

RODRYGO LUIS TEODORO SANTOS

**WHIZKEY: UM AMBIENTE PARA INSTALAÇÃO DE  
BIBLIOTECAS DIGITAIS**

Belo Horizonte  
19 de dezembro de 2007

RODRYGO LUIS TEODORO SANTOS  
ORIENTADOR: MARCOS ANDRÉ GONÇALVES

**WHIZKEY: UM AMBIENTE PARA INSTALAÇÃO 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.

Belo Horizonte  
19 de dezembro de 2007



UNIVERSIDADE FEDERAL DE MINAS GERAIS

FOLHA DE APROVAÇÃO

WhizKEY: Um Ambiente para Instalação de Bibliotecas Digitais

RODRYGO LUIS TEODORO SANTOS

Dissertação defendida e aprovada pela banca examinadora constituída por:

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

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

Ph. D. ALBERTO HENRIQUE FRADE LAENDER  
Universidade Federal de Minas Gerais

Ph. D. RAQUEL OLIVEIRA PRATES  
Universidade Federal de Minas Gerais

Belo Horizonte, 19 de dezembro de 2007

*A meus pais.*

# Agradecimentos

Agradeço a meus pais, Luiz e Vanda, pela generosa oferta de educação como herança inalienável e, ao lado de minhas irmãs, pelo apoio incondicional desde sempre. Aos meus primos e primas de BH, pelo companheirismo e pela cerveja incondicionais!

Aos amigos do LBD, antigos e novos, pelo fantástico ambiente de trabalho. Aos amigos do DCC, companheiros de café e tequila, por caminharmos juntos. Aos amigos de Divi, poucos, bons e – espero – eternos por, ao lado de minha família, lembrarem-me do conceito de “casa”.

Ao meu orientador, Prof. Marcos, pela visão crítica e pelas cobranças, fundamentais na busca pelo sucesso. Ao Prof. Alberto, pelo interesse sincero e pelo exemplo como líder.

À Elisa e aos diversos voluntários que contribuíram para a avaliação deste trabalho, aos membros da banca examinadora e aos diversos revisores anônimos que o enriqueceram com valiosos comentários e ao CNPq, por fomentar seu desenvolvimento e divulgação.

A todos que, de alguma forma, fizeram a diferença ao longo desse ótimo tempo.

# Resumo

Nesta dissertação, apresentamos WhizKEY – *Wizard-based block Ensemble Yielder*, um ambiente para assistência à instalação de bibliotecas digitais a partir de arcabouços de software. WhizKEY foi desenvolvido como um ambiente integrado para a construção e personalização de bibliotecas digitais com base em arcabouços de software potencialmente distintos.

Do ponto de vista do usuário, WhizKEY permite que projetistas de bibliotecas digitais se concentrem nos requisitos do sistema sendo instanciado em vez de se preocuparem com a complexidade da tarefa de instalação subjacente. Para tanto, os projetistas são guiados através da tarefa de instalação e cada parâmetro por eles configurado é verificado com base em um conjunto de restrições definido a fim de garantir a correção da instalação produzida.

Do ponto de vista do desenvolvedor de um arcabouço, WhizKEY implementa a tarefa de instalação como um fluxo, considerando cada passo nesse fluxo como responsável pela configuração de instâncias de um aspecto de uma biblioteca digital (e.g., a própria biblioteca, suas coleções ou catálogos de metadados e os serviços que ela pode oferecer). Esse fluxo é totalmente configurável para permitir a instalação de bibliotecas a partir de diferentes arcabouços.

Por meio de duas sessões de testes de usabilidade envolvendo potenciais usuários, a efetividade do ambiente WhizKEY no auxílio à tarefa de instalação foi atestada e uma lista de problemas foi levantada e utilizada no desenvolvimento da versão atual do ambiente.

# Abstract

In this dissertation, we present WhizKEY – Wizard-based blocK Ensemble Yielder, an environment to assist users in the installation of digital libraries from existing software frameworks. WhizKEY has been designed to provide users with an integrated environment for constructing and personalizing digital libraries from possibly several different software frameworks.

From a user’s viewpoint, WhizKEY aims at shielding digital library designers from the intricacies of the underlying installation task, allowing them to concentrate on the requirements of the system being instantiated rather than on the installation task itself. For such, designers are guided through the task, and every parameter configured by them is verified against a set of constraints in order to guarantee the correctness of the produced installation.

From a framework developer’s viewpoint, WhizKEY implements the installation task as a workflow, regarding each step in this workflow as responsible for the configuration of instances of major aspects within a digital library (e.g., the library itself, its collections or metadata catalogs, and the services it may provide). This workflow is fully configurable to enable the installation of digital libraries based on different software frameworks.

Through two separate usability test sessions comprising potential WhizKEY users, the effectiveness of the environment with respect to easing the installation task was attested and a list of problems was devised and employed in the development of its current version.

# Sumário

<b>1</b>	<b>Introdução</b>	<b>1</b>
1.1	Contexto e Motivação . . . . .	1
1.2	Contribuições da Dissertação . . . . .	3
1.3	Organização da Dissertação . . . . .	3
<b>2</b>	<b>Trabalhos Relacionados</b>	<b>4</b>
2.1	Arcabouços de Software . . . . .	4
2.1.1	Arcabouço ODL . . . . .	4
2.1.2	Arcabouço WS-ODL . . . . .	6
2.2	Ferramentas para Construção de Bibliotecas Digitais . . . . .	8
2.2.1	BLOX . . . . .	8
2.2.2	5SGraph . . . . .	10
2.2.3	Soluções Específicas . . . . .	12
2.2.4	Comparação das Soluções . . . . .	12
<b>3</b>	<b>O Ambiente WhizKEY</b>	<b>13</b>
3.1	Visão Geral da Arquitetura . . . . .	14
3.2	Modelo: Representação de Entidades . . . . .	15
3.2.1	Especificação de Arcabouços . . . . .	16
3.2.2	Gerenciamento de Configurações . . . . .	19
3.3	Visualização: Geração de Interfaces Gráficas de Usuário . . . . .	23
3.3.1	Painel de Navegação . . . . .	24
3.3.2	Painéis de Configuração . . . . .	24
3.3.3	Diálogos de Configuração . . . . .	26
3.4	Controle: Gerenciamento do Fluxo de Instalação . . . . .	29
3.4.1	Passos de Configuração . . . . .	29
3.4.2	Fluxo de Instalação . . . . .	29
3.5	Persistência: Preparação e Configuração de Software . . . . .	31
3.5.1	Preparação de Software . . . . .	31
3.5.2	Configuração de Software . . . . .	32
3.6	Exemplo de Uso . . . . .	33



<b>4</b>	<b>Avaliação do Ambiente</b>	<b>41</b>
4.1	Projeto Experimental . . . . .	41
4.1.1	Preparação dos Testes . . . . .	41
4.1.2	Execução dos Testes . . . . .	43
4.2	Análise dos Resultados . . . . .	44
<b>5</b>	<b>Conclusões e Trabalhos Futuros</b>	<b>48</b>
5.1	Revisão do Trabalho . . . . .	48
5.2	Trabalhos Futuros . . . . .	49
	<b>Referências Bibliográficas</b>	<b>50</b>
<b>A</b>	<b>Documento Descritor do Arcabouço</b>	<b>53</b>
<b>B</b>	<b>Material para Testes com Usuários</b>	<b>75</b>
B.1	Perfil dos Usuários . . . . .	75
B.2	Contextualização . . . . .	76
B.3	Descrição das Tarefas . . . . .	77
B.3.1	Descrição da Tarefa $T_1$ . . . . .	77
B.3.2	Descrição da Tarefa $T_2$ . . . . .	78
B.3.3	Descrição da Tarefa $T_3$ . . . . .	79
B.4	Questionário de Avaliação . . . . .	80
B.5	Roteiro para Entrevista . . . . .	81
<b>C</b>	<b>Problemas Detectados pelos Testes</b>	<b>83</b>

# Lista de Figuras

2.1	Biblioteca digital segundo o arcabouço ODL . . . . .	5
2.2	Biblioteca digital segundo o arcabouço WS-ODL . . . . .	7
2.3	Instantâneo da parte cliente da ferramenta BLOX . . . . .	8
2.4	Visão geral da arquitetura da ferramenta BLOX . . . . .	9
2.5	Abordagem 5S para o desenvolvimento de bibliotecas digitais . . . . .	10
2.6	Instantâneo da ferramenta 5SGraph . . . . .	11
3.1	Processo de instalação assistido pelo ambiente WhizKEY . . . . .	13
3.2	Esquema de funcionamento do padrão MVC . . . . .	15
3.3	Visão geral da camada modelo . . . . .	15
3.4	Visão geral de um DDA . . . . .	16
3.5	Ciclo de vida de um bloco . . . . .	19
3.6	Organização visual da interface gráfica . . . . .	23
3.7	Elementos de interface para configuração de parâmetros . . . . .	27
3.8	Ajuda contextual na configuração de parâmetros . . . . .	28
3.9	Exemplo de instruções de preparação para um componente . . . . .	31
3.10	Exemplo de fontes associadas a um parâmetro . . . . .	32
3.11	Boas-vindas . . . . .	33
3.12	Seleção da tarefa de instalação . . . . .	33
3.13	Identificação de um sistema . . . . .	34
3.14	Configuração de uma biblioteca (1) . . . . .	34
3.15	Configuração de uma biblioteca (2) . . . . .	34
3.16	Configuração de uma biblioteca (3) . . . . .	35
3.17	Configuração de uma biblioteca (4) . . . . .	35
3.18	Configuração de catálogos (1) . . . . .	35
3.19	Configuração de catálogos (2) . . . . .	36
3.20	Configuração de catálogos (3) . . . . .	36
3.21	Configuração de catálogos (4) . . . . .	36
3.22	Configuração de catálogos (5) . . . . .	37
3.23	Configuração de catálogos (6) . . . . .	37
3.24	Configuração de catálogos (7) . . . . .	37
3.25	Configuração de serviços (1) . . . . .	38
3.26	Configuração de serviços (2) . . . . .	38

3.27	Configuração de serviços (3)	38
3.28	Configuração de serviços (4)	39
3.29	Resumo das configurações	39
3.30	Resultado da instalação	39
3.31	Execução do sistema (1)	40
3.32	Execução do sistema (2)	40
4.1	Evolução temporal da execução da tarefa	45
4.2	Evolução temporal da perícia	46

# Lista de Tabelas

2.1	Requisições definidas pelo protocolo OAI-PMH . . . . .	4
2.2	Componentes ODL . . . . .	5
2.3	Componentes WS-ODL . . . . .	6
2.4	WhizKEY versus ferramentas relacionadas . . . . .	12
3.1	Atributos de um componente . . . . .	17
3.2	Atributos de um parâmetro . . . . .	17
3.3	Regras para expansão de referências . . . . .	21
3.4	Regras para geração dinâmica de elementos de interface . . . . .	27
3.5	Mensagens de atualização do fluxo de instalação . . . . .	30
4.1	Tempo de execução e correção por tarefa . . . . .	44
4.2	Desempenho e perícia por tarefa . . . . .	46
C.1	Problemas detectados por avaliações de usabilidade . . . . .	83

# Capítulo 1

## Introdução

### 1.1 Contexto e Motivação

Ao longo das últimas décadas, a quantidade e a variedade de fontes de informação digital proliferaram. O avanço dos sistemas de computação e a contínua revolução das redes de comunicação resultaram em uma expansão notável da habilidade de produzir, processar e disseminar informação digital. Juntos, esses desenvolvimentos criaram novas formas para repositórios de conhecimento, bem como tornaram economicamente viáveis mecanismos para a distribuição de informação (Young, 1994). Ainda hoje, nossa capacidade de gerar e coletar informação digital continua a crescer mais rapidamente que a de organizá-la, gerenciá-la e, efetivamente, utilizá-la. Nesse cenário, as bibliotecas digitais surgiram com o objetivo principal de tornar a informação globalmente distribuída acessível para um público amplo e diverso.

Embora continue a evoluir à medida que novos resultados de pesquisa são obtidos e novas tecnologias são desenvolvidas, a definição do que seja uma biblioteca digital não é consensual e parece se estender somente à noção de coleções acessíveis de informação (Borgman, 1999). Além de dificultar uma convergência da área, essa falta de consenso tem influências negativas nos projetos de desenvolvimento de bibliotecas digitais. A maioria dos sistemas existentes que são classificados como bibliotecas digitais resultam de projetos de desenvolvimento ad hoc, caracterizados por ciclos dispendiosos de desenho, implementação e testes. Além disso, à medida que os sistemas desenvolvidos tornam-se mais complexos, sua extensão se torna mais difícil e, conseqüentemente, sua manutenibilidade é comprometida (Suleman e Fox, 2001). Em geral, esse modelo de desenvolvimento é pouco previsível, uma vez que o processo de desenvolvimento é constantemente modificado ao longo do projeto. A consistência de prazos, orçamentos, funcionalidade e qualidade dos produtos em relação ao planejamento inicial geralmente depende da capacidade de membros individuais das equipes de desenvolvimento e varia de acordo com suas habilidades, seu conhecimento e sua motivação (Paulk et al., 1993).

Outro problema decorrente desse modelo de desenvolvimento diz respeito à interoperabilidade dos sistemas desenvolvidos (Paepcke et al., 1998). Uma vez que duas bibliotecas não compartilham uma base de desenvolvimento comum, torná-las interoperáveis constitui-se uma tarefa complexa, agravada ainda mais quando um grande número de sistemas deve interoperar.

Em uma tentativa de reduzir esses efeitos, nos últimos anos, o desenvolvimento de bibliotecas digitais passou a incorporar práticas consolidadas da engenharia de software, tais como a utilização de arcabouços de software – arquiteturas semi-acabadas, reutilizadas no desenvolvimento de sistemas funcionais (Pree, 1995). No campo de bibliotecas digitais, em particular, muitos arcabouços foram desenvolvidos. Dentre os mais conhecidos, podemos citar: Greenstone<sup>1</sup>, EPrints<sup>2</sup>, DSpace<sup>3</sup> e Fedora<sup>4</sup>. Com isso, bibliotecas digitais antes implementadas do zero passaram a ser construídas por meio do reúso de arquiteturas, o que resultou em projetos de desenvolvimento mais eficientes, confiáveis e baratos.

Por outro lado, a instanciação de sistemas funcionais a partir de um arcabouço de software não é uma tarefa trivial, particularmente para usuários não-especialistas. Seguindo a terminologia adotada por Pree (1995), arcabouços de software são constituídos de duas partes: *frozen spots* e *hot spots*. De um lado, *frozen spots* definem a arquitetura global de um sistema de software, seus componentes básicos e os relacionamentos entre eles. Correspondem à porção do arcabouço pronta para ser utilizada, i.e., a parte que permanece inalterada em qualquer instanciação do arcabouço. De outro lado, *hot spots* representam as partes que precisam ser ajustadas de forma a se construírem sistemas especializados. Esses ajustes podem demandar diferentes procedimentos dependendo do arcabouço subjacente. Aqui, diferenciamos dois casos: instanciação por *implementação* e instanciação por *instalação*. A instanciação por implementação exige que os programadores adicionem seu próprio código para especializar *hot spots* no arcabouço. Em um ambiente orientado a objetos, isso é feito por meio da composição ou da herança de classes existentes (Buschmann et al., 1996). Na instanciação por instalação, ao contrário, nenhuma adaptação à implementação do arcabouço subjacente é necessária. Nesse caso, a implementação engloba somente *frozen spots*, que precisam ser genéricos o suficiente para permitir a instanciação do arcabouço por meio da configuração de *hot spots* parametrizados, principalmente na forma de documentos de configuração. Aqui, tratamos somente da tarefa de instanciação por instalação.

A tarefa de instalação engloba a preparação e a configuração de diversos artefatos contidos no arcabouço subjacente: arquivos de programas, diretórios de instalação, documentos de configuração, variáveis de ambiente, *links* ou atalhos, etc. Além de trabalhosa, essa tarefa é altamente propensa a erros: diversas partes do arcabouço precisam ser corretamente configuradas e partes potencialmente interdependentes precisam ser corretamente combinadas de forma a garantir o funcionamento apropriado do sistema instanciado. Apesar dessa inerente complexidade, os arcabouços de software existentes para a construção de bibliotecas digitais geralmente exigem que os usuários realizem a tarefa de instalação de forma manual ou por meio da interação com *scripts* em linha de comando. Ambas as alternativas, entretanto, mostram-se inadequadas em um contexto amplo de utilização de bibliotecas digitais, no qual esses sistemas devem ser instalados por organizações com requisitos distintos e administrados por usuários com diferentes níveis de conhecimento de computadores (Suleman et al., 2005).

---

<sup>1</sup><http://www.greenstone.org>

<sup>2</sup><http://www.eprints.org>

<sup>3</sup><http://www.dspace.org>

<sup>4</sup><http://www.fedora.info>

## 1.2 Contribuições da Dissertação

Para tentar facilitar a tarefa de instalação de bibliotecas digitais a partir de arcabouços de software, nesta dissertação, apresentamos WhizKEY – *Wizard-based blocK Ensemble Yelder*. Do ponto de vista de desenvolvedores de arcabouços de software para construção de bibliotecas digitais, WhizKEY possibilita a disponibilização desses arcabouços a um público mais amplo, na forma de assistentes de instalação. Do ponto de vista de projetistas de bibliotecas digitais, WhizKEY permite que eles se concentrem nas bibliotecas sendo construídas e não na complexidade do processo de instalação do arcabouço subjacente. Para tanto, WhizKEY implementa um fluxo de instalação, guiando o projetista, por meio de uma interface gráfica intuitiva, ao longo de passos bem definidos em direção à instalação de sistemas funcionais. O ambiente possibilita uma completa personalização desse fluxo, sem a necessidade de recodificação, de modo a permitir a instalação de sistemas baseados em diferentes arcabouços.

Em resumo, as principais contribuições desta dissertação são:

- Desenvolvimento de um ambiente para instalação de bibliotecas digitais a partir de diferentes arcabouços de software (Roberto et al., 2006; Santos et al., 2008);
- Desenvolvimento de um modelo geral para arcabouços de software voltados à construção de bibliotecas digitais (Santos, 2005; Santos et al., 2007);
- Avaliação de um protótipo do ambiente quanto a uma série de fatores relacionados à sua usabilidade (Santos et al., 2006, 2008).

## 1.3 Organização da Dissertação

Esta dissertação está organizada como segue. No Capítulo 2, são apresentados dois arcabouços de software que serviram de base para o desenvolvimento do modelo geral para arcabouços implementado pelo ambiente WhizKEY, além de uma seleção de abordagens relacionadas à tarefa de construção de bibliotecas digitais, comparativamente à desenvolvida nesta dissertação. No Capítulo 3, são descritas a arquitetura e a implementação do ambiente WhizKEY, além de um exemplo de uso do ambiente para a instalação de uma biblioteca digital funcional. No Capítulo 4, são discutidos o projeto experimental e os resultados de avaliações de usabilidade conduzidas com um protótipo de WhizKEY, os quais atestaram a efetividade da abordagem de assistência implementada pelo protótipo e serviram de base para sua reformulação em uma nova versão, apresentada nesta dissertação. Finalmente, no Capítulo 5, são apresentadas as conclusões desta dissertação, além de direções para trabalhos futuros.

## Capítulo 2

# Trabalhos Relacionados

Neste capítulo, apresentamos uma seleção de trabalhos relacionados em dois grandes grupos. A Seção 2.1 descreve dois arcabouços de software que serviram como ponto de partida para o desenvolvimento do ambiente WhizKEY. A Seção 2.2 apresenta uma seleção de ferramentas destinadas à construção de bibliotecas digitais.

### 2.1 Arcabouços de Software

Vários são os trabalhos apresentados na literatura que descrevem arcabouços de software para a construção de bibliotecas digitais. Nesta seção, descrevemos a organização e o funcionamento de dois desses arcabouços, que serviram de base para o desenvolvimento do modelo para arcabouços implementado pelo ambiente WhizKEY, descrito no Capítulo 3.

#### 2.1.1 Arcabouço ODL

O desenvolvimento do arcabouço ODL – *Open Digital Libraries* (Suleman e Fox, 2001) foi inspirado no conceito de arquivos abertos (*open archives*). Esse conceito surgiu a partir de uma iniciativa em resposta à necessidade de soluções de baixo custo para a interoperabilidade de bibliotecas digitais. A iniciativa encorajava, ainda, uma abordagem de desenvolvimento em camadas, com uma separação clara entre provedores de dados e serviços. Para possibilitar a comunicação entre dados e serviços, foi definido um protocolo de rede para a transferência de metadados, o OAI-PMH – *Open Archives Initiative Protocol for Metadata Harvesting* (Lagoze e Van de Sompel, 2001). Esse protocolo define seis verbos, descritos na Tabela 2.1.

Tabela 2.1: Requisições definidas pelo protocolo OAI-PMH

Requisição	Definição
1 Identify	Recupera informações sobre um repositório
2 ListSets	Recupera a estrutura de conjuntos de um repositório
3 ListMetadataFormats	Recupera os formatos de metadados disponíveis em um repositório
4 ListIdentifiers	Recupera cabeçalhos de registros de metadados em um repositório
5 ListRecords	Recupera registros de metadados em um repositório
6 GetRecord	Recupera um único registro de metadados de um repositório



O arcabouço ODL foi um dos primeiros arcabouços para a construção de bibliotecas digitais a utilizar a abordagem de desenvolvimento baseado em componentes de software. Essa abordagem enfatiza a decomposição de sistemas em componentes funcionais ou lógicos, com interfaces bem definidas para comunicação com outros componentes, e visa à construção de grandes sistemas de software a partir da integração desses componentes (Heineman e Councill, 2001). Para comunicação entre componentes, o arcabouço ODL definiu uma família de protocolos – extensões do protocolo OAI-PMH – e um conjunto inicial de nove componentes, descritos na Tabela 2.2, como implementações de referência dos protocolos definidos.

Tabela 2.2: Componentes ODL

Componente	Descrição
1 Box	Repositório para submissão e recuperação de registros
2 IRDB	Mecanismo de busca
3 DBBrowse	Navegador baseado em categorias
4 DBRate	Gerenciador de avaliações
5 DBReview	Gerenciador de fluxos de revisão por pares
6 DBUnion	Unificador de fontes de dados
7 Suggest	Sistema de recomendação
8 Thread	Mecanismo para anotações hierarquizadas
9 WhatsNew	Rastreador de registros recentes

Cada um desses componentes, implementados em Perl<sup>1</sup>, foi concebido como um arquivo aberto estendido, com recursos adicionais para possibilitar a execução do serviço por ele implementado. Uma biblioteca digital baseada no arcabouço ODL pode ser vista, dessa forma, como uma rede de arquivos abertos estendidos, onde cada nodo atua como um provedor de dados, um provedor de serviços ou, mesmo, de forma híbrida, conforme ilustra a Figura 2.1.

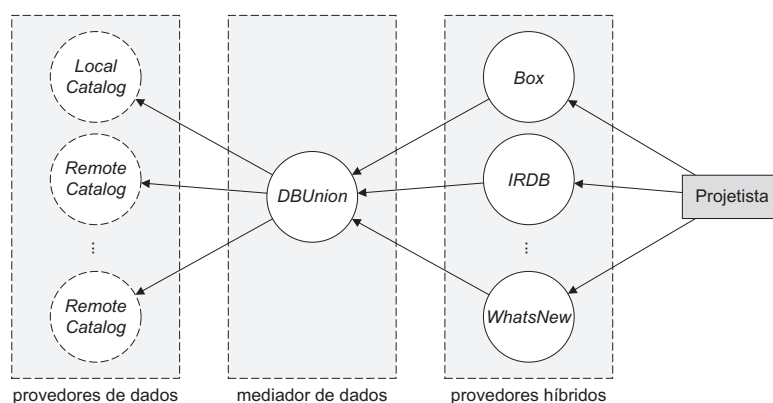


Figura 2.1: Biblioteca digital segundo o arcabouço ODL

A instalação de uma biblioteca digital baseada no arcabouço ODL envolve a seleção e a preparação de um subconjunto dos componentes por ele disponibilizados, bem como a configuração dos diversos parâmetros associados a cada componente e, ainda, a resolução de dependências entre componentes ou, mesmo, dependências dentro de um mesmo componente.

<sup>1</sup><http://www.perl.org/>

### 2.1.2 Arcabouço WS-ODL

Apesar dos diversos benefícios decorrentes de um embasamento na filosofia de arquivos abertos e da utilização da abordagem de desenvolvimento baseado em componentes, o arcabouço ODL apresenta problemas relacionados a eficiência, escalabilidade e interoperabilidade. Os problemas de eficiência decorrem principalmente de algumas decisões de implementação, que também desencadearam problemas de escalabilidade, a maioria deles relacionada à implementação de funções de recuperação de informação. Os problemas de interoperabilidade, por sua vez, são relacionados à utilização de um protocolo específico para esse arcabouço.

Para abordar esses problemas, foi desenvolvido o arcabouço WS-ODL – *Web Services-based Open Digital Libraries* (Roberto et al., 2006) para a construção de bibliotecas digitais. Desenvolvido em camadas, esse arcabouço é composto de três partes:

- uma camada de dados, com um repositório que suporta serviços básicos de infraestrutura, tais como um provedor de dados OAI;
- uma camada lógica, com um conjunto de serviços Web para a provisão de serviços específicos de uma biblioteca digital, tais como busca e navegação;
- uma camada de fronteira, responsável pela geração de interfaces de usuário a partir do conteúdo retornado pelos serviços Web.

A camada de dados foi originalmente baseada no repositório Fedora (Lagoze et al., 2006), uma arquitetura orientada a serviços para o armazenamento, gerência e disseminação de objetos complexos e seus relacionamentos. Essa arquitetura é implementada como um serviço Web e provê um alicerce para a construção de outras aplicações, como uma biblioteca digital.

Um serviço Web é uma solução utilizada para possibilitar a comunicação entre aplicações diferentes. As bases para a construção de um serviço Web são os padrões XML<sup>2</sup> e SOAP<sup>3</sup>, utilizados para a codificação de mensagens entre aplicações, normalmente transportadas via HTTP. Essa solução é utilizada pelo arcabouço WS-ODL para a implementação da camada lógica, que contempla um conjunto de serviços de valor agregado sobre a infraestrutura disponibilizada pelo repositório de dados. A Tabela 2.3 lista cada um desses serviços, implementados como componentes de software, bem como sua descrição.

Tabela 2.3: Componentes WS-ODL

Componente	Descrição
1 Administration	Conjunto de serviços de administração
2 Annotation	Mecanismo para anotações
3 Browsing	Navegador multidimensional
4 Recent Works	Rastreador de registros recentes
5 User Registration	Mecanismo para registro de usuários
6 Searching	Mecanismo de busca por palavras-chave ou estruturada

<sup>2</sup><http://www.w3.org/XML/>

<sup>3</sup><http://www.w3.org/TR/soap/>

A independência entre as camadas de dados e lógica é garantida por meio da utilização de um banco de dados relacional. Esse banco de dados funciona como um mediador entre as camadas e possibilita a operação dos serviços do arcabouço sobre diferentes repositórios.

Por fim, a camada de fronteira, implementada como um conjunto de Java Servlets<sup>4</sup>, provê interfaces gráficas para os serviços disponibilizados pelo arcabouço. Cada um dos componentes de interface é responsável por tratar e encaminhar requisições a um serviço, além de apresentar o conteúdo retornado pelo serviço ao usuário, por meio de transformações XSL<sup>5</sup>.

A divisão em camadas e a utilização de serviços Web permite que se construam bibliotecas digitais distribuídas a partir do arcabouço WS-ODL. Em outras palavras, cada uma das camadas que compõem o arcabouço pode ser hospedada em um servidor diferente. A Figura 2.2 mostra uma visão geral de uma biblioteca digital construída a partir do arcabouço WS-ODL.

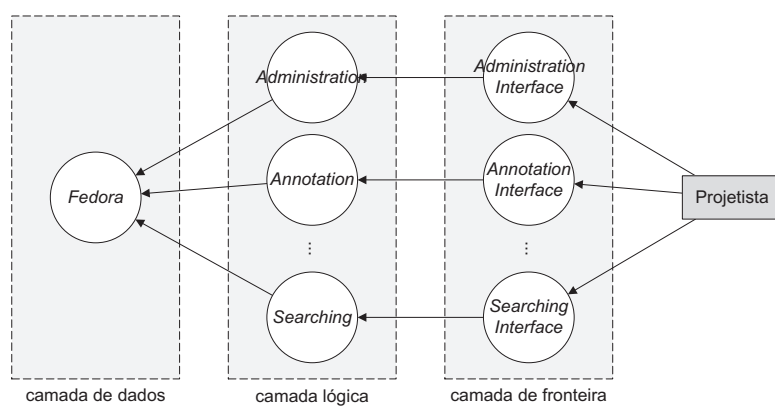


Figura 2.2: Biblioteca digital segundo o arcabouço WS-ODL

A instalação de uma biblioteca digital a partir do arcabouço WS-ODL envolve passos específicos para os componentes em cada camada. Primeiramente, deve ser instalado o repositório de dados. No caso do Fedora, essa atividade envolve a configuração de dezenas de parâmetros e variáveis de ambiente, além da criação de um banco de dados para apoio à execução do repositório. A seguir, devem ser instalados os serviços Web desejados, bem como seus respectivos componentes de interface. Para a instalação dos serviços Web, é necessária a instalação do Apache Axis<sup>6</sup>, um arcabouço para criação e distribuição de aplicações baseadas nesse tipo de solução. Após essa preparação, os serviços Web selecionados devem ser instalados no arcabouço Axis e novas variáveis de ambiente devem ser configuradas. Por fim, devem ser instalados os componentes de interface, o que envolve a integração desses componentes à estrutura do repositório de dados, bem como a configuração de novos parâmetros.

<sup>4</sup><http://java.sun.com/products/servlet/>

<sup>5</sup><http://www.w3.org/TR/xsl/>

<sup>6</sup><http://ws.apache.org/axis/>

## 2.2 Ferramentas para Construção de Bibliotecas Digitais

Apesar da grande quantidade de ambientes para geração de assistentes de instalação de propósito geral disponíveis, tanto proprietários (e.g., InstallShield e InstallAnywhere<sup>7</sup>, Wise<sup>8</sup>) quanto livres (e.g., NSIS<sup>9</sup>, Inno Setup<sup>10</sup>, WiX<sup>11</sup>, IzPack<sup>12</sup>), há poucos trabalhos dedicados à construção de bibliotecas digitais especificamente. Nesta seção, apresentamos dois desses trabalhos, além de outras duas ferramentas desenvolvidas para arcabouços específicos.

### 2.2.1 BLOX

Eyambe e Suleman (2004) desenvolveram a ferramenta BLOX para a construção de bibliotecas digitais a partir de componentes de software potencialmente distribuídos. Como principal objetivo, esse trabalho visa a simplificar a criação de bibliotecas digitais, colocando-as ao alcance de usuários não necessariamente tecnologistas, como bibliotecários, por exemplo.

A ferramenta BLOX foi implementada como uma arquitetura cliente-servidor, de modo a permitir a instanciação de bibliotecas digitais remotamente. Nesse modelo, o cliente é representado por uma interface gráfica, única parte com a qual os usuários interagem diretamente. Essa interface é apresentada na Figura 2.3.

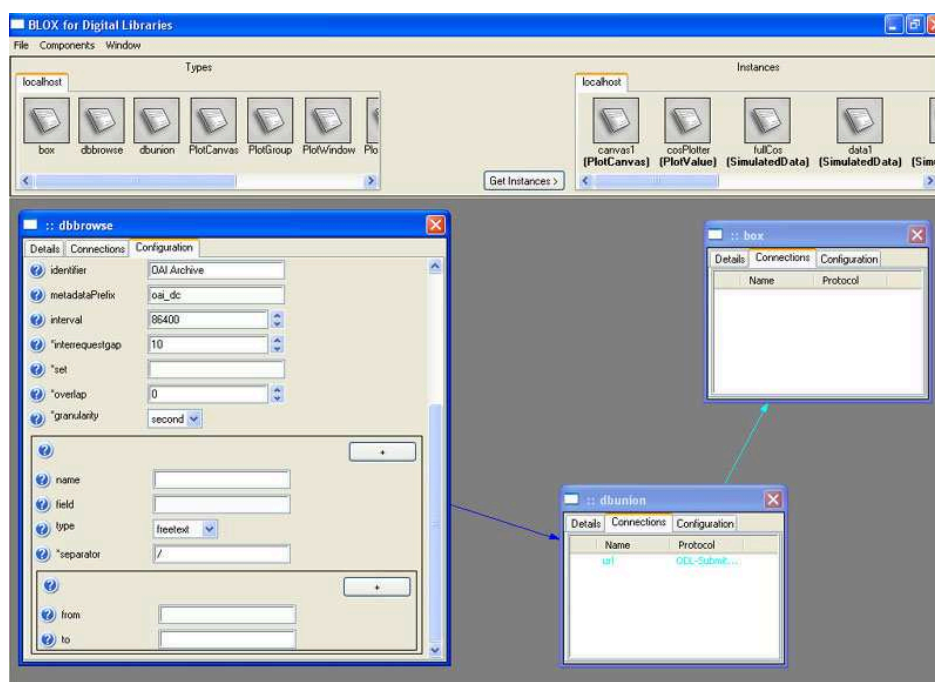


Figura 2.3: Instantâneo da parte cliente da ferramenta BLOX

<sup>7</sup><http://www.macrovision.com>

<sup>8</sup><http://www.wise.com>

<sup>9</sup><http://nsis.sourceforge.net>

<sup>10</sup><http://www.jrsoftware.org/isinfo.php>

<sup>11</sup><http://wix.sourceforge.net>

<sup>12</sup><http://www.izpack.org>

Nessa interface, os vários componentes disponíveis são representados por ícones, um ícone para cada tipo de componente. A criação de instâncias de um determinado componente é feita arrastando-se o ícone desse componente para a área principal da interface. Cada instância de um componente é representada por uma janela e os parâmetros associados a cada instância são representados por um formulário nessa janela (na Figura 2.3, são configuradas instâncias dos componentes ODL Box, DBUnion e DBBrowse). As conexões entre componentes, indicando fluxos de dados entre suas instâncias, são representadas por setas entre janelas.

A parte servidora é responsável por gerenciar toda a comunicação entre o cliente e os componentes a serem instanciados. Para tanto, ela é executada no mesmo sistema onde se encontram esses componentes. A utilização de diferentes componentes é possibilitada por meio de uma API que, essencialmente, encapsula esses componentes de modo a prover uma interface padronizada para o gerenciamento de suas instâncias. Um elemento importante dessa interface é a descrição do tipo do componente, que contém a informação necessária para se configurarem instâncias desse componente. Essa descrição é implementada como um esquema XSD<sup>13</sup>, de modo que instâncias de um determinado componente são descritas por documentos XML em conformidade com esse esquema. Além de uma descrição, a interface de um componente define um conjunto de requisições que ele deve atender, tais como a criação, a recuperação e a remoção de instâncias desse componente. A Figura 2.4 fornece uma visão geral da arquitetura da ferramenta BLOX.

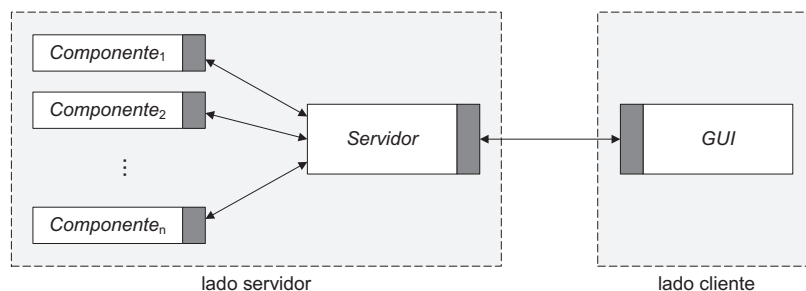


Figura 2.4: Visão geral da arquitetura da ferramenta BLOX

Alternativamente à organização ilustrada na Figura 2.4, essa arquitetura oferece flexibilidade quanto à utilização de clientes e servidores: clientes e servidores podem residir em um mesmo sistema; um cliente pode se comunicar com diversos servidores remotos e, reciprocamente, um servidor pode se comunicar com diversos clientes.

A ferramenta BLOX foi testada em um ambiente controlado por 23 estudantes de Ciência da Computação e 11 de outros cursos, a maioria com pouco ou nenhum conhecimento sobre bibliotecas digitais (Eyambe, 2005). Embora os testes realizados tenham atestado a efetividade da ferramenta BLOX quanto a seu propósito principal, i.e., simplificar a criação de bibliotecas digitais, acreditamos que a provisão de uma interação guiada, em contrapartida à interação flexível oferecida por essa ferramenta, possa melhorar ainda mais a experiência dos usuários na realização da tarefa de criação desses sistemas a partir de arcabouços de software.

<sup>13</sup><http://www.w3.org/XML/Schema>

### 2.2.2 5SGraph

Zhu et al. (2003) desenvolveram a ferramenta 5SGraph para a modelagem visual de bibliotecas digitais a partir de metamodelos predefinidos. Essa ferramenta auxilia a especificação e a análise de requisitos no processo proposto pelo arcabouço 5S – *Streams, Structures, Spaces, Scenarios, Societies* (Gonçalves et al., 2004) para a geração de bibliotecas digitais. Esse arcabouço provê um modelo formal para bibliotecas digitais, possibilitando a especificação das características e comportamentos de uma biblioteca digital de maneira unívoca, bem como o seu mapeamento para implementações reais. Gonçalves e Fox (2002) propuseram uma linguagem, denominada 5SL, para a especificação declarativa e a geração de bibliotecas digitais com base no modelo 5S. A linguagem 5SL pode ser vista como uma materialização em XML desse modelo. Para possibilitar a geração semi-automática de bibliotecas digitais com base nessa linguagem, o arcabouço propõe um processo, segmentado em duas fases, conforme ilustrado no diagrama da Figura 2.5.

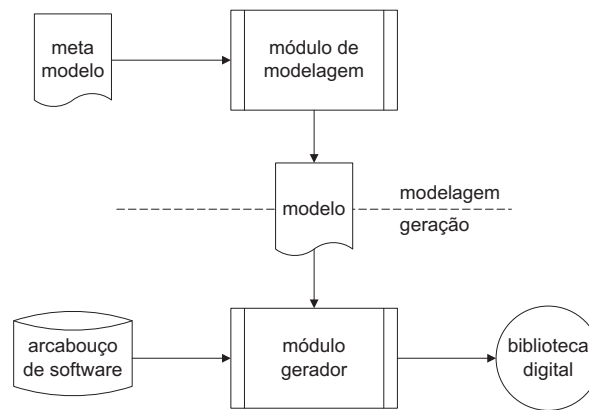


Figura 2.5: Abordagem 5S para o desenvolvimento de bibliotecas digitais

A primeira fase envolve uma tarefa de modelagem, na qual é gerada uma especificação dos requisitos da biblioteca digital a ser criada. Essa especificação é codificada em um documento 5SL, que servirá como entrada para a fase seguinte. A segunda fase é responsável pela geração de um sistema funcional de acordo com a modelagem produzida na fase anterior. A partir do arcabouço 5S, foram desenvolvidos módulos para a execução de cada uma dessas fases.

A segunda fase é executada automaticamente por um módulo gerador. A ferramenta 5SGen (Kelapure, 2003) apresentou uma primeira implementação desse módulo. Essa ferramenta recebe uma especificação 5SL e gera grande parte do código necessário para a implementação da biblioteca digital correspondente, por meio da instanciação de um conjunto de componentes de software. Orientada a serviços, a ferramenta possibilita a utilização de especificações em um nível detalhado para produzir as classes que executarão os serviços da biblioteca digital criada. Seus resultados são demonstrados através de múltiplos estudos de caso (Gonçalves et al., 2001; Kelapure, 2003). Posteriormente, uma segunda implementação desse módulo foi desenvolvida por Gorton (2007) para a geração de bibliotecas digitais baseadas no arcabouço DSpace (Tansley et al., 2003).

O módulo gerador recebe como entrada o modelo produzido na primeira fase do processo, na forma de uma especificação 5SL. A geração manual dessa especificação mostrou-se uma tarefa tediosa e demorada, além de demandar um profundo conhecimento da linguagem 5SL por parte dos projetistas de bibliotecas digitais. Para tentar reduzir esses problemas, foi desenvolvida a ferramenta 5SGraph para a modelagem visual de bibliotecas digitais a partir de metamodelos produzidos por especialistas. Essa ferramenta tem o objetivo principal de abstrair a tarefa de modelagem baseada na linguagem 5SL de seus detalhes técnicos, facilitando a geração de modelos e a compreensão desses modelos. Essa abstração é efetuada por meio da representação visual dos diversos conceitos de uma biblioteca digital e dos relacionamentos entre esses conceitos. Para tanto, a interface gráfica da ferramenta provê um editor estruturado para permitir ao projetista construir uma biblioteca digital por meio da manipulação e da composição de componentes visuais (Figura 2.6).

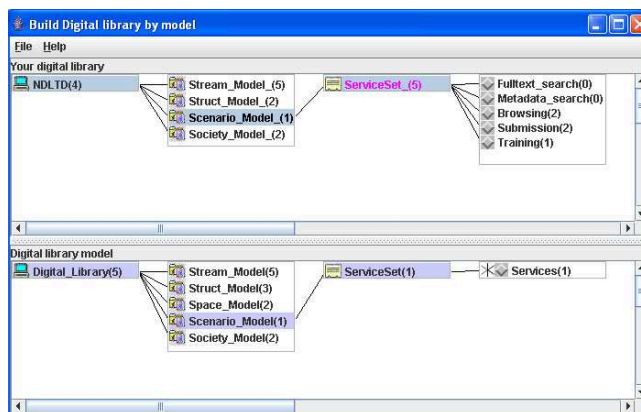


Figura 2.6: Instantâneo da ferramenta 5SGraph

Esses componentes são organizados visualmente (área inferior na interface) em uma estrutura hierárquica correspondente ao metamodelo utilizado pela ferramenta. Nessa interface, a modelagem é feita de maneira *top-down*, por meio da construção incremental de uma árvore onde cada nodo, obtido a partir do metamodelo, corresponde à instância de um conceito do metamodelo. A biblioteca digital sendo modelada (área superior na interface), instância do metamodelo, é representada de maneira semelhante. A ferramenta permite, ainda, a reutilização de partes de modelos previamente elaborados, por meio de sua anexação à árvore sendo construída. A visualização de ambas as árvores – modelo e metamodelo – é sincronizada, i.e., o foco em um elemento de uma árvore traz o foco para o elemento correspondente da outra árvore automaticamente. Além disso, a ferramenta mantém restrições semânticas, especificadas pelo metamodelo, sobre as instâncias produzidas (e.g., dependências entre partes do modelo), de forma a garantir a consistência e a correção semânticas dessas instâncias.

A ferramenta 5SGraph difere do ambiente WhizKEY e das demais ferramentas descritas neste capítulo por duas características principais: (1) essa ferramenta realiza uma tarefa de modelagem conceitual, em contrapartida à tarefa de instalação realizada pelas outras ferramentas, e (2) diferentemente das demais ferramentas aqui descritas, 5SGraph possui um objetivo didático, a saber, ensinar a teoria em que se baseia, i.e., o modelo 5S.

### 2.2.3 Soluções Específicas

Além das soluções de propósito geral anteriormente descritas, alguns arcabouços para a construção de bibliotecas digitais apresentam ferramentas para auxiliar sua instalação (ou parte dela), conforme discutido nesta seção.

A suíte Greenstone (Buchanan et al., 2005) incorpora um assistente que permite a usuários não-especialistas criar e organizar coleções digitais a partir de documentos locais ou remotos. Guiando o usuário passo a passo, o assistente obtém informações como o nome e o propósito da coleção, o e-mail de seu administrador, coleções existentes para serem utilizadas como modelo, diretórios ou URLs-fonte, etc. Esse assistente, entretanto, não trata da configuração de componentes provedores de serviços.

A aplicação OAIB – *Open Archives in a Box*<sup>14</sup>, parte do arcabouço COCOA – *Components for Constructing Open Archives* (Plutchak et al., 2002), provê uma ferramenta assistente para a configuração de provedores OAI (Lagoze e Van de Sompel, 2001) a partir de catálogos de metadados armazenados em SGBDs relacionais. Sua interface consiste de uma série de abas, cada uma com diferentes opções de configuração. Assim como o assistente provido pela suíte Greenstone, este não trata da configuração de provedores de serviços.

### 2.2.4 Comparação das Soluções

A Tabela 2.4 sumariza as características de todas as ferramentas descritas nesta seção comparativamente às do ambiente WhizKEY, com destaque para os pontos de distinção.

Tabela 2.4: WhizKEY versus ferramentas relacionadas

	<b>WhizKEY</b>	<b>BLOX</b>	<b>5SGraph</b>	<b>Greenstone</b>	<b>OAIB</b>
geral	sim	sim	sim	<i>não</i>	<i>não</i>
tarefa	instalação	instalação	<i>modelagem</i>	instalação	instalação
objetos	abstrações	<i>componentes</i>	<i>conceitos</i>	<i>coleções</i>	<i>catálogos</i>
interação	guiada	<i>não-guiada</i>	<i>não-guiada</i>	guiada	guiada
didática	não	não	<i>sim</i>	não	não

Conforme discutido no Capítulo 3, o ambiente WhizKEY permite a manipulação de partições arbitrárias de uma visão do arcabouço subjacente, não necessariamente correspondente a um mapeamento direto da subdivisão arquitetural do arcabouço. Por meio da abstração do software subjacente em conceitos de mais alto nível, do domínio do usuário, como catálogos de metadados e serviços, esse particionamento possibilita uma reorganização do arcabouço quando de sua apresentação ao usuário de forma a tornar sua instalação mais inteligível. Essa característica, aliada à generalidade do modelo implementado pelo ambiente de forma a possibilitar a instalação de bibliotecas digitais com base em diferentes arcabouços de software e à provisão de um mecanismo de assistência para guiar o usuário ao longo do fluxo de instalação desses arcabouços, tornam o ambiente WhizKEY uma poderosa alternativa.

<sup>14</sup><http://dlt.ncsa.uiuc.edu/oaib>



## Capítulo 3

# O Ambiente WhizKEY

Com a crescente demanda por bibliotecas digitais por parte de comunidades com necessidades bastante particulares e a oferta de diferentes opções de arcabouços de software para a construção de bibliotecas digitais, um ambiente integrado para a instalação e modificação desses sistemas e que permita a realização dessas tarefas de uma maneira unificada, independente de um arcabouço específico, torna-se altamente desejável. Tal ambiente deve apresentar três características principais: (i) *flexibilidade* para suportar as especificidades dos fluxos de instalação de diferentes arcabouços de software, (ii) *consistência* para garantir a funcionamento adequado dos sistemas por ele instanciados, e (iii) *facilidade de uso* para possibilitar sua utilização mesmo por usuários não-especialistas.

Para satisfazer esses requisitos, desenvolvemos WhizKEY (Santos et al., 2006, 2007, 2008), um ambiente para a instalação assistida de bibliotecas digitais a partir de arcabouços de software potencialmente distintos. WhizKEY foi desenvolvido com o intuito de se diferenciar dos ambientes para a geração de assistentes de instalação disponíveis no mercado, voltados, em sua maioria, para a instalação de softwares aplicativos. Para tanto, seu desenvolvimento foi baseado em fluxos de instalação de bibliotecas digitais, caracterizados por uma intensa carga de configuração e pela presença de dependências entre diversas partes do sistema sendo instalado, e tendo projetistas de bibliotecas digitais como seus usuários-alvo. A Figura 3.1 mostra uma visão geral do processo de instalação assistido pelo ambiente WhizKEY.

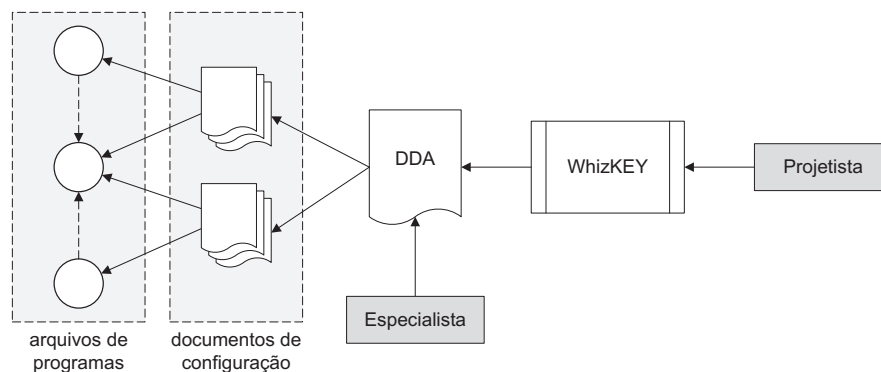


Figura 3.1: Processo de instalação assistido pelo ambiente WhizKEY

De acordo com a Figura 3.1, a partir de um arcabouço de software, constituído, basicamente, por um conjunto de arquivos de programas potencialmente interdependentes e um conjunto de documentos de configuração que expressam o comportamento desses programas, um especialista – possivelmente, o próprio desenvolvedor do arcabouço – produz um documento de especificação, chamado DDA – *Documento Descritor do Arcabouço*. Esse documento funciona como um mapeamento entre um arcabouço específico e o modelo geral implementado pelo ambiente WhizKEY. A partir desse mapeamento, o ambiente WhizKEY apresenta o processo de instalação do arcabouço de maneira assistida, como um fluxo a ser percorrido por um projetista, não necessariamente conhecedor da organização interna e do funcionamento desse arcabouço, a fim de que possa construir uma biblioteca digital nele baseada.

Embora focado em bibliotecas digitais, o ambiente WhizKEY não é restrito à instalação desse tipo de sistema. Sua arquitetura extensível poderia ser utilizada para a geração de assistentes para a instalação de outros sistemas que não bibliotecas digitais, mas essa investigação está além do escopo desta dissertação. As decisões de projeto e as principais características da arquitetura do ambiente WhizKEY são discutidas na próxima seção. As seções seguintes descrevem a implementação e o funcionamento de cada uma das quatro camadas que compõem essa arquitetura, bem como um exemplo completo de uso do ambiente WhizKEY para instalação de uma biblioteca digital funcional baseada no arcabouço ODL.

### 3.1 Visão Geral da Arquitetura

O desenvolvimento do ambiente WhizKEY foi norteado por princípios consagrados da engenharia de software orientado a objetos (Pressman, 1982), dentre os quais se destacam modularidade, portabilidade e tratamento de erros. Tais considerações foram decisivas para facilitar a futura manutenção de sua implementação e eventuais extensões à sua arquitetura.

Implementada em Java SE 6<sup>1</sup>, a arquitetura do ambiente WhizKEY é baseada no padrão MVC (Modelo-Visualização-Controle) com a adição de uma camada de persistência. Originalmente criado como um padrão de projeto arquitetural para o ambiente Smalltalk (Burbeck, 1987), o MVC passou a ser largamente utilizado em aplicações interativas. Essencialmente, esse padrão busca separar as tarefas de acesso a dados, interação com o usuário e lógica do negócio, dividindo-as em três camadas, a fim de evitar o efeito cascata provocado por alterações em qualquer uma delas. Essas camadas são ilustradas na Figura 3.2 e assim descritas:

**Modelo.** Representa as entidades do domínio em que a aplicação opera sem, necessariamente, especificar detalhes relacionados ao seu armazenamento.

**Visualização.** Apresenta as entidades do modelo em uma forma específica para a interação – geralmente, uma interface de usuário.

**Controle.** Processa e responde a eventos – geralmente, ações do usuário – e pode invocar alterações sobre as entidades modeladas.

---

<sup>1</sup><http://java.sun.com/javase/6/>

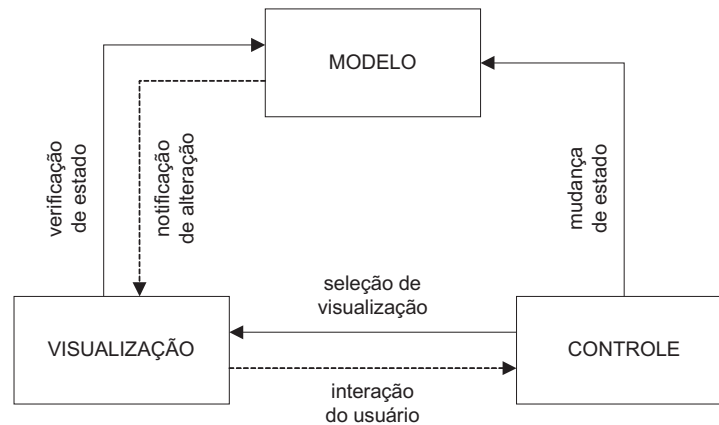


Figura 3.2: Esquema de funcionamento do padrão MVC

Ainda que existam variações, o padrão MVC geralmente funciona da seguinte maneira:

1. O usuário interage com a camada de visualização.
2. A camada de controle manipula o evento recebido pela camada de visualização.
3. A camada de controle efetua as alterações correspondentes na camada modelo.
4. A camada de visualização é atualizada conforme o novo estado da camada modelo.

Para garantir a independência entre as camadas, as notificações de interações do usuário e de alterações no estado do modelo (setas tracejadas na Figura 3.2) são feitas por meio de associações indiretas. Em outras palavras, a camada de visualização não possui conhecimento da implementação da camada de controle, assim como a camada modelo não possui conhecimento da implementação da camada de visualização. No primeiro caso, eventos recebidos pela camada de visualização são automaticamente repassados à camada de controle para que sejam devidamente tratados. No segundo caso, a camada de visualização registra-se para receber notificações da camada modelo sempre que o estado dessa camada é modificado.

## 3.2 Modelo: Representação de Entidades

A fim de satisfazer os requisitos de flexibilidade e consistência, principalmente, a camada modelo foi organizada de modo a possibilitar a instalação de bibliotecas digitais a partir de diferentes arcabouços de software, bem como a garantir a correção dos sistemas instalados. A Figura 3.3 ilustra essa organização, detalhada nas subseções seguintes.

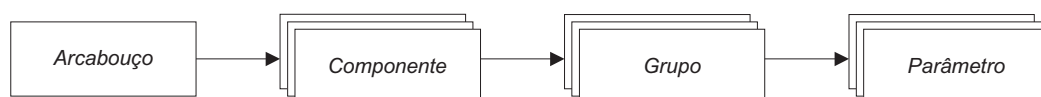


Figura 3.3: Visão geral da camada modelo

### 3.2.1 Especificação de Arcabouços

O modelo implementado pelo ambiente WhizKEY representa uma visão do arcabouço de software subjacente. Essa visão é especificada por um especialista a fim de que o fluxo de instalação do arcabouço seja apresentado para o usuário da maneira mais inteligível possível. Idealmente, arcabouços distintos devem ser apresentados de maneira semelhante de modo a permitir que o usuário tenha uma experiência unificada quando da instalação de sistemas a partir de diferentes arcabouços. Essa visão unificada é organizada de maneira hierárquica, conforme ilustra a Figura 3.3. No nível mais alto, um *arcabouço* é composto por um conjunto de componentes. Um *componente* representa uma subdivisão arbitrária do arcabouço subjacente<sup>2</sup> e engloba um conjunto de *parâmetros*, organizados semanticamente em *grupos*.

A especificação das diversas estruturas que compõem essa organização é expressa por meio de um DDA – *Documento Descritor do Arcabouço*. A Figura 3.4 apresenta o excerto de um DDA para o arcabouço ODL, listado completamente no Apêndice A. Um DDA contém três seções principais: *definitions*, *steps* e *components*. Na primeira e mais simples delas, *definitions*, podem-se declarar variáveis que estarão disponíveis globalmente para o restante da especificação. Na segunda seção, *steps*, são especificados os diversos passos que comporão o fluxo de instalação do arcabouço. Finalmente, na última e, geralmente, mais extensa das seções, *components*, são especificados os diversos componentes que serão instanciados pela camada modelo, conforme detalhado a seguir. As discussões sobre a utilização de variáveis e a especificação do fluxo de instalação são retomadas nas Seções 3.2.2 e 3.4, respectivamente.

```
<framework id="odl">
  <name>Open Digital Libraries</name>
  <definitions>
    <definition id="baseDir">/var/www/test</definition>
    ...
  <steps>
    <system>
      <step id="welcome">
        ...
      </step>
    </system>
    <user>
      <step id="catalogs" fixed="false" min="1" max="unbounded">
        ...
      </step>
    </user>
  </steps>
  <components>
    <component id="lrepo" step="catalogs" max="unbounded">
      <name>Local Catalog</name>
      <deployment/>
      <configuration>
        <group id="general">
          <label>General Configurations</label>
          <parameters>
            <parameter id="catalogURL" visible="once">
              ...
            </parameter>
          </parameters>
        </group>
      </configuration>
    </component>
  </components>
</framework>
```

Figura 3.4: Visão geral de um DDA

---

<sup>2</sup>Se o arcabouço subjacente é baseado em componentes de software, cada um deles pode ser diretamente representado como um componente no modelo, mas esse mapeamento não é obrigatório.

### 3.2.1.1 Especificação de Componentes

A especificação de um componente engloba um conjunto de atributos responsáveis pela identificação e localização de instâncias desse componente dentro do fluxo de instalação do arcabouço, bem como restrições de cardinalidade relacionadas ao número permitido dessas instâncias. A Tabela 3.1 apresenta esses atributos, incluindo sua descrição, definição do domínio de valores que podem assumir (em termos de expressões regulares ou conjuntos) e seu valor padrão. Em particular, o atributo *step* deve receber o identificador de um dos passos especificados no fluxo de instalação, conforme discutido na Seção 3.4. De acordo com o DDA da Figura 3.4, por exemplo, um número ilimitado de instâncias do componente “lrepo” – um catálogo de metadados – podem ser configuradas no passo “catalogs”.

Tabela 3.1: Atributos de um componente

Atributo	Descrição	Domínio	Padrão
<i>id</i>	Identificador	$[a-zA-Z\_][a-zA-Z0-9]^+$	–
<i>step</i>	Passo de configuração	$\#(step.id)$	–
<i>min</i>	Cardinalidade mínima	$\{a \in \mathbb{Z} \mid a \geq 0\}$	0
<i>max</i>	Cardinalidade máxima	$\{b \in \mathbb{Z} \mid b \geq a\}$	1

Além de um conjunto de atributos, a especificação de cada componente possui duas seções principais: *deployment* e *configuration*. A seção *deployment* carrega instruções a serem executadas quando da instalação ou desinstalação de uma instância do componente, conforme discutido na Seção 3.5. A seção *configuration*, por sua vez, detalha a especificação dos diversos parâmetros que compõem um componente, conforme detalhado na subseção seguinte.

### 3.2.1.2 Especificação de Parâmetros

Assim como ocorre com componentes, a especificação de um parâmetro engloba um conjunto de atributos que determinam esse parâmetro segundo vários aspectos. De acordo com o DDA da Figura 3.4, por exemplo, o parâmetro identificado por “catalogURL” deve ser exibido somente uma vez (atributo *visible*=“once”) quando da configuração de qualquer instância do componente “lrepo”. Os demais atributos associados a esse parâmetro foram omitidos de sua especificação, uma vez que possuem valores padrão, conforme indicados na Tabela 3.2.

Tabela 3.2: Atributos de um parâmetro

Atributo	Descrição	Domínio	Padrão
<i>id</i>	Identificador	$[a-zA-Z\_][a-zA-Z0-9]^+$	–
<i>visible</i>	Deve ser exibido?	$\{always, never, once\}$	<i>always</i>
<i>enabled</i>	Pode ser editado?	$\{true, false\}$	<i>true</i>
<i>persistent</i>	Deve ser persistido?	$\{true, false\}$	<i>true</i>
<i>fixed</i>	Conteúdo é fixo?	$\{true, false\}$	<i>false</i>
<i>min</i>	Cardinalidade mínima	$\{a \in \mathbb{Z} \mid a \geq 0\}$	1
<i>max</i>	Cardinalidade máxima	$\{b \in \mathbb{Z} \mid b \geq a\}$	1
<i>structure</i>	Organização estrutural	$\{scalar, flat, hierarchical\}$	<i>scalar</i>

De acordo com esses atributos, um parâmetro pode ser classificado sob várias dimensões:

**Visibilidade.** Com base no atributo *visible*, um parâmetro pode ser definido como visível ou invisível. Um parâmetro invisível não é apresentado pela camada de visualização, tendo seu valor determinado automaticamente, com base em alguma diretiva do ambiente ou no valor de outro parâmetro, sem o conhecimento do usuário. Ainda, a apresentação de um parâmetro pode ser limitada a uma única vez, de modo a evitar que seu valor seja posteriormente alterado.

**Disponibilidade.** A disponibilidade de um parâmetro pode ser condicionada pelo atributo *enabled*. Esse atributo é útil para possibilitar a visualização, por parte do usuário, de um parâmetro cujo valor é determinado automaticamente e que não pode ser alterado.

**Persistência.** O atributo *persistent* define um parâmetro como persistente ou transiente. Um parâmetro persistente tem seu valor carregado a partir de um documento de configuração no início do processo de instalação e, potencialmente, salvo em um ou mais documentos de configuração ao final do processo<sup>3</sup>. Um parâmetro transiente, por outro lado, não tem seu valor salvo em um documento de configuração. Parâmetros desse tipo são utilizados meramente para auxiliar a configuração de outros parâmetros.

**Cardinalidade.** O valor de um parâmetro pode ser composto por um ou mais itens. Um parâmetro cujo valor é composto por, no máximo, um item é classificado como monovalorado; caso contrário, esse parâmetro é classificado como multivalorado.

**Mutabilidade.** O atributo *fixed* determina se o conjunto de itens que compõem o valor de um parâmetro pode ou não ser alterado, casos em que o parâmetro considerado é classificado como mutável ou imutável, respectivamente. No caso de um parâmetro mutável, uma restrição de cardinalidade pode ser definida sobre o número de itens contidos em seu valor; para um parâmetro imutável, essa restrição define o número de itens que podem ser selecionados dentre todos os contidos no valor desse parâmetro. Restrições adicionais, aplicáveis sobre cada item individualmente, são descritas na Subseção 3.2.2.2.

**Estrutura.** Com base no atributo *structure*, o conjunto de itens que compõem o valor de um parâmetro multivalorado é organizado em uma estrutura de lista (estrutura linear) ou árvore (estrutura hierárquica). Parâmetros monovalorados têm estrutura obrigatoriamente escalar. Além de determinar o modo como parâmetros são armazenadas e tratados, a estrutura de um parâmetro é o principal atributo utilizado para sua exibição pela camada de visualização, conforme discutido na Seção 3.3.

---

<sup>3</sup>As tarefas de carga e salvamento são gerenciadas pela camada de persistência, descrita na Seção 3.5.

### 3.2.2 Gerenciamento de Configurações

Uma instância configurada de um componente é chamada um *bloco de configuração*. A camada modelo permite que vários blocos sejam configurados a partir de um mesmo componente (e.g., diversos catálogos de metadados podem ser configurados com base na infraestrutura provida pelo componente “lrepo”, especificado no DDA da Figura 3.4). Nessa representação, a configuração de um sistema contempla um conjunto de blocos associados aos diversos componentes especificados no DDA – daí o acrônimo WhizKEY, *Wizard-based Block Ensemble Yielder*.

A fim de que não haja dependências entre diversos sistemas durante o fluxo de instalação, somente um sistema é instalado por vez. Uma configuração engloba uma lista de blocos de cada componente especificado e é responsável pelo gerenciamento do ciclo de vida de cada um desses blocos, conforme ilustrado na Figura 3.5.

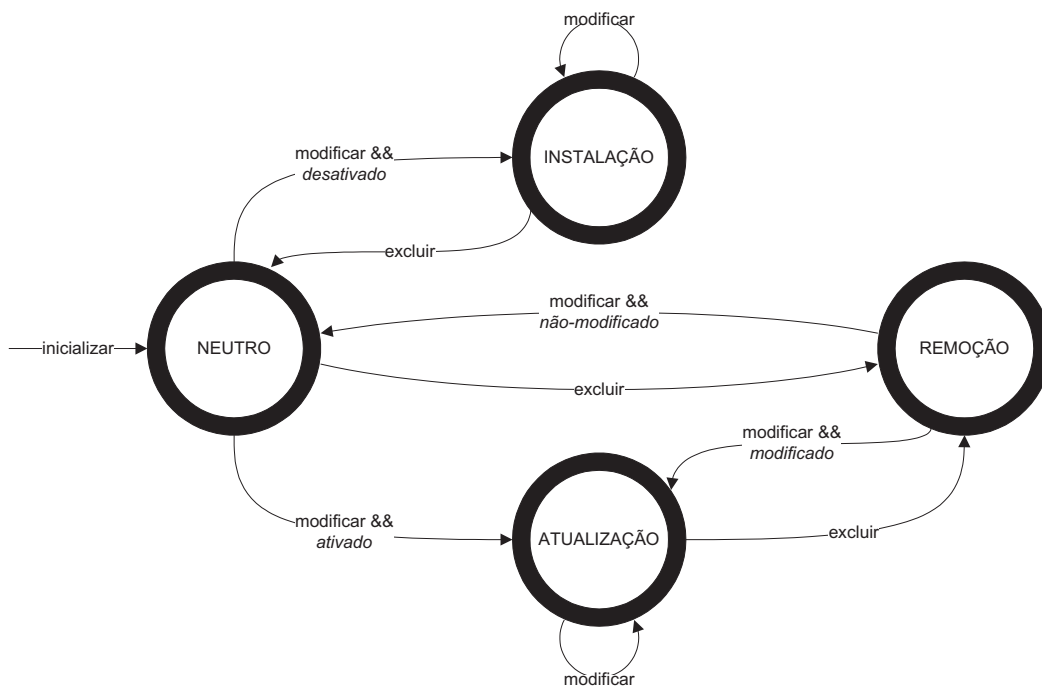


Figura 3.5: Ciclo de vida de um bloco

As transições no ciclo de vida de um bloco são realizadas por meio de três operações definidas sobre esse bloco:

- *inicializar*, que envolve a instanciação do bloco;
- *modificar*, que consiste na edição de parâmetros associados ao bloco;
- *excluir*, que consiste na remoção do bloco da configuração corrente.

De acordo com a Figura 3.5, um bloco é inicializado no estado NEUTRO e inicialmente marcado como *não-modificado*. Se essa inicialização se der por meio do carregamento de um bloco previamente instalado (e.g., um catálogo de metadados anteriormente configurado),

este é marcado como *ativado*; senão, caso um novo bloco seja criado, este é marcado como *desativado*. A modificação de um bloco no estado NEUTRO faz com que ele se seja marcado como *modificado* e altera seu estado para INSTALAÇÃO ou ATUALIZAÇÃO, dependendo se ele foi previamente marcado como *desativado* ou *ativado*, respectivamente. A modificação de um bloco nos estados INSTALAÇÃO ou ATUALIZAÇÃO não provoca transições no estado do bloco. A exclusão de um bloco no estado INSTALAÇÃO provoca seu retorno ao estado NEUTRO e sua marcação como *desativado*. A exclusão de um bloco nos estados NEUTRO ou ATUALIZAÇÃO, além de provocar sua marcação como *desativado*, provoca uma transição de seu estado para REMOÇÃO. Por fim, a modificação de um bloco no estado REMOÇÃO faz com que ele retorne ao estado ATUALIZAÇÃO ou NEUTRO, dependendo se ele foi previamente marcado como *modificado* ou *não-modificado*, respectivamente.

As operações sobre blocos são disparadas pelo usuário a partir da camada de visualização (Seção 3.3). O estado de um bloco, ao final do fluxo de instalação, é utilizado pela camada de controle (Seção 3.4) para determinar a operação apropriada a ser executada sobre esse bloco pela camada de persistência (Seção 3.5).

### 3.2.2.1 Resolução de Dependências

Dependências são um mecanismo implementado pelo ambiente WhizKEY para garantir a consistência de partes de um arcabouço de software individualmente, bem como em relação a outras partes. Esse mecanismo funciona a partir da declaração, em meio ao conteúdo textual de qualquer elemento XML contido no DDA (Subseção 3.2.1), de referências de dois tipos:

1. *referências internas*, que denotam dependências em relação a *variáveis*, que podem corresponder a definições globais disponibilizadas pelo ambiente WhizKEY (e.g., variáveis de ambiente do sistema operacional, diretivas da máquina virtual Java) ou especificadas na seção *definitions* do DDA ou, ainda, a valores de parâmetros especificados na seção *components* do DDA;
2. *referências externas*, que denotam dependências em relação a valores retornados por chamadas de sistema, i.e., valores retornados por *scripts* executados externamente.

Referências internas são expressas na forma  $\{\textit{referência}\}$ , em que *referência* é um espaço de nomes que satisfazem a expressão regular  $[a-zA-Z_][a-zA-Z0-9]^*$ . A utilização de um espaço de nomes permite que variáveis diferentes possuam um mesmo nome, porém especificado por contextos diferentes (e.g., as referências  $\{\textit{java.home}\}$ ,  $\{\textit{whizkey.home}\}$  e  $\{\textit{user.home}\}$  representam variáveis correspondentes aos diretórios-base da máquina virtual Java, do ambiente WhizKEY e do usuário corrente, respectivamente). Para o caso de variáveis correspondentes a parâmetros especificados no DDA, esse espaço de nomes corresponde à hierarquia do parâmetro, na forma  $\{\textit{componente.grupo.parâmetro}\}$ , em que *componente*, *grupo* e *parâmetro* são os identificadores dos elementos correspondentes nessa hierarquia (e.g., no DDA da Figura 3.4, uma referência à URL-base de uma instância do componente “Local Catalog” seria expressa na forma  $\{\textit{lrepo.general.catalogURL}\}$ ). Essa organização permite que se façam referências



parcialmente qualificadas, sensíveis ao contexto, a qualquer parâmetro especificado no DDA. Essas referências são feitas no escopo da especificação de um componente, grupo ou parâmetro, na forma  $\${referência}$ . O modificador ‘!’ deve ser interpretado como um desejo de que a referência seja expandida com base no escopo em que foi declarada, de acordo com as regras definidas na Tabela 3.3.

Tabela 3.3: Regras para expansão de referências

	Referência	Escopo	Expansão
1	ccc.ggg.ppp	componente, grupo ou parâmetro	ccc.ggg.ppp
2	ggg.ppp	componente, grupo ou parâmetro	<i>componente.ggg.ppp</i>
3	ppp	grupo ou parâmetro componente	<i>componente.grupo.ppp</i> –

De acordo com o DDA da Figura 3.4, uma referência da forma  $\${catalogURL}$ , declarada no escopo do grupo “general” ou de qualquer parâmetro desse grupo, seria expandida, segundo a regra 3 da Tabela 3.3, para  $\${lrepo.general.catalogURL}$ . A utilização de referências parcialmente qualificadas aumenta tanto a facilidade de leitura quanto a de escrita de DDAs.

Referências externas, por sua vez, são expressas na forma  $@{comando}$ , em que *comando* corresponde a qualquer instrução executável por um interpretador de comandos específico. Em um ambiente UNIX, por exemplo, a referência  $@{ls ${user.home}}$  retorna a listagem (comando *ls*) de arquivos e diretórios contidos no diretório do usuário corrente (variável *user.home*). Conforme ilustra esse exemplo, além de possibilitar a declaração de referências de maneira bastante flexível, o mecanismo implementado pelo ambiente WhizKEY permite a combinação e, até mesmo, o aninhamento de referências. Para tanto, a resolução de dependências é processada em três passos, conforme ilustra o seguinte exemplo:

$$\begin{aligned}
 & @\{\${baseDir}\}/oai-listMetadataFormats.pl \${catalogURL}\} \\
 & \quad \downarrow_1 \\
 & @\{\${baseDir}\}/oai-listMetadataFormats.pl \${lrepo.general.catalogURL}\} \\
 & \quad \downarrow_2 \\
 & @\{sh/oai-listMetadataFormats.pl http://www.my-dl.org/oai.pl\} \\
 & \quad \downarrow_3 \\
 & \quad oai\_dc, oai\_marc
 \end{aligned}$$

No primeiro passo, referências parcialmente qualificadas são expandidas (no exemplo, a referência  $\${catalogURL}$  é expandida para  $\${lrepo.general.catalogURL}$ , com base no contexto em que foi declarada). No segundo passo, é efetuada a resolução de referências internas. Nesse passo, as referências  $\${baseDir}$ , correspondente ao diretório-base do ambiente WhizKEY, e  $\${lrepo.general.catalogURL}$ , correspondente à URL-base de um catálogo de metadados, são resolvidas para os valores “sh” e “http://www.my-dl.org/oai.pl”, respectivamente. No terceiro e último passo, ocorre a resolução de referências externas. Nesse caso, o mecanismo para resolução de dependências invoca o *script* “sh/oai-listMetadataFormats.pl”, que envia requisições OAI-PMH ao provedor correspondente à URL-base resolvida e retorna os formatos de metadados por ele suportados (itens “oai\_dc” e “oai\_marc”, no exemplo).

### 3.2.2.2 Restrições sobre Parâmetros

A fim de possibilitar uma verificação extensa da correção de um parâmetro, o ambiente WhizKEY implementa sete tipos de restrição que podem ser avaliadas sobre os valores que esse parâmetro pode assumir. A configuração de um parâmetro verifica todas as restrições definidas sobre ele de forma a garantir que satisfaça todas elas. Cada uma dessas restrições recebe uma lista de argumentos como entrada e produz uma avaliação do valor informado segundo os critérios por ela definidos, conforme descritos a seguir.

**Cardinalidade.** A restrição de cardinalidade é automaticamente adicionada a todos os parâmetros e verifica se o número de itens que compõem o valor informado de um parâmetro (ou o número de itens selecionados desse valor, no caso de parâmetros imutáveis) pertence ao intervalo fechado definido pelos atributos *min* e *max* desse parâmetro.

**Unicidade.** A restrição de unicidade recebe um conjunto de itens e verifica se cada item do valor informado é único nesse conjunto.

**Padrão.** A restrição de padrão recebe um conjunto de itens, utilizado para inicializar o valor do parâmetro sobre o qual está definida, e retorna sempre verdadeiro.

**Formato.** A restrição de formato recebe uma expressão regular e verifica se cada item do valor informado casa com essa expressão.

**Enumeração.** A restrição de enumeração recebe um conjunto de itens e verifica se todos os itens do valor informado pertencem a esse conjunto.

**Recurso.** A restrição de recurso recebe uma especificação de tipo de recurso – arquivo, diretório, URL ou provedor OAI – e verifica se cada item do valor informado corresponde a um recurso válido do tipo considerado. Arquivos e diretórios são verificados por meio de consultas ao sistema operacional, URLs são verificadas por meio de requisições HTTP e provedores OAI são verificados por meio de requisições OAI-PMH.

**Avaliação.** A restrição de avaliação recebe como entrada uma expressão lógico-aritmética e retorna o resultado da avaliação dessa expressão. Em geral, a especificação dessa restrição inclui uma referência para o próprio parâmetro em que é declarada, de forma que o valor atribuído a esse parâmetro, após resolvido pelo mecanismo de resolução de dependências, faça parte da expressão lógico-aritmética a ser avaliada. Por exemplo, a expressão lógica  $\$!\{interrequestGap\} > 0$  testa se o valor do parâmetro “interrequestGap”, que determina o intervalo entre requisições a um provedor OAI, é positivo.

### 3.3 Visualização: Geração de Interfaces Gráficas de Usuário

A camada de visualização é a responsável pela interface do ambiente WhizKEY com o usuário. A fim de auxiliar a instalação de bibliotecas digitais a partir de um arcabouço de software, WhizKEY implementa uma técnica de assistência a usuários comumente empregada para esse tipo de tarefa: um assistente. De acordo com Ekenstierna e Ekenstierna (2002), um assistente é uma forma especial de ajuda ao usuário, que o direciona ao longo de tarefas com estruturas bem definidas. Uma vez que devem operar sobre dados reais, os assistentes não são adequados para tutoriais nem devem ser considerados para assistência instrucional. Uma grande desvantagem de assistentes é sua linearidade. Entretanto, uma vez que muitas tarefas possuem um caráter linear, esse tipo de ferramenta pode ser bastante útil, sobretudo para tarefas infrequentes, como é o caso da instalação de bibliotecas digitais. O usuário médio enxerga o assistente como uma técnica muito conveniente devido, principalmente, a suas propriedades interativas. Por outro lado, esse usuário é inseguro quanto à sua capacidade de realizar tarefas similares sem o auxílio do assistente.

Outra técnica de assistência a usuários implementada pelo ambiente WhizKEY é a ajuda contextual. Ekenstierna e Ekenstierna (2002) afirmam que esse tipo de ajuda fornece ao usuário assistência imediata sobre um objeto específico e seu contexto, sem que ele precise abandonar sua área de trabalho corrente, o que é especialmente interessante para a apresentação de conceitos complexos envolvidos na tarefa de instalação.

Essas técnicas foram implementadas em uma interface gráfica de usuário (GUI) baseada na API Swing<sup>4</sup>, estendida por meio da implementação de alguns componentes visuais extras, como listas e árvores com caixas de marcação, que possibilitam a seleção de um subconjunto dos itens que compõem o valor de parâmetros multivalorados e imutáveis. A Figura 3.6 ilustra a organização da janela principal dessa interface.



Figura 3.6: Organização visual da interface gráfica

<sup>4</sup><http://java.sun.com/javase/6/docs/technotes/guides/swing>

Essa janela gerencia a exibição de três áreas distintas a partir de notificações recebidas da camada de controle, conforme discutido na Subseção 3.4.2. De acordo com a Figura 3.6, na área central, um painel apresenta os diferentes passos especificados no fluxo de instalação. Na área de status, a fim de apresentar uma visão global do fluxo de instalação, é exibida uma lista com todos os passos definidos, com destaque para o passo atualmente exibido na área central. Por fim, a área inferior apresenta uma barra com botões que possibilitam a navegação através dos diversos passos definidos.

### 3.3.1 Painel de Navegação

O painel de navegação possui três botões de comando que disparam transições no fluxo de instalação em execução. Essas transições possibilitam um caminhamento bidirecional através dos diversos passos definidos pelo fluxo, bem como sua interrupção a qualquer momento. Os rótulos desses botões, bem como sua disponibilidade, são atualizados a partir de notificações da camada de controle com base na satisfação de regras definidas sobre o passo corrente, discutidas na Subseção 3.4.1.

A tentativa de interrupção de uma instalação em andamento alerta o usuário para o fato de que toda a configuração por ele definida até o momento será perdida a menos que a instalação seja conduzida até o fim. A execução das operações de instalação somente ao final do processo permite o cancelamento de uma instalação sem a necessidade de que sejam desfeitas as operações realizadas até o momento do cancelamento.

### 3.3.2 Painéis de Configuração

O fluxo de instalação é organizado em três etapas:

- uma etapa *preparatória*, em que são colhidas informações sobre o sistema a ser instalado e sobre a tarefa de instalação a ser executada (instalação propriamente dita, atualização ou desinstalação de um sistema);
- uma etapa *intermediária*, em que os blocos associados aos diversos componentes especificados no DDA são configurados;
- uma etapa *final*, em que uma biblioteca digital é devidamente instalada com base nas instruções de preparação especificadas para os diversos componentes e nas configurações definidas ao longo do fluxo para os blocos associados a cada componente.

A cada passo definido no fluxo de instalação corresponde um elemento de interface, chamado *painel de configuração*. A camada de visualização implementa dois tipos básicos de painel: *painel de sistema* e *painel de usuário*. Painéis de sistema são utilizados na apresentação dos passos executados nas etapas preparatória e final, os quais devem ser obrigatoriamente especificados no DDA. Painéis de usuário, por sua vez, são utilizados na apresentação dos passos executados na etapa intermediária do fluxo de instalação. Ao contrário de painéis de sistema, a especificação de painéis de usuário é opcional.

Na etapa preparatória, são apresentados três painéis de sistema:

- *Painel de boas-vindas* (Figura 3.11), responsável por introduzir a tarefa a ser executada, apresentando ao usuário o propósito do ambiente de instalação;
- *Painel de seleção* (Figura 3.12), responsável por determinar a tarefa a ser executada (instalação, atualização ou desinstalação) e selecionar o sistema a ser instalado;
- *Painel de identificação* (Figura 3.13), responsável por configurações básicas (e.g., diretório-base da instalação) do sistema selecionado no passo anterior e que serão utilizadas no restante da tarefa.

Na etapa intermediária, são possivelmente apresentadas diversas instâncias de dois tipos de painel de usuário, cada uma relacionada à configuração de blocos associados a um conjunto de componentes especificados no DDA:

- *Painel fixo* (Figura 3.25), em que blocos associados aos componentes no passo correspondente ao painel são apresentados em uma lista com caixas de marcação, de modo a garantir que somente um bloco associado a cada componente possa ser configurado;
- *Painel não-fixo* (Figuras 3.14 e 3.18), em que blocos associados aos componentes no passo correspondente ao painel são apresentados em uma lista simples, o que permite que diversos blocos associados a cada componente possam ser configurados.

Por último, a etapa final apresenta outros dois painéis de sistema:

- *Painel de confirmação* (Figura 3.29), responsável por apresentar um resumo da configuração realizada ao longo do fluxo de instalação para conferência por parte do usuário;
- *Painel de finalização* (Figura 3.30), responsável por apresentar o resultado da execução da tarefa de instalação, incluindo apontadores para os diversos blocos instalados.

A apresentação da configuração de blocos ao longo de vários painéis de usuário tem uma importância fundamental para o funcionamento do mecanismo para resolução de dependências (descrito na Subseção 3.2.2.1) implementado pelo ambiente WhizKEY. O funcionamento desse mecanismo é baseado em uma premissa de precedência, a saber, a de que uma dependência somente é válida se o valor referenciado tiver sido previamente configurado. Dessa forma, para que uma dependência entre componentes distintos possa ser estabelecida, é necessário que a configuração dos blocos associados ao componente referenciador seja apresentada em um painel de usuário localizado posteriormente àquele responsável pela apresentação da configuração dos blocos associados ao componente referenciado (e.g., blocos correspondentes a serviços que operam sobre catálogos de metadados devem ser configurados posteriormente aos blocos correspondentes a esses catálogos, a fim de que dependências entre suas configurações possam ser devidamente estabelecidas).

### 3.3.3 Diálogos de Configuração

Em painéis não-fixos, as operações de inicialização e exclusão de blocos são acionadas clicando-se nos botões de comando apropriados, localizados à direita da lista de blocos (na Figura 3.24, botões “Add” e “Remove”, respectivamente). No caso de painéis fixos, essas operações são acionadas marcando-se e desmarcando-se um bloco na lista de blocos (Figura 3.28), respectivamente. A operação de modificação é acionada de maneira idêntica em ambos os painéis, clicando-se no botão de comando apropriado (botão “Configure”), também localizado à direita das listas de blocos. A disponibilidade dos botões de comando associados às operações de modificação e exclusão é determinada pela seleção de um bloco da lista; a disponibilidade do botão de inicialização, por sua vez, é determinada por regras especificadas no DDA sobre o passo corrente, conforme discutido na Subseção 3.4.1.

A configuração de um bloco é realizada em uma caixa de diálogo independente da janela principal da interface. Essa caixa de diálogo é utilizada tanto para a inicialização de um novo bloco quanto para a modificação de um bloco existente e apresenta uma interface especialmente desenvolvida para a configuração dos diversos parâmetros associados ao bloco. No caso da operação de modificação, a fim de possibilitar que as alterações efetuadas sobre a configuração de um bloco possam ser eventualmente canceladas pelo usuário, essa caixa de diálogo é inicializada com um clone do bloco em questão, em lugar do próprio bloco. Caso a ação de modificação seja confirmada, o bloco original é automaticamente atualizado a partir da configuração do bloco clonado.

A caixa de diálogo para inicialização de um novo bloco é apresentada em duas etapas. Uma vez que um passo pode tratar da configuração de blocos associados a diferentes componentes, a primeira dessas etapas, chamada *configuração de primeira ordem*, inicia-se pela escolha do componente a ser utilizado para a criação de um novo bloco (Figura 3.19). Uma vez escolhido esse componente, o diálogo de configuração de primeira ordem apresenta a configuração de parâmetros-chave a ele associados, i.e., parâmetros dos quais outros parâmetros associadas ao componente dependem (Figura 3.20). A visibilidade de tais parâmetros é especificada de modo que sua exibição seja limitada a uma única vez.

Concluída a primeira etapa, é iniciada a *configuração de segunda ordem*, a qual também é aplicável à operação de modificação de um bloco. Nessa etapa, é apresentada a configuração dos parâmetros associados a todos os parâmetros visíveis contidos no componente associado ao bloco selecionado, incluindo parâmetros de visibilidade limitada, cujos valores foram previamente informados na configuração de primeira ordem – na configuração de segunda ordem, esses parâmetros não podem ser alterados; sua exibição possui um caráter meramente informativo. A configuração desses parâmetros é organizada com base nos grupos aos quais pertencem (Figura 3.23). Uma lista à esquerda exhibe todos os grupos contidos no componente associado ao bloco sendo configurado. Clicando-se em um item dessa lista, a configuração dos parâmetros do grupo correspondente a esse item é exibida no painel à direita.

Analogamente à distribuição da configuração de blocos ao longo de vários painéis, o que permite o estabelecimento de dependências entre componentes, a divisão do diálogo de confi-

guração de um bloco em etapas distintas possibilita o estabelecimento de dependências dentro de um mesmo componente. Um primeiro nível de dependências é estabelecido entre parâmetros cujos valores são determinados nas configurações de primeira e segunda ordens e um segundo nível é estabelecido, ao final da exibição do diálogo de configuração, entre parâmetros invisíveis, cujos valores são determinados automaticamente, e parâmetros cujos valores foram determinados nas configurações de primeira ou segunda ordens.

A configuração de parâmetros nas caixas de diálogo de configuração é apresentada em uma interface especialmente desenvolvida para esse fim. Essa interface gera, dinamicamente, um elemento apropriado para cada parâmetro a ser configurado, de acordo com os atributos relacionados a estrutura, cardinalidade e mutabilidade desse parâmetro, conforme especificado na Tabela 3.4 e ilustrado na Figura 3.7.

Tabela 3.4: Regras para geração dinâmica de elementos de interface

Estrutura	Cardinalidade	Mutabilidade	Elemento	Figura
escalar	–	–	texto ou seletor de arquivos <sup>5</sup>	3.7(a)-3.7(b)
linear	monovalorado	–	caixa de seleção	3.7(c)
linear	multivalorado	mutável	lista simples	3.7(d)
linear	multivalorado	imutável	lista com caixas de marcação	3.7(e)
hierárquica	–	mutável	árvore simples	3.7(f)
hierárquica	–	imutável	árvore com caixas de marcação	3.7(g)

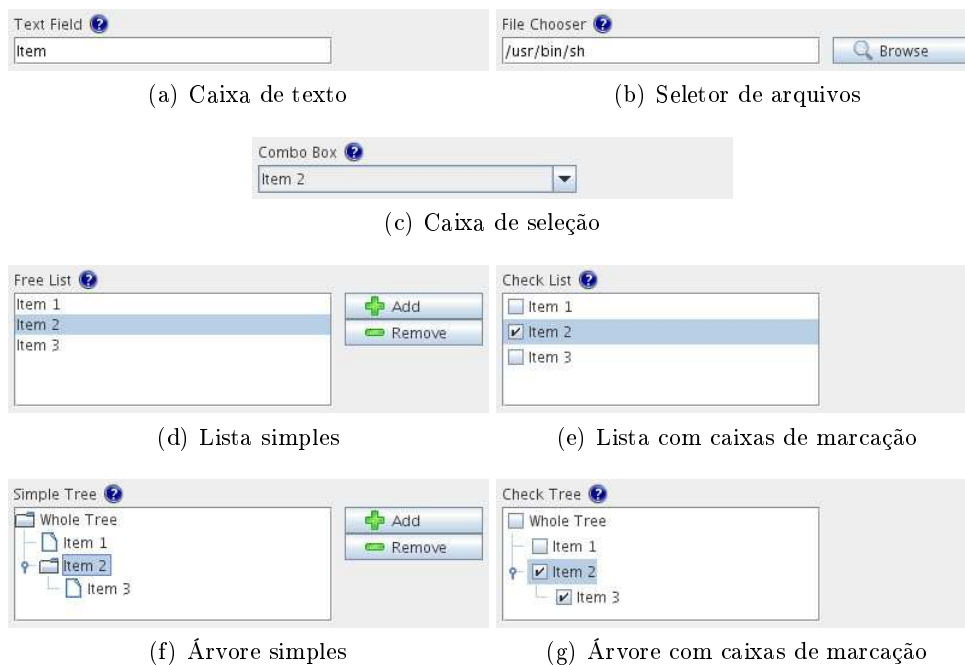


Figura 3.7: Elementos de interface para configuração de parâmetros

<sup>5</sup>Parâmetros com restrições de recurso do tipo arquivo ou diretório são exibidos como um seletor de arquivos; caso contrário, são exibidos como uma caixa de texto.

A exibição de cada parâmetro também inclui um mecanismo para apresentação da descrição desse parâmetro, bem como para notificação de eventuais erros durante sua configuração. Esse mecanismo é implementado como uma ajuda contextual a fim de que o usuário tenha acesso direto à semântica de um parâmetro qualquer, bem como receba um retorno imediato acerca da correção do valor sendo atribuído a esse parâmetro. A Figura 3.8 ilustra o funcionamento desse mecanismo.

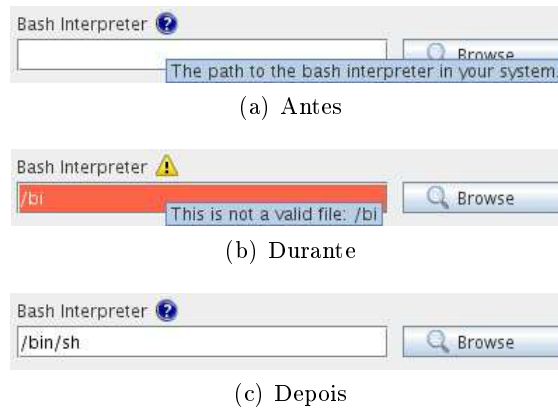


Figura 3.8: Ajuda contextual na configuração de parâmetros

A Figura 3.8 ilustra o exemplo de configuração de um parâmetro correspondente ao caminho para um interpretador de comandos. Inicialmente (Figura 3.8(a)), a semântica desse parâmetro é exibida como uma dica na interface quando o usuário passa o cursor do *mouse* sobre o ícone de interrogação. Ao iniciar a configuração desse parâmetro, as diversas restrições a ele associadas são verificadas. A primeira dessas restrições a ser violada provoca uma alteração na interface (Figura 3.8(b)): o elemento muda de cor e o ícone de ajuda é substituído por um ícone de erro. Ao passar o *mouse* sobre esse ícone, o usuário é informado acerca da violação cometida em relação uma das restrições adicionadas ao parâmetro sendo configurado (no exemplo, ao informar um caminho inexistente, uma restrição do tipo recurso é violada). Corrigido o erro, o elemento de interface volta a seu estado normal (Figura 3.8(c)). Todas as restrições definidas na Subseção 3.2.2.2 podem ser utilizadas na validação de parâmetros. Tanto a mensagem descritiva desses parâmetros quanto as mensagens de erro associadas aos diversos tipos de restrição podem ser personalizadas no DDA.

Além de oferecer ao usuário um retorno imediato sobre a correção de parâmetros, o mecanismo de verificação de restrições garante a correção da configuração de um bloco como um todo. A violação de uma restrição bloqueia o botão de finalização de diálogos de configuração, de modo que uma configuração incorreta não possa ser produzida. Para notificar o usuário sobre parâmetros incorretamente configurados espalhados por diversos grupos, a lista de grupos é atualizada automaticamente de forma a refletir o estado das configurações dentro de cada grupo – grupos corretamente configurados possuem um ícone verde; grupos em que alguma restrição foi violada possuem um ícone vermelho.



## 3.4 Controle: Gerenciamento do Fluxo de Instalação

A camada de controle implementada pelo ambiente WhizKEY tem duas atribuições principais:

1. Intermediar alterações à camada modelo requisitadas pela camada de visualização;
2. Gerenciar o fluxo de instalação de sistemas.

A primeira dessas atribuições é implementada internamente à camada de visualização – uma simplificação bastante comum em relação ao padrão MVC originalmente proposto – a fim de tratar eventos gerados por interações do usuário. Esses eventos podem provocar alterações no estado da camada modelo, as quais devem ser devidamente validadas pelos mecanismos de verificação de restrições (Subseção 3.2.2.2) e de gerenciamento de fluxo.

O gerenciamento do fluxo de instalação de sistemas constitui a segunda atribuição da camada de controle e envolve, basicamente, a definição de uma seqüência de passos e um conjunto de regras sobre esses passos. As subseções seguintes tratam dessas questões.

### 3.4.1 Passos de Configuração

A cada um dos painéis definidos na Subseção 3.3.2 corresponde um passo. A especificação dos diversos passos, analogamente organizados em *passos de sistema* e *passos de usuário* (*system* e *user steps* no DDA da Figura 3.4), permite personalizar as mensagens exibidas pelos painéis correspondentes na camada de apresentação.

Além de especificações relacionadas à apresentação, passos de usuário especificam também restrições de cardinalidade sobre o número de blocos que podem ser configurados em cada passo. Essa restrição complementa aquela definida sobre componentes (que limita o número de blocos associados a cada componente) e é utilizada pelo mecanismo de gerenciamento do fluxo de instalação, conforme discutido na subseção seguinte.

### 3.4.2 Fluxo de Instalação

O fluxo de instalação de um sistema é determinado por três fatores:

1. A seqüência de passos de usuário especificada no DDA<sup>6</sup>;
2. As restrições de cardinalidade associadas a cada um dos passos especificados e a cada um dos componentes a eles associados;
3. O modo de operação do ambiente WhizKEY (*instalação*, *atualização* ou *desinstalação*).

A seqüência de passos determina a ordem de apresentação dos painéis correspondentes e, por conseguinte, a ordem em que os blocos associados aos componentes de cada passo são configurados. Essa ordem é importante para assegurar a premissa de precedência sobre a qual foi construído o mecanismo para resolução de dependências apresentado na Subseção 3.2.2.1.

---

<sup>6</sup>A seqüência de passos de sistema é predefinida e, portanto, não influi no fluxo.

As restrições de cardinalidade sobre o número de blocos permitido para cada componente em um passo e o número total de blocos nesse passo funcionam como um semáforo no fluxo de instalação, bloqueando-o e liberando-o à medida que essas restrições são violadas ou satisfeitas, respectivamente. Na camada de visualização, o estado desse semáforo é refletido no botão de avanço no painel de navegação (Subseção 3.3.1). Conjuntamente ao mecanismo de verificação de restrições, que garante a correção de um bloco com base nas restrições definidas pelo componente a ele associado, o mecanismo de gerenciamento do fluxo de instalação garante a correção da configuração como um todo, assegurando-se da existência de um conjunto correto de blocos em cada passo definido.

Finalmente, o modo de operação do ambiente WhizKEY determina a disponibilidade de um determinado passo ou, mesmo, sua exibição. No modo de instalação, são apresentados painéis correspondentes a todos os passos definidos. No modo de atualização, a painel de identificação é exibido somente para leitura, de modo a impedir que configurações básicas do sistema sejam modificadas. Por fim, no modo de desinstalação, nem o painel de identificação nem quaisquer painéis de usuário são exibidos; nesse caso, o fluxo dá um salto do passo de seleção para o passo de confirmação (e vice-versa), onde é retomado normalmente.

As notificações da camada de controle à camada de visualização são feitas por meio de uma combinação de mensagens que denotam os diversos estados que o fluxo de instalação pode assumir, conforme descrito na Tabela 3.5.

Tabela 3.5: Mensagens de atualização do fluxo de instalação

	<b>Mensagem</b>	<b>Descrição</b>
1	INIT	O fluxo de instalação atingiu seu início
2	MODE	O modo de operação do ambiente WhizKEY foi alterado
3	RELOAD	O sistema selecionado foi alterado
4	STEP	O passo corrente foi alterado
5	STOP	O passo corrente foi bloqueado
6	GO	O passo corrente foi desbloqueado
7	CONFIRM	O fluxo de instalação atingiu o passo de confirmação
8	EXIT	O fluxo de instalação atingiu o passo de finalização

A mensagem INIT desabilita o botão de retrocesso. A mensagem MODE atualiza a lista de passos no painel à esquerda da janela principal e RELOAD provoca uma atualização em todos os painéis de modo a refletirem a configuração do novo sistema selecionado. A mensagem STEP indica qual painel deve ser exibido na área central, de acordo com os fatores considerados pelo mecanismo de gerenciamento do fluxo. As mensagens STOP e GO desabilitam e habilitam o botão de avanço, respectivamente. A mensagem CONFIRM altera o rótulo do botão de avanço a fim de indicar que a instalação está pronta para ser executada. Finalmente, a mensagem EXIT desabilita os botões de navegação e altera o rótulo do botão de encerramento para denotar o fim da execução do ambiente WhizKEY.

## 3.5 Persistência: Preparação e Configuração de Software

A camada de persistência é a responsável pela preparação do software correspondente aos diversos componentes do arcabouço de software subjacente, com base no estado dos blocos configurados ao final do fluxo de instalação. É ela também a responsável por prover à camada modelo serviços de carga e salvamento das configurações associadas aos diversos blocos.

### 3.5.1 Preparação de Software

Conforme discutido na Subseção 3.2.2, o estado no ciclo de vida de um bloco ao final do fluxo de instalação determina as ações a serem executadas pela camada de persistência para a preparação do software correspondente a esse bloco.

Blocos no estado NEUTRO não sofrem qualquer ação. As ações executadas sobre blocos nos demais estados são especificadas no DDA para os componentes em que se baseiam esses blocos. Para cada um desses possíveis estados, são especificados dois conjuntos de instruções a serem executados antes e depois do salvamento das configurações de um bloco, conforme exemplificado na listagem da Figura 3.9 para o estado INSTALAÇÃO.

```
<pre>
  cd ${system.general.baseDir};
  tar zxvf ${templateDir}/OAI-XMLFile-2.1.tar.gz;
  cd OAI-XMLFile-2.1/XMLFile;
  cp -r template ${!general.catalogId};
</pre>
<pos>
  ${system.general.baseDir}/ODL-DBUnion-1.2/DBUnion/${library.general.libraryId}/harvest.pl
</pos>
```

Figura 3.9: Exemplo de instruções de preparação para um componente

No exemplo da Figura 3.9, o primeiro conjunto (*pre*) contém instruções relacionadas à preparação do ambiente que irá receber o software correspondente ao bloco (um catálogo de metadados baseado no componente “lrepo” do DDA da Figura 3.4), como o desempacotamento (comando *tar*) de diversos artefatos que compõem esse software e a criação (comando *cp*) do diretório-base para o seu armazenamento. O segundo conjunto (*pos*) contém instruções de pós-instalação, como a reinicialização de serviços do sistema instalado (*script harvest.pl*).

Assim como instruções embutidas utilizando-se a sintaxe `@{comando}`, as instruções para preparação do software correspondente a um bloco podem incluir qualquer código executável pelo interpretador de comandos do sistema operacional. A escolha dessas instruções, por outro lado, tem implicações na portabilidade do ambiente WhizKEY. Embora a implementação do ambiente seja portátil, as instruções especificadas no DDA podem não ser. Para o caso de arcabouços de software desenvolvidos para uma plataforma específica, isso não constitui um problema. Para arcabouços desenvolvidos para múltiplas plataformas, entretanto, deve haver uma preocupação no sentido de tornar essas instruções portáveis, ou alternativamente, produzir DDAs específicos para cada plataforma onde o arcabouço possa ser instalado.

### 3.5.2 Configuração de Software

A proposta do ambiente WhizKEY de modelar uma visão do arcabouço subjacente traz um desafio adicional para a camada de persistência: mapear parâmetros, divisões arbitrárias da visão modelada, para parâmetros de configuração do arcabouço subjacente. Esse mapeamento pode envolver a persistência da configuração de um bloco em diferentes documentos de configuração e, reciprocamente, um documento pode encerrar configurações de blocos distintos.

Para tratar esse problema, o ambiente WhizKEY implementa um mecanismo que funciona como um armazém para acesso unificado a diversos documentos de configuração. Esse acesso é encapsulado por um mecanismo auxiliar, chamado *fonte*, que funciona como um ponto de acesso para leitura e escrita dos diversos parâmetros, conforme ilustra a Figura 3.10.

```
<sources>
  <source>
    <addr>${xmlfileConfig}</addr>
    <path>/xmlfile/archiveId</path>
  </source>
  <source>
    <addr>${dbunionConfig}</addr>
    <path>/odlunion/archive[identifiser='${catalogId}']/identifiser</path>
  </source>
  ...
```

Figura 3.10: Exemplo de fontes associadas a um parâmetro

O exemplo da Figura 3.10 apresenta duas fontes (*sources*), cada uma composta por dois elementos: o endereço (*addr*) de um documento XML e uma expressão XPath (*path*) que univocamente localiza um parâmetro de configuração nesse documento. Esse endereço é utilizado para a recuperação de um documento do armazém de documentos a fim de que possam ser processadas requisições de leitura ou escrita sobre ele. Caso o documento referenciado não exista, o armazém de documentos cria um documento vazio associado ao endereço informado.

A cada parâmetro persistente podem ser associadas diversas fontes, sendo que apenas a primeira é utilizada para leitura<sup>7</sup>, enquanto todas são utilizadas para escrita. A operação de leitura retorna uma lista de nodos a partir do documento XML e da expressão XPath especificados em uma fonte e é utilizada para instanciar o valor associado ao parâmetro que contém a fonte. A cardinalidade dessa lista equivale à cardinalidade do valor instanciado. A operação de escrita utiliza as diversas fontes associadas a um parâmetro para persistir o valor a ele associado nos documentos correspondentes a essas fontes. Para tanto, é utilizada uma heurística simples: a seqüência de passos (nodos separados por '/') na expressão XPath de uma fonte é percorrida até o penúltimo passo<sup>8</sup>, que corresponde a um nodo-alvo; ao nodo-alvo, são acrescentados tantos nodos quantos forem os itens contidos no valor, nomeados pelo valor do último passo na expressão XPath.

<sup>7</sup> Assume-se que as diversas manifestações de um parâmetro de configuração sejam consistentes.

<sup>8</sup> Caso não exista um caminho correspondente, a expressão XPath é utilizada para criá-lo.

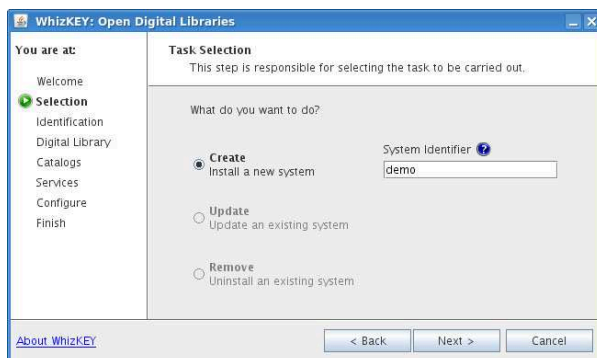
### 3.6 Exemplo de Uso

Nesta seção, demonstramos, passo a passo, um exemplo de uso do ambiente WhizKEY para instalação de uma biblioteca digital funcional a partir do arcabouço ODL. Ao final da execução do ambiente, a biblioteca construída possuirá um serviço de busca (instância do componente IRDB) operando sobre dois catálogos de metadados (um catálogo local, instância do componente OAI-XMLFile – um provedor de dados XML – e um catálogo remoto), acessíveis através de um mediador (instância do componente DBUnion).



No painel de boas-vindas, o ambiente WhizKEY descreve seu propósito e indica que executará sobre o arcabouço ODL.

Figura 3.11: Boas-vindas



No painel de seleção, o usuário indica que pretende instalar um novo sistema, identificado por “demo”.

Figura 3.12: Seleção da tarefa de instalação

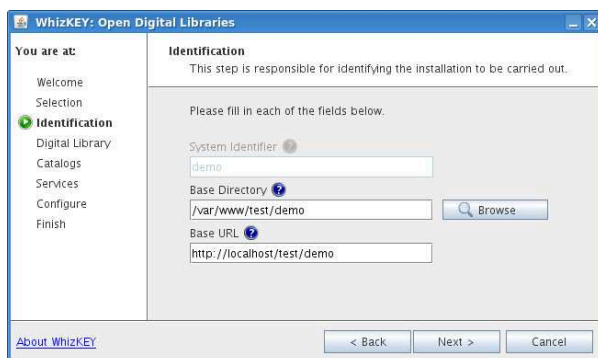


Figura 3.13: Identificação de um sistema

No painel de identificação, o sistema recém-criado é configurado. Essas configurações servirão de base para o restante do fluxo de instalação.

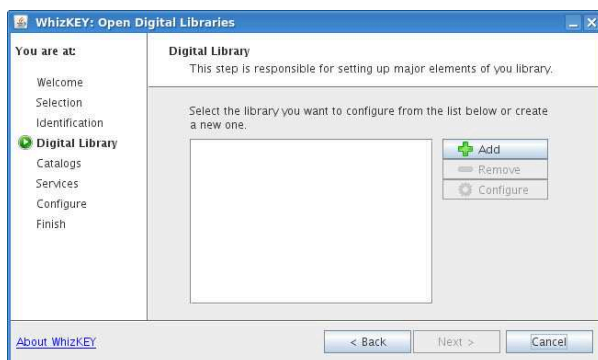


Figura 3.14: Configuração de uma biblioteca (1)

O painel seguinte – um painel de usuário – é destinado à configuração de bibliotecas digitais. De acordo com a especificação do arcabouço ODL, uma, e somente uma biblioteca deve ser criada. O botão de avanço é desabilitado até que essa restrição seja verificada.

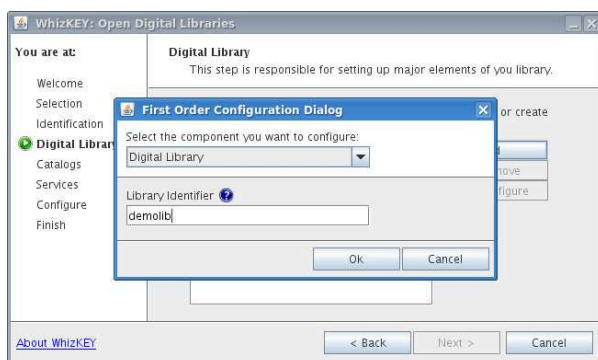


Figura 3.15: Configuração de uma biblioteca (2)

Como ocorre com todos os blocos, a configuração de uma biblioteca digital é dividida em duas etapas. Na configuração de primeira ordem, iniciada quando o usuário requisita a criação de uma nova biblioteca digital, clicando no botão apropriado, um diálogo lhe é apresentado para a configuração de parâmetros-chave dessa biblioteca, como seu identificador.

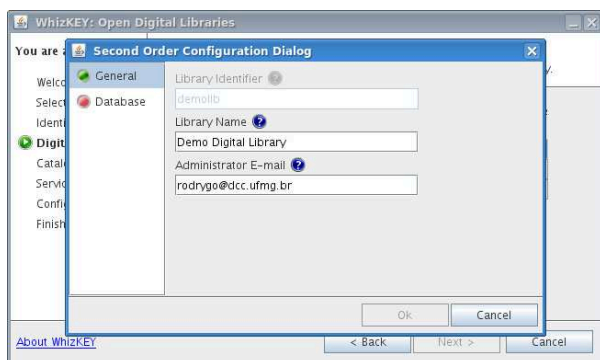


Figura 3.16: Configuração de uma biblioteca (3)

Na configuração de segunda ordem, são configurados diversos parâmetros associados à biblioteca, como seu nome e o endereço de e-mail de seu administrador.

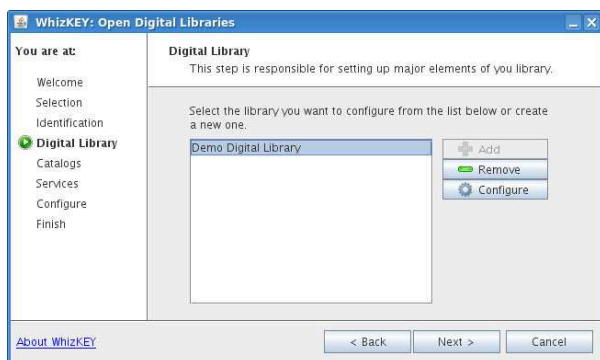


Figura 3.17: Configuração de uma biblioteca (4)

Confirmada a configuração, a biblioteca criada é adicionada à lista de bibliotecas do painel e o botão de avanço é habilitado, permitindo que o usuário prossiga no fluxo de instalação.

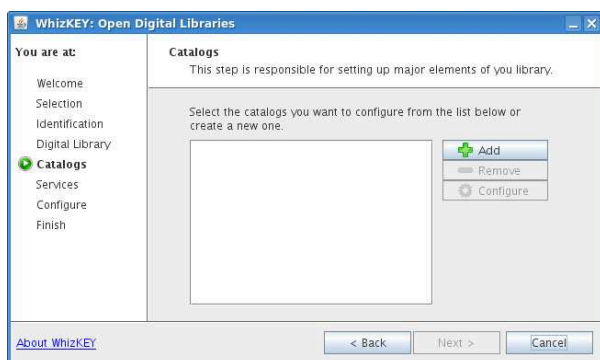


Figura 3.18: Configuração de catálogos (1)

O próximo passo de usuário definido pelo arcabouço ODL é responsável pela configuração de catálogos de metadados. Nesse passo, ao menos um catálogo deve ser criado para que o botão de avanço seja habilitado.

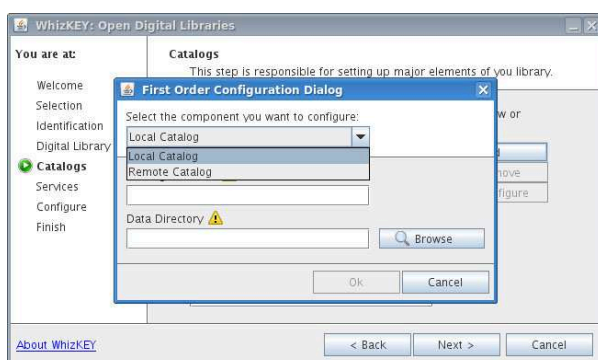


Figura 3.19: Configuração de catálogos (2)

Ao requisitar a criação de um novo catálogo, o usuário deve escolher em qual componente ele deve ser baseado: um catálogo local ou remoto. No exemplo, é feita a opção por um catálogo local.

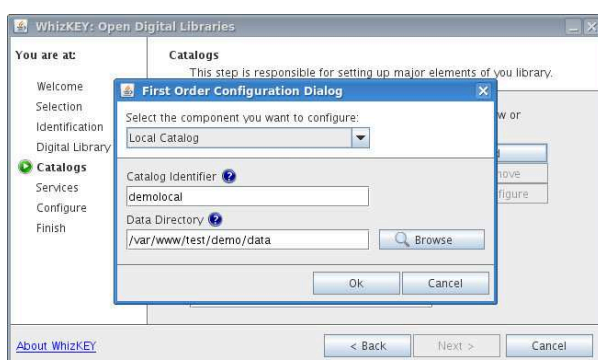


Figura 3.20: Configuração de catálogos (3)

Escolhido o tipo do catálogo, são configurados os parâmetros correspondentes. Além de um identificador, um catálogo local tem como parâmetro-chave o diretório-base onde serão armazenados os seus metadados.

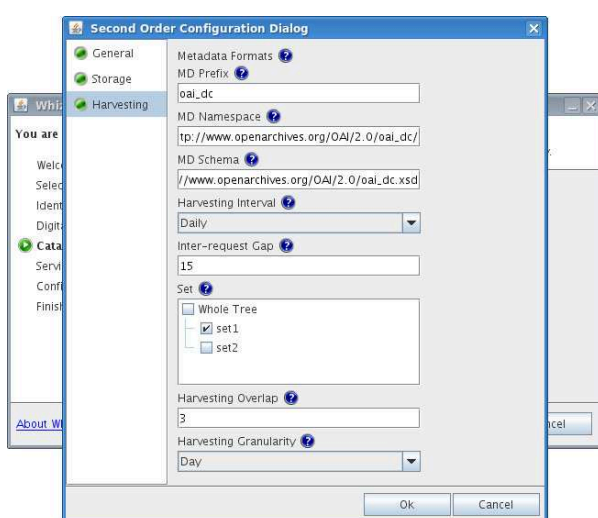


Figura 3.21: Configuração de catálogos (4)

Na configuração de segunda ordem, o usuário configura diversos parâmetros associados ao catálogo, como o formato de metadados que ele irá suportar, além de configurações relacionadas a como a biblioteca digital colherá metadados desse catálogo. Aqui, o usuário percebe um primeiro exemplo de dependência: o parâmetro “Set” lista a sub-árvore de diretórios contida no diretório-base informado durante a configuração de primeira ordem do catálogo.



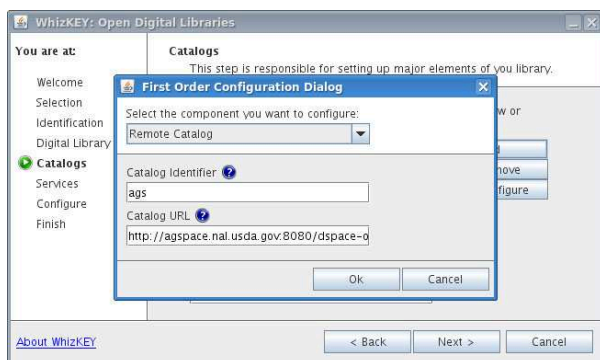


Figura 3.22: Configuração de catálogos (5)

Ainda no passo de configuração de catálogos, o usuário requisita a criação de um novo catálogo. Desta vez, porém, é escolhido um catálogo remoto. Como parâmetro-chave, o usuário informa a URL-base desse catálogo, um provedor OAI válido. Neste exemplo, o usuário informa a URL-base do provedor OAI da biblioteca digital do Departamento de Agricultura dos Estados Unidos, que provê conteúdo de pesquisa nas áreas de agricultura, alimentos e meio ambiente.

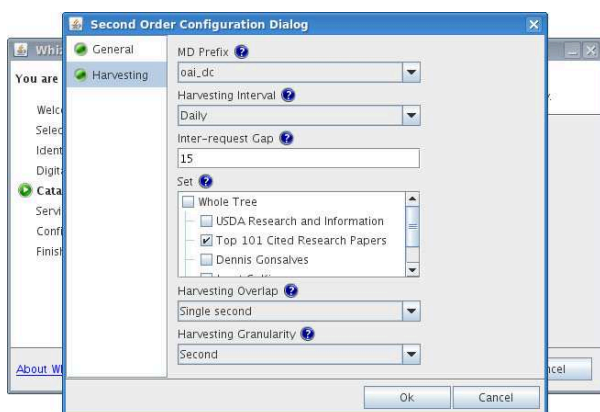


Figura 3.23: Configuração de catálogos (6)

A partir da URL-base informada na etapa anterior, são recuperadas diversas informações sobre o provedor correspondente a fim de auxiliar sua configuração, como os formatos de metadados por ele suportados e a estrutura de conjuntos segundo a qual são armazenados seus metadados.

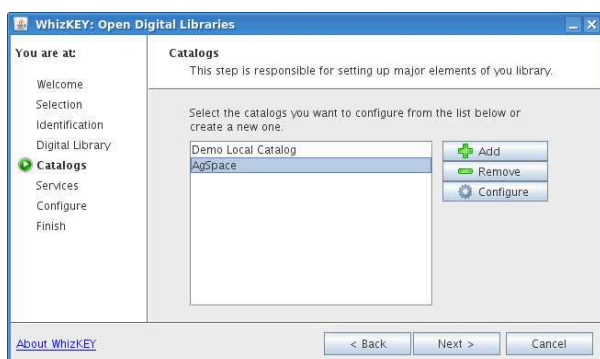


Figura 3.24: Configuração de catálogos (7)

Concluída a configuração dos catálogos, ambos são apresentados na lista do painel e o botão de avanço é novamente habilitado, permitindo o prosseguimento da tarefa de instalação.

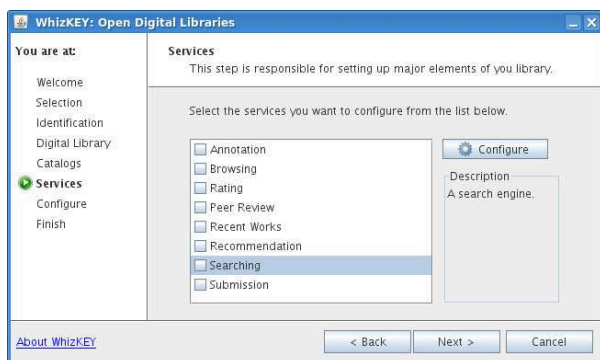


Figura 3.25: Configuração de serviços (1)

No último passo envolvendo esforços de configuração, o usuário deve configurar os serviços de sua biblioteca digital. Neste exemplo, o usuário seleciona o serviço de busca, cuja descrição é apresentada.



Figura 3.26: Configuração de serviços (2)

Assim como ocorreu com bibliotecas digitais e catálogos de metadados, a configuração de serviços também se dá em duas etapas. Para o serviço de busca, na configuração de primeira ordem, o usuário deve informar um identificador único.

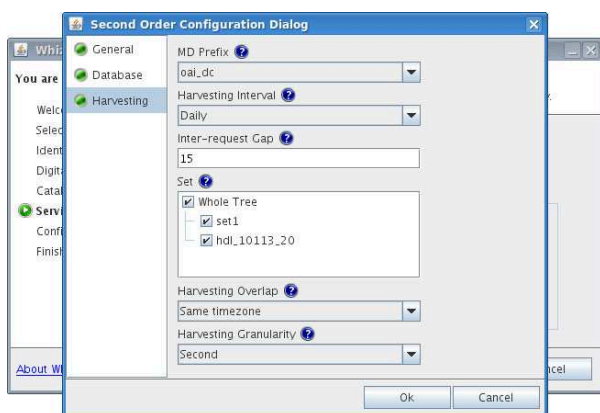


Figura 3.27: Configuração de serviços (3)

A seguir, o usuário deve configurar uma série de parâmetros associados ao serviço de busca, incluindo parâmetros dependentes de configurações já efetuadas ao longo do fluxo de instalação, como o formato e os conjuntos de metadados a serem indexados pelo serviço.

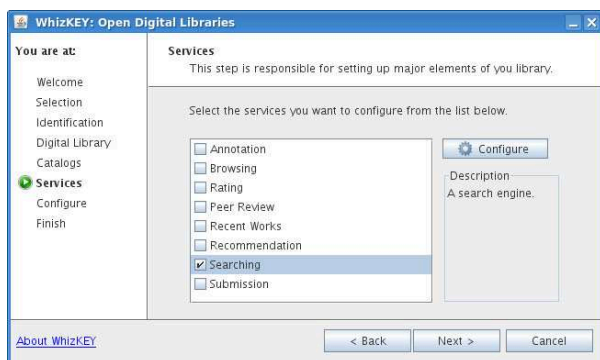


Figura 3.28: Configuração de serviços (4)

Ao confirmar a configuração, o usuário percebe que o serviço de busca aparece marcado na lista de serviços.

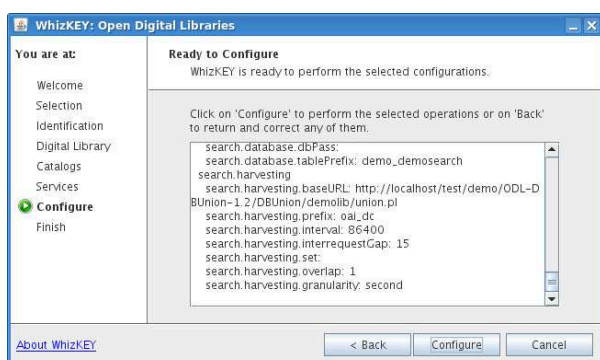


Figura 3.29: Resumo das configurações

Seguindo o fluxo de instalação, um resumo das configurações efetuadas pelo usuário lhe é apresentado. Caso queira corrigir alguma configuração, o usuário volta no ponto correspondente do fluxo de instalação, clicando no botão de retrocesso. Caso contrário, ele aciona o botão de configuração para iniciar a execução da instalação do sistema por ele configurado.

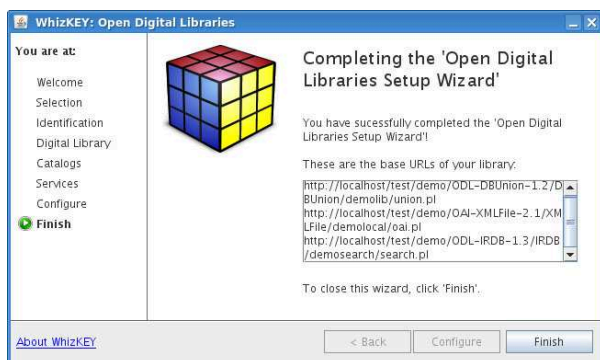
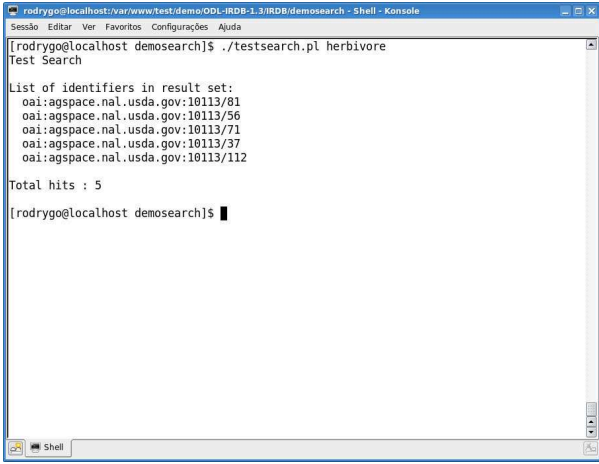


Figura 3.30: Resultado da instalação

Se a instalação é bem sucedida, o usuário é direcionado para o último passo, que lhe apresenta, além da confirmação do sucesso da tarefa executada, uma lista com os pontos de acesso ao sistema instalado. No caso do arcabouço ODL, são exibidas as URLs-base dos componentes instalados.



```
rodrygo@localhost:~/var/www/test/demo/ODL-IRDB-1.3/IRDB/demosearch - Shell - Konsole
Sessão Editar Ver Favoritos Configurações Ajuda
[rodrygo@localhost demosearch]$ ./testsearch.pl herbivore
Test Search

List of identifiers in result set:
oai:agspace.nal.usda.gov:10113/81
oai:agspace.nal.usda.gov:10113/56
oai:agspace.nal.usda.gov:10113/71
oai:agspace.nal.usda.gov:10113/37
oai:agspace.nal.usda.gov:10113/112

Total hits : 5
[rodrygo@localhost demosearch]$
```

Figura 3.31: Execução do sistema (1)

O arcabouço ODL oferece uma interface textual para utilização de seus componentes. Neste exemplo, o usuário efetua uma consulta ao serviço de busca recém-instalado pelo termo “herbivore”, para o qual cinco registros são retornados.

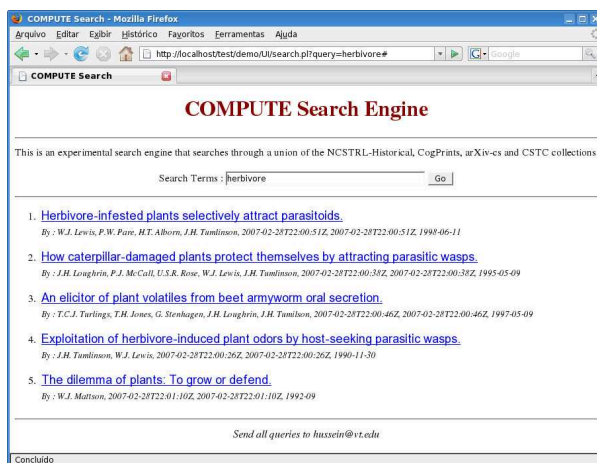


Figura 3.32: Execução do sistema (2)

Não incluído no arcabouço ODL mas oferecido como uma implementação de referência, a especificação desse arcabouço utilizada pelo ambiente WhizKEY provê um componente de interface gráfica para o serviço de busca. No exemplo, o usuário submete a consulta anteriormente feita ao serviço de busca por meio dessa interface gráfica e obtém, consistentemente, o mesmo conjunto de registros.

## Capítulo 4

# Avaliação do Ambiente

Do ponto de vista do usuário, a qualidade da interface e da interação determina a qualidade do sistema, e não seus algoritmos, arquitetura ou modelos de dados. Para ele, o sistema é a interface (Prates e Barbosa, 2003). É de extrema importância, para a produção de interfaces de qualidade, o conhecimento do comportamento do usuário diante do sistema. As ações, desejos, necessidades e frustrações dos usuários devem ser analisados de forma a garantir que o sistema os satisfaça da maneira mais eficiente possível. Uma das formas de se verificar a satisfação desses requisitos é por meio de testes com o sistema (Hix e Hartson, 1993).

O conceito de qualidade de uso mais amplamente utilizado é o de usabilidade. Essa qualidade está relacionada à facilidade e eficiência de aprendizado e de uso, bem como à satisfação do usuário (Nielsen, 1993). Neste capítulo, apresentamos diversos resultados obtidos a partir da avaliação de usabilidade de um protótipo do ambiente WhizKEY rodando sobre o arcabouço WS-ODL (Santos et al., 2008), a fim de validar a abordagem por ele implementada para a instalação de bibliotecas digitais, bem como identificar problemas nessa implementação.

### 4.1 Projeto Experimental

Para avaliar a usabilidade do ambiente WhizKEY, empregamos o teste de usabilidade, um método de avaliação empírica. Esse tipo de método envolve usuários em laboratório, de modo que o avaliador tenha um maior controle sobre o ambiente e as atividades do usuário. Assim, o avaliador consegue identificar problemas da interface que dificultam a interação, sem se preocupar com fatores externos, tais como interrupções (Prates e Barbosa, 2003).

#### 4.1.1 Preparação dos Testes

Testes em laboratório requerem um planejamento minucioso para que o avaliador tenha, de fato, controle sobre as condições de teste. Isso envolve se certificar de que as condições de teste são as mesmas para todos os participantes e de que o teste sendo executado permite a avaliação dos critérios desejados (Prates e Barbosa, 2003). A seguir, descrevemos os principais aspectos considerados, validados a partir da execução de testes-piloto.

**Seleção de usuários.** Como público-alvo do ambiente WhizKEY, consideramos projetistas de bibliotecas digitais, i.e., pessoas com conhecimentos sobre bibliotecas digitais (aqui consideradas como sistemas de informação) e de informática (interação básica através de GUIs) e que desejam construir uma biblioteca digital a partir de um arcabouço de software.

Inicialmente, contamos com um total de 20 voluntários, sendo 10 estudantes de Ciência da Computação e 10 de Ciência da Informação. Entretanto, durante os experimentos, verificou-se que 4 voluntários do grupo de Ciência da Informação não se encaixavam no perfil previamente definido. Nesse aspecto, acreditou-se que esses usuários possuíam um conhecimento maior de conceitos relacionados a bibliotecas digitais e de informática que se pôde observar de fato. A consequência imediata da quebra dessa premissa foi a de que os testes desses usuários não puderam ser gravados devido ao enorme tempo gasto e ao alto nível de dificuldade de interação por eles apresentado – em comum, todos eles necessitaram da intervenção de um especialista para que a tarefa pudesse ser continuada a fim de que todas as partes do protótipo pudessem ser avaliadas. Os problemas detectados nos testes com esses usuários, entretanto, foram considerados no desenvolvimento da versão atual do ambiente WhizKEY.

**Seleção de tarefas.** Para execução dos testes, foram elaboradas três tarefas representativas do processo de instalação de bibliotecas digitais, semelhantes à tarefa exemplificada na Seção 3.6. Essas tarefas foram organizadas em duas sessões independentes.

Na primeira sessão, foi executada uma tarefa simples, visando a familiarizar o usuário com o ambiente WhizKEY, e uma tarefa complexa, elaborada de modo a ser comparável à tarefa de instalação manual do arcabouço WS-ODL conduzida por Roberto (2006).

Na segunda sessão de testes com o protótipo do ambiente WhizKEY, queríamos avaliar melhor o entendimento dos usuários acerca das tarefas sendo executadas. Para tanto, elaboramos uma tarefa de instalação especificada em mais alto nível, na qual os usuários receberam uma descrição da biblioteca digital que deveriam construir em vez de um procedimento passo a passo mostrando como construí-la, como havia sido feito nos testes da primeira sessão.

**Geração de material para os testes.** Ao todo, foram gerados cinco documentos para auxiliar a execução dos testes e a coleta de dados:

- um questionário pré-teste para determinação do perfil dos usuários (Apêndice B.1);
- um texto descritivo do contexto do teste a ser realizado (Apêndice B.2);
- um roteiro com orientações sobre as tarefas a serem realizadas (Apêndice B.3);
- um questionário pós-teste para avaliação (Apêndice B.4);
- um roteiro para entrevista pós-teste (Apêndice B.5).

**Verificação de questões éticas.** Aos participantes, foram explicitadas as condições do teste, incluindo seus objetivos, a duração estimada e o tipo de dado a ser coletado. Além disso, foi-lhes garantido o anonimato de todas as informações obtidas durante o teste.

### 4.1.2 Execução dos Testes

A fim de avaliar a usabilidade do ambiente WhizKEY, conduzimos duas sessões de testes. Na primeira sessão, contamos com um total de 8 voluntários no papel de projetistas de bibliotecas digitais, sendo 4 estudantes de Ciência da Computação (CC) e 4 de Ciência da Informação (CI). Os testes consistiram na execução de duas tarefas de instalação,  $T_1$  e  $T_2$ , e no preenchimento de um questionário de avaliação. Ambas as tarefas exploram todos os elementos de interface do ambiente WhizKEY, tais como listas e seletores de arquivos.

A primeira e mais simples dessas tarefas,  $T_1$  (Apêndice B.3.1), destinada a familiarizar os usuários com o ambiente WhizKEY, consistia na modificação de alguns parâmetros de uma biblioteca digital previamente instalada. A segunda e mais complexa das tarefas,  $T_2$  (Apêndice B.3.2), envolvia a instalação, desde o início, de uma biblioteca digital completa. Uma vez que esse protótipo foi executado para a instalação de bibliotecas digitais baseadas no arcabouço WS-ODL, elaboramos essa segunda tarefa de modo que fosse comparável a outra tarefa, aqui denominada  $T_2^*$ , conduzida em um teste de instalação desse arcabouço via linha de comando, sem o auxílio de qualquer ferramenta de assistência (Roberto, 2006).

Na segunda sessão de testes, contamos com 8 novos voluntários, sendo 6 estudantes de Ciência da Computação (CC) e 2 de Ciência da Informação (CI). Nessa sessão, foi executada uma tarefa comparável às tarefas  $T_2$  e  $T_2^*$ , denominada  $T_3$  (Apêndice B.3.3). Entretanto, diferentemente das tarefas anteriores, apresentadas como um roteiro passo-a-passo da instalação a ser efetuada, a tarefa  $T_3$  foi elaborada com o objetivo de melhor avaliar o entendimento do usuário quanto à utilização do ambiente WhizKEY. Para tanto, em vez de uma lista de ações a serem executadas em direção à instalação de uma biblioteca digital, essa tarefa procurou descrever o objeto a ser instalado, ou seja, a própria biblioteca.

A condução dos testes incluiu um conjunto de passos a serem seguidos. Primeiramente, os usuários responderam ao questionário para identificação de seu perfil. Em seguida, foram-lhes apresentados o objetivo do teste e o contexto em que ele se insere, além da solicitação das permissões dos usuários para o registro e a utilização dos resultados obtidos nos testes. Posteriormente, foi entregue aos usuários o roteiro da tarefa que deveria ser realizada. Cada uma das tarefas elaboradas foi organizada como um conjunto de itens a serem executados para o cumprimento da tarefa. Solicitou-se que os itens listados fossem executados e, se possível, que comentários, dúvidas, críticas e sugestões fossem verbalizadas durante o teste<sup>1</sup>. Uma lista de acompanhamento foi entregue aos avaliadores presentes, de forma que os comentários e observações fossem por eles documentados. Durante todo o teste, os avaliadores não interferiram nas atividades ou forneceram qualquer tipo de ajuda aos usuários. Ao término da realização da tarefa fornecida, solicitou-se aos usuários que respondessem a um questionário de avaliação do sistema, bem como a uma entrevista para discussão de aspectos qualitativos relacionados ao sistema. Após essas atividades, o teste foi considerado concluído.

---

<sup>1</sup>Embora instruídos para que falassem, em nenhuma das sessões de teste (com exceção do teste-piloto) os usuários fizeram comentários durante a execução da tarefa, de modo que o tempo registrado para cada usuário não sofreu qualquer tipo de influência.

## 4.2 Análise dos Resultados

Para cada uma das tarefas executadas, foram medidos seu tempo total de execução e sua correção, essa última definida como o percentual de itens corretamente executados na tarefa. Esses resultados são sumarizados na Tabela 4.1 e analisados ao longo desta seção. Os valores obtidos por usuários especialistas também são mostrados.

Tabela 4.1: Tempo de execução e correção por tarefa

Usuários	Tempo de Execução ( <i>hh:mm:ss</i> )				Correção			
	$T_1$	$T_2$	$T_{2^*}$	$T_3$	$T_1$	$T_2$	$T_{2^*}$	$T_3$
CC (média)	00:07:01	00:11:26	01:24:00	00:15:23	0,98	0,99	0,83	0,95
CC (desvio)	00:01:11	00:04:15	00:16:58	00:06:28	0,04	0,02	0,07	0,06
CI (média)	00:11:05	00:18:33	01:35:30	00:21:09	1,00	0,95	0,62	0,81
CI (desvio)	00:03:33	00:02:08	00:00:42	00:01:02	0,00	0,09	0,10	0,04
Global (média)	00:09:03	00:15:00	01:29:45	00:16:49	0,99	0,97	0,72	0,92
Global (desvio)	00:03:17	00:04:55	00:11:51	00:06:06	0,03	0,06	0,14	0,09
Especialista	00:01:53	00:04:33	00:37:00	00:07:43	1,00	1,00	1,00	1,00

Os resultados obtidos foram analisados em conjunto com observações qualitativas, realizadas ao longo das sessões de teste, no contexto de um conjunto de fatores comumente considerados em avaliações desse tipo (Sharp et al., 2007)<sup>2</sup>:

1. *Eficiência de uso*, relacionada a quão bem o sistema desempenha aquilo a que se destina.
2. *Produtividade*, referente ao tempo gasto na realização de uma tarefa através do sistema.
3. *Utilidade*, relacionada à completude do sistema quanto ao conjunto de funcionalidades necessárias à realização de uma tarefa.
4. *Facilidade de aprendizado*, referente ao tempo e esforço necessários à utilização de uma determinada porção do sistema com determinado nível de competência e desempenho.
5. *Facilidade de uso*, relacionada ao esforço cognitivo para interagir com o sistema e também com o número de erros cometidos durante essa interação.
6. *Satisfação*, relacionada à avaliação subjetiva do sistema feita por seus usuários, incluindo emoções que possam surgir durante a interação, sejam elas positivas ou negativas.
7. *Segurança no uso*, referente ao grau de proteção de um sistema contra condições desfavoráveis para os usuários, incluindo a prevenção e a recuperação de erros.

Na primeira sessão de testes, o fator *eficiência de uso* foi medido com base nos valores de correção para a tarefa  $T_2$  em relação aos valores correspondentes para a tarefa  $T_{2^*}$ , conforme

<sup>2</sup>Outro fator tipicamente considerado, *flexibilidade*, não se aplica a esta avaliação, uma vez que a abordagem implementada pelo ambiente WhizKEY pressupõe uma ordem fixa de interação, própria de assistentes.



lista a Tabela 4.1. A comparação da abordagem guiada oferecida pelo ambiente WhizKEY (tarefa  $T_2$ ) à abordagem baseada em linha de comando (tarefa  $T_{2*}$ ) mostra que a correção das configurações é substancialmente aumentada (cerca de 34%, em média) pela utilização do ambiente WhizKEY (hipótese estatisticamente validada pelo teste- $t$  com  $\alpha = 0,05$ ). Isso se deve, principalmente, aos mecanismos de verificação de restrições e de resolução de dependências implementados pelo ambiente, conforme discutido no Capítulo 3.

Além do ganho quanto à correção, a instalação de componentes WS-ODL com o auxílio do ambiente WhizKEY é muito mais rápida (cerca de 500%, em média) que manualmente (hipótese estatisticamente validada pelo teste- $t$  com  $\alpha = 0,05$ ), o que é um forte indício do ganho de *produtividade* proporcionado pela utilização do ambiente.

Já o fator *utilidade* foi avaliado subjetivamente pelos participantes de ambas as tarefas e medido com base em uma escala bipolar de 5 pontos, variando de 1 (pior avaliação) a 5 (melhor avaliação). Na média, a *utilidade* do ambiente WhizKEY foi avaliada em 4,5.

Em conjunto, *eficiência de uso*, *produtividade* e *utilidade* atestam a efetividade do ambiente WhizKEY no auxílio à instalação de bibliotecas digitais baseadas no arcabouço WS-ODL. A tarefa  $T_3$  confirma esses resultados. Embora similar às tarefas anteriores, essa tarefa foi considerada bem mais difícil, demandando uma compreensão mais ampla por parte dos usuários. O tempo de execução e a correção da tarefa, entretanto, não foram significativamente afetados, o que atesta ainda mais a efetividade do ambiente WhizKEY no cenário considerado.

Como um ponto adicional, medimos a evolução dos usuários ao longo da tarefa  $T_3$ . O gráfico na Figura 4.1 mostra, para cada um dos 18 itens que compõem a tarefa, o tempo gasto por cada usuário até a execução daquele item. A partir do gráfico, podemos observar os pontos em que as curvas dos usuários mais divergem em relação à curva de um especialista (linha inferior), o que corresponde aos itens da tarefa com maior grau de dificuldade. No caso dessa tarefa, a criação de diretórios demandada por alguns componentes (itens 2 e 10 no gráfico), seguida pela instanciação do repositório Fedora (item 5), foram os itens mais demorados.

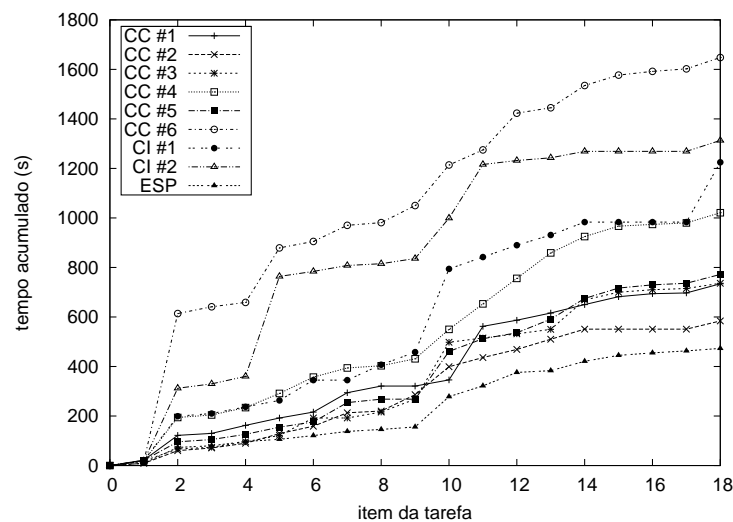


Figura 4.1: Evolução temporal da execução da tarefa

A *facilidade de aprendizado* do ambiente WhizKEY foi derivada a partir da Tabela 4.1. Para tanto, criamos duas medidas: desempenho e perícia. O desempenho é calculado como o número total de itens em uma tarefa dividido pelo tempo total de execução da tarefa. A perícia, por sua vez, é calculada como a razão entre o desempenho de cada usuário e o do especialista. A Tabela 4.2 mostra os valores para essas duas medidas de *facilidade de aprendizado*. O desempenho é medido em termos do número de itens configurados por minuto.

Tabela 4.2: Desempenho e perícia por tarefa

Usuários	Desempenho				Perícia			
	$T_1$	$T_2$	$T_{2^*}$	$T_3$	$T_1$	$T_2$	$T_{2^*}$	$T_3$
CC (média)	1,90	2,67	1,08	1,31	0,28	0,43	0,45	0,56
CC (desvio)	0,38	0,80	0,22	0,42	0,06	0,13	0,09	0,18
CI (média)	1,26	1,52	0,93	0,85	0,18	0,25	0,39	0,37
CI (desvio)	0,36	0,18	0,01	0,04	0,05	0,03	0,00	0,02
Global (média)	1,58	2,10	1,01	1,20	0,23	0,34	0,42	0,51
Global (desvio)	0,48	0,81	0,15	0,39	0,07	0,13	0,06	0,17
Especialista	6,90	6,15	2,41	2,33	1,00	1,00	1,00	1,00

A partir da Tabela 4.2, podemos observar que o desempenho é aumentado (cerca de 33%, em média) da tarefa  $T_1$  para a tarefa  $T_2$ . Nessa avaliação, consideramos todos os itens que compõem uma tarefa como sendo igualmente difíceis, o que é bastante razoável, uma vez que todos eles consistem da configuração de parâmetros de estrutura semelhante. A perícia também é aumentada (cerca de 49%, em média) da tarefa  $T_1$  para a tarefa  $T_2$ , o que poderia atestar a *facilidade de aprendizado* do ambiente. Essa hipótese, entretanto, foi rejeitada estatisticamente (teste- $t$  com  $\alpha = 0,05$ ), o que sugere que a tarefa  $T_1$  talvez não tenha sido suficiente para que os usuários se familiarizassem com o ambiente. Ainda com relação à *facilidade de aprendizado*, a Figura 4.2 mostra a evolução da perícia ao longo da tarefa  $T_3$ .

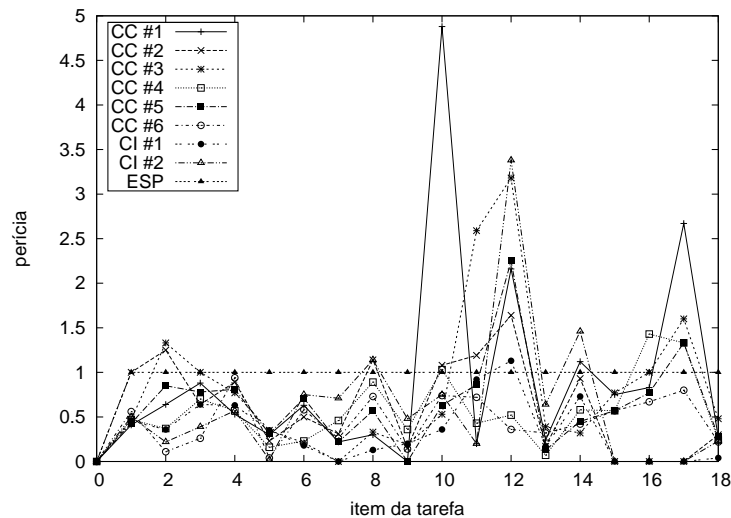


Figura 4.2: Evolução temporal da perícia

A partir do gráfico na Figura 4.2, observamos que a maioria dos usuários logo atinge um nível de perícia próximo ao do especialista (linha horizontal na ordenada 1), o que vemos como um indício da *facilidade de aprendizado* do ambiente. Podemos observar, também, que há picos nas curvas de alguns usuários, denotando valores de perícia até 5 vezes maior que a do especialista na execução de um dado item. Em sua grande maioria, esses picos correspondem a itens incorretamente executados ou, mesmo, não executados por parte dos usuários. Esses itens, entretanto, correspondem à configuração de parâmetros não essenciais à execução de uma biblioteca digital baseada no arcabouço WS-ODL, de forma que a execução da tarefa não foi comprometida. De qualquer forma, esses ganhos abruptos são contrabalançados pelo restante da execução, de modo que esses usuários possuem uma perícia global inferior à do usuário especialista, conforme esperado.

A partir do questionário preenchido pelos usuários ao final dos testes, medimos os fatores *facilidade de uso*<sup>3</sup> e *satisfação*, ambos com base em escalas bipolares de 5 pontos, variando de 1 (pior avaliação) a 5 (melhor avaliação). Na média, esses fatores receberam avaliações 4 e 4,25, respectivamente, para a primeira sessão. Na segunda sessão, esses fatores foram avaliados em 3,50 e 3,83, respectivamente. A pior avaliação desses fatores na segunda sessão reflete a maior dificuldade da tarefa  $T_3$  em relação às demais. Em geral, esses fatores foram pior avaliados pelos estudantes de Ciência da Informação, o que atribuímos à maior dificuldade por eles demonstrada em relação ao entendimento dos conceitos envolvidos. Como um todo, dada a complexidade das tarefas realizadas, consideramos bons esses resultados.

Por fim, o fator *segurança no uso* foi avaliado qualitativamente. Observações ao longo de todas as sessões de teste detectaram falhas graves na implementação do mecanismo de prevenção e recuperação de erros do protótipo do ambiente WhizKEY. A ajuda contextual, associada a todos os parâmetros, não era percebida pelos usuários, uma vez que nenhum elemento na interface explicitava a existência desse mecanismo. Além disso, o mecanismo de notificação de erros era incapaz de indicar visualmente a localização de um erro.

Esses e outros problemas detectados ao longo da segunda sessão de testes são listados na Tabela C.1 do Apêndice C. Essa avaliação serviu de base para a reformulação da interface gráfica do ambiente WhizKEY, conforme apresentada ao longo desta dissertação. Como principais melhorias, o estado dos botões no painel de navegação (Subseção 3.3.1) passou a refletir a correção global do sistema sendo instalado, por meio da verificação de restrições associadas a cada passo especificado no fluxo de instalação do arcabouço de software subjacente (Seção 3.4). Em um nível local, i.e., de modo a garantir a correção de cada parte do sistema individualmente, os diálogos de configuração (Subseção 3.3.3) foram completamente remodelados. Nesses diálogos, um novo mecanismo de ajuda contextual foi implementado a fim de oferecer uma resposta imediata à interação dos usuários durante a configuração de cada parâmetro do arcabouço.

---

<sup>3</sup>A *facilidade de uso* é tradicionalmente medida com base no número de erros cometidos pelos usuários durante a interação. Dada a dificuldade de se quantificar esse número em uma aplicação altamente interativa, como é o caso, optamos por recorrer à avaliação subjetiva por parte dos usuários.

## Capítulo 5

# Conclusões e Trabalhos Futuros

### 5.1 Revisão do Trabalho

Bibliotecas digitais têm importância destacada na educação, principalmente no que diz respeito à preservação e à disseminação do conhecimento humano. Nesse contexto, projetos que visem a fomentar a utilização de bibliotecas digitais por um público com necessidades e habilidades heterogêneas são estratégicos. Nos últimos anos, diversos arcabouços de software foram desenvolvidos a fim de possibilitar a construção de bibliotecas digitais personalizadas a partir de um alicerce comum. A maioria dos arcabouços desenvolvidos, entretanto, praticamente desconsidera a heterogeneidade de seus potenciais usuários, oferecendo-lhes uma única – e complexa – alternativa para a instalação de novas bibliotecas digitais.

Nesta dissertação, apresentamos WhizKEY – *Wizard-based block Ensemble Yielder*, um ambiente para auxiliar a instalação de bibliotecas digitais a partir de arcabouços de software. O ambiente implementa uma arquitetura em camadas com vistas a facilitar sua manutenção e extensão. Compõem essa arquitetura uma camada modelo, flexível o suficiente para a representação de diferentes arcabouços de software, suas interdependências e as restrições sobre sua configuração; uma camada de visualização, implementada como um assistente de instalação, responsável por guiar o usuário através de interfaces intuitivas, automaticamente geradas e com elementos que refletem o estado de correção das configurações produzidas; uma camada de controle, que implementa um mecanismo de gerenciamento de fluxo para tarefas de instalação, atualização e desinstalação de sistemas; e uma poderosa camada de persistência, que permite a utilização de qualquer *script* executável em linha de comando para a preparação do software subjacente, ao mesmo tempo que mantém uma estrutura unificada para acesso aos diversos documentos de configuração desse software.

Por meio de testes de usabilidade envolvendo participantes de dois grupos representativos de seus potenciais usuários, a saber, estudantes de Ciência da Computação e Ciência da Informação, um protótipo do ambiente WhizKEY foi avaliado segundo sete fatores de usabilidade. As avaliações atestaram a efetividade do ambiente no auxílio à tarefa de instalação de bibliotecas digitais a partir de um arcabouço de software, bem como detectaram uma série de problemas na implementação do protótipo, corrigidos na versão atual do ambiente.

## 5.2 Trabalhos Futuros

Como próximos passos deste trabalho, um ponto que precisa ser abordado é a formalização da especificação de arcabouços por meio da criação de um esquema para DDAs. Além disso, seria interessante elaborar uma meta-especificação que possibilite a geração de DDAs para arcabouços específicos por meio do próprio ambiente WhizKEY. Além de tornar a tarefa de especificação menos laboriosa, isso permitiria testar a generalidade do modelo implementado por meio da especificação de novos arcabouços. Um ponto de partida interessante é o arcabouço DSpace<sup>1</sup>, cuja arquitetura, diferentemente dos arcabouços ODL e WS-ODL, não se baseia em componentes de software.

Outro ponto importante a ser considerado é uma política clara para propagação de alterações quando da violação da premissa de precedência exigida pelo mecanismo de verificação de dependências implementado pela camada modelo. O retrocesso no fluxo de instalação para alteração de um valor pode produzir um estado inconsistente em parâmetros configurados subseqüentemente, de maneira análoga à violação de integridade referencial em SGBDs relacionais. Diferentemente desses sistemas, entretanto, o usuário precisa estar ciente da propagação a ser efetuada (e.g., se o novo valor deve ser propagado, se um valor nulo deve ser propagado ou se a alteração deve ser simplesmente bloqueada), o que envolve a implementação de uma solução eficaz – em termos de usabilidade – na camada de visualização.

Ainda na camada modelo, seria interessante estender o conceito de parâmetro de modo a permitir a existência de parâmetros complexos, i.e., parâmetros compostos por outros parâmetros. Esse tipo de abstração é útil na configuração de parâmetros do arcabouço subjacente representados por elementos XML compostos por outros elementos XML. Isso poderia ser feito, por exemplo, dividindo-se um item em partes menores e associando-se a cada parte um parâmetro contido no parâmetro complexo.

Na camada de controle, seria útil desenvolver um mecanismo para empacotamento e exportação de instalações a fim de permitir a execução de uma mesma instalação produzida através do ambiente WhizKEY em diferentes máquinas.

Por fim, uma nova sessão de testes poderia ser conduzida com a versão final do ambiente WhizKEY de forma a se anexarem novos registros às medidas realizadas com a versão anterior, além de se identificarem possíveis novos problemas a serem corrigidos em versões futuras. Além disso, seria interessante realizar uma avaliação comparativa da abordagem de assistência oferecida pelo ambiente WhizKEY e uma abordagem de interação não-guiada, tal qual a oferecida pela ferramenta BLOX.

---

<sup>1</sup><http://www.dspace.org>

# Referências Bibliográficas

- Albergaria, E. T. e Santos, R. L. T. (2006). Avaliação de interfaces de usuário para construção de bibliotecas digitais.
- 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.
- Burbeck, S. (1987). Applications programming in Smalltalk-80: How to use Model-View-Controller (MVC). Technical report, Softsmarts Inc., Palo Alto, CA, USA.
- Buschmann, F.; Meunier, R.; Rohnert, H.; Sommerlad, P. e Stal, M. (1996). *Pattern-oriented Software Architecture: a System of Patterns*. John Wiley & Sons, Inc., New York, NY, USA.
- Ekenstierna, M. e Ekenstierna, M. (2002). Evaluation of user assistance in graphical user interface software. Master's thesis, Department of Communication Systems at Lund Institute of Technology, Lund, Sweden.
- Eyambe, L. (2005). A digital library component assembly environment. Master's thesis, University of Cape Town, Cape Town, WC, South Africa.
- Eyambe, L. e Suleman, H. (2004). A digital library component assembly environment. In *Proceedings of the 2004 Annual Research Conference of the South African Institute for Computer Scientists and Information Technologists on IT Research in Developing Countries*, pp. 15–22, Stellenbosch, WC, South Africa. SAICSIT.
- Gonçalves, M. A. e Fox, E. A. (2002). 5SL: a language for declarative specification and generation of digital libraries. In *Proceedings of the 2nd ACM/IEEE-CS Joint Conference on Digital Libraries*, pp. 263–272, Portland, OR, USA. ACM Press.
- Gonçalves, M. A.; Fox, E. A.; Watson, L. T. e Kipp, N. A. (2004). Streams, Structures, Spaces, Scenarios, Societies (5S): A formal model for digital libraries. *ACM Transactions on Information Systems*, 22(2):270–312.

- Gonçalves, M. A.; Zafer, A. A.; Ramakrishnan, N. e Fox, E. A. (2001). Modeling and building personalized digital libraries with PIPE and 5SL. In *Proceedings of the 2nd DELOS Network of Excellence Workshop on Personalisation and Recommender Systems in Digital Libraries*, pp. 67–72, Dublin, Ireland.
- Gorton, D. (2007). Practical digital library generation into DSpace with the 5S framework. Master's thesis, Virginia Polytechnic Institute and State University, Blacksburg, VA, USA.
- Heineman, G. T. e Councill, W. T., editores (2001). *Component-based Software Engineering: Putting the Pieces Together*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.
- Hix, D. e Hartson, H. R. (1993). *Developing User Interfaces: Ensuring Usability through Product & Process*. John Wiley & Sons, Inc., New York, NY, USA.
- Kelapure, R. D. (2003). Scenario-based generation of digital library services. Master's thesis, Virginia Polytechnic Institute and State University, Blacksburg, VA, USA.
- Lagoze, C.; Payette, S.; Shin, E. e Wilper, C. (2006). Fedora: an architecture for complex objects and their relationships. *International Journal on Digital Libraries*, 6(2):124–138.
- Lagoze, C. e Van de Sompel, H. (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.
- Nielsen, J. (1993). *Usability Engineering*. Academic Press, San Diego, CA, USA.
- Paepcke, A.; Chang, C.-C. K.; Winograd, T. e García-Molina, H. (1998). Interoperability for digital libraries worldwide. *Communications of the ACM*, 41(4):33–42.
- Paulk, M.; Weber, C.; Garcia, S.; Chrissis, M. B. e Bush, M. (1993). Key practices of the Capability Maturity Model, version 1.1. Technical Report 25, Software Engineering Institute, Pittsburgh, PA, USA.
- Plutchak, J.; Futrelle, J. e Gaynor, J. (2002). Components for constructing open archives. In *Proceedings of the 2nd ACM/IEEE-CS Joint Conference on Digital Libraries*, pp. 403–403, Portland, OR, USA. ACM Press.
- Prates, R. O. e Barbosa, S. D. J. (2003). *Avaliação de Interfaces de Usuário – Conceitos e Métodos*, capítulo 6. XXII Jornadas de Atualização em Informática. SBC.
- Pree, W. (1995). *Design Patterns for Object-oriented Software Development*. ACM Press / Addison-Wesley Publishing Co., New York, NY, USA.
- Pressman, R. S. (1982). *Software Engineering: A Practitioner's Approach*. McGraw-Hill, Inc., New York, NY, USA.

- Roberto, P. A. (2006). Um arcabouço baseado em componentes, serviços web e arquivos abertos para construção de bibliotecas digitais. Master's thesis, Universidade Federal de Minas Gerais, Belo Horizonte, MG, Brasil.
- Roberto, P. A.; Santos, R. L. T.; Gonçalves, M. A. e Laender, A. H. F. (2006). On RDBMS and workflow support for building componentized digital libraries. In *Anais do XXI Simpósio Brasileiro de Banco de Dados*, pp. 87–101, Florianópolis, SC, Brasil. SBC.
- Santos, R. L. T. (2005). Um assistente para configuração de bibliotecas digitais componentizadas. In *Anais do I Workshop em Bibliotecas Digitais*, pp. 11–20, Uberlândia, MG, Brasil. SBC.
- Santos, R. L. T.; Gonçalves, M. A. e Laender, A. H. F. (2007). WhizKEY: A wizard-based tool for installing digital libraries. In *Anais da IV Seção de Demos do SBBD*, pp. 51–56, João Pessoa, PB, Brasil. SBC.
- Santos, R. L. T.; Roberto, P. A.; Gonçalves, M. A. e Laender, A. H. F. (2006). Design, implementation, and evaluation of a wizard tool for setting up component-based digital libraries. In *Proceedings of the 10th European Conference on Research and Advanced Technology for Digital Libraries*, pp. 135–146, Alicante, Spain. Springer-Verlag.
- Santos, R. L. T.; Roberto, P. A.; Gonçalves, M. A. e Laender, A. H. F. (2008). A Web services-based framework for building componentized digital libraries. *Journal of Systems and Software*, 81(5):809–822.
- Sharp, H.; Rogers, Y. e Preece, J. (2007). *Interaction Design: Beyond Human Computer Interaction*. John Wiley & Sons, New York, NY, USA.
- Suleman, H.; Feng, K.; Mhlongo, S. e Omar, M. (2005). Flexing digital library systems. In *Proceedings of the 8th International Conference on Asian Digital Libraries*, pp. 33–37, Bangkok, Thailand. Springer-Verlag.
- Suleman, H. e Fox, E. A. (2001). A framework for building Open Digital Libraries. *D-Lib Magazine*, 7(12). <http://www.dlib.org/dlib/december01/suleman/12suleman.html>. Online, dezembro de 2007.
- Tansley, R.; Bass, M.; Stuve, D.; Branschovsky, 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, Houston, TX, USA. ACM Press.
- Young, P. (1994). NSF announces awards for digital libraries research. NSF PR 94-52.
- Zhu, Q.; Gonçalves, M. A.; Shen, R.; Cassell, L. e Fox, E. A. (2003). Visual semantic modeling of digital libraries. In *Proceedings of the 7th European Conference on Research and Advanced Technology for Digital Libraries*, pp. 325–337, Trondheim, Norway. Springer-Verlag.



## Apêndice A

# Documento Descritor do Arcabouço

Neste apêndice, apresentamos um exemplo do Documento Descritor do Arcabouço (DDA) elaborado para o arcabouço ODL (Suleman e Fox, 2001). Este exemplo contempla a especificação de três dos nove componentes providos por esse arcabouço: DBUnion (um mediador de dados, cujas instâncias são apresentadas como “biblioteca digitais”), XMLFile (um catálogo de metadados baseado em XML, cujas instâncias são apresentadas como “catálogos locais”) e IRDB (uma máquina de busca, cujas instâncias são apresentadas como “serviços de busca”). Além desses três componentes, a adição de instâncias de catálogos remotos ao mediador de dados (DBUnion) é abstraída por meio da especificação de um componente separado, cujas instâncias são apresentadas como “catálogos remotos”.

```
<?xml version="1.0" encoding="UTF-8"?>
<framework id="odl">
  <name>Open Digital Libraries</name>
  <description></description>
  <definitions>
    <definition id="string">.*</definition>
    <definition id="nestring">.+</definition>
    <definition id="integer">\-?\d*</definition>
    <definition id="decimal">\-?\d*\.\d*</definition>
    <definition id="boolean">(true|false)</definition>
    <definition id="mail">${login}@${host}</definition>
    <definition id="url">
      ${protocol}://(${login}@)?(${host})(:${port})?(/([\w#!:.?+=&#!-/~]+)|/)?
    </definition>
    <definition id="login">[a-zA-Z0-9._%+<->+</definition>
    <definition id="password">\S{0,32}</definition>
    <definition id="protocol">(http|https|ftp)</definition>
    <definition id="host">[a-z0-9.-]+(\.[a-z]{2,4})?</definition>
    <definition id="port">[0-9]{1,4}</definition>
    <definition id="identifier">\w*</definition>
    <definition id="templateDir">/xchange/repo/odl/xoai</definition>
    <definition id="scriptDir">sh</definition>
    <definition id="logFile">res/odl-libs.xml</definition>
    <definition id="dbunionConfig">
      ${system.general.baseDir}/ODL-DBUnion-1.2/DBUnion/${library.general.libraryId}/config.xml
    </definition>
    <definition id="xmlfileConfig">
      ${system.general.baseDir}/OAI-XMLFile-2.1/XMLFile/${lrepo.general.catalogId}/config.xml
    </definition>
  </definitions>
</framework>
```

```
</definition>
<definition id="irdbConfig">
  ${system.general.baseDir}/ODL-IRDB-1.3/IRDB/${search.general.serviceId}/config.xml
</definition>
</definitions>
<steps>
  <system>
    <step id="welcome">
      <name>Welcome</name>
      <icon>res/init.png</icon>
      <label>Welcome to the 'Open Digital Libraries Setup Wizard'</label>
      <description>
        This wizard helps you set up a new digital library or reconfigure an existing one.
      </description>
      <help>To continue, click 'Next'.</help>
    </step>
    <step id="select">
      <name>Selection</name>
      <label>Task Selection</label>
      <description>This step is responsible for selecting the task to be carried out.</description>
      <help>What do you want to do?</help>
    </step>
    <step id="identify">
      <name>Identification</name>
      <label>Identification</label>
      <description>
        This step is responsible for identifying the installation to be carried out.
      </description>
      <help>Please fill in each of the fields below.</help>
    </step>
    <step id="configure">
      <name>Configure</name>
      <label>Ready to Configure</label>
      <description>WhizKEY is ready to perform the selected configurations.</description>
      <help>
        Click on 'Configure' to perform the selected operations or
        on 'Back' to return and correct any of them.
      </help>
    </step>
    <step id="finish">
      <name>Finish</name>
      <icon>res/done.png</icon>
      <label>Completing the 'Open Digital Libraries Setup Wizard'</label>
      <description>
        You have successfully completed the 'Open Digital Libraries Setup Wizard'!
        These are the base URLs of your library:
      </description>
      <help>To close this wizard, click 'Finish'.</help>
    </step>
  </system>
  <user>
    <step id="libraries" fixed="false" min="1" max="1">
      <name>Digital Library</name>
      <label>Digital Library</label>
      <description>
        This step is responsible for setting up major elements of you library.
      </description>
      <help>Select the library you want to configure from the list below or create a new one.</help>
    </step>
  </user>
</steps>
```

```

</step>
<step id="catalogs" fixed="false" min="1" max="unbounded">
  <name>Catalogs</name>
  <label>Catalogs</label>
  <description>
    This step is responsible for setting up major elements of you library.
  </description>
  <help>Select the catalogs you want to configure from the list below or create a new one.</help>
</step>
<step id="services" fixed="true" min="0" max="unbounded">
  <name>Services</name>
  <label>Services</label>
  <description>
    This step is responsible for setting up major elements of you library.
  </description>
  <help>Select the services you want to configure from the list below.</help>
</step>
</user>
</steps>
<components>
  <system>
    <deployment>
      <deploy>
        <pre> mkdir -p ${general.baseDir}; </pre>
      </deploy>
      <undeploy>
        <pos> rm -fR ${general.baseDir}; </pos>
      </undeploy>
    </deployment>
    <configuration key="general.systemId">
      <source>
        <addr>${logFile}</addr>
        <path>/config/system/@id</path>
      </source>
      <group id="general">
        <label>General</label>
        <description></description>
        <parameters>
          <parameter id="systemId" visible="once" enabled="false">
            <label>System Identifier</label>
            <description>The unique identifier of your system configuration.</description>
            <sources>
              <source>
                <addr>${logFile}</addr>
                <path>/config/system[@id='${sid}']/@id</path>
              </source>
            </sources>
            <constraints>
              <constraint type="pattern" message="This is not a valid identifier">
                ${identifier}
              </constraint>
              <constraint type="unique" message="The identifier must be unique">
                ${systemId}
              </constraint>
            </constraints>
          </parameter>
          <parameter id="baseDir" visible="once">
            <label>Base Directory</label>

```

```

<description>The base directory of your installation.</description>
<sources>
  <source>
    <addr>${logFile}</addr>
    <path>/config/system[@id='${sid}']/baseDir</path>
  </source>
</sources>
<constraints>
  <constraint type="resource" message="This is not a valid directory">
    directory
  </constraint>
</constraints>
</parameter>
<parameter id="baseURL" visible="once">
  <label>Base URL</label>
  <description>
    The base URL to serve as the access point to your installation.
  </description>
  <sources>
    <source>
      <addr>${logFile}</addr>
      <path>/config/system[@id='${sid}']/baseURL</path>
    </source>
  </sources>
  <constraints>
    <constraint type="pattern" message="This is not a valid URL">${url}</constraint>
  </constraints>
</parameter>
</parameters>
</group>
</configuration>
</system>
<component id="library" step="libraries" min="1">
  <name>Digital Library</name>
  <label>${general.libraryName}</label>
  <description/>
  <deployment>
    <deploy>
      <pre>
        cd ${system.general.baseDir};
        tar zxvf ${templateDir}/DBUnion-1.2.tar.gz;
        cd ODL-DBUnion-1.2/DBUnion;
        cp -r template ${general.libraryId};
        mkdir -p ${general.libraryId}/data
      </pre>
    </deploy>
    <undeploy>
      <pos>
        cd ${system.general.baseDir}/ODL-DBUnion-1.2/DBUnion;
        rm -fr ${general.libraryId};
        if [ `ls |wc -l` == 2 ]; then
          rm -fr ${system.general.baseDir}/ODL-DBUnion-1.2;
        fi;
        cat 'drop table ${database.tablePrefix}; drop table ${database.tablePrefix}md' >
        ${database.tablePrefix}.sql
        ${database.dbDriver} --user=${database.dbUser} --password='${database.dbPass}'
        --database=${database.dbName} &lt; ${database.tablePrefix}.sql;
        rm -f ${database.tablePrefix}.sql
      </pos>
    </undeploy>
  </deployment>
</component>

```

```

    </pos>
  </undeploy>
</deployment>
<configuration key="general.libraryId">
  <source>
    <addr>${logFile}</addr>
    <path>/config/system[@id='${sid}']/blocks/block[@component='library']/@id</path>
  </source>
  <group id="general">
    <label>General</label>
    <description></description>
    <parameters>
      <parameter id="libraryId" visible="once">
        <label>Library Identifier</label>
        <description>The unique identifier of your digital library.</description>
        <sources>
          <source>
            <addr>${logFile}</addr>
            <path>
              /config/system[@id='${sid}']/blocks/block[@component='library']
              [@id='${libraryId}']/@id
            </path>
          </source>
        </sources>
        <constraints>
          <constraint type="pattern" message="This is not a valid identifier">
            ${identifier}
          </constraint>
          <constraint type="unique" message="The identifier must be unique">${bid}</constraint>
        </constraints>
      </parameter>
      <parameter id="libraryURL" visible="never" persistent="false">
        <constraints>
          <constraint type="default" message="">
            ${system.general.baseURL}/ODL-DBUnion-1.2/DBUnion/${libraryId}/union.pl
          </constraint>
        </constraints>
      </parameter>
      <parameter id="libraryName">
        <label>Library Name</label>
        <description>The name of your digital library.</description>
        <sources>
          <source>
            <addr>${dbunionConfig}</addr>
            <path>/odlunion/repositoryName</path>
          </source>
        </sources>
        <constraints>
          <constraint type="default" message="">Digital Library ${libraryId}</constraint>
          <constraint type="pattern" message="The library name must not be empty">
            ${nestring}
          </constraint>
        </constraints>
      </parameter>
      <parameter id="adminEmail">
        <label>Administrator E-mail</label>
        <description>The digital library administrator's e-mail address.</description>
        <sources>

```

```

    <source>
      <addr>${dbunionConfig}</addr>
      <path>/odlunion/adminEmail</path>
    </source>
  </sources>
</constraints>
  <constraint type="default">${USER}@${HOSTNAME}</constraint>
  <constraint type="pattern" message="This is not a valid e-mail address">
    ${mail}
  </constraint>
</constraints>
</parameter>
</parameters>
</group>
<group id="database">
  <label>Database</label>
  <description></description>
  <parameters>
    <parameter id="database" visible="never">
      <sources>
        <source>
          <addr>${dbunionConfig}</addr>
          <path>/odlunion/database</path>
        </source>
      </sources>
      <constraints>
        <constraint type="default">DBI:${dbDriver}:${dbName}</constraint>
      </constraints>
    </parameter>
    <parameter id="dbDriver" fixed="true" persistent="false" structure="flat">
      <label>DB Driver</label>
      <description>The name of the driver to be used to connect to a database.</description>
      <constraints>
        <constraint type="default" message="">
          @{expr match '${database}' 'DBI:\(.*\):.*'}
        </constraint>
        <constraint type="enumeration" message="You must choose a value from the list">
          @{${scriptDir}/dbi.pl}
        </constraint>
      </constraints>
    </parameter>
    <parameter id="dbName" persistent="false">
      <label>DB Name</label>
      <description>The name of the database to store and retrieve data.</description>
      <constraints>
        <constraint type="default" message="">
          @{expr match '${database}' 'DBI:.*:\(.*\)'}
        </constraint>
        <constraint type="pattern" message="The database name must be not empty">
          ${identifier}
        </constraint>
      </constraints>
    </parameter>
    <parameter id="dbUser">
      <label>DB User</label>
      <description>A username with write privileges on the specified database.</description>
      <sources>
        <source>

```

```

        <addr>${dbunionConfig}</addr>
        <path>/odlunion/dbusername</path>
    </source>
</sources>
<constraints>
    <constraint type="pattern" message="This is not a valid user name">
        ${login}
    </constraint>
</constraints>
</parameter>
<parameter id="dbPass" min="0">
    <label>DB Password</label>
    <description>The password associated to the specified username.</description>
    <sources>
        <source>
            <addr>${dbunionConfig}</addr>
            <path>/odlunion/dbpassword</path>
        </source>
    </sources>
    <constraints>
        <constraint type="pattern" message="This is not a valid password">
            ${password}
        </constraint>
    </constraints>
</parameter>
<parameter id="tablePrefix" visible="never">
    <sources>
        <source>
            <addr>${dbunionConfig}</addr>
            <path>/odlunion/table</path>
        </source>
    </sources>
    <constraints>
        <constraint type="default" message="">${sid}_${general.libraryId}</constraint>
    </constraints>
</parameter>
<parameter id="recordLimit" min="0">
    <label>Record Limit</label>
    <description>
        The number of records to be issued in each batch before a resumption token is issued.
    </description>
    <sources>
        <source>
            <addr>${dbunionConfig}</addr>
            <path>/odlunion/recordlimit</path>
        </source>
    </sources>
    <constraints>
        <constraint type="default" message="">500</constraint>
        <constraint type="pattern" message="This is not a valid integer">
            ${integer}
        </constraint>
        <constraint type="check" message="Record limit must be a positive integer">
            ${recordLimit} \> 0
        </constraint>
    </constraints>
</parameter>
</parameters>

```

```

    </group>
  </configuration>
</component>
<component id="lrepo" step="catalogs" max="unbounded">
  <name>Local Catalog</name>
  <label>${general.catalogName}</label>
  <description></description>
  <deployment>
    <deploy>
      <pre>
        cd ${system.general.baseDir};
        tar zxvf ${templateDir}/OAI-XMLFile-2.1.tar.gz;
        cd OAI-XMLFile-2.1/XMLFile;
        cp -r template ${general.catalogId};
      </pre>
      <pos>
        ${system.general.baseDir}/ODL-DBUnion-1.2/DBUnion/${library.general.libraryId}/harvest.pl
      </pos>
    </deploy>
    <undeploy>
      <pos>
        cd ${system.general.baseDir}/OAI-XMLFile-2.1/XMLFile;
        rm -fr ${general.catalogId};
        if [ `ls |wc -l` == 2 ]; then
          rm -fr ${system.general.baseDir}/OAI-XMLFile-2.1;
        fi;
        ${system.general.baseDir}/ODL-DBUnion-1.2/DBUnion/${library.general.libraryId}/harvest.pl
      </pos>
    </undeploy>
  </deployment>
</component>
<configuration key="general.catalogId">
  <source>
    <addr>${logFile}</addr>
    <path>/config/system[@id='${sid}']/blocks/block[@component='lrepo']/@id</path>
  </source>
  <group id="general">
    <label>General</label>
    <description></description>
    <parameters>
      <parameter id="catalogId" visible="once">
        <label>Catalog Identifier</label>
        <description>A unique identifier for the catalog.</description>
        <sources>
          <source>
            <addr>${xmlfileConfig}</addr>
            <path>/xmlfile/archiveId</path>
          </source>
          <source>
            <addr>${dbunionConfig}</addr>
            <path>/odlunion/archive[identifier='${catalogId}']/identifier</path>
          </source>
          <source>
            <addr>${logFile}</addr>
            <path>
              /config/system[@id='${sid}']/blocks/block[@component='lrepo']
              [@id='${catalogId}']/@id
            </path>
          </source>
        </sources>
      </parameter>
    </parameters>
  </group>
</configuration>

```



```

</sources>
<constraints>
  <constraint type="pattern" message="This is not a valid identifier">
    ${identifier}
  </constraint>
  <constraint type="unique" message="The identifier must be unique">${bid}</constraint>
</constraints>
</parameter>
<parameter id="catalogURL" visible="never">
  <sources>
    <source>
      <addr>${dbunionConfig}</addr>
      <path>/odlunion/archive[identifier='${catalogId}']/url</path>
    </source>
  </sources>
  <constraints>
    <constraint type="default" message="">
      ${system.general.baseURL}/OAI-XMLFile-2.1/XMLFile/${catalogId}/oai.pl
    </constraint>
  </constraints>
</parameter>
<parameter id="catalogName">
  <label>Catalog Name</label>
  <description>The name of this catalog.</description>
  <sources>
    <source>
      <addr>${xmlfileConfig}</addr>
      <path>/xmlfile/repositoryName</path>
    </source>
  </sources>
  <constraints>
    <constraint type="default" message="">Local Catalog ${catalogId}</constraint>
    <constraint type="pattern" message="Catalog name must not be empty">
      ${nestring}
    </constraint>
  </constraints>
</parameter>
<parameter id="adminEmail">
  <label>Administrator E-mail</label>
  <description>The catalog administrator's e-mail address.</description>
  <sources>
    <source>
      <addr>${xmlfileConfig}</addr>
      <path>/xmlfile/adminEmail</path>
    </source>
  </sources>
  <constraints>
    <constraint type="default" message="">${library.general.adminEmail}</constraint>
    <constraint type="pattern" message="This is not a valid e-mail address">
      ${mail}
    </constraint>
  </constraints>
</parameter>
</parameters>
</group>
<group id="storage">
  <label>Storage</label>
  <description></description>

```

```
<parameters>
  <parameter id="dataDir" visible="once">
    <label>Data Directory</label>
    <description>The directory that contains all the data files.</description>
    <sources>
      <source>
        <addr>${xmlfileConfig}</addr>
        <path>/xmlfile/datadir</path>
      </source>
    </sources>
    <constraints>
      <constraint type="resource" message="This is not a valid directory">
        directory
      </constraint>
    </constraints>
  </parameter>
  <parameter id="recordLimit">
    <label>Record Limit</label>
    <description>
      The number of records to be issued in each batch before a resumption token is issued.
    </description>
    <sources>
      <source>
        <addr>${xmlfileConfig}</addr>
        <path>/xmlfile/recordlimit</path>
      </source>
    </sources>
    <constraints>
      <constraint type="default">500</constraint>
      <constraint type="pattern" message="This is not a valid integer">
        ${integer}
      </constraint>
      <constraint type="check" message="Record limit must be a positive integer">
        ${recordLimit} \> 0
      </constraint>
    </constraints>
  </parameter>
  <parameter id="longIds" fixed="true" structure="flat" max="1" isep=";">
    <label>Long Identifiers</label>
    <description>
      Do you want to use the full pathname as part of the identifiers?
    </description>
    <sources>
      <source>
        <addr>${xmlfileConfig}</addr>
        <path>/xmlfile/longids</path>
      </source>
    </sources>
    <constraints>
      <constraint type="default">yes</constraint>
      <constraint type="enumeration" message="">yes#Yes;no#No</constraint>
    </constraints>
  </parameter>
  <parameter id="fileMatch">
    <label>File Match</label>
    <description>
      The regular expression that matches all files with actual metadata.
    </description>
```

```

    <sources>
      <source>
        <addr>${xmlfileConfig}</addr>
        <path>/xmlfile/filematch</path>
      </source>
    </sources>
  </constraints>
  <constraint type="default">[^\.\.]+\.[xX][mM][lL]${</constraint>
  <constraint type="pattern" message="This is not a valid regular expression">
    ${nestring}
  </constraint>
</constraints>
</parameter>
</parameters>
</group>
<group id="harvesting">
  <label>Harvesting</label>
  <description></description>
  <parameters>
    <parameter id="metadata">
      <label>Metadata Formats</label>
      <description>
        Choose the metadata formats to be harvested from this catalog and stored.
      </description>
      <sources>
        <source>
          <addr>${xmlfileConfig}</addr>
          <path>/xmlfile/metadata</path>
        </source>
      </sources>
    </complex>
    <parameter id="prefix">
      <label>MD Prefix</label>
      <description></description>
      <sources>
        <source>
          <path>prefix</path>
        </source>
        <source>
          <addr>${dbunionConfig}</addr>
          <path>/odlunion/archive[identifier='${general.catalogId}']/metadataPrefix</path>
        </source>
      </sources>
    </constraints>
    <constraint type="default">oai_dc</constraint>
    <constraint type="pattern" message="This is not a valid metadata prefix">
      ${nestring}
    </constraint>
  </constraints>
</parameter>
<parameter id="namespace">
  <label>MD Namespace</label>
  <description></description>
  <sources>
    <source>
      <path>namespace</path>
    </source>
  </sources>

```

```

    <constraints>
      <constraint type="default">http://www.openarchives.org/OAI/2.0/oai_dc/</constraint>
      <constraint type="pattern" message="This is not a valid namespace">
        ${nestring}
      </constraint>
    </constraints>
  </parameter>
  <parameter id="schema">
    <label>MD Schema</label>
    <description></description>
    <sources>
      <source>
        <path>schema</path>
      </source>
    </sources>
    <constraints>
      <constraint type="default">
        http://www.openarchives.org/OAI/2.0/oai_dc.xsd
      </constraint>
      <constraint type="pattern" message="This is not a valid XML schema">
        ${nestring}
      </constraint>
    </constraints>
  </parameter>
</complex>
</parameter>
<parameter id="interval" structure="flat" fixed="true" isep=";">
  <label>Harvesting Interval</label>
  <description>Choose how often you want to harvest from this catalog.</description>
  <sources>
    <source>
      <addr>${dbunionConfig}</addr>
      <path>/odlunion/archive[identifier='${general.catalogId}']/interval</path>
    </source>
  </sources>
  <constraints>
    <constraint type="enumeration" message="">86400#Daily;3600#Hourly</constraint>
  </constraints>
</parameter>
<parameter id="interrequestGap" min="0">
  <label>Inter-request Gap</label>
  <description>
    Number of seconds to wait between sending back resumption tokens.
  </description>
  <sources>
    <source>
      <addr>${dbunionConfig}</addr>
      <path>/odlunion/archive[identifier='${general.catalogId}']/interrequestgap</path>
    </source>
  </sources>
  <constraints>
    <constraint type="pattern" message="This is not a valid integer">
      ${integer}
    </constraint>
    <constraint type="check" message="Inter-request gap must be a positive integer">
      ${interrequestGap} \> 0
    </constraint>
  </constraints>
</parameter>

```

```

</parameter>
<parameter id="set" structure="hierarchical" fixed="true">
  <label>Set</label>
  <description>
    Choose the set to be harvested (leave blank to harvest the whole archive).
  </description>
  <sources>
    <source>
      <addr>${dbunionConfig}</addr>
      <path>/odlunion/archive[identifier='${general.catalogId}']/set</path>
    </source>
  </sources>
  <constraints>
    <constraint type="enumeration" message="">
      @${scriptDir}/tree.sh ${storage.dataDir} ':'
    </constraint>
  </constraints>
</parameter>
<parameter id="overlap" min="0">
  <label>Harvesting Overlap</label>
  <description>
    The number of seconds by which to overlap harvesting to handle differences in times.
  </description>
  <sources>
    <source>
      <addr>${dbunionConfig}</addr>
      <path>/odlunion/archive[identifier='${general.catalogId}']/overlap</path>
    </source>
  </sources>
  <constraints>
    <constraint type="pattern" message="This is not a valid integer">
      ${integer}
    </constraint>
    <constraint type="check" message="Overlap must be a positive integer">
      ${overlap} \> 0
    </constraint>
  </constraints>
</parameter>
<parameter id="granularity" fixed="true" structure="flat" min="0" isep=";">
  <label>Harvesting Granularity</label>
  <description>
    This specifies whether harvesting is performed by day or second.
  </description>
  <sources>
    <source>
      <addr>${dbunionConfig}</addr>
      <path>/odlunion/archive[identifier='${general.catalogId}']/granularity</path>
    </source>
  </sources>
  <constraints>
    <constraint type="enumeration" message="">day#Day;second#Second</constraint>
  </constraints>
</parameter>
</parameters>
</group>
</configuration>
</component>
<component id="rrepo" step="catalogs" max="unbounded">

```

```

<name>Remote Catalog</name>
<label>${general.catalogName}</label>
<description></description>
<deployment>
  <deploy>
    <pos>
      ${system.general.baseDir}/ODL-DBUnion-1.2/DBUnion/${library.general.libraryId}/harvest.pl
    </pos>
  </deploy>
  <update>
    <pos>
      ${system.general.baseDir}/ODL-DBUnion-1.2/DBUnion/${library.general.libraryId}/harvest.pl
    </pos>
  </update>
</deployment>
<configuration key="general.catalogId">
  <source>
    <addr>${logFile}</addr>
    <path>/config/system[@id='${sid}']/blocks/block[@component='rrepo']/@id</path>
  </source>
  <group id="general">
    <label>General</label>
    <description></description>
    <parameters>
      <parameter id="catalogId" visible="once">
        <label>Catalog Identifier</label>
        <description>A unique identifier for the catalog.</description>
        <sources>
          <source>
            <addr>${dbunionConfig}</addr>
            <path>/odlunion/archive[identifier='${catalogId}']/identifier</path>
          </source>
          <source>
            <addr>${logFile}</addr>
            <path>
              /config/system[@id='${sid}']/blocks/block[@component='rrepo']
              [@id='${catalogId}']/@id
            </path>
          </source>
        </sources>
        <constraints>
          <constraint type="pattern" message="This is not a valid identifier">
            ${identifier}
          </constraint>
          <constraint type="unique" message="The identifier must be unique">${bid}</constraint>
        </constraints>
      </parameter>
      <parameter id="catalogURL" visible="once">
        <label>Catalog URL</label>
        <description>The base URL for this catalog.</description>
        <sources>
          <source>
            <addr>${dbunionConfig}</addr>
            <path>/odlunion/archive[identifier='${catalogId}']/url</path>
          </source>
        </sources>
        <constraints>

```

```

        <constraint type="pattern" message="This is not a valid URL">${url}</constraint>
        <constraint type="resource" message="This is not a valid OAI provider">oai</constraint>
    </constraints>
</parameter>
<parameter id="catalogName" enabled="false" persistent="false">
    <label>Catalog Name</label>
    <description>The name of this catalog.</description>
    <constraints>
        <constraint type="default" message="">
            @${scriptDir}/oai-name.pl ${catalogURL}
        </constraint>
    </constraints>
</parameter>
</parameters>
</group>
<group id="harvesting">
    <label>Harvesting</label>
    <description></description>
    <parameters>
        <parameter id="prefix" structure="flat" fixed="true">
            <label>MD Prefix</label>
            <description>
                Choose the metadata format to be harvested from this catalog and stored.
            </description>
            <sources>
                <source>
                    <addr>${dbunionConfig}</addr>
                    <path>/odlunion/archive[identifier='${general.catalogId}']/metadataPrefix</path>
                </source>
            </sources>
            <constraints>
                <constraint type="enumeration" message="">
                    @${scriptDir}/oai-listMetadataFormats.pl ${general.catalogURL}
                </constraint>
            </constraints>
        </parameter>
        <parameter id="interval" structure="flat" fixed="true" isep=";">
            <label>Harvesting Interval</label>
            <description>Choose how often you want to harvest from this catalog.</description>
            <sources>
                <source>
                    <addr>${dbunionConfig}</addr>
                    <path>/odlunion/archive[identifier='${general.catalogId}']/interval</path>
                </source>
            </sources>
            <constraints>
                <constraint type="default" message="">86400</constraint>
                <constraint type="enumeration" message="">86400#Daily;3600#Hourly</constraint>
            </constraints>
        </parameter>
        <parameter id="interrequestGap" min="0">
            <label>Inter-request Gap</label>
            <description>
                Number of seconds to wait between sending back resumption tokens.
            </description>
            <sources>
                <source>
                    <addr>${dbunionConfig}</addr>

```

```

        <path>/odlunion/archive[identifier='${general.catalogId}']/interrequestgap</path>
    </source>
</sources>
<constraints>
    <constraint type="default" message="">15</constraint>
    <constraint type="pattern" message="This is not a valid integer">
        ${integer}
    </constraint>
    <constraint type="check" message="Inter-request gap must be a positive integer">
        ${interrequestGap} \> 0
    </constraint>
</constraints>
</parameter>
<parameter id="set" structure="hierarchical" fixed="true">
    <label>Set</label>
    <description>
        Choose the set to be harvested (leave blank to harvest the whole archive).
    </description>
    <sources>
        <source>
            <addr>${dbunionConfig}</addr>
            <path>/odlunion/archive[identifier='${general.catalogId}']/set</path>
        </source>
    </sources>
    <constraints>
        <constraint type="enumeration" message="">
            @${scriptDir}/oai-listSets.pl ${general.catalogURL}
        </constraint>
    </constraints>
</parameter>
<parameter id="overlap" structure="flat" fixed="true" max="1">
    <label>Harvesting Overlap</label>
    <description>
        The number of seconds by which to overlap harvesting to handle differences in times.
    </description>
    <sources>
        <source>
            <addr>${dbunionConfig}</addr>
            <path>/odlunion/archive[identifier='${general.catalogId}']/overlap</path>
        </source>
    </sources>
    <constraints>
        <constraint type="enumeration" message="">
            @${scriptDir}/oai-overlap.pl ${general.catalogURL}
        </constraint>
    </constraints>
</parameter>
<parameter id="granularity" structure="flat" fixed="true" max="1">
    <label>Harvesting Granularity</label>
    <description>
        This specifies whether harvesting is performed by day or second.
    </description>
    <sources>
        <source>
            <addr>${dbunionConfig}</addr>
            <path>/odlunion/archive[identifier='${general.catalogId}']/granularity</path>
        </source>
    </sources>

```



```

        <constraints>
            <constraint type="enumeration" message="">
                @!${scriptDir}/oai-granularity.pl $!{general.catalogURL}
            </constraint>
        </constraints>
    </parameter>
</parameters>
</group>
</configuration>
</component>
<component id="search" step="services">
    <name>Searching</name>
    <label>Searching</label>
    <description>A search engine.</description>
    <deployment>
        <deploy>
            <pre>
                cd ${system.general.baseDir};
                tar zxvf ${templateDir}/IRDB-1.3.tar.gz;
                tar zxvf ${templateDir}/compute_ui.tar.gz;
                replace "my \${baseURL} = '" "my \${baseURL} = '${general.serviceURL}'"; -- UI/search.pl;
                cd ODL-IRDB-1.3/IRDB;
                cp -r template $!{general.serviceId};
                mkdir -p $!{general.serviceId}/data
            </pre>
            <pos>
                ${system.general.baseDir}/ODL-IRDB-1.3/IRDB/$!{general.serviceId}/harvest.pl
            </pos>
        </deploy>
        <update>
            <pos>
                ${system.general.baseDir}/ODL-IRDB-1.3/IRDB/$!{general.serviceId}/harvest.pl
            </pos>
        </update>
    </deployment>
</configuration key="general.serviceId">
    <source>
        <addr>${logFile}</addr>
        <path>/config/system[@id='${sid}']/blocks/block[@component='search']/@id</path>
    </source>
    <group id="general">
        <label>General</label>
        <description></description>
        <parameters>
            <parameter id="serviceId" visible="once">
                <label>Service Identifier</label>
                <description>The unique identifier of this service.</description>
                <sources>
                    <source>
                        <addr>${logFile}</addr>
                        <path>
                            /config/system[@id='${sid}']/blocks/block[@component='search']
                            [@id='${serviceId}']/@id
                        </path>
                    </source>
                </sources>
            </parameter>
        </parameters>
    </group>
</constraints>

```

```

    <constraint type="pattern" message="This is not a valid identifier">
      ${identifier}
    </constraint>
    <constraint type="unique" message="The service identifier must be unique">
      ${bid}
    </constraint>
  </constraints>
</parameter>
<parameter id="serviceURL" visible="never" persistent="false">
  <constraints>
    <constraint type="default" message="">
      ${system.general.baseURL}/ODL-IRDB-1.3/IRDB/${serviceId}/search.pl
    </constraint>
  </constraints>
</parameter>
<parameter id="serviceName">
  <label>Service Name</label>
  <description>The name of this service.</description>
  <sources>
    <source>
      <addr>${irdbConfig}</addr>
      <path>/odlsearch/repositoryName</path>
    </source>
  </sources>
  <constraints>
    <constraint type="default">${serviceId} Searching Service</constraint>
    <constraint type="pattern" message="The service name must not be empty">
      ${nestring}
    </constraint>
  </constraints>
</parameter>
<parameter id="adminEmail">
  <label>Administrator E-mail</label>
  <description>This service administrator's e-mail address.</description>
  <sources>
    <source>
      <addr>${irdbConfig}</addr>
      <path>/odlsearch/adminEmail</path>
    </source>
  </sources>
  <constraints>
    <constraint type="default">${library.general.adminEmail}</constraint>
    <constraint type="pattern" message="This is not a valid e-mail address">
      ${mail}
    </constraint>
  </constraints>
</parameter>
</parameters>
</group>
<group id="database">
  <label>Database</label>
  <description></description>
  <parameters>
    <parameter id="database" visible="never">
      <sources>
        <source>
          <addr>${irdbConfig}</addr>
          <path>/odlsearch/database</path>
        </source>
      </sources>
    </parameter>
  </parameters>
</group>

```

```

    </source>
  </sources>
  <constraints>
    <constraint type="default">DBI:${dbDriver}:${dbName}</constraint>
  </constraints>
</parameter>
<parameter id="dbDriver" fixed="true" persistent="false" structure="flat">
  <label>DB Driver</label>
  <description>The name of the driver to be used to connect to a database.</description>
  <constraints>
    <constraint type="default" message="">
      @{\expr match '${database}' 'DBI:\(.*\):.*'}
    </constraint>
    <constraint type="enumeration" message="You must choose a value from the list">
      @{\scriptDir}/dbi.pl}
    </constraint>
  </constraints>
</parameter>
<parameter id="dbName" persistent="false">
  <label>DB Name</label>
  <description>The name of the database to store and retrieve data.</description>
  <constraints>
    <constraint type="default" message="">
      @{\expr match '${database}' 'DBI:.*:\(.*\)'}
    </constraint>
    <constraint type="pattern" message="The database name must be not empty">
      ${identifier}
    </constraint>
  </constraints>
</parameter>
<parameter id="dbUser">
  <label>DB User</label>
  <description>A username with write privileges on the specified database.</description>
  <sources>
    <source>
      <addr>${irdbConfig}</addr>
      <path>/odlsearch/dbusername</path>
    </source>
  </sources>
  <constraints>
    <constraint type="pattern" message="This is not a valid user name">
      ${login}
    </constraint>
  </constraints>
</parameter>
<parameter id="dbPass" min="0">
  <label>DB Password</label>
  <description>The password associated to the specified username.</description>
  <sources>
    <source>
      <addr>${irdbConfig}</addr>
      <path>/odlsearch/dbpassword</path>
    </source>
  </sources>
  <constraints>
    <constraint type="pattern" message="This is not a valid password">
      ${password}
    </constraint>

```

```

    </constraints>
  </parameter>
  <parameter id="tablePrefix" visible="never">
    <sources>
      <source>
        <addr>${irdbConfig}</addr>
        <path>/odlsearch/table</path>
      </source>
    </sources>
    <constraints>
      <constraint type="default" message="">${sid}_${general.serviceId}</constraint>
    </constraints>
  </parameter>
</parameters>
</group>
<group id="harvesting">
  <label>Harvesting</label>
  <description></description>
  <parameters>
    <parameter id="baseUrl" visible="never">
      <sources>
        <source>
          <addr>${irdbConfig}</addr>
          <path>/odlsearch/archive[identifier='${library.general.libraryId}']/url</path>
        </source>
      </sources>
      <constraints>
        <constraint type="default" message="">${library.general.libraryURL}</constraint>
      </constraints>
    </parameter>
    <parameter id="prefix" structure="flat" fixed="true" max="1">
      <label>MD Prefix</label>
      <description></description>
      <sources>
        <source>
          <addr>${irdbConfig}</addr>
          <path>
            /odlsearch/archive[identifier='${library.general.libraryId}']/metadataPrefix
          </path>
        </source>
      </sources>
      <constraints>
        <constraint type="enumeration" message="">
          ${lrepo.harvesting.metadata.prefix} ${rrepo.harvesting.prefix}
        </constraint>
      </constraints>
    </parameter>
    <parameter id="interval" structure="flat" fixed="true" isep=";">
      <label>Harvesting Interval</label>
      <description>Choose how often you want to harvest from this catalog.</description>
      <sources>
        <source>
          <addr>${irdbConfig}</addr>
          <path>/odlsearch/archive[identifier='${library.general.libraryId}']/interval</path>
        </source>
      </sources>
      <constraints>
        <constraint type="default" message="">86400</constraint>
      </constraints>
    </parameter>
  </parameters>
</group>

```

```

        <constraint type="enumeration" message="">86400#Daily;3600#Hourly</constraint>
    </constraints>
</parameter>
<parameter id="interrequestGap" min="0">
    <label>Inter-request Gap</label>
    <description>
        Number of seconds to wait between sending back resumption tokens.
    </description>
    <sources>
        <source>
            <addr>${irdbConfig}</addr>
            <path>
                /odlsearch/archive[identifier='${library.general.libraryId}']/interrequestgap
            </path>
        </source>
    </sources>
    <constraints>
        <constraint type="default" message="">15</constraint>
        <constraint type="pattern" message="This is not a valid integer">
            ${integer}
        </constraint>
        <constraint type="check" message="Inter-request gap must be a positive integer">
            ${interrequestGap} \> 0
        </constraint>
    </constraints>
</parameter>
<parameter id="set" structure="hierarchical" fixed="true" min="0" nsep="">
    <label>Set</label>
    <description>
        Choose the set to be harvested (leave blank to harvest the whole archive).
    </description>
    <sources>
        <source>
            <addr>${irdbConfig}</addr>
            <path>/odlsearch/archive[identifier='${library.general.libraryId}']/set</path>
        </source>
    </sources>
    <constraints>
        <constraint type="enumeration" message="">
            #Whole Tree
            ${lrepo.harvesting.set}
            ${rrepo.harvesting.set}
        </constraint>
    </constraints>
</parameter>
<parameter id="overlap" structure="flat" fixed="true" max="1" isep="">
    <label>Harvesting Overlap</label>
    <description>
        The number of seconds by which to overlap haversting to handle differences in times.
    </description>
    <sources>
        <source>
            <addr>${irdbConfig}</addr>
            <path>/odlsearch/archive[identifier='${library.general.libraryId}']/overlap</path>
        </source>
    </sources>
    <constraints>
        <constraint type="default">1</constraint>

```

```
<constraint type="enumeration" message="">
  1#Same timezone;86401#Different or unknown timezone
</constraint>
</constraints>
</parameter>
<parameter id="granularity" structure="flat" fixed="true" max="1" isep=";">
  <label>Harvesting Granularity</label>
  <description>
    This specifies whether harvesting is performed by day or second.
  </description>
  <sources>
    <source>
      <addr>${irdbConfig}</addr>
      <path>
        /odlsearch/archive[identifier='${library.general.libraryId}']/granularity
      </path>
    </source>
  </sources>
  <constraints>
    <constraint type="default" message="">second</constraint>
    <constraint type="enumeration" message="">day#Day;second#Second</constraint>
  </constraints>
</parameter>
</parameters>
</group>
</configuration>
</component>
</components>
</framework>
```

## Apêndice B

# Material para Testes com Usuários

### B.1 Perfil dos Usuários

1. Qual o seu grau de instrução?
  - 3º grau incompleto
  - 3º grau completo
  - Pós-Graduação completo
  - Pós-Graduação incompleto
2. Em que local você utiliza o computador? (pode-se marcar mais de uma opção)
  - Em casa
  - No trabalho
  - Na escola
  - Outros (detalhes)
3. Em média, quantas horas por dia você utiliza o computador?
  - Menos de 1 hora
  - Entre 1 e 3 horas
  - Entre 3 e 6 horas
  - Mais de 6 horas
4. Em média, quantos dias na semana você utiliza o computador?
  - 1 dia
  - De 2 a 4
  - De 4 a 6
  - Todos os dias
5. Qual seu nível de conhecimento em bibliotecas digitais?

- Nenhum
  - Tenho alguns conhecimentos
  - Domino o assunto
6. Qual seu nível de conhecimento sobre o arcabouço WS-ODL?
- Nenhum
  - Tenho alguns conhecimentos
  - Domino o assunto
7. Você já criou/configurou alguma biblioteca digital?
- Não
  - Sim (detalhes)
8. Se você respondeu sim à pergunta anterior: Para a construção da(s) biblioteca(s) mencionada(s), você utilizou algum pacote de componentes?
- Não, implementei minha própria
  - Sim (detalhes)

## B.2 Contextualização

Obrigado por participar deste experimento! Gostaríamos que você nos ajudasse a avaliar o protótipo de uma nova ferramenta, um assistente para construção de bibliotecas digitais a partir de um conjunto de componentes de software. Essa avaliação nos permitirá identificar falhas nessa ferramenta e implementar melhorias em suas futuras versões.

Nossa ferramenta é destinada a projetistas de bibliotecas digitais e visa a auxiliá-los na tarefa de construção de bibliotecas a partir de componentes de software. Neste teste, serão utilizados os componentes providos pelo arcabouço WS-ODL (*Web Services-based Open Digital Libraries*). Um componente de software é uma unidade independente, que encapsula uma série de funcionalidades e pode ser utilizado com outros componentes para formar um sistema mais complexo. A tarefa de construção consiste, basicamente, na configuração de cada um dos componentes providos pelo arcabouço, bem como na composição desses componentes para a constituição de uma biblioteca digital funcional.

Informalmente, podemos dizer que uma biblioteca digital engloba coleções gerenciadas de informação digital, acessíveis em uma rede e com serviços associados para suportar as necessidades de informação das sociedades com ela envolvidas. Em uma biblioteca digital, a informação é armazenada na forma de objetos digitais ou documentos, os quais podem conter streams multimídia (imagem, vídeo, áudio) e podem ter organizações internas complexas baseadas em metadados estruturais. Metadados descritivos descrevem diferentes propriedades semânticas de objetos digitais e normalmente obedecem a certo formato de metadados (por



exemplo, Dublin Core, MARC). Instâncias de metadados descritivos são agrupados em catálogos de metadados. Serviços básicos de uma biblioteca digital incluem busca e navegação. Esses serviços podem ser adaptados a diferentes sociedades dependendo de seus papéis (por exemplo, criadores de material, bibliotecários), necessidades ou interesses particulares.

Durante esta sessão, você deverá executar tarefas de construção de bibliotecas digitais utilizando a ferramenta assistente. Lembre-se de que você está ajudando a avaliar a ferramenta; nós não estamos avaliando você. Sinta-se à vontade para expressar qualquer opinião sobre qualquer aspecto do sistema ou das tarefas que lhe foram designadas. Estamos interessados em questões referentes à facilidade de uso da ferramenta, por isso nós iremos registrar suas atividades com ela. E, para obter sua opinião sobre o sistema, iremos pedir a você que responda um pequeno questionário de avaliação da ferramenta, depois de terminar de usá-la.

Antes de começarmos, alguma dúvida?

## B.3 Descrição das Tarefas

### B.3.1 Descrição da Tarefa $T_1$

1. Execute o assistente.
2. Na tela de configuração de bibliotecas, configure a biblioteca “NDLTD”:
  - a) Altere a descrição da biblioteca para “Networked Digital Library of Theses and Dissertations”.
3. Na tela de configuração de repositórios, configure o repositório “NDTLD Repository”:
  - a) Adicione seu e-mail à lista de e-mails de administradores do repositório.
  - b) Altere o nome do servidor para “escalvada”.
  - c) Adicione “escalvada” à lista de servidores com permissão de acesso ao repositório.
  - d) Altere o nome do repositório OAI para “NDLTD OAI Repository”.
  - e) Adicione seu e-mail à lista de e-mails de administradores do repositório OAI.
4. Na tela de configuração de serviços, configure os serviços associados à “NDLTD”:
  - a) Tente desmarcar o serviço “Administration”.
  - b) Marque e configure o serviço “Browsing”:
    - i. Adicione “dc:creator” à lista de dimensões de navegação.
  - c) Marque e configure o serviço “Searching”:
    - i. Adicione “@” à lista de delimitadores.
    - ii. Altere a folha de estilo para “busca.xml”.
5. Na tela seguinte, salve as configurações efetuadas.
6. Encerre o assistente.

### B.3.2 Descrição da Tarefa $T_2$

1. Execute o assistente.
2. Configure uma nova biblioteca digital:
  - a) Altere o diretório base para “/home/rodrygo/test”.
  - b) Altere o nome da biblioteca para “BDBComp”.
  - c) Altere a descrição para “Biblioteca Digital Brasileira de Computação”.
3. Configure um novo repositório para a biblioteca criada:
  - a) Altere o nome do repositório para “BDBComp Repository”.
  - b) Adicione seu e-mail à lista de e-mails de administradores do repositório.
  - c) Altere a senha do administrador para “test”.
  - d) Altere o nome do servidor para “escalvada”.
  - e) Altere a senha de acesso ao SGBD para “dbAdmin”.
  - f) Altere o PID Namespace para “bdbcomp”.
  - g) Altere o diretório para armazenamento dos objetos para “/home/rodrygo/test/fedora-2.0/objects”.
  - h) Altere o diretório temporário para “/home/rodrygo/test/fedora-2.0/temp”.
  - i) Altere o diretório dos datastreams para “/home/rodrygo/test/fedora-2.0/datastreams”.
  - j) Altere o diretório utilizado pelo Resource Index para “/home/rodrygo/test/fedora-2.0/resourceIndex”.
  - k) Adicione “localhost” à lista de servidores com permissão de acesso ao repositório.
  - l) Altere o nome do repositório OAI para “BDBComp OAI Repository”.
  - m) Altere o domínio do repositório OAI para “lbd.dcc.ufmg.br”.
  - n) Adicione seu e-mail à lista de e-mails de administradores do repositório OAI.
4. Configure os serviços associados à biblioteca:
  - a) Marque e configure o serviço “Browsing”:
    - i. Adicione “dc:creator” e “dc:date” à lista de dimensões de navegação.
    - ii. Altere o número de registros por página para “15”.
  - b) Marque e configure o serviço “Ingestor”:
    - i. Altere o diretório de armazenamento de novos objetos para “/home/rodrygo/test/fedora-2.0/data/objects”.
  - c) Marque e configure o serviço “Recent Works”:
    - i. Altere a folha de estilo para “trabalhosrecentes.xsl”.

- d) Marque e configure o serviço “Searching”:
    - i. Adicione “dc:creator” à lista de campos para indexação.
    - ii. Adicione “about” e “www” à lista de stopwords.
  - e) Marque e configure o serviço “Annotation”:
    - i. Altere a folha de estilo para “annotation.xml”.
5. Salve as configurações efetuadas.
  6. Encerre o assistente.

### B.3.3 Descrição da Tarefa $T_3$

Você foi designado para configurar e administrar uma nova biblioteca digital, que compreenderá todas as teses e dissertações produzidas na UFMG. A fim de facilitar a criação dessa biblioteca, você utilizará a ferramenta assistente para guiá-lo ao longo da tarefa de construção.

A seguir, estão todas as informações referentes à biblioteca a ser criada:

- O diretório base da biblioteca deve ser “c:\temp\ufmgdl”.
- O nome da biblioteca deve ser “UFMG-DL”.
- A descrição da biblioteca deve ser “Biblioteca Digital de Teses e Dissertações”.

Essa biblioteca terá um único repositório de dados, do qual você será administrador. Por esse motivo, seu *e-mail* deve ser adicionado à lista de *e-mails* dos administradores e “localhost” deve ser adicionado à lista de servidores com permissão de acesso ao repositório. Aproveite e mude as seguintes configurações:

- A senha do administrador do repositório deve ser “test”.
- O servidor do repositório deve ser “www.ufmg.br”.

O repositório de sua biblioteca utiliza quatro diretórios para armazenamento de seu conteúdo. Configure os parâmetros correspondentes a cada um desses diretórios:

- Para armazenamento de objetos, “c:\temp\ufmgdl\fedora-2.0\objects”.
- Para armazenamento temporário, “c:\temp\ufmgdl\fedora-2.0\temp”.
- Para armazenamento de *datastreams*, “c:\temp\ufmgdl\fedora-2.0\datastreams”.
- Para armazenamento de relacionamentos, “c:\temp\ufmgdl\fedora-2.0\resourceIndex”.

A fim de que os usuários possam navegar pelo conteúdo de sua biblioteca, o serviço “Navegação” deve ser configurado. Para esse serviço, você deve permitir navegações por duas dimensões do formato Dublin Core: “dc:creator” e “dc:date”. Além disso, as páginas de navegação devem exibir um máximo de 15 registros por página.

## B.4 Questionário de Avaliação

1. A configuração do arcabouço através do assistente é simples.

- Concordo com veemência
- Concordo
- Neutro
- Discordo
- Discordo com veemência

2. Senti-me confortável ao utilizar o assistente.

- Concordo com veemência
- Concordo
- Neutro
- Discordo
- Discordo com veemência

3. O assistente é fácil de usar.

- Concordo com veemência
- Concordo
- Neutro
- Discordo
- Discordo com veemênciaa

4. As instruções do roteiro são claras.

- Concordo com veemência
- Concordo
- Neutro
- Discordo
- Discordo com veemência

5. Entendi o significado de todos os conceitos envolvidos na tarefa.

- Concordo com veemência
- Concordo
- Neutro
- Discordo
- Discordo com veemência

6. Executar o roteiro de instalação melhorou meu entendimento sobre o arcabouço WS-ODL.
  - Concordo com veemência
  - Concordo
  - Neutro
  - Discordo
  - Discordo com veemência
7. Com base nessa experiência, sinto-me capaz de configurar uma nova biblioteca digital, através do assistente, sem o auxílio de um roteiro.
  - Concordo com veemência
  - Concordo
  - Neutro
  - Discordo
  - Discordo com veemência

## B.5 Roteiro para Entrevista

1. A ferramenta o ajudou a compreender melhor os conceitos de bibliotecas digitais e, em particular, os do arcabouço WS-ODL?
2. A configuração do arcabouço através do assistente é simples?
3. O roteiro das tarefas é fácil?
4. Na sua percepção, como o assistente organiza os passos de configuração de uma biblioteca digital? Em outras palavras, a que corresponde cada um dos passos da configuração através do assistente?
5. (O assistente organiza a tarefa de configuração em três passos básicos: (1) a configuração da biblioteca, (2) a configuração dos repositórios que irão armazenar o conteúdo da biblioteca e (3) a configuração dos serviços através dos quais a biblioteca irá oferecer o conteúdo armazenado em seus repositórios.) O que você acha dessa organização? Lógica? Intuitiva? Indicaria outra forma?
6. O que você acharia de uma forma de interação não-guiada (por outro lado, mais flexível), em que todos os componentes pudessem ser configurados sem uma ordem pré-estabelecida? Seria mais simples ou mais complicada?
7. Como um todo, o que você achou da interface da ferramenta? Fácil de usar? Confortável? Intuitiva? Apontaria algo que pudesse ser melhorado?

8. E o sistema de ajuda (*tooltips* com a descrição de cada parâmetro), você o utilizou? Foi útil?

## Apêndice C

# Problemas Detectados pelos Testes

Os problemas encontrados na avaliação de usabilidade do protótipo do ambiente WhizKEY são apresentados na Tabela C.1. Além dos problemas detectados nos testes com usuários, a Tabela C.1 inclui problemas detectados por meio da inspeção da interface do protótipo por uma especialista em Interação Humano-Computador (Albergaria e Santos, 2006), não necessariamente enfrentados pelos usuários nas sessões de teste conduzidas. Para cada problema, são apresentadas sua localização (referente ao módulo do protótipo que implementa o elemento de interface onde o problema ocorreu), descrição, gravidade e seu número de ocorrências ao longo dos testes. Para a avaliação de gravidade, utilizamos a classificação proposta por Nielsen (1993):

2. *problema catastrófico*, que impede a execução da tarefa;
1. *problema sério*, que atrapalha a execução da tarefa;
0. *problema cosmético*, que atrasa a execução ou irrita os usuários.

Tabela C.1: Problemas detectados por avaliações de usabilidade

id	Local	Descrição	Gravidade	Ocorrências
1	ConfigureDialog	A existência de ajuda contextual não é explícita.	1	12
2	ConfigureDialog	O rótulo “Abrir” é ambíguo, sugerindo que será carregada uma instância pré-configurada.	0	3
3	ConfigureDialog	O botão “Cancelar” é executado sem nenhuma confirmação adicional.	2	3
4	ConfigureDialog	Alguns parâmetros têm nomes fora do domínio dos usuários.	1	2
5	ConfigureDialog	Alguns parâmetros possuem nomes parecidos.	1	1
6	ConfigureDialog	A tecla ENTER não aciona o botão “OK” com o foco em outros controles.	0	1

Tabela C.1: Problemas detectados por avaliações de usabilidade (continuação)

id	Local	Descrição	Gravidade	Ocorrências
7	ConfigureDialog	A organização em abas e o botão “OK” habilitado dão idéia de não obrigatoriedade de se configurarem parâmetros de abas que não aquela em primeiro plano.	2	1
8	ConfigureDialog	A verificação de correção é feita globalmente, e não no nível de abas ou, mesmo, de parâmetros.	1	1
9	ConfigureDialog	A organização geral dos parâmetros em grupos é ambígua.	1	1
10	ConfigureDialog	Alguns rótulos referentes à configuração de caminhos não explicitam que os valores dos parâmetros correspondentes devem corresponder a caminhos.	1	0
11	ConfigureDialog	A janela de configurações dos parâmetros de um determinado serviço não identifica o serviço que está sendo configurado.	1	0
12	ConfigureDialog	A organização de abas em fileiras horizontais pode sugerir erroneamente uma relação entre abas.	1	0
13	ConfigureDialog	Os parâmetros de preenchimento obrigatório não aparecem destacados.	2	0
14	ConfigureDialog	As abas nas telas dos parâmetros possuem tamanhos diferentes, o que pode levar o usuário a achar que possuem relevâncias diferentes.	0	0
15	ErrorDialog	O texto das mensagens sobre parâmetros com valores inválidos às vezes refere-se a parâmetros em abas não visíveis.	2	8
16	ErrorDialog	O texto das mensagens não é claro.	2	8
17	ErrorDialog	O texto da mensagem de erro referentes a formatos inválidos de um determinado parâmetro não indica possível soluções.	1	4
18	FileChooser	A seleção de um diretório de dentro dele faz com que a última parte de seu caminho apareça duplicada.	0	7
19	FileChooser	O diálogo para seleção de diretórios não é aberto a partir do último diretório selecionado	0	4



Tabela C.1: Problemas detectados por avaliações de usabilidade (continuação)

<b>id</b>	<b>Local</b>	<b>Descrição</b>	<b>Gravidade</b>	<b>Ocorrências</b>
20	FileChooser	Diretórios e arquivos não têm representações distintas para os usuários, talvez acostumados com os símbolos do sistema operacional.	0	4
21	FileChooser	Tentativas de criação de pastas com nomes inválidos (segundo as regras do sistema operacional) criam pastas com o nome “New Folder” sem que o usuário perceba.	1	3
22	FileChooser	O acionamento da tecla ENTER retorna o diretório corrente, em vez de acessar esse diretório.	1	2
23	FileChooser	Não é possível mover, copiar e deletar pastas como no controle análogo provido pelo sistema operacional.	1	1
24	FileChooser	O diálogo não implementa localização, apresentando-se em idioma diferente daquele do restante da aplicação.	0	1
25	FinishPanel	Os links para os serviços da biblioteca configurada não aparecem na lista, conforme indicado pela mensagem de conclusão da tarefa.	2	0
26	InputDialog	O ícone desse janela é um ponto de interrogação, sugerindo erroneamente que se trata de uma ajuda contextual.	0	0
27	LibraryPanel	Uma biblioteca recém criada não é selecionada automaticamente, deixando o botão “Avançar” inabilitado, o que dificulta a continuidade da tarefa.	1	3
28	List	O controle tem aparência de editável, sugerindo que se deva clicar nele para se adicionar um item.	0	8
29	List	Itens inseridos em uma lista aparecem no final dela, não ficando visíveis para o usuário.	1	0
30	List	Os itens de uma lista não são ordenados alfabeticamente.	1	0
31	NavigationPanel	O botão “Avançar” não verifica a correção das instâncias em um determinado passo.	2	1
32	RepositoryPanel	A identificação do próximo passo não é clara, talvez pela quantidade de parâmetros associados ao repositório.	2	2

Tabela C.1: Problemas detectados por avaliações de usabilidade (continuação)

id	Local	Descrição	Gravidade	Ocorrências
33	RepositoryPanel	O rótulo “Adicionar” é ambíguo, talvez influenciado pela mensagem descritiva errônea deste passo.	0	2
34	RepositoryPanel	A mensagem descritiva do passo não suporta o caso de criação de novas bibliotecas, assumindo que existe ao menos um repositório criado.	0	2
35	RepositoryPanel	Este passo é visualmente semelhante ao LibraryPanel, o que dificulta a percepção de que se está em um novo passo.	0	1
36	RepositoryPanel	A descrição do que é um repositório é insuficiente para que os usuários entendam esse conceito.	1	0
37	ServicePanel	O acesso à configuração de parâmetros associados a cada serviço tem pouca visibilidade ou sugere que essa configuração seja opcional.	2	6
38	ServicePanel	O botão “Detalhes” fica desabilitado quando um serviço que não possui parâmetros a serem configurados é selecionado.	0	0
39	StepPanel	A descrição das ações a serem tomadas pelos usuários não é consistente entre os diversos passos.	1	0
40	WelcomePanel	A seqüência de passos a serem executados não é explicitada para os usuários.	1	0
41	WelcomePanel	O arcabouço sendo utilizado não é especificado.	1	0
42	WizardFrame	A implementação de ConfigurePanel como janela modal causa problemas quando a WizardFrame perde o foco, fazendo com que ConfigurePanel fique em segundo plano e a WizardFrame fique desabilitada.	2	4

---

<sup>1</sup>Problema detectado por um especialista a partir dos testes com usuários.