dos metadados registrados, minimamente são considerados: as informações, Proprietário (Nome e Sobrenome), E-Mail, Data de Registro de Metadado, Data de Ultima Atualização, Padrão e Versão.

Com esses dados, é possível notificar o proprietário quando houver a necessidade da atualização de versão de determinado padrão e verificar se um metadado nunca foi modificado, ou seja, regras como atualizações periódicas podem ser criadas para manter os metadados sempre atualizados.

3.5 Considerações Finais

Este capítulo apresentou o Metapadrão, um ambiente para gestão de metadados que tem o objetivo de resolver problemas provenientes da administração de diversos padrões de metadadados, descrição e integração dos mesmos.

Com uma arquitetura em três camadas, provê uma divisão da responsabilidade do controle sobre os dados, em que uma camada é responsável pelo repositório de padrões de metadados e outra pelo armazenamento dos mesmos. Como especificado, ambas devem prover um sistema de interface que possibilite que uma terceira camada, a aplicação Cliente, possa interagir com as mesmas, realizando consultas no Repositório de Padrões e registrando informações no Repositório de Metadados.

CAPÍTULO 4

EXPERIMENTOS

Os metadados armazenam e descrevem informações de diversos tipos de dados e podem ser utilizados em diversas aplicações como ambientes *web*, *Data Warehouses*, arquivos, documentos, artigos, livros e vários outros.

Propondo a unificação de um ambiente que provê um local de acesso aos metadados registrados de diversos padrões diferentes e diversas finalidades, como foi apresentado na Seção 3.1, o repositório de metadados descrito pelo Metapadrão propõe uma divisão na arquitetura de construção do repositório.

- Repositório de Padrões
- Clientes
- Repositório de Metadados

As três camadas possuem finalidades e características diferentes no processo de implementação do repositório. Nesse experimento esses conceitos vistos anteriormente são abordados de forma prática.

O experimento consiste na modelagem do banco de dados conforme a especificação do Metapadrão, utilizando uma aplicação para realizar a troca de dados entre o Repositório de Padrões e a aplicação Cliente; um Cliente genérico que pode ser utilizado para vários padrões; e outra aplicação que registre os dados de maneira centralizada em formato XML, para que possam ser localizados os dados de todos os metadados de padrões diferentes. Esse experimento valida o Metapadrão proposto, com a especificação e implementação de um repositório de padrões de metadados.

4.1 Tecnologias utilizadas na implementação dos experimentos

4.1.1 *Extensible Markup Language* (XML)

Extensible Markup Language (XML) é uma linguagem de marcação de texto flexível, definida pela Norma ISO 8879 e desenvolvida pelo *World Wide Web Consortium* (W3C), que desenvolve e mantém diversos padrões *web*. Essa linguagem é uma evolução da linguagem *Standard Generalized Markup Language* (SGML) e foi criada com as premissas de prover uma linguagem que fosse extensível, estruturada e que pudesse ser validada[32, 42].

Pela sua simplicidade na troca de informações, diversos tipos de aplicações podem ser baseadas em XML, como, por exemplo, carga e descarga de dados em banco de dados; simplificação do processo de troca de dados entre aplicações de comércio eletrônico, no qual diferentes organizações precisam cooperar para servir um Cliente; e também na troca de informações entre dispositivos móveis como *palmtops*, celulares, *smart phones* entre outros[22].

Além da diversidade de aplicações que podem utilizar essa linguagem de marcação, outros fatores influenciaram o seu sucesso:

- XML pode armazenar e organizar dados de qualquer natureza.
- Por ser um padrão aberto e não estar vinculado a qualquer companhia e a nenhum software em particular.
- Descrito com o padrão de caracteres *Unicode*, XML pode dar suporte a diversos sistemas de qualquer ambiente.
- XML oferece muitas formas de validar a qualidade de um documento, com regras de sintaxe, links internos de checagem e comparação de documentos.
- Possui um formato de apresentação dos dados simples e claro, facilmente compreendido por humanos e outros programas.

- XML pode ser combinado com *stylesheets* para criar a formatação dos dados na forma que o usuário preferir.

4.1.1.1 Conceitos básicos de XML

A unidade básica de marcação é o elemento. Elementos podem conter textos, caminhos de imagens, tabela de dados e outros tipos de elementos.

Quando são criados modelos de informação, uma hierarquia de informação deve ser definida. Essa hierarquia descreve quais elementos podem ser utilizados e como eles podem se encaixar.

Os elementos são organizados hierarquicamente, um dentro do outro. Eles possuem relação com os elementos que os contêm. Essa relação se torna útil quando é necessário recuperar ou ordenar algum tipo de informação[1].

Algumas regras básicas para definir elementos na hierarquia:

- O elemento principal da hierarquia deve estar entre todos os outros elementos definidos no modelo. Esse elemento é chamado de raiz.
- Um elemento que possui outros elementos hierarquicamente entre o início e o fim das tags é chamado de elemento pai, e os elementos que estão mais adentro são chamados de elementos filhos.
- Um elemento pode ser o pai de outros elementos enquanto for filho de um elemento. Essa regra só não é aplicada ao elemento raiz.
- Elementos que possuem o mesmo pai são considerados elementos irmãos.

4.1.1.2 Escrevendo documentos XML

É preciso levar em consideração que as informações geralmente estão armazenadas em algum formato, como, por exemplo, banco de dados, arquivos, entre outros. Podendo ser apresentado ao usuário como mostra na Figura 4.1.

A linguagem XML possibilita que o usuário transforme essas informações e define quais marcações em torno dos dados devem ser adicionadas para identificar o tipo de

```
Alcione Benacchio
10/10/1979
Foz do Iguaçu - PR
```

Figura 4.1: Dados simples sem marcações

informação. Na Figura 4.2, pode ser visto que para marcar essas informações deve ser adicionado ao início e ao fim da informação as tags de marcação.

```
<Elemento>conteúdo</elemento>
```

Figura 4.2: Exemplo de marcação XML

No qual <`elemento`>, à esquerda do conteúdo, representa o início da marcação e </`elemento`>, à direita, o fim da marcação. Portando os dados citados na Figura 4.1, seriam descritos em XML como apresentado na Figura 4.3.

```
<pessoa>
  <nome>Alcione Benacchio</nome>
  <nascimento>10/10/1979</nascimento>
  <cidade>Foz do Iguaçu</cidade>
  <estado>PR</estado>
</pessoa>
```

Figura 4.3: Dados com marcações XML

4.1.2 *Extensible Stylesheet Language* (XSL)

Especificado pelo W3C no intuito de simplificar a apresentação dos dados XML, o XSL é uma linguagem que define folhas de estilo para documentos XML. Essas folhas de estilo possuem um formato próprio de tags que definem a maneira como a informação e um documento de formato padrão devem ser apresentados [43, 5].

Uma transformação XSL envolve quatro etapas:

1. A criação do código que identifica uma origem XML
2. Um XSL *stylesheet*
3. O método de transformação de saída
4. O destindo da transformação, que geralmente descreve o processador XSL

O processador XSL lê o documento XML de origem e realiza uma transformação dos atributos, elementos e textos XML em instruções nas folhas de estilo do XSL[22].

4.1.3 Java Servlets

Servlets é uma tecnologia Java similar à tecnologia *Common Gateway Interface* (CGI). E de certa maneira são códigos Java que adicionam funcionalidades a um servidor *web*.

Foram desenvolvidos para dar suporte ao modelo computacional de requisição e resposta, que é um modelo extremamente popular entre os servidores *web*. Nesse modelo de requisição e resposta, um cliente envia uma mensagem ao servidor fazendo uma requisição e o servidor responde enviando uma mensagem de resposta [33].

Podemos utilizar os servlets para realizar diversos tipos de tarefas, como processar formulários HTML, camadas intermediárias para o processamento de conexão de recursos e manter serviços como sessões de banco de dados [14].

4.1.4 Servidor *Web* - Apache Tomcat

O Tomcat foi o primeiro servidor *web* Java e começou a ser desenvolvido por James Duncan Davidson, que escreveu esse servidor *web* baseado na idéia de servlet e JSP[35].

De uso livre, possui uma implementação de código aberto focado nas tecnologias Java Servlet e Java Server Pages. Atualmente é desenvolvido pelo projeto Jakarta da *Apache Software Foundation*.

Diversos desenvolvedores e empresas contribuem e utilizam o Tomcat como servidor *web*. Sua implementação está disponível para que qualquer empresa ou desenvolvedor possa utilizar esse servidor *web* para o desenvolvimento de ferramentas e a criação de sites dinâmicos e interativos[4].

4.1.5 *Web Services*

Web services são aplicações que fazem interface entre duas aplicações, proporcionando uma camada de abstração que separa as aplicações como cliente e provedor de serviços. Usam documentos XML, criados no formato de mensagens, no qual um programa envia uma requisão a um *web service* através da rede, utilizando protocolos como o HTTP ou SMTP, que ao receber a mensagem pode responder utilizando também um documento no formato XML[27, 7].

Essa tecnologia pode ser utilizada de diversas formas e ser aplicada em diferentes ambientes. Pode ser utilizada para aplicações *desktop* e clientes *handheld* para acessar outras aplicações, como aplicações da internet por exemplo[27].

Algumas características dos *web services*[7]:

- **Baseado em XML**: Utiliza o XML como camada de representação de dados para todos os protocolos e tecnologias que são criados. Elimina qualquer dependência de rede ou sistema operacional para o transporte dos dados.
- **Acoplamento Fraco**: Uma aplicação dependente de um *web service* não está totalmente amarrada, sendo possível que a interface seja modificada sem comprometer a habilidade de interação das aplicações clientes.
- **Simplicidade**: Tecnologias orientadas a objetos podem expor seus serviços através de métodos individuais.
- **Pode ser Síncrono e Assíncrono**: Uma aplicação cliente pode ser síncrona ou assíncrona. De forma síncrona deve aguardar a sua conclusão do processamento para continuar, enquanto na forma assíncrona a aplicação pode realizar uma requisição ao *web service* e seguir com a execução da aplicação.
- **Suporte a *Remonte Procedure Calls* (RPCs)**: *Web services* podem ser clientes que invocam *procedures*(procedimentos), funções e outros métodos de objetos remotos, utilizando um protocolo baseado em

XML. Essas *procedures* remotas devem possuir parâmetros de entrada e saída que o *web service* deve dar suporte.

Web services é um *framework* de troca de mensagens. O único requisito para um *web service* é que ele seja capaz de enviar e receber mensagens utilizando uma combinação de protocolos da internet. Os protocolos utilizados na definição dos *web services* são SOAP, WSDL e UDDI, que serão abordados a seguir.

O *Simple Object Access Protocol* (SOAP) provê um padrão de estrutura para transporte de documentos XML, podendo ser utilizado através de diversos padrões de tecnologias utilizados na internet, como SMTP, HTTP e FTP. SOAP provê uma estrutura simples para a realização de troca de documentos através de chamadas de procedimentos remotos (RPC)[37, 40].

A *Web Service Description Language* (WSDL) é uma tecnologia XML que descreve a interface de um *web service* de forma padronizada. WSDL padroniza como um *web service* representa o formato de invocação dos parâmetros de entrada e saída. Permite que clientes compreendam automaticamente como interagir com o *web service*[37, 40].

Universal Description, Discovery and Integration (UDDI) provê um amplo registro dos *web services* para a sua divulgação, descoberta e integração. Essa tecnologia é muito útil, pois simplifica e torna possível a localização de *web services* por nome, identificadores, categorias ou qualquer especificação implementada pelo *web service*[37, 40].

4.1.6 Apache AXIS

O projeto *Apache Extensible Interaction System* (AXIS) é a terceira geração de processadores SOAP com código aberto. É um *framework* completo para a construção e instalação de transações XML usando SOAP[13].

É utilizado basicamente de duas formas[41]:

- **Cliente**: Fornece uma API para invocar os serviços do SOAP RPC realizando a troca de mensagens pelo protocolo SOAP.

- **Servidor**: Como biblioteca para solicitar os serviços SOAP de outra máquina, ou executar os serviços acessíveis.

As tecnologias *Java Development Kit* (JDK), Apache Tomcat e Apache AXIS, proporcionam todo o ambiente necessário para o desenvolvimento de *web services* que serão utilizados nos experimentos deste trabalho.

4.1.7 PostgreSQL

PostgreSQL[31] é um sistema gerenciador de banco de dados, código aberto, compatível com Plataformas Unix e Windows. É distribuído sob a licença *Berkeley Software Distribution* (BSD) clássica, que possibilita que os usuários tenham total liberdade de manipulação do código fonte.

Possui funcionalidades de sistemas relacionais, como, transações, subconsultas, gatilhos, visões, integridade referencial, bloqueios, funções, inclusive funções para manipulação de dados XML e linguagem de programação procedural, chamada PL/pgSQL.

Esse SGBD foi escolhido por ser de amplo conhecimento da comunidade acadêmica e por possuir todas as funcionalidades necessárias para a realização dos experimentos neste trabalho.

4.2 Características dos Experimentos

O Repositório de Padrões possui um banco de dados que armazena todos os padrões de metadados que poderão ser utilizados pelo repositório, e um conjunto de softwares que irão retornar um arquivo de dados que deverá ser lido pela aplicação Cliente. O usuário preenche os dados nesse Cliente que grava os dados no Repositório de Metadados, conforme a descrição do padrão já visto na Seção 3.1.

Para que pudesse ser testado o Metapadrão em um ambiente multiplataforma e que pudesse ser utilizado por qualquer linguagem, foram realizados experimentos que utilizaram as tecnologias Java, Apache Tom Cat 5.5, Apache AXIS(*Web Services*), XML, XSLT e PostgreSQL 8.2 com funções de manipulação de dados XML.

De acordo com a especificação do Metapadrão, sua base está na distribuição de recursos e alto nível de acessibilidade. Por essas razões, optou-se por desenvolver esse experimento baseado na tecnologia *web service*.

4.3 Repositório de Padrões

No Repositório de Padrões, o experimento consiste da modelagem do diagrama de classes descrito na Seção 3.3 em um banco de dados, como mostra o esquema de dados na Figura 4.4.

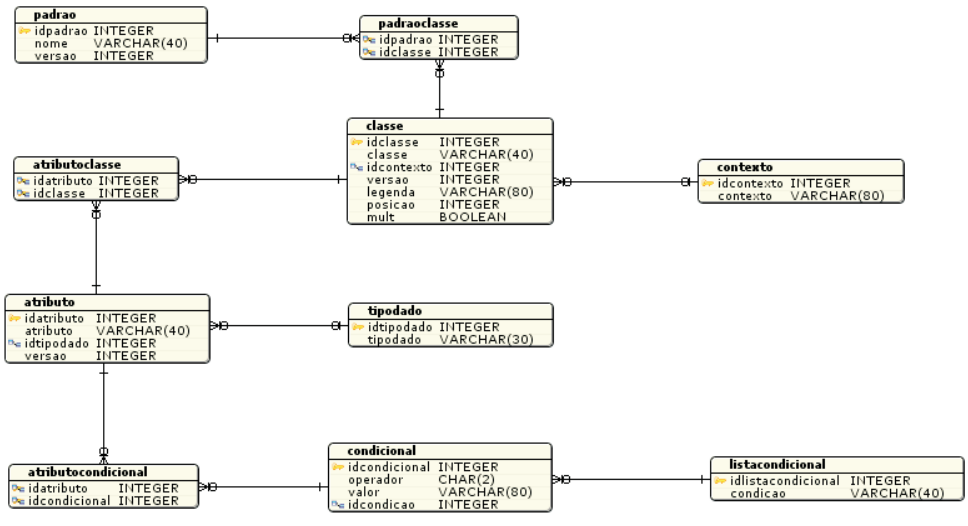

Figura 4.4: Esquema de dados do Repositório de Padrões.

Para o experimento, não foi desenvolvida nenhuma ferramenta de manipulação de dados no banco para uso do Administrador. As descrições dos padrões foram inseridas manualmente nas tabelas do banco de dados. Mesmo com o processo manual, foi possível constatar que os atributos e classes registrados podem ser reutilizados.

Para que os dados desse banco de dados possam ser acessados e recuperar a última versão disponível do padrão do interesse da aplicação cliente, o usuário deve requisitar ao *web service*, chamado Metapadrão, passando por parâmetro o padrão desejado. O *web service* Metapadrão faz uma consulta no banco de dados através do SQL descrito na Figura 4.5.

```
SELECT
   padrao.nome,
   padrao.versao as versaopadrao,
   classe.legenda,
   classe.classe,
   classe.mult,
   contexto.contexto,
   atributo.atributo,
   atributo.legenda as legatributo,
   tipodado.tipodado,
   condicao,
   operador,
   valor
FROM
   padrao
LEFT JOIN
   padraoclasse ON
   padraoclasse.idpadrao = padrao.idpadrao
LEFT JOIN
   classe ON
   padraoclasse.idclasse = classe.idclasse
LEFT JOIN
   contexto ON
   classe.idcontexto = contexto.idcontexto
LEFT JOIN
   atributoclasse ON
   classe.idclasse = atributoclasse.idclasse
LEFT JOIN
   atributo ON
   atributoclasse.idatributo = atributo.idatributo
LEFT JOIN
   tipodado ON
   atributo.idtipodado = tipodado.idtipodado
LEFT JOIN
   atributocondicional ON
   atributo.idatributo = atributocondicional.idatributo
LEFT JOIN
   condicional ON
   atributocondicional.idcondicional = condicional.idcondicional
LEFT JOIN
   listacondicional ON
   condicional.idcondicao = idlistacondicional
WHERE
   padrao.versao = (SELECT max(padrao.versao) FROM padrao WHERE padrao.idpadrao = 1) AND
   padrao.idpadrao = 1
ORDER BY
   classe.posicao;
```

Figura 4.5: Consulta responsável por resgatar a última versão do padrão solicitado ao *web service* Repositório de Padrões.

Com os resultados da pesquisa, o *web service* escreve um documento XML que

será retornado à aplicação Cliente. Esse documento XML descreve todos os atributos utilizados pelo padrão que foi solicitado pela aplicação Cliente, os tipos de dados e tamanho de cada atributo, quais atributos são obrigatórios e se existe algum tipo de regra de validação para os atributos listados. Esse documento XML pode ser visto na Figura 4.6 apresentando o padrão *Dublin Core*.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<dados_padrao>
      <padrao>dublin core</padrao>
      <versao>1</versao>
      <campos>
            <classe>tituloDescricao</classe>
            <legenda>Título </legenda>
            <mult>false</mult>
            <tipodado>string</tipodado>
            <condicoes>
                  <condicao>tamanho</condicao>
                  <valor>20</valor>
            </condicoes>
      </campos>
      <campos>
            <classe>criadorNome</classe>
            <legenda>Criador (Nome)</legenda>
            <mult>true</mult>
            <tipodado>string</tipodado>
            <condicoes>
                  <condicao>tamanho</condicao>
                  <valor>40</valor>
            </condicoes>
      </campos>
      <campos>
            <classe>assuntoDescricao</classe>
            <legenda>Assunto </legenda>
            <mult>false</mult>
            <tipodado>string</tipodado>
            <condicoes>
                  <condicao>tamanho</condicao>
                  <valor>20</valor>
            </condicoes>
      </campos>
      <campos>
            <classe>descricaoDescricao</classe>
            <legenda>Descrição </legenda>
            <mult>false</mult>
            <tipodado>string</tipodado>
            <condicoes>
                  <condicao>tamanho</condicao>
                  <valor>20</valor>
            </condicoes>
      </campos>
      <campos>
            <classe>publicadorNome</classe>
            <legenda>Publicador (Nome)</legenda>
            <mult>true</mult>
            <tipodado>string</tipodado>
            <condicoes>
                  <condicao>tamanho</condicao>
                  <valor>40</valor>
            </condicoes>
      </campos>
```

Continua...

Figura 4.6: XML gerado pelo *web service* do Repositório de Padrões para a construção da tela aplicação Cliente.

Nesse exemplo, pelo fato do padrão possuir diversos atributos e registrar todas as informações referentes ao padrão, tem-se o retorno de como as aplicações Cliente devem proceder ao receber esses dados e processá-los para montar a interface da aplicação Cliente.

4.4 Cliente

O experimento feito para testar uma aplicação Cliente foi desenvolvido de forma genérica, não sendo utilizado para apenas um padrão, mas podendo ser aplicado a qualquer padrão registrado no Repositório de Padrões. Através de um Servlet chamado ClienteServlet, que requisita o *web service* Metapadrão e passa por parâmetro, o padrão que deve montar a interface de registro.

Nesse caso, por se tratar de um Cliente genérico, a aplicação possui as características mínimas para registrar dados a respeito de qualquer padrão de metadados armazenado no repositório de modelos que, ao requisitar o padrão e receber um arquivo XML com os dados para montar a tela de cadastro, executa uma transformação com um arquivo XSLT chamado padrao.xsl, apresentando uma tela com os características descrita no XML retornado pelo *web service* Metapadrão, como pode ser visto na Figura 4.7.

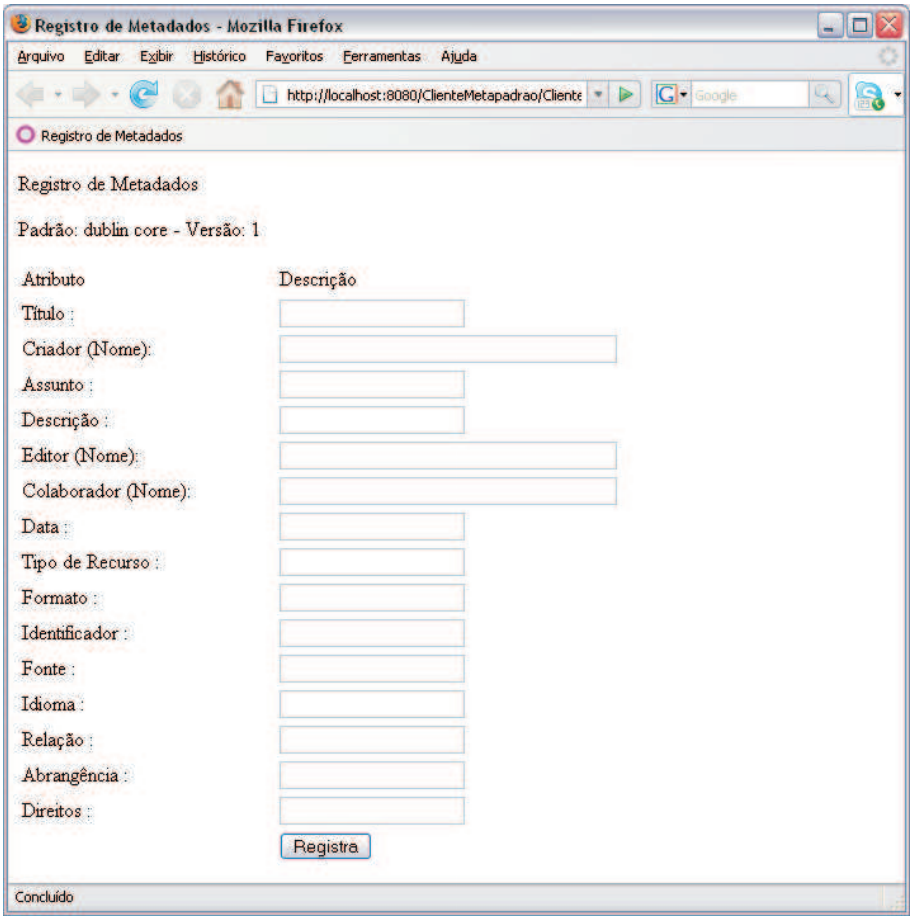

Figura 4.7: Tela montada a partir do XML retornado do *web service* do Repositório de Padrões

Ao preencher os dados na interface de Registro de Metadados da ClienteServlet, a aplicação requisita um *web service* chamado Metadado, que registra os metadados em uma base de dados PostgreSQL em formato XML e apresenta uma mensagem da situação do registro, se possível ou não registrar os dados no repositório, como mostra na Figura 4.8.

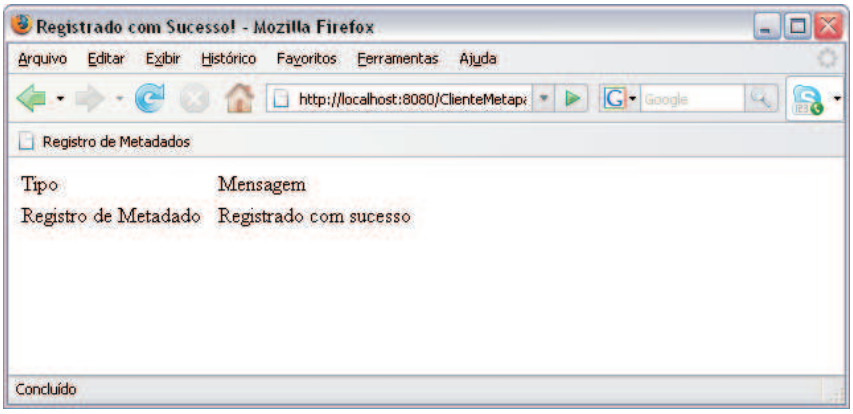

Figura 4.8: Tela de resultado do registro de metadados da aplicação Cliente.

Nesse momento os dados preenchidos no formulário foram registrados no banco de dados. No Metapadrão não foi especificado o formato para essa base de dados nem diagrama de classes, pois se trata de detalhes de implementação, nesse experimento optou-se por utilizar banco de dados.

4.5 Repositório de Metadados

Na fase final do experimento, foi utilizado outro *web service* chamado Metadados e, para o registro dos dados armazenados e a busca por metadados, optou-se em utilizar o armazenamento em banco de dados PostgreSQL e instalar funções de manipulações de dados XML. Essa decisão baseou-se no fato de já se possuir o ambiente com banco de dados e por já dispor do recurso de manipulação de dados XML dentro das tabelas.

No experimento, ao usuário clicar no botão registra na aplicação ClienteServlet, são coletados todos os dados preenchidos no formulário e um arquivo XML é montado, como pode ser visto na Figura 4.9, e requisita o *web service* Metadados passando por parâmetro esse XML para ser armazenado na tabela do banco de dados pelo *web service*.

```
<metadado>
   <hpadrao>dublin core</hpadrao>
   <hversao>1</hversao>
   <tituloDescricao>Metadados - Relacionados e Integrados</tituloDescricao>
   <criadorNome>Alcione Benacchio</criadorNome>
   <assuntoDescricao>Metadados</assuntoDescricao>
   <descricaoDescricao>O artigo esclarece as duvidas mais frequentes
   sobre Metadados</descricaoDescricao>
   <editorNome>Revista UEPG</editorNome>
   <colaboradorNome>Maria Salete Marcon Gomes Vaz</colaboradorNome>
   <dataData>10/10/2008</dataData>
   <tipoderecursoDescricao>Eletronico</tipoderecursoDescricao>
   <idiomaDescricao>Portugues</idiomaDescricao>
   <abrangenciaDescricao>Nacional</abrangenciaDescricao>
</metadado>
```

Figura 4.9: XML com os dados registrados pela aplicação Cliente no banco de dados.

O *web service* Metadados insere os dados XML na tabela dados, e a partir daí os dados podem ser acessados por consultas SQL utilizando as funções XML do PostgreSQL, como mostra na Figura 4.10.

```
SELECT
       id,
       datacriacao,
       padrao,
       versao,
       metadado
FROM
       metadados.dados
WHERE
       XPATH_STRING(metadado, 'idioma') = 'Portugues';
```

Figura 4.10: Seleção de dados para localizar informações dentro metadados em dados XML.

Nesse exemplo de busca apresentado na Figura 4.10, pode ser visto como utilizar a função "XPATH STRING"do PostgreSQL para localizar todos os metadados que são escritos em português.

Nesse experimento, foi apresentado o ciclo completo de utilização da arquitetura proposta pelo Metapadrão, mostrando que não apenas é funcional, mas que se forem utilizadas as tecnologias adequadas pode ser relativamente simples de ser implementado.

4.6 Teste de Extensibilidade

Além do padrão *Dublin Core*, foi registrado um modelo simplificado para armazenamento de metadados de dados pessoais. Inicialmente o modelo registrava apenas o Nome Completo, Departamento, Cargo e E-mail. Para avaliar a possibilidade de estender padrões no Metapadrão, o teste consiste em criar uma nova versão do padrão Dados Pessoais.

Como foi demonstrado no experimento anterior, a interface da aplicação Cliente é gerada a partir de um documento XML retornado por uma requisição de um *web service*. Ao requisitar ao *web service* o padrão 2 (modelo de Dados Pessoais), na versão 1, esse *web service* retornava o XML descrito na Figura 4.11.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<dados_padrao>
      <padrao>dados pessoais</padrao>
      <versao>1</versao>
      <campos>
            <classe>nomeNome</classe>
            <legenda>Nome Completo (Nome)</legenda>
            <mult>false</mult>
            <tipodado>string</tipodado>
            <condicoes>
                  <condicao>tamanho</condicao>
                  <valor>40</valor>
            </condicoes>
      </campos>
      <campos>
            <classe>dptoDescricao</classe>
            <legenda>Departamento </legenda>
            <mult>false</mult>
            <tipodado>string</tipodado>
            <condicoes>
                  <condicao>tamanho</condicao>
                  <valor>20</valor>
            </condicoes>
      </campos>
      <campos>
            <classe>cargoDescricao</classe>
            <legenda>Cargo </legenda>
            <mult>false</mult>
            <tipodado>string</tipodado>
            <condicoes>
                  <condicao>tamanho</condicao>
                  <valor>20</valor>
            </condicoes>
      </campos>
      <campos>
            <classe>emailDescricao</classe>
            <legenda>E-Mail </legenda>
            <mult>false</mult>
            <tipodado>string</tipodado>
            <condicoes>
                  <condicao>tamanho</condicao>
                  <valor>20</valor>
            </condicoes>
      </campos>
</dados_padrao>
```

Figura 4.11: XML gerado pelo *web service* para Dados Pessoais versão 1

Essa nova versão deveria ser capaz de registrar o nome e sobrenome das pessoas

de forma separada. Como foi optado em criar um novo padrão, no banco de dados foi adicionado um novo registro na entidade Padrão, com o nome do padrão de Dados Pessoais sendo registrado como a versão 2. Em seguida, os atributos que estavam associados à versão 1 foram associados também à versão 2. Optou-se por essa solução para que fosse possível manter a versão 1 funcional. Após ser feita a associação dos metadados utilizados pelo modelo Dados Pessoais versão 1, foi criada uma nova Classe e associada com outros registros em Atributos para que pudesse representar o nome e o sobrenome em atributos separados.

Após a criação do novo padrão, quando a aplicação Cliente solicita ao *web service* o padrão de Dados Pessoais, ele realiza uma busca pela última versão do padrão, que agora é a versão 2, e monta o XML de acordo a Figura 4.12.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<dados_padrao>
      <padrao>dados pessoais</padrao>
      <versao>2</versao>
      <campos>
            <classe>nomeNome</classe>
            <legenda>Nome Completo (Nome)</legenda>
            <mult>false</mult>
            <tipodado>string</tipodado>
            <condicoes>
                  <condicao>tamanho</condicao>
                  <valor>40</valor>
            </condicoes>
      </campos>
      <campos>
            <classe>nomeSobrenome</classe>
            <legenda>Nome Completo (Sobrenome)</legenda>
            <mult>false</mult>
            <tipodado>string</tipodado>
            <condicoes>
                  <condicao>tamanho</condicao>
                  <valor>40</valor>
            </condicoes>
      </campos>
      <campos>
            <classe>dptoDescricao</classe>
            <legenda>Departamento </legenda>
            <mult>false</mult>
            <tipodado>string</tipodado>
            <condicoes>
                  <condicao>tamanho</condicao>
                  <valor>20</valor>
            </condicoes>
      </campos>
      <campos>
            <classe>cargoDescricao</classe>
            <legenda>Cargo </legenda>
            <mult>false</mult>
            <tipodado>string</tipodado>
            <condicoes>
                  <condicao>tamanho</condicao>
                  <valor>20</valor>
            </condicoes>
      </campos>
      <campos>
            <classe>emailDescricao</classe>
            <legenda>E-Mail </legenda>
            <mult>false</mult>
            <tipodado>string</tipodado>
            <condicoes>
                  <condicao>tamanho</condicao>
                  <valor>20</valor>
            </condicoes>
      </campos>
</dados_padrao>
```

Figura 4.12: XML gerado pelo *web service* para Dados Pessoais versão 2

A versão 1, como foi citado anteriormente, registrava apenas Nome Completo, Departamento, Cargo e E-mail; quando a aplicação Cliente solicitava o padrão 2, ele apresentava a tela conforme mostra na Figura 4.13.

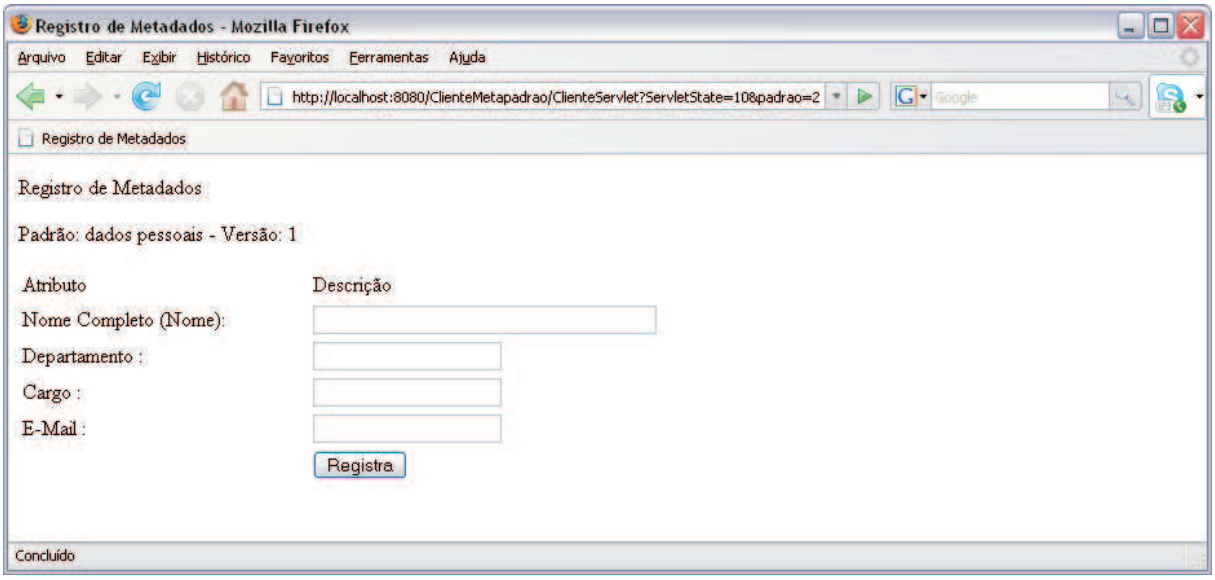

Figura 4.13: Tela gerada pela aplicação Cliente para Dados Pessoais versão 1

Depois da adição do novo padrão de Dados Pessoais, com o atributo Sobrenome associado à classe de Nome Completo, a tela gerada pela aplicação Cliente possui o atributo Sobrenome junto à classe Nome Completo, como pode ser visto na Figura 4.14.

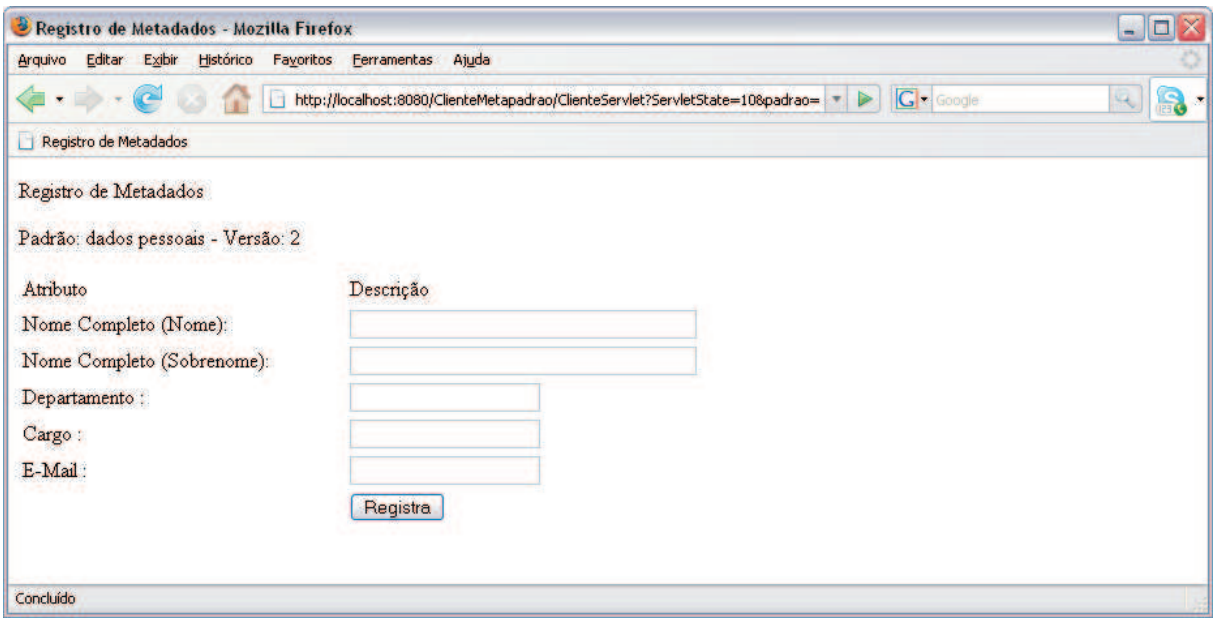

Figura 4.14: Tela gerada pela aplicação Cliente para Dados Pessoais versão 2

Ao serem registrados os dados pela aplicação Cliente utilizando o padrão de metadados Dados Pessoais versão 1, o XML gerado e registrado no Repositório de Metadados tinha o formato apresentado na Figura 4.15.

```
<metadado>
   <hpadrao>dados pessoais</hpadrao>
   <hversao>1</hversao>
   <nomeNome>Alcione Benacchio</nomeNome>
   <dptoDescricao>Sistemas de Informacao</dptoDescricao>
   <cargoDescricao>Professor</cargoDescricao>
   <emailDescricao>benacchio@gmail.com</emailDescricao>
</metadado>
```

Figura 4.15: XML armazenado no Repositório de Metadados - Dados Pessoais versão 1

Após criada a nova versão, a aplicação Cliente atualiza o formato e registra os metadados no Repositório de Metadados com o atributo Sobrenome separadamente do Nome, como descrito na Figura 4.16.

```
<metadado>
   <hpadrao>dados pessoais</hpadrao>
   <hversao>2</hversao>
   <nomeNome>Alcione</nomeNome>
   <nomeSobrenome>Benacchio</nomeSobrenome>
   <dptoDescricao>Sistemas de Informacao</dptoDescricao>
   <cargoDescricao>Professor</cargoDescricao>
   <emailDescricao>benacchio@gmail.com</emailDescricao>
</metadado>
```

Figura 4.16: XML armazenado no Repositório de Metadados - Dados Pessoais versão 2

4.7 Considerações Finais

Neste capítulo foram apresentadados os experimentos para avaliar o Metapadrão. Esses experimentos tiveram três objetivos: avaliar a arquitetura, verificar a implementação do Metapadrão de acordo com a especificação e avaliar a extensibilidade de padrões de metadados.

A arquitetura implementada utiliza o banco de dados PostgreSQL, uma aplicação *web* (Servlet) que representa um cliente genérico e dois *web services* que fazem a interface com o repositório de padrões e o repositório de metadados. Comprovou-se que a arquitetura do Metapadrão, além de funcional, pode ser simples de ser implementada

com tecnologias adequadas. Também foi possível verificar que os padrões registrados no Repositório de Padrões podem ser estendidos e as aplicações que dependem desse padrão não serão afetadas pela sua modificação.

CAPÍTULO 5

TRABALHOS RELACIONADOS

Neste capítulo são abordados os trabalhos Norma ISO/IEC 11179 e *Meta Object Facility* (MOF), os quais possuem relação com o estudo de padronização e gestão de metadados, com especificações de repositórios de metadados e interfaces para troca de informações entre repositórios.

5.1 Norma ISO/IEC 11179

Criada pela International Standard Organization (ISO) e International Electrotechnical Commission (IEC), a Norma ISO/IEC 11179 descreve um repositório de metadados, Metadata Registries (MDR), para ser utilizada por grandes organizações, para padronizar a criação de repositórios de metadados e a forma como esses metadados são gerenciados pelas organizações. Essa norma é dividida em seis partes, as quais são *Framework*, Classificação, Metamodelo e atributos básicos, Formulação e definição de dados, Nomeação e Identificação de princípios, Registro [17, 19].

5.1.1 Benefícios e Funcionalidades

O MDR proposto pela Norma ISO/IEC 11179 registra o significado dos dados e como esses dados são representados e descritos. A partir dessas descrições pode ser compreendido como esses dados estão contextualizados e principalmente garantir que um tipo de metadados que já foi criado não seja duplicado [18, 20].

A norma pode ser aplicada para obter os seguintes resultados:

- Padronização na descrição de dados;
- Entendimento claro dos dados por meio dos elementos organizacionais e entre as organizações;

- Reuso dos dados através do tempo, espaço e aplicações;
- Padronização dos dados dentro de uma organização e através das organizações;
- Gestão e reuso dos componentes de dados.

Um MDR baseado na norma deve prover funcionalidades para o processo de registro de metadados, baseadas em três aspectos [17], como mostra a Tabela 5.1:

Aspecto	**Descrição**
Identificação	Através de um identificador único para cada objeto registrado
Procedência	Oferece a fonte dos metadados e os objetos descritos
Monitoramento	Garante que os metadados façam o trabalho para o qual foram projetados

Tabela 5.1: Aspectos esperados em um repositório de metadados baseado na Norma ISO/IEC 11179

5.1.2 Gerenciamento dos Metadados

Nessa norma, os Metadados são chamados de Itens Administrados. O Item Administrado é submetido por uma Organização e recebe um identificador único que irá distingui-lo dos demais. Uma Autoridade de Registro irá armazená-lo, e outra Organização será incumbida de administrá-lo.

No registro de metadados do proprietário, terá como parte do registro de administração dois diferentes *status*, sendo um o armazenamento, que indica em que situação no ciclo de vida está o registro do Item Administrado registrado, e outro, o administrativo, que indica em que situação está no processo de registro da Autoridade de Registro.

Os pedidos de registro recebem um *status* que é definido pelas autoridades de registro, que é uma organização que opera e gerencia um registro de metadados, a fim de armazenar Itens Administrados. Mas, além disso, um Item Administrado irá passar por vários *status* durante o seu ciclo de vida, esses níveis são avaliados pela Autoridade de Registro e pela Organização Responsável [20].

A Autoridade de Registro deve ser aquela organização que deseja operar e gerenciar o MDR. É vista como qualquer organização e poderá assim receber os pedidos de registros de Itens Administrados.

Uma Autoridade de Registro é composta por três membros: o Comitê Executivo, o Comitê de Controle e o Registrador. Cada um deles desempenha um papel durante o ciclo de vida de um Item Administrado [20].

O Comitê Executivo é responsável por administrar as responsabilidades e poderes delegados pela Autoridade de Registro, incluindo, sobretudo, as políticas e a direção do negócio do MDR.

O Comitê de Controle provê as instruções técnicas e definição de defeitos técnicos associados ao Repositório de Metadados. Outros aspectos como estrutura, procedimentos e o grupo de membros do Comitê de Controle são definidos pela Autoridade de Registro.

O Registrador é o contato responsável por gerenciar e manter informações sobre os dados no Repositório de Metadados, que possui experiência nos processos de registro simplificando o registro dos Itens Administrados e disponibilizando a comunidade [20].

As entidades que fazem parte do processo de registro durante o ciclo de vida de um Item Administrado estão ilustradas na Figura 5.1. Entre elas pode-se destacar a Organização Responsável, designada para garantir a consistência dos Itens Administrados, relacionados e controlados pelas Organizações Fornecedoras e o Administrador que é o responsável pela precisão, confiança e uso dos metadados descritivos para Itens Administrados.

A Organização Fornecedora registra os Itens Administrados de acordo com os procedimentos estabelecidos pela Autoridade de Registro; Submissão, é autorizado a identificar e informar Itens Administrados adequados para registro; e Usuários de Leitura possuem acesso ao conteúdo do Repositório de Metadados, mas não é permitido realizar submissão, alterar ou apagar conteúdo.

Baseadas nessa norma, aplicações licenciadas foram desenvolvidas pela iniciativa privada e já estão disponíveis no mercado, como o *Data Foundations OneData Registry* [8] e o *Oracle Enterprise Metadata Manager (EMM)* [30]. Também é possível encontrar

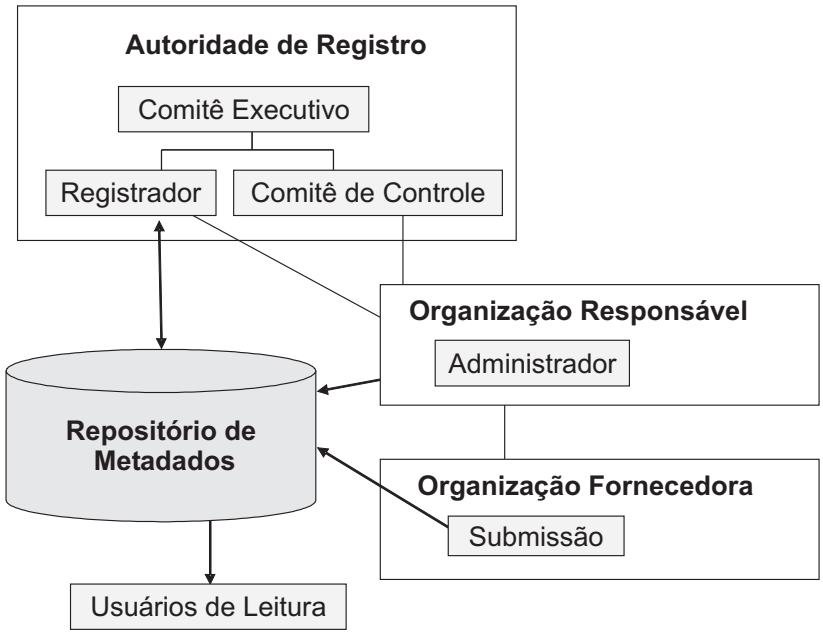

Figura 5.1: Norma ISO/IEC 11179 - Papéis e Responsabilidades [20]

implementações desenvolvidas por órgãos governamentais como o Instituto Australiano de Saúde e Bem-estar que desenvolveu o *Metadata Online Registry (METeOR)* [2] e o Instituto Canadense para a Informação da Saúde que desenvolveu o *Canadian Institute for Health Information (CIHI) Data Dictionary* [6]. Esses são repositórios baseados em padrões de dados relacionados à saúde, estatísticas de serviços para comunidade e informações, respectivamente.

5.1.3 Ciclo de Vida

A norma sugere alguns níveis para classificar os Itens Administrados durante o processo de registro. É durante esse processo que se garante a qualidade dos metadados. A qualidade é avaliada pelo Administrador e pelo Comitê de Controle, e é avaliado se um Item Administrado realmente faz o que ele se propõe a fazer [20].

- **Incompleto**: O Item Administrado registrado pode não conter todos os atributos obrigatórios.
- **Candidato**: O Item Administrado foi aceito para passar para os próximos níveis de registro.

- **Registrado**: A Autoridade de Registro confirma que todos os atributos obrigatórios do Item Administrado foram completados.
- **Qualificado**: A Autoridade de Registro confirma que todos os atributos obrigatórios do Item Administrado estão completos e que esses atributos estão em conformidade às exigências de qualidade aplicáveis.
- **Padrão**: A Autoridade de Registro confirma que o Item Administrado possui qualidade suficiente de acordo com o que se propõe e é de amplo interesse da comunidade que utiliza o Repositório de Metadados.
- **Padrão Indicado**: A Autoridade de Registro confirma que o Item Administrado é indicado para uso pela comunidade que utiliza o Repositório de Metadados.
- **Histórico**: Um Item Administrado Incompleto ou Candidato ao ser rejeitado recebe este nível para que a informação dessa submissão seja preservada.
- **Aplicação**: Ao receber esse nível, provavelmente um Item Administrado não irá mais progredir de nível. Mas o Administrador ou o Registrador poderá revisar o Item Administrado e, se considerar relevante o Item Administrado, poderá ser registrado como Candidato.
- **Aposentado**: O Item Administrado que é registrado com esse nível não deverá mais passar por alterações de níveis e não deve mais ser submetido a avaliações.
- **Substituído**: O Administrador e o Registrador irão periodicamente revisar Itens Administrados com o nível de Substituído para avaliar a possibilidade de atualizar esse item para o nível de Aposentado.

O processo de registro é iniciado a partir da identificação de um Item Administrado pela Organização Fornecedora. Durante a coleta de dados do Item Administrado, esse será classificado como Incompleto até que possua todos os dados necessários para a Submissão.

A Organização Fornecedora registra no Repositório de Metadados a Submissão de uma requisição de Item Administrado como Candidato e envia ao Administrador uma solicitação de revisão nos dados desse item.

Após avaliação, o Administrador irá registrar o nível e notificar à Organização Fornecedora sua decisão. Se aprovado, o item será classificado como Candidato, conforme pode ser visualizado na Figura 5.2, senão ficará apenas registrada no Histórico a submissão desse item.

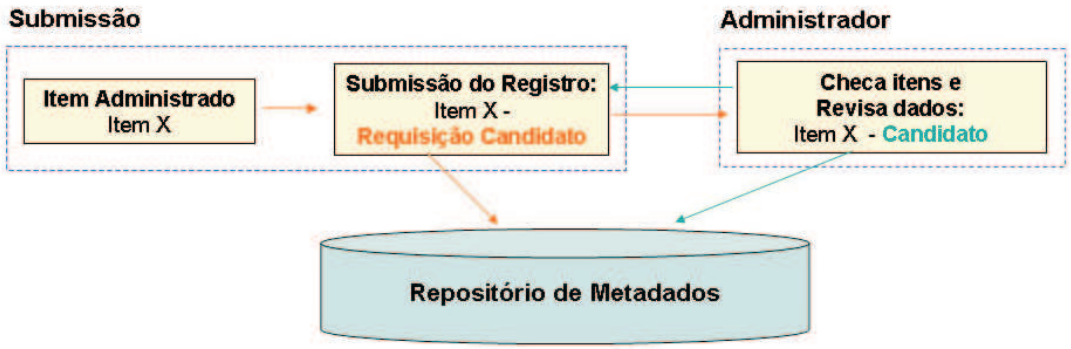

Figura 5.2: Submissão de padrão de metadados para avaliação como Candidato

Após receber a notificação do Administrador que o Item Administrado submetido foi classificado como Candidato, o responsável pela Submissão irá verificar se todos os atributos estão corretamente preenchidos e irá submeter um novo pedido ao Registrador para que o item receba o nível Registro.

O Registrador avalia a integridade e classifica o Item Administrado. Se o item foi classificado com o nível Registrado, o Registrador registra e notifica a Submissão, o resultado da sua avaliação, e que a partir de agora estará disponível no Repositório de Metadados, nesse caso, o Registrador também irá notificar o Administrador para que revise e avalie a qualidade desse item. Essa avaliação definirá se esse item será Qualificado Provisoriamente, como ilustra a Figura 5.3, ou Aposentado, ou seja, não irá mais mudar de nível. Caso o item seja rejeitado, ele será registrado no histórico.

Ao registrar um Item Administrado como Qualificado Provisoriamente, o Administrador notifica ao Registrador que avalia a qualidade do Item Proposto. Essa avaliação irá determinar se o Item Administrado será classificado como Qualificado ou Aposentado.

Figura 5.3: Padrão registrado no repositório e considerado como Registrado ou Qualificado Provisoriamente.

Essa alteração no processo hierárquico do Item Administrado é apresentada na Figura 5.4.

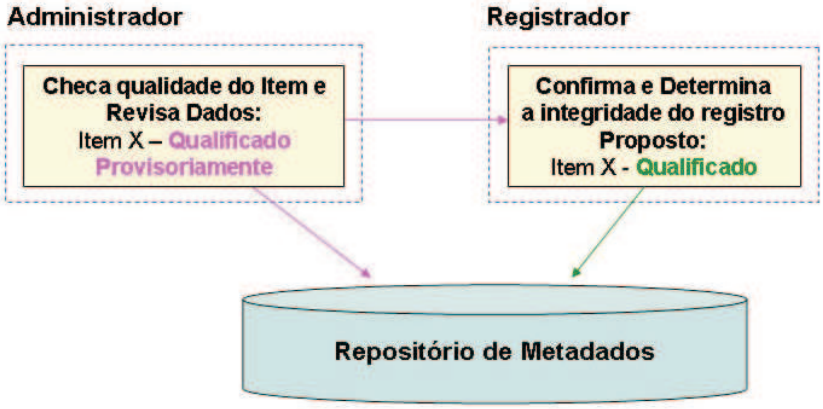

Figura 5.4: Padrão de metadados é classificado como Qualificado Provisoriamente ou Qualificado

Como pode ser visto na Figura 5.5, o Administrador e o Registrador irão revisar periodicamente Itens Administrados que possam progredir e avaliarão a possibilidade de que esse registro seja considerado um Padrão. O Administrador enviará ao Registrador os registros que forem considerados para Padrão, que serão registrados pelo Registrador como Padrão Provisório, e o Administrador irá enviar uma breve explicação por que ele acredita que esse registro deve ser considerado um padrão.

O Registrador irá revisar constantemente todos os Itens Administrados que estiverem registrados com o nível "Padrão Provisório", avaliar a possibilidade de tornar um padrão, fazer uma breve declaração das razões de por que ele acredita ser um padrão, e

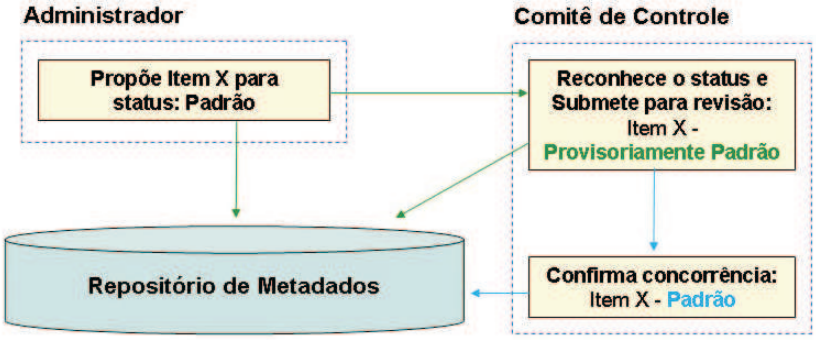

Figura 5.5: Provisoriamente Padrão ou Padrão

envia todos os Itens Administrados para avaliação do Comitê de Controle que, por sua vez, irá avaliar e definir caso a caso se o registro progredirá para o nível de Padrão.

Além desses níveis ilustrados nas Figuras 5.2 a 5.5, vários outros são sugeridos pela norma, esses níveis não são obrigatórios e devem ser definidos pela Autoridade de Registro, Comitê de Controle e Registrador, cabendo a eles definir os níveis mais indicados para cada Organização.

5.2 *Meta Object Facility* (MOF)

A especificação *Meta Object Facility* (MOF) foi desenvolvida pela Object Management Group (OMG), que é uma organização internacional não lucrativa, formada por um consórcio de empresas da indústria da computação, que centraliza seus esforços no desenvolvimento de instrumentos de integração de padrões para diversas áreas tecnológicas[29].

Focado no gerenciamento de metadados com extensibilidade, a fim de prover um *framework* que desse suporte a qualquer tipo de metadados e que permitisse a adição de novos tipos de metadados, essa especificação começou a ser elaborada em 1997 e sua principal finalidade foi definir:

- Uma linguagem abstrata para descrição de metamodelos
- Um modelo padrão de programação para gerenciamento de modelos instanciados de um metamodelo, padronizando suas interfaces de acesso a esses metadados.

O MOF define um padrão de interface que pode ser utilizado para facilitar a troca de informações entre ferramentas de warehouses, metadados de negócio, repositórios de metadados e ambientes distribuídos heterogêneos, provendo uma linguagem padronizada para a descrição de metamodelos.

5.2.1 Arquitetura de Metadados em Quatro Camadas

A arquitetura do *framework* MOF clássico é baseada em uma arquitetura de quatro camadas. Essas camadas são descritas a seguir[28]:

- **Informação**: Camada responsável por registrar a informação que se pretende descrever.
- **Modelo**: Camada para metadados que descrevem dados na camada de informação. Metadados agregados como modelos.
- **Metamodelo**: Camada utilizada nas descrições que definem a estrutura e semântica dos dados. Meta-metadados agregado como metamodelos. Um metamodelo é uma linguagem abstrata para descrever diferentes tipos de dados, ou seja, é uma linguagem sem sintaxe ou notação concreta.
- **Meta-metamodelo**: Camada responsável pela descrição da estrutura e semântica dos meta-metadados. É uma linguagem abstrata para definir outros tipos de metadados.

As camadas desse *framework* estão ilustradas na Figura 5.6. Descrevem como base a informação, seguido pelo modelo que a constrói e o metamodelo para descrever e o Meta-metamodelo.

5.2.2 Modelo MOF

O modelo MOF descreve o formato como os metamodelos são criados, por isso as características usadas para descrevê-lo são as mesmas definidas no modelo.

As classes, no modelo MOF, podem ser consideradas como a base desse modelo. Elas são especificadas de forma muito similar às Classes descritas pela notação UML.

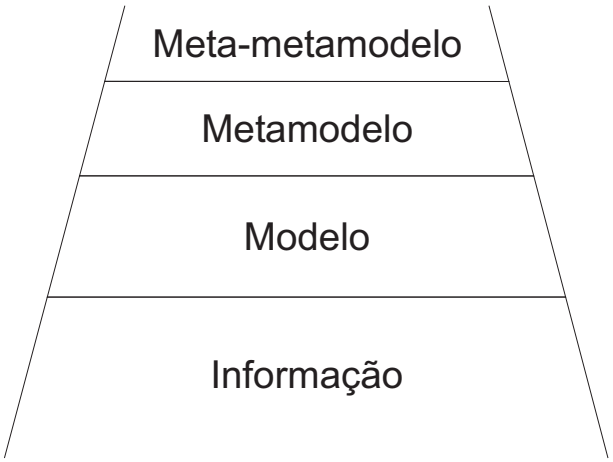

Figura 5.6: Arquitetura MOF em quatro camadas

Podem possuir três tipos de características; Atributos, Referências e Operações. Elas podem herdar de Classes e ter relação com outras classes por Associações.

- **Atributos**: O cabeçalho atributo lista os Atributos para as Classes no modelo MOF, sendo que estes Atributos herdam da super Classe e não são listados.
- **Referências**: Da mesma forma como o Atributo, o cabeçalho referência lista as Referências para a Classe no Modelo MOF. A Referência conecta o conteúdo da Classe a uma Associação que pertence a uma Associação que envolve as Classes, possibilitando ao cliente navegar diretamente de uma instância de Classe a outra instância que esteja relacionada pelos links de Associação.
- **Operações**: Todas as operações descritas no Modelo MOF devem possuir visibilidade pública.

Para esses casos citados acima, o cabeçalho é de vital importância, pois sem ele a Classe não identifica os Atributos, Referências ou Operações.

As associações descrevem os relacionamentos entre instâncias e classes. Uma associação pode relacionar duas classes, ou pode relacionar uma como sendo ela mesma, que define a ligação que existe entre duas instâncias de uma Classe.

5.2.2.1 Tipos De Dados

Os Tipos De Dados, utilizados no modelo MOF, são instâncias de outros tipos de dados que são descritos pelo modelo MOF, sendo eles: Primitivos, Multiplicidade, Visibilidade, Direção, Escopo, Agregação, Avaliação.

5.2.2.2 Exceções

As exceções descritas pelo Modelo MOF são geradas por interfaces que fazem uso dessas exceções, que são definidas e associadas a operações com base em critérios determinados durante a sua geração, sendo geradas principalmente duas exceções:

- **Nome Não Encontrado**: Essa exceção ocorre quando a busca por um nome falha.
- **Nome Não Resolvido**: Ocorre quando o nome existe, mas possui erros na especificação de parâmetros.

Suas características são:

- **Nome** - Capturar o nome do cabeçalho da seção
- **Parâmetro** - Que são capturados e passados aos cabeçalhos correspondentes.

5.2.2.3 Restrições

As restrições se aplicam à semântica do modelo MOF, são definidas no metamodelo e fazem parte desse modelo que são representados como parte requerida do modelo como os metaobjetos.

5.3 Análise Comparativa

A análise comparativa foi realizada considerando os aspectos compatíveis das especificações Norma ISO/IEC 11179, Meta Object Facility (MOF) e o Metapadrão. Com-

parando entre elas suas arquiteturas e a capacidade para solucionar os problemas comuns de gestão de metadados.

Na comparação da arquitetura, foram utilizados os critérios a seguir:

1. **Especificação de Modelagem**: Possui uma modelagem que descreve como seus dados devem ser armazenados.
2. **Classificação**: Esquema de classificação que define como os atributos de um padrão estão associados.
3. **Processo de Registro**: Definição de um processo padrão para o registro dos padrões de metadados.
4. **Ciclo de Vida**: Especificação do ciclo de vida para os padrões e para o registro dos metadados.
5. **Implementações homologadas**: Implementações que utilizaram a especificação para desenvolvimento de ambientes de gestão de metadados.
6. **Definição de armazenamento de dados**: Formato definido pela especificação para armazenamento dos metadados.
7. **Camadas**: Dentro de sua especificação possui um sistema que divide os problemas a serem resolvidos em camadas.
8. **Isolamento de Camadas**: As camadas não são fortemente ligadas, podendo ser implementadas de forma separada e podem ser substituídas sem que as outras camadas sejam afetadas.

	1	2	3	4	5	6	7	8
ISO/IEC 11179	x	x	x	x				
MOF	x	x			x	x	x	
Metapadrão	x	x	x	x		x	x	x

Tabela 5.2: Comparação entre as arquiteturas

Na Tabela 5.3, as linhas correspondem aos trabalhos analisados e as colunas, os critérios para problemas de gestão de metadados descritos na Seção 2.8.

Para comparar os problemas de gestão de metadados, foram utilizados os critérios detalhados na Seção 2.8.

	ISO/IEC 11179	MOF	Metapadrão
Variedade de Formas	x	x	x
Novos padrões de metadados	x	x	x
Extensão de padrões	x	x	x
Tipos diferentes de metadados	x	x	x
Usuários de metadados	x		x
Vocabulários de metadados	x		x
Mapeamento entre padrões	x		x

Tabela 5.3: Comparação dos problemas de gestão de metadados

As duas especificações abordadas são mais abrangentes em outros fatores que não correspondem entre si e com o Metapadrão. No caso da Norma ISO/IEC 11179, sua especificação serviu de inspiração para este trabalho e muitas vezes como guia para a tomada de decisões.

No entanto, a Norma ISO/IEC 11179 se mostra muito complexa e de difícil implementação, sendo que várias empresas e organizações fizeram implementações baseadas nessa norma, mas ainda não foram homologadas.

Em relação ao processo de administração dos dados, a padronização de metadados, segundo sua especificação, define diversos responsáveis. Isso cria um grande esforço para a organização, o que dificulta manter esse ambiente em uma organização de porte médio.

O *Meta Object Facility* (MOF) possui uma especificação muito detalhada sobre sua modelagem e a forma de sua implementação, mas não define um processo de registro e um ciclo de vida para os metadados.

Tanto o MOF quanto a Norma ISO/IEC 11179 não especificam o uso de um sistema de versionamento para os padrões de metadados. O Metapadrão propõe que seja utilizado um sistema de versionamento para as metaClasses e para os metaAtributos, possibilitando assim que padrões de metadados utilizados pelas aplicações não sofram no momento da atualização de um padrão.

No caso do formato de armazenamento dos dados, a Norma ISO/IEC 11179 não

especifica a tecnologia a ser utilizada. O Metapadrão especifica que o formato deve ser XML, enquanto o MOF define a utilização de Orientação a Objetos.

Outro ponto a ser destacado é o desenvolvimento em camadas, a Norma ISO/IEC 11179 não especifica tecnologia de desenvolvimento nem como o serviço do repositório deve ser disposto. Enquanto o MOF possui um sistema de camadas bem definido e estruturado, de forma parecida, o Metapadrão também possui uma especificação de desenvolvimento em camadas e propõe que as camadas sejam implementadas de forma isolada, diferentemente do MOF.

Uma desvantagem do Metapadrão em relação às outras especificações é o seu nível de maturidade. Ainda faltam diversos pontos a serem definidos para o Metapdrão que já estão definidos nas outras especificações, como, por exemplo, Formulação e definição de dados[16], Nomeação e Identificação de princípios[18], entre outros.

5.4 Considerações Finais

Neste capítulo foram apresentados os trabalhos relacionados, fazendo um levantamento das características de cada um deles que estão associadas a ambientes de gestão de metadados.

Realizou-se uma análise comparativa entre as especificações Norma ISO/IEC 11179, *Meta Object Facility* e o Metapadrão. Descreveu-se também como essas especificações funcionam, como estão modeladas e como foi definida sua arquitetura, apresentando suas vantagens e desvantagens.

CAPÍTULO 6

CONCLUSÕES E TRABALHOS FUTUROS

Através da utilização de metadados e padrões de metadados, é possível promover a integração, interpretação, localização e reutilização dos dados dentro de uma organização, assegurando que esses dados possam ser utilizados para pesquisas e recuperações futuras.

A arquitetura de repositório de padrões e metadados especificada nesta dissertação provê um repositório em que as funcionalidades para o gerenciamento e padronização dos dados dentro de uma organização são integradas, provendo ao Administrador um ambiente de gestão de metadados completo. Esse ambiente permite que os metadados registrados possam ser acessados e utilizados por pessoas e aplicações.

Pode ser ressaltado que o Metapadrão é mais apropriado para organizações que precisam padronizar e integrar seus dados sem precisar integrá-los com outras organizações. Das vantagens dessa arquitetura de repositório de metadados, podemos destacar a reusabilidade das aplicações criadas a partir dele, proporcionando em um único ambiente todas as funcionalidades necessárias para gerenciar, consultar e registrar dados sobre todos os padrões descritos.

Por outro lado, o padrão para sistema de registro de metadados proposto pela Norma ISO/IEC 11179 é mais indicado para organizações que controlam outras organizações e que precisam promover a integração dos dados entre elas, demandando um maior número de pessoas para a administração e gerência dos processos de registro.

6.1 Contribuições

Como contribuições do trabalho realizado, podemos citar:

1. Definição de um metamodelo genérico para padrões de metadados e modelagem de um banco de dados baseado nesse metamodelo;

2. Especificação de uma arquitetura de repositório genérico, dividida em camadas não acopladas;
3. Desenvolvimento de um ambiente, no qual os metadados registrados estejam centralizados, com o intuito de integrar todos os padrões utilizados pela organização;
4. Implementação de um web service para comunicação com o repositório de padrões e outro para comunicação com o repositório de metadados;
5. Implementação de um cliente genérico, que através dos web services, solicita os padrões e armazena os dados no repositório de metadados.

Este trabalho apresenta uma divisão, em três camadas, para a arquitetura de um repositório genérico, sendo a primeira camada responsável pelo repositório de padrões, a segunda, pelo repositório dos metadados, e a terceira, pela aplicação cliente.

Com camadas isoladas, não acopladas, o repositório de padrões é independente do repositório de metadados, e a aplicação cliente, através de uma interface de software, solicita ao repositório de padrões o padrão desejado e, conforme a especificação de cada padrão descrito, é capaz de registrar os dados no repositório de metadados.

Os padrões de metadados são organizados por um gestor de padrões chamado de Administrador e que, de acordo com as solicitações dos usuários de metadados da organização, é o responsável por avaliar os padrões que são registrados no repositório de padrões e garantir sua integração e reusabilidade.

Por não utilizar um sistema de camadas acoplado, também possibilita que outras aplicações façam a interface com a arquitetura proposta, tornando possível a comunicação com outros repositórios de metadados e outros clientes.

Outra contribuição dessa especificação é a flexibilidade para constantes extensões dos padrões utilizados pela organização, através do uso de versionamento dos mesmos. O uso de versões possibilita que novos padrões possam ser criados, a partir de outros, e

também podem ser estendidos sem qualquer impacto sobre os metadados já registrados, no repositório de metadados.

6.2 Trabalhos Futuros

Como perspectiva de trabalhos futuros, pretende-se desenvolver a especificação de um modelo de autenticação integrado, entre o repositório de padrões, o repositório de metadados e as aplicações clientes. Além de criar um sistema de validação de documentos XML que deve avaliar o formato do documento XML apresentado pela aplicação cliente, e checar se atende às especificações registradas no repositório de padrões.

Também realizar um aperfeiçoamento do cliente genérico, com funcionalidades para inserir dados múltiplos e validar as regras descritas pelos padrões. Como extensão futura, transformar os experimentos, apresentados nesta dissertação, em um protótipo com aplicações para gerenciamento do repositório de padrões e do repositório de metadados.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] Amiano, M., D´Cruz, C., Ethier, K., Thomas, M. D. *XML Problem-Design-Solution*. Wiley Publishing, 2006.

[2] Australian Institute of Health and Welfare. Metadata Online Registry - METeOR. http://meteor.aihw.gov.au, Última consulta em 28 de abril de 2007.

[3] Baldan, R. *EDMS: Gerenciamento Eletrônico de Documentos Técnicos*, volume 1. Érica, 2004.

[4] Brittain, J., Darwin, I. F. *Tomcat: The Definitive Guide*. O'Reilly, 2003.

[5] Burke, E. M. *Java and XSLT*. O'Reilly, 2001.

[6] Canadian Institute for Health Information. CIHI Data Dictionary. https://eservices.cihi.ca/ddexternal/welcome.do, Última consulta em 28 de abril de 2007.

[7] Chappell, D. Jewell, T. *Java Web Services*. O'Reilly, 2002.

[8] Data Foundations. Onedata-registry, 2005. http://datafoundations.com, Última consulta em 16 de julho de 2007.

[9] Deitel, H. M., Deitel, P. J., Nieto R. *XML: Como Programar*. Bookman, 2003.

[10] Dublin Core Metadata Initiative - DCMI. Dublin core metadata element set, 2007. http://www.dublincore.org/documents/dces, Última consulta em 16 de julho de 2007.

[11] Duval, E., Hodgins, W., Sutton, S., Weibel S. Metadata principles and practicalities. *D-Lib Magazine 8(4)*, 2002.

[12] Gamma, E., Helm, R., Johnson, R., Vlissides, John. *Padrões de Projeto*. Bookman, 2000.

[13] Goodwill J. *Apache Axis Live: A Web Services Tutorial*. SourceBeat, LLC, 2004.

[14] Hunter, J., Crawford, W. *Java Servlet Programming*. O'Reilly, 2001.

[15] International Organization for Standardization - ISO/IEC. *Information technology - Metadata registries (MDR) - Registry metamodel and basic attributes*. http://metadata-standards.org/11179/, Última consulta em 30 de Janeiro de 2008, 2003.

[16] International Organization for Standardization - ISO/IEC. *Information technology - Metadata registries (MDR) - Formulation of Data Definitions*. http://metadata-standards.org/11179/, Última consulta em 30 de Janeiro de 2008., 2004.

[17] International Organization for Standardization - ISO/IEC. *Information technology - Metadata registries (MDR) - Framework*. http://metadata-standards.org/11179/, Última consulta em 30 de Janeiro de 2008., 2004.

[18] International Organization for Standardization - ISO/IEC. *Information technology - Metadata registries (MDR) - Naming and identification principles*. http://metadata-standards.org/11179/, Última consulta em 30 de Janeiro de 2008., 2004.

[19] International Organization for Standardization - ISO/IEC. *Information technology - Metadata registries (MDR) - Classification*. http://metadata-standards.org/11179/, Última consulta em 30 de Janeiro de 2008., 2005.

[20] International Organization for Standardization - ISO/IEC. *Information technology - Metadata registries (MDR) - Registration*. http://metadata-standards.org/11179/, Última consulta em 30 de Janeiro de 2008, 2005.

[21] Larsen, G. *Designing component-based frameworks using patterns in the UML*. Communications of the ACM, 42(10):38-45, October 1999.

[22] Marchal, B. *XML by Example*. QUE, 2000.

[23] Metadata Encoding and Transmission Standard - METS. http://www.loc.gov/standards/mets - Última Consulta em 20 de Junho de 2007.

[24] Mit Libraries. Metadata Mappings (Crosswalks), 2007. http://libraries.mit.edu/guides/subjects/metadata/mappings.html, Última Consulta em 21 de julho de 2007.

[25] Moving Picture Experts Group - MPEG. Mpeg-7 overview, 2004. http://www.chiariglione.org/mpeg/standards/mpeg-7/mpeg-7.htm, Última Consulta em 18 de Julho de 2007.

[26] National Information Standards Organization - NISO. Understanding metadata, 2004.

[27] Newcomer, E. *Understanding Web Services- XML, WSDL, SOAP and UDDI.* Addison-Wesley, 2002.

[28] OMG - Object Management Group. Meta object facility (mof) specification - 1.4. http://www.omg.org/docs/formal/02-04-03.pdf, Última Consulta em 30 de Janeiro de 2008, 2002.

[29] OMG - Object Management Group. Meta object facility (mof) 2.0 core proposal. http://www.omg.org/docs/ad/03-04-07.pdf, Última Consulta em 30 de Janeiro de 2008, 2003.

[30] Oracle. Oracle enterprise metadata manager - emm, 2004. http://www.oracle.com, Última Consulta em 16 de julho de 2007.

[31] PostgreSQL. Documentation. http://www.postgresql.org/docs/ - Última Consulta em 1 de Fevereiro de 2008.

[32] Ray, E. T. *Creating self-data describind - Learning XML.* O'Reilly, 2001.

[33] Sun Microsystems. Fundamentals of java servlets. http://java.sun.com/developer/onlineTraining/Servlets/Fundamentals/ - Última Consulta em 1 de Fevereiro de 2008, 1999.

[34] Tannenbaum, A. *Metadata Solutions: Using Metamodels, Repositories, XML and Enterprise Portals to Generate Information on Demand*. Addison Wesley, 2002.

[35] The Apache Software Foundation. Apache tomcat. http://tomcat.apache.org/, Última Consulta 1 de fevereiro de 2008.

[36] The J. Paul Getty Trust. Metadata Standards Crosswalks, 2008. http://www.getty.edu/research/conducting_research/standards/intrometadata/dc_ead.html, Última Consulta em 23 de abril de 2008.

[37] Tidwell, D. Snell, J., Kulchenko, P. *Programming Web Services with SOAP*. O'Reilly, 2001.

[38] Vaz, M. S. M. G. *MetaMídia - Um Modelo de Metadados na Indexação e Recuperação de Objeto Multimídia*. Tese de Doutorado, UFPE, 2000.

[39] Vetterli, T., Vaduva, A., Staudt, M. Metadata standards for data warehousing: Open information model vs. common warehouse metamodel. *ACM SIGMOD Record. ACM Press.*, 2000.

[40] W3C Working Group. Web services architecture. http://www.w3.org/TR/ws-arch/ - Última Consulta em 1 de Fevereiro de 2008, 2004.

[41] W3C Working Group. Simple object access protocol (soap). http://www.w3.org/TR/soap/ - Última Consulta em 1 de Fevereiro de 2008, 2007.

[42] World Wide Web Consortium. Extensible markup language (xml). http://www.w3.org/TR/2004/REC-xml-20040204 - Última Consulta em 1 de Fevereiro de 2008, 2004.

[43] World Wide Web Consortium. Extensible stylesheet language (xsl) version 1.1. http://www.w3.org/TR/xsl/ - Última Consulta em 1 de Fevereiro de 2008, 2006.