

Pablo Alexandre Roberto

**Um Arcabouço Baseado em Componentes, Serviços
Web e Arquivos Abertos para Construção de
Bibliotecas Digitais**

Belo Horizonte

31 de março de 2006

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

**Um Arcabouço Baseado em Componentes, Serviços
Web e Arquivos Abertos para Construção de
Bibliotecas Digitais**

Dissertação apresentada ao Programa de
Pós-Graduação em Ciência da Computação
da Universidade Federal de Minas Gerais
como requisito parcial para a obtenção do
grau de Mestre em Ciência da Computação.

Pablo Alexandre Roberto

Belo Horizonte

31 de março de 2006



UNIVERSIDADE FEDERAL DE MINAS GERAIS

FOLHA DE APROVAÇÃO

Um Arcabouço Baseado em Componentes, Serviços Web e
Arquivos Abertos para Construção de Bibliotecas Digitais

PABLO ALEXANDRE ROBERTO

Dissertação defendida e aprovada pela banca examinadora constituída por:

Ph. D. MARCOS ANDRÉ GONÇALVES – Orientador
Universidade Federal de Minas Gerais

Ph. D. ALBERTO HENRIQUE FRADE LAENDER – Co-orientador
Universidade Federal de Minas Gerais

Ph. D. RICARDO DA SILVA TORRES
Universidade Estadual de Campinas

Ph. D. ILMÉRIO REIS DA SILVA
Universidade Federal de Uberlândia

Belo Horizonte, 31 de março de 2006

Resumo

Bibliotecas digitais constituem uma das formas mais avançadas e complexas de sistemas de informação. Dada sua complexidade, a construção desse tipo de sistema normalmente demanda um projeto de software com um longo ciclo de desenvolvimento. A componentização de bibliotecas digitais é uma abordagem que permite a redução do tempo deste ciclo, através de conceitos como modularidade e reusabilidade. Neste trabalho apresentamos o arcabouço WS-ODL, baseado em um conjunto de componentes, serviços web e arquivos abertos, para construção de bibliotecas digitais. O arcabouço opera sobre a arquitetura Fedora, que atua como um repositório de objetos digitais complexos e um provedor de serviços de infra-estrutura. Dentre as contribuições do nosso trabalho, destacam-se a criação de componentes com funções complexas, como busca combinando dados estruturados e dados textuais, e uma avaliação experimental com usuários reais. Realizamos uma ampla experimentação do arcabouço com várias coleções e mostramos que o mesmo é viável, apresentando desempenho superior ao arcabouço ODL – *Open Digital Libraries*, que serviu como ponto de partida para este trabalho.

Abstract

Digital libraries constitute one of the most advanced and complex types of information system. Due to this complexity, the construction of such a system normally demands a software project with an intense development cycle. Digital library componentization is an approach that allows decreasing this cycle's time, through concepts like modularity and reusability. In this work, we present the WS-ODL framework, based on a pool of components, web services and open archives, for building digital libraries. Such framework operates on the top of the Fedora architecture, which works like a complex digital objects repository and an infrastructure services provider. Amongst the contributions of our work, we distinguish the development of components with complex functions, like search integratind structured and textual data, and an experimental evaluation with real users. We provide an extensive evaluation of the framework with some collections e show that it is feasible and presents superior performance than ODL – Open Digital Libraries, the starting point of our work.

Agradecimentos

Agradeço primeiramente a Deus, por mais esta etapa de minha vida.

Aos meus pais, Ilze e Antonio, pelo apoio e incentivo incondicionais.

À minha noiva Poliana, pelo seu amor, dedicação e compreensão, principalmente nos momentos de minha ausência.

Ao meu irmão Gladyston, por me motivar a seguir em frente.

Aos meus orientadores, Marcos Gonçalves e Alberto Laender, pelo excelente trabalho de orientação.

A todo o pessoal do Laboratório de Bancos de Dados, pela sinergia que mantemos, sempre com muita alegria e descontração.

Aos meus amigos, pelo apoio e amizade sinceros, e pelos bons momentos de lazer fundamentais em nossas vidas.

Ao Laboratório Synergia/DCC e ao CNPq, pelo suporte financeiro.

A todos aqueles que, direta ou indiretamente, participaram e me apoiaram no desenvolvimento deste trabalho.

Sumário

1	Introdução	1
2	Conceitos e Trabalhos Relacionados	6
2.1	Abordagem Baseada em Componentes	6
2.2	Serviços Web	8
2.3	Arquivos Abertos	9
2.3.1	O Protocolo OAI	10
2.3.2	<i>Dublin Core</i>	12
2.4	A Arquitetura Fedora	13
2.4.1	Perspectiva Representacional	15
2.4.2	Perspectiva Funcional	16
2.4.3	Relacionamentos entre Objetos	17
2.5	Arcabouços para Construção de Bibliotecas Digitais	18
2.6	Mecanismos para Recuperação de Informação em Bancos de Dados . .	20
3	Descrição do Arcabouço	22
3.1	Esquema Geral de Funcionamento dos Componentes do Arcabouço . .	24

3.2	Componente de Exibição de Trabalhos Recentes	25
3.3	Componentes de Navegação	27
3.3.1	Componente Baseado em Serviços Web	28
3.3.2	Componente de Navegação Multidimensional	29
3.4	Componente de Busca	32
3.5	Componentes de Registro de Usuário e de Anotação	34
3.6	Componentes de Administração	35
4	Exemplos de Utilização	38
5	Avaliação e Análise do Arcabouço	44
5.1	Experimento de Instalação	44
5.2	Experimento de Carga de Dados	50
5.3	Experimento de Desempenho	53
6	Conclusões	58
	Referências Bibliográficas	61
	Roteiro de Instalação do Arcabouço WS-ODL	65

Lista de Figuras

2.1	Processo de colheita baseado no protocolo OAI	11
2.2	Visão representacional de um objeto do Fedora	16
2.3	Datastream relations	18
3.1	Visão geral do arcabouço WS-ODL	24
3.2	Esquema geral de funcionamento dos componentes	24
3.3	Exemplo de uma requisição SOAP	26
3.4	Exemplo de uma resposta SOAP	26
3.5	Exemplo de fragmento do metamodelo do componente de navegação multi- dimensional	30
3.6	Exemplo do esquema relacional do banco de dados do componente de na- vegação multidimensional (antes da modificação)	31
3.7	Exemplo do esquema relacional do banco de dados do componente de na- vegação multidimensional (depois da modificação)	31
3.8	Exemplo do banco de dados do componente de Busca	33
3.9	Exemplo de uma consulta SQL gerada pelo Componente de Busca	34
3.10	Componentes de administração	37
4.1	Página de trabalhos recentes	38

4.2	Página de trabalhos da coleção	39
4.3	Página de trabalhos do autor	40
4.4	Página inicial do componente de navegação multidimensional	40
4.5	Página da coleção – componente de navegação multidimensional	41
4.6	Consulta integrando dados estruturados e dados textuais	42
4.7	Registro de usuário e anotações do trabalho	43

Lista de Tabelas

2.1	Tipos de requisição do protocolo OAI	12
2.2	Campos <i>Dublin Core</i>	14
5.1	Experiência e conhecimento dos usuários	45
5.2	Tempo gasto pelos usuários no teste de instalação	46
5.3	Erros cometidos pelos usuários no teste de instalação	47
5.4	Respostas ao questionário da tarefa de instalação	49
5.5	Tempo médio de carga dos dados (em segundos)	51
5.6	Aumento do tempo de carga da coleção LDB em relação à coleção BDBComp	52
5.7	Tempo de resposta dos componentes para a coleção LDB	54
5.8	Ganho relativo sobre o ODL	55
5.9	Comparação com a BDBComp	56

Capítulo 1

Introdução

A Internet, com seu vasto repositório de documentos digitais em constante crescimento, tornou-se um importante ambiente para disseminação de publicações e serviços de informação. Porém, a disponibilidade desta vasta coleção de documentos está longe de automaticamente implicar acessibilidade natural e sem esforço por parte de seus usuários. Em outras palavras, a informação existe, mas não é facilmente localizada e acessada.

A grande maioria dos usuários utiliza máquinas de busca para (tentar) encontrar a informação que necessitam. No entanto, as máquinas de busca podem não ser a melhor opção para satisfazer a necessidade de informação desses usuários, principalmente quando estes estão em busca de uma informação específica (por exemplo, a produção científica de um pesquisador brasileiro da área de ciência da computação). Isto acontece devido aos métodos utilizados para recuperação de informação pelas máquinas de busca, tipicamente limitados a casamentos exatos ou parciais entre um conjunto de palavras-chave, denominado *consulta*, e o texto dos documentos. Além disso, tem-se observado que as máquinas de busca vêm apresentando com o passar do tempo uma cobertura em termos percentuais cada vez menor do total de páginas da *Web*, apesar de o número absoluto de páginas indexadas estar em constante crescimento. Isto se deve ao fato de o número de páginas da *Web* criadas ser muito maior que o número de páginas que as máquinas de busca conseguem indexar. Com isto, a qualidade da resposta das máquinas de busca tende a cair (Bar-Ilan, 1999), levando-as a não atender satisfatoriamente a necessidade de informação dos usuários, que então necessitam de outras formas de acesso à informação.

Neste contexto, surge a necessidade de se criar mecanismos para tratar o volume, a dinâmica e a complexidade deste universo de informações. As bibliotecas digitais têm sido consideradas por vários pesquisadores como uma alternativa para a organização deste espaço de informação na Internet (Bax, 1997). A definição de o que é uma biblioteca digital não é consenso entre os pesquisadores, existindo muitas definições utilizadas na prática, cada uma tentando modelar diferentes aspectos dos sistemas existentes. Algumas definições enumeram os serviços existentes em uma biblioteca digital, tais como busca e navegação, listando-os como necessários ou suficientes para distinguir uma biblioteca digital de outros sistemas (Borgman, 1999; Digital Library Federation, 2006)). Também há definições mais rigorosas, que procuram conceituar as bibliotecas digitais por meio de arcabouços formais, que podem incluir a utilização de modelos matemáticos, como o trabalho de Locuratolo e Meghini (2003) ou o arcabouço 5S (Gonçalves, 2004), no qual uma biblioteca digital típica é informalmente definida como um conjunto de componentes matemáticos (por exemplo, coleções, serviços) e cada componente é definido como uma composição funcional ou uma combinação baseada em conjuntos de construções formais do arcabouço.

De acordo com Fox e Marchionini (1998), uma biblioteca digital é uma das formas mais avançadas e complexas de sistemas de informação, pois envolve, frequentemente, suporte de forma colaborativa, preservação de documentos digitais, gerenciamento de um banco de dados distribuído, filtragem e recuperação de informação, gerenciamento de direitos autorais, serviços de informação multimídia, serviços de referência, busca de recursos e disseminação seletiva de informação. Por outro lado, uma biblioteca digital estende e amplia os recursos e serviços existentes na Internet e dá oportunidades para o intercâmbio em nível mundial.

Baldacci (1993) conceitua uma biblioteca digital como uma enorme rede que conecta ou interliga o mundo, proporcionando a instalação de uma biblioteca global contendo todo tipo de informação. Já Bax (1997) acredita que são entidades capazes de vencer as limitações naturais e espaço-temporais impostas a outros objetos físicos (livros, estantes, salas, prédios), permitindo novas práticas de trabalhos e oportunidades.

Dada esta falta de consenso, não causa surpresa o fato de os padrões e tecnologias da área não convergirem facilmente. A maioria das bibliotecas digitais resultam de projetos de desenvolvimento de software com ciclos intensivos de desenho, implementação e testes, existindo várias razões que fazem com que este esforço de desenvolvimento seja repetido a cada novo projeto (Suleman, 2002):

- Muitas bibliotecas digitais são construídas de forma isolada, para atender às necessidades de uma comunidade particular, na maioria das vezes envolvendo pessoal com experiência anterior. Com isso, a lógica de programação por trás do sistemas pode variar muito.
- A maioria das bibliotecas digitais surge como uma solução rápida às necessidades de uma comunidade, e por causa disso não há muito planejamento sobre futuros desenvolvimentos das mesmas.
- Bibliotecas digitais podem ser muito complexas, e por isso novos projetos muitas vezes começam do zero, pois esta solução é mais econômica do que adaptar sistemas existentes para cenários diferentes.
- A extensibilidade e manutenção se tornam problemas à medida que uma biblioteca digital cresce e se torna mais complexa, pois a agregação de novos serviços pode trazer grande impacto ao sistema, principalmente quando o seu planejamento é deficitário.

Uma das formas de se atacar esses problemas é através da criação de conjuntos de componentes (elemento de software que encapsula uma série de funções) para construção de bibliotecas digitais, onde cada componente é responsável por uma pequena parte da funcionalidade da biblioteca digital e se integra com os demais de forma a se construir um sistema completo (Buchanan et al., 2005; OpenDLib, 2006; Suleman, 2002). Tais conjuntos oferecem um arcabouço genérico, extensível e reusável para construção de bibliotecas digitais, o que permite, por exemplo, diminuir o esforço necessário para o desenvolvimento desses sistemas.

Existem vários trabalhos que tratam de arcabouços baseados em componentes para criação de bibliotecas digitais, conforme mostrado na Seção 2.5. Porém, a maioria deles apresenta algum problema de interoperabilidade, desempenho ou integração destes componentes. Tais problemas motivaram o desenvolvimento deste trabalho, no qual criamos o arcaboço WS-ODL, um novo arcabouço baseado em componentes para a construção de bibliotecas digitais. Os componentes criados funcionam sobre a arquitetura Fedora (Lagoze et al., 2005), que provê ao arcabouço um repositório para os objetos digitais, que podem ser simples (por exemplo, metadados) ou complexos (por exemplo, objetos multimídia), além de fornecer serviços básicos de infra-estrutura. Toda comunicação entre os componentes e entre estes e o Fedora é realizada através de serviços web, utilizando SOAP – *Simple Object Access Protocol* (World Wide Web Consortium, 2006b), um protocolo de comunicação baseado em XML que traz vantagens como maior

interoperabilidade e validação de parâmetros de entrada, via XML Schema¹. A escolha do Fedora como provedor da infra-estrutura do arcabouço se deve ao fato de que este é uma arquitetura extensível e de código aberto, além de possuir um modelo de objetos digitais bastante robusto e flexível. No entanto, o arcabouço não se restringe ao uso do Fedora, sendo possível a substituição deste por outras arquiteturas, com as devidas adaptações. Por exemplo, uma outra opção possível seria construir uma infra-estrutura própria para o arcabouço baseada na tecnologia de bancos de dados relacionais. Porém, esta opção não teria uma boa relação custo/benefício, pois ou demandaria um grande esforço de desenvolvimento ou não seria tão abrangente quanto o Fedora, uma vez que este possui vários serviços de infra-estrutura (como um provedor de dados OAI, por exemplo) já prontos para uso e apresenta a possibilidade de agregação de novos serviços. Não obstante, a tecnologia de bancos de dados relacionais ainda é utilizada no contexto do nosso arcabouço, para dar suporte à funcionalidade dos componentes, incluindo aqueles que provêem serviços mais avançados, como é o caso do serviço de busca, que possui a capacidade de realizar busca textual por palavras-chave combinada com a utilização de dados estruturados, e do serviço de navegação multidimensional.

Através da avaliação experimental realizada, mostramos que o arcabouço é viável e que os componentes desenvolvidos apresentam um desempenho superior quando comparados aos componentes do arcabouço ODL – *Open Digital Libraries* (Suleman, 2002), que serviu como ponto de partida para este trabalho. Também apresentamos algumas vantagens do arcabouço em sua versão atual, que devem ser levadas em consideração quando da sua escolha como uma solução para a construção de bibliotecas digitais.

Em suma, as principais contribuições deste trabalho são:

- Desenvolvimento de um novo arcabouço para construção de bibliotecas digitais, seguindo a abordagem de componentes, permitindo extensibilidade, manutenção e reuso.
- Criação de componentes com funções complexas, como busca combinando dados estruturados e dados textuais e navegação multidimensional.
- Estudo da viabilidade do arcabouço, através de experimentação com usuários reais e avaliação do arcabouço em termos de tempo de instalação e configuração, de carga de dados e de desempenho dos componentes.

¹<http://www.w3.org/XML/Schema>

- Estudo da arquitetura Fedora, mostrando suas vantagens e desvantagens como repositório de objetos e provedor de serviços para bibliotecas digitais.

Esta dissertação está organizada da seguinte forma. No Capítulo 2, fornecemos conceitos e descrevemos trabalhos relacionados ao tema da dissertação. No Capítulo 3, descrevemos o arcabouço WS-ODL desenvolvido para construção de bibliotecas digitais. Em seguida, exemplos de utilização dos componentes são ilustrados no Capítulo 4. O Capítulo 5 trata da avaliação experimental do arcabouço. Por fim, no Capítulo 6, apresentamos as conclusões e trabalhos futuros.

Capítulo 2

Conceitos e Trabalhos Relacionados

Neste capítulo são fornecidas as bases para o desenvolvimento deste trabalho. A Seção 2.1 trata da abordagem baseada em componentes para desenvolvimento de software, um conceito importante para o nosso arcabouço, uma vez que uma de suas premissas é a utilização de componentes já desenvolvidos para a construção de uma biblioteca digital. Na Seção 2.2 descrevemos a tecnologia de serviços web (*web services*), utilizada na comunicação dos componentes do arcabouço com o Fedora, e também pelos serviços providos por esta arquitetura. A Seção 2.3 apresenta o conceito de arquivos abertos e descreve o protocolo OAI (OAI-PMH) e o formato de metadados *Dublin Core*, utilizado para armazenar os objetos digitais no arcabouço. Na Seção 2.4 é descrita a arquitetura Fedora, com suas principais características. Na Seção 2.5 são apresentados os principais trabalhos relacionados ao tema desta dissertação. Por fim, a Seção 2.6 trata do uso da tecnologia de bancos de dados relacionais para recuperação de informação.

2.1 Abordagem Baseada em Componentes

Componentes de software é a denominação dada aos elementos que oferecem um ou mais serviços pré-determinados e que são capazes de se comunicar com outros componentes. Eles possuem várias características, como por exemplo, ser encapsulados e de uso geral.

Segundo o Software Engineering Institute (2006), a abordagem baseada em componentes para desenvolvimento de software tem seu foco na construção de grandes sistemas de software através da integração de componentes previamente existentes. Esta abor-

dagem pode ser potencialmente utilizada para reduzir os custos e o tempo de desenvolvimento de um software, através do aumento da flexibilidade e da manutenibilidade.

Na base desta abordagem está a suposição de que certas partes de grandes sistemas de software aparecem com bastante frequência em vários pontos destes sistemas, de modo que estas partes comuns possam ser implementadas apenas uma vez e que sistemas possam ser construídos através do reuso destas partes, em vez de serem implementados novamente. A noção da construção de um sistema através de codificação é substituída pela construção através da montagem e integração de componentes de software. No desenvolvimento tradicional, a integração do sistema geralmente se encontra no fim do processo de implementação, o que não ocorre na abordagem baseada em componentes, onde a integração é uma peça central. Sendo assim, o grau de integração é uma consideração chave na decisão de se adquirir, reusar ou construir um componente de software.

A abordagem é caracterizada por quatro atividades principais, descritas a seguir:

1. *Qualificação do componente.* Trata do processo de se determinar o quão adaptado para uso em um novo sistema um componente previamente desenvolvido está. Esta atividade se divide em duas fases: a de descoberta, na qual as propriedades (funcionalidade, interface, confiabilidade, usabilidade, etc.) do componente são identificadas, e a de avaliação, que consiste em estudar um grupo de componentes semelhantes, de modo a identificar qual o melhor adaptado para fazer parte do novo sistema.
2. *Adaptação do componente.* Devido ao fato de serem escritos para atender a diferentes requisitos, e serem baseados em diferentes suposições com relação ao seu contexto, os componentes geralmente precisam ser adaptados para uso em um novo sistema. A adaptação deve procurar minimizar os conflitos entre os componentes. Existem três abordagens para a adaptação: a de caixa branca, onde o código do componente é modificado; a de caixa cinza, onde o componente fornece uma API (*application programming interface*); e a de caixa preta, onde apenas o binário executável do componente está disponível. Cada abordagem possui vantagens e desvantagens mas, em geral, a abordagem de caixa branca pode resultar em problemas de manutenção e evolução, pois o código fonte precisa ser modificado.
3. *Integração do componente.* O componente deve ser integrado no sistema através de uma infra-estrutura bem definida, que provê a ligação entre os componentes

para formar um sistema. Várias arquiteturas, todas com pontos positivos e negativos, são possíveis, tais como: controle centralizado dos dados compartilhados entre os componentes; barramento de mensagens e interfaces independentes de plataforma ou linguagem, sendo que existe atualmente uma tendência à utilização desta última (por exemplo, SOAP).

4. *Evolução do sistema.* À primeira vista, sistemas construídos segundo a abordagem baseada em componentes são simples de se manter e atualizar, uma vez que apenas componentes precisam ser substituídos no caso de um problema, ou adicionados no caso de uma nova função. No entanto, esta é uma visão simplista e otimista, pois a troca de um componente é um trabalho que consome tempo e esforço, uma vez que o novo componente nunca será igual ao antigo, devendo, pois, ser estudado, adaptado e testado.

Várias considerações, de curto e longo prazos, devem ser feitas com relação à utilização da abordagem baseada em componentes. As de curto prazo envolvem tópicos como processo de desenvolvimento, planejamento, requisitos do sistema e dos componentes, arquitetura do sistema, utilização de padrões, etc. As de longo prazo se relacionam com a evolução do sistema e com a dependência em relação a componentes de terceiros, integrados ao sistema. Todos esses itens devem ser bem estudados e avaliados para que a abordagem possa funcionar realmente como uma metodologia que proporcione a redução de riscos, custos, tempo e esforço de desenvolvimento.

2.2 Serviços Web

Um serviço web (World Wide Web Consortium, 2006d) é um sistema de software desenhado para suportar uma interação interoperável entre aplicações em rede. Em outras palavras, ele provê uma forma padronizada para se alcançar interoperabilidade entre diferentes aplicações, executando em uma variedade de plataformas e/ou arcabouços. Suas principais características são a interoperabilidade e a extensibilidade, bem como sua descrição em XML, capaz de ser entendida tanto por usuários como por agentes de software. Além disso, serviços web podem ser combinados (de uma maneira fracamente acoplada) de modo a realizar tarefas mais complexas, uma vez que programas que provêem serviços de natureza mais simples podem interagir entre si para fornecer serviços mais sofisticados e de maior valor agregado. O conceito de serviços web tem se mostrado como uma tecnologia promissora para a efetiva automação do processo

de integração de aplicações distribuídas e autônomas na Web, através da qual se pode encapsular processos de negócios pré-existent, publicá-los como serviços, descobrir dinamicamente serviços publicados, e trocar informações que ultrapassem as fronteiras de uma corporação.

Os serviços web se comunicam com outros sistemas através de uma interface descrita em WSDL – *Web Service Description Language* (World Wide Web Consortium, 2006c) e do protocolo SOAP – *Simple Object Access Protocol* (World Wide Web Consortium, 2006b). A interface em WSDL define características do serviço web, como seu modo de acesso, as operações disponíveis, os valores de entrada e de retorno. SOAP é um protocolo para troca de informação estruturada em um ambiente descentralizado e distribuído. Através do uso da tecnologia XML, o protocolo define um arcabouço extensível para transferência de mensagens, sob uma variedade de protocolos de nível mais baixo, como o HTTP. O arcabouço do protocolo é desenhado para ser independente de modelos de programação e semânticas de implementação.

Um terceiro padrão relacionado à tecnologia de serviços web é o UDDI – *Universal Description, Discovery and Integration* (UDDI, 2006), um protocolo que cria uma plataforma padronizada e interoperável que permite a empresas e aplicações encontrar e utilizar serviços web na Internet de forma fácil, rápida e dinâmica. UDDI pode ser visto como um cadastro global de serviços web, descrevendo os objetivos, a forma de utilização e os autores desses serviços, além de outras informações.

2.3 Arquivos Abertos

De acordo com a proposta Open Archives Initiative (2006), o termo “arquivo” tem origem na comunidade de *e-prints*, comunidade que promove e mantém repositórios de documentos científicos acessíveis via Web como uma maneira de aumentar o acesso à pesquisa científica. No entanto, nos arquivos abertos o termo é utilizado de uma maneira mais ampla, significando um repositório de informação, sem as conotações de preservação a longo prazo e políticas geralmente associadas ao termo no âmbito daquela comunidade. Já o termo “aberto” diz respeito à perspectiva arquitetural, definindo e promovendo interfaces de máquina que facilitam a disponibilidade do conteúdo procedente de vários provedores. Não significa, pois, acesso livre ou ilimitado à informação de um repositório.

Dois conceitos importantes estão ligados aos arquivos abertos: o protocolo OAI e o padrão de metadados *Dublin Core*.

2.3.1 O Protocolo OAI

O protocolo OAI (*Open Archives Initiative Protocol for Metadata Harvesting – OAI-PMH*) (Open Archives Initiative, 2006) é um protocolo para transferência de metadados entre aplicações (por exemplo, bibliotecas digitais). Ele fornece um arcabouço independente de aplicação e baseado em colheita (*harvesting*) de metadados para permitir interoperabilidade. Esta interoperabilidade não é imediata, não se destinando a buscas *on-line*, pois um repositório apenas disponibiliza seus dados de acordo com o protocolo, e uma aplicação externa deve fazer a colheita destes dados a intervalos regulares, para depois possibilitar buscas a partir da informação coletada.

A premissa do protocolo é promover a interoperabilidade através do mínimo de esforço por parte de seus usuários, sendo, portanto, de fácil implementação e baixo custo, além de não exigir equipes altamente especializadas. Em contrapartida, o protocolo não é tão completo como outros padrões para interoperabilidade, como o Z39.50¹, por exemplo.

O protocolo OAI possui duas classes de participantes:

- Provedores de dados: sistemas que suportam o protocolo como meio de disponibilizar seus metadados.
- Provedores de serviços: sistemas que utilizam os metadados obtidos através do protocolo como base para construir serviços de valor agregado.

Um provedor de dados, além de disponibilizar metadados de documentos digitais, pode também dispor de meios para o acesso a estes documentos, oferecendo serviços de recuperação e mecanismos de localização e acesso aos mesmos. Já um provedor de serviços é responsável por fornecer uma interface unificada para busca de metadados provenientes de diferentes provedores de dados, obtidos através de um *harvester*, um programa desenhado para obter informações de um provedor de dados via protocolo OAI.

¹<http://www.niso.org/z39.50/z3950.html>

A comunicação entre o *harvester* do provedor de serviços e o provedor de dados ocorre via protocolo HTTP (métodos GET ou POST), e é unidirecional do primeiro para o segundo. O *harvester* faz requisições ao provedor de dados, solicitando os dados que ele deseja obter. O provedor de dados, por sua vez, responde enviando os (meta)dados solicitados. Esses metadados podem ser colhidos de vários provedores de dados diferentes e armazenados em um repositório único, a partir do qual um usuário ou outro provedor de serviços pode realizar consultas. As requisições correspondem aos diferentes tipos de solicitação do protocolo OAI (*verbs*) e as respostas são documentos XML bem formados. A Figura 2.1 ilustra este processo de comunicação.

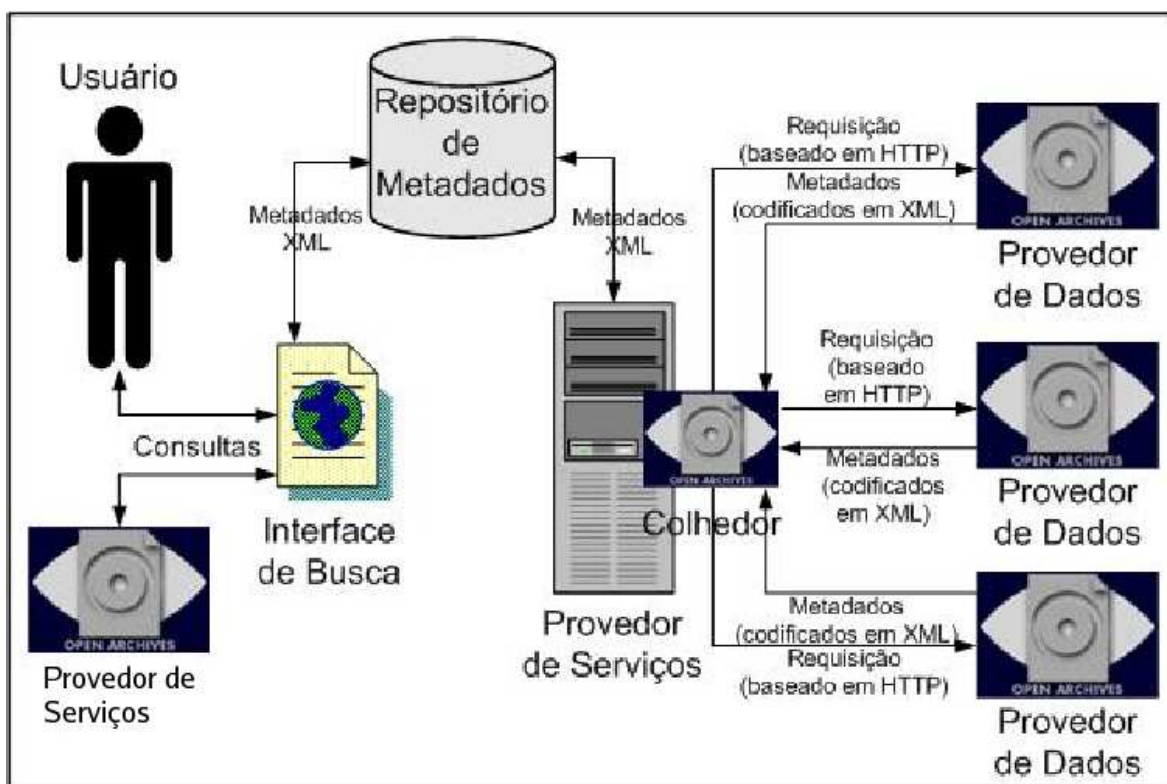


Figura 2.1: Processo de colheita baseado no protocolo OAI

O protocolo OAI estabelece o formato de metadados *Dublin Core* (*Dublin Core Metadata Element Set*) (Dublin Core, 2006) como conjunto mínimo de metadados a ser suportado pelos provedores de dados em resposta a uma requisição de um provedor de serviços. Apesar disto, o protocolo suporta múltiplos conjuntos de metadados, permitindo que as comunidades exponham sua informação em formatos que são específicos a suas aplicações e domínios (por exemplo, ETD-MS² ou MARC³). A estrutura téc-

²<http://www.ndltd.org/standards/metadata/current.html>

³<http://www.loc.gov/marc/>

nica do protocolo não coloca nenhuma limitação à natureza destes outros conjuntos de metadados, exceto a de que a resposta retornada seja estruturada em formato XML e possua um esquema correspondente para sua validação (Lagoze e de Sompel, 2001).

O protocolo define seis tipos de requisição que um *harvester* pode enviar a um provedor de dados para coletar informações contidas neste. Os tipos de requisição e suas funções são apresentados na Tabela 2.1.

Requisição	Função
<i>Identify</i>	Obtém dados administrativos sobre o provedor de dados e a política de publicação de documentos.
<i>ListSets</i>	Lista os agrupamentos sob os quais os documentos são organizados no provedor de dados, ou seja, lista os conjuntos dos quais os documentos fazem parte.
<i>ListMetadataFormats</i>	Lista os formatos de metadados por meio dos quais os metadados dos documentos armazenados no provedor de dados podem ser apresentados.
<i>ListIdentifiers</i>	Lista os identificadores de registros armazenados no provedor de dados, podendo opcionalmente limitar esses registros em um intervalo de datas ou pertencentes a um conjunto (<i>set</i>).
<i>ListRecords</i>	Lista os metadados dos registros armazenados no provedor de dados segundo um formato de metadados, especificando todos que pertencem a um conjunto (<i>set</i>) ou todos em um intervalo de datas.
<i>GetRecord</i>	Obtém os metadados de um registro armazenado segundo um formato de metadados, dado um identificador do registro.

Tabela 2.1: Tipos de requisição do protocolo OAI

2.3.2 *Dublin Core*

O padrão de metadados *Dublin Core* (Dublin Core, 2006) foi criado originalmente com o nome de *Dublin Core Metadata Set* em março de 1995, em um *workshop* sobre metadados realizado em Dublin, Ohio, EUA. Ele reúne esforços para alcançar uma solução comum de metadados para a comunidade de bibliotecas digitais e de repositórios de *e-prints*. O objetivo principal desse formato foi identificar e definir um conjunto mínimo de elementos capazes de descrever em larga escala recursos disponíveis na Internet.

O padrão possui dois níveis: Simples e Qualificado. O padrão *Dublin Core* Simples possui quinze elementos, um a menos que o padrão *Dublin Core* Qualificado, que possui

um elemento adicional: *Audience*. Além disso, o padrão *Dublin Core* Qualificado inclui um grupo de refinamentos dos elementos, chamados de qualificadores, que refinam a semântica dos elementos de maneira que eles possam ser úteis na descoberta de recursos na Internet. A semântica do formato *Dublin Core* foi estabelecida por um grupo internacional de profissionais de várias áreas, tais como, ciência da informação, informática, arqueologia e outros campos do conhecimento.

O padrão *Dublin Core* pode ser percebido como uma “pequena linguagem de uma classe particular de sentenças sobre recursos disponíveis na Internet”. Nesta linguagem, há duas classes de termos: os elementos (substantivos) e os qualificadores (adjetivos), que podem ser organizados dentro de um formato de sentença simples.

O conjunto básico de elementos do padrão *Dublin Core*, suas descrições e seus respectivos qualificadores são mostrados na Tabela 2.2. Cada elemento é opcional e pode ser repetido. A maioria dos elementos possui um conjunto limitado de qualificadores. É incentivado o uso de esquemas codificadores e de vocabulários controlados para determinados elementos (*format* e *language*, por exemplo).

O padrão *Dublin Core* é adotado mundialmente e tem sido utilizado em vários projetos que buscam um entendimento entre diferentes comunidades de usuários. Ao todo, são mais de 50 países utilizando este padrão, em setores como governos, bibliotecas, museus, empresas comerciais, comunidades educacionais e de pesquisa, etc (Dublin Core, 2006).

2.4 A Arquitetura Fedora

O Fedora (*Flexible Extensible Digital Object Repository Architecture*) é um arcabouço, desenvolvido em Java, para armazenamento, gerenciamento e disseminação de objetos digitais complexos, bem como os relacionamentos entre estes objetos (Lagoze et al., 2005). Tais objetos podem possuir conteúdo local ou distribuído, além de serviços associados a si. Isto permite que um objeto tenha várias representações (disseminações), que podem ser inclusive produzidas dinamicamente. Os relacionamentos entre os objetos são expressos em RDF (*Resource Description Framework*) e armazenados para consulta através de um serviço da arquitetura. A arquitetura, de código aberto, é implementada como um serviço web e provê a base para a construção de outras aplicações, como uma biblioteca digital, por exemplo.

O modelo de objetos do Fedora suporta vários tipos complexos de objeto, como do-

Elemento	Descrição	Qualificadores
<i>Title</i>	Título que um autor ou editor atribui a um trabalho.	<i>Alternative</i>
<i>Creator</i>	Autor ou autores responsáveis pelo conteúdo do trabalho.	
<i>Subject</i>	Assunto de que trata o trabalho definido por meio de palavras-chave ou tópicos.	
<i>Description</i>	Uma breve descrição do trabalho, tal como um resumo (<i>abstract</i>).	<i>Table of Contents</i> <i>Abstract</i>
<i>Publisher</i>	Um agente ou agência responsável pela disponibilização do trabalho em sua forma atual; normalmente uma editora ou instituição educacional.	
<i>Contributor</i>	Pessoas, além do(s) autor(es), que contribuíram substancialmente para o trabalho, por exemplo, editores e orientadores.	
<i>Date</i>	Data da disponibilização do trabalho em sua forma descrita (é usado normalmente o ano).	<i>Created</i> <i>Valid</i> <i>Available</i> <i>Issued</i> <i>Modified</i> <i>Date Copyrighted</i> <i>Date Submitted</i>
<i>Type</i>	Tipo do objeto, como por exemplo: coleção ou texto.	
<i>Format</i>	Formato em que o trabalho digital se encontra, por exemplo: PDF, HTML ou JPG.	<i>Extent</i> <i>Medium</i>
<i>Identifier</i>	Uma cadeia de caracteres que identifica exclusivamente o objeto, por exemplo: ISBN e URL.	<i>Bibliographic Citation</i>
<i>Source</i>	Outras fontes (caso se aplique) das quais o trabalho se deriva.	
<i>Language</i>	Idioma usado no trabalho.	
<i>Relation</i>	Relacionamento, se existir algum, do trabalho com outros trabalhos (normalmente descreve um trabalho como parte de um conjunto maior).	<i>Is Version Of</i> <i>Has Version</i> <i>Is Replaced By</i> <i>Replaces</i> <i>Is Required By</i> <i>Requires</i> <i>Is Part Of</i> <i>Has Part</i> <i>Is Referenced By</i> <i>References</i> <i>Is Format Of</i> <i>Has Format</i> <i>Conforms To</i>
<i>Coverage</i>	A área geográfica que o trabalho engloba, se aplicável.	<i>Spatial</i> <i>Temporal</i>
<i>Rights</i>	Direitos ou outras propriedades intelectuais especificando as condições através das quais o trabalho pode ou não ser usado.	<i>Access Rights</i>
<i>Audience</i>	Pessoa ou entidade a quem o recurso está destinado ou é útil.	<i>Mediator</i> <i>Education Level</i>

Tabela 2.2: Campos *Dublin Core*

cumentos, imagens, livros eletrônicos, objetos multimídia, programas de computador, etc. Além disso, estes tipos podem ser combinados para formar novos objetos e é possível associar serviços a objetos para produzir conteúdo dinâmico. O modelo também suporta relacionamentos entre os objetos, de modo a permitir que objetos dentro do repositório possuam alguma relação semântica entre si (por exemplo, capítulos de um livro ou trabalhos pertencentes aos anais de uma conferência).

O modelo pode ser entendido através de duas perspectivas, a *representacional* e a *funcional*, descritas a seguir.

2.4.1 Perspectiva Representacional

A perspectiva representacional define uma abstração simplificada para os objetos, onde estes são modelados como recursos unicamente identificados (através de URIs⁴), projetando uma ou mais visões (representações). Nesta perspectiva, a estrutura interna do objeto digital é transparente para o usuário final, mas os relacionamentos entre os objetos são acessíveis.

A perspectiva representacional declara que cada objeto digital pode disseminar uma ou mais representações de si mesmo (por exemplo, um documento nos formatos DOC, HTML e PDF), e que cada objeto pode se relacionar com um ou mais objetos. Esta perspectiva esconde a complexidade e expõe apenas os pontos de acesso para o conteúdo do objeto no repositório do Fedora. A Figura 2.2 ilustra a perspectiva representacional de um objeto digital. O objeto digital identificado por *info:fedora/wsodl:10* possui quatro representações:

- O registro Dublin Core, identificado por *info:fedora/wsodl:10/DC*
- O documento em formato DOC, identificado por *info:fedora/wsodl:10/DOC*
- O documento em formato HTML, identificado por *info:fedora/wsodl:10/HTML*
- O documento em formato PDF, identificado por *info:fedora/wsodl:10/PDF*

Vale a pena observar que as fontes de dados por trás dessas representações não foram definidas ainda e, de fato, tais detalhes da arquitetura são transparentes a uma aplicação cliente, que apenas precisa se preocupar em acessar as representações.

⁴<http://www.w3.org/Addressing/>

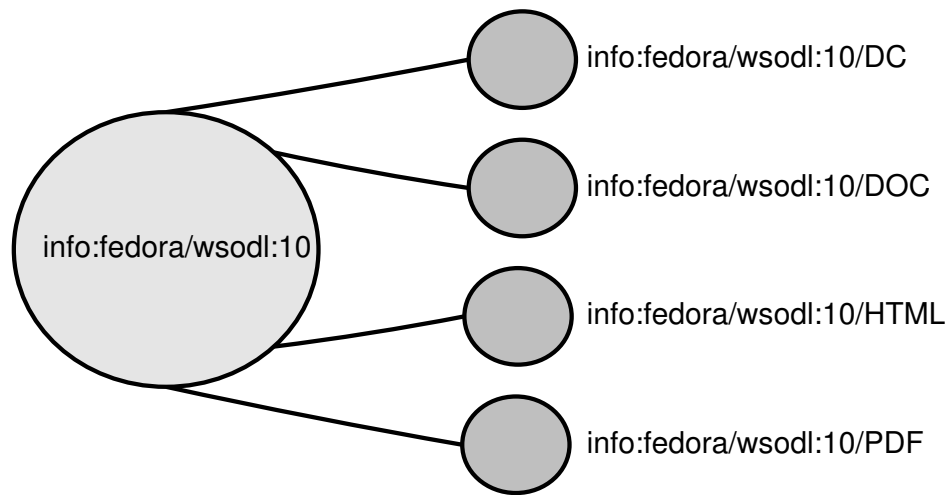


Figura 2.2: Visão representacional de um objeto do Fedora

2.4.2 Perspectiva Funcional

A perspectiva funcional expõe os componentes de um objeto e provê a base para se entender como o modelo de objetos do Fedora se relaciona com os serviços de gerenciamento disponibilizados pela arquitetura.

Em sua forma mais simples, um objeto digital no Fedora é uma agregação de itens de conteúdo, onde cada item é mapeado para uma representação. Tais itens de conteúdo são representados por componentes denominados *datastreams*, que podem tanto encapsular o conteúdo internamente como um conjunto de *bytes* como referenciar um conteúdo externo ao repositório. Existe uma correspondência entre um *datastream* e uma representação do objeto. No caso mais simples, a representação é uma simples transcrição do conteúdo do *datastream*.

Além destas representações transcritas de *datastreams*, o modelo de objetos também define representações virtuais de um objeto. Uma representação virtual ou disseminação (*dissemination*) é uma visão do objeto produzida a partir de uma operação de um serviço (uma chamada a um método, por exemplo) que recebe como parâmetro um ou mais *datastreams* do objeto em questão. Sendo assim, ela é um meio para disponibilizar conteúdo dinâmico a partir de um objeto do Fedora.

Na Figura 2.2, os nodos identificados por *info:fedora/wsodl:10/HTML* e *info:fedora/wsodl:10/PDF* podem representar disseminações do objeto *info:fedora/wsodl:10*. Do ponto de vista de uma aplicação cliente, não existe diferença entre essas disseminações e as demais representações do objeto.

A noção de representação virtual é suportada no modelo por um componente denominado *disseminator*, que contém três informações: uma referência para a descrição de uma operação de um serviço em uma sintaxe abstrata, uma referência para a descrição WSDL de um serviço (define qual o serviço web que executa a operação) e os identificadores dos *datastreams* utilizados como parâmetros de entrada para a operação.

Ainda dentro da perspectiva funcional, o modelo suporta dois tipos de *datastream* relacionados ao gerenciamento da integridade dos objetos digitais. Políticas de acesso aos objetos são expressas através do *datastream policy*, descrito em XACML (XACML, 2006), uma linguagem baseada em XML para declarar as permissões dos usuários aos objetos. O *datastream* de auditoria (*audit trail*) mantém um registro de todas as modificações em um objeto durante sua existência, sendo de importância para fins de preservação do objeto digital. Associado ao conceito de auditoria, está o versionamento dos objetos dentro do repositório, suportado no nível de componente (*datastreams* e *disseminations*).

2.4.3 Relacionamentos entre Objetos

O modelo de objetos do Fedora pode ser visto como uma grafo dirigido, formado por arestas internas (que ligam nodos de objetos digitais aos nodos de representações destes objetos), e arestas externas (que ligam nodos de objetos digitais entre si). O serviço *Resource Index* do Fedora permite que este grafo seja armazenado e consultado.

Cada objeto digital possui um *datastream* reservado, denominado *relations*, que é utilizado para descrever os relacionamentos deste objeto com outros. Triplas representando o grafo de relacionamento, expressas em RDF (*Resource Description Framework*) (World Wide Web Consortium, 2006a), são derivadas deste *datastream* e do próprio modelo de objetos e inseridas no *triplestore* Kowari (Tucana Technologies, 2006), que fornece a capacidade de pesquisa no grafo por meio de consultas expressas em iTQL (Tucana Technologies, 2006), uma linguagem de consulta com sintaxe semelhante àquela do SQL. Na Figura 2.3 mostramos um exemplo do *datastream relations* para o objeto identificado por *info:fedora/wsodl:29464*, onde se declara que este objeto é membro da coleção representada pelo objeto *info:fedora/wsodl:29422*. Neste caso, por exemplo, o primeiro objeto poderia ser um capítulo do livro representado pelo segundo objeto.

```

<rdf:RDF xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:fedora=
  "info:fedora/fedora-system:def/relations-external#" xmlns:oai_dc=
  "http://www.openarchives.org/OAI/2.0/oai_dc/" xmlns:rdf=
  "http://www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns:rdfs=
  "http://www.w3.org/2000/01/rdf-schema#">
  <rdf:Description rdf:about="info:fedora/wsodl:29464">
    <fedora:isMemberOfCollection rdf:resource="info:fedora/wsodl:29422"/>
  </rdf:Description>
</rdf:RDF>

```

Figura 2.3: Datastream relations

2.5 Arcabouços para Construção de Bibliotecas Digitais

Vários projetos lidam com o desenvolvimento de arcabouços para a construção de bibliotecas digitais.

O Dienst (Lagoze e Davis, 1995) é uma arquitetura de biblioteca digital em rede com disseminação baseada em protocolo de objetos digitais em múltiplos formatos. O protocolo utilizado também é denominado Dienst e é uma especificação formal do modo como vários componentes interagem com uma biblioteca digital, utilizando HTTP e XML. Este sistema possui forte influência no desenvolvimento do protocolo OAI, na arquitetura Fedora e também no arcabouço ODL.

O arcabouço ODL – *Open Digital Libraries* (Suleman, 2002) foi um dos primeiros a defender uma abordagem baseada em componentes para o desenvolvimento de bibliotecas digitais. ODL propõe uma extensão ao protocolo OAI para suportar interações entre os componentes da biblioteca digital. O arcabouço possui algumas vantagens, decorrentes da utilização de uma filosofia baseada no protocolo OAI e da componentização dos serviços, dentre as quais se destacam:

1. Simplicidade: o protocolo OAI é simples e fácil de se implementar.
2. Abertura: o protocolo OAI é um padrão de domínio público.
3. Reuso: a construção de sistemas segundo a abordagem de componentes leva ao reuso tanto da solução final como dos resultados intermediários como, por exemplo, o processo de desenho.

No entanto, o arcabouço apresenta problemas com relação a desempenho, escalabilidade e interoperabilidade. Os problemas relacionados ao desempenho são devidos a decisões de implementação específicas, que levam a problemas de escalabilidade, principalmente no que diz respeito aos componentes que implementam funções de recuperação de informação. Já os problemas de interoperabilidade estão relacionados à utilização do protocolo XOAI (*Extended Open Archives Initiative Protocol for Metadata Harvesting – XOAI-PMH*), um protocolo desenvolvido no contexto do arcabouço ODL e que não é um padrão largamente utilizado por outras aplicações.

OpenDLib (OpenDLib, 2006) é um *toolkit* para criação de bibliotecas digitais, de acordo com os requisitos de uma dada comunidade de usuários, através da instanciação do software da maneira adequada e da posterior carga ou colheita (*harvesting*) do conteúdo a ser gerenciado. O *toolkit* é uma federação de serviços que implementam a funcionalidade de uma biblioteca digital, fazendo poucas suposições a respeito da natureza dos documentos sendo armazenados. Além disso, ele pode ser estendido com outros serviços. OpenDLib é bastante semelhante ao ODL, uma vez que ambos utilizam a abordagem baseada em componentes e dividem um ponto fraco em comum: o uso de protocolos proprietários.

Greenstone (Buchanan et al., 2005) é um sistema extensível para construção de bibliotecas digitais bastante popular, e que usa uma arquitetura modular. No entanto, não se pode considerar que ele faz uso da abordagem de componentes, uma vez que não é possível anexar ou desanexar partes funcionais do sistema com o objetivo de adaptá-lo a certos requisitos e/ou ambientes nem incluir novos serviços simplesmente através da inclusão de novos componentes que se comunicam com o restante do sistema. Além disso, a interoperabilidade com outros sistema não é facilmente alcançada no Greenstone, como demonstrado em Bainbridge e Witten (2005).

Congia et al. (2004) apresenta o protocolo SOAP-PMH, um protocolo com funcionalidade semelhante ao protocolo OAI, mas que utiliza SOAP em lugar de HTTP como camada inferior.

Em Apps (2004) é apresentada uma interface baseada em Serviços Web / SOAP para o zetoc, um serviço de entrega de documentos baseado na *British Library's Electronic Table of Contents* de artigos de periódicos e conferências. Esta interface utiliza metadados em XML padrão para buscas e respostas, de maneira semelhante ao protocolo OAI.

O projeto OpenArXiv (Fisher e Lee, 2006) tem por objetivo aprimorar o sistema ar-

Xiv⁵, uma das bibliotecas digitais de conteúdo científico mais populares, através do uso das tecnologias de bancos de dados relacionais e serviços web. Os serviços web fornecem uma interface acessível tanto para usuários como para agentes de software e o banco de dados é utilizado para prover uma aplicação de larga escala. No entanto, o projeto faz uso das facilidades de busca textual de um SGBD específico (Microsoft SQL Server), restringindo o seu escopo de aplicação.

2.6 Mecanismos para Recuperação de Informação em Bancos de Dados

Uma grande parcela das informações armazenadas digitalmente está na forma de dados textuais não estruturados, tanto em arquivos de texto normais quanto em documentos formatados. Enquanto um grande volume desses dados textuais é armazenado em sistemas de arquivos, muitas organizações estão atualmente armazenando dados deste tipo em bancos de dados relacionais. Isto torna as vantagens de um banco de dados amplamente disponíveis – incluindo acesso de alto desempenho, capacidade de consulta, interfaces de usuário mais simples, baseadas em aplicações, e acesso remoto seguro (Microsoft, 2006).

Vários SGBDs existentes no mercado (Microsoft SQL Server, MySQL, Oracle, por exemplo) fornecem a facilidade de busca em texto completo nos dados armazenados. No entanto, cada um deles implementa esta facilidade de uma maneira diferente, com suas vantagens e desvantagens. Isso torna as aplicações desenvolvidas dependentes de um SGBD específico, não sendo possível migrar facilmente de um sistema para outro conforme a necessidade, pois a linguagem SQL padrão não é utilizada por nenhuma das soluções.

Grossman et al. (1997) mostram que é possível utilizar o modelo relacional padrão para implementar um sistema de recuperação de informação e que tal sistema, em cenários específicos, apresenta bom desempenho e é escalável. Além disso, ele torna possível a integração de dados estruturados e dados textuais, o que provê um mecanismo de busca mais poderoso do que aqueles de máquinas de busca tradicionais. Um mecanismo como este torna-se uma característica bastante interessante e vantajosa em uma biblioteca digital, onde os usuários possuem necessidades de informação mais específicas que aquelas de um usuário comum da Web.

⁵<http://arxiv.org>

Neste trabalho, partimos da premissa que uma biblioteca digital é um dos cenários onde este mecanismo pode ser aplicado e, assim, o utilizamos como base para o serviço de busca fornecido por nosso arcabouço.

Capítulo 3

Descrição do Arcabouço

Neste capítulo, descrevemos o arcabouço WS-ODL desenvolvido para a construção de bibliotecas digitais a partir de um conjunto de componentes e da arquitetura Fedora. Na Seção 3.1, descrevemos o esquema geral de funcionamento dos componentes do arcabouço e nas seções subseqüentes descrevemos cada componente especificamente.

Os componentes que fazem parte do arcabouço são o de Exibição de Trabalhos Recentes, Navegação, Busca, Registro de Usuário e Anotação, além de componentes de administração do arcabouço. Estes componentes foram escolhidos para fazer parte do arcabouço devido a serem serviços comumente disponíveis em uma biblioteca digital, além de serem de fácil entendimento, utilização, configuração e implementação.

O arcabouço é composto de basicamente três partes:

- Um repositório de dados, baseado na arquitetura Fedora, que suporta serviços básicos de infra-estrutura, como um provedor de dados OAI, versionamento e auditoria de objetos digitais, e armazenamento e consulta do grafo de relacionamento entre os objetos digitais.
- Um conjunto de serviços web, desenvolvidos em Java, responsáveis por prover os serviços da biblioteca digital, tais como busca e navegação.
- Uma camada cliente (camada de apresentação), responsável pela interface com o usuário, pelo envio de dados aos serviços web e pelo tratamento das respostas retornadas por estes. Esta camada é constituída de um conjunto de *servlets*¹

¹<http://java.sun.com/products/servlet/>

Java, que executam transformações XSL (XSLT²) nos dados retornados pelos serviços web. Com a utilização destas tecnologias, o administrador da biblioteca digital possui a flexibilidade para criar o *layout* mais adequado aos usuários do sistema, através da utilização de folhas de estilo XML, que são configuradas para uso pelos componentes através de parâmetros de arquivos de configuração em XML.

Esta divisão do arcabouço possibilita a distribuição dos componentes da biblioteca digital, o que, por sua vez, leva a uma maior flexibilidade dos sistemas construídos com o arcabouço. O repositório de dados, os serviços web e a camada cliente podem estar hospedados em servidores distintos. Além disso, não há necessidade de os serviços web estarem localizados no mesmo servidor, podendo haver vários serviços distribuídos por vários servidores.

A utilização da tecnologia de serviços web e arquivos abertos facilita a interoperabilidade e padroniza o acesso aos dados do sistema, tanto por usuários quanto por agentes de software.

Na camada cliente, pode haver alguma dificuldade por parte dos usuários para a criação das folhas de estilo XML, que possuem uma sintaxe relativamente complexa. Para contornar esta dificuldade, todos os componentes provêm folhas de estilo básicas, que podem ser utilizadas pelos usuários como ponto de partida para a criação de *layouts* mais sofisticados. Além disso, podem ser utilizados editores gráficos (Progress Software Corporation, 2006) com interface *WYSIWYG* – *What you see is what you get*.

Uma característica importante do arcabouço é que ele utiliza tecnologias abertas e de código livre, o que possibilita a disponibilização do arcabouço sem ônus para os usuários interessados.

A Figura 3.1 fornece uma visão geral do arcabouço WS-ODL, onde os elementos ligados à arquitetura Fedora estão ilustrados em cinza claro e os componentes do arcabouço estão em amarelo. Esta figura ilustra apenas o tipo de comunicação que ocorre entre os elementos do arcabouço, uma vez que o fluxo de trabalho é mostrado na Figura 3.2.

²<http://www.w3.org/TR/xslt>

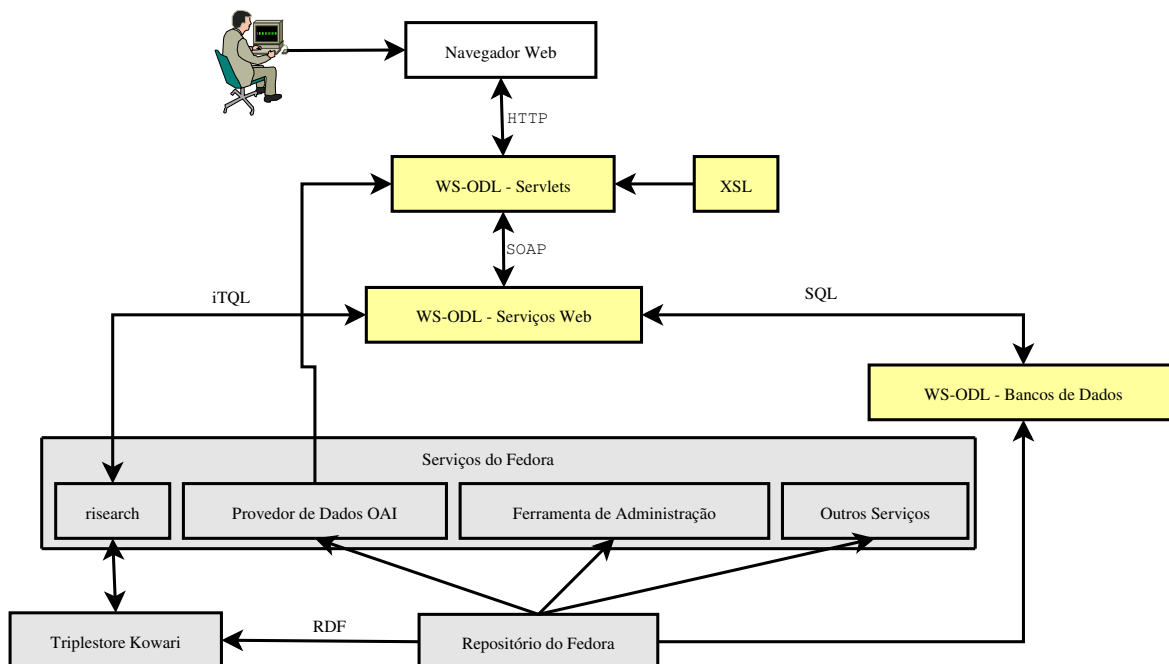


Figura 3.1: Visão geral do arcabouço WS-ODL

3.1 Esquema Geral de Funcionamento dos Componentes do Arcabouço

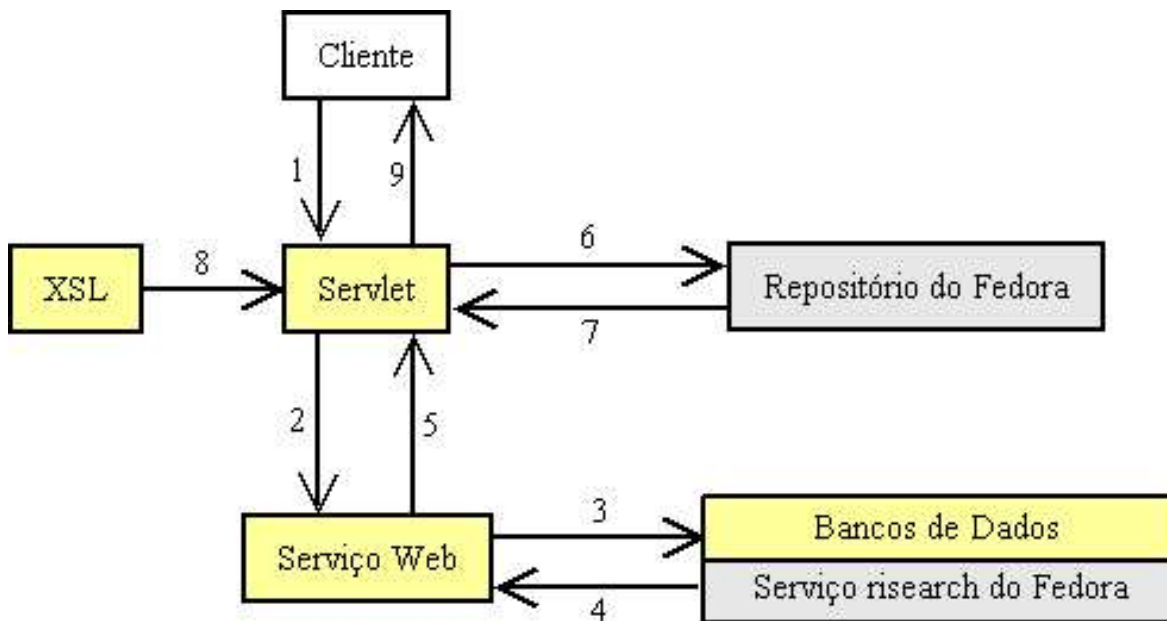


Figura 3.2: Esquema geral de funcionamento dos componentes

Um componente típico do arcabouço (por exemplo, Exibição de Trabalhos Recentes), cujo esquema geral de funcionamento é ilustrado na Figura 3.2, trabalha da seguinte

forma:

1. O cliente (usuário final), através de um navegador Web, faz uma requisição HTTP ao *servlet* do componente.
2. O *servlet* requisita a um serviço web os identificadores (*pids*) de interesse (por exemplo, *pids* dos trabalhos mais recentes).
3. O serviço web se comunica com um banco de dados ou com o serviço *research* – *Fedora Research Index Query Service* do Fedora.
4. O banco de dados ou o serviço *research* retorna os *pids* para o serviço web.
5. O serviço web encaminha os *pids* para o *servlet*.
6. O *servlet* requisita ao Fedora os dados armazenados no repositório, referentes aos *pids*.
7. Um serviço web associado ao repositório do Fedora retorna os metadados referentes aos *pids*, em formato *Dublin Core*.
8. O *servlet* executa uma transformação XSL nos metadados retornados.
9. Os dados transformados são exibidos no navegador Web do cliente.

A Figuras 3.3 e 3.4 mostram um exemplo de uma requisição e de uma resposta SOAP, respectivamente. A requisição é realizada no passo 2, e a resposta é obtida no passo 5. Basicamente, a requisição contém o nome do serviço sendo invocado (*getMostRecent*) e o(s) parâmetro(s) do mesmo (*amountToGet*). Já a resposta contém os elementos retornados pelo serviço web em atendimento à requisição (no caso do exemplo, os dados contidos nas *tags item*).

3.2 Componente de Exibição de Trabalhos Recentes

O componente de Exibição de Trabalhos Recentes tem por objetivo estimular a curiosidade do usuário, através da exibição dos trabalhos mais recentemente incluídos (ou modificados) no repositório da biblioteca digital, em ordem decrescente de data de inclusão (ou modificação). O usuário tem a opção de especificar, como um parâmetro do

```

<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/
2001/12/soap-envelope" soap:encodingStyle=
"http://www.w3.org/2001/12/soap-encoding">
  <soap:Body>
    <getMostRecent>
      <amountToGet>2</amountToGet>
    </getMostRecent>
  </soap:Body>
</soap:Envelope>

```

Figura 3.3: Exemplo de uma requisição SOAP

```

<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv=
"http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
  <soapenv:Body>
    <getMostRecentResponse soapenv:encodingStyle=
"http://schemas.xmlsoap.org/soap/encoding/">
      <getMostRecentReturn href="#id0"/>
    </getMostRecentResponse>
    <multiRef id="id0" soapenc:root="0" soapenv:
encodingStyle="http://schemas.xmlsoap.org/soap/
encoding/" xsi:type="ns1:Vector" xmlns:soapenc=
"http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://xml.apache.org/xml-soap">
      <item xsi:type="xsd:string">wsodl:4142</item>
      <item xsi:type="xsd:long">1137248479234</item>
      <item xsi:type="xsd:string">wsodl:4141</item>
      <item xsi:type="xsd:long">1137248478832</item>
    </multiRef>
  </soapenv:Body>
</soapenv:Envelope>

```

Figura 3.4: Exemplo de uma resposta SOAP

componente, o número de trabalhos que ele deseja visualizar. No caso de tal parâmetro não ser informado, um número padrão de dez trabalhos é exibido.

Os *pids* dos trabalhos recentes são obtidos a partir de uma tabela (*doFields*) existente no banco de dados relacional do Fedora e que armazena (meta-)informação sobre os objetos digitais. O serviço web do componente de Exibição de Trabalhos Recentes acessa esta tabela e recupera o conteúdo dos campos *pid* e *mDate*, ordenado pelo último, que contém a data da última modificação do objeto. Os dados obtidos são então enviados ao *servlet*, que os utiliza para recuperar os objetos digitais no repositório do Fedora.

Embora baseado na arquitetura Fedora, o componente pode ser modificado para trabalhar com outras infra-estruturas com pouco esforço, bastando que seja criada uma tabela similar à descrita anteriormente. Assim, o componente não depende desta arquitetura em particular.

A opção de se realizar um acesso ao banco de dados do Fedora e em seguida um acesso ao repositório para recuperar os objetos digitais propriamente ditos pode parecer um desperdício de recursos. No entanto, ela se justifica pelo fato de que o Fedora não possui capacidade de ordenação em sua API de busca, sempre listando os objetos digitais do mais antigo para o mais recente. Sendo assim, para exibir os trabalhos mais recentes, seria necessário buscar todos os dados e ordená-los localmente no cliente, o que é claramente muito mais ineficiente que a estratégia adotada.

3.3 Componentes de Navegação

O arcabouço WS-ODL incorpora dois componentes com diferentes características para o serviço de navegação. O primeiro deles é composto por um conjunto de serviços web e *servlets* desenhados para prover uma estrutura navegacional a partir dos metadados armazenados no repositório do Fedora. O segundo é um componente de navegação multidimensional baseado em um metamodelo e um banco de dados. Cada um desses componentes é descrito nas seções abaixo.

3.3.1 Componente Baseado em Serviços Web

A estrutura de navegação oferecida por este componente se integra com os demais componentes do arcabouço, como o de Exibição de Trabalhos Recentes e de Busca, fornecendo a possibilidade de utilização de vários *links* de navegação, o que permite ao usuário uma experiência mais rica na utilização de uma biblioteca digital construída a partir destes componentes.

Os serviços web associados a este componente são capazes de obter: os *pids* referentes a um dado nome de autor, os *pids* referentes aos trabalhos de uma coleção, o nome de uma coleção dado seu *pid*, e informações sobre a coleção dado um *pid* de trabalho pertencente à mesma. No contexto deste componente, uma coleção é um grupo relacionado de objetos digitais. Por exemplo, os trabalhos pertencentes aos anais de uma conferência formam uma coleção. Podemos então ter uma coleção para os anais do XXI Simpósio Brasileiro de Banco de Dados (SBBD 2006), outra para os anais do XXIV Simpósio Brasileiro de Redes de Computadores (SBRC 2006), etc. Este conceito é similar ao conceito de conjunto (*set*) do protocolo OAI, com a diferença de que aqui um objeto digital pertence a apenas uma coleção, enquanto no modelo de dados do OAI, um objeto pode pertencer a vários conjuntos³.

Os *servlets* fazem uso dos dados obtidos a partir dos serviços web para produzir páginas de trabalhos do autor (ordenados por data) e de trabalhos da coleção. A funcionalidade deste componente é similar àquela descrita na Seção 3.1. No entanto, ao invés de o componente acessar um banco de dados para obter os *pids* de interesse, ele utiliza o serviço *research* do Fedora, que permite a consulta dos relacionamentos, expressos em RDF, entre os objetos digitais armazenados no repositório. Desta forma, para se criar uma página de trabalhos do autor, um dos serviços web de navegação envia para o *research* a seguinte consulta iTQL, onde “*Algum Autor*” é um parâmetro recebido pelo serviço web:

```
select $object $date
from <#ri>
where $object <dc:creator> 'Algum Autor'
      and $object <dc:date> $date
order by $date desc
```

³<http://www.openarchives.org/OAI/openarchivesprotocol.html#Set>

Esta consulta busca no *triplestore* Kowari mantido pelo Fedora por todas as triplas RDF (na forma sujeito-predicado-objeto) que satisfazem o predicado “<dc:creator> é *Algum Autor*” e retorna os sujeitos que “casam” (associados à variável *\$object*). Uma vez que desejamos que o resultado final (na página de trabalhos do autor) esteja ordenado por data, a variável *\$date* também deve fazer parte da consulta, e deve estar relacionada com os sujeitos de interesse, o que é feito na segunda parte da cláusula *where*. Como um efeito colateral, a data dos objetos também será retornada pela consulta. O serviço web que enviou a consulta recebe os resultados e repassa apenas os *pids* ao *servlet*, que pode então recuperar os objetos digitais completos a partir do repositório do Fedora.

De maneira análoga, para se construir uma página de trabalhos da coleção, um serviço web envia a seguinte consulta, onde “*pid_alguma_colecao*” é um parâmetro de entrada do serviço:

```
select $object
from <#ri>
where $object <fedora-rels-ext:isMemberOfCollection>
    <info:fedora/pid_alguma_colecao>
order by $object
```

Com base no predicado <fedora-rels-ext:isMemberOfCollection>, a consulta associa a variável *\$object* a identificadores de objetos digitais que são membros da coleção identificada por <info:fedora/pid_alguma_colecao> e ordena o resultado por esta variável. O serviço web recebe estes identificadores como resposta e os envia a um *servlet*, que os utiliza para obter os objetos digitais no repositório do Fedora.

3.3.2 Componente de Navegação Multidimensional

O componente de Navegação Multidimensional (Raghavan et al., 2005) provê uma estrutura hierárquica de navegação baseado em um metamodelo expresso em XML e em um banco de dados relacional. Este componente permite que outras formas de navegação (por exemplo, por data ou por local de publicação de um trabalho) sejam facilmente disponibilizadas ao usuário, através da alteração da configuração do metamodelo e da atualização do banco de dados. Este componente pode funcionar como um substituto para os serviços web e *servlets* responsáveis por gerar as páginas de trabalhos de um autor e de trabalhos de uma coleção. Uma característica interessante

deste componente é que ele é independente do repositório utilizado pelo arcabouço, uma vez que ele é baseado em um banco de dados construído a partir de metadados do repositório, conforme descrito na Seção 3.6. Desta forma, se substituirmos o Fedora por uma outra arquitetura, nenhuma modificação no componente será necessária.

A Figura 3.5 apresenta um exemplo de fragmento do metamodelo, onde são configuradas duas dimensões de navegação. Cada dimensão possui um nome e uma expressão *XPath*⁴, que é utilizada para se recuperar os dados desejados dos objetos digitais e armazená-los em um banco de dados. Com este fragmento, é possível criar índices que suportam a navegação por data (*date*) e local de publicação de um trabalho (*coverage*). A partir destas dimensões de navegação, um componente de administração (detalhado na Seção 3.6) cria um banco de dados relacional, que é utilizado como índice para navegação pelos objetos da biblioteca digital. Para se criar uma nova dimensão ou se excluir alguma previamente existente, basta modificar o arquivo XML do metamodelo e executar o componente para criar (ou atualizar) o banco de dados.

```
<browse>
  <dimension name="coverage">
    <level>
      <levelname>Coverage</levelname>
      <path>//dc:coverage</path>
    </level>
  </dimension>

  <dimension name="date">
    <level>
      <levelname>Date</levelname>
      <path>//dc:date</path>
    </level>
  </dimension>
</browse>
```

Figura 3.5: Exemplo de fragmento do metamodelo do componente de navegação multidimensional

O componente de Navegação Multidimensional foi originalmente criado pelo grupo de bibliotecas digitais da Virginia Tech⁵. Para atender às necessidades do nosso arcabouço, o componente foi modificado, com o objetivo de permitir atributos multivalorados para os objetos digitais, uma característica necessária para dar mais flexibilidade às aplicações que fizerem uso do componente. No contexto original do componente, um trabalho

⁴<http://www.w3.org/TR/xpath>

⁵<http://www.dlib.vt.edu/>

pode possuir apenas um título, uma data, um autor, etc. Para alguns atributos, como o título, isto pode fazer sentido; para outros, como autor, pode não fazer. Por exemplo, em uma biblioteca digital contendo artigos publicados em anais de eventos, certamente haverá trabalhos que possuem mais de um autor, que não seriam corretamente tratados sem a utilização de atributos multivalorados.

Antes da modificação realizada, existia no banco de dados do componente uma tabela principal – contendo o *pid* do objeto digital, os *pids* das dimensões de navegação (chaves estrangeiras) e uma data/hora (*datestamp*) – e as tabelas das dimensões de navegação, contendo os identificadores e valores das dimensões. Na Figura 3.6 é mostrado um exemplo do esquema relacional do banco de dados do componente, configurado para trabalhar com as dimensões *Autor* e *Local*. Como pode ser observado, o esquema não permite atributos multivalorados, uma vez que os *pids* das dimensões de navegação estão presentes na tabela principal. Após a modificação, os *pids* das dimensões de navegação foram removidos da tabela principal, passando a integrar uma “tabela de relacionamento”, que contém o *pid* do objeto digital, o *pid* da dimensão de navegação e um campo de posição que indica a ordem do atributo do contexto do atributo multivalorado. Este campo de posição foi criado porque pode haver situações em que é importante preservar a ordem do atributo (por exemplo, a ordem dos autores de um trabalho). A Figura 3.7 ilustra um exemplo do novo esquema do banco de dados, também utilizando as dimensões *Autor* e *Local*. É importante ressaltar que as Figuras 3.6 e 3.7 representam apenas uma instanciiação do esquema do banco de dados, com base em algum metamodelo.

```
Autor(autor_id, autor)
Local(local_id, local)
Principal(pid, autor_id, local_id, datestamp)
```

Figura 3.6: Exemplo do esquema relacional do banco de dados do componente de navegação multidimensional (antes da modificação)

```
Autor(autor_id, autor)
Local(local_id, local)
Principal(pid, datestamp)
Principal_Autor(pid, autor_id, posicao)
Principal_Local(pid, local_id, posicao)
```

Figura 3.7: Exemplo do esquema relacional do banco de dados do componente de navegação multidimensional (depois da modificação)

3.4 Componente de Busca

Um serviço de busca, quer seja busca estruturada ou por palavras-chave, é uma parte importante de qualquer biblioteca digital, uma vez que provê ao usuário uma forma prática de ele encontrar a informação que deseja.

A arquitetura Fedora provê facilidades de busca em sua API, bem como uma interface de busca para o usuário final. No entanto, tal funcionalidade apresenta algumas limitações; por exemplo, o serviço não suporta a recuperação dos resultados de uma consulta ordenados por relevância. Para contornar esta deficiência, criamos um componente de busca que pode funcionar como um substituto para a API de busca nativa do Fedora, adicionando a capacidade de processar consultas baseadas em relevância e possibilitando a integração de dados estruturados e dados textuais em uma mesma consulta.

O componente de Busca possui três parâmetros de entrada: as palavras-chave da consulta, o tipo de busca e o número de documentos que serão retornados, e produz uma página Web com os resultados que satisfazem a consulta. O tipo de busca pode ser *booleana* (e/ou) ou por relevância, na qual os documentos são ordenados de acordo com sua similaridade com a consulta. A similaridade é calculada com base no modelo de espaço vetorial padrão (Salton e McGill, 1983), sem nenhuma modificação adicional. O número de documentos retornados limita o tamanho da resposta, uma vez que o usuário normalmente está interessado apenas nos documentos no topo da ordenação (ou seja, os mais similares à consulta). Opcionalmente, o usuário pode adicionar informação estruturada à consulta, o que funcionará como uma espécie de filtro sobre os resultados obtidos.

O componente de Busca funciona de forma semelhante ao componente de Exibição de Trabalhos Recentes, utilizando um banco de dados relacional para retornar os *pids* os objetos que satisfazem a consulta do usuário. Este banco de dados possui relações que simulam um arquivo invertido de uma máquina de busca e uma consulta SQL pode ser realizada nesse banco para se obter os *pids* desejados, conforme o trabalho de Grossman et al. (1997). Desta forma, existe uma tabela representando um documento da coleção (com seu identificador – *doc_id* – e seu peso – *weight*), uma tabela representando os termos da coleção (com o identificador do termo e seu *idf* (*inverse document frequency*)), uma tabela para representar a associação documento-termo (com a frequência do termo no documento – *tf*), e uma tabela utilizada por consultas estruturadas, que indica em qual campo de um documento um termo ocorre. Os termos

weight, *idf* e *tf* fazem parte da terminologia do modelo vetorial (Salton e McGill, 1983). A Figura 3.8 ilustra as relações descritas acima. O esquema de funcionamento do componente vai além de um simples mecanismo de busca baseado em consultas SQL, pois as consultas não funcionam por casamento exato, como geralmente ocorre quando se utiliza SQL, mas são capazes de ordenar os documentos por ordem similaridade destes com a consulta, a exemplo do que faz uma máquina de busca. Além disso, o casamento da consulta com o documento pode ser parcial.

Documento	
doc_id	weight
1	5,7
2	6,9
3	10,0
4	0,5

Termo	
termo	idf
a	4,6
b	1,3
c	0,8
d	9,9

Documento_Termo		
doc_id	termo	tf
1	c	2
1	b	1
2	b	3
3	c	5
3	a	5
4	d	1

Documento_Termo_Estruturado		
doc_id	campo	conteudo
1	creator	c
1	title	b
2	creator	b
3	creator	c
3	description	a
4	title	d

Figura 3.8: Exemplo do banco de dados do componente de Busca

A utilização de um banco de dados relacional torna possível realizar consultas que combinam dados estruturados e dados textuais. Por exemplo, pode-se procurar por objetos contendo os termos *based* e *zonas* (dados textuais) e ainda cujo nome do autor contém o fragmento *assis* (dado estruturado). Desta forma, se o usuário opta por habilitar a porção estruturada da consulta, ele pode informar um campo de dado (por exemplo, autor), um operador relacional (por exemplo, contém) e um valor (por exemplo, assis), e isto funcionará como um filtro sobre a porção textual. A consulta SQL gerada por este exemplo de busca é ilustrada a seguir (Figura 3.9) e a utilização da capacidade de integração de dados estruturados e dados textuais é mostrada na Figura 4.6 do Capítulo 4, também utilizando este mesmo exemplo.

```

SELECT D.fedora_pid, SUM(DT.tf * I.idf * I.idf) / D.weight AS rank
FROM doc D, doc_term DT, idf I, doc_term_struct DTS
WHERE I.term IN ("based", "zonas")
      AND D.doc_id = DT.doc_id
      AND DT.term = I.term
      AND D.doc_id = DTS.doc_id
      AND DTS.field = "dc:creator"
      AND DTS.content LIKE "\%assis\%"
GROUP BY DT.doc_id
ORDER BY rank DESC
LIMIT 30

```

Figura 3.9: Exemplo de uma consulta SQL gerada pelo Componente de Busca

3.5 Componentes de Registro de Usuário e de Anotação

Nesta seção descrevemos os componentes de Registro de Usuário e de Anotação que, dada sua simplicidade, são descritos conjuntamente.

O componente de Anotação fornece ao usuário uma forma de realizar anotações ou comentários a respeito dos itens da biblioteca digital. Entretanto, para fazer uma anotação, o usuário deve estar registrado no sistema. Sendo assim, foi desenvolvido um componente de Registro de Usuário para dar suporte ao componente de Anotação e a outros que possam vir a ser desenvolvidos no futuro, como um componente de Avaliação dos trabalhos.

O componente de Registro de Usuário é de fato muito simples. O usuário que deseja se registrar informa um e-mail sintaticamente válido e uma senha. O componente valida o e-mail e, se ele não existe no repositório, o armazena juntamente com a senha como um objeto digital do Fedora. Caso o e-mail já exista, o usuário é informado deste fato. Após esta validação, o usuário pode fazer anotações nos objetos.

Para fazer uma anotação, o usuário informa o objeto a ser anotado, seu e-mail e senha, e o texto da anotação. Os dados informados são validados: é verificado se o objeto existe, se o usuário é válido, se a senha está correta, etc. Ocorrendo a validação, o texto da anotação, sua data e o usuário que a realizou são armazenados no repositório do Fedora como um novo objeto digital.

Os objetos digitais criados pelos componentes de Registro de Usuário e de Anotação

não são visíveis a nenhum dos outros componentes (Busca, Trabalhos Recentes e Navegação), uma vez que estes objetos são armazenados no repositório do Fedora como objetos inativos e os demais componentes do arcabouço são projetados para ignorá-los.

Existe uma interface onde o usuário pode visualizar as anotações realizadas em um objeto digital. Esta interface exibe o título do objeto e suas anotações (data, usuário e texto), ordenadas cronologicamente.

O funcionamento destes componentes é similar ao dos demais. Os dados utilizados para validar a informação dada pelo usuário são obtidos através do *Fedora Research Index Query Service – rsearch*.

3.6 Componentes de Administração

Para que os componentes descritos anteriormente possam funcionar adequadamente, alguns componentes adicionais, que se relacionam basicamente com tarefas de administração, são necessários.

O primeiro desses componentes é o de inclusão de dados. A arquitetura Fedora provê uma ferramenta de administração que permite, entre outras funções, criar um objeto digital, incluir um objeto digital a partir de um arquivo e incluir um conjunto de objetos digitais a partir de um conjunto de arquivos contidos em um diretório. Os arquivos são documentos XML em um formato específico (*Fedora Object XML – FOXML*). No entanto, existem algumas dificuldades para a utilização desta ferramenta:

- Criar cada objeto digital manualmente não é uma solução viável, pois demanda tempo e esforço.
- Criar um arquivo FOXML para cada objeto pode ser uma tarefa complexa.
- Inserir cada arquivo FOXML é uma atividade que consome muito tempo, uma vez que cada inserção será realizada por um novo processo. Assim, por exemplo, ao final de 5.000 inserções, 5.000 processos terão sido criados, executados e destruídos.
- Inserir um conjunto de arquivos FOXML com a ferramenta pode não ser a melhor opção, pois pode ser necessário inserir os objetos em uma certa ordem, o que pode não ser respeitado por esta função.

No caso de nosso arcabouço, a ordem de inserção é muito importante por causa das coleções, pois o registro representando a coleção, isto é, contendo informações como o nome e o ano da coleção, deve ser inserido antes dos registros que pertencem à coleção, uma vez que estes registros fazem referências em seus relacionamentos RDF ao registro da coleção. Com a ferramenta de administração do Fedora pode ocorrer de um registro ser inserido antes do registro da sua coleção, o que levaria à falha do processo, já que o Fedora tentaria criar um relacionamento para um objeto que ainda não existe no repositório.

Para resolver estes problemas, o componente de inclusão de dados recebe como parâmetro de entrada um arquivo XML – em um formato similar àquele retornado por uma requisição *ListRecords*⁶ do protocolo OAI – contendo todos os registros a serem inseridos no repositório. A partir deste arquivo, o componente gera um conjunto de arquivos FOXML, com os nomes de *object0*, *object1*, *object2* e assim por diante, onde os sufixos dos arquivos (0, 1, 2, ...) são utilizados para preservar a ordem de inserção. Deste modo, o *object0* é o primeiro a ser inserido, seguido de *object1*, *object2*, etc. A inserção é realizada com a ajuda da API do Fedora.

Um outro componente é utilizado para criar um banco de dados que serve como fonte para os bancos de dados dos componentes de Busca e de Navegação. O banco de dados fonte funciona como uma camada intermediária entre o Fedora e os bancos de dados dos componentes. Se tal banco não existisse, os bancos de dados dos componentes teriam que se comunicar diretamente com o Fedora para obter os dados necessários a partir do repositório. Uma vez que esta comunicação, baseada em HTTP, é mais lenta que uma comunicação via um banco de dados, optou-se por armazenar os dados em um banco de dados intermediário, para melhorar o desempenho no momento de criar os bancos de dados dos componentes.

Este componente também melhora a generalidade do arcabouço, uma vez que há a flexibilidade de se substituir o provedor de dados OAI do Fedora por qualquer outro componente com a mesma funcionalidade, sem que isso afete a criação dos bancos de dados. Deste modo, não existe uma dependência da arquitetura Fedora.

Basicamente, este componente realiza requisições OAI ao provedor de dados do Fedora e armazena os resultados no banco de dados, que é então utilizado pelos bancos de dados dos componentes. As requisições são incrementais, o que significa que apenas novos conteúdos são recuperados do repositório. No caso de objetos que sofreram

⁶ <http://www.openarchives.org/OAI/openarchivesprotocol.html#ListRecords>

modificações e foram versionados pelo Fedora, apenas a última versão é considerada.

A partir do banco de dados fonte descrito acima e de um metamodelo expresso em XML, um terceiro componente constrói o banco de dados de navegação, utilizado pelo componente de Navegação Multidimensional (Seção 3.3). A construção deste banco de dados também ocorre de maneira incremental.

O último dos componentes é o que cria o banco de dados utilizado pelo componente de Busca. É possível informar alguns parâmetros para a criação do banco, como os delimitadores entre as palavras, a lista de *stop words* e quais são os campos a serem indexados (por exemplo, título, data, autor, etc.). O resultado final é um banco de dados nos moldes daquele ilustrado na Figura 3.8.

A Figura 3.10 ilustra o funcionamento dos componentes de administração. O componente de inclusão de dados recebe um arquivo fonte e insere seu conteúdo no repositório do Fedora. Um segundo componente obtém registros OAI do repositório e gera um banco de dados fonte a partir desses registros. Dois outros componentes geram os bancos de dados dos componentes de Busca e de Navegação, a partir do banco de dados fonte.

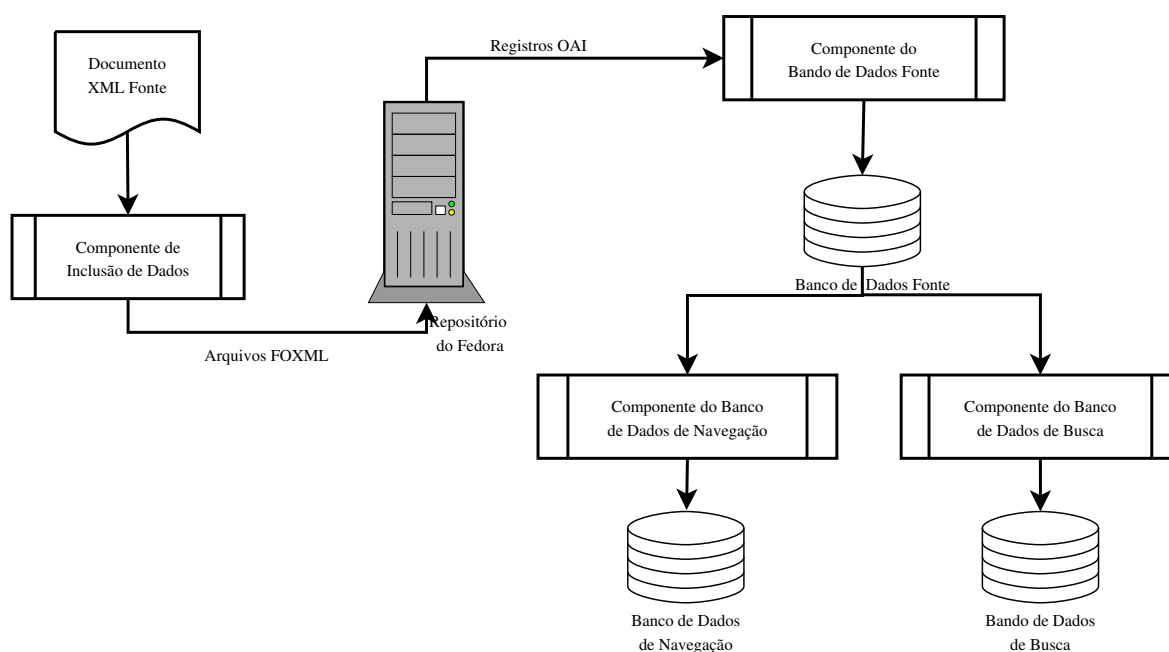


Figura 3.10: Componentes de administração

Capítulo 4

Exemplos de Utilização

Neste capítulo são mostrados alguns exemplos de utilização dos componentes desenvolvidos, com os quais foi criado um protótipo de biblioteca digital com funcionalidade similar àquela apresentada pela BDBComp¹ (Laender et al., 2004), uma biblioteca digital de trabalhos científicos publicados em anais de conferências e periódicos da área de ciência da computação realizados no Brasil.

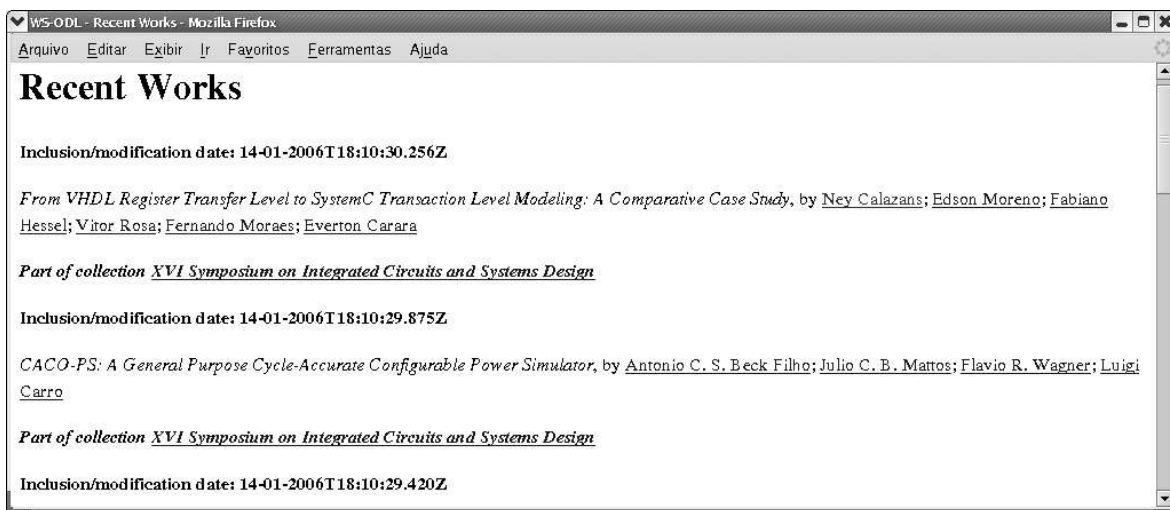


Figura 4.1: Página de trabalhos recentes

A Figura 4.1 mostra a página gerada pelo componente de Exibição Trabalhos Recentes, na qual os trabalhos são exibidos de acordo com a sua data de inclusão ou modificação no repositório, ordenados cronologicamente do mais para o menos recente. A figura

¹<http://www.lbd.dcc.ufmg.br/bdbcomp>

também ilustra a estrutura navegacional do arcabouço, com *links* nos nomes de autores e de coleções, quando estas existem. Desta forma, seguindo-se nesta página o *link* para o *XVI Symposium on Integrated Circuits and Systems Design*, alcança-se a página desta coleção, exibida na Figura 4.2. A partir desta página, seguindo-se o link *Fernando Silveira*, chega-se à página de trabalhos deste autor, que é mostrada na Figura 4.3.

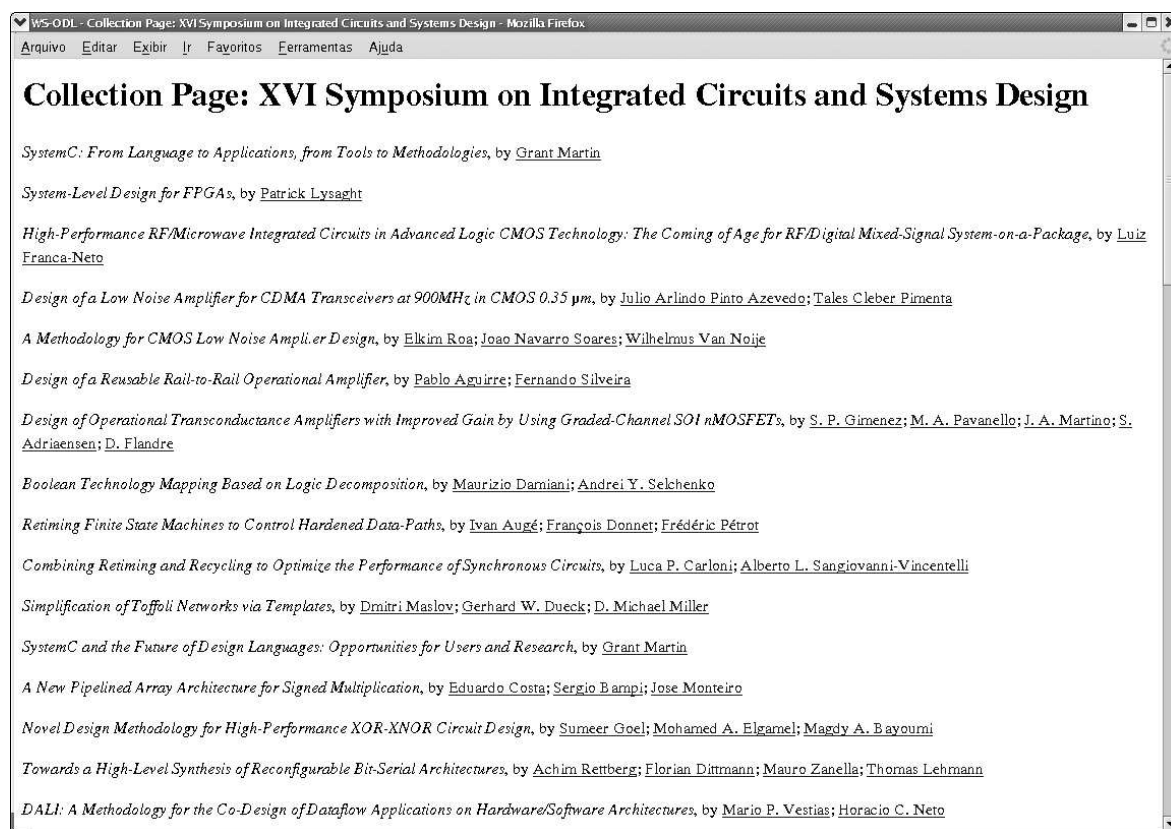


Figura 4.2: Página de trabalhos da coleção

A Figura 4.4 mostra a página inicial gerada pelo componente de Navegação Multidimensional, onde é possível observar a capacidade de navegação por local da publicação, data da publicação e autor (campos *dc:coverage*, *dc:date* e *dc:creator* do *Dublin Core*, respectivamente). Já a Figura 4.5 exibe a página da coleção *XVI Symposium on Integrated Circuits and Systems Design* gerada por este componente. Pode-se notar que a informação é idêntica àquela mostrada na Figura 4.2, o que demonstra que é possível gerar a página de trabalhos da coleção (e também de autor) de duas formas distintas.

A Figura 4.6 apresenta a interface de busca. Na consulta de exemplo, deseja-se recuperar os 30 documentos mais relevantes contendo os temas *based* e *zonas*, mas apenas se o campo *dc:creator* contiver o fragmento *assis*. O resultado da consulta é mostrado

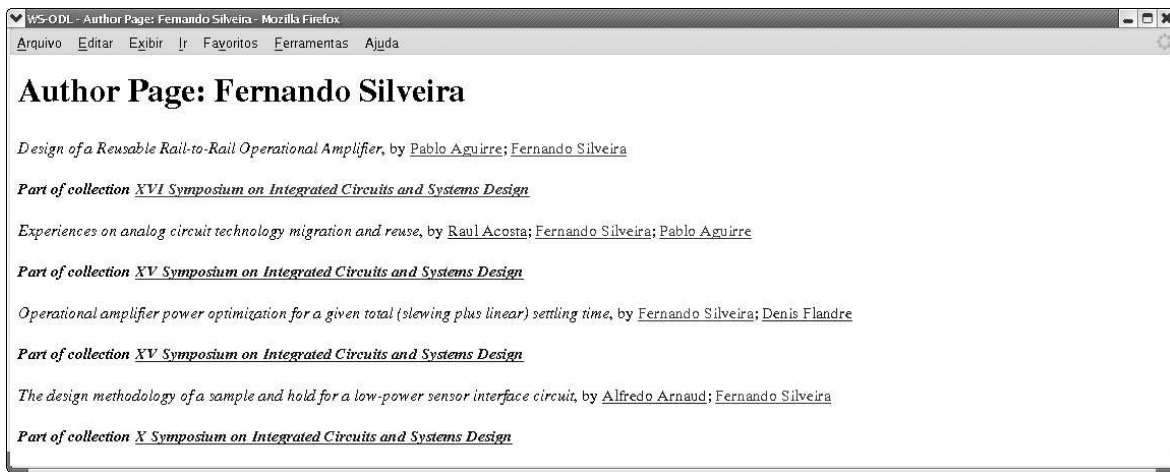


Figura 4.3: Página de trabalhos do autor

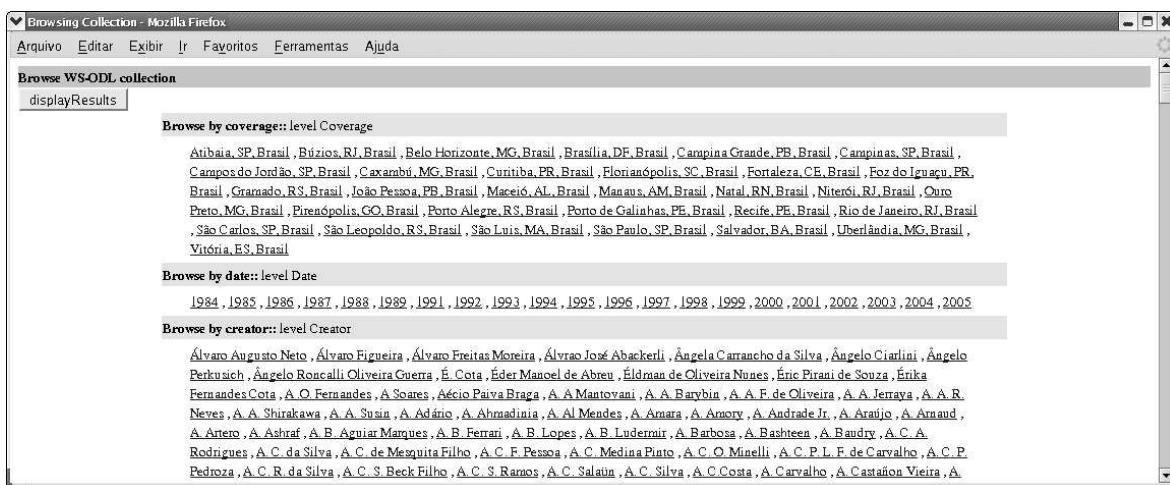


Figura 4.4: Página inicial do componente de navegação multidimensional

na mesma figura. Pode-se observar que o trabalho *Zonas de desenvolvimento proximal, autonomia e atividades insterpsicologicas: princípios do projeto Khouse Open Br* aparece como o documento mais relevante (o do topo da lista), o que é um resultado esperado, segundo o modelo de espaço vetorial, uma vez que este trabalho é o único no repositório que contém o termo *zonas*. Também pode ser observada a influência da parte estruturada da consulta. Embora o usuário tenha solicitado a recuperação dos 30 documentos mais relevantes, apenas 3 satisfizeram a restrição estruturada, e foram exibidos para o usuário. O segundo e o terceiro trabalhos apresentados na figura não exibem nenhum dos termos na consulta nos dados apresentados ao usuário (título, autores e nome da coleção). No entanto, eles possuem algum desses termos em outros campos que foram utilizados para indexação mas não estão sendo exibidos ao usuário, devido à formatação do *layout* definida. Um destes campos é o *dc:description*, que contém o resumo dos trabalhos.

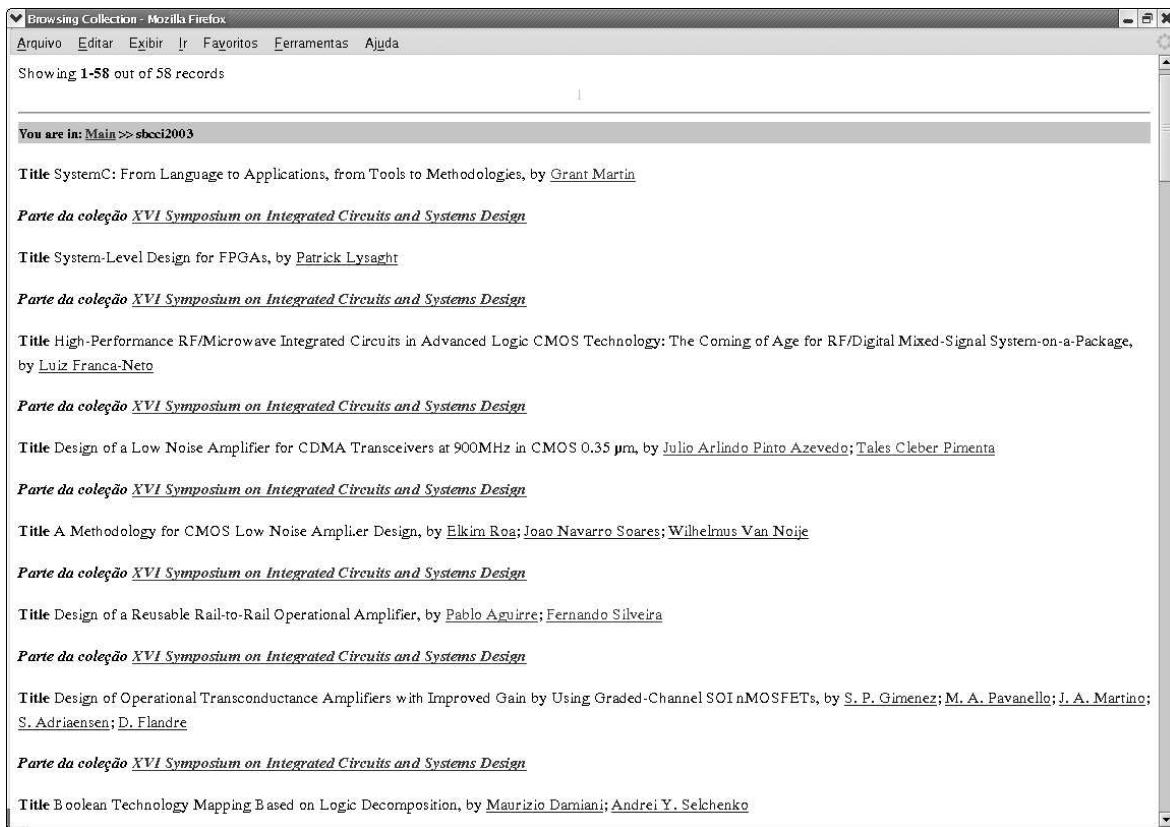


Figura 4.5: Página da coleção – componente de navegação multidimensional

Por fim, a Figura 4.7 apresenta exemplos relacionados à realização de anotações nos objetos. São mostradas três telas: a primeira ilustra o processo de registro de um usuário; a segunda, a realização de uma anotação em um objeto e a terceira, as anotações para o objeto.

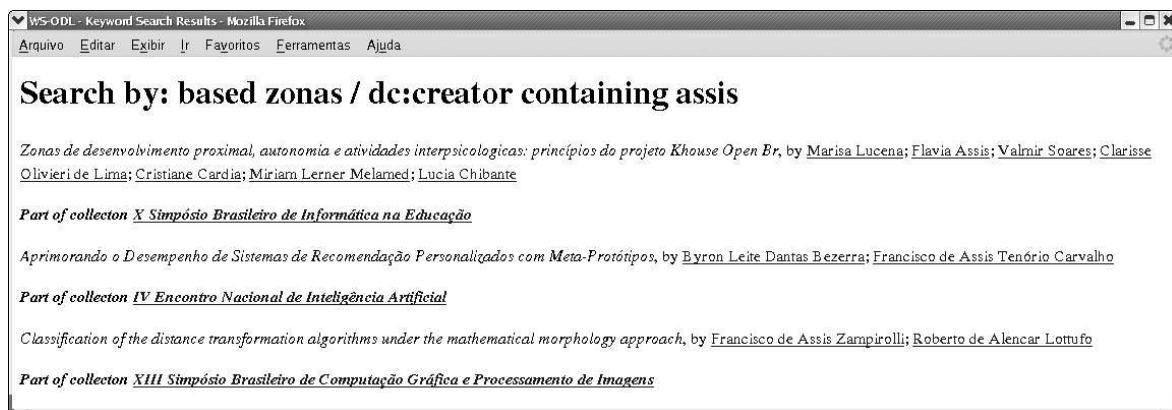
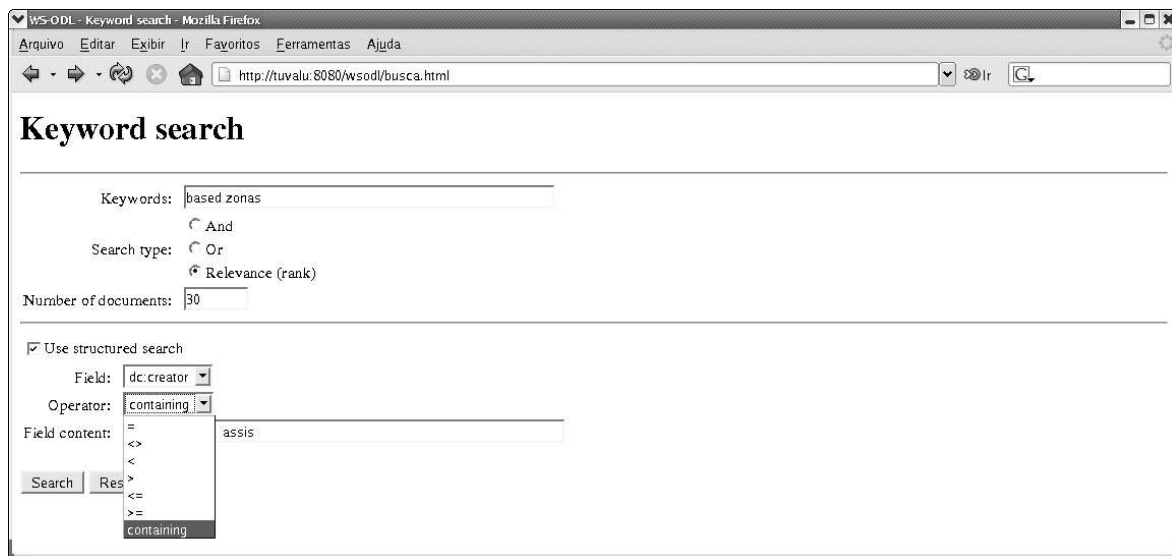


Figura 4.6: Consulta integrando dados estruturados e dados textuais

WS-ODL - Insert user - Mozilla Firefox

Arquivo Editar Exibir Ir Favoritos Ferramentas Ajuda

Insert user

E-mail: pabloa@dcc.ufmg.br

Password: *****

Insert Reset

WSODL - Annotate Work - Mozilla Firefox

Arquivo Editar Exibir Ir Favoritos Ferramentas Ajuda

Annotate Work

Object PID: wsodl:4068

E-mail: pabloa@dcc.ufmg.br

Password: *****

Annotation: This is a very good digital library framework, which uses Java Servlets, Web Services, Open Archives and the component based approach.

Annotate Reset

WS-ODL - Work Annotations: Um Arcabouço Baseado em Componentes, Serviços Web e Arquivos Abertos para Criação de Bibliotecas Digitais - Mozilla Firefox

Arquivo Editar Exibir Ir Favoritos Ferramentas Ajuda

http://tuvalu:8080/wsodl/AnotacoesTrabalho?pid=wsodl:4068

WS-ODL - Work Annotations: Um Arcabouço Baseado em Componentes, Serviços Web e Arquivos Abertos para Criação de Bibliotecas Digitais

Date: 2006-04-28T23:32:26.932

Author: mgoncalv@dcc.ufmg.br

Annotation: But there are still some points to improve.

Date: 2006-04-28T23:30:11.787

Author: pabloa@dcc.ufmg.br

Annotation: This is a very good digital library framework, which uses Java Servlets, Web Services, Open Archives and the component based approach.

Figura 4.7: Registro de usuário e anotações do trabalho

Capítulo 5

Avaliação e Análise do Arcabouço

Com o objetivo de verificar a viabilidade do arcabouço, foram realizados vários experimentos com o mesmo. Estes incluem um experimento de instalação, um experimento de carga de dados e um experimento de desempenho. Os resultados obtidos são apresentados e discutidos a seguir.

5.1 Experimento de Instalação

O primeiro dos experimentos realizados foi um experimento de instalação que teve por objetivo mensurar o quão difícil é a instalação e configuração do arcabouço, principalmente para usuários não-especialistas. Para tanto, usuários receberam um roteiro de instalação, contendo instruções para instalar e configurar todas as partes do arcabouço.

O experimento foi realizado com oito usuários, no papel de administradores de uma biblioteca digital, sendo quatro usuários da área de Ciência da Computação (CC) e quatro da área de Ciência da Informação (CI). A Tabela 5.1 apresenta informações, obtidas através de um questionário, sobre a experiência e o conhecimento desses usuários.

O tempo gasto no experimento foi medido em cinco pontos de verificação que compreendem macro-fases do processo de instalação. O número de erros cometidos pelos usuários também foi registrado. Os pontos de verificação considerados foram:

Pergunta	Respostas
Formação	Ciência da Computação: 4 Ciência da Informação: 4
Grau de Instrução	Graduação em Andamento: 3 Graduação Concluída: 2 Pós Graduação em Andamento: 1 Pós Graduação Concluída: 2
Você já utilizou Linux anteriormente? (fazer <i>login</i> , manipular arquivos, configurar variáveis de ambiente)	Sim: 7 Não: 1
Você já utilizou algum ambiente de desenvolvimento baseado em componentes anteriormente? (Delphi, Eclipse, VB, etc.)	Sim: 6 Não: 2
Você já desenvolveu aplicações web? (CGIs, <i>servlets</i> , JSPs, etc.)	Sim: 6 Não: 2
Você sabe como serviços web funcionam? (por exemplo, envolvendo SOAP, WSDL, UDDI)	Sim: 4 Não: 4
Você possui experiência de desenvolvimento utilizando serviços web?	Sim: 4 Não: 4

Tabela 5.1: Experiência e conhecimento dos usuários

- A – Instalação e configuração da arquitetura Fedora.
- B – Instalação do Apache Axis¹.
- C – Instalação dos serviços web no Axis.
- D – Instalação dos *servlets*.
- E – Inserção de dados na biblioteca digital sendo construída.

A Tabela 5.2 mostra os tempos gastos pelos usuários em cada ponto de verificação da tarefa, enquanto a Tabela 5.3 mostra o número de erros cometidos. Os valores da média e do desvio padrão para as duas medidas, por grupo de usuários e total, também são mostrados.

Como pode ser observado a partir dos dados das tabelas, o processo de instalação não é muito trivial, uma vez que os usuários necessitam de quase duas horas para completá-lo. A principal razão por trás disso é o fato de o processo ser inteiramente baseado em linha de comando. Não existe uma ferramenta gráfica disponível para ajudar o usuário

¹O Axis é uma plataforma Java para criação e distribuição de aplicações baseadas em serviços web. Mais informações podem ser obtidas em <http://ws.apache.org/axis/>

Usuários de Ciência da Computação						
Usuário	Pontos de Verificação					Total
	A	B	C	D	E	
CC1	0:41	0:11	0:36	0:08	0:19	1:55
CC2	0:35	0:12	0:28	0:09	0:18	1:42
CC3	0:35	0:10	0:20	0:07	0:17	1:29
CC4	0:30	0:15	0:20	0:15	0:25	1:45
Média	0:35:15	0:12:00	0:26:00	0:09:45	0:19:45	1:42:45
Desvio Padrão	0:04:30	0:02:10	0:07:40	0:03:36	0:03:36	0:10:43

Usuários de Ciência da Informação						
Usuário	Pontos de Verificação					Total
	A	B	C	D	E	
CI1	0:40	0:15	0:35	0:06	0:22	1:58
CI2	0:26	0:08	0:19	0:06	0:12	1:11
CI3	1:04	0:17	0:35	0:08	0:20	2:24
CI4	0:40	0:15	0:30	0:10	0:25	2:00
Média	0:42:30	0:13:45	0:29:45	0:07:30	0:19:45	1:53:15
Desvio Padrão	0:15:47	0:03:57	0:07:33	0:01:55	0:05:34	0:30:33

Todos os Usuários						
	Pontos de Verificação					Total
	A	B	C	D	E	
Média	0:38:53	0:12:52	0:27:52	0:08:38	0:19:45	1:48:00
Desvio Padrão	0:11:25	0:03:05	0:07:19	0:02:55	0:04:20	0:21:55

Tabela 5.2: Tempo gasto pelos usuários no teste de instalação

a instalar e configurar o arcabouço. Esta também é a razão pela qual os usuários cometeram a maior parte dos erros.

Pode-se notar que a tarefa mais complexa é a instalação e configuração do Fedora, consumindo aproximadamente 36% do tempo médio gasto e sendo responsável por 34% do número médio de erros. Em seguida, temos a instalação dos serviços web no Axis, com 25,8% do tempo médio gasto e 32% do número médio de erros. Isto ocorre porque justamente estas duas tarefas são as mais intensivas em termos de uso da linha de comando. Além disso, a instalação do Fedora é uma tarefa naturalmente complexa, devido às atividades envolvidas, com a configuração do banco de dados, parâmetros de uso, etc.

Com relação aos tipos de erros cometidos pelos usuários, os que ocorrem com mais frequência são os de digitação incorreta de comandos, seguidos pelos erros de “esqueci-

Usuários de Ciência da Computação						
Usuário	Pontos de Verificação					Total
	A	B	C	D	E	
CC1	10	2	6	2	2	22
CC2	7	2	5	1	1	16
CC3	5	3	3	0	2	13
CC4	4	1	4	2	3	14
Média	6,50	2,00	4,50	1,25	2,00	16,25
Desvio Padrão	2,65	0,82	1,29	0,96	0,82	4,03

Usuários de Ciência da Informação						
Usuário	Pontos de Verificação					Total
	A	B	C	D	E	
CI1	14	6	18	2	7	47
CI2	2	1	3	1	0	7
CI3	17	5	12	2	7	43
CI4	9	4	13	2	10	38
Média	10,50	4,00	11,50	1,75	6,00	33,75
Desvio Padrão	6,56	2,16	6,24	0,50	4,24	18,21

Todos os Usuários						
	Pontos de Verificação					Total
	A	B	C	D	E	
Média	8,50	3,00	8,00	1,50	4,00	25,00
Desvio Padrão	5,10	1,85	5,61	0,76	3,55	15,38

Tabela 5.3: Erros cometidos pelos usuários no teste de instalação

mento”, onde os usuários se esqueciam de qual local obter um recurso para a tarefa ou mesmo saltavam passos inteiros do roteiro de instalação.

Comparando-se os dois grupos de usuários, é possível observar que os usuários da área de Ciência da Computação (grupo CC) completaram a tarefa com menos da metade do número de erros cometidos pelos usuários da área de Ciência da Informação (grupo CI). Entretanto, o tempo gasto pelo primeiro grupo é apenas ligeiramente menor que o tempo gasto pelo segundo. Existe uma correlação entre estes dois fatos: durante o experimento, observamos que os usuários do grupo CC foram mais cuidadosos na execução da tarefa, prestando mais atenção ao roteiro de instalação, o que os levou a cometerem menos erros. Por outro lado, os usuários do grupo CI não foram tão cuidadosos; conseqüentemente, cometeram mais erros e gastaram mais tempo. Entretanto, os usuários do primeiro grupo gastaram um tempo considerável lendo e assimilando o roteiro e, ao final da tarefa, isto fez com que seu tempo fosse quase o mesmo daquele

gasto pelo segundo grupo.

Fora estas observações, o melhor desempenho dos usuários do grupo CC era esperado, uma vez que estes usuários geralmente estão mais acostumados a este tipo de tarefa e também a lidar com computadores. Uma evidência deste fato é que o tempo e o número de erros cometidos por estes usuários variou muito menos neste grupo do que no grupo CI. Apesar do pior desempenho dos usuários do grupo CI, consideramos que os resultados alcançados por eles indicam que o arcabouço, se melhorado para prover algum tipo de ferramenta de instalação, pode ser utilizado por usuários que não sejam especialistas em Ciência da Computação.

Ao término da tarefa, foi solicitado aos usuários que preenchessem um questionário de avaliação, com o objetivo de se obter suas impressões sobre o processo de instalação e o grau de confiabilidade de suas respostas, com base em suas experiências e conhecimento anteriores a respeito dos conceitos e tecnologias envolvidos na tarefa. Os resultados obtidos são apresentados nas Tabelas 5.1 e 5.4

Dois dos usuários que indicaram não saber como serviços web funcionam responderam afirmativamente quando questionados se possuíam experiência de desenvolvimento utilizando esta tecnologia. Isto pode ter ocorrido devido ao fato de estes usuários já terem desenvolvido aplicações sem conhecer os conceitos por trás dos serviços web, o que é consistente com o vago entendimento que muitas pessoas possuem desta tecnologia.

A primeira pergunta da Tabela 5.4 mostra que os usuários não conhecem os conceitos que permeiam nosso arcabouço. Isto era um resultado esperado, uma vez que eles não receberam nenhuma informação previamente ao experimento. Mesmo assim, todos os usuários foram capazes de completar o experimento e a grande maioria concorda que as instruções foram claras. De fato, as instruções são detalhadas o suficiente para que mesmo que os usuários não compreendam todos os passos da tarefa possam chegar ao final da mesma. Além disso, a execução do roteiro melhorou o entendimento de 37,5% dos participantes sobre o arcabouço.

Apenas 25% dos usuários consideraram que a instalação e configuração do arcabouço é simples. Novamente, este era um resultado esperado, uma vez que todo o processo ocorre via linha de comando, uma interface nada intuitiva para a maioria dos usuários.

Apesar de todos estes fatores, 87,5% dos usuários indicaram que considerariam utilizar o arcabouço caso necessitassem construir uma biblioteca digital similar à BDBComp. Assim, mesmo sem um entendimento dos conceitos do arcabouço e com outras opções

disponíveis, a natureza simples e reusável dos componentes, retirando dos usuários a responsabilidade da codificação do sistema, parece ser atrativa a eles.

Seis dos oito usuários preencheram o campo opcional de comentários a respeito do arcabouço e do processo de instalação. Dois dos usuários observaram a falta de uma explicação sobre o arcabouço e afirmaram que a tarefa teria sido feita de forma mais eficaz se conhecessem os conceitos por trás do arcabouço. Dois usuários afirmaram que não possuíam domínio do ambiente Linux, o que dificultou a execução da tarefa. Um usuário afirmou que as instruções poderiam ser mais refinadas e outro que o roteiro estava mal formatado em alguns pontos (notadamente, quando era necessário informar comandos que ultrapassavam a capacidade de uma linha de texto). Quatro usuários informaram que o roteiro é simples, claro e completo, possibilitando o cumprimento da tarefa.

Pergunta	Respostas				
	Concordo Fortemente	Concordo	Neutro	Discordo	Discordo Fortemente
Eu entendo os conceitos do arcabouço		1	4	3	
As instruções do roteiro são claras	3	4		1	
Instalar o arcabouço é simples		2	2	4	
Configurar o arcabouço é simples		2	3	3	
Executar o roteiro de instalação melhorou meu entendimento sobre o arcabouço		3	4	1	
Você consideraria utilizar o arcabouço caso necessitasse construir uma biblioteca digital similar à BDBComp	2	5	1		

Tabela 5.4: Respostas ao questionário da tarefa de instalação

5.2 Experimento de Carga de Dados

O segundo experimento realizado foi o de carga de dados, cujo o objetivo foi medir o tempo gasto para se inserir toda uma coleção no repositório do arcabouço. O tempo gasto foi medido para seis atividades que fazem parte deste processo:

1. Transformação de um arquivo de entrada contendo os registros em um único arquivo FOXML.
2. Divisão do arquivo FOXML em vários outros FOXML, um por objeto digital.
3. Inserção dos objetos no repositório do Fedora.
4. Criação do banco de dados fonte.
5. Criação do banco de dados de navegação.
6. Criação do banco de dados de busca.

O experimento foi repetido 10 vezes para cada atividade e o tempo médio de execução calculado.

Três coleções de dados foram utilizadas: BDBComp (com 4142 objetos), LDB (com 29480 objetos) e CiteSeer² (com 574901 objetos). A coleção LDB é formada pela união de três fontes de dados: Lattes³, DBLP⁴ e BDBComp, com os objetos repetidos sendo descartados e considerando apenas a comunidade de ciência da computação, uma vez que o Lattes é um sistema de informação curricular utilizado por toda a comunidade científica brasileira. A máquina utilizada para este experimento foi um computador Pentium 4 HT 3.2GHz de 32 *bits*, com 2GB de memória RAM, disco rígido de 400GB e sistema operacional Linux Red Hat Fedora Core 3.

Os tempos obtidos são mostrados na Tabela 5.5. Para a coleção de dados do CiteSeer, foi necessário executar um passo inicial de preparação dos dados (*A0*), que não é necessário para as demais coleções. Este passo consiste em, para cada grupo de objetos (um documento XML contendo 100 objetos digitais), inserir uma *tag* raiz no documento (denominada *<ListRecords>*) e realizar uma limpeza de dados (remoção de caracteres fora das *tags* do documento, que causam a falha do *parsing*).

²<http://citeseer.ist.psu.edu/>

³<http://lattes.cnpq.br/>

⁴<http://dblp.uni-trier.de/>

Atividade	BDBComp	LDB	CiteSeer
A0	-	-	367,80
A1	4,90	23,24	9648,00
A2	125,15	248,54	6768,00
A3	1362,60	4238,05	-
A4	49,30	294,82	-
A5	117,00	3333,32	-
A6	42,34	668,60	-
Tempo Total	1701,29	8806,87	-

Tabela 5.5: Tempo médio de carga dos dados (em segundos)

A partir dos dados apresentados na Tabela 5.5, pode-se observar que a tarefa mais demorada é a inserção de objetos no repositório do Fedora (A3). Isto acontece porque, para cada objeto inserido, o Fedora cria e armazena várias triplas RDF no *triplestore* Kowari utilizado para manter os relacionamentos entre objetos. Quanto maior e mais relacionados são os objetos, maior será o número de triplas criado. Em nosso caso, em média 30 triplas são geradas por objeto e isto causa grande impacto no desempenho desta atividade. Além disso, a versão do Kowari utilizada pelo Fedora 2.0 (a versão que nosso arcabouço utiliza atualmente) possui um problema de gerenciamento de memória (*memory leak*), o que também leva a problemas de desempenho, especialmente com grande volumes de dados. A cada inserção de um objeto digital no repositório, uma porção da memória utilizada nesta operação não é liberada pelo Kowari. Deste modo, eventualmente, ocorre um problema de falta de memória para prosseguir as inserções. Mesmo realizando-se cargas incrementais, ou seja, inserindo-se uma certa quantidade de objetos (por exemplo, 5000) e reiniciando o Fedora a cada carga, o problema também ocorre, com a diferença de que neste caso é possível realizar um total de inserções maior do que aquele de uma carga não incremental.

Para a coleção de dados do CiteSeer, o processo de carga em sua totalidade não pôde ser completado. Uma das razões é o problema de gerenciamento de memória citado anteriormente. A outra está relacionada ao tamanho da coleção. Como ela possui mais de 570 mil objetos, o processo de geração das triplas RDF que acontece na inserção dos objetos no Fedora causa a exaustão do espaço de endereçamento da máquina utilizada, levando a máquina virtual Java (JVM), utilizada para executar o arcabouço, a ser abortada e lançar uma exceção de falta de memória. Uma vez que a inserção dos objetos é um pré-requisito para os passos subseqüentes, estes também não puderam ser executados.

Um dado interessante refere-se à variação do tempo de carga com o tamanho das coleções. Dividindo-se o tempo de carga para a coleção LDB pelo tempo de carga da coleção BDBComp, obtemos a tabela abaixo:

Atividade	Aumento do Tempo
A1	4,74
A2	1,98
A3	3,11
A4	5,98
A5	28,49
A6	15,79
Total	5,17

Tabela 5.6: Aumento do tempo de carga da coleção LDB em relação à coleção BDBComp

É possível observar que para quase todas as atividades, o aumento do tempo de carga foi menor que o aumento do tamanho da coleção, que foi de 7,11 vezes. Além disso, este aumento não foi linear. No entanto, duas das atividades, que envolvem a criação dos bancos de dados de navegação (*A5*) e de busca (*A6*) tiveram aumento de tempo bastante superior ao aumento do tamanho da coleção. Este fato pode estar relacionado com a forma de trabalho dos componentes que criam estes bancos, que trabalham com DOM – *Document Object Model*⁵, uma tecnologia para manipulação de documentos XML de forma hierárquica (i.e., como uma estrutura de árvore). Uma característica do DOM é trabalhar com a árvore que representa o documento em memória. Embora os documentos XML tratados nestes casos não sejam grandes, podem estar ocorrendo *memory leaks* que degradam o desempenho, principalmente na criação do banco de dados de navegação, cujo componente foi originalmente desenvolvido pelo grupo de bibliotecas digitais da Virginia Tech. Uma investigação mais minuciosa nestes componentes pode levar à melhoria do tempo de execução. Uma outra solução é utilizar outra tecnologia para manipular os documentos XML, como SAX – *Simple API for XML*⁶. Apesar do desempenho ruim destas duas atividades, o tempo de carga total ainda aumentou menos que o aumento do tamanho da coleção.

Para fins de comparação, medimos o tempo de carga gasto pelo arcabouço ODL, para gerar seus bancos de dados para os componentes de Exibição de Trabalhos Recentes, Busca e Navegação, com as coleções BDBComp e LDB. Para completar a tarefa, o ODL levou 1740 segundos para a primeira coleção e 11520 segundos para segunda. Em

⁵<http://www.w3.org/DOM/>

⁶<http://www.saxproject.org/>

relação à BDBComp, que é uma coleção pequena, os dois arcabouços são praticamente idênticos, com uma pequena vantagem do nosso. Entretanto, em relação à coleção LDB, que é uma coleção maior, o processo de carga de dados no nosso arcabouço é cerca de 1,31 vezes mais rápido que o do ODL. Uma das razões por trás deste fato é que o ODL faz o *harvesting* da coleção para cada um dos seus componentes, enquanto nosso arcabouço o faz apenas uma vez (para o banco de dados fonte), e utiliza a infraestrutura fornecida pelo Fedora sempre que possível.

5.3 Experimento de Desempenho

O terceiro experimento foi uma comparação de tempo entre os componentes de nosso arcabouço e aqueles que fazem parte do arcabouço ODL. Conforme mencionado anteriormente, um de nossos objetivos neste trabalho era atacar os problemas de desempenho apresentados pelo arcabouço ODL, que serviu como ponto de partida para o desenvolvimento de nosso trabalho. Sendo assim, uma comparação com o ODL é o caminho natural para verificar se obtivemos sucesso ou não.

Utilizando o *Apache jMeter*⁷, medimos os tempos de resposta das requisições HTTP feitas aos componentes. Foram realizados dois tipos de medida: o tempo gasto por cada componente para retornar os *pids* de interesse e o tempo gasto por cada componente para mostrar a resposta ao usuário. Para o ODL, o primeiro tipo de medida significa o tempo gasto por um componente para produzir um documento XML contendo os *pids* de interesse, baseado na requisição XOAI correspondente; para o nosso arcabouço, significa o tempo de resposta do serviço web. Já o segundo tipo de medida significa o tempo gasto por um componente para produzir uma página web contendo os resultados a serem exibidos para o usuário.

Foram comparados os componentes de Exibição de Trabalhos Recentes, Busca e Navegação. Para os dois primeiros, foram requisitados 100 registros como resultado. Para o componente de Navegação, requisitamos uma página de trabalhos do autor, contendo 58 registros, e uma página de trabalhos da coleção, com 152 registros. Também foram realizadas requisições para medir o tempo gasto pelo componente de Navegação Multidimensional para criar estas páginas. Em todos os casos, foram realizadas 100 requisições, tomando-se o tempo médio e calculando-se o desvio padrão.

⁷<http://jakarta.apache.org/jmeter/>

A Tabelas 5.7 e 5.8 apresentam os resultados para a coleção LDB. Na primeira tabela são mostrados os tempos médios gastos pelos componentes para completar a tarefa. Já a segunda tabela exibe o ganho dos componentes do nosso arcabouço sobre os componentes do ODL. Desta forma, o valor $(+)$ x indica que um componente do nosso arcabouço é x vezes mais rápido que o respectivo componente do arcabouço ODL. Similarmente, $(-)$ x indica que um componente do nosso arcabouço é x vezes mais lento.

Existem quatro tipos de entrada nas tabelas:

- Entradas MD, pertinentes ao componente de navegação multidimensional.
- Entradas ODL, relativas aos componentes do arcabouço ODL.
- Entradas WS-ODL, representando componentes do arcabouço WS-ODL.
- Entradas WS-ODL 2, que representam uma variação das entradas WS-ODL, conforme explicado a seguir.

Componente	Tempo PID (ms)		Tempo Página (ms)	
	Média	Desvio Padrão	Média	Desvio Padrão
ODL Trabalhos Recentes	232,55	5,96	1748,18	144,48
WS-ODL Trabalhos Recentes	31,59	3,58	3312,18	120,52
WS-ODL 2 Trabalhos Recentes	31,59	3,58	465,63	43,20
ODL Busca	229,81	17,29	1068,64	43,65
WS-ODL Busca	54,44	4,38	3590,52	58,08
WS-ODL 2 Busca	54,44	4,38	447,68	7,29
ODL Nav. Autor	180,35	18,93	662,55	14,12
WS-ODL Nav. Autor	38,02	5,73	1598,11	33,67
WS-ODL 2 Nav. Autor	38,02	5,73	249,58	6,32
MD Nav. Autor	4240,92	24,78	6432,47	63,42
ODL Nav. Coleção	169,57	16,20	1466,13	36,49
WS-ODL Nav. Coleção	60,11	3,87	676,97	13,70
WS-ODL 2 Nav. Coleção	60,11	3,87	643,76	14,84
MD Nav. Coleção	30,51	6,27	6710,94	127,11

Tabela 5.7: Tempo de resposta dos componentes para a coleção LDB

Pode ser observado a partir dos resultados obtidos que o arcabouço que desenvolvemos é muito mais rápido que o ODL, no que diz respeito à obtenção dos *pids* de interesse. Com relação ao tempo gasto para se gerar as páginas Web, nosso arcabouço é mais lento que o ODL em todos menos um caso. Entretanto, isto ocorre porque nossos componentes

Componente	Ganho sobre ODL	
	Tempo PID	Tempo Página
WS-ODL Trabalhos Recentes	(+) 7,36	(-) 1,89
WS-ODL 2 Trabalhos Recentes	(+) 7,36	(+) 3,75
WS-ODL Busca	(+) 4,22	(-) 3,36
WS-ODL 2 Busca	(+) 4,22	(+) 2,39
WS-ODL Nav. Autor	(+) 4,74	(-) 2,41
WS-ODL 2 Nav. Autor	(+) 4,74	(+) 2,65
MD Nav. Autor	(-) 23,51	(-) 9,71
WS-ODL Nav. Coleção	(+) 2,82	(+) 2,16
WS-ODL 2 Nav. Coleção	(+) 2,82	(+) 2,28
MD Nav. Coleção	(+) 5,56	(-) 4,58

Tabela 5.8: Ganho relativo sobre o ODL

realizam uma carga de trabalho consideravelmente maior que os componentes ODL. Por exemplo, se um objeto recuperado do repositório (por qualquer um dos componentes) pertence a uma coleção, nós fazemos uma chamada a um serviço web adicional para obter o nome desta coleção, com o objetivo de criar um *link* para a página de trabalhos desta coleção. Além disso, também inserimos *links* nos nomes de cada autor, fornecendo acesso direto à página de trabalhos dos autores. A chamada ao serviço web, os dados adicionais recuperados e os *links* penalizam o desempenho do nosso arcabouço.

Se esta carga de trabalho adicional não for realizada, nosso arcabouço é novamente mais rápido que o ODL, como pode ser observado nas Tabelas 5.7 e 5.8, nas entradas WS-ODL 2, que representam justamente os componentes de uma maneira mais similar aos componentes do ODL. O processo de criação da página de trabalhos da coleção em nosso arcabouço é mais rápido que no ODL em ambos os casos, pois neste caso não há muito trabalho extra a ser realizado. Por esta razão, a diferença de tempo entre as duas versões do componente não é significativa.

Três razões principais explicam o melhor desempenho de nosso arcabouço em relação ao ODL:

- O uso das tecnologias de *servlets* Java e serviços web, que possibilitam um desempenho superior que aquele oferecido pela tecnologia CGI – *Common Gateway Interface*⁸, a base do arcabouço ODL.
- A utilização efetiva da infra-estrutura do repositório, uma arquitetura específica

⁸<http://www.w3.org/CGI/>

para se trabalhar com armazenamento e acesso a objetos digitais, oferecendo assim meios para se alcançar melhor desempenho (por exemplo, *cache* de objetos).

- Um uso mais efetivo da tecnologia de bancos de dados relacionais, especialmente no que diz respeito ao componente de Busca.

Também pode ser observado a partir das Tabelas 5.7 e 5.8 que o componente de Navegação Multidimensional ainda não está maduro o suficiente e precisa ser melhor trabalhado, uma vez que seu tempo de resposta é muito maior que aquele oferecido pelo restante dos componentes do nosso arcabouço, ou mesmo pelos componentes do ODL. Uma possível razão para este fato é a falta de índices secundários (índices não baseados em chave primária) nas tabelas do banco de dados do componente. Este problema será futuramente investigado para melhorar o desempenho deste componente.

Além da comparação com o arcabouço ODL, também comparamos nosso arcabouço com a BDBComp, para comparar uma biblioteca digital construída especificamente voltada para um domínio de aplicação com uma biblioteca digital construída a partir de componentes genéricos. Foram comparados os componentes de Exibição de Trabalhos Recentes e de Navegação (por autor e coleção) com as respectivas páginas da BDBComp. Com relação aos trabalhos recentes, foram requisitados apenas os 10 primeiros trabalhos, em lugar dos 100 do experimento anterior. Para o componente de Navegação, não houve alteração na configuração do experimento. Neste experimento não foram utilizadas as versões modificadas dos componentes do nosso arcabouço (WS-ODL 2 no experimento anterior) pois a BDBComp também gera *links* para as páginas de trabalhos dos autores e autores da coleção. Os resultados são apresentados na Tabela 5.9.

Componente	Tempo Página (ms)		Tamanho (<i>bytes</i>)	Taxa (<i>bytes/ms</i>)
	Média	Desvio Padrão		
BDBComp Nav. Autor	49,35	7,05	12902	261,44
WS-ODL Nav. Autor	1598,11	33,67	27806	17,40
BDBComp Nav. Coleção	112,61	6,17	40779	362,13
WS-ODL Nav. Coleção	676,97	13,70	47786	70,59
BDBComp Trab. Recentes	44,56	3,01	6411	143,87
WS-ODL Trab. Recentes	457,82	24,20	6367	13,91

Tabela 5.9: Comparação com a BDBComp

Na tabela, a coluna *Tamanho* representa os tamanho em *bytes* da página web construída por cada componente, no caso de nosso arcabouço, ou serviço, no caso da BDBComp.

Já a coluna *Taxa* representa a velocidade de construção das páginas pelos componentes ou serviços. Esta medida foi introduzida para compatibilizar o tempo gasto pelos componentes em relação ao tamanho das páginas, pois como se pode observar, o tamanho das páginas não são semelhantes para todos os componentes.

A observação dos tempos de resposta e das taxas de construção das páginas indicam que a BDBComp possui um desempenho muito superior aos componentes do arcabouço. Duas razões principais explicam este fato: a primeira é que a BDBComp é uma biblioteca digital construída do zero, com uma arquitetura bastante integrada e um banco de dados especificamente desenhado para a aplicação. A biblioteca foi construída para ser o mais eficiente possível. Já os componentes do nosso arcabouço estão voltados para aplicações mais genéricas e são baseados sobre várias camadas de software (serviços web, Fedora, etc.), o que causa um *overhead* no desempenho.

Além disso, os componentes do nosso arcabouço sempre recuperam um objeto digital completo do repositório do Fedora, mesmo que nem todos os dados do objeto sejam exibidos. Por exemplo, nas telas mostradas no Capítulo 4, em nenhum momento mostra-se o campo *dc:description* do *Dublin Core*. No entanto, vários dos objetos mostrados naquelas telas possuem tal campo, que foi recuperado do repositório. O fato de sempre se recuperar estes dados, mesmo que eles não sejam utilizados também penaliza o desempenho, algo que não ocorre na BDBComp, pois nesta biblioteca digital o dado certo é recuperado no momento certo (por exemplo, o campo de descrição de um trabalho só é recuperado quando se acessa a página do trabalho).

Capítulo 6

Conclusões

Bibliotecas digitais são complexos sistemas de software voltados para atender as necessidades de informação de comunidades de usuários específicas. Uma das formas de se lidar com a complexidade desses sistemas é utilizar uma abordagem baseada em componentes para a sua construção. Neste trabalho, apresentamos um arcabouço para construção de bibliotecas digitais segundo esta abordagem. O arcabouço possui a vantagem de ser completamente baseado em tecnologias de código aberto, o que permite sua disponibilização sem ônus para os usuários e/ou comunidades interessados em utilizá-lo, que podem assim contribuir para o seu desenvolvimento, melhorando deficiências e agregando novos serviços.

Através da construção de um protótipo de biblioteca digital, mostramos que o arcabouço é viável, provendo uma forma de se construir bibliotecas digitais com o serviços básicos geralmente encontrados nesses sistemas (por exemplo, busca, navegação e exibição de trabalhos recentes) sem a necessidade de se escrever código, exceto para a criação das folhas de estilo XML da camada cliente, que são construídas utilizando-se XSL – *Extensible Stylesheet Language*¹. No entanto, para este caso específico, podem ser utilizados editores gráficos (Progress Software Corporation, 2006) com interface *WYSIWYG* – *What you see is what you get*. Além disso, folhas de estilo básicas, que produzem as interfaces de usuários exibidas no Capítulo 4, serão disponibilizadas juntamente com os demais componentes. No contexto do arcabouço, destaca-se o componente de Busca, que é capaz de prover vários tipos de consulta – *booleana* (e/ou) ou por relevância – e de combinar dados estruturados e dados textuais, utilizando-se apenas da tecnologia padrão de bancos de dados relacionais para simular o funcionamento

¹<http://www.w3.org/Style/XSL/>

de uma máquina de busca.

Os experimentos realizados mostram que o arcabouço possui um desempenho superior ao apresentado pelo arcabouço ODL, atingindo assim um dos objetivos propostos, pois tanto no processo de carga de dados quanto no desempenho dos componentes em si (quanto comparados de forma semelhante), nosso arcabouço responde em menor tempo que o ODL.

Entretanto, o arcabouço apresenta alguns problemas. Um deles é a falta de uma ferramenta para auxiliar o usuário durante o processo de instalação. Os experimentos de instalação realizados demonstram que os usuários cometem muitos erros e gastam muito tempo instalando e configurando o arcabouço via linha de comando. Desta forma, uma ferramenta gráfica que guie o usuário neste processo é uma característica necessária e de grande importância, pois tornará a instalação mais fácil e amigável.

Um outro problema do arcabouço está relacionado com o *triplestore* Kowari utilizado pelo Fedora. Conforme discutido na Seção 5.2, o Kowari apresenta um *memory leak* na versão utilizada pelo Fedora 2.0, que leva à perda de desempenho, notadamente quando se trabalha com grandes volumes de dados. Além disso, durante o processo de inserção de objetos digitais no repositório, muitas triplas RDF são geradas e armazenadas no *triplestore*. Com coleções muito extensas, como é o caso da coleção de dados do CiteSeer, o Kowari necessita de mais espaço de endereçamento do que pode prover uma máquina virtual Java rodando em um computador de arquitetura de 32 *bits*. As soluções para estes problemas incluem atualizar o Fedora para a versão 2.1, uma vez que esta versão possui uma atualização do Kowari para evitar o problema de *memory leak*, e utilizar um computador de arquitetura de 64 *bits* para executar o servidor do Fedora, alcançando assim um maior espaço de endereçamento.

Outro ponto de merecer ser observado é o fato que os componentes do arcabouço perderam em desempenho para seus equivalentes da BDBComp. De certa forma, este já era um resultado esperado, uma vez que o arcabouço é de uso genérico, focando-se mais na extensibilidade, interoperabilidade e reuso, ao passo que a BDBComp é um sistema com propósitos específicos, com sua arquitetura voltada atingi-los com o melhor desempenho.

Levando-se em consideração as questões discutidas acima, é importante, no momento da definição do sistema de biblioteca digital a ser construído, realizar-se um levantamento cuidadoso de quais são os objetivos do sistema e qual será o volume de dados tratados por ele. Se os objetivos são desempenho máximo e se volumes extensos de

dados vierem a ser incluídos no sistema em uma única vez, nosso arcabouço em sua versão atual pode não ser uma boa solução.

Um outro item que precisa ser melhor avaliado no arcabouço é a replicação de metadados do repositório do Fedora nos bancos de dados criados pelos componentes de Administração. Esta replicação traz o benefício de um melhor desempenho do arcabouço, mas no entanto causa problemas no que diz respeito à atualização dos dados. Quando acontece uma atualização em algum objeto digital no repositório, esta não é automaticamente refletida nos bancos de dados. É preciso desenvolver uma abordagem para que os componentes de administração possam atualizar os bancos de dados de modo a refletir as atualizações no repositório, preferencialmente de uma forma automática e transparente para o usuário, a partir de uma melhor integração destes componentes com o repositório.

No futuro, planejamos incorporar uma ferramenta de instalação ao arcabouço. Tal ferramenta deverá ser semelhante aos assistentes (*wizards*) existentes em vários sistemas disponíveis no mercado. Atualmente, existe um trabalho em andamento no Laboratório de Bancos de Dados do DCC/UFMG que trata do desenvolvimento desta ferramenta (Santos, 2005). No que diz respeito à melhoria do desempenho, podem ser investigados e corrigidos os problemas relativos ao componente de Navegação Multidimensional e a alguns dos componentes de Administração. Também pretendemos atualizar o Fedora para a versão 2.1 e testar o arcabouço com a coleções de dados disponíveis (BDBComp, LDB e CiteSeer) em um computador de arquitetura de 64 *bits*. A experimentação do arcabouço com objetos digitais mais complexos também deve ser considerada no futuro, pois podem ser necessárias algumas adaptações nos componentes criados. Uma investigação de outras arquiteturas (como o DSpace (Tansley et al., 2003), por exemplo) para a infra-estrutura do arcabouço também é algo a ser levado em consideração. Além destas atividades, pretendemos disponibilizar os componentes para utilização da comunidade e utilizá-los para aperfeiçoar serviços ou mesmo substituir a versão da BDBComp atualmente em produção, notadamente no que diz respeito ao serviço de busca, que não é muito flexível, uma vez que permite apenas casamentos exatos de padrões.

Referências Bibliográficas

- Apps, A. (2004). zetoc SOAP: A Web Services Interface for a Digital Library Resource. In *Proceedings of the 8th European Conference on Research and Advanced Technology for Digital Libraries*, pp. 198–208, Bath, UK.
- Bainbridge, D. e Witten, I. H. (2005). Practical Digital Library Interoperability Standards. In *Proceedings of the 5th ACM/IEEE-CS Joint Conference on Digital Libraries*, Tutoriais, p. 426. ACM Press.
- Baldacci, M. B. (1993). La Biblioteca del 2000: dal Punto di Vista delle Biblioteche. *Bolletino AIB*, 33(4):423–436.
- Bar-Ilan, J. (1999). Search Engine Results over Time – A Case Study on Search Engine Stability. *International Journal of Scientometrics, Informetrics and Bibliometrics*, 2/3(1). On-line. Disponível em: <http://cybermetrics.cindoc.csic.es/cybermetrics/articles/v2i1p1.html>.
- Bax, M. P. (1997). Agentes de Interface para Bibliotecas Digitais: a Arquitetura SABiO. In *VI Seminário sobre Automação em Bibliotecas e Centros de Documentação*, Águas de Lindóia, SP, Brasil. On-line. Disponível em: <http://www.bax.com.br/research/publications/viseab.pdf>.
- Borgman, C. L. (1999). What are Digital Libraries? Competing Visions. *Information Processing and Management*, 35(3):227–243.
- Buchanan, G.; Bainbridge, D.; Don, K. J. e Witten, I. H. (2005). A new framework for building digital library collections. In *Proceedings of the 5th ACM/IEEE-CS Joint Conference on Digital Libraries*, pp. 23–31, Denver, CO, USA. ACM Press.
- Congia, S.; Gaylord, M.; Merchant, B. e Suleman, H. (2004). Applying SOAP to OAI-PMH. In *Proceedings of the 8th European Conference on Research and Advanced Technology for Digital Libraries*, pp. 411–420, Bath, UK.

- Digital Library Federation (2006). A Working Definition of Digital Library [1998]. On-line. Disponível em: <http://www.diglib.org/about/dldefinition.htm>.
- Dublin Core (2006). Dublin Core Metadata Initiative (DCMI). On-line. Disponível em: <http://dublincore.org/>.
- Fisher, J. e Lee, D. (2006). The OpenArXiv. On-line. Disponível em: <http://openarxiv.ist.psu.edu/>.
- Fox, E. A. e Marchionini, G. (1998). Toward a Worldwide Digital Library. *Communications of the ACM*, 41(4):29–32.
- Gonçalves, M. A. (2004). *Streams, Structures, Spaces, Scenarios, and Societies (5S): A Formal Digital Library Framework and Its Applications*. PhD thesis, Virginia Polytechnic Institute and State University, Blacksburg, VA, USA.
- Grossman, D. A.; Frieder, O.; Holmes, D. O. e Roberts, D. C. (1997). Integrating Structured Data and Text: A Relational Approach. *Journal of the American Society of Information Science*, 48(2):122–132.
- Laender, A. H. F.; Gonçalves, M. A. e Roberto, P. A. (2004). BDBComp: Building a Digital Library for the Brazilian Computer Science Community. In *Proceedings of the 4th ACM/IEEE-CS Joint Conference on Digital Libraries*, pp. 23–24, Tucson, AZ, USA. ACM Press.
- Lagoze, C. e Davis, J. R. (1995). Dienst: an Architecture for Distributed Document Libraries. *Communications of the ACM*, 38(4):47.
- Lagoze, C. e de Sompel, H. V. (2001). The Open Archives Initiative: Building a Low-Barrier Interoperability Framework. In *Proceedings of the 1st ACM/IEEE-CS Joint Conference on Digital Libraries*, pp. 54–62, Roanoke, VA, USA. ACM Press.
- Lagoze, C.; Payette, S.; Shin, E. e Wilper, C. (2005). Fedora: An Architecture for Complex Objects and their Relationships. *Journal of Digital Libraries*. Special Issue on Complex Objects, (a ser publicado).
- Locuratolo, E. e Meghini, C. (2003). Modelling a digital library architecture. Technical Report 2003-TR-06, Istituto di Scienza e Tecnologie della Informazione, Pisa, Itália.
- Microsoft (2006). Full-Text Search (Pesquisa de Texto Completo). On-line. Disponível em: <http://www.microsoft.com/brasil/sql/overview/recursos/fulltext.msp>.

- Open Archives Initiative (2006). Open Archives Initiative – Protocol for Metadata Harvesting – v.2.0. On-line. Disponível em: <http://www.openarchives.org/OAI/openarchivesprotocol.html>.
- OpenDLib (2006). Open DLib Official Site. On-line. Disponível em: <http://opendlib.iei.pi.cnr.it/home.html>.
- Progress Software Corporation (2006). Stylus Studio. On-line. Disponível em: <http://www.stylusstudio.com/>.
- Raghavan, A.; Vemuri, N. S.; Shen, R.; Gonçalves, M. A.; Fan, W. e Fox, E. A. (2005). Incremental, Semi-automatic, Mapping-Based Integration of Heterogeneous Collections into Archaeological Digital Libraries: Megiddo Case Study. In *Proceedings of the 9th European Conference on Research and Advanced Technology for Digital Libraries*, pp. 139–150, Vienna, Austria.
- Salton, G. e McGill, M. J. (1983). *Introduction to Modern Information Retrieval*. McGraw-Hill, New York.
- Santos, R. L. T. (2005). Um Assistente para Configuração de Bibliotecas Digitais Componentizadas. In *I Workshop em Bibliotecas Digitais*, pp. 11–20, Uberlândia, MG, Brasil.
- Software Engineering Institute (2006). Component-Based Software Development / COTS Integration. On-line. Disponível em: http://www.sei.cmu.edu/str/descriptions/cbsd_body.html.
- Suleman, H. (2002). *Open Digital Libraries*. PhD thesis, Virginia Polytechnic Institute and State University, Blacksburg, VA, USA.
- Tansley, R.; Bass, M.; Stuve, D.; Branschofsky, M.; Chudnov, D.; McClellan, G. e Smith, M. (2003). The DSpace Institutional Digital Repository System: Current Functionality. In *Proceedings of the 3rd ACM/IEEE-CS Joint Conference on Digital Libraries*, pp. 87–97. IEEE Computer Society.
- Tucana Technologies (2006). Kowari. On-line. Disponível em: <http://www.kowari.org/>.
- UDDI (2006). Uddi.org. On-line. Disponível em: <http://uddi.org/>.
- World Wide Web Consortium (2006a). Resource Description Framework (RDF) / W3C Semantic Web Activity. On-line. Disponível em: <http://www.w3.org/RDF/>.
- World Wide Web Consortium (2006b). SOAP Specifications. On-line. Disponível em: <http://www.w3.org/TR/soap>.

World Wide Web Consortium (2006c). Web Service Description Language (WSDL). On-line. Disponível em: <http://www.w3.org/TR/wsdl>.

World Wide Web Consortium (2006d). Web Services. On-line. Disponível em: <http://www.w3.org/2002/ws/>.

XACML (2006). OASIS eXtensible Access Control Markup Language (XACML) TC. On-line. Disponível em: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml.

Apêndice A

Roteiro de Instalação do Arcabouço WS-ODL

Pré-requisitos

=====

- * Sistema de Gerenciamento de Bancos de Dados MySQL (versão 3.23.x ou superior).
- * Sun Java Platform, Standard Edition 5.0 (JDK ou JRE + tools.jar, que se localiza no diretório \textit{lib} do JDK).
- * Sistema Operacional Linux (Fedora Core, SuSE, etc.).

Observações

=====

- * Recomenda-se realizar a configuração das variáveis de ambiente no arquivo de configuração do shell (e.g., .bashrc), para que elas sejam permanentes. O arquivo .bashrc é um arquivo oculto que se encontra no diretório raiz (home) do usuário, podendo ser acessado pelos comandos "nano .bashrc" ou "vi .bashrc", por exemplo.
- * Nos passos a seguir <download_dir> se refere ao diretório para o qual os softwares necessários são baixados e <install_dir> se refere ao diretório de instalação.

* O roteiro considera que os componentes do arcabouço WS-ODL se encontram em <download_dir>.

A) Instalação do Fedora 2.0

=====

- 1) Baixe para <download_dir> o Fedora Server e o Fedora Client (<http://www.fedora.info/download/2.0/>)
- 2) cd <install_dir>
- 3) tar -zxvf <download_dir>/fedora-2.0-server.tar.gz
- 4) tar -zxvf <download_dir>/fedora-2.0-client.tar.gz
- 5) chmod 755 fedora-2.0/server/bin/*
- 6) chmod 755 fedora-2.0/client/bin/*
- 7) cd
- 8) nano .bashrc ou vi .bashrc
- 9) Configure a variável de ambiente JAVA_HOME para o diretório de instalação do Java (ex.: export JAVA_HOME=/opt/jdk1.5.0_05)
- 10) Configure a variável de ambiente FEDORA_HOME para o diretório de instalação do Fedora (ex.: export FEDORA_HOME=/var/tmp/fedora-2.0)
- 11) Adicione os diretórios bin do Java e do Fedora (Server e Client) ao path. O comando "export" deve ser digitado em uma única linha. (ex.: export PATH=\$PATH:\$JAVA_HOME:\$JAVA_HOME/bin:\$FEDORA_HOME/server/bin:\$FEDORA_HOME/client/bin:..).
- 12) Reinicialize o shell (saia e entre novamente), para que as configurações das variáveis de ambiente sejam reconhecidas
- 13) Execute mysql-config com os seguintes parâmetros:
 - installDir -> diretório onde esta o executavel do mysql
 - dbaUser -> usuário para acesso ao banco de dados
 - dbaPass -> senha do usuário dbaUser
 - fedoraDBUser -> usuário do Fedora
 - fedoraDBPass -> senha do usuário fedoraDBUser
 - dbName -> nome do banco de dados do Fedora
 - [mysql41_flag] -> deve ser informado apenas no caso da utilização do MySQL 4.1.x ou superior. Do contrário deve ser omitido. Qualquer valor pode ser informado para este parâmetro, pois apenas sua presença já indica o uso da versão 4.1.x ou superior do MySQL. (ex.: mysql-config /usr root "" fedoraAdmin fedoraAdmin fedora20 a). Remova o arquivo .sql ao final da operacao, respondendo

"y" à pergunta.

14) Configure o Fedora, abrindo o arquivo

<install_dir>/fedora-2.0/server/config/fedora.fcfg no seu editor de texto preferido

- informe o nome do repositório (param repositoryName) (ex.: "WS-ODL Fedora Repository")
- informe o(s) e-mail(s) do(s) administrador(es) (param adminEmailList) (ex.: pabloa@dcc.ufmg.br)
- informe a senha do administrador (param adminPassword) (ex.: fedoraAdmin)
- informe o diretório para armazenamento dos objetos (param object_store_base) (ex.: /var/tmp/fedora-2.0/objects)
- informe o diretório temporário (param temp_store_base) (ex.: /var/tmp/fedora-2.0/temp)
- informe o diretório dos datastreams (param datastream_store_base) (ex.: /var/tmp/fedora-2.0/datastreams)
- informe as portas a serem utilizadas pelo Fedora (params fedoraServerPort, fedoraShutdownPort, fedoraRedirectPort). Pode-se utilizar os valores já configurados (8080, 8005 e 8443), a não ser que haja outra aplicação utilizando estas portas
- informe o nome da máquina (param fedoraServerHost) (ex.: localhost)
- informe o namespace dos pids (param pidNamespace em DOManager) (ex.: wsodl)
- informe o IP ou faixa de IPs autorizadas a acessar o Fedora (param allowHosts em Management) (ex. 127.0.0.1,100.0.0.0-180.0.0.0) -> digitar como está informado, ou seja, sem nenhum espaço
- informe o nome do repositório OAI (param repositoryName em OAIProvider) (ex.: WS-ODL Fedora Repository)
- informe o domínio do repositório OAI (param repositoryDomainName em OAIProvider) (ex.: lbd.dcc.ufmg.br)
- informe o(s) e-mail(s) do(s) administrador(es) do repositório OAI (param adminEmails em OAIProvider) (ex.: pabloa@dcc.ufmg.br)
- comente ou apague a linha do param friends
- informe o usuário do banco de dados (param dbUsername no datastore localMySQLPool) (deve ser o mesmo informado no passo 11, ex.: fedoraAdmin)
- informe a senha do usuário do banco de dados (param dbPassword no datastore localMySQLPool) (deve ser a mesma informada no passo 11, ex.: fedoraAdmin)
- informe a URL do banco de dados (param jdbcURL no datastore localMySQLPool) (o banco deve ser o mesmo informado no passo 11, ex.: jdbc:mysql://localhost/fedora20?useUnicode=true&characterEncoding=UTF-8&

autoReconnect=true)

- informe o driver do banco de dados (param jdbcDriverClass no datastore localMySQLPool). Informar o valor com.mysql.jdbc.Driver

- informe o diretório para utilização pelo Resource Index (param path no datastore localKowariTriplestore) (ex.: /var/tmp/fedora-2.0/resourceIndex)

15) Abra o arquivo <install_dir>/fedora-2.0/server/bin/fedora.sh no seu editor de texto

- substitua o parâmetro 15 por 60 nas linhas 82,83,88 e 89; isto dá mais tempo para o servidor Tomcat inicializar

16) cd

17) Execute fedora-start para inicializar o Fedora

18) Execute fedora-admin para entrar no administrador do Fedora. Isto serve para testar se a instalação foi bem sucedida

B) Instalação do Apache Axis 1.1

=====

1) Baixe para <download_dir> o axis 1.1

(http://archive.apache.org/dist/ws/axis/1_1/axis-1_1.zip)

2) cd <install_dir>

3) unzip <download_dir>/axis-1_1.zip

4) cd axis-1_1/webapps

5) cp -r axis/ \$FEDORA_HOME/server/jakarta-tomcat-5.0.28/webapps/

6) cd ../..

7) rm -Rf axis-1_1/

8) Baixe para <download_dir> o JavaBeans(TM) Activation Framework 1.0.2

(<http://java.sun.com/products/javabeans/glasgow/jaf.html>)

9) unzip <download_dir>/jaf-1_0_2-upd2.zip

10) cd jaf-1.0.2

11) cp activation.jar \$FEDORA_HOME/server/jakarta-tomcat-5.0.28/common/lib

12) cd ..

13) rm -Rf jaf-1.0.2

14) Coloque o arquivo tools.jar no diretorio lib do tomcat

(cp \$JAVA_HOME/lib/tools.jar \$FEDORA_HOME/server/jakarta-tomcat-5.0.28/common/lib)

15) cd

16) Execute fedora-stop e em seguida fedora-start para restartar o servidor

17) Acesse <http://localhost:8080/axis/happyaxis.jsp> e verifique se todos os componentes obrigatórios do axis foram encontrados

C) Instalação dos Serviços Web no Axis

=====

- 1) `cd <download_dir>/wsdl_ws`
- 2) `cp *.class ws-config.xml $FEDORA_HOME/server/jakarta-tomcat-5.0.28/webapps/axis/WEB-INF/classes`
- 3) Utilize o editor de texto para abrir o arquivo de configuração (`$FEDORA_HOME/server/jakarta-tomcat-5.0.28/webapps/axis/WEB-INF/classes/ws-config.xml`) e modifique-o para refletir os valores do arquivo de configuração do Fedora. Não é necessário modificar este arquivo se os valores padrão foram utilizados para configurar o Fedora.
- 4) `cp <download_dir>/administracao.jar $FEDORA_HOME/server/jakarta-tomcat-5.0.28/common/lib`
- 5) `cp $FEDORA_HOME/client/client.jar $FEDORA_HOME/server/jakarta-tomcat-5.0.28/webapps/axis/WEB-INF/lib`
- 6) Baixe para `<download_dir>` o arquivo `xalan-j_2_7_0-bin-2jars.tar.gz` (<http://www.apache.org/dyn/closer.cgi/xml/xalan-j>)
- 7) `cd <install_dir>`
- 8) `tar -zxvf <download_dir>/xalan-j_2_7_0-bin-2jars.tar.gz`
- 9) `cd xalan-j_2_7_0`
- 10) `cp xalan.jar serializer.jar $FEDORA_HOME/server/jakarta-tomcat-5.0.28/common/lib`
- 11) `cd ..`
- 12) `rm -Rf xalan-j_2_7_0`
- 13) `cd`
- 14) `nano .bashrc` ou `vi .bashrc`, e posicione o cursor no fim do arquivo
- 15) Configure as seguintes variáveis de ambiente
 - `export TOMCAT_HOME=$FEDORA_HOME/server/jakarta-tomcat-5.0.28`
 - `export TOMCAT_LIB=$TOMCAT_HOME/common/lib`
 - `export AXISCLASSPATH=$TOMCAT_LIB/axis.jar:$TOMCAT_LIB/commons-discovery.jar:$TOMCAT_LIB/commons-logging.jar:$TOMCAT_LIB/jaxrpc.jar:$TOMCAT_LIB/saaj.jar:$TOMCAT_LIB/log4j-1.2.8.jar:$TOMCAT_LIB/xml-apis.jar:$TOMCAT_LIB/xercesImpl.jar:$TOMCAT_LIB/wsdl4j.jar`
 - `export CLASSPATH=$TOMCAT_LIB/servlet-api.jar:$AXISCLASSPATH:$TOMCAT_LIB/administracao.jar:$TOMCAT_LIB/xalan.jar:$TOMCAT_LIB/serializer.jar:$TOMCAT_LIB/mysql-connector-java-3.0.15-ga-bin.jar:.`
- 16) Reinicialize o shell (saia e entre novamente), para que as configurações das

variáveis de ambiente sejam reconhecidas

17) Execute `fedora-stop` e em seguida `fedora-start` para reiniciar o servidor

18) Realize o deploy dos serviços web do arcabouço WS-ODL

- `cd <download_dir>/wsodl_ws`

- `$JAVA_HOME/bin/java org.apache.axis.client.AdminClient deploy.wsdd`

19) Acesse `http://localhost:8080/axis/servlet/AxisServlet` e verifique se os serviços web são listados (Recent, Collection etc.)

D) Instalação dos Servlets

=====

1) `cd $FEDORA_HOME/server/jakarta-tomcat-5.0.28/webapps/`

2) `mkdir wsodl`

3) `cp -r <download_dir>/wsodl_servlets/* wsodl/`

4) `cp -r <download_dir>/wsodl/ $FEDORA_HOME`

5) Realize as configurações necessárias no arquivo `config.xml` dentro de `$FEDORA_HOME/wsodl`, utilizando o editor de textos. Não é necessário modificar este arquivo se os valores padrão foram utilizados para configurar o Fedora, exceto o parâmetro `"absolutefilepath"`, que deve ser modificado e refletir uma estrutura de diretórios válida, ou seja, os diretórios devem ser criados existir ou devem ser criados manualmente antes de sua utilização. Sugere-se utilizar `$FEDORA_HOME/dados/objetos`, onde `$FEDORA_HOME` deve ser substituído pelo caminho real do Fedora (e. g., `/var/tmp/fedora-2.0`)

6) Crie a estrutura de diretórios informada no parâmetro `"absolutefilepath"`

- `cd $FEDORA_HOME`

- `mkdir dados`

- `cd dados`

- `mkdir objetos`

7) `cd`

8) Execute `fedora-stop` e em seguida `fedora-start` para reinicializar o servidor

E) Inserção dos Dados

=====

1) `cd <download_dir>`

2) `cp wdl2005.xml dsearch_create.sql wsodlunion.sql $FEDORA_HOME/dados`

3) `cd $FEDORA_HOME/dados`

4) Realize a transformação dos dados de entrada

(\$JAVA_HOME/bin/java Transformador wdl2005.xml \$FEDORA_HOME/wsodl/xsl/fedora.xsl objetos_fedora.xml)

5) Particione arquivo de entrada (criará um arquivo para cada objeto digital e armazenará os resultados no caminho dado pelo parametro "absolutefilepath")

(\$JAVA_HOME/bin/java FileGenerator objetos_fedora.xml)

6) Insira os objetos no repositório do Fedora

(\$JAVA_HOME/bin/java -cp \$CLASSPATH:\$FEDORA_HOME/client/client.jar Ingester <numFiles>) (numFiles = 8 para wdl2005.xml)

7) Acesse <http://localhost:8080/wsodl/TrabalhosRecentese> verifique se o serviço está funcionando

8) Crie os bancos de dados dos demais serviços (wsodlunion, search e browse)

- acesse o cliente mysql como root ou um usuário capaz de criar databases e dar permissões (ex.: mysql -u root)

- execute o comando create database wsodlunion; --> ponto e virgula faz parte do comando

- execute o comando create database dsearch;

- execute o comando create database dbrowse;

9) Dê permissão ao usuário do banco de dados do Fedora nestes novos bancos

- ainda dentro do mysql, execute os seguintes comandos

- GRANT ALL PRIVILEGES ON wsodlunion.* TO fedoraAdmin@localhost IDENTIFIED BY 'fedoraAdmin';

- GRANT ALL PRIVILEGES ON wsodlunion.* TO fedoraAdmin@'%' IDENTIFIED BY 'fedoraAdmin';

- GRANT ALL PRIVILEGES ON dsearch.* TO fedoraAdmin@localhost IDENTIFIED BY 'fedoraAdmin';

- GRANT ALL PRIVILEGES ON dsearch.* TO fedoraAdmin@'%' IDENTIFIED BY 'fedoraAdmin';

- GRANT ALL PRIVILEGES ON dbrowse.* TO fedoraAdmin@localhost IDENTIFIED BY 'fedoraAdmin';

- GRANT ALL PRIVILEGES ON dbrowse.* TO fedoraAdmin@'%' IDENTIFIED BY 'fedoraAdmin';

O usuário e senha informados devem ser os mesmos utilizados em todos os passos anteriores (fedoraAdmin, neste roteiro)

10) Execute scripts de criação das tabelas dos bancos de dados

- ainda dentro do mysql, execute os seguintes comandos, nesta ordem:

- use wsodlunion;

- source wsodlunion.sql;

- use dsearch;

- source dsearch_create.sql;
 - exit
- 11) Insira os dados nos bancos de dados criados
- Execute os seguintes programas, nesta ordem:
 - \$JAVA_HOME/bin/java SimulateUnionDatabase
 - \$JAVA_HOME/bin/java databaseMaintenanceModule
 - \$JAVA_HOME/bin/java SearchDatabaseMaintenance
- 12) Acesse <http://localhost:8080/wsodl/BrowseInterface> e verifique se o serviço está funcionando