UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

MONTGOMERY BARROSO FRANÇA

Proposta e Implementação de uma Máquina de *Workflow* para o Projeto CEMT

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação

Prof. Dr. José Valdeni de Lima
Orientador

Porto Alegre, março de 2004

CIP-CATALOGAÇÃO NA PUBLICAÇÃO

França, Montgomery Barroso

Proposta e Implementação de uma Máquina de Workflow para o Projeto CEMT / Montgomery Barroso França. – Porto Alegre: Programa de Pós-Graduação em Computação, 2004.

97 f.: il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR-RS, 2004. Orientador: José Valdeni de Lima.

1. Workflow. 2. Máquina de workflow. 3. Protótipo. 4. XPDL. 5. WfMC. I. Lima, José Valdeni de. II. Título.

AGRADECIMENTOS

Quero começar agradecendo a Deus por estar terminando este trabalho. Agradeço pelo título que devo conquistar em breve e também pelas experiências e pelos amigos que conheci no Rio Grande do Sul. Agradeço a minha família: ao meu pai pelo exemplo de trabalho e de vida; a minha mãe por todo amor e apoio que me deu; a Irene e Dirce pelo conforto e sabedoria de suas palavras nos momentos mais difíceis; a Moacir e a Laura pelo espaço de trabalho em seu lar gentilmente cedido; e, enfim, a toda a minha família.

Agradeço ao Banco Central do Brasil pela licença remunerada que me possibilitou fazer este trabalho. Agradeço aos funcionários do Banco Central: Wallace P. Araújo e José Roberto de Oliveira, meus superiores imediatos, a José Félix Furtado de Mendonça, meu orientador técnico e a todos os funcionários dessa instituição que confiaram em mim e apoiaram a minha liberação.

Agradeço ao professor José Valdeni de Lima, sempre receptivo aos seus orientandos; e aos demais professores e funcionários do Instituto de Informática. Agradeço aos meus professores da UFMG, Clarindo Isaias P.S. e Pádua e Antônio Otávio Fernandes, pela confiança e pelas cartas de recomendação.

Agradeço aos meus colegas e amigos da universidade, que caminharam comigo e muito me ajudaram: Tiago, Rafael, Maximira, Niccholas, Leila, Igor, Luis e Tharso; à Erika Cota, pelos móveis emprestados; enfim a todos os colegas do Instituto com os quais convivi.

Agradeço a minha amiga Lizandra Bringhenti, que muito me ajudou na minha adaptação ao Rio Grande do Sul. Agradeço às crianças da Vila Bom Jesus e às crianças do Condomínio Jardim do Salso, pela companhia e pelo carinho e peço a Deus que todas elas tenham oportunidades tão boas na vida quanto as que eu tive e tenho.

SUMÁRIO

LISTA DE ABREVIATURAS

ACL	Access Control List
API	Application Programming Interface
AW	Amaya *Workflow*
BSF	Bean Scripting Framework
CEMT	Concepção de um Ambiente de Edição Cooperativa Multimídia com Tecnologia de *Workflow*
CLOB	Character Large Object
CSV	Comma separeted values
DBA	Database Administrator
DTD	Document Type Definition
DTML	Document Template Markup Language
FTP	File Transfer Protocol
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
JAAS	Java Authentication and Authorization Service
JDBC	Java Database Connectivity
JDK	Java Development Kit
JSP	JavaServer Pages
LALR	Look Ahead Left-Right
LAN	Local Area Network
LDAP	Lightweight Directory Access Protocol
LSA	Lotus Solution Architecture
NXD	Native XML Database
OFBIZ	Open For Business
RDBMS	Relational Database Management System
SQL	Structured Query Language
SVG	Scalable Vector Graphics
UFRGS	Universidade Federal do Rio Grande do Sul
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
WAN	Wide Area Network
WebDAV	Distributed Authoring and Versioning on the Web
WfMC	Workflow Management Coalition
XHTML	Extensible Hypertext Markup Language
XML	Extensible Markup Language
XML-DBMS	XML Database Management System
XML-RPC	Extensible Markup Language – Remote Procedure Call
XPDL	Extensible Markup Language Process Definition Language

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Este trabalho apresenta um protótipo de uma máquina de *workflow*, de uso geral, implementado em plataforma de software livre. O protótipo utiliza um servidor *web* com PHP, em sistema operacional Linux, alguns programas desenvolvidos em C e o banco de dados MySql. O projeto CEMT demanda o uso da tecnologia de *workflow*, com o objetivo de controlar a execução de cursos a distância. Antes de ser iniciado o desenvolvimento do protótipo, foi feito um estudo sobre algumas máquinas de *workflow* existentes, com o objetivo de encontrar alguma que tivesse licença livre e pudesse ser utilizada no projeto CEMT, ou colher subsídios para o desenvolvimento de uma máquina de *workflow* própria. Foram testadas duas máquinas de *workflow* de licença livre (Openflow e OFBIZ), uma máquina com cópia de demonstração (Reactor) e foram consultadas as documentações fornecidas pelos fabricantes. Além disso foi consultada também a documentação do Domino Workflow, que não disponibilizou cópia de avaliação e cuja licença não é livre.

Um dos requisitos do protótipo é a compatibilidade com os padrões de interface recomendados pela WfMC. Esses padrões permitem a interoperabilidade entre softwares de *workflow*. O primeiro benefício da adoção desses padrões é a interação com o editor gráfico de *workflow* AW (Amaya Workflow), desenvolvido no Instituto de Informática da UFRGS. Este editor gera definições de processos de *workflow* no formato da linguagem XPDL (XML Process Definition Language), que alimentam a máquina de *workflow*. O esquema XPDL foi traduzido para um esquema de banco de dados relacional e foi desenvolvido um compilador que lê um arquivo no formato XPDL e gera comandos SQL de inserção das informações desse arquivo no banco de dados. Foi desenvolvida uma interface *web* para demonstrar o funcionamento do protótipo. A API definida na Interface 2 da WfMC foi implementada parcialmente. Essa API permite o desenvolvimento independente de outras interfaces de usuário.

Foram propostas algumas extensões à Interface 1 e modificações na definição de estados recomendada pela Interface 2 da WfMC. Com isso foi possível aumentar o controle sobre a execução das instâncias de *workflow*. Foram incluídas as restrições de data e possibilidade de bloqueio na execução de instâncias de atividades. Outras extensões possibilitam um serviço de notificações e atividades em grupo e oferecem novas possibilidades de alocação de atividades. O funcionamento básico do protótipo é descrito e inclui as funcionalidades de carga da definição de processo, instanciação de processo, visualização da lista de trabalho e execução das atividades, entre outras.

Palavras-Chaves: *workflow*, máquina de *workflow*, protótipo, XPDL, WfMC.

Proposal and Implementation of a Workflow Engine for the CEMT Project

ABSTRACT

This work presents a workflow engine prototype, of general use, implemented in an open-source platform. This prototype uses a web server with PHP, a Linux operating system, some C-developed programs and the MySql database. The CEMT project demands the use of workflow technology with the goal of controlling the execution of distance courses. Before the start of this prototype development, some existent workflow engines were researched, whit the goal of finding an open-source engine viable to be used in the CEMT project, or getting aid for this workflow engine development. Two open-source workflow engines (Openflow and OFBIZ) and a workflow engine with demonstration copy (Reactor) were tested and the documentations provided by the vendors were consulted. Furthermore, the Domino Workflow documentation was also consulted. Domino Workflow is not an open-source software and did not provide a demonstration copy.

One of the requirements of the prototype is the compatibility with WfMC interface standards. These standards allow the interoperability of workflow software. The first benefit of the adoption of these standards is the interaction with the workflow graphic editor AW (Amaya Workflow), developed in the Informatic Institute of UFRGS. This editor creates workflow process definitions in the XPDL (XML Process Definition Language) format, which can be used by the workflow engine. The XPDL schema was translated to a relational database schema and a compiler was developed which reads the XPDL-format file and creates SQL commands that, furthermore, insert this file information into the database. A web interface was developed to demonstrate the prototype operation. The API defined in the WfMC Interface 2 was partially implemented. This API allows the independent development of other user interfaces.

Some extensions to the Interface 1 and some changes in the state definitions recomended by Interface 2 were proposed. These extensions make possible to improve the workflow instances execution control. Restrictions concerning date and the possibility of locking were included in the activity instances' execution. Other extensions allow a notification service and group activities and offer new possibilities of activities allocation. The basic prototype operation is described and this description includes the process definition upload, the process instantiation, the worklist view, the activity's executions and other functions.

Keywords: workflow, worklfow engine, prototype, XPDL, WfMC.

Proposition et Implementation d' une machine de Workflow pour le Projet CEMT

RÉSUMÉ

Ce travail présente un prototype d' une machine de *workflow*, d' utilisation général, implémenté en plateforme de logiciel libre. Le prototype utilise un serveur *web* avec PHP, en système opérationnel Linux, quelques programmes développés en C et une base de donnés MySql. Le projet CEMT demande l' utilisation de la technologie de *workflow*, avec l' objectif de contrôler l' éxecution de cours à distance. Avant de commencer le développement du prototype, un étude sur quelques machines de *workflow* existantes a été fait, avec l' objectif de trouver quelqune qu'ait licence libre et puisse être utilisée au projet CEMT, ou trouver des aides pour le développement d' une machine de *workflow*. Deux machine de *workflow* de licence libre (Openflow et OFBIZ) et une machine avec copie de démonstration (Reactor) on été testées et ses documentations, fournies pour les fabricants, ont été examinées. De plus, la documentation de Domino Workflow, qui n' a pas de licence libre et ne disposait pas de copie de démonstration, a été examinée.

Un de les réquisits du prototype est la compatibilité avec les standards de la WfMC. Ces standards permettent l'interoperabilité entre les logiciels de *workflow*. Le premier bénéfice de l' adoption de ces standards est l' interaction avec l' éditeur graphique AW (Amaya Workflow), développé au Institute d'Informatique de la UFRGS. C'éditeur produit des definitions de procès de *workflow* au format de la langage XPDL (XML Process Definition Language), qui alimentent la machine de *workflow*. Le schéma XPDL a été traduit en un schéma de base de donnés relationnel et un compilateur a été développé. Ce compilateur lit un fichier au format XPDL et produit des instructions SQL d' insertion de les informations de ce fichier à la base de donnés. Une interface *web* a été développée pour démontrer l'opération du prototype. L'API définie à l'Interface 2 de la WfMC a été implementée partiellement. Cette API permet le développement independent d' autres interfaces de l'utilisateur.

Quelques extensions à l'Interface 1 et quelques modifications à la definition de les états recomandée pour l'Interface 2 de la WfMC ont été proposées. En conséquence, le contrôle sur l'exécution de les instances de *workflow* a été augmenté. Des restrictions de date et la possibilité de blocage à l'exécution de les instances de les activités ont été inclues. D'autres extensions permettent un service de notification et des activités en groupe et offrent des neuves possibilités d' attribution de les activités. L' opération basique du prototype est décrit et inclut des fonctionnalités de la charge de la definition de procès, la création de les instances de procès, la visualisation de la liste de travail, l'exécution de les activités et autres.

Mots-Clées: *workflow*, machine de *workflow*, prototype, XPDL, WfMC.

1 INTRODUÇÃO

Uma máquina de *workflow* deve receber uma definição de processo de *workflow* e executá-la. Essa definição é constituída basicamente de atividades, transições, participantes e chamadas a aplicações. Executar esse processo significa alocar as atividades aos participantes, controlar a ordem de execução destas atividades, de acordo com a precedência estabelecida, disponibilizar os recursos e controlar outras restrições para suas execuções, tais como a data/horário para início, término ou tempo máximo de execução.

Além disso, a máquina de *workflow* pode integrar as atividades que dependem de interação humana com aquelas que podem ser totalmente automatizadas pelos sistemas de computação. O fluxo de informação e dados gerado por uma atividade pode ser automaticamente reencaminhado para outra atividade que o tenha como entrada. Isso evita o trabalho humano de receber uma saída de uma aplicação e alimentar a próxima aplicação com esses dados.

As atividades podem ser externas ou integradas ao sistema de computador. No primeiro caso pode cer citado como exemplo uma atividade que diz ao participante para postar uma correspondência no correio. Já no segundo caso, as atividades chamam aplicações, que podem rodar com ou sem a interação com uma pessoa. Essas aplicações podem enviar um correio eletrônico, encaminhar um formulário, etc.

O capítulo 2 faz um estudo sobre algumas máquinas de *workflow* existentes: o Openflow, o Reactor, o OFBIZ e o Domino Workflow. O objetivo foi colher subsídios para o desenvolvimento de um protótipo de máquina de workflow. A maioria dessas máquinas implementa ou usa um servidor de Internet e disponibiliza uma interface HTML para os usuários. Foram vistas algumas implementações em Java, JSP e Python.

Sistemas de *workflow* podem ser utilizados em ínumeros tipos de organizações e, mais do que isso, podem ser utilizados no relacionamento entre organizações. Com o crescimento do comércio eletrônico, a automatização de tarefas pode ser implementada com a utilização de sistemas de *workflow*. A WfMC (Workflow Management Coalition) (WfMC-Workflow Management Coalition, 2002) é uma organização internacional, sem fins lucrativos, que objetiva promover e desenvolver o uso de *workflow* através do estabelecimento de padrões para a terminologia de software, a interoperabilidade e a conectividade entre produtos de *workflow*. Quando os processos de *workflow* extrapolam as fronteiras de uma organização, é importante usar padrões para que os sistemas de *workflow* dessa organização possam interoperar com os sistemas de *workflow* de outra. Além disso, o uso de padrões torna possível o uso de componentes do sistema que sejam produzidos por fabricantes diversos. A WfMC desenvolveu uma estrutura (HOLLINGSWORTH, 1995) para o estabelecimento de padrões de interoperabilidade e padrões de comunicação que permitirá a coexistência e a

interoperação de vários produtos de *workflow*. Essa estrutura (figura 1.1) é composta de cinco interfaces principais (WfMC: PUBLISHED DOCUMENTS, 2003), a seguir:

Interface 1 – Process Definition Interchange.
Interface 2 – Workflow Client Application Application Programming Interface.
Interface 3 – Invoked Applications.
Interface 4 – Interoperability.
Interface 5 – Audit Data Specifications.

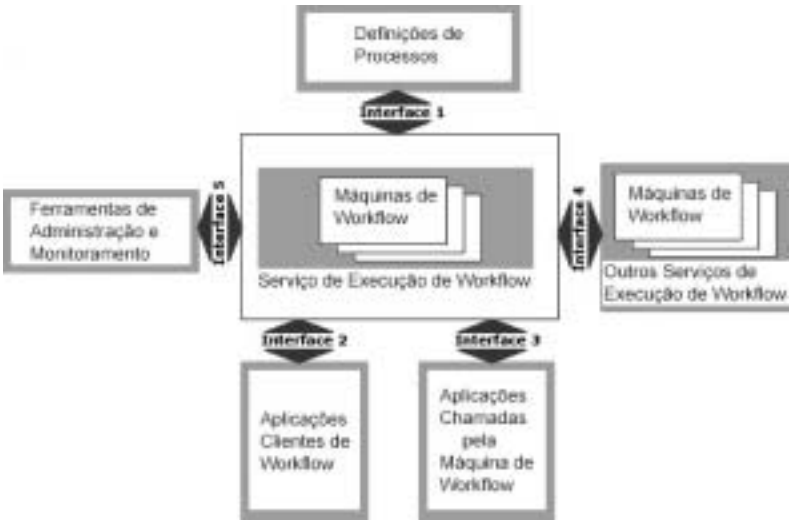


Figura 1.1: Diagrama do modelo de referência do *workflow* - WfMC

A interface 3 foi fundida à interface 2. Existem grupos de trabalho desenvolvendo as interfaces 6, 7 e 8, que ainda não têm nomes. Cada interface pode ser detalhada em mais de um documento. As interfaces 1, 2, 4 e 5 já possuem alguns documentos publicados, sendo que as interfaces 1 e 4 já dispõem de alguns documentos em sua versão final. Informações atualizadas sobre essas interfaces são disponíveis em (WfMC: PUBLISHED DOCUMENTS, 2003).

O objetivo deste trabalho é a especificação e implementação de um protótipo de uma máquina de *workflow*, de uso geral, e que seja compatível com os padrões das interfaces 1 e 2, definidos pela WfMC. Algumas funcionalidades adicionais foram propostas, que podem ser úteis para uma aplicação de controle de execução de cursos a distância, um dos objetivos do projeto CEMT (Concepção de um Ambiente de Edição Cooperativa Multimídia na Web com Tecnologia de *Workflow*) (LIMA et al., 2001).

A plataforma escolhida para a implantação da máquina de *workflow* é um servidor Linux, o banco de dados MySQL e o PHP, seguindo a filosofia do projeto CEMT de utilizar software livre. As definições de processos, assim como as instâncias e as informações sobre os usuários, são armazenadas neste banco de dados.

Em relação ao ambiente de implementação, objetiva-se um ambiente que seja simplificado e portável, uma vez que os cursos a distância podem ser cursados por pessoas de diferentes lugares, com plataformas de computador diferentes. A interface *web* portanto é naturalmente adequada a este tipo de aplicação.

O capítulo 3 apresenta o editor gráfico de workflow AW (Amaya Workflow), em desenvolvimento no Instituto de Informática da UFRGS por (TELECKEN et al., 2002), também compatível com a Interface 1 da WfMC (WfMC: WORKFLOW PROCESS DEFINITION INTERFACE, 2002), e mostra como as definições de processo produzidas por esse software são interpretadas e armazenadas na máquina de workflow. Para atingir esse objetivo, foi desenvolvido um compilador XPDL e feito um mapeamento entre o esquema XPDL e o esquema de um banco de dados relacional. Embora já existam softwares que façam o trabalho de *parsing* de arquivos XML, validando-os em relação a DTD's ou esquemas, as ações a serem tomadas são dependentes do esquema XPDL. Sendo assim, mesmo que esses *parser*s fossem utilizados, demandariam numerosas adaptações em seus códigos. Isso justificou o desenvolvimento de um compilador específico para o XPDL.

O capítulo 4 descreve os estados internos da máquina de workflow e um subconjunto das funções definidas pela Interface 2 (WfMC: WORKFLOW CLIENT APPLICATION, 1997), que foi implementado. Estas funções provêem as seguintes funcionalidades:

• Carregar uma definição de processo.
• Ver a lista de definições de processos.
• Ativar uma definição de processo.
• Instanciar uma definição de processo
• Mudar o estado de uma instância de processo.
• Pegar a lista de atividades de um usuário (*worklist*).
• Executar uma aplicação de uma atividade.
• Informar o término de uma atividade.
• Cancelar uma atividade.
• Ver o histórico das atividades executadas.

O capítulo 5 descreve a arquitetura da máquina, mostra algumas extensões feitas no modelo de definição de processos e descreve o funcionamento da máquina. É mostrado um exemplo de utilização do protótipo, ilustrado com algumas telas da interface do usuário. Trata-se de uma interface gráfica simples, desenvolvida com o único propósito de demonstrar o funcionamento da máquina.

O capítulo 6 apresenta as conclusões e os trabalhos futuros que podem ser feitos. O Apêndice A mostra como a máquina de *workflow* pode ser utilizada para controlar a execução de cursos.

2 MÁQUINAS DE *WORKFLOW* PESQUISADAS

A decisão de implementar uma máquina de *workflow* própria deve-se, primeiramente, à flexibilidade de testar novas funcionalidades, pelo fato de ter o domínio do código, e, em segundo lugar, à necessidade de trabalhar com software livre. Existem hoje inúmeras máquinas de *workflow*, variando em complexidade e ambientes de implementação. Foi feito um estudo sobre algumas máquinas de *workflow* existentes, com o objetivo de encontrar alguma que tivesse licença livre e pudesse ser utilizada no projeto CEMT, ou colher subsídios para o desenvolvimento de uma máquina própria, decisão que acabou sendo tomada. Foram testadas duas máquinas de *workflow* de licença livre (Openflow e OFBIZ), uma máquina com licença de demonstração (Reactor) e foram consultadas as documentações fornecidas pelos fabricantes. Além disso foi consultada também a documentação do Domino Workflow, cuja licença não é livre e não foi obtida cópia de avaliação. Os testes foram efetuados em um computador com o sistema operacional Windows 2000 Advanced Server[1].

2.1 Openflow

Openflow (OPENFLOW,2002) (OPENFLOW, 2003) é uma máquina de *workflow* de código aberto, escrita em Python, e funciona como uma aplicação do Zope. O Zope é um servidor de aplicações de código aberto, especializado em gerenciamento de conteúdo, portais e aplicações personalizadas (ZOPE, 2003). Ele inclui serviços próprios de HTTP (Hypertext Transfer Protocol), FTP (File Transfer Protocol), WebDAV (Distributed Authoring and Versioning on the Web) e XML-RPC (Extensible Markup Language – Remote Procedure Call), mas pode também ser usado junto com o Apache ou outros servidores WEB. Uma vez que o Openflow trabalha integrado com o Zope, esse último serve de camada intermediária entre o Openflow e o sistema operacional.

Para instalar o Zope, deve-se executar o arquivo de instalação específico para o sistema operacional. No caso dos sistemas operacionais Windows NT, Windows 2000 Server ou Windows XP, existem duas opções para disparar a execução do Zope, sendo uma delas a manual e a outra através da criação de um serviço a ser rodado pelo sistema operacional. Essa escolha deve ser feita quando da instalação do Zope. No teste realizado, foi escolhida a segunda opção. Também são pedidos o nome de uma conta e uma senha do usuário que será o primeiro administrador do Zope. A versão instalada foi a 2.6.1.

[1] A escolha do sistema operacional foi feita de acordo com a disponibilidade e familiarização com o ambiente.

Uma vez disparado o serviço do Zope, este disponibiliza páginas HTTP na porta 8080 do servidor. Para acessar a página inicial (ou do diretório raiz) do Zope (fig. 2.1) deve-se abrir, com o navegador, o endereço http://<*nome_do_servidor*>:8080/manage, onde <*nome_do_servidor*> deve ser substituído pelo nome do servidor onde foi instalado o Zope, como pode ser visto no campo *Address* (fig. 2.1). Esta página é composta por três quadros. O quadro superior mostra o nome do usuário autenticado e permite, através de uma pequena lista de opções, acessar uma tela de instruções, configurar preferências de visualização ou encerrar a sessão. O quadro inferior esquerdo mostra a estrutura de navegação dentro do sítio, em forma de árvore. Finalmente, o quadro inferior direito é o quadro principal, que mostra o conteúdo do documento. Quando selecionado o diretório raiz no quadro inferior esquerdo, o quadro principal mostra todos os subdiretórios e aplicações do Zope. Esse quadro mostra várias opções de menu como *Contents*, *View*, *Properties*, *Security*, etc, que podem ser herdadas e redefinidas pelas aplicações que trabalham com o Zope.

2.1.1 Instalação

A instalação do Openflow é bem simples. Primeiramente, deve-se expandir o arquivo de instalação e copiar a pasta resultante para a pasta /lib/python/products, que é criada pela instalação do Zope. A versão do Openflow usada nos testes foi a 1.0.6. Depois o serviço do Zope deve ser reiniciado para que o Zope reconheça o Openflow.

Acessando a página inicial do Zope, nota-se, no quadro principal, uma lista de tipos de aplicações que podem ser inseridas no Zope (fig. 2.1). Para inserir uma aplicação do Openflow, deve-se selecionar Openflow na lista e clicar no botão *Add*. Uma próxima página pede o nome da aplicação a ser criada e apresenta um botão para que o usuário complete a operação.

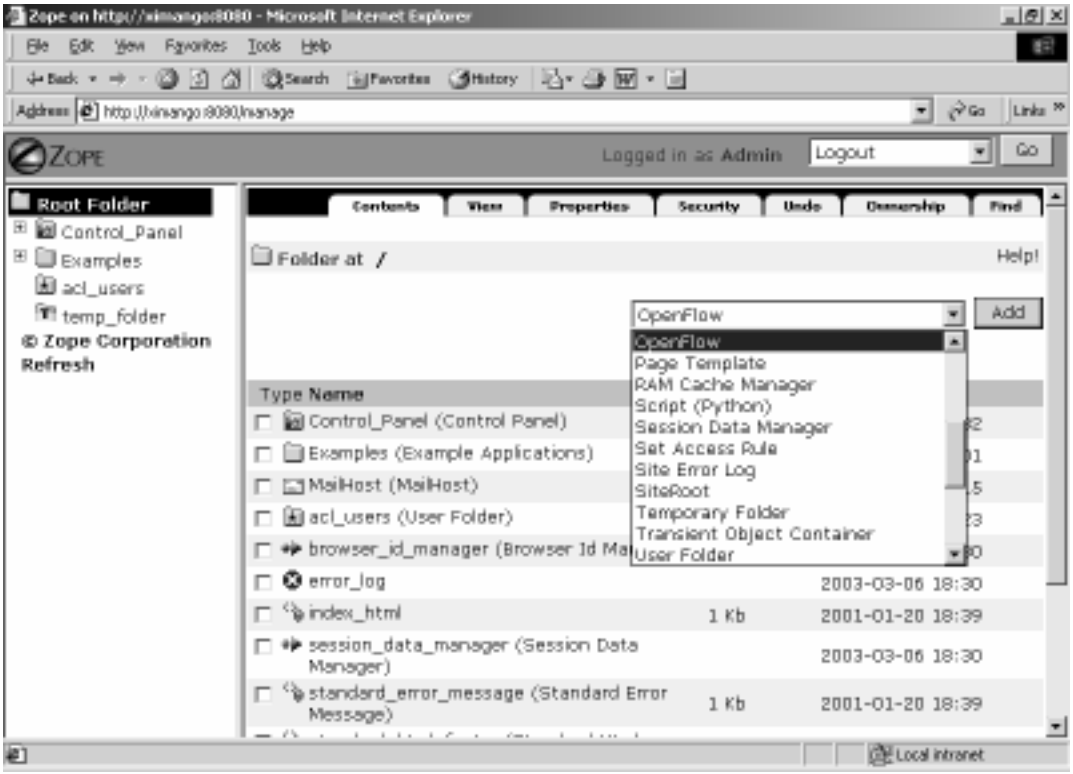


Figura 2.1: Página inicial do Zope

Uma nova aplicação do tipo Openflow (nesse exemplo com o nome de MeuWorkflow). é listada na página inicial do Zope, no quadro principal (4ª. linha) e no quadro de navegação (3ª. linha), como pode ser visto na fig. 2.2. Neste ponto a instalação do Openflow está finalizada.

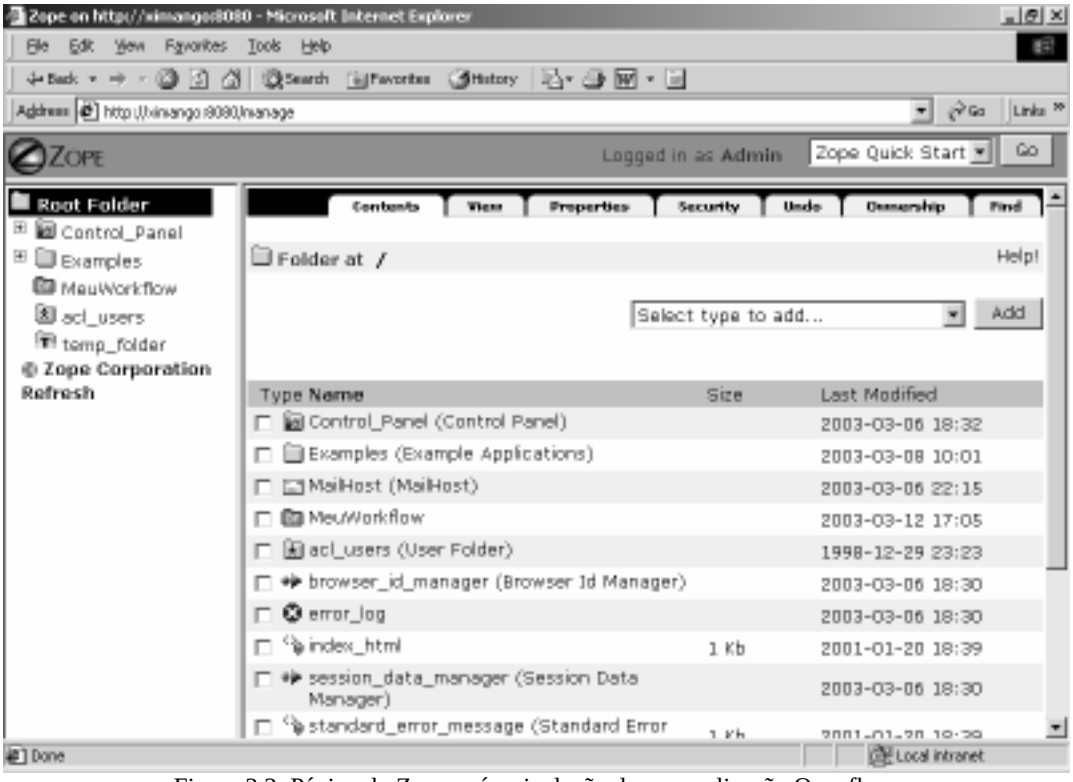


Figura 2.2: Página do Zope após a inclusão de uma aplicação Openflow.

2.1.2 Utilização do Openflow

O Openflow aproveita a gerência de usuários feita pelo Zope. Dessa forma, os mesmos usuários podem ser utilizados em várias aplicações. A criação de usuários é feita abrindo-se a pasta *acl_users*. Ao clicar o botão *Add*, aparece uma tela (fig 1.3) onde devem ser preenchidos o nome do usuário, a sua senha e, opcionalmente, o domínio. Podem ainda ser atribuídos papéis ao usuário.

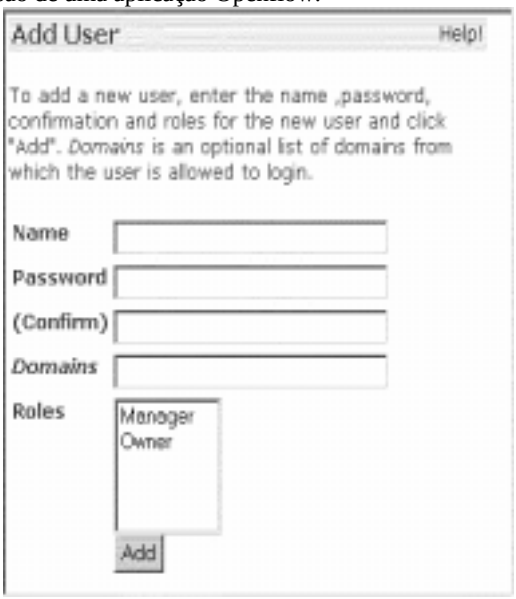


Figura 2.3: Criação de usuários no Zope.

Para começar a usar o Openflow, deve-se abrir a página inicial do Zope (fig. 2.2) e clicar na pasta de Openflow criada. É carregada uma tela (fig. 2.4) que apresenta várias opções de menu que serão explanadas a seguir.

Figura 2.4: Página da aplicação Openflow em sua opção de menu padrão: *Worklist*

A opção de menu Applications possibilita a definição de chamadas a aplicações. As aplicações devem ser previamente adicionadas ou criadas no Zope e devem estar em uma URL (Uniform Resource Locator) acessível a partir do diretório raiz do Zope. Aplicações podem ser, entre outras, páginas HTML (Hypertext Markup Language), *script*s Python, formulários DTML (Document Template Markup Language), consultas SQL (Structured Query Language), aplicações do Zope ou documentos associados a um editor de texto. Para definir a chamada de uma aplicação no Openflow, deve-se clicar no botão *Add Application* da opção de menu *Applications*. Na próxima tela (fig. 2.5) devem ser informados um nome pelo qual a aplicação será referenciada no Openflow e o caminho e nome da aplicação no Zope. O caminho informado refere-se ao diretório do Openflow, de tal modo que "../" refere-se ao diretório pai, que é o raiz do Zope.

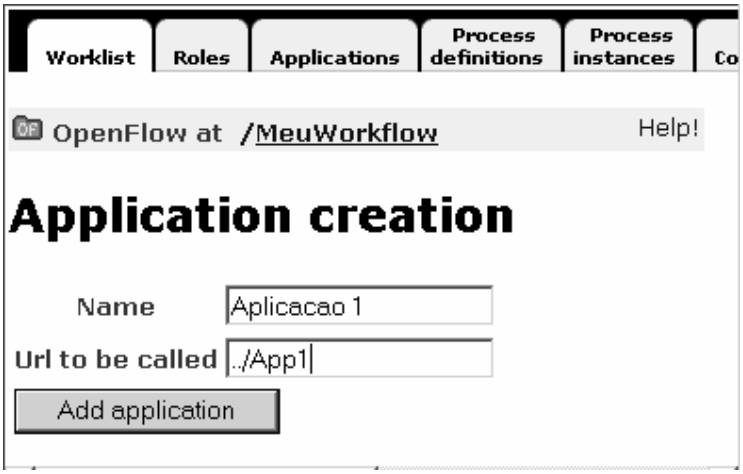


Figura 2.5: Definição da chamada de uma aplicação.

Na opção de menu *Process Definition* é feita a definição de processos. Clicando no botão *Add process definition*, abre-se uma tela (fig. 2.6) onde devem ser informados uma identificação da definição do processo (Id), um título e uma descrição. Existe ainda uma opção que quando marcada faz com que as atividades *Begin* e *End* padrões sejam criadas automaticamente. Por fim, pode-se definir o nível de prioridade do processo. Uma vez criado o processo, a opção de menu *Process Definition* é novamente exibida, desta vez com um *hyperlink* para a edição da definição do processo recém-criado (fig. 2.7). O próximo passo que deve ser feito é a edição da definição do processo para acrescentar atividades e transições. Clicando-se neste *hyperlink*, abre-se a tela de definição do processo onde existem botões para passar às telas de adição de atividades e adição de transições.

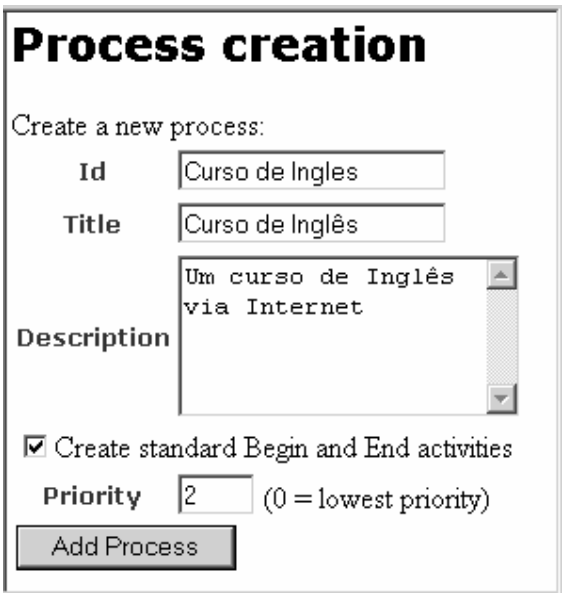


Figura 2.6: Criação de uma definição de processo.

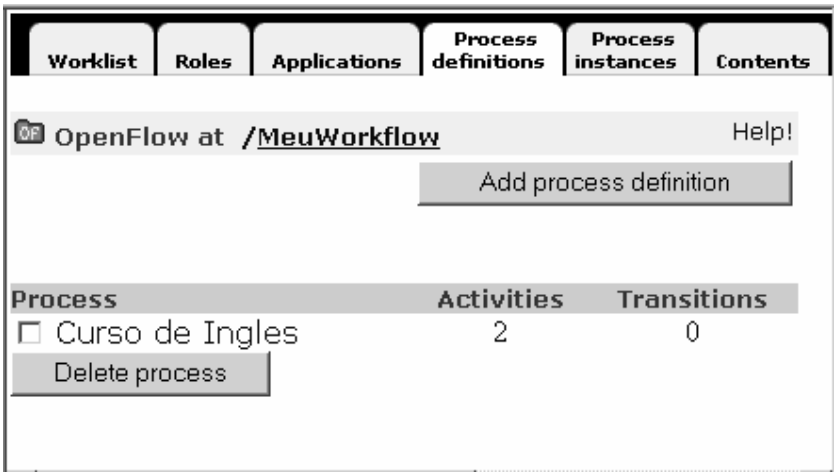


Figura 2.7: Opção de menu Process Definition.

Na tela de adição de atividades (fig. 2.8) devem ser informados a identificação (ID) da atividade, o título e uma descrição. Três tipos de atividades são possíveis de serem criadas:

- *Dummy*: são atividades que não fazem nada, mas que definem o início ou o fim de um roteamento do fluxo. Quando marcada essa opção, deverá também ser escolhido o tipo de roteamento *AND* ou *OR* a ser aplicado aos tipos *Join Kind* ou *Split Kind.*

- *Application*: são as atividades que disparam uma aplicação do Zope. São oferecidas algumas opções para fazer isso, como *Automatic start*, *Automatic finish* e *pushing application*. Quando marcada a *Automatic start*, a atividade não é alocada a nenhum usuário. A própria máquina de *workflow* se encarrega de disparar a aplicação. A opção de *Automatic finish* dispensa o usuário de informar que a aplicação terminou e a execução do *workflow* passa automaticamente para a próxima atividade. No entanto a aplicação deve ser programada para chamar a API (Application Programming Interface) *completeWorkitem* do Openflow para informar o seu término. Na opção *pushing application*, uma aplicação se encarrega de decidir qual será o usuário a fazer a atividade e aloca a atividade a este usuário.

- *Subprocess*: são atividades que chamam outro processo, previamente definido e que deve ser escolhido de uma lista.

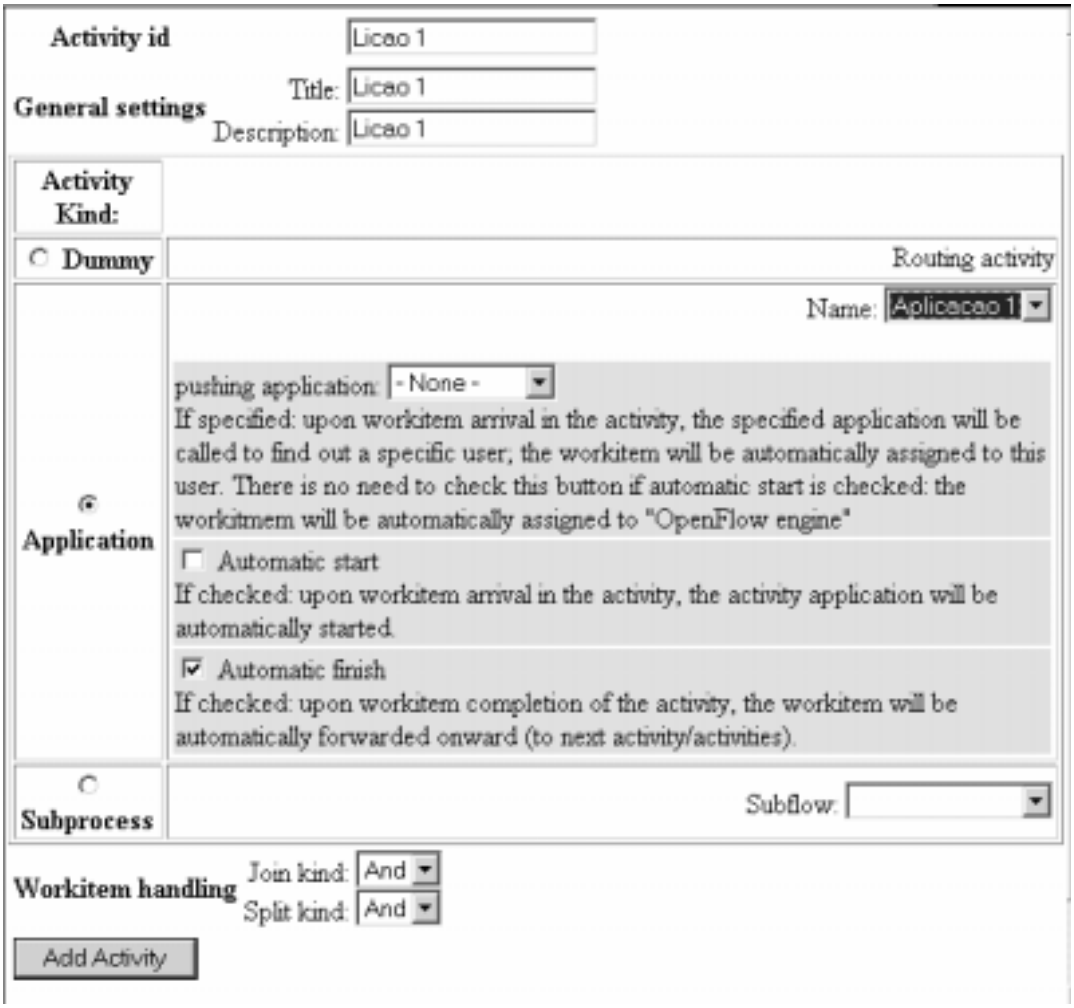


Figura 2.8: Tela de adição de atividade.

Uma vez criadas as atividades, a partir da tela de edição da definição do processo, prossegue-se com a criação das transições, clicando-se no botão *Add Transition*. A tela de criação de transições (fig. 2.9) pede um identificador (Id), uma condição lógica para sua execução, uma descrição e mostra duas listas de atividades de onde são escolhidas as atividades de origem e destino.

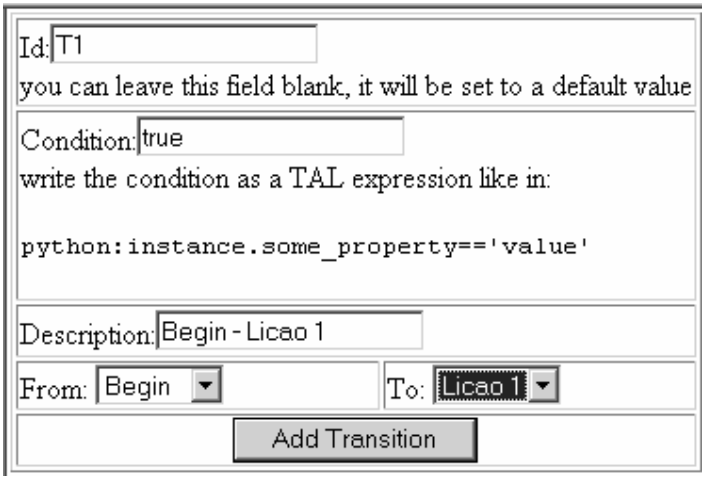


Figura 2.9: Tela de adição de transição.

Depois de inseridas as atividades e transições a tela de edição da definição do processo (fig. 2.10) mostra-as em uma tabela. Para as atividades são listadas as informações (quando aplicáveis) sobre o tipo de *join* e *split* usado (*And* ou *Or*), o nome da aplicação a ser chamada, os modos de início e fim (manual ou automático) e os subprocessos chamados. Para as transições são listadas as informações sobre identificador, condição de execução, origem e destino. Essa tela permite a edição, a remoção ou adição de novas atividades e transições. Não é provida nenhuma visualização gráfica da definição do processo.

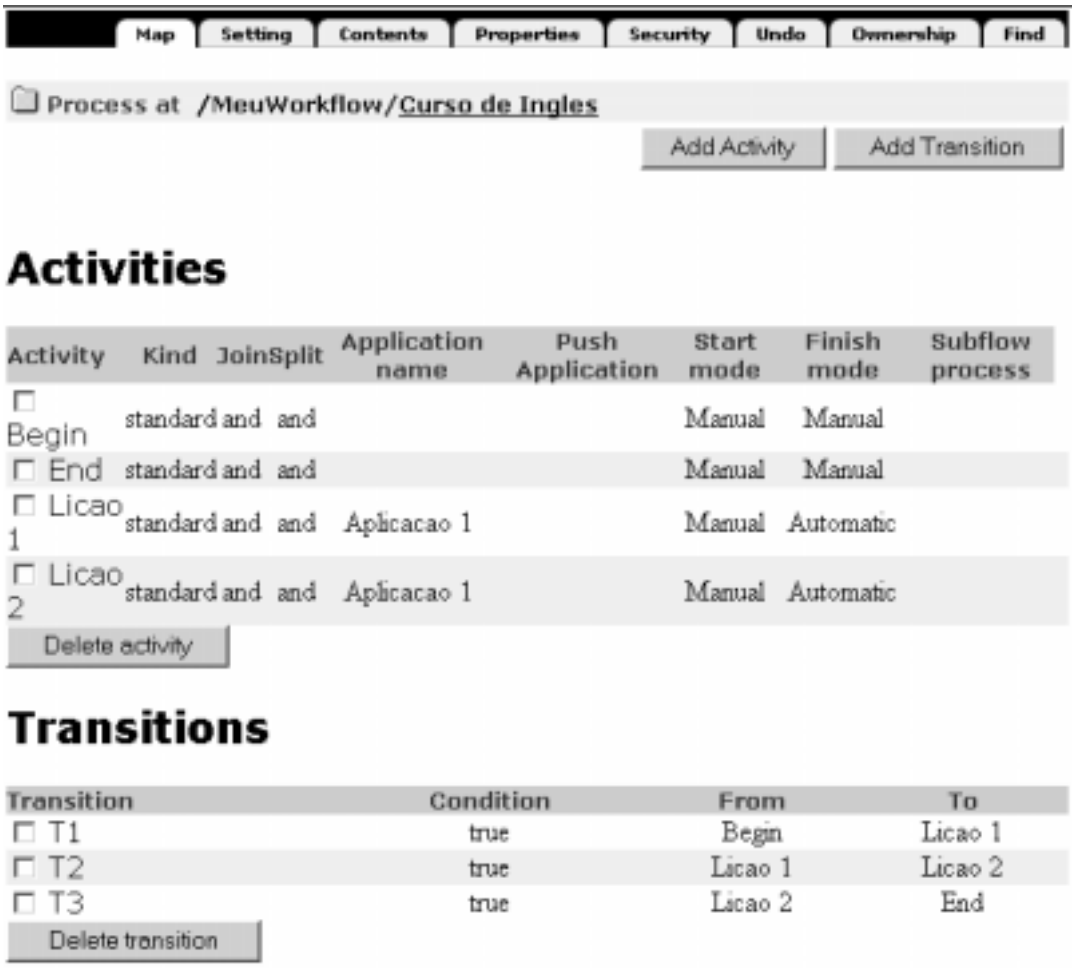


Figura 2.10: Edição da definição de um processo.

A opção de menu *Security* (fig. 2.11) é herdada do Zope e permite a criação de papéis e a gerência de permissões. Numa tabela pode-se marcar, para cada papel, cada permissão que lhe deve ser atribuída. Existem quatro papéis pré-definidos no Zope: *anonymous*, *authenticated*, *manager* e *owner*. Além disso, cada aplicação permite a criação de papéis (chamados de papéis definidos pelo usuário), que têm escopo local àquela aplicação. Quando esses papéis são definidos no diretório raiz do Zope, eles são propagados para todas as aplicações. O *hyperlink local roles* permite a atribuição de usuários a papéis.

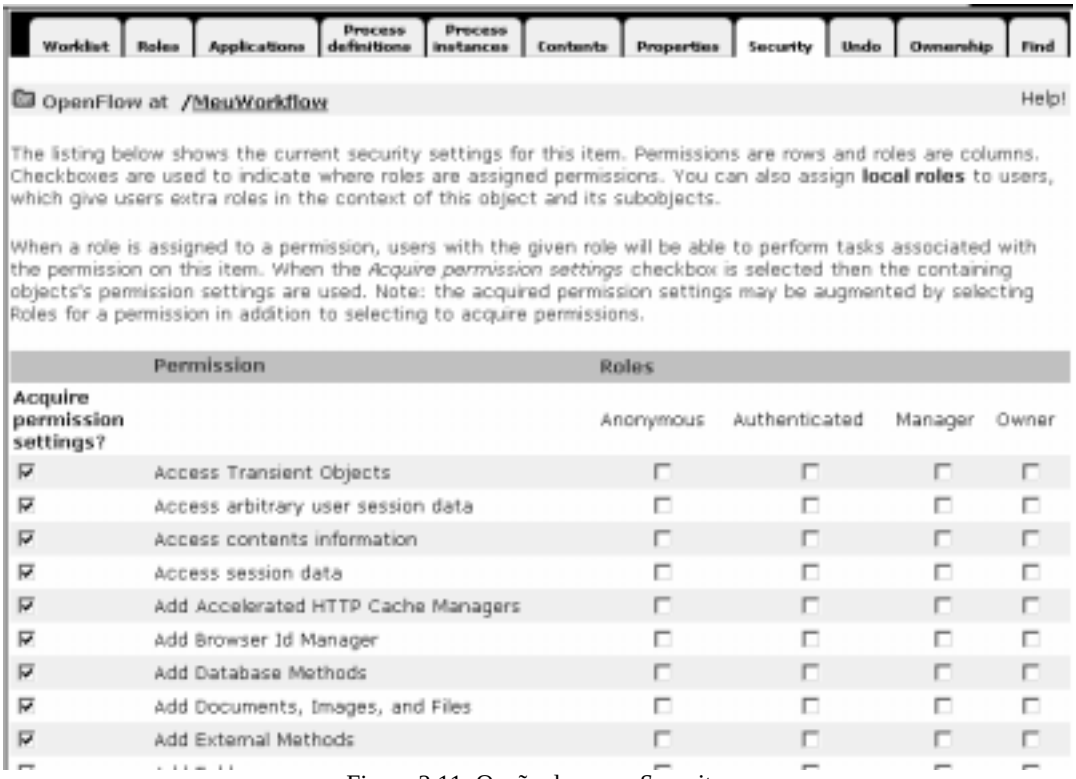


Figura 2.11: Opção de menu *Security*.

O último passo para a definição do processo é a atribuição de papéis a atividades. A opção de menu *Roles* (fig. 2.12) mostra uma tabela onde uma coluna de cabeçalho mostra os papéis existentes e uma linha de cabeçalho mostra os tipos de permissões que podem ser atribuídas que são *Pushable activities* e *Pullable activities*. As células da tabela contêm *hyperlinks* para a atribuição de papéis a atividades. Quando há alguma permissão já atribuída a um papel, seja de *Pushable activities* ou de *Pullable activities*, o *hyperlink* correspondente exibe o texto "*assigned*" e em caso contrário o texto "*not assigned*". Para dar permissão a usuários de um determinado papel para executarem uma atividade, deve-se clicar no *hyperlink* correspondente à linha do papel e à coluna *Pullable activities*. De maneira análoga, para permitir a usuários desse papel para redirecionarem uma atividade a outro usuário, deve-se clicar no *hyperlink* correspondente à linha do papel e à coluna *Pushable activities*. Após isso, é mostrada uma tela com a lista de atividades existentes, onde as atividades podem ser escolhidas e atribuídas àquele papel.

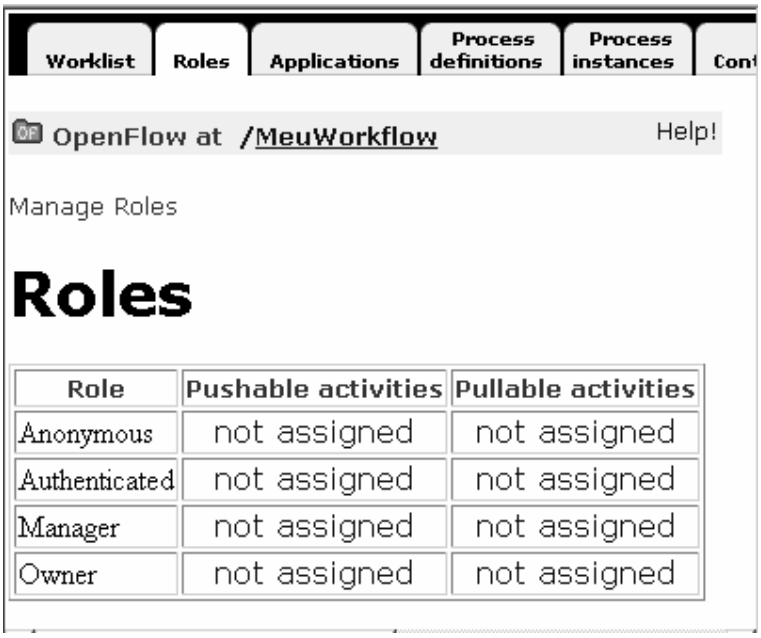


Figura 2.12: Atribuição de papéis a atividades.

A opção de menu *Process Instances* permite criar instâncias de processos e acompanhar a execução de instâncias em andamento. Para criar uma instância de processo é utilizado o botão *Add process instance*, que retorna uma tela para criação de instância (fig. 2.13). Nela são pedidos uma identificação para o processo (campo *Customer name*), um título e um comentário. Também é exibida uma lista das definições de processo existentes, para que uma possa ser escolhida. Para garantir uma identificação única para a instância de processo, o Openflow gera números únicos que são concatenados ao final da identificação fornecida.

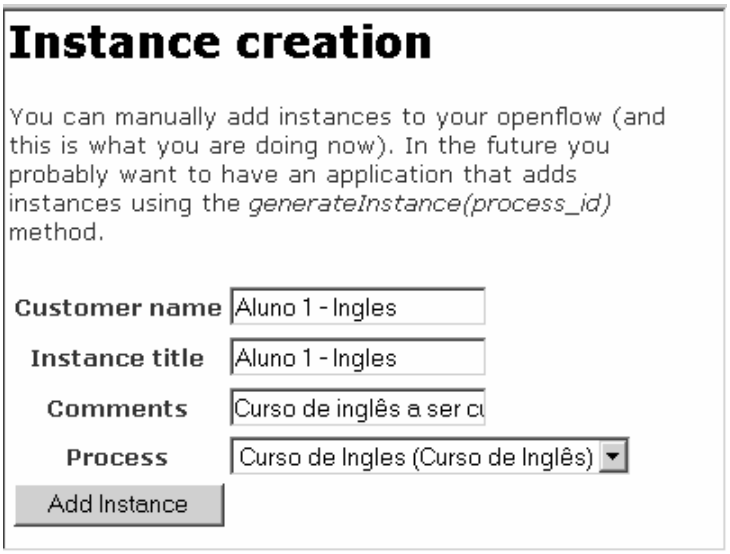


Figura 2.13: Criação de instância de processo.

Após a criação da instância, é adicionado um *hyperlink* para esta na opção de menu *Process definition*, a partir da qual pode-se abrir uma tela onde o administrador pode fazer todo o acompanhamento da execução da instância, vendo as atividades em execução e o histórico, além de poder fazer algumas intervenções. A primeira ação a ser

tomada é a ativação da instância. A partir daí, o Openflow mostra as instâncias de atividades a serem executadas na lista de atividades dos usuários (opção de menu *Worklist*) e lista todas as instâncias de atividades que iniciaram sua execução (fig. 2.14), com informações sobre qual foi a atividade anterior, a que horas a atividade foi instanciada, a quem está alocada a instância da atividade, a que horas a instância da atividade completou e para qual atividade seguiu o fluxo de execução. Se alguma instância de atividade estiver em um estado de exceção, um *hyperlink fallin* é mostrado, onde pode-se tratar a exceção. Clicando-se em *fallin*, pode-se desviar o fluxo da execução para qualquer outra atividade daquele ou de qualquer outro processo.

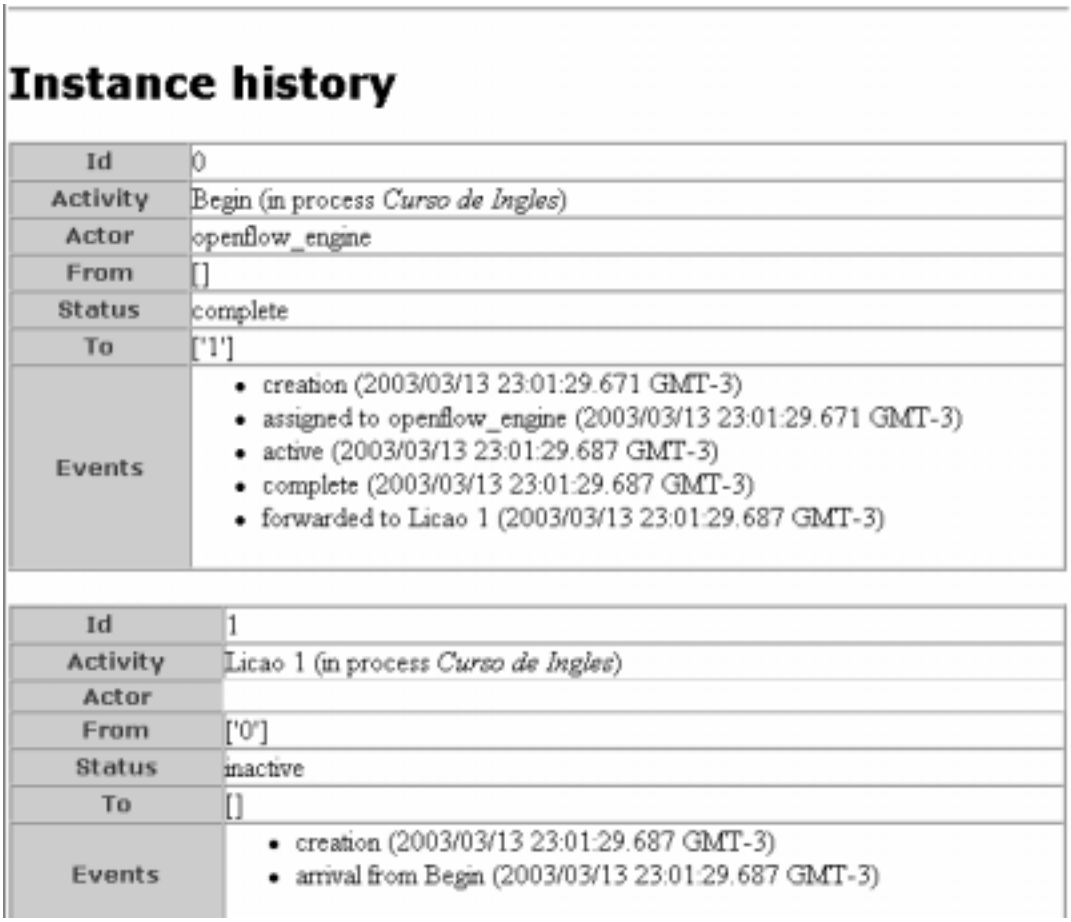


Figura 2.14: Acompanhamento da execução da instância de processo.

Na opção de menu *Worklist* são mostrados todos os processos existentes, estejam eles ativos ou não e todas as suas atividades, independentemente se o usuário tem permissão para executá-las ou não (fig. 2.15). As atividades são listadas em ordem alfabética, dentro de cada processo. A ordem dos processos não pôde ser determinada nos testes, mas não é alfabética, nem de prioridade, nem de criação. Para cada atividade é mostrada uma tabela que contém um cabeçalho e linhas correspondentes a cada instância ativa dessa atividade. O cabeçalho da tabela é mostrado mesmo que a atividade não tenha instâncias ativas. Pode haver várias instâncias de atividade ativas para cada atividade, uma vez que pode haver várias instâncias daquela definição de processo ativas. As linhas da tabela mostram o nome da instância do processo, o número da instância da atividade, o status, o ator da instância de atividade e, de acordo com a

permissão do usuário, *hyperlinks* para iniciar a atividade (*start*), atribuí-la a outro usuário (*assign*), ou gerar uma exceção (*exception*).

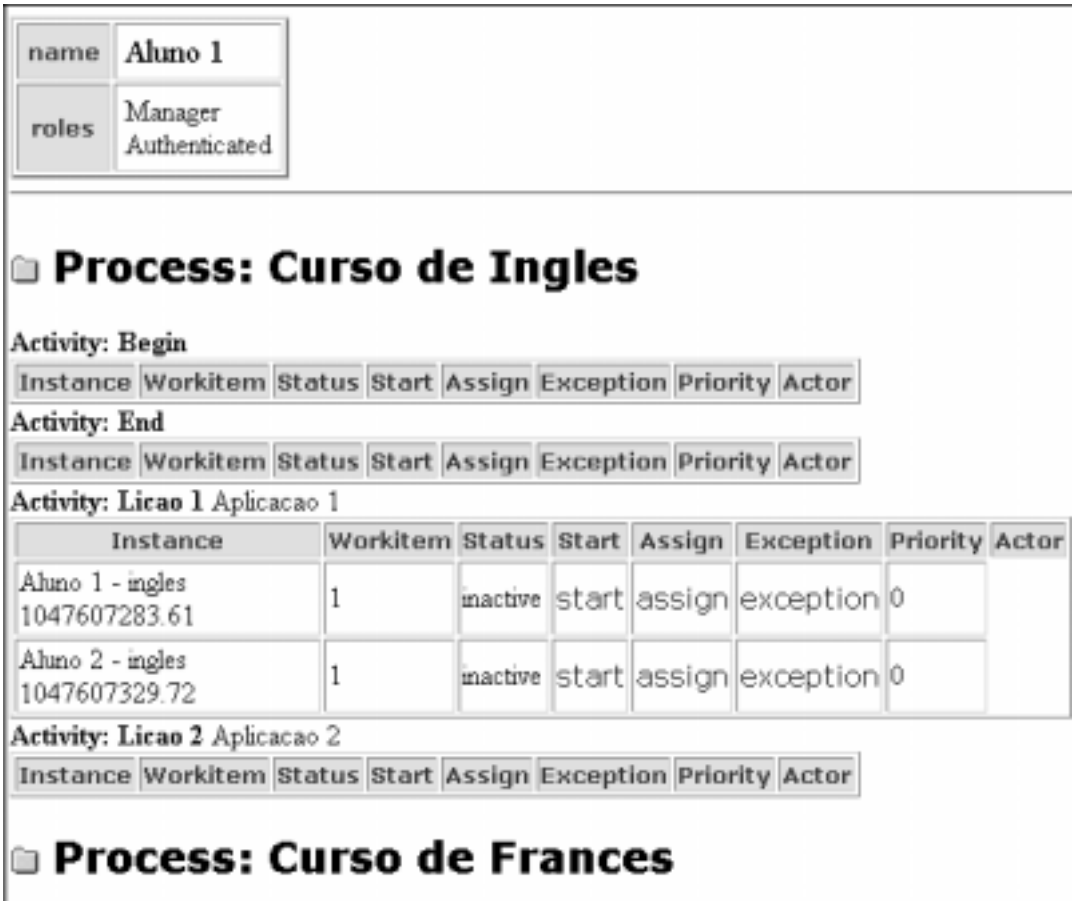


Figura 2.15: Opção de menu *Worklist*

2.1.3 Testes

Nos testes realizados, os usuários sem o papel *Manager* global (papel adicionado pela pasta *acl_users*), ainda que tivessem todas as permissões possíveis, não tiveram permissão para iniciar uma atividade que aparece na sua lista de atividades. Mesmo usuários com o papel *Manager* no Openflow ou na pasta raiz do Zope não puderam fazê-lo. A única opção permitida a esses usuários foi a de gerar uma exceção. A opção de menu *Worklist* não mostrou os papéis locais pertencentes ao usuário autenticado, nem os papéis que este possuía na pasta raiz do Zope. Os únicos papéis que são mostrados na *Worklist* são aqueles atribuídos via pasta *acl_users*. O papel *Manager* em uma aplicação local é diferente do papel *Manager* do escopo geral e diferente do mesmo na pasta raiz do Zope. Papéis criados na pasta raiz ficam disponíveis para serem usados na pasta *acl_users* e também como papéis locais em todas as outras pastas, mas a atribuição de usuários a esses papéis é feita de maneira totalmente independente.

Para executar uma atividade o usuário deve ser *Manager* global e a atividade deve estar atribuída ao papel *Manager*. Se o usuário pertencer ao papel Manager e a outro papel (por exemplo o papel Aluno) ao qual a atividade esteja atribuída, se a atividade não estiver atribuída ao papel *Manager*, o usuário não terá permissão de executá-la. Disso se conclui que os outros papéis não têm utilidade. Essas situações são de difícil

compreensão e não foi encontrada explicação na documentação. Esperava-se que um usuário sem o papel de *Manager* global, ou seja, com menor poder, pudesse executar as atividades de um *workflow*.

O redirecionamento de atividades não foi testado com sucesso, pois independentemente do papel estar relacionado às *Pushables activities*, o comportamento foi o mesmo. Para todos os usuários do papel Manager global era exibida o *hyperlink* para redirecionamento da atividade. Mas após a sua utilização, a atividade continuava disponível para o usuário original.

Ao disparar uma aplicação que era uma página HTML, ou um documento do Word, o Openflow apresentou a página no navegador, mas não atualizou no histórico da instância que a aplicação havia sido iniciada. Conseqüentemente, ela ficou também disponível para todos os outros usuários que tinham permissão para executá-la, causando execução em duplicidade. A aplicação precisa chamar a API completeWorkitem do Openflow para avisar que ela terminou.

O openflow pode exportar uma definição de processo para um arquivo XML (extensible Markup Language), em formato próprio.

2.1.4 Vantagens do Openflow

Trabalhando sobre o Zope, Openflow é multiplataforma. Se o Zope for disponibilizado para outro sistema operacional, nehuma mudança precisa ser feita no Openflow. Se o Zope for bem sucedido e bem difundido entre a comunidade, o Openflow pode tirar um grande proveito disso.

O Openflow herda muitos objetos do Zope, facilitando o seu desenvolvimento com esse reaproveitamento.

Utiliza a interface *web*, não havendo necessidade de instalar qualquer software adicional nos clientes (apenas o navegador).

Aproveita os usuários e papéis definidos pelo Zope e utiliza o serviço de autenticação deste.

Permite a redefinição do processo após sua execução e as instâncias ativas seguem a nova versão. Este tipo de implementação é denominado em (CASATI et. al, 1996) como migração para o *workflow* final.

Permite aos usuários dispararem exceções que podem depois ser tratadas pelo administrador. Mesmo não ocorrendo exceção o administrador pode alterar a execução de uma instância, desviando a execução para outras atividades do mesmo ou de outro processo e pode também cancelar ou finalizar uma instância. Isso provê bastante flexibilidade para resolver problemas.

É possível acompanhar o andamento da execução das instâncias de processos, de forma detalhada. O Openflow informa os horários de início e término das atividades, o caminho seguido pelo fluxo e os usuários executores, guardando inclusive um histórico das instâncias já completadas.

Permite a exportação e importação de uma definição de processo para/de um arquivo XML, embora utilize um formato próprio.

2.1.5 Desvantagens do Openflow

Não provê uma ferramenta gráfica para a criação da definição de processo ou de visualização desta.

Não é compatível com o padrão da Interface 1 – Linguagem de Definição de Processo (XPDL) estabelecido pela WfMC (WfMC: WORKFLOW PROCESS DEFINITION INTERFACE, 2002). Isso dificulta a interoperabilidade com outras ferramentas de definição de processo, como, por exemplo, a ferramenta gráfica desenvolvida por Telecken (TELECKEN et al., 2002).

As aplicações precisam ser programadas para informar à máquina sobre o seu início e término, através de chamadas de API's do Openflow.

Para que as aplicações sejam executadas elas precisam se referir a arquivos que estejam em uma URL a partir da instalação do Openflow, o que faz com que estes arquivos tenham que ser carregados pelo Zope.

Os arquivos a serem executados pelas aplicações não são do tipo executáveis. Em geral são documentos, aos quais o sistema operacional tem associado um aplicativo para abri-lo. Quando se coloca um arquivo com extensão .exe, é exibido um *prompt* para fazer o *download* do arquivo.

O uso de papéis não foi testado com sucesso. Apenas o papel *Manager* teve permissão para executar as atividades de um *workflow*.

2.2 Reactor

O Reactor (REACTOR, 2003) é uma máquina de *workflow* implementada em Java. Sua modelagem foi influenciada, mas não segue todos os padrões da WfMC. O Reactor não é um software livre e a licença instalada é de avaliação. Ele é composto basicamente de 3 componentes:

- Reactor Studio: é uma ferramenta gráfica para a criação de *workflow*. Gera arquivos XML que são carregados na máquina de *workflow*.

- Reactor Server: é a máquina de *workflow* em si. Ela usa um servidor *web* Apache Tom Cat e pode ser acessada em http://<nome_do servidor>:8080/ReactorPortal , onde <nome_do_servidor> é o nome da máquina onde é instalado o Reactor Server.

- Reactor Portal Framework: é um *framework* JSP/Servlet para desenvolvimento de interfaces de usuário baseadas na *web* para aplicações.

O Reactor é multiplataforma. Ele possui dois tipos de arquivos de lotes, para ambiente Windows (extensão .bat) e ambiente Linux/Unix (extensão .sh). Os testes foram feitos com a versão 5.0.7 em um sistema operacional Windows 2000 Advanced Server.

2.2.1 Instalação

O pré-requisito para a instalação é ter instalado o JDK (Java Development Kit) 1.3.1 ou superior. Quando se utiliza essa versão do JDK, deve-se expandir o arquivo javaExt.zip para o diretório %JAVA_HOME%/jre/lib/ext/. No caso da versão 1.4 do JDK, não há essa necessidade. Deve ser criada uma variável de ambiente com o nome JAVA_HOME, que deve guardar o caminho da instalação do JDK. É conveniente que esse comando seja acrescentado nos arquivos de lote `Reactor.bat` e `Studio.bat`. Para colocar o servidor no ar deve-se rodar o arquivo de lote `Reactor.bat`. É importante que a porta 8080 do servidor não esteja sendo utilizada por outro aplicativo.

2.2.2 Reactor Studio

Para rodar o Studio, deve-se rodar o `studio.bat`. O Studio pode ser rodado em uma outra máquina que não seja aquela em que o Server esteja rodando. O Studio se comunica com o Server via HTTP. A interface gráfica do Studio (fig. 2.16) mostra uma linha de menus, uma paleta para a construção dos elementos do processo e dois painéis, sendo o da esquerda utilizado para desenhar o processo de *workflow* e o da direita para editar suas propriedades.

A paleta é bem simples e é constituída de quatro símbolos: uma seta para seleção, um símbolo para criar atividades, um símbolo para criar atividades de roteamento e um símbolo para criar transições. Após desenhado o elemento do *workflow*, podem-se editar suas propriedades clicando-se nele com o botão direito do mouse. No painel à direita podem-se definir os operandos, os statuses, as listas de controle de acesso e as políticas para cada atividade ou transição.

Processos e atividades são tratados pelo Reactor como objetos do mesmo tipo. As atividades podem ser vistas como subprocessos. A árvore de objetos mostrada no painel direito do Studio (fig. 2.16) mostra que o processo e cada atividade podem conter objetos dos tipos operandos, statuses, políticas, ACL's (Access Control List) e atividades. Nodos que contém filhos são indicados pelo desenho de uma pasta colorida, precedida de um pequeno círculo. Quando não existe nenhum objeto daquele tipo, a simbologia usada é uma folha em branco.

Operandos são variáveis que armazenam valores úteis à execução do processo. Os statuses são usados para fazer com que uma transição seja executada. Cada atividade pode originar mais de uma transição, mesmo que essa atividade não seja uma atividade de roteamento. Para cada transição deve ser associado um status. Os nomes desses statuses aparecerão em botões na interface *web* do usuário, quando da execução do *workflow*. De acordo com o botão clicado pelo usuário, a execução prossegue para a transição correspondente. Esses statuses necessitam ser definidos antes de serem associados às transições, com exceção dos statuses *Finish*, *Cancel*, *Oakay* e *Done*, que já são pré-definidos. Políticas são objetos que associam uma instrução executável, como por exemplo um *script* BSF (Bean Scripting Framework) a uma troca de status de uma atividade, ou a um outro evento. Algumas políticas são fornecidas junto com a instalação. Podem ser citadas como exemplo uma política para comparar valores e uma política para enviar *e-mail*s aos usuários das atividades. Outras políticas podem ser

implementadas via programação. Os usuários que podem executar a atividade ou o processo são colocados na ACL.

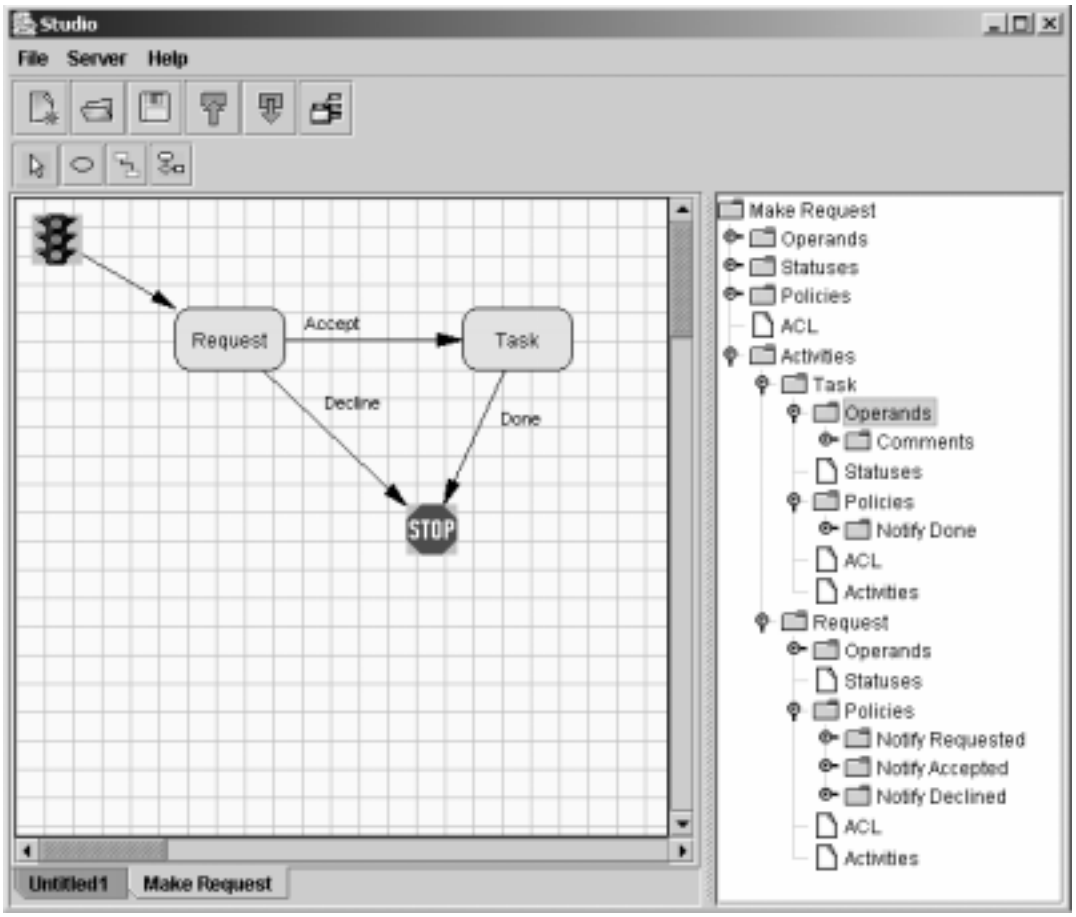


Figura 2.16: Interface do Reactor Studio mostrando um processo exemplo.

O Reactor não utiliza o conceito de papéis para agrupar usuários. Os papéis não são relacionados às atividades e processos. Estas são diretamente relacionadas aos usuários. Os papéis servem para qualificar essa relação. Por exemplo um usuário pode ser um *process invoker* de um processo, que é o usuário que cria uma instância do processo, ou ser um *initiative participant* de uma atividade, que é o usuário que executa a atividade. Podem ser criados grupos, mas não foi vista uma opção para preenchê-los. Além disso os grupos são criados dentro da definição dos processos, não sendo vistos externamente. O Reactor oferece três opções de autenticação: a mais simples é o uso de arquivos no formato CSV (comma separeted values), contendo o nome dos usuários, senhas e outras informações básicas como endereço de *e-mail*. Essa é a opção padrão que acompanha os arquivos de instalação. A segunda opção é usar o serviço de autenticação JAAS (Java Authentication and Authorization Service), que depende da instalação e configuração de um módulo JAAS. A terceira opção é usar um servidor LDAP (Lightweight Directory Access Protocol). Apenas a primeira opção foi testada.

O Reactor pode se conectar a bancos de dados que utilizem conexão via JDBC (Java Database Connectivity), como Oracle 8, Microsoft SQL Server 7, PostgreSQL 6.5 e InstantDB.

A linha de menus apresenta três opções: *File*, *Server* e *Help*. O menu *File* apresenta as opções para criar novos documentos, abrir documentos existentes, salvar o documento, fechar o documento e encerrar o Studio. O Studio salva as definições de processos em arquivos do tipo XML, em padrão próprio, não seguindo as definições da Interface 1 da WfMC (XPDL) (WfMC: WORKFLOW PROCESS DEFINITION INTERFACE, 2002). O menu *Server* apresenta as opções *login*, *upload*, *upload as*, *download* e *browse*. A opção *login* faz a autenticação com o servidor e é exigida pelo menos uma vez durante a sessão, antes que se faça qualquer operação com o servidor. Quando selecionada, uma tela pede a URL do servidor em que se quer autenticar, o nome do usuário e a senha. As opções *upload* e *upload as* são utilizadas para carregar para o servidor as definições de processo que foram criadas ou editadas no Studio. A opção *download* permite buscar do servidor uma definição de processo (para ser editada pelo Studio e depois devolvida ao servidor). A opção *browse* mostra a lista de definições de processos que estão carregados no servidor e permite buscar ou apagar alguma dessas definições de processos.

As atividades definidas em um processo podem ser manuais, quando atribuídas a um usuário ou automáticas, quando são executadas pela própria máquina de *workflow*, que pode tomar uma ação. Segundo a documentação, o Reactor Studio permite programar ações com *script*s JavaScript, Tcl, Python e outras linguagens, mas não foi testado. É possível também enviar *e-mail*s, de acordo com os eventos que ocorrem com uma atividade, como por exemplo o início e o fim desta. O Studio permite a criação de formulários que aparecem no início (*start form*) e no fim (*completion form*) das atividades ou processos. Esses formulários recebem valores que são armazenados nos operandos do processo.

2.2.3 A interface do ReactorPortal

Essa é uma interface simples, consistindo apenas de três opções de menu: *login*, *Work List* e *Process Manager*. A opção de menu *login* é usada para a autenticação do usuário. A opção de menu *Process Manager* (fig. 2.17) mostra uma tabela que contém o nome e a descrição de todos os processos carregados no servidor e exibe *hyperlinks* relacionados a esses processos. O *hyperlink start* é usado para criar uma instância do processo, que é imediatamente colocada em execução. Os *hyperlinks Running* e *Completed* fazem o servidor listar as instâncias daquele processo que estão em execução ou completadas, respectivamente. Cada lista de instâncias, por suas vez, exibe um *hyperlink view*, que permite visualizar os detalhes daquela instância (fig. 2.18), como o estado das atividades e do processo, seus horários de início e término, o usuário que definiu o processo (*process engineer*), o usuário que criou a instância do processo (*process invoker*) e os valores dos operandos. As atividades de uma instância de processo são chamadas nessa interface de subprocessos.

Figura 2.17: Interface do Reactor Portal – opção de menu *Process Manager*

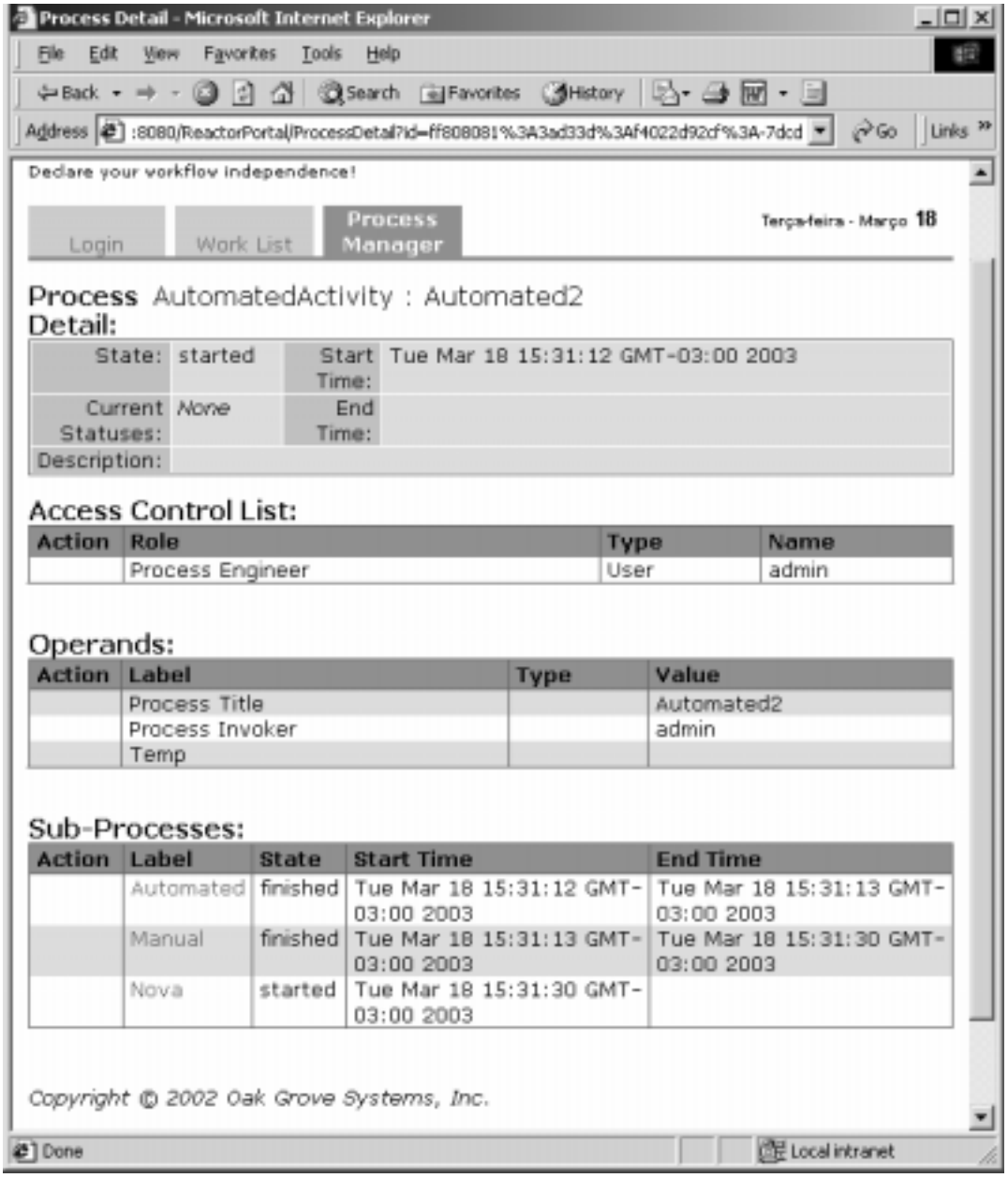


Figura 2.18: Detalhes da execução de uma instância de processo.

A opção de menu *Work List* (fig. 2.19) mostra, para aquele usuário que está autenticado, a lista de atividades pelas quais ele está responsável e estão disponíveis para a execução. Para cada atividade é listada a data em que a atividade foi ativada, o nome da atividade, o nome da instância do processo e o nome da definição do processo. É disponibilizado um *hyperlink* para iniciar a execução da atividade.

Figura 2.19: Opção de menu *Work List*.

2.2.4 Testes

Foram testados os exemplos que acompanham a instalação e os resultados são descritos a seguir:

O exemplo *Automated Activity* implementa uma atividade automática e uma manual. A atividade automática roda uma política que atribui ao status o valor *Done*. Dessa forma a atividade é completada automaticamente.

O exemplo *Make Request* permite escolher o usuário que irá executar uma instância quando da sua criação. O nome do usuário que irá executar o processo é informado no formulário de criação do processo e armazenado como operando no processo. Uma política do processo coloca-o nas ACL's das atividades.

O exemplo *Business Rule* implementa uma atividade automática que roda uma política de comparação, que compara o valor de um operando informado na criação da instância com o de uma constante (informada como parâmetro na chamada da política, na definição da atividade). De acordo com o resultado da operação, a política atribui um valor ao status (*High* ou *Low*), o que determina o roteamento da execução para uma ou outra atividade.

O exemplo *Meeting Reminder* simula o funcionamento de um despertador. Ele é composto de duas atividades. A primeira mostra ao usuário uma mensagem sobre um compromisso e permite a escolha de dois statuses: *Snooze*, para ser lembrado novamente dentro de algum tempo ou *Dismiss*, para encerrar o processo. Quando o usuário escolhe *Snooze*, o processo é roteado para uma outra atividade que executa uma política que espera por um tempo (informado como parâmetro na definição da atividade). Depois o processo retorna à primeira atividade, executando assim um ciclo com um tempo de espera, até que o usuário escolha a opção *Dismiss*. Os testes funcionaram durante algumas iterações, mas depois ocorreram erros. O mau funcionamento do Reactor Server impediu que todos os usuários acessassem a sua opção de menu *Worklist*. Não há recursos na interface do Reactor Portal para que o

administrador tome ações como a suspensão ou finalização de uma instância de processo ou de atividade, ou fazer um redirecionamento no fluxo. Também não existe a opção de apagar instâncias completadas. Mesmo quando se apaga a definição do processo, as instâncias continuam existindo e são novamente acessíveis quando a definição de processo é novamente carregada. O Reactor Server precisou ser reiniciado, o que, numa situação de produção, não seria uma boa alternativa. Ainda assim, a atividade de espera não terminou sua execução. Foi feita uma outra tentativa de resolver o problema, apagando a definição de processo do servidor para depois recarregá-la, através do Studio. Na tentativa de recarregar a definição de processo, ocorreu uma mensagem de erro dizendo que o processo já existia no servidor, apesar do fato de que usando a opção *browse*, ela não ser listada. Depois de algumas tentativas foi possível recarregar a definição do processo, mas as instâncias daquele processo continuaram com problema.

Quando há uma mudança na definição de um processo, as instâncias existentes continuam rodando de acordo com a versão antiga, enquanto as instâncias novas são criadas com a versão nova. Este tipo de implementação é denominado por (CASATI et al, 1996) como concomitância para a finalização.

2.2.5 Vantagens do Reactor

O Reactor Server e o Reactor Studio são implementados em Java, possibilitando uma independência de plataforma para esses componentes. As interfaces do Reactor Portal são disponibilizadas em HTML e XML, possibilitando também o acesso via *web*, a partir de qualquer plataforma.

Permite o acompanhamento da execução das instâncias, inclusive o histórico daquelas que foram completadas.

Permite a gravação/leitura de uma definição de processo para/de um arquivo XML, embora utilize um formato próprio.

Possui um editor de processos gráfico.

Pode enviar *e-mail*s, dependendo da ocorrência de eventos.

Possui políticas que podem manusear a ACL dos processos e atividades em tempo de execução.

2.2.6 Desvantagens do Reactor

Não é compatível com o padrão da Interface 1 – Linguagem de Definição de Processo (XPDL) estabelecido pela WfMC (WfMC: WORKFLOW PROCESS DEFINITION INTERFACE, 2002). Isso dificulta a interoperabilidade com outras ferramentas de definição de processo, como, por exemplo, a ferramenta gráfica desenvolvida por Telecken (TELECKEN et al., 2002).

Não cria papéis em um escopo externo ao processo e não atribui atividades a papéis. Atividades devem ser sempre atribuídas a um usuário, juntamente com o papel. Isso

implica que em qualquer adição de usuário na organização, é necessário editar as definições de processo para acrescentá-lo.

Não permite ao administrador, em caso de erros de execução, fazer alguma intervenção, como cancelar, ou terminar uma instância de atividade ou processo, ou redirecionar o fluxo de execução.

Não permite ao administrador apagar as instâncias já completadas.

Nos testes, o servidor não demonstrou ser robusto. Erros causados pela execução de uma instância de um usuário causaram danos ao servidor, que não foi mais capaz de atender a outro usuário que iria executar outro processo diferente. Para resolver o problema o servidor teve que ser reiniciado.

2.3 OFBIZ

OFBIZ (Open For Business) (OFBIZ - OPEN FOR BUSINESS SOFTWARE, 2002) é um projeto de software de licença aberta, que provê ferramentas e aplicações para negócios, entre elas uma aplicação de *workflow*. OFBIZ não possui uma ferramenta de edição de definição de processos. Ele é implementado em Java e JSP (JavaServer Pages) e utiliza, como servidor *web*, o Apache Tom Cat. O OFBIZ é multiplataforma e possui dois tipos de arquivos de lotes, para ambiente Windows (extensão .bat) e ambiente Linux/Unix (extensão .sh). Os testes foram feitos com a versão 2.0.0 em um sistema operacional Windows 2000 Advanced Server.

2.3.1 Instalação

A instalação é simples, bastando expandir o arquivo compactado para um diretório qualquer e fazer uma atribuição para uma variável de ambiente JAVA_HOME para o caminho do diretório do JDK. É conveniente que esse comando seja acrescentado no arquivo de lote `/catalina/bin/ofbiz.bat`.

2.3.2 Utilização

Para colocar o servidor no ar, deve-se executar o arquivo de lote `/catalina/bin/ofbiz.bat`, que inicia o serviço que disponibiliza um sítio *web* no servidor, na porta 8080. Esta porta não poderá estar ocupada por outro serviço.

A principal interface do OFBIZ é a tela *Web Tools*, acessada em http://<nome do servidor>:8080/webtools/control/main, onde <nome_do_servidor> é o nome da máquina onde OFBIZ foi instalado (fig. 2.20). Quando o usuário tenta acessar essa interface pela primeira vez é exibida primeiramente uma tela que demanda uma autenticação. A tela *Web Tools* provê duas opções de ferramentas para *workflow*. A opção *Read XPDL Files* possibilita a leitura de arquivos de definição de processo no formato padrão da Interface 1 da WfMC (XPDL) (WfMC: WORKFLOW PROCESS DEFINITION INTERFACE, 2002) e a opção *Workflow* Monitor permite o monitoramento da execução de uma instância de *workflow*.

Figura 2.20: Ferramentas do OFBIZ

Ao escolher a opção *Read XPDL Files é* aberta uma tela onde pode ser informado o nome do arquivo XPDL (com o seu caminho). Esse caminho pode estar acessível via um sistema de arquivos (em uma unidade de disco local ou de rede) ou via uma URL. Clicando no botão *View*, são mostradas as entidades identificadas no arquivo (fig. 2.21). Se a opção *Importe/Update to DB* estiver marcada, essas entidades são carregadas para o banco de dados.

A opção *Workflow* Monitor permite acompanhar a execução de uma instância de processo.

2.3.3 Testes

A importação de arquivos XPDL foi testada a partir de arquivos de exemplo disponibilizados pela WfMC em (WfMC: SAMPLE WORKFLOW, 2003). O resultado foi um erro em que o OFBIZ recusou o formato de data AM/PM utilizado no arquivo exemplo. A XML Process Definition Interface (WfMC: WORKFLOW PROCESS DEFINITION INTERFACE, 2002) não especifica o formato de data. Para prosseguir com os teste, os arquivos exemplos foram copiados para o computador local e seus formatos de data foram alterados para ficarem compatíveis com o do OFBIZ. Não foram encontrados outros erros e o OFBIZ identificou corretamente as entidades componentes da definição dos processos. No entanto, OFBIZ falhou ao tentar inseri-las no seu banco de dados.

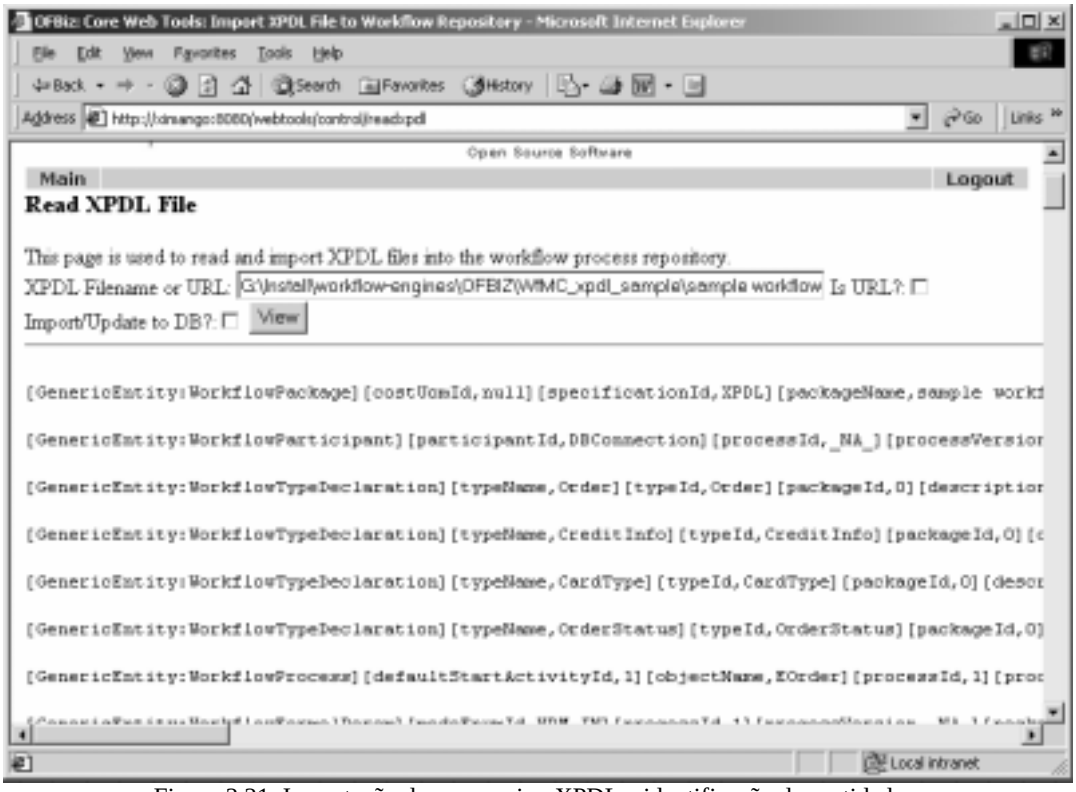


Figura 2.21: Importação de um arquivo XPDL e identificação das entidades.

A estrutura completa do banco de dados do OFBIZ pode ser vista em http://<nome do servidor>:8080/webtools/control/view/entityref (fig. 2.22). Navegando a partir desta tela podem ser vistos e editados as estruturas das tabelas, as relações de integridade referenciais e os dados. Foi feita a inserção manual de alguns registros nas tabelas que definem os pacotes, os processos e as atividades, criando-se assim uma definição de processo rudimentar, para prosseguir com os testes. Muitas falhas de visualização das páginas *web* ocorreram devido a problemas com o Javascript. Mas os registros foram inseridos.

Em nenhuma parte da interface do OFBIZ foi encontrada alguma opção para criar instâncias de uma definição de processo. Como o *Workflow* Monitor mostra apenas instâncias de processos, não foi possível trabalhar com o processo criado neste teste. A única opção foi visualizar uma instância de processo exemplo que vem com o OFBIZ (fig. 2.23).

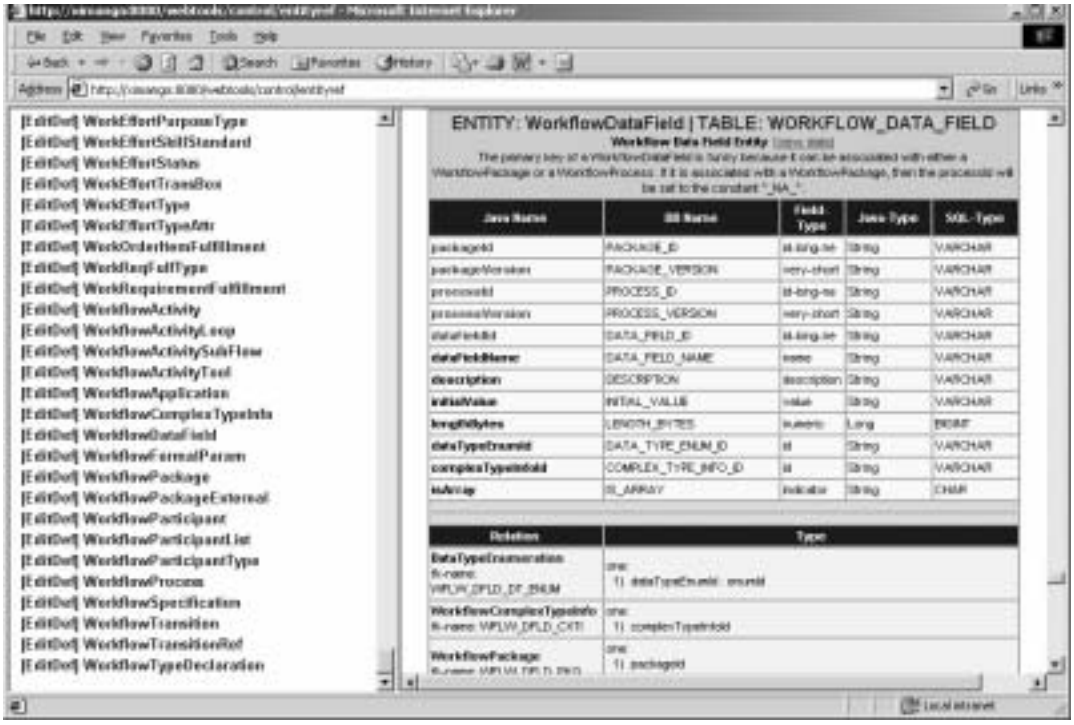


Figura 2.22: Estrutura do banco de dados do OFBIZ.

Figura 2.23: O Workflow Monitor com uma instância de exemplo.

2.3.4 Vantagens do OFBIZ

OFBIZ é implementado em Java e portanto é multiplataforma. As interfaces são disponibilizadas em HTML e Java Script, podendo ser acessadas via *web*, a partir de qualquer plataforma, com a ressalva de que o Java Script limita a portabilidade provida pela HTML.

Os testes demonstraram que OFBIZ pode interpretar as definições de processo feitas de acordo com o padrão da Interface 1 – Linguagem de Definição de Processo (XPDL) estabelecido pela WfMC (WfMC: WORKFLOW PROCESS DEFINITION INTERFACE, 2002), com uma pequena ressalva em relação ao formato de datas, o que possibilita o seu uso juntamente com ferramentas de definição de processo que utilizem esse padrão, em especial a ferramenta gráfica desenvolvida por Telecken (TELECKEN et al., 2002).

OFBIZ possui um banco de dados bem documentado e com toda a estrutura para armazenar definições de processo feitas em XPDL.

2.3.5 Desvantagens do OFBIZ

Os testes demonstraram que as ferramentas de *workflow* do OFBIZ ainda encontram-se incompletas e apresentam erros. De imediato constata-se a necessidade de corrigir o erro de interpretação do formato de data e os erros de inserção das entidades no banco de dados. Deve ainda ser implementada uma interface que gerencie a operação dos processos de *workflow*, permitindo, entre outras ações, a instanciação destes.

2.4 Domino Workflow

Domino Workflow é um produto da IBM/Lotus, de licença proprietária. Não foi obtida licença de avaliação e portanto o estudo desse software foi baseado apenas na documentação fornecida pelo fabricante em (NIELSEN et al., 2000).

2.4.1 Conceitos Básicos

Pasta de trabalho: São múltiplos documentos de trabalho que passam pelas pessoas. Uma pasta de trabalho é composta por uma capa, um documento principal e os outros documentos. Uma pasta de trabalho é uma estrutura lógica. Não há conexão física entre seus componentes.

Atividades e proprietários: Uma atividade é um grupo de tarefas relacionadas a uma pessoa, que é chamada proprietária da atividade. Essa pessoa recebe a pasta de trabalho e se torna a responsável única por ela (excluindo-se os casos de trabalho paralelo), até o fim da atividade. Existem também atividades automatizadas que não estão relacionadas a pessoas.

Proprietários potenciais de atividades X Proprietários de atividades: Normalmente não se define em um cenário de negócios quem exatamente irá executar cada atividade. Provavelmente será definido um grupo de pessoas para cada atividade, que serão proprietárias potenciais da atividade. Quando da execução do *workflow* uma pessoa irá pedir a atividade para si e tornar-se-á a proprietária desta.

Grupos e papéis: É bom evitar o uso específico de pessoas como proprietários potenciais de atividades. Criando grupos e papéis pode-se evitar muito trabalho, quando as coisas mudam na instituição.

Processos e suas instâncias: Após projetado, um processo pode ser ativado. Ativar um processo significa escrever sua definição em um banco de dados específico (Process Definition Database), permitindo que ele seja instanciado.

Proprietários de instâncias de processo: é uma pessoa ou grupo que tem total controle da execução de um processo. Nem sempre é uma pessoa que tem que decidir como as coisas devem ser feitas ou participar na definição do processo. Essa é a responsabilidade do proprietário do processo. Por outro lado, os proprietários de instâncias de processo têm que conhecer muito bem a estrutura do processo, uma vez que eles podem ser requisitados a intervirem em qualquer ponto do processo. Se qualquer erro ou comportamento imprevisível ocorre em tempo de execução, o proprietário do processo é notificado. Os proprietários de instância de processo são designados na definição do processo. O proprietário da instância de processo deve observar cada instância de processo e verificar as restrições de tempo e carregamento para cada participante do *workflow*. O grupo de proprietários de instâncias de processo é o mesmo para todos as instâncias de processo pertencentes a um processo (embora ele possa mudar sua composição com o tempo).

Equipe: é um grupo de pessoas trabalhando em uma mesma atividade. Cada equipe tem um líder que é responsável pela atividade e também seu proprietário. O proprietário de uma atividade é a única pessoa que pode decidir quando a atividade está completa.

Caminhos paralelos: Quando existem muitos caminhos vindo de uma atividade existem duas possibilidades. Uma é que apenas um caminho seja tomado para cada instância de processo e o outro permaneça inativo. Outra situação é quando os dois caminhos são tomados simultaneamente e, nesse caso, a pasta de trabalho é passada para as duas (ou mais) atividades. Domino Workflow permite que muitas pessoas trabalhem sobre um mesmo documento apenas num cenário de equipe, quando há um proprietário da atividade (o líder) que irá cuidar de possíveis conflitos de salvamento ou de replicação. Nos outros casos, são feitas múltiplas cópias da pasta de trabalho. O custo desta abordagem é a necessidade de juntar as cópias vindas de caminhos paralelos.

Roteamento: é a passagem dos documentos de uma atividade para a próxima e pode ocorrer de duas formas: roteamento no cliente e roteamento no servidor. A escolha é feita na definição do processo. O roteamento no cliente é geralmente preferido quando a transição tem que acontecer imediatamente. Nesse caso os agentes de *workflow* rodam no momento em que a atividade é completada. Entretanto, partes do *script* rodam na estação do usuário e ela tem que esperar até que ele complete. O roteamento no servidor ocorre algum tempo depois que a atividade finalizou. Os agentes rodam no servidor, de acordo com seus agendamentos. Quando ocorre um erro no roteamento, é emitida uma notificação para o proprietário da instância de processo.

2.4.2 Requisitos de Instalação:

- Domino Workflow 2.0
- Domino server R4.6.1 ou superior para rodar e testar a aplicação
- Uma estação Windows 95/98/NT/2000 com:
    - Domino Designer R5.0 e Domino Administrator R5.0
    - Notes Designer para Domino R4.6

2.4.3 Arquitetura Básica do Domino Workflow

A máquina de *workflow* do Domino Workflow é composta por um banco de dados de aplicação, um banco de dados de diretório de organização e um banco de dados de definição de processo. Ela pode ainda ser melhorada com a adição de um banco de

dados para auditoria e arquivo. O banco de dados de aplicação é onde o trabalho principal acontece. Nele são armazenadas as informações sobre as instâncias de *workflow*, checa-se por novos itens de trabalho e obtem-se informação sobre os itens existentes.

Architect: é uma ferramenta de modelagem gráfica para definição de processos e roda sobre a plataforma Windows. É também necessário ter o Notes Client. O Architect necessita do diretório Notes para trabalhar. O Architect roda no PC local e acessa a máquina de *workflow* no servidor. Para criar um processo podem-se usar objetos previamente projetados através da biblioteca de objetos de negócios (Business Object Library), reusar partes de algum outro processo que esteja armazenado no Design Repository, ou criar tudo. Depois deve-se conectar ao banco de dados de aplicação. Para atribuir pessoas e grupos às atividades é necessário conectar ao diretório de organização. No final, quando o processo está pronto, ele deve ser gravado no banco de dados de definição de processo.

Design Repository: é um banco de dados Notes que deve ser acessado unicamente via Architect. Ele é usado para armazenar informações de objetos relacionadas com definições de processos. Ele também trabalha com informações de processos que não foram ainda completados ou ativados.

Business Object Library (BOL): é a parte do Architec que permite utilizar todos os recursos disponíveis para a modelagem de processos. Ela mostra onde os recursos estão localizados e permite o seu uso, estejam eles no Design Repository, Organization Directory ou banco de dados de aplicação.

Domino Designer: é utilizado para a criação dos formulários utilizados na execução das atividades.

2.4.4 Definição dos processos

Para se definir um processo utiliza-se a ferramenta Architect. O primeiro passo é configurar os perfis de bancos de dados a serem utilizados. Depois deve-se iniciar um novo processo e devem-se definir, logo no início, as propriedades deste, que podem ser básicas ou avançadas. As propriedades básicas do processo são:

- Proprietário da instância de processo: são as pessoas responsáveis pela execução das instâncias de processo. É importante atribuir essa propriedade a mais de uma pessoa, uma vez que ela não poderá ser mudada após o início da execução da instância de processo.

- Iniciadores do processo: são as pessoas que podem iniciar uma instância de processo. Existem duas opções:

    - Todos: significa qualquer pessoa com acesso ao banco de dados de aplicação
    - Proprietários potenciais da primeira atividade: opção padrão.

- Formulários de processo: São especificados os formulários usados para o documento principal de uma pasta de trabalho. Também pode ser determinado se outros documentos na pasta de trabalho são permitidos e, nesse caso, quais os

formulários eles usarão. Essas configurações são consideradas padrões do processo, uma vez que essas podem ser definidas para cada atividade.

- *Process Time and Versions*: A opção de menu *Timing* é usada para especificar quanto tempo o processo deve levar. Pode-se também marcar se notificações de atraso de uma instância de processo serão enviadas ao seu proprietário e quando. A propriedade de versão é marcada e atualizada cada vez que o Architect salva o processo. Não é possível modificá-la diretamente.

- Descrição do processo: uma breve explanação sobre o processo.

Uma vez configuradas as propriedades do processo, inicia-se o desenho do diagrama do *workflow*, inserindo as atividades e as transições. Também as atividades possuem propriedades básicas e avançadas. As propriedades básicas são:

- Proprietário da atividade: é a pessoa que ficará responsável por esta.

- Tarefas da atividade: Cada atividade pode ser composta de várias tarefas, que devem ser feitas pela mesma pessoa. Durante a execução, a pessoa irá marcar cada tarefa completada e, somente após todas elas houverem sido finalizadas, a pasta de trabalho passará para a próxima atividade.

- Formulários da atividade: podem-se acrescentar formulários ao conjunto definido no processo.

- Timing e descrição: pode ser marcado quanto tempo a atividade deverá levar e se deverão ser enviadas notificações em caso de atraso.

- Reatribuição de atividade (propriedade avançada): Permite três opções: não permitido, permitido para outros proprietários potenciais e permitido para qualquer pessoa.

Atividades automatizadas são desenhadas usando-se um símbolo específico da paleta de símbolos. Elas permitem três tipos de ação mutuamente exclusivos: executar um agente Lotus Script, enviar um *e-mail* ou rodar um programa no servidor. Para enviar um *e-mail* é necessário definir uma fórmula para o campo “enviar para”, que pode ter um valor variável ou constante. As fórmulas podem acessar valores armazenados em campos de algum formulário da pasta de trabalho, ou outros atributos da instância de processo. Similarmente devem ser colocadas fórmulas para preencher o assunto e o corpo da mensagem.

As transições oriundas de um split apresentam as seguintes condições de roteamento:

- sempre (*default*)
- escolha exclusiva
- múltipla escolha
- condicional
- outra

Quando ocorrem caminhos paralelos, são feitas cópias da pasta de trabalho que seguirão por cada caminho. Essas cópias poderão ser modificadas pelas atividades de cada um dos caminhos e ao final deverá ser feita uma mescla dessas modificações na atividade que recebe essas cópias. Quando é utilizada a opção de escolha exclusiva pode-se desabilitar a opção de *join* na atividade que irá receber as transições oriundas de vários caminhos, uma vez que, seguramente, somente um caminho será executado. Isso é importante por questões de performance.

Quando um roteamento é configurado como condicional, cada transição pode ser ou não ativada de acordo com o valor de uma condição. É interessante criar uma transição do tipo *else* (ainda que seja para conduzir o *workflow* para uma situação de tratamento de erro), que será executada caso nenhuma outra condição seja executada.

Para permitir a uma pessoa escolher a transição a ser executada, deve ser criada uma tarefa do tipo decisão na atividade. São informados os valores possíveis a serem escolhidos, por exemplo “Aprova” ou “Desaprova”. As transições deverão ser configuradas como do tipo condicional e devem ser informados quais os valores esperados para que elas sejam executadas.

Uma vez completada a definição de um *workflow*, deve-se checar sua sintaxe através da opção *Check Syntax*, no menu. Quando não houver mais erros o processo pode ser salvo e ativado escolhendo-se a opção *Activate Process*, no menu. A ativação de um processo salva sua definição no banco de dados de definição de processos e também no banco de dados de repositório de projeto (*design repository database*). O processo pode também ser apenas salvado, sem ser ativado, e, nesse caso, será gravado apenas no banco de dados de repositório de projeto.

Mudanças em uma definição de processo: Uma vez que um processo tenha sido criado, existem duas maneiras de fazer alterações:

- Alterar diretamente a definição do processo. O Domino Workflow suporta versionamento de definições de processos. Quando uma nova versão de processo é ativada, a versão anterior é congelada, sendo usada pelas instâncias de processo já existentes. As instâncias de processo novas serão criadas segundo a última versão.

- Fazer as alterações em uma cópia da definição do processo. Isso implica em trocar o nome do processo, uma vez que os nomes de processos devem ser únicos em um banco de dados de definição de processos. Deve-se acessar o banco de dados de aplicação e marcar para que novas instâncias de processo não sejam iniciadas com a definição de processo antiga. Uma vez que todos as instâncias de processo rodando sobre a definição de processo antiga completarem, deve-se apagar essa definição, com todas as suas versões.

Recomenda-se alterar diretamente a definição do processo em casos de mudanças pequenas e incrementais. Fazer alterações em uma cópia é recomendado nos seguintes casos:

- Caso as mudanças sejam tão substanciais que o processo resultante possa ser considerado um novo *workflow*

- Estar sendo desenvolvido sobre um servidor caixa de areia, onde se quer testar, avaliar e comparar as diferentes mudanças antes de se decidir qual será a versão final.

2.4.5 Execução de uma instância de processo

Para executar uma instância de processo é necessário iniciá-la, o que significa criar uma nova pasta de trabalho que irá reunir toda a informação de um caso de negócio e ser passado pelas pessoas de acordo com as regras definidas. Uma instância de processo pode ser iniciada de várias maneiras: A mais básica é usar uma ação provida no banco de dados de aplicação que permite escolher um entre os processos ativos e criar uma instância de processo. Outra forma é usar o Domino Workflow Viewer . Pode-se ainda criar uma instância de processo via e-mail.

Durante a sua execução, uma atividade pode ser atribuída (ou oferecida) a um grupo de pessoas, mas alguém tem que pedir a responsabilidade para si. Isso pode ser feito pela visão *Inbox* ou por um documento pasta de trabalho. Após isso, a pessoa torna-se a proprietária e responsável pela atividade. Ela pode permitir que outras pessoas a ajudem designando-as como membros da equipe. É possível delegar a responsabilidade da atividade re-atribuindo-a a outra pessoa. Após completar todas as tarefas de uma atividade, o proprietário dessa deverá marcá-la como finalizada e então o Domino Workflow irá passar a pasta de trabalho para a próxima atividade.

Uma instância de *workflow* sempre é executada com a versão do *workflow* com que ela foi iniciada. Se houver necessidade de apagar uma versão de *workflow*, deve-se atentar para as instâncias que estiverem rodando com essa versão. Não há meios de fazer isso automaticamente, portanto isso deve ser feito manualmente.

Para visualizar instâncias de processo, é utilizada a ferramenta Viewer. Ela pode ser usada unicamente para instâncias de processo existentes e pode mostrar informações como a rota particular que uma instância de processo tomou.

2.4.6 Acessando Domino Workflow via clientes *web*

É possível fazer Domino Workflow acessível via *web* utilizando os recursos padrões do Domino para fazê-lo. Tudo o que se pode fazer no Domino, pode-se fazer no Domino Workflow. Adicionalmente, Domino Workflow oferece uma interface de usuário *web* "*out-of-the-box*".

Para permitir aos usuários unicamente iniciar novos processos, pode-se utilizar o recurso de iniciação baseada em *e-mail* ou baseada em formulário do Domino Workflow. Assim, pode-se criar um documento ou enviar um *e-mail* para o servidor de aplicação sempre que um usuário *web* preenche um formulário de ordem.

Para permitir toda a funcionalidade do *workflow* ser acessível via *web*, devem-se:

- Habilitar os formulários de aplicação usados no processo para o uso na *web*.
- Ativar os *Domino Workflow web Agents*.
- Ajustar as configurações de documento.
- Atualizar o *cache* de processo.

Vantagens de usar a interface de usuário *web* padrão:

- Pode ser feito com pequenos ajustes no banco de dados de aplicação, não requerendo mão de obra muito especializada.
- Os processos que já foram desenvolvidos com Architect não precisam ser modificados.
- Personalização não é difícil.
- Para quem está familiarizado com a interface Domino Workflow Notes, não é difícil se habituar com a interface *web* padrão.

Desvantagens da interface de usuário *web* padrão:

- Nem todas as funcionalidades do *workflow* podem ser acessadas por essa interface.
- Ela é muito básica e isso às vezes afeta sua amigabilidade.
- Ela é construída sobre o Domino R4.6 e não utiliza os recursos *web* do Domino R5.0

2.4.7 O Diretório de Organização

Nesse banco de dados são cadastradas as pessoas e suas relações dentro da organização. Por exemplo, podem ser criados departamentos, que são compostos de pessoas. Cada departamento contém um gerente, que não é automaticamente inserido como membro do departamento e deverá ser incluído explicitamente. Existem também os grupos de trabalho, que se diferem dos departamentos porque uma pessoa pode pertencer a mais de um grupo de trabalho, enquanto ela pertence a apenas um departamento e os grupos de trabalho admitem aninhamento, ou seja, um grupo de trabalho pode conter outros grupos de trabalho. As pessoas podem ter um ou mais papéis na organização ou no processo de *workflow*. Pode ser criado um calendário que define informação sobre os horários de trabalho na organização e os feriados. Isso é útil, por exemplo, quando uma atividade é definida para ser feita em um certo número de dias úteis. Nesse caso não são contados os feriados.

O diretório de organização é uma parte importante do ambiente de *workflow*. Domino Workflow tem seu próprio diretório de organização, onde podem ser definidos pessoas, departamentos, grupos de trabalho, papéis, delegação e perfis de ausência do escritório. Além disso, podem-se armazenar informações específicas do sistema, como relações, propriedades das instâncias de processo, bancos de dados de aplicações, programas dirigidos a clientes e a servidores e endereços de correio.

Nas organizações que utilizam o Domino, cada pessoa é registrada no Domino Directory como um usuário do servidor certificado. Adicionalmente podem ser definidos grupos. Domino Workflow Directory possui alguns recursos a mais, que podem ser implementados no Domino Directory através de uma personalização. No entanto, em muitas organizações, as personalizações no Domino Directory não são permitidas. As seguintes funcionalidades de Domino Workflow Directory não são oferecidas no Domino Directory:

- Hierarquia da organização: embora Domino Directory permita criar grupos de pessoas e uma estrutura hierárquica, essa hierarquia não é mostrada nas visões. Tampouco é possível definir o gerente de um grupo.
- Perfis de ausência do trabalho e de delegação: quando uma unidade da organização não está funcionando porque todos os seus membros estão fora, Domino Workflow é capaz de rotear as tarefas para os delegados.
- Papéis gerais da organização: podem ser criados papéis que sejam usados em múltiplas aplicações
- Local de armazenamento de recursos: podem ser armazenadas informações específicas de sistema, como a definição de relações, propriedades das instâncias de processo, endereços de *e-mail*, bancos de dados de aplicações e programas dirigidos a clientes e a servidores.

Domino Workflow provê funções para importar informações do Domino Directory para o Domino Workflow Directory. Em algumas organizações não é desejável ter o trabalho de manutenção de dois diretórios de organização. Para isso, Domino Workflow permite a integração com o Domino Directory e outros diretórios de organização baseados no Notes, como o LSA (Lotus Solution Architecture) Orga.

Relações são definidas no banco de dados organizacional e podem ser usadas em múltiplos processos. Uma relação retorna um valor e pode ser usada dentro da definição de um processo como uma fórmula.

Propriedades da instância de processo são fórmulas que representam informação gerada durante a execução da instância de processo e que portanto não são disponíveis durante o tempo de projeto. Existem propriedades de instâncias de processo pré-definidas, como, por exemplo, o proprietário da instância de processo, proprietário da atividade e proprietário da atividade anterior; mas outras propriedades também podem ser criadas. Um recurso pode ser:

- um endereço de mail de um banco de dados de aplicação. Quando se usa subprocessamento em outros bancos de dados, os diferentes bancos de dados devem ser registrados como recursos no banco de dados de organização.
- um endereço de mail de banco de dados ou de pessoas
- um programa dirigido ao cliente, que deve rodar na estação cliente. Esse programa pode estar armazenado no diretório de organização como arquivos instaláveis ou pode-se simplesmente definir um caminho e nome de arquivo no disco rígido da máquina cliente.
- um programa dirigido ao servidor, que deve rodar no servidor Domino. O caminho e nome do programa são armazenados no diretório de organização. Esses programas podem ser disparados unicamente por uma atividade automática.

Diretórios de organização alternativos: Para se usar outro diretório de organização é necessário usar o Domino Workflow Developer’s Toolkit 2.0. Ele vem com exemplos de como usar o Domino Directory e o LSA Organization Directory com o Domino Workflow. Outros diretórios de organização também podem ser usados.

2.4.8 Processamento Distribuído:

O processamento distribuído no ambiente Domino Workflow pode ser feito de duas formas:

- Múltiplos bancos de dados de aplicação Domino Workflow
- Múltiplas réplicas do banco de dados de aplicação Domino Workflow

Múltiplos bancos de dados: A solução Domino Workflow pode constituir de múltiplos bancos de dados de aplicação Domino Workflow. Eles podem compartilhar o mesmo servidor e banco de dados de definição de processos ou podem estar em servidores diferentes e ter bancos de dados de definição de processos distintos.

Subconjuntos de processos: um conjunto completo de processos pode ser dividido entre vários bancos de dados de aplicações, mas no mesmo servidor e compartilhando o mesmo banco de dados de definição de processos e o mesmo banco de dados de organização. As vantagens para essa abordagem são:

- usa um único banco de dados de definição de processo e de repositório de projetos;
- permite alguns processos rodarem em mais de um banco de dados.

A desvantagem é a manutenção de múltiplos bancos de dados de aplicações Domino Workflow.

Subprocessos: são usados quando os subprocessos são possuídos por diferentes partes na organização (ou fora dela) em relação ao processo principal. O banco de dados destino para o subprocesso é chamado via mail e pode estar em um servidor diferente. As vantagens são:

- Possibilidade de rodar processos interdepartamentais
- Possibilidade de usar servidores diferentes para o processo principal e o subprocesso.
- O subprocesso pode rodar numa aplicação que utilize um outro banco de dados de definição de processos.

A desvantagem é a manutenção de múltiplos bancos de dados de aplicações Domino Workflow.

Réplicas múltiplas: os bancos de dados do Domino Workflow podem ser replicados para outros servidores ou podem ser replicados localmente. Sugere-se que os agentes do Domino Workflow sejam agendados em um só servidor para diminuir a chance de conflitos de replicação. Se na aplicação existem múltiplos servidores, por causa dos agentes estarem agendados em apenas um servidor, os usuários de outros servidores poderão receber notificações referentes a instâncias de processo que não chegaram nos seus servidores. Não é possível evitar isso completamente, mas quanto maior a freqüência do agendamento dos agentes, menores serão as chances disso ocorrer.

Cluster: Os servidores dos bancos de dados do Domino Workflow podem fazer parte de um *cluster*. As vantagens são:

- Alta disponibilidade
- Um dos servidores pode ser usado como servidor de agentes.

A desvantagem é o alto custo de infraestrutura.

Réplicas locais – usuários móveis: O cliente Notes irá procurar pelos bancos de dados (definição de processos, organização e de aplicação) na mesma localização. Portanto, todos eles devem ser replicados. Trabalhando com uma réplica local aumenta a chance de conflitos de replicação, principalmente quando existem mais de um proprietário potencial para uma atividade. Para diminuir esse risco é recomendável que se restrinja o número de proprietários potenciais de atividades ao mínimo e que se faça o pedido de atividade na cópia que fica no servidor. Outra maneira de evitar esses conflitos de replicação é bloquear o documento em que se pretende trabalhar *off-line* usando a ação *Save & Write Protect*, antes de copiar a pasta de trabalho para o computador local. Ao final do trabalho, o bloqueio deve ser retirado com a ação *Release Protection*. As vantagens são:

- Melhor performance do que trabalhando sobre a rede.
- Habilidade de trabalhar *off-line*

As desvantagens são:

- Altas chances de conflito de replicação
- Dependente do agendamento da replicação do usuário.
- Dados não atualizados no servidor e no disco local.
- Grande uso de discos no lado cliente.

Usando um servidor de agentes: Se os processos contêm muitas atividades automatizadas, que gastam muito poder de processamento, pode ser interessante colocar um servidor extra para agendar todos os agentes do Domino Workflow. Ele pode fazer parte de um *cluster* ou ser conectado aos outros servidores via um agendamento de aplicação normal. Os usuários finais não deverão ter acesso a esse servidor. As vantagens são:

- Usuários finais não notam o carregamento dos agentes.
- Pode ser combinado com um servidor de cluster.

As desvantagens são:

- Notificações podem ser enviadas muito cedo.
- Alto custo da infraestrutura.

Se performance é uma questão importante, é necessário manter os bancos de dados tão pequenos quanto possível, por exemplo arquivando as instâncias de processos já finalizadas.

Réplicas localizadas: O tempo de resposta do Domino Workflow vai depender da velocidade da rede LAN (Local Area Network), ou WAN (Wide Area Network) quando for o caso. O cliente Notes é sensível à latência da rede porque ele tem várias pequenas

transações com o servidor para cada ação do usuário. Para melhorar esse problema pode-se re-projetar a aplicação, fazendo-a menos sensível a longas distâncias de rede, ou colocar outros servidores mais próximos aos clientes. A vantagem para essa segunda abordagem é uma melhor performance e as desvantagens são o problema da notificação poder chegar muito cedo e o alto custo da infraestrutura.

Replicação seletiva: é a cópia de apenas um subconjunto dos documentos. Pode ser feita entre servidores Domino ou entre um servidor Domino e um cliente Notes.

Resolução de conflitos de replicação: é importante que esses conflitos sejam reparados o mais rápido possível, uma vez que as pastas de trabalhos com conflitos irão falhar para rotear. Um agente pode ser agendado para enviar alertas de conflitos de replicação para os administradores.

3 A EDIÇÃO E O ARMAZENAMENTO DAS DEFINIÇÕES DOS PROCESSOS DE *WORKFLOW*

Uma máquina de workflow deve ler uma definição de processo, interpretá-la e executá-la. A máquina de workflow deve conhecer a linguagem na qual a definição de processo foi escrita pelo editor de workflow. Para facilitar essa comunicação a WfMC recomenda a adoção do padrão da Interface 1 – Process Definition Interface (WfMC: WORKFLOW PROCESS DEFINITION INTERFACE, 2002). Com a adoção desse padrão, a máquina de workflow pode receber definições de processos criadas por diferentes editores e, em especial, pelo editor gráfico AW (Amaya Workflow). Depois de interpretada, a definição de processo é armazenada em um banco de dados relacional.

3.1 O Amaya Workflow

O AW (Amaya Workflow) é um editor gráfico de *workflow* que está sendo desenvolvido no Instituto de Informática da UFRGS (Universidade Federal do Rio Grande do Sul) por Telecken (TELECKEN et al., 2002), atendendo a uma proposta do projeto CEMT. Este editor funciona integrado com o software Amaya (VATTON, 2003), que é o navegador/editor oficial da W3C (World Wide Web Consortium). O AW gera uma interface gráfica baseada no padrão SVG (Scalable Vector Graphics) e uma definição do processo de *workflow* baseada no padrão da XML (Extensible Markup Language) Process Definition Language, também conhecida como XPDL (WfMC: WORKFLOW PROCESS DEFINITION INTERFACE, 2002), que faz parte do padrão da Interface 1, definido pela WfMC. Nesse editor, o projetista de *workflow* pode desenhar um processo de *workflow* com o auxílio de uma paleta que contém os símbolos correspondentes a cada elemento da técnica de modelagem (figura 3.1). Posteriormente, o projetista pode editar os atributos dos elementos inseridos no processo de *workflow*. Ao final da edição o AW gera dois arquivos. Um deles contém as definições SVG, de modo que o desenho do processo de *workflow* possa ser visto em qualquer outra aplicação que interprete esse padrão. O outro arquivo é gerado no formato XPDL e contém as definições do processo do *workflow*, que poderá então ser executado em uma máquina de *workflow* que seja compatível com esse padrão.

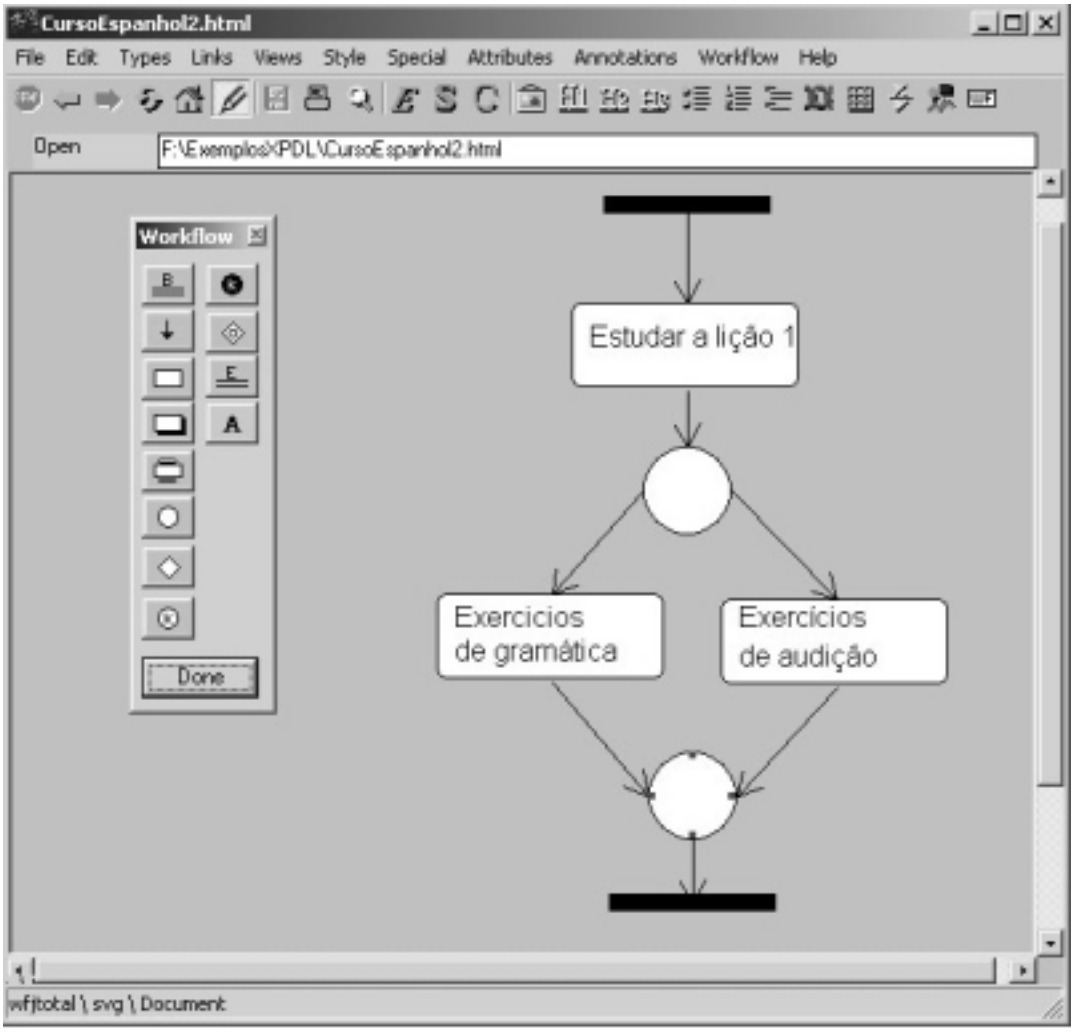


Figura 3.1: O AW - Amaya *Workflow*

3.2 Justificativa para o uso de um banco de dados relacional

A WfMC não padronizou qualquer formato de representação interna das definições de processo de workflow, deixando aos desenvolvedores a liberdade de escolherem aquele que melhor lhes convier. Três opções se mostraram possíveis para manusear as informações dos arquivos XPDL: manter o formato original dos arquivos e trabalhar com um banco de dados XML nativo; importar as informações para um banco de dados relacional ; ou importar as informações para um banco de dados relacional estendido a XML. A seguir serão discutidas as vantagens e desvantagens de cada abordagem e que fatores influem na decisão de usar uma ou outra.

Existe uma dificuldade em mapear a estrutura de XML para um RDBMS (Relational Database Management System) (CHAMPION, 2001). RDBMS são provavelmente melhores para manter integridade de dados e XML DBMS (XML Database Management System) são melhores para manter documentos XML. RDBMS provêem uma metodologia bem estabelecida para manter a consistência lógica dos dados. Mesmo num mundo em que muitas transações de clientes são feitas via XML, faz sentido manter dados de missão crítica em um RDBMS. Isso porque o modelo relacional foi

desenvolvido e refinado para manter consistência lógica sobre várias proposições da realidade de negócios (CHAMPION, 2001). Em (CHAMPION, 2001) sugere-se que em situações onde haja um grande intercâmbio de documentos XML contendo transações comerciais, estes sejam armazenados em sua forma original, inclusive preservando sua assinatura eletrônica, ao invés de serem armazenados em um RDBMS. Documentos, ao contrário de dados, podem ser produzidos por humanos e tendem a conter estruturas difíceis de serem normalizadas para RDBMS. Livros que tratam do assunto de mapear XML em RDBMS geralmente sugerem que os projetistas evitem os conteúdos do tipo *mixed* ou recursivos.

Documentos XML podem ser classificados em dois tipos: documentos XML orientados a documentos e XML orientados a dados, embora essa divisão não seja exata. Documentos XML orientados a dados são geralmente fáceis de mapear para um RDBMS e, nesse caso, RDBMS é um candidato óbvio para o armazenamento (DODDS, 2001).

"...apesar de ser possível usar um documento XML ou documentos como um banco de dados em ambientes com pequena quantidade de dados, poucos usuários e requisitos de performance modestos, ele irá falhar na maioria dos ambientes de produção, os quais têm muitos usuários, requisitos de integridade de dados e necessidade de boa performance." (BOURRET, 2003)

Documentos XML orientados a dados são projetados para serem consumidos por máquinas e usam o XML como meio de transporte. São caracterizados por uma estrutura bastante regular, dados finamente granulados e pouco ou nenhum conteúdo do tipo *mixed*. Exemplos desse tipo de documento são ordens de venda e agendamento de vôos (BOURRET, 2003) . Documentos XML orientados a documentos são geralmente projetados para consumo humano, como livros, *e-mail* e quase qualquer documento XHTML (Extensible Hypertext Markup Language) escrito à mão. São caracterizados por uma estrutura menos regular ou irregular, maior granulosidade dos dados e muito conteúdo do tipo *mixed*.

"Se existe um documento XML orientado a dados que seja facilmente normalizado para RDBMS, um RDBMS ou um RDBMS estendido para XML irão provavelmente trabalhar no mínimo tão bem como XML DBMS nativos." (Michael Champion) (BOURRET, 2003).

"Eu tenho visto DTD's com centenas de elementos e encontrar uma forma útil de mapeá-las para um banco de dados é distintamente não-trivial, especialmente quando uma porção de elementos são coberturas que não representam uma estrutura real num banco de dados." (Ronald Bourret) (DODDS, 2001).

Tratando-se de XML orientado a documento, um NXD (Native XML Database) pode ser a melhor opção. (DODDS, 2001). "Além disso, se existem documentos com *mixed content*, modelos de conteúdo recursivo e uma mistura complexa de elementos e atributos e quer-se pesquisar sobre a estrutura e conteúdo XML , um XML DBMS nativo será superior." (Michael Champion) (DODDS, 2001).

RDBMS são particularmente bons para armazenar informações altamente estruturadas e não são bons para trabalhar com dados semi-estruturados. XML orientados a documentos possuem uma estrutura variável (DODDS, 2001).

“Dados semi-estruturados são dados que possuem uma certa estrutura, mas que não é rígida. Um exemplo é um registro de saúde. Para um paciente pode haver uma lista de vacinas, para outro uma lista de cirurgias, etc. Outros exemplos são documentos legais e registros genealógicos...“ (Ronald Bourret) (DODDS, 2001).

“Dados semi-estruturados são difíceis de armazenar em RDBMS, porque haveria a necessidade de muitas tabelas diferentes, ou poucas tabelas mas com muitas colunas com valores nulos. Dados semi-estruturados são facilmente armazenados como XML e caem bem para um XML DBMS nativo.” (Ronald Bourret) (DODDS, 2001).

Se o XML que está sendo armazenado tem um esquema com uma estrutura fixa (com nenhum ou pouco uso de *mixed content* ou modelos de conteúdo recursivo) um RDBMS pode ser a melhor solução. (DODDS, 2001).

Alguns XML orientados a documentos possuem esquemas do tipo cabeçalho - corpo, onde os metadados são armazenados no cabeçalho e o conteúdo no corpo. Se a maioria das pesquisas é feita sobre os metadados, esses documentos podem ser facilmente armazenados em um RDBMS, colocando-se os metadados em tabelas e a parte menos estruturada em CLOBs (*character large object*). Mas se as pesquisas serão feitas freqüentemente na estrutura e no conteúdo dos documentos, um XML DBMS nativo é melhor (DODDS, 2001).

XML DBMS nativos são melhores para fazerem pesquisas de texto ou manusearem modelos de conteúdo recursivos.

SQL tem um conjunto de operadores de manipulação de dados muito mais rico do que Xpath e é baseada em uma teoria geral de dados. Se não existem XML orientados a documentos relativamente complexos, não há um ganho significativo em usar XML DBMS nativos em relação a um RDBMS. Mas se existem esquemas XML interessantes, isso é, com elementos recursivos, consultas Xpath podem resolver problemas que seriam bastante difíceis em SQL (DODDS, 2001).

“Em muitas situações pode-se trabalhar com um RDBMS se a maioria das aplicações projetadas para o banco de dados possam não necessitar ou manipular ou ver os dados como XML.“ (Tom Bradford). “Se os dados serão expostos para aplicações existentes de RDBMS e de XML, será provavelmente melhor deixá-los em um RDBMS. XML não tem uma boa noção de integridade de referência.” (Michael Champion) (DODDS, 2001).

Se a empresa já possui uma grande base em RDBMS, inclusive o treinamento de pessoal, não é conveniente mudar para um XML DBMS nativo. O esforço para desenvolver um mapeamento relacional para documentos com múltiplos esquemas ou sem esquema pode ser proibitivo, principalmente se esse esquema evolui continuamente. Nesse caso é melhor usar um XML DBMS nativo (DODDS, 2001).

A tabela 3.1 mostra um resumo das características que favorecem o uso de um banco de dados XML Nativo ou RDBMS e destaca cada aspecto em relação à máquina de *workflow*. A maior parte das questões levantadas pode ser resumida em uma só. É difícil ou fácil fazer o mapeamento do esquema XML para o relacional? Se a resposta for que é fácil, é recomendado trabalhar com o RDBMS. A definição do esquema XPDL em sua maior parte é bem estruturada e pode ser considerada como do tipo XML orientado a dados. Destina-se ao processamento por uma máquina e não para leitura de humanos. Isto justificou o uso de um banco de dados relacional.

Tabela 3.1: Pontos fortes de cada tecnologia e o que ocorre na aplicação da máquina de *workflow*.

Máquina de *Workflow*	RDBMS	XML DBMS Nativo
XML orientado a dados	XML orientado a dados	XML orientado a documento
Necessidade de dados consistentes.	Mantém consistência dos dados	
Fluxo de XML predominantemente de entrada. Pequeno volume (apenas definições de processos) em comparação com outras operações executadas na base de dados. Não está previsto o uso de assinaturas digitais.		Grande tráfego de documentos XML, inclusive com assinaturas digitais.
A grande maioria dos dados é bem estruturada. Ocorrem alguns poucos elementos do tipo *mixed content*.	Dados bem estruturados	Dados semi-estruturados, inclusive com a presença de elementos do tipo *mixed content* e elementos recursivos.
Integridade referencial é muito importante, facilitando bastante o desenvolvimento de aplicativos. Sua falta implica na necessidade de escrita de código adicional para verificar situações de inconsistência.	Provê boa integridade referencial.	Não provê boa integridade referencial.
A maior parte da definição do XPDL está descrita em um esquema. Existem poucas referências a esquemas externos.	Poucos esquemas para um determinado tipo de documento.	Para um determinado tipo de documento existem múltiplos esquemas e referência a esquemas externos.
Por enquanto, a necessidade de trabalhar com XML é devida apenas à Interface 1 da WfMC. A Interface 2 e 3 e a Interface 5 não utilizam XML. A interface 4 é bem mais simplificada e não seria grande problema.	Poucos tipos de documento.	Vários tipos de documento.
A aposta é de que a definição do XPDL não mude com freqüência e não mude muito.	Esquema estável, com poucas mudanças.	Esquema evolutivo, com muitas mudanças.
Na maior parte dos elementos o mapeamento foi fácil, embora o volume seja considerável.	Esquema fácil de mapear para o esquema relacional.	Esquema difícil de mapear para o esquema relacional.

3.3 A Estrutura do Banco de Dados

A partir do esquema do XPDL foi criada uma estrutura correspondente no banco de dados relacional. Um documento XPDL é composto de elementos e atributos. Os elementos formam uma estrutura aninhada, onde um elemento pai contém outros elementos filhos, que por suas vezes podem também conter outros elementos e assim sucessivamente até que se chegue a elementos que não contém filhos.

A princípio, todo elemento pai é mapeado para uma tabela no modelo relacional. Atributos e elementos que não contém filhos são mapeados para campos na tabela correspondente ao elemento pai. Se esses elementos ou atributos são opcionais, os campos serão do tipo que aceita o valor nulo. Quando um elemento filho é mapeado para uma tabela, esta terá um relacionamento com a tabela correspondente ao elemento pai, obedecendo à mesma cardinalidade definida no esquema XPDL. Se o elemento filho for obrigatório, o relacionamento também o será. Se a cardinalidade for 1:1, a tabela correspondente ao elemento filho (tabela filha) poderá ser embutida na tabela correspondente ao elemento pai (tabela mãe), desde que:

- O relacionamento seja obrigatório, ou
- Todos os campos da tabela filha sejam opcionais.

A tabela 3.2 mostra as ações a serem tomadas, de acordo com a situação do elemento encontrado.

Tabela 3.2: Mapeamento do esquema do XPDL para o esquema relacional.

<table>
<tr><th colspan="4">Situação</th><th>Ação</th></tr>
<tr><td rowspan="4">Elemento contém mais de uma informação (atributos ou outros elementos)</td><td colspan="3">Ocorre múltiplas vezes dentro de um outro elemento (elemento pai)</td><td>Cria uma tabela para esse elemento e um relacionamento N:1 com a tabela do elemento pai</td></tr>
<tr><td rowspan="3">Ocorre uma única vez dentro do elemento pai</td><td colspan="2">É obrigatório</td><td rowspan="2">A tabela pode ser embutida na tabela do elemento pai.</td></tr>
<tr><td rowspan="2">É opcional</td><td>Todos os elementos que contém são opcionais.</td></tr>
<tr><td>Contém algum elemento obrigatório</td><td>Cria uma tabela com relacionamento 1:1 com a tabela do elemento pai.</td></tr>
<tr><td colspan="4">Elemento contém só uma informação</td><td rowspan="2">É inserido como campo na tabela do elemento pai</td></tr>
<tr><td colspan="4">Atributo</td></tr>
</table>

Se uma tabela não possuir nenhum campo e tiver apenas uma tabela filha, ela pode ser eliminada e a tabela filha passa a se relacionar com a tabela mãe daquela que foi eliminada. Esse caso é comum no esquema do XPDL e ocorre com elementos que têm nomes no plural, como por exemplo os elementos `Activities` e `Transitions`.

Foi utilizado o banco de dados MySQL, versão 4.0.14-max, que permite a criação de tabelas do tipo INNODB que suporta restrição de integridade referencial.

3.3.1 Chaves Primárias

O atributo `Id` é usado como identificador em vários elementos e é um candidato natural a ser chave primária. O problema é que os arquivos XPDL podem ser gerados separadamente e ainda por pessoas diferentes e, assim, a chance de colisão desses identificadores é muito grande. Para minimizar o problema, todas as entidades (tabelas) foram definidas como entidades fracas em relação à entidade `Package`. Assim, a chave primária de cada tabela contém pelo menos dois campos, sendo um deles um campo que se refere ao campo `Id` da tabela `Package`. Dessa forma, uma vez evitada a duplicidade de chave na tabela `Package`, todas as outras tabelas estarão livres desse problema.

3.3.2 Tabelas de enumeração

Alguns campos seriam definidos, a princípio, como sendo do tipo enumeração. Quando se trabalha com esse tipo pode-se usar, no comando SQL, um dos valores de *string* válidos, ou um número inteiro correspondente a sua posição na lista de enumeração. Porém, no MySQL, quando se insere um valor inválido (*string* ou inteiro fora da faixa pré-determinada), este é armazenado como uma *string* vazia, com valor numérico igual a zero. Tal problema é reconhecido como uma falha ou deficiência do MySQL em seu manual de referência (MySql Reference Manual for Version 4.0.14, 2003). Para solucionar esse problema, foram criadas tabelas que são preenchidas com os valores válidos para os campos que seriam do tipo enumeração e esses campos passam a se referir a essas tabelas, através de uma chave estrangeira. Dessa forma garante-se a não inclusão de valores inválidos. Essas tabelas têm seus conteúdos constantes. São preenchidas apenas na criação do banco de dados e poderão posteriormente ser modificadas em uma situação de manutenção desse. Portanto seus acessos devem ser restritos ao DBA (Database Administrator). São as únicas entidades que não são entidades fracas em relação à entidade `Package`.

3.3.3 As tabelas Element e Attribute

O esquema XPDL prevê a grande maioria dos elementos que podem ocorrer em um documento XPDL. Esses elementos conhecidos são mapeados para tabelas no modelo relacional, que têm o mesmo nome do elemento. Por exemplo, o elemento `Activity` é mapeado para a tabela `Activity`. No entanto o esquema XPDL permite também a utilização de esquemas externos, que podem inclusive ser conhecidos quando do envio do documento XPDL para a máquina de *workflow*. Assim, podem ocorrer, em alguns pontos do documento XPDL (dentro de `ExtendedAttribute`, `SchemaType` e `Condition`), elementos com qualquer nome, o que torna impossível criar uma tabela com o nome do elemento na modelagem relacional. A solução encontrada foi criar duas tabelas genéricas, `Element` e `Attribute`, que podem armazenar qualquer informação proveniente de um documento XPDL. Na tabela `Element`, o nome do elemento é armazenado em um campo. São criados uma chave primária, utilizando-se um campo de inteiro, e um auto-relacionamento, de cardinalidade N:1, onde cada registro da tabela aponta para o registro correspondente ao elemento pai. Os atributos do elemento são armazenados na tabela `Attribute` que possui um relacionamento N:1 com a tabela `Element`. Essa implementação permite o armazenamento de qualquer

informação no banco de dados, mas a máquina de *workflow* deverá ter conhecimento suficiente para utilizá-la.

3.3.4 As Especializações

Foram criadas diversas especializações no modelo relacional. Por exemplo, o elemento `Activity` pode ser um `BlockActivity`, um `Route` ou um `Implementation`. Foram criadas as tabelas `BlockActivity` e `Implementation` como especializações da tabela `Activity`. Não foi criada uma tabela `Route`, tendo em vista que não há nenhuma informação adicional nesse elemento. Assim, um registro da tabela `Activity` que não contenha correspondente na tabela `BlockActivity` ou na tabela `Implementation` será um registro de uma atividade do tipo `Route`. Foram utilizados alguns artifícios para impedir inconsistências entre essas tabelas, que podem ser ilustrados pela figura 3.2.

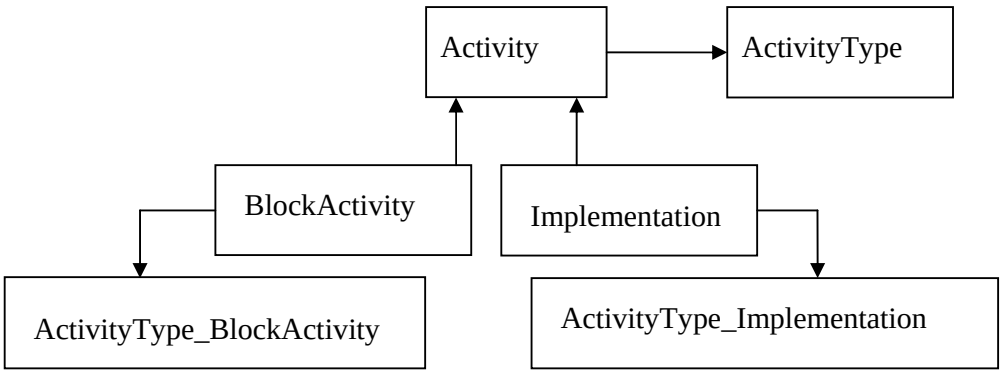


Figura 3.2: Fragmento do diagrama relacional do banco de dados de definição de processo.

Foi criada uma tabela com o nome `ActivityType`, onde são armazenados os tipos possíveis para uma atividade, que são `BlockActivity`, `Route` e `Implementation`. Essa inserção é feita apenas na criação do banco de dados e essa tabela não deverá sofrer alterações, a não ser em caso de manutenção do banco de dados. Suas permissões de acesso deverão ficar restritas ao DBA. Foi feito um relacionamento entre a tabela `Activity` e a tabela `ActivityType`, assim a tabela `Activity` deverá sempre ter um tipo válido. Duas tabelas semelhantes à `ActivityType` são criadas com os nomes de `ActivityType_Implementation` e `ActivityType_BlockActivity`. A diferença é que cada uma dessas tabelas armazena apenas um registro, contendo os tipos `Implementation` e `BlockActivity` respectivamente. Essas tabelas são referenciadas pelos campos `ActivityType` das tabelas `Implementation` e `BlockActivity`, respectivamente. Esses mesmos campos referem-se ao campo da tabela `Activity`. Assim, garante-se que um registro da tabela `Implementation` tenha tipo unicamente igual a `Implementation` e esse tipo é igual ao tipo do registro da tabela `Activity`, portanto torna-se impossível que na tabela `Activity` haja um registro de um tipo, relacionado com uma especialização de outro tipo.

3.4 O Compilador XPDL para SQL

Para inserir, na máquina de *workflow*, as definições de processos escritas em XPDL, foi criado um compilador que faz uma interpretação do arquivo em formato XPDL e o armazenamento de suas informações em um banco de dados. A forma mais simples encontrada para fazer isso foi a geração de comandos SQL, que são gravados em um arquivo de *script*, que posteriormente pode ser executado pelo RDBMS. O processo de compilação é composto de duas tarefas. A primeira delas é a tarefa de *parsing*, que consiste na verificação do arquivo de entrada, ou fonte, em relação ao padrão XPDL. A outra tarefa é a execução de ações, no caso a escrita de comandos SQL, correspondentes a cada elemento do XPDL, à medida que esses são detectados. Embora já existam softwares que façam o trabalho de *parsing* de arquivos XML, validando-os em relação a DTD's ou esquemas, as ações a serem tomadas são dependentes do esquema XPDL. Sendo assim, mesmo que esses *parser*s fossem utilizados, demandariam numerosas adaptações em seus códigos. Isso justificou o desenvolvimento de um compilador específico para o XPDL.

Para esse desenvolvimento foram utilizados o Flex e o Bison, que são ferramentas para construção de compiladores, que correspondem, para o sistema operacional Linux, às ferramentas Lex e Yacc (yet another compiler-compiler), bastante utilizadas no Unix. O Flex é um programa que lê um arquivo com definições de expressões regulares para o reconhecimento de *token*s e gera como resultado uma função, em linguagem C, que executa uma análise léxica (MASON; BROWN, 1990). Essa função lê caracteres do arquivo fonte e gera *token*s que alimentam o *parser*. Já o Bison lê um arquivo com uma gramática que descreve a estrutura que deve ter o arquivo fonte. Junto com essa gramática, são inseridas ações, que são executadas quando cada regra da gramática é reconhecida no arquivo fonte. O produto gerado pelo Bison é a função que faz o *parsing*, que compilada e lincada com a função gerada pelo Flex, resultam no compilador.

O trabalho consistiu das seguintes etapas: a construção de uma gramática livre de contexto para um *parser* do tipo LALR (*look ahead left-right*), que corresponda ao esquema do XPDL; a definição da estrutura do banco de dados; a inserção das ações a serem executadas quando do reconhecimento de cada regra da gramática; e uma verificação mais detalhada de inconsistências do fonte em XPDL, que embora possa estar de acordo com a sintaxe da gramática pode não gerar uma definição de processo de *workflow* válida.

3.4.1 A Construção da Gramática

Uma gramática livre de contexto é composta de um conjunto de símbolos terminais, chamados *token*s; um conjunto de símbolos não-terminais; um conjunto de regras de produção, compostas de um símbolo não-terminal, chamado de lado esquerdo e uma seqüência de símbolos terminais ou não-terminais, chamada de lado direito; e a designação de um não-terminal como símbolo inicial (AHO, 1988).

A construção da gramática foi feita de forma manual. Analisando-se o esquema do XPDL, criou-se um não-terminal para cada elemento `xsd:element` e `xsd:attribute` do esquema e foram criadas regras específicas, de acordo com o atributo `name` do elemento, que definiam esses não-terminais. Essas regras foram construídas, já se tendo em vista a inclusão das informações em tabelas também

específicas no banco de dados. Após a escrita inicial das regras, fizeram-se necessárias várias alterações para desfazer conflitos de redução/redução e redução/deslocamento, que não podem ocorrer neste tipo de *parser*, que tem um comportamento previsível.

O elemento `xsd:any` do esquema permite a ocorrência, no arquivo XPDL, de uma variedade de elementos, definidos em outros esquemas externos. O problema criado é que a gramática é estática, definida a priori, enquanto que esses esquemas externos podem até mesmo ser enviados juntos com o fonte. A solução encontrada foi combinar a flexibilidade com a rigidez. No caso do elemento `xsd:any` são usadas regras que apenas verificam se aquele trecho do arquivo está bem formado, permitindo depois que essas informações fossem inseridas no banco de dados em algumas tabelas `Element` e `Attribute`.

O esquema do XPDL especifica os atributos que os elementos devem conter, mas não impõe uma ordem específica. Isso gera um problema na construção da gramática, uma vez que as regras determinam uma ordem dos símbolos do lado direito. Foi necessário criar uma regra para cada permutação possível entre os atributos de um elemento, levando-se em conta também que alguns deles são opcionais. Essa abordagem foi adotada para elementos com poucos atributos, tipicamente dois ou três. O exemplo 1 mostra as regras definidas para a aceitação de dois atributos, `A` e `B,` em qualquer ordem.

```
Atributos : A B
          | B A
```

Exemplo 1

Para elementos com quatro atributos, seria gerado um número muito grande de regras e, portanto, foi adotada outra abordagem. Nesse caso foi feita uma regra onde qualquer dos atributos é aceito como um elemento não-terminal, mostrado no exemplo 2 com o nome de `Atributo`. A regra para reconhecer os quatro atributos em seqüência é escrita como uma seqüência do elemento não-terminal `Atributo,` que ocorre quatro vezes. O erro que pode acontecer é que um atributo pode ocorrer duas vezes e ser aceito como `Atributo` duas vezes e, ao final, a gramática pode aceitar uma expressão inválida. No entanto, esse erro é facilmente detectável escrevendo-se um pequeno código na ação correspondente à regra.

```
Atributos : Atributo Atributo Atributo Atributo
Atributo : A | B | C | D
```

Exemplo 2

É possível inserir regras na gramática que capturem os erros de sintaxe encontrados durante o processo de *parsing*. Essas regras usam um *token* especial chamado *error*. Com isso é possível emitir uma mensagem de erro personalizada e continuar o *parsing* para detectar todos os erros. É importante que o compilador gere mensagens de erros, o mais claramente e bem localizadas possível. Neste trabalho foram inseridas regras desse tipo apenas em alguns elementos principais.

3.4.2 As ações implementadas para as regras

As ações são escritas em um bloco de código, que pode ficar no final de uma regra, ou em qualquer ponto intermediário dessa. Assim que o *parser* casa a seqüência de *token*s ou elementos não-terminais à esquerda desse bloco, ele o executa. Esse bloco contém código em C e alguns comandos específicos do yacc, que permitem a recuperação de valores associados aos *token*s ou aos elementos não-terminais. Basicamente, esses valores são armazenados em variáveis, que, no momento oportuno, comporão uma expressão de um comando de inserção na linguagem SQL. Esse comando então é escrito para um arquivo de *script*. Também as mensagens de erro são produzidas por esses blocos de código, quando associados a regras de detecção de erro.

3.4.3 Verificações de Consistência

Apenas a sintaxe gramatical correta não é suficiente para garantir que um documento XPDL possa representar corretamente uma definição de processo de *workflow*. Por exemplo, podem ocorrer atributos `id` com valores repetidos para um mesmo elemento, o que acarreta um erro na execução do comando SQL. Outras inconsistências são a existência de atividades ou blocos de atividades que não possuam transições para uma atividade final do *workflow*, ou que não sejam atingíveis através de uma atividade inicial.

4 A API DA MÁQUINA DE *WORKFLOW*

A WfMC definiu a API que deve ser implementada pela máquina de *workflow* em (WfMC: WORKFLOW CLIENT APPLICATION, 1997). Nesse documento, são especificados as estruturas de dados e os cabeçalhos das funções, além de ser mostrada uma breve descrição de seus funcionamentos. A API foi escrita para uma implementação em C. Como este trabalho foi feito em PHP, algumas modificações precisaram ser feitas. A principal diferença é que no PHP não existe declaração de tipo para variável. As variáveis assumem um tipo quando são atribuídas a algum valor e podem inclusive mudar de tipo.

As funções possuem parâmetros de entrada (*in*) e de saída (*out*) e retornam um valor que significa uma mensagem de erro, ou de sucesso, que são também constantes já definidas na especificação (WfMC: WORKFLOW APPLICATION PROGRAMER'S INTERFACE NAMING CONVENTIONS, 1997). Para se modificar o valor de um parâmetro de saída, uma vez que o C e o PHP somente passam parâmetros por valor, faz-se necessário passar um ponteiro para a variável, ao invés de passar a variável propriamente dita.

Uma função do PHP muito útil em consulta de bancos de dados é aquela que retorna um objeto que corresponde a um registro da tabela pesquisada. Dessa forma, esse objeto pode ser retornado diretamente pela função, ou às vezes necessita apenas algumas modificações para compatibilizar alguns nomes de campos da tabela com nomes de campos nas *structs* .

A criação de uma API não é a única forma de implementar este tipo de software, usando a interface *web*. A primeira opção que pode ser pensada é escrever o código que recupera informações ou age sobre o banco de dados e que também apresenta uma interface para o usuário. Com a criação da API, as consultas e ações da máquina de *workflow* ficam separadas da apresentação, permitindo diferentes implementações de apresentação, inclusive desenvolvidas por terceiros, que usem a mesma API. Ao se desenvolver uma interface, utilizando PHP, deve-se colocar na página uma inclusão do arquivo de API e chamar suas funções, de acordo com a necessidade.

Na implementação de um sistema devem ser tomados cuidados para se evitar que ações indesejadas sejam executadas. Isso deve ser feito em vários níveis, uma vez que cada camada do sistema pode ser desenvolvida e acessada por diferentes pessoas. Assim, torna-se necessário agir:

- Na definição do esquema do banco de dados, evitando que a aplicação crie situações inconsistentes.

- Na segurança do acesso ao banco de dados, evitando que um usuário não autorizado possa alterar registros indevidamente.
- Na API, que deverá verificar se a ação requisitada pode ou não ser feita.
- Na interface, que deverá oferecer opções apenas para as ações válidas.

Um acesso ilícito pode ser feito usando outra interface ou usando comandos por fora da API. Por isso, as primeiras opções são as mais importantes.

4.1 Os estados das instâncias de processos

A WfMC definiu um conjunto de estados padrões para uma instância de processo. Os estados são definidos em níveis de granulosidade diferentes, permitindo uma implementação mais superficial ou mais detalhada. Além disso, a WfMC permite que alguns estados sejam omitidos e outros sejam adicionados. No nível mais alto, ou seja, menos detalhado, existem dois estados: `aberto` e `fechado`, com uma transição unidirecional permitida de `aberto` para `fechado`. O estado `fechado` pode ser dividido em três sub-estados: `completado`, `abortado` e `terminado`. Já o estado `aberto` pode ser dividido entre os estados *`rodando`* e `nãoRodando`. Este último pode ainda ser dividido entre nãoIniciado e `suspenso`. O estado nãoIniciado é o estado inicial da instância do processo. Os outros estados são atingidos pela invocação das funções WMStartProcess, WMTerminateProcess, WMAbortProcessInstance e WMChangeProcessInstanceState. Apenas a transição o para o estado `completado` não é executada por uma função da API, pois este estado é atingido pelo término de todas as atividades da instância do processo. A figura 4.1 mostra o quadro dos estados e as transições permitidas.

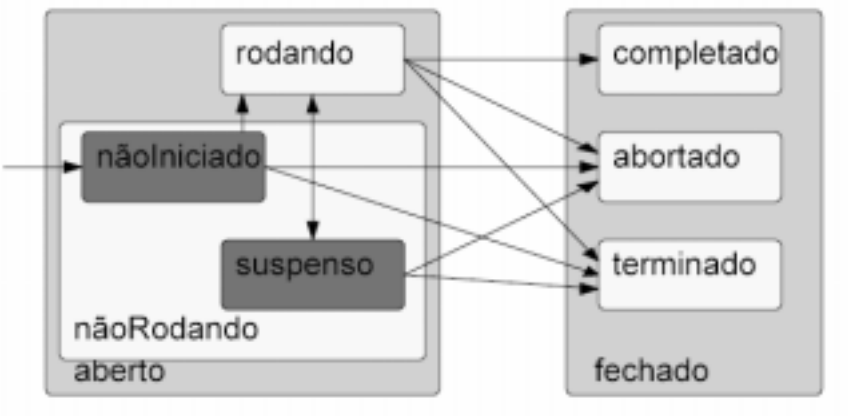


Figura 4.1: Diagrama de estados para as instâncias de processos - WfMC

O significado dos estados é descrito a seguir:

- `NãoIniciado`: É o estado inicial da instância de processo, assim que ela é criada.
- `Rodando`: Significa que a instância está sendo executada.
- `Suspenso`: Quando a execução é suspensa temporariamente, podendo voltar a ser executada.
- `Completado`: A instância terminou sua execução após a execução da última atividade prevista.

- `Abortado`: A execução da instância foi interrompida de forma definitiva, inclusive sendo interrompidas instâncias de atividades que porventura estivessem em execução
- `Terminado`: A execução da instância foi interrompida de forma definitiva, mas foi permitido que as instâncias de atividades que porventura estivessem em execução completassem a execução.

Foram implementados todos esses estados, na sua granulosidade mais fina, sem alterações.

4.2 Os estados das instâncias de atividades

Os estados das instâncias de atividades propostos pela WfMC são bastante semelhantes àqueles das instâncias de processos e são mostrados na figura 4.2. As diferenças são: o estado inicial que se chama `nãoRodando`; a presença de apenas dois níveis de granulosidade no estado `aberto` que passa a ter como sub-estados: `nãoRodando`, `suspenso` e `rodando`; e a total bidirecionalidade nas transições entre os sub-estados do estado `aberto`. As diferenças entre os estados `suspenso` e `nãoRodando` não são claras, uma vez que ambos possuem transições de e para os mesmos estados.

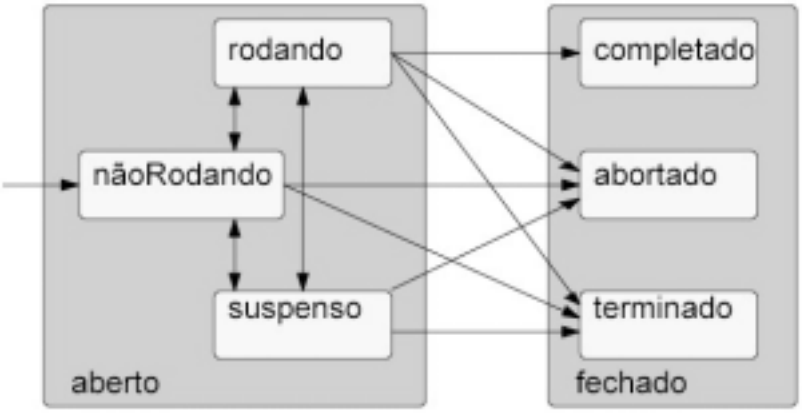


Figura 4.2: Diagrama de estados para as instâncias de atividades proposto pela WfMC.

Propõe-se outro modelo de estados e transições, que pode ser visto na figura 4.3, inspirado no modelo de estados para os processos.

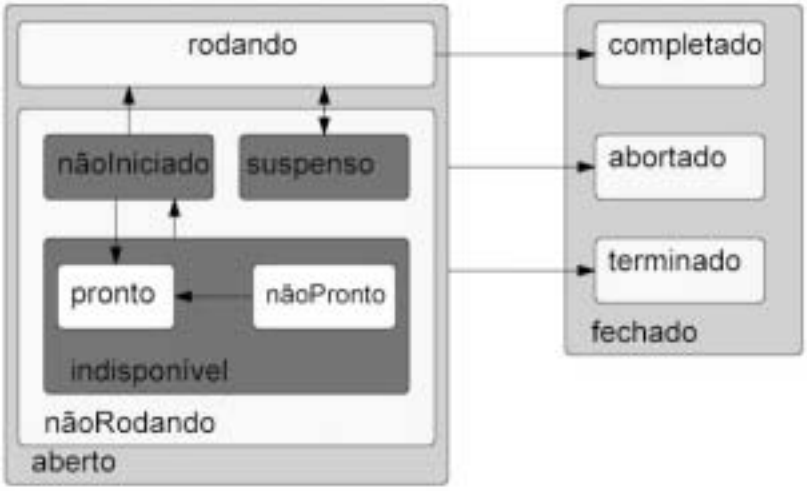


Figura 4.3: Diagrama de estados para as instâncias de atividades.

Este modelo adiciona o estado `indisponível` que pode ser detalhado em vários sub-estados, de acordo com o cumprimento ou não das seguintes condições para que a atividade seja iniciada:

- `Pronto`: A atividade seja a atividade inicial do processo de *workflow*; ou uma transição para essa atividade seja executada e sua condição avaliada como verdadeira, a menos que a atividade seja um AND-JOIN, caso em que todas as transições que chegam nesta atividade deverão ser executadas e ter suas condições avaliadas como verdadeiras.
- `NoHorário`: Uma restrição temporal, se houver, relativa ao início da atividade seja cumprida. Essa restrição pode ser única (por exemplo, a partir de 23 de outubro de 2003) ou periódica (por exemplo, do dia 15 ao dia 18, todos os meses).
- `Desbloqueado` : A atividade deve estar desbloqueda. O bloqueio ou desbloqueio de uma atividade são efetuados pelo administrador do processo de *workflow*. Além disso, a definição do processo pode determinar a situação inicial da atividade. O valor padrão é desbloqueado.

Em oposição a estas condições, existem, respectivamente, as condições `nãoPronto`, `foraDoHorário` e `bloqueado`. Uma vez que a atividade adquira a condição `Pronto`, ela não poderá mais voltar à condição `nãoPronto`. As condições `foraDoHorário` e `noHorário` podem se alternar indefinidamente, quando uma atividade é definida para estar disponível a intervalos periódicos. O mesmo pode acontecer em relação às condições `bloqueado` e `desbloqueado`, uma vez que o administrador pode bloquear ou desbloquear a atividade a qualquer instante. A restrição temporal e o bloqueio da atividade podem ser implementados na definição de processos como *ExtendedAttributes*. Sendo três condições, cada uma com dois valores possíveis, é possível uma combinação de 2^3, ou seja, 8 estados. Dentre esses estados, apenas um cumpre com todas as condições para que tenha uma transição para o estado `rodando`. Este estado é chamado de `nãoIniciado`. Os outros estados são agrupados no estado `indisponível` que pode ser visto na figura 4.4.

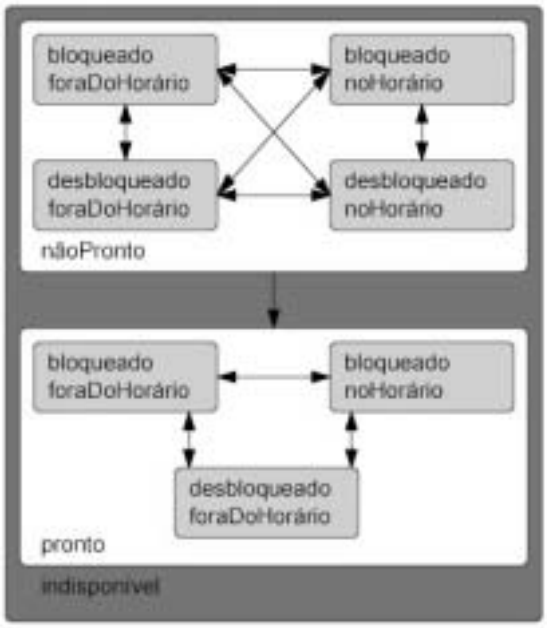


Figura 4.4: Detalhamento do estado `indisponível`, proposto para o diagrama de estados e transições para instâncias de atividades.

4.3 Atividades em Grupo

A atividade em grupo é definida na CEMT Workflow como sendo uma atividade a ser executada por todos os participantes do grupo. Isso é diferente de uma atividade individual que fica disponível em um *pool*, mas que após ser requisitada fica alocada a apenas um participante. Uma instância de processo de *workflow* consiste em uma seqüência de instâncias de atividades. Cada atividade tem uma ou mais atividades precedentes, com exceção da atividade inicial, e uma ou mais atividades sucessoras, com exceção da atividade final. A concepção de uma atividade em grupo pode ser imaginada como um ponto comum entre várias instâncias de *workflow*. Esta teria como antecessoras e sucessoras atividades individuais alocadas ao mesmo usuário do grupo. Em uma aplicação que controle a execução de um curso, a sua funcionalidade poderia ser vista, embora não implementada, como na figura 4.5. Isso porque a seqüência das atividades é definida na definição de processo, que é estática. Uma definição de processo que atendesse à necessidade da figura 4.5 teria alguns inconvenientes. Em primeiro lugar, é possível que um processo tenha várias atividades em grupo, com grupos de tamanhos diferentes, o que dificultaria bastante o projeto da definição do processo. Mas ainda que todas as atividades fossem feitas por grupos do mesmo tamanho, essa abordagem implicaria na necessidade de manter o mesmo grupo do início ao fim da execução do processo. Essa imposição pode ser bastante problemática, por exemplo em uma aplicação de controle de execução de um curso, pois em casos de desistências ou atrasos de alunos, todo o grupo ficaria prejudicado. Portanto é mais viável uma abordagem que permita aos usuários formarem os grupos dinamicamente, podendo escolher outros usuários que estejam prontos para executarem as atividades em grupo.

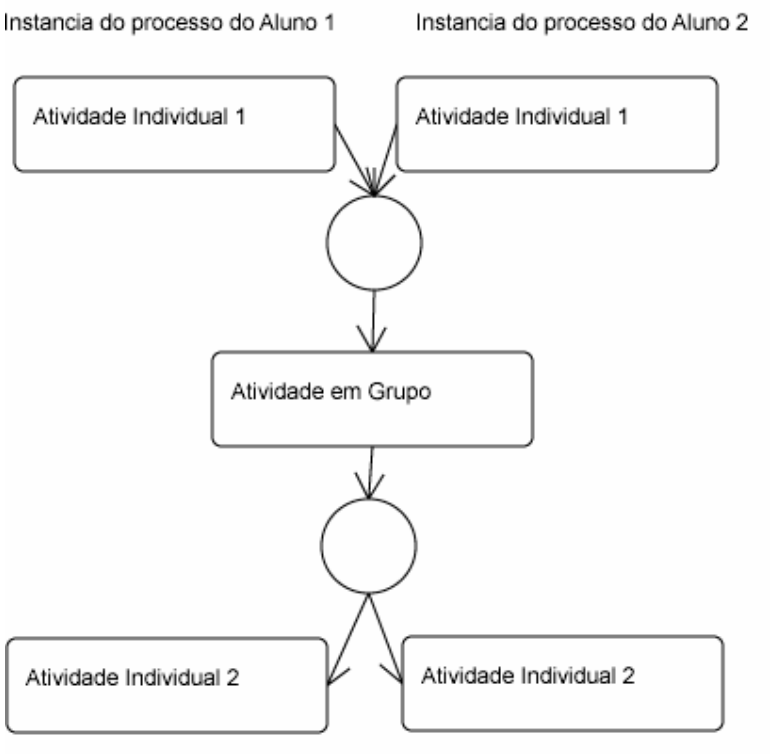


Figura 4.5: Atividade em grupo

Cada instância de atividade é representada no banco de dados como um registro na tabela `ActivityInstance`. Para uma atividade em grupo, é criado um registro para cada participante da atividade e esses registros têm em comum o valor do campo `Group.` Este conjunto de registros é chamado de grupo. Um grupo completo ou fechado é aquele cujo número de registros é igual ao número de participantes definido para a atividade. Um grupo incompleto ou aberto é aquele que ainda não atingiu esse número. Um tempo máximo de espera pode ser estabelecido para a formação de um grupo, de tal forma que, expirado este tempo, o grupo seja fechado com um número de participantes menor do que o previsto. O diagrama de estados das instâncias de atividades em grupo foi alterado, sendo incluídos dois estados entre os estados `indisponível` e `nãoIniciado`, como pode ser visto na figura 4.6.

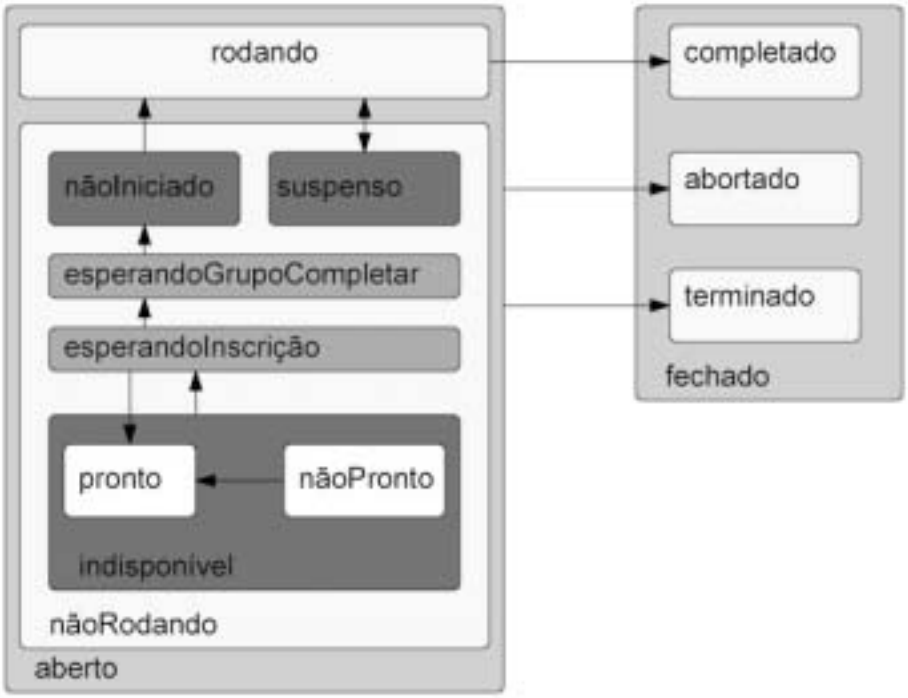


Figura 4.6: Diagrama de estados das instâncias de atividades em grupo.

O primeiro deles é o estado `esperandoInscrição` que é atingido assim que a atividade sai do estado `indisponível`. Para atividades nesse estado, a página da *worklist* disponibiliza uma ação para que o participante se inscreva em um grupo. Quando isso é feito, a máquina de *workflow* procura por um grupo aberto e inclui o participante neste grupo, atribuindo o número deste grupo ao campo `Group` do registro correspondente àquela atividade e participante. Caso não seja encontrado nenhum grupo aberto, é criado um novo grupo, atribuindo um número novo ao campo `Group` do registro. A atividade passa, então, para o estado `esperandoGrupoCompletar`. Quando o grupo se completa a atividade é mudada para o estado `nãoIniciado`. A partir daí, a execução da atividade segue como no caso das atividades individuais, com a página da *worklist* disponibilizando uma ação para que cada participante inicie a execução da atividade. A aplicação chamada por uma atividade em grupo poderá acessar o valor do campo `Group` e saber quais são os outros participantes. Aplicações típicas podem ser um *chat* ou um jogo.

Da maneira como foi exposto, essa alocação do usuário a um grupo é feita de maneira automática. Nesse caso, haverá, para cada definição de atividade, no máximo um grupo aberto. Outra opção é permitir ao usuário escolher em que grupo deseja entrar. Isso pode ser particularmente útil quando o usuário prefere trabalhar com outros que falem o mesmo idioma, por exemplo. A máquina de *workflow* poderá mostrar ao usuário uma lista dos grupos abertos. Essa informação poderá ser detalhada com os nomes dos componentes, os idiomas destes, localização geográfica, etc. O usuário poderá entrar em um desses grupos, ou ainda criar outro.

4.4 As funções implementadas

A WfMC definiu 63 funções em sua API em (WfMC: WORKFLOW CLIENT APPLICATION, 1997). Dessas, as seguintes foram selecionadas para serem implementadas:

- WMConnect: faz a conexão com a máquina de *workflow* e retorna um ponteiro para uma variável que é usada nas outras funções.
- WMOpenProcessDefinitionList : Faz uma consulta na máquina de *workflow* que recupera um conjunto de definições de processos existentes. Algumas condições podem ser especificadas para filtrar a consulta. Essa função retorna um ponteiro que é depois usado pela função WMFetchProcessDefinition.
- WMFetchProcessDefinition: Retorna a próxima definição de processo do conjunto de definições retornado pela função WMOpenProcessDefinitionList.
- WMChangeProcessDefinitionState: Troca o estado de uma definição de processo. Foram definidos dois estados possíveis para as definições de processo: ativo e inativo. Somente definições de processo no estado ativo podem gerar instâncias de processo de *workflow*.
- WMCreateProcessInstance: Cria uma instância de processo a partir de uma definição de processo.
- WMStartProcess: Inicia uma instância de processo criada. Quando isso é feito, o estado da instância do processo é mudado de `nãoRodando` para `rodando` e a primeira atividade do processo é criada e iniciada.
- WMTerminateProcess: Termina uma instância de processo e o coloca no estado `terminado` . Essa função, quando chamada pelo usuário, termina a execução da instância de processo de maneira graciosa, permitindo que as instâncias de atividades dessa instância de processo que estejam rodando completem suas execuções.
- WMAbortProcessInstance: Aborta a instância de processo, tentando interromper também todas as instâncias de atividades que estejam sendo executadas.
- WMChangeProcessInstanceState: Muda o estado de uma instância de processo. Algumas mudanças de estado possuem funções específicas, que são WMStartProcess, WMTerminateProcess e WMAbortProcessInstance. Portanto o uso dessa função deve ficar restrito, atualmente, às mudanças entre os estados `rodando` e `suspenso`. Essa função chama uma função interna que verifica se a transição requisitada é permitida, de acordo com o estabelecido na figura 4.1.
- WMOpenProcessInstanceList: Faz uma consulta que recupera um conjunto de instâncias de processos. Algumas condições podem ser especificadas para filtrar a consulta. Essa função retorna um ponteiro que é depois usado pela função WMFetchProcessInstance.
- WMFetchProcessInstance: Retorna a próxima instância de processo do conjunto de instâncias retornado pela função WMOpenProcessInstanceList.
- WMOpenActivityInstanceList: Faz uma consulta que recupera um conjunto de instâncias de atividades. Algumas condições podem ser especificadas para filtrar a consulta. Essa função retorna um ponteiro que é depois usado pela função WMFetchActiviyInstance.
- WMFetchActivityInstance: Retorna a próxima instância de atividade do conjunto de instâncias retornado pela função WMOpenActivityInstanceList.

5 A ARQUITETURA E O FUNCIONAMENTO DA MÁQUINA DE WORKFLOW

A máquina é implementada com a utilização de um banco de dados MySQL e um servidor *web* Apache, junto com o PHP, rodando em um sistema operacional Linux, seguindo a filosofia do projeto CEMT de trabalhar com software livre. A figura 5.1 mostra a arquitetura da máquina de *workflow*.

O compilador lê as definições de processo geradas pelo editor de *workflow*, no formato XPDL, e gera um *script* de comandos SQL. Esse script é lido por uma página PHP e seus comandos são executados, inserindo assim as informações referentes à definição do processo, no banco de dados.

As páginas implementadas em PHP recebem requisições dos usuários que normalmente são atendidas através de consultas ou atualizações no banco de dados. Por exemplo, um usuário pode demandar a sua lista de tarefas. Uma página em PHP faz uma consulta para saber os nomes dos processos de *workflow* e atividades ativas para aquele nome de usuário e mostra o resultado no formato HTML. Outras vezes o usuário pode informar à máquina, através de uma interface HTML, que terminou a execução de uma atividade e a máquina deverá atualizar o banco de dados com essa informação, além de executar as mudanças que sejam conseqüências disso.

Além disso, o programa engine, desenvolvido em linguagem C, roda na condição de serviço do sistema operacional e executa um laço infinito, onde, a todo tempo, verifica o banco de dados e confere se algumas atividades atingiram um tempo limite para seu término ou se chegou a hora de ativar algumas atividades programadas para um determinado horário.

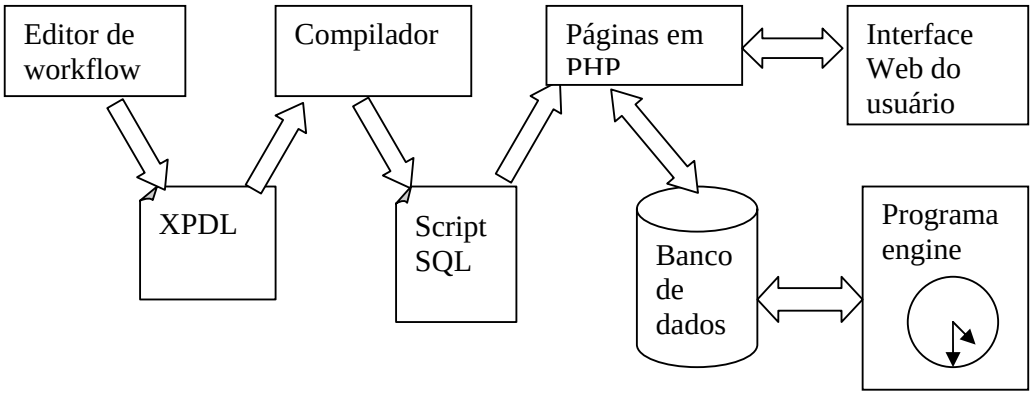


Figura 5.1: Arquitetura da máquina de *workflow*.

5.1 Extensões ao modelo de definição de processos

A WfMC definiu, na Interface 1 (WfMC: WORKFLOW PROCESS DEFINITION INTERFACE, 2002), um conjunto de funcionalidades mínimo que deve ser atendido por todas as máquinas de *workflow*. No entanto, ela permite também uma extensão dessas funcionalidades para atender a objetivos de uma máquina de *workflow* específica e isso pode ser feito através da utilização de atributos estendidos (elemento `ExtendedAttributes`), que estão presentes em quase todos os elementos do esquema XPDL. O elemento `ExtendedAttribute` possui dois atributos: `Name` e `Value`. No caso do elemento `ExtendededAttribute` pertencente ao elemento `Activity`, o atributo `Name` poderá receber os seguintes valores: `Pool`, `Group`, `StartDate`, `Bloqueado` e `Notifications`. Isto servirá para implementar as funcionalidades descritas a seguir.

Pode ser definido que uma atividade não possa ser iniciada antes de uma data e horário estabelecidos. Para isto, um elemento `ExtendedAttribute` é colocado na definição da atividade e a expressão `StartDate` é atrbuída ao atributo `Name`, e ao atributo `Value` é atribuída a data e o horário. Uma instância de atividade ainda não iniciada, cujo valor de `Value` seja maior do que o valor da data atual, estará na condição `foraDoHorário`. Caso o valor deste elemento seja menor do que a data atual ou este elemento `ExtendedAttribute` não esteja presente, a instância estará na condição `noHorário`. Esta definição de data inicial poderá evoluir para intervalos de data ou disponibilidade períodica. Por exemplo, uma atividade pode estar na condição `noHorário` toda sexta-feira de 08:00 às 11:00 hs.

Quando o valor do atributo `Name` do elemento `ExtendedAttribute` de uma atividade for igual à expressão `Bloqueado`, isso faz com que todas as instâncias dessa atividade fiquem inicialmente bloqueadas, ou seja, estarão na condição `bloqueado`. A ausência deste elemento faz com que as instâncias dessa atividade fiquem inicialmente desbloqueadas.

As atividades para as quais são definidas datas e horários de início ou prazo máximo para conclusão podem ter também definidas datas e horários de notificação. Neste caso haverá um elemento `ExtendedAttribute` dentro de uma definição de atividade e o valor do atributo `Name` será igual a `Notifications`. Dentro deste elemento haverá outros elementos do tipo `Notification` que terá atributos com informações sobre a data e horário da notificação e o tipo de notificação. A data e horário da notificação podem ter um valor absoluto ou relativo à data de início ou prazo da atividade. Podem ser definidas várias notificações, como por exemplo, uma semana antes, um dia antes e uma hora antes.

Atividades em grupo podem ser definidas acrescentando-se ao elemento `Participant` um elemento `ExtendedAttribute`. O atributo `Name` será igual à expressão `Group`. Este elemento deverá conter dois outros elementos. O elemento `Number` define o número de participantes do grupo. O elemento `Timeout` estipula um tempo máximo para espera da formação do grupo e possui o atributo `Unit` que especifica a que unidade de tempo o elemento `Timeout` se refere.

O elemento `Participant` pode conter também um elemento `ExtendedAttribute` com o valor `Pool` para o atributo `Name`. Isto significa que a instância de atividade será alocada a um *pool*, acessível a um grupo de pessoas, a exemplo do que faz o Domino Workflow (NIELSEN et al., 2000). A atividade é individual e quando uma pessoa a inicia, ela desaparece do *pool*, não ficando mais disponível aos outros integrantes do grupo.

Os exemplos a seguir demonstram o uso dos elementos `ExtendedAttribute` propostos. O exemplo 1 mostra a definição de um participante que corresponde a um grupo de 5 pessoas. Após 2 dias que a primeira pessoa se registrar na atividade, esta inicia-se com qualquer número de participantes.

```
<Participant id="Alunos" Name="ProcessCreator">
  <ParticipantType Type="ROLE" />
  <Description>Alunos criadores das instancias dos processos</Description>
  <ExtendedAttributes>
        <ExtendedAttribute Name=”Group”>
              <Number value=”5”>
              <Timeout unit=”days”>2</Timeout>
        </ExtendedAttribute>
   </ExtendedAttributes>
</Participant>
```

Exemplo 1 – Definição de participante como um grupo.

O exemplo 2 mostra a definição de um pool. A atividade será alocada a esse *pool* e aparecerá na *worklist* de todos os usuários do papel `Professores`. Mas quando um usuário iniciar a atividade, ela é retirada do *pool* e da *worklist* dos outros usuários.

```
<Participant id="Professores" Name="Professores">
  <ParticipantType Type="ROLE" />
  <Description>Professores</Description>
  <ExtendedAttributes>
        <ExtendedAttribute Name=”Pool”/>
   </ExtendedAttributes>
</Participant>
```

Exemplo 2 – Definição de participante como um *pool*.

No exemplo 3, a instância de atividade, quando criada, estará bloqueada. Além de necessitar de um desbloqueio, ela só poderá ser iniciada a partir do dia 18/01/2004. Uma definição de notificação com um valor absoluto determina que no dia 15/01/2004 será emitida uma notificação sobre o início da atividade. Outras notificações com valores relativos serão enviadas dois dias antes, um dia antes e uma hora antes do início da atividade, também notificando sobre o início da atividade. Se esta data for alterada, a data da notificação também é alterada. Dez minutos antes do prazo final do término da atividade é emitida uma notificação alertando sobre essa situação. As mensagens são pré-definidas: uma para início de atividade e outra para prazo final e sempre expressam a data e o horáro do evento em valores relativos e absolutos, para alertar bem o usuário. Independente da definição de horário de notificação, toda vez que a data e horário de

ínicio ou de prazo final de uma atividade for alterada, é emitida uma notificação aos participantes, imediatamente. A notificação é enviada por *e-mail* e também é enviada à *worklist* do usuário.

```
<Activity id=”Exemplo3”>
  <ExtendedAttributes>
        <ExtendedAttribute Name=”Locked”/>
        <ExtendedAttribute Name=”StartDate” Value=”18/01/2004”/>
        <ExtendedAttribute Name=”Notifications”>
              <Notification Date=”15/01/2004 08:40” Type=”StartDate”/>
              <Notification Days=”2” Type=”StartDate”/>
              <Notification Days=”1” Type=”StartDate”/>
              <Notification Hours=”1” Type=”StartDate”/>
              <Notification Minutes=”10” Type=”Deadline”>
        </ExtendedAttribute>
  </ExtendedAttributes>
</Actitivity>
```

Exemplo 3 – Definição de atividade

5.2 O armazenamento dos dados relevantes do *workflow*

Dados relevantes do *workflow* representam as variáveis de uma definição de processo de *workflow* ou de um pacote. Eles são usados para tomada de decisões ou para passagem de parâmetros (WfMC: WORKFLOW PROCESS DEFINITION INTERFACE, 2002). Cada instância de processo contém seu próprio conjunto de dados relevantes. Os dados podem ser de diversos tipos, entre eles os tipos básicos como *integer*, *string*, etc; e ainda tipos complexos como *array*, *record*, etc, que são compostos pelos tipos básicos ou outros tipos complexos. Cada definição de processo pode criar tipos diferentes e o número de combinações usando tipos complexos é infinito. Esses dados precisam ser armazenados de forma persistente e a opção escolhida foi armazená-los no banco de dados relacional. Os dados são armazenados em tabelas da seguinte forma:

A tabela `DataField_Instance` relaciona-se com a tabela `ProcessInstance`, que contém as informações sobre a instância de processo, e com a tabela `DataField`, que contém as informações sobre o elemento `DataField` na definição do processo. Dessa forma, é possível saber a que instância o dado se refere e como é sua estrutura. Os valores dos dados não podem ser armazenados diretamente na tabela `DataField_Instance`, uma vez que estes podem ter estruturas diferentes. Para cada tipo de dado é criada uma tabela que armazena o seu valor. Na tabela `DataField_Instance` é armazenado o endereço para acessar esse valor. Por exemplo, para armazenar um dado do tipo *string*, deve-se:

- Inserir um registro na tabela `DataFieldString` com o valor do dado.

- Essa tabela tem uma chave primária do tipo inteiro auto-numeração cujo valor é então armazenado na tabela `DataField_Instance`, juntamente com o código da instância de processo, o `Id` do pacote e o código do `DataField` correspondente.

5.3 A chamada de aplicações

A execução de uma atividade pode implicar na invocação de uma aplicação. Isso pode acontecer de maneira manual, quando o usuário interage com a máquina de *workflow* para chamar a aplicação, ou de maneira automática. Quando da chamada da aplicação são passados parâmetros, que podem ser de entrada (*IN*), saída (*OUT*) ou entrada e saída (*INOUT*). Esses parâmetros são passados por referência e correspondem a variáveis da instância do processo, que são armazenados no banco de dados. Além disso são passadas, implicitamente, uma *string* para a conexão com o banco de dados e a chave primária da instância do processo. A aplicação, portanto, precisa ser programada para acessar o banco de dados para ler esses parâmetros e devolver os valores de saída. Além disso, a aplicação precisa informar à máquina de *workflow* quando do término de sua execução. O nome e caminho do executável da aplicação são armazenados na tabela `Application`. Essa informação poderá ser obtida do arquivo XPDL, que poderá ter para isso um elemento `ExtendedAttribute` dentro do elemento `Application`. Ou então poderá ser carregada e até modificada a posteriori através da interface web da máquina de *workflow*. A aplicação pode também ser simplesmente o acesso de uma URL. Nesse caso, a máquina de *workflow* lê os valores dos parâmetros, que são só de entrada, e acrescenta-os a URL, de acordo com a sintaxe de passagem de parâmetros pelo método *get*. Se a chamada da aplicação for manual, é montado um *hyperlink* com essa *string* e este é apresentado na interface do usuário. Quando a aplicação é chamada em modo automático, é feito um redirecionamento de página pelo comando HTML *refresh*. O término da apresentação da página normalmente é manual. É possível que o autor da página implemente um formulário, onde o usuário informe à máquina de *workflow* o fim da atividade. Caso contrário, ele deverá retornar à *worklist* e informá-lo.

5.4 Os papéis desempenhados pelos usuários

Num ambiente organizacional, as pessoas possuem permissões variadas para acessarem os recursos de tecnologia de informação. Para facilitar a gerência de permissões, são criados papéis e grupos. A máquina de *workflow* CEMT possui os seguintes papéis pré-definidos:

- Administradores gerais: são as pessoas autorizadas a intervirem na execução de todas as instâncias de processo de worklow e gerenciar a máquina de workflow.
- Administradores de processo: são pessoas autorizadas a intervirem na execução das instâncias de uma determinada definição de processo.
- Criadores de processo: são as pessoas autorizadas a carregar uma definição de processo de *workflow*.
- Instanciadores de processo: para cada definição de processo há um grupo de pessoas que têm autorização para criarem uma instância desse processo.
- Agentes: são as pessoas autorizadas a executar uma atividade.

Além disso, os administradores podem criar grupos como professores, alunos, gerentes, etc.

5.5 As ações dos administradores

Existem dois níveis de administradores: administrador geral e administrador de processo. O administrador de processo pode:

- Modificar a execução de uma instância de processo de *workflow* através de mudanças no seu estado. Ele pode suspender, reiniciar, abortar ou terminar a execução de uma instância de processo. Quando a instância de processo se encontra no estado suspenso, a página `worklist.php` não disponibiliza ao usuário as opções de ação para as atividades daquele processo. Essas mudanças de estado podem ser feitas para uma instância de processo individualmente, ou para todas as instâncias de uma definição de processo (de uma ou de todas as versões).

- Bloquear, desbloquear ou mudar a restrição de data de início de uma ou de todas as instâncias de atividade de uma instância de processo ou de todas as instâncias daquela definição de processo.

- Bloquear, desbloquear ou mudar a restrição de data de início de uma definição de atividade. Neste caso a mudança será aplicada às próximas instâncias de atividade criadas.

- Suspender, reiniciar, abortar ou terminar uma ou todas as instâncias de atividade de uma instância de processo ou de todas as instâncias daquela definição de processo.

- Apagar a definição de processo, desde que não haja instâncias deste em execução ou suspensas.

- O administrador geral pode fazer as mesmas operações de um administrador de processo, em um escopo geral, ou seja, para todas as definições de processo. Pode ainda gerenciar as contas de usuários, papéis e grupos e fazer alguma manutenção necessária na máquina de *workflow*.

5.6 A autenticação dos usuários

Ao acessar a máquina de *workflow* via web, a primeira página mostra um formulário pedindo um *login* de usuário e uma senha. Além disso existe um *hyperlink* para que usuários novos possam se cadastrar. Quando o usuário se cadastra, a máquina de *workflow* grava na tabela `Usuario` o *login* do usuário e alguns dados pessoais. Ao invés de armazenar a senha do usuário diretamente, é feito o cálculo de uma função *hash* e o valor resultante é armazenado na tabela. Essa função utiliza o algoritmo MD5, que é bastante utilizado nos protocolos de criptografia atuais. Uma função de *hash* tem como características desejáveis:

- Uma baixa taxa de colisão, ou seja, dado um valor resultante da aplicação da função, a probabilidade de se encontrar outro valor original que gere o mesmo valor resultante é muito pequena.

- Dado um valor resultante da aplicação de uma função de *hash*, é computacionalmente inviável calcular o valor original.

A vantagem da utilização de uma função de *hash* é que, em caso de algum acesso indevido ao banco de dados, não se conseguem obter as senhas dos usuários. A obtenção de um valor de *hash* não acarreta quebra de segurança.

Após o usuário informar seu nome e senha na página inicial, é chamada a página `autentica.php`. Essa página cria uma sessão e registra nelas essas duas variáveis. A função autentica(nome,senha) foi implementada e colocada na página `wapi.php`, que é incluída em todas as páginas do projeto. Todas essas páginas, com exceção da inicial e daquelas que cadastram um novo usuário, chamam a função autentica para conferir se o nome de usuário e a senha são válidos. Se a resposta for negativa, a página é redirecionada para a página inicial, `index.HTML`. Dessa forma, evita-se que um usuário consiga acessar uma página qualquer, sem antes passar pela página inicial e ter seu nome e senha aceitos.

5.7 O menu de opções

Uma vez que o usuário tenha sido autenticado, a interface *web* apresenta, em um quadro vertical à esquerda, o menu de opções da máquina de *workflow*, como pode ser visto na figura 5.2. As opções são: ver os pacotes de definições de processos carregados, as definições de processos, carregar uma nova definição de processo, ver os processos instanciados e ver a lista de atividades alocadas ao usuário autenticado, que também é chamada de *worklist*.

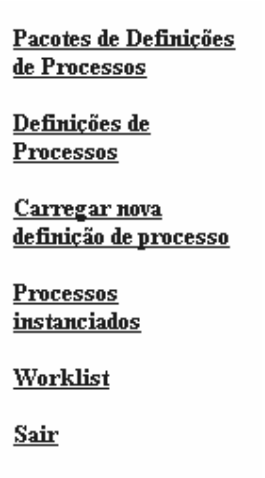


Figura 5.2: Menu de opções

5.8 A carga de definições de processos

As definições de processos são carregadas fazendo-se uma cópia do arquivo XPDL para o servidor, um processamento desse arquivo através do compilador XPDL para SQL e a execução de *script*s SQL. Um problema encontrado ao inserir, no banco de dados, os pacotes definidos em arquivos XPDL, foi a possibilidade de colisão no valor do atributo `Id` do pacote. Esse atributo é utilizado como chave primária no banco de dados e portanto necessita ser único. No entanto os arquivos XPDL podem ser produzidos por pessoas diversas, o que torna a coincidência inevitável. Uma das alternativas para solucionar o problema seria mudar a chave primária da tabela `Package` para um campo do tipo inteiro com auto-incremento. Mas isso permitiria que um mesmo pacote fosse carregado múltiplas vezes e, portanto, essa alternativa foi descartada. A solução encontrada foi manter o `Id` como chave primária e informar o usuário em caso de

colisão. O usuário pode, nesse caso, informar um novo valor para o `Id` do pacote ou gravá-lo como uma nova versão.

A carga das definições de processos é feita por três páginas, escritas em PHP. A primeira página, `carregaDefinicao.php`, é mostrada na figura 5.3 e, a exemplo do que é implementado no OFBIZ (figura 2.21), mostra um formulário onde o usuário pode informar o nome e a localização do arquivo XPDL contendo a definição do processo. Esse arquivo pode estar no sistema de arquivos do computador local (ou em um computador acessível por este, via rede) ou em uma *URL*. Após clicar no botão, a segunda página `carregaDefinicao2.php` é executada.

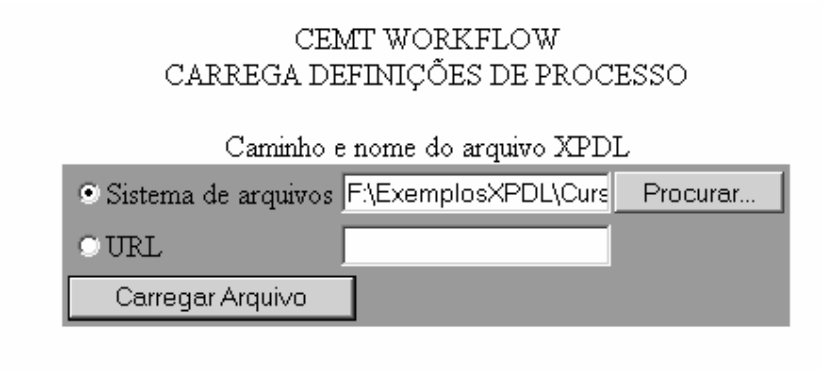


Figura 5.3: Carga da definição do processo de *workflow*

Esta página copia o arquivo para um diretório no servidor em que a máquina de *workflow* está instalada. O arquivo recebe um prefixo com o nome do usuário autenticado, seguido por um ponto. Isso é feito para se evitarem colisões de arquivos quando mais de um usuário estiver carregando definições de processo ao mesmo tempo. Espera-se que um mesmo usuário não faça duas operações desse tipo ao mesmo tempo. Após feita a cópia, são executados três programas, cujos comportamentos são descritos a seguir:

- o programa `getPackageId` lê o arquivo XPDL, obtém o valor do atributo `Id` do elemento `Package` e grava-o em um arquivo. A página PHP lê esse arquivo e monta uma consulta SQL para ver se já existe um pacote com aquela identificação. Em caso afirmativo é mostrado um formulário que pede ao usuário que renomeie a identificação do pacote ou escolha a opção de gravar o pacote como uma nova versão, como pode ser visto na figura 5.4. Esse formulário chama a página `carregaDefinicao3.php`. A máquina de *workflow* possui um controle de versão dos pacotes de definições de processo carregadas. Quando um pacote é carregado pela primeira vez, ele recebe o valor 1 para a versão. Cada vez que o usuário carrega um pacote com a mesma identificação, escolhendo a opção de gravar como nova versão, o novo pacote é gravado com o valor da versão incrementado. Quando isso acontece, a máquina muda o estado das definições de processo pertencentes a pacotes com versões anteriores para o estado inativo.

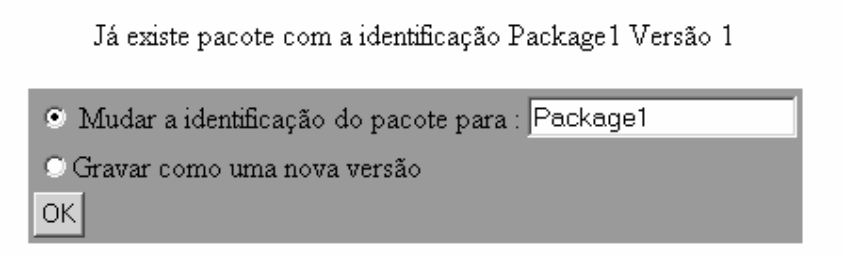


Figura 5.4: Mostra colisão entre as identificações dos pacotes

- Se não existe nenhum pacote no banco de dados com mesmo valor do atributo `Id`, é chamado o programa `insertPackage`, que lê novamente o arquivo XPDL e grava, em um arquivo, um comando SQL que insere um registro na tabela `Package`. Esse comando SQL é lido e executado pela página PHP.

- Por fim é chamado o programa compilador que faz um *parsing* completo do arquivo XPDL, obtendo todas as suas informações. Os erros porventura encontrados são gravados em um arquivo de nome `erros.txt`. A página PHP lê esse arquivo e o escreve para a saída em HTML. Caso não sejam encontrados erros, o compilador grava comandos SQL em um arquivo de nome `script.txt`. A página PHP, nesse caso, lê o arquivo de *script* e executa os comandos SQL, linha por linha.

É possível que, em caso de renomeação do `Id` do pacote, haja colisão com outros pacotes existentes e, nesse caso, a página `carregaDefinicao3.php` continuará sendo chamada repetidamente até que o usuário informe um valor inédito para o atributo `Id`. Após isso é feita a inserção do pacote, como descrito anteriormente.

Todos os arquivos gravados têm como prefixo o *login* do usuário autenticado, evitando-se assim conflitos quando dois usuários estejam carregando pacotes ao mesmo tempo. Antes da execução dos comandos SQL , é executado um comando de início de transação no banco de dados. Caso ocorra algum erro durante a execução do arquivo de *script*, um comando de *rollback* é enviado ao MySQL. Esse comando desfaz todas as alterações efetuadas desde o início da transação. Do contrário é enviado um comando de *COMMIT*, que confirma as operações executadas.

5.9 Vendo a lista de definições de processos

A página `listaProcessos.php` mostra a lista de definições de processo (figura 5.5). Essas definições de processos são mostradas em duas tabelas, sendo uma delas para as que estejam no estado inativo e a outra para aquelas que estejam no estado ativo. Assim como no Domino Workflow, definições de processos no estado ativo são aquelas que podem gerar instâncias. Logo que são carregadas, as definições de processos assumem o estado inativo. A cada linha da tabela são colocados *hyperlinks* para ações que o usuário pode requisitar em relação a estas definições de processos. As definições de processos inativas podem ser ativadas enquanto que as definições de processos ativas podem ser desativadas ou podem ser instanciadas. Uma definição de processo pode ter várias versões, mas apenas uma pode estar no estado ativo. Qualquer versão pode ser ativada, não necessariamente a mais recente. Quando uma versão é ativada, todas as outras versões daquela definição de processo são desativadas. Quando uma versão é

desativada, ela não pode gerar novas instâncias, mas as instâncias já criadas podem continuar sua execução normalmente. Apenas definições de processos de nível de acesso público podem ser ativadas via ação de um usuário. Processos de acesso privado não têm o conceito de estado (ativo/inativo) e são concebidos para serem utilizados como subprocessos, que são chamados por outros processos, dentro do mesmo pacote e da mesma versão. A mudança de estado, requisitada nesta página, é efetuada por outra página, `mudaEstado.php`, que chama a função WMChangeProcessDefinitionState. Em caso de sucesso, nenhuma informação é enviada e a página `listaProcessos.php` é novamente exibida, através de um comando HTML de redirecionamento de página.

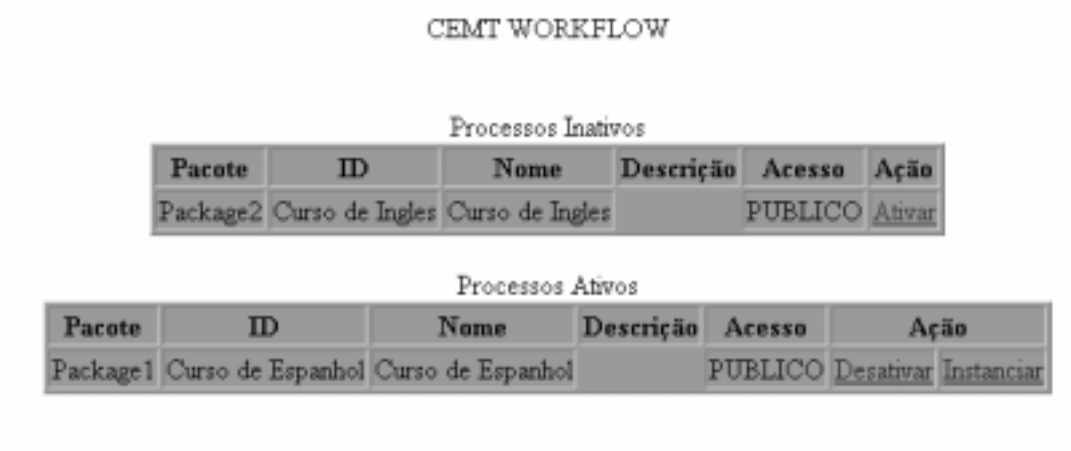
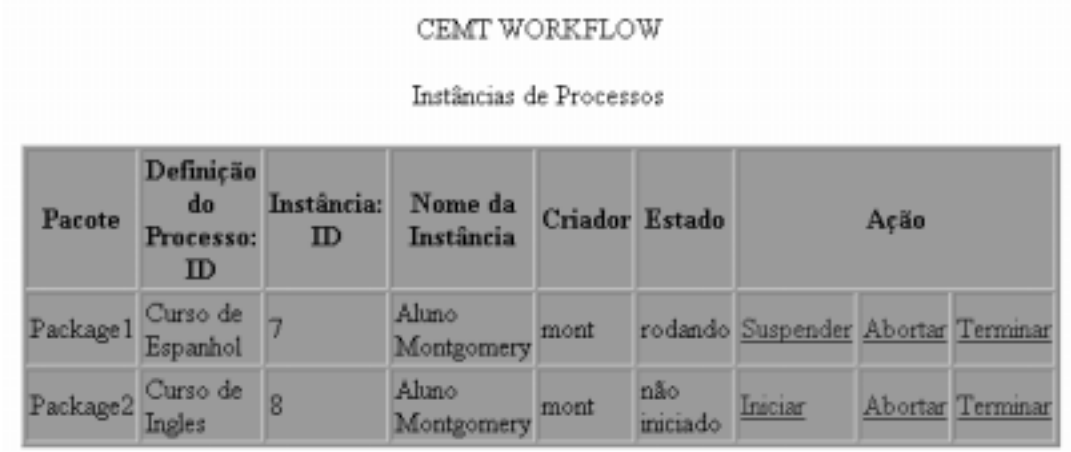


Figura 5.5: Lista de definições de processos

5.10 Instanciando um processo

Ao clicar no *hyperlink* Instanciar, na página `listaProcessos.php`, é carregada a página `criaInstancia.php`. Esta página mostra um formulário, que pede um nome para a instância, e chama a página `criaInstancia2.php`. Esta segunda página cria um registro na tabela `ProcessInstance` com as informações que lhe são passadas: `ID` do pacote, `ID` da definição do processo e nome da instância. A tabela contém ainda um campo inteiro com auto-numeração para servir de chave primária. Esta ação é executada pela função WMCreateProcessInstance. Após a execução, a página se redireciona para a página `listaInstanciasDeProcessos.php` (figura 5.6). Toda vez que um usuário pede a criação de uma instância de processo, é criada também uma instância do pacote a que ele pertence. Isso se faz necessário devido à existência de dados relevantes de *workflow* (elemento `DataField`) que são comuns ao pacote e que precisam ser instanciados e referenciados. Por outro lado, quando um processo é instanciado a pedido de outro processo que já esteja rodando, não é criado outro pacote e os dois processos compartilham a mesma instância de pacote.


CEMT WORKFLOW

Instâncias de Processos

Pacote	Definição do Processo: ID	Instância: ID	Nome da Instância	Criador	Estado	Ação		
Package1	Curso de Espanhol	7	Aluno Montgomery	mont	rodando	Suspender	Abortar	Terminar
Package2	Curso de Ingles	8	Aluno Montgomery	mont	não iniciado	Iniciar	Abortar	Terminar

Figura 5.6: Lista de instâncias de processos.

5.11 As instâncias de processos

As instâncias de processos, assim que criadas, assumem o estado de `nãoIniciado`. A página `listaInstanciasDeProcessos.php` lista, em uma tabela, as instâncias de processos, seus estados e as ações que o usuário pode tomar sobre elas. As ações são: iniciar, suspender, abortar e terminar. Para cada instância de processo, são colocados *hyperlinks* apenas para as ações possíveis de serem tomadas, de acordo com as transições permitidas definidas na figura 4.1. Assim, para uma instância que esteja no estado `nãoIniciado` são mostrados *hyperlinks* com as ações Iniciar, Abortar e Terminar. As instâncias no estado `rodando` poderão ser suspensas, abortadas ou terminadas. Finalmente, as instâncias no estado *supended* poderão ser reiniciadas, abortadas ou terminadas. Para as instâncias no estado `fechado` (`completado`, `abortado` ou `terminado`) não é apresentada nenhuma opção de ação. Nenhuma instância pode, via intervenção direta do usuário, ter seu estado modificado para o estado `completado` pois esse estado só é alcançado após o término das atividades previstas na definição do processo de *workflow*. Após clicar no *hyperlink* correspondente à ação desejada, é chamada a página `mudaEstadoInstanciaProcesso.php`. Essa página faz a mudança de estado pedida, chamando a função da API adequada para isso: WMStartProcess, WMAbortProcessInstance, WMTerminateProcessInstance ou WMChangeProcessInstanceState. Após, a página se redireciona para a página `listaInstanciasDeProcessos.php`.

5.12 As instâncias de atividades

As instâncias de atividades são armazenadas na tabela `ActivityInstance`. Essa tabela se relaciona com a tabela `ProcessInstance`, com a tabela `WorkflowProcess` e com a tabela `Activity`, que contém, respectivamente, a instância do processo, a definição deste e a definição da atividade. A tabela `ActivityInstance` possui ainda, como chave primária, um campo de inteiro com auto-numeração. Isso é necessário pelo fato de que, em uma mesma instância de processo poder haver mais de uma instância da mesma definição de atividade, se ocorrer um ciclo. A tabela guarda ainda as datas e horários em que a atividade ficou pronta para ser executada, quando foi iniciada e quando foi completada.

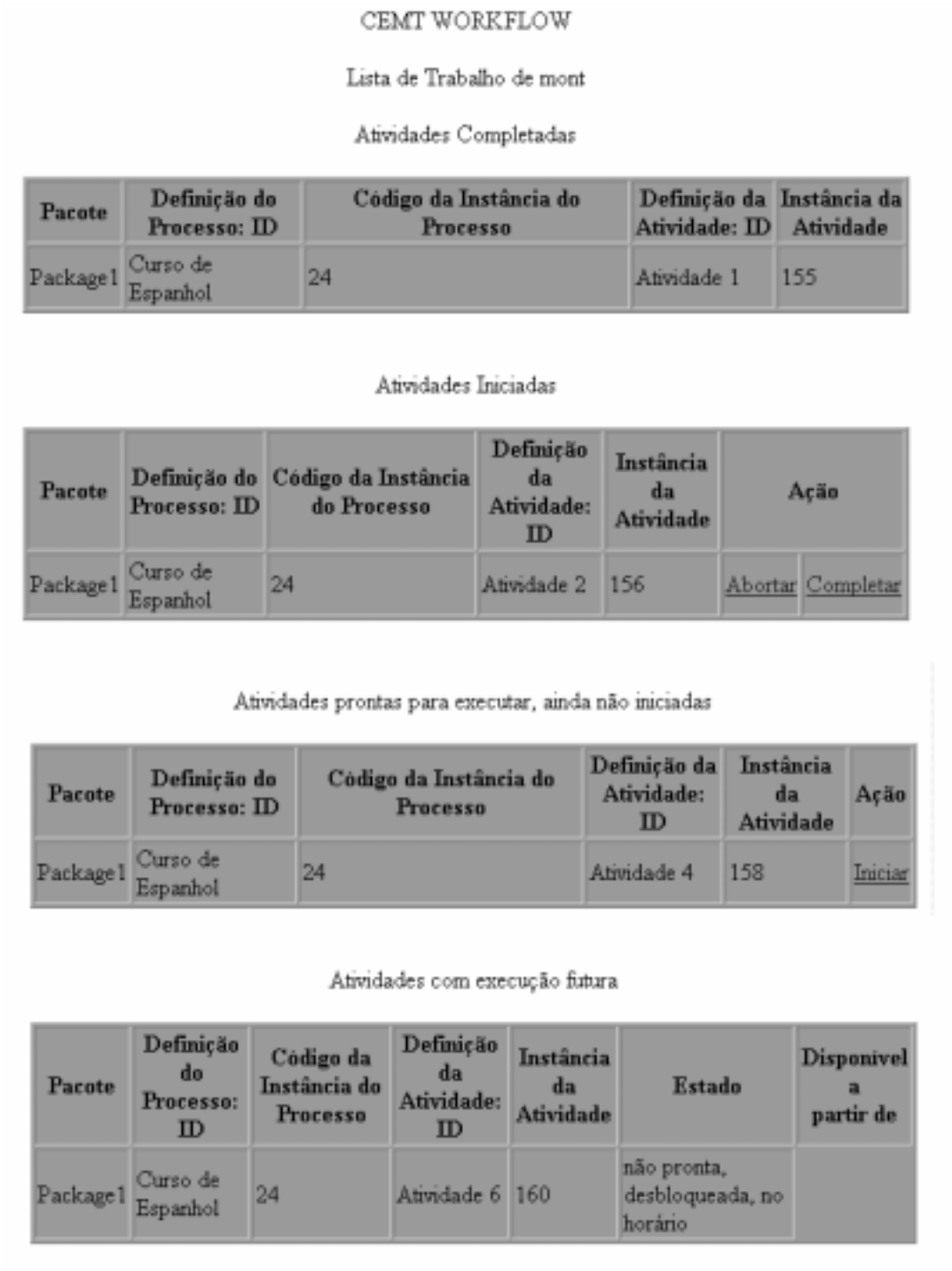
Uma instância de atividade possui um participante, que pode ser conhecido desde a definição da atividade, ou não. Em muitos casos é difícil saber exatamente quem executará uma atividade, quando da definição do processo. Além disso, uma mesma definição de processo pode ser executada por várias pessoas. Por isso, é importante trabalhar com papéis. Esses papéis podem ser atribuídos às atividades na sua definição ao passo que a instância da atividade será alocada ao agente correspondente àquele papel, em tempo de execução. A exemplo do que foi implementado no Domino Workflow, foi criado o papel ProcessCreator, que corresponde ao usuário que cria a instância de processo. Assim, se na definição do processo, existe um participante do tipo ROLE e nome igual a ProcessCreator, o executor das atividades, que tenham esse participante como *performer*, será o usuário que criou a instância do processo.

5.13 A lista de trabalho do usuário

Uma vez iniciada uma instância de processo de *workflow*, suas instâncias de atividades são criadas. Uma abordagem simples é criar apenas as atividades que estejam prontas para executar, ou seja, a primeira atividade do processo, inicialmente, e as outras à medida que as transições para elas sejam executadas. Outra forma de implementação é criar todas as atividades possíveis de serem executadas, sendo que algumas estarão em um estado que não permite sua execução imediata, até que as pré-condições sejam satisfeitas. Neste projeto foi escolhida a segunda opção. Isso permite mostrar ao usuário uma visão um pouco mais a frente sobre as suas tarefas, como pode ser visto na figura 5.7.

Nessa página são mostradas, em tabelas separadas, as atividades completadas, aquelas iniciadas, aquelas que estão prontas para serem iniciadas e aquelas com execução futura, por esperarem alguma condição, que pode ser: uma transição que tenha a atividade como destino ser executada (e a condição da transição ser avaliada como verdadeira); um desbloqueio ser feito pelo administrador; ou uma restrição de data ser atendida. Quando existe uma restrição de data, é mostrada a data a partir da qual a atividade pode ser iniciada. Para atividades prontas para serem executadas, é disponibilizado um *hyperlink* para a ação Iniciar, enquanto para as atividades iniciadas é disponibilizado um *hyperlink* para a ação Completar.

Devem ser consideradas algumas dificuldades que se apresentam: a instanciação de atividades que façam parte de um ramo do processo que nunca venha a ser executado e a necessidade de instanciação, durante a execução do processo, de atividades pertencentes a ciclos, dentro do processo.


CEMT WORKFLOW

Lista de Trabalho de mont

Atividades Completadas

Pacote	Definição do Processo: ID	Código da Instância do Processo	Definição da Atividade: ID	Instância da Atividade
Package1	Curso de Espanhol	24	Atividade 1	155

Atividades Iniciadas

Pacote	Definição do Processo: ID	Código da Instância do Processo	Definição da Atividade: ID	Instância da Atividade	Ação	
Package1	Curso de Espanhol	24	Atividade 2	156	Abortar	Completar

Atividades prontas para executar, ainda não iniciadas

Pacote	Definição do Processo: ID	Código da Instância do Processo	Definição da Atividade: ID	Instância da Atividade	Ação
Package1	Curso de Espanhol	24	Atividade 4	158	Iniciar

Atividades com execução futura

Pacote	Definição do Processo: ID	Código da Instância do Processo	Definição da Atividade: ID	Instância da Atividade	Estado	Disponivel a partir de
Package1	Curso de Espanhol	24	Atividade 6	160	não pronta, desbloqueada, no horário	

Figura 5.7: Worklist

A máquina de *workflow* deverá listar essas atividades, classificando-as de acordo com as características a seguir:

- Atividades de execução obrigatória e única
- Atividades de execução opcional e única
- Atividades de execução obrigatória e múltipla
- Atividades de execução opcional e múltipla

Esta classificação deverá ser atualizada durante a execução do *workflow*. Por exemplo, sejam duas atividades opcionais, mutuamente excludentes. Após a avaliação de uma determinada condição, uma delas (e talvez a seqüência de atividades que essa precede) se torna obrigatória, enquanto a outra (e outra seqüência de atividades) se torna obsoleta e, portanto, deve ser apagada.

A máquina deverá ainda tentar listar as atividades na ordem correta de sua execução. O primeiro critério para se definir qual a atividade deve ser listada primeiro é a relação de sucessão ocorrida na definição do *workflow*. Porém, para atividades que se encontrem em ramos paralelos do *workflow*, outros critérios necessitam ser implementados como:

- listar primeiro as atividades no estado prontas para execução (`pronto desbloqueado noHorário`)
- listar as atividades desbloqueadas antes das bloqueadas;
- listar aquelas que não tenham restrição para data de início, ou cujas datas de início estejam mais próximas;
- listar aquelas cujas prioridades foram definidas como maiores;
- listar aquelas cujo prazo máximo para conclusão (*deadline*) esteja mais próximo.

Esta é uma abordagem simplista e parte do pressuposto de que a informação sobre atividades não prontas para a execução tem como finalidade apenas mostrar ao usuário uma idéia sobre seu trabalho futuro e essa informação se torna mais precisa à medida que o futuro se torna mais próximo.

Cada instância de atividade está relacionada a um participante, que pode ser de vários tipos. Quando o participante é um usuário, a atividade será listada para este em sua tela pela página *worklist*.php. Nesta página, o usuário poderá ver as atividades já fechadas, aquelas que estão aguardando a sua intervenção para serem executadas e aquelas que o usuário deverá executar no futuro. Além de ver as atividades, o usuário poderá tomar ações como iniciá-las, abortá-las, terminá-las ou completá-las. As atividades podem chamar aplicações que executam e finalizam independentemente do usuário, ou podem ser totalmente dependentes deste. Por exemplo, uma atividade poderá mostrar um documento ao usuário para que este o leia e esperar que esse usuário informe, à máquina de *workflow*, que já completou essa leitura.

5.14 Execução do processo de *workflow*

Quando um processo de *workflow* é instanciado, todas as suas atividades e transições são instanciadas. Uma instância de transição possui referências à definição da transição e às instâncias de atividades de origem e destino desta. À medida que as instâncias de atividades vão completando sua execução, a máquina de *workflow* procura pelas instâncias de transições que têm essa atividade como origem e, para cada instância de transição, avalia a sua condição. Para aquelas em que o resultado foi verdadeiro, a instância de transição é apagada e podem ocorrer as seguintes situações:

- A instância de atividade destino da instância de transição não é um join e seu estado é uma composição de `nãoPronto`[2]. Nesse caso o estado da atividade é mudado para `pronto`, preservando-se as outras características.
- A instância de atividade destino da instância de transição não é um join e seu estado é uma composição de `pronto`. Isso significa que existe um ciclo na execução do *workflow* e uma nova instância de atividade é criada. O campo `life` dessa atividade tem o valor incrementado em relação à atividade precedente. A atividade é colocada no estado `pronto`. São criadas instâncias para todas as atividades e transições subseqüentes a essa atividade. As atividades terão o mesmo valor para o campo `life`. Mas antes da criação de uma instância de atividade é verificado se já não existe uma instância que se refira à mesma definição de atividade e mesma instância de processo e tenha o mesmo valor para `life`. Se isso acontecer, a criação da instância de atividade deve ser ignorada. O algoritmo empilha as instâncias de atividades criadas e repete essa execução para cada atividade desempilhada.
- A instância de atividade destino é um `AND-JOIN`. Verifica-se se existem outras transições que referem à mesma `TransitionRestriction` do AND-JOIN. Em caso negativo, significa que todas as transições necessárias já foram executadas e a atividade `AND-JOIN` tem seu estado mudado para `pronto`. Em caso positivo, significa que outras transições precisam ainda ser executadas e nada é feito.
- A instância de atividade destino é um `XOR-JOIN`. Nesse caso basta uma transição ser executada para que a atividade do `XOR-JOIN` tenha seu estado mudado para `pronto` e é isso o que é feito. Além disso, todas as outras transições pertencentes à mesma `TransitionRestriction` são apagadas, pois não são mais necessárias.

Quando a avaliação da condição de uma transição resulta em falso, pode ocorrer que a instância de atividade destino da transição (e talvez a seqüência que ela preceda) torne-se inatingível pelo fluxo de trabalho. Assim, as seguintes verificações são feitas nessa atividade:

- Se a atividade é um `AND-JOIN` e a transição em questão faz parte de sua `TransitionRestriction`, ela é inatingível.

- Caso contrário, a transição é apagada. Se não restarem transições, cujo destino seja aquela atividade, ela é inatingível.

As atividades inatingíveis são apagadas e as transições dela originadas são marcadas com o valor falso. E o mesmo algoritmo é aplicado a todas essas transições, recursivamente.

Atividades que tenham participante do tipo `system`, ou não tenham participante, são executadas pela máquina de *workflow*. O programa engine, escrito em C e que funciona como um serviço, verifica, a espaços de tempo regulares, se existem atividades desse tipo e prontas para executar. Além disso o programa verifica as atividades com

[2] Os estados para atividades não inciadas são uma combinação de três características: (`pronto` ou `nãoPronto`) + (`bloqueado` ou `desbloqueado`) + (`noHorário` ou `foraDoHorário`).

restrições de datas para início ou término, comparando essas datas com a data atual e mudando o estado da atividade se for o caso.

6 CONCLUSÃO E TRABALHOS FUTUROS

Embora existam inúmeras máquinas de *workflow*, foram encontradas poucas máquinas com licença de software livre e funcionamento adequado. Dentre os softwares pesquisados neste trabalho, apenas o OFBIZ apresentou a possibilidade de interpretar definições de processo escritas de acordo com o padrão da Interface 1, definido pela WfMC (WfMC: WORKFLOW PROCESS DEFINITION INTERFACE, 2002), o que possibilitaria a sua interoperação com outras ferramentas de definições de processo que utilizam esse padrão. No entanto, esse aspecto e a existência de um banco de dados contendo a estrutura pronta para armazenar essas definições de processo são os únicos aspectos positivos de OFBIZ (quanto ao uso de *workflow*). OFBIZ ainda mostra incorreções no seu funcionamento e não está completo para ser usado como uma máquina de *workflow*.

A alocação de usuários a atividades no Openflow funciona bem para ambientes de produção. Por exemplo, um escritório recebe pedidos de vários clientes. Para cada pedido é instanciado um processo e as atividades desse processo podem ser executadas por qualquer dos funcionários que se enquadrem num determinado perfil. Esses funcionários são agrupados em um papel. Assim, um mesmo funcionário pode executar uma determinada atividade para várias instâncias de processos e uma instância de processo pode ter cada uma de suas atividades executadas por um funcionário diferente.

O Openflow oferece três maneiras de alocar atividades a usuários: Em *pull* as atividades ficam disponíveis aos usuários em um *pool* e os usúarios decidem que atividades pegar para si. Nesse caso caberia ao usuário a responsabilidade de não fazer uma atividade que deveria ser feita por outro. Em *manual push*, um usuário aloca uma atividade para outro. Pode haver nesse caso a figura de um coordenador que possa distribuir as atividades para os usuários. Em um volume grande de atividades, fazer isso manualmente é inviável. A última opção é o *automatic push*, onde uma aplicação cuida da alocação das atividades aos usuários e pode ser viável para resolver esse tipo de problema. Domino Workflow permite uma solução que é guardar em uma variável a identificação do usuário que executou a primeira atividade do processo. Na definição do processo pode ser estabelecido que as atividades subseqüentes sejam permitidas ao usuário executor da primeira atividade, que será conhecido em tempo de execução. No Reactor pode ser implementada uma política que copie o nome do usuário criador da instância para a lista de controle de acesso das suas atividades. A máquina de *workflow* CEMT usa procedimento semelhante ao do Reactor.

As máquinas de *workflow* pesquisadas não têm uma preocupação com o inter-relacionamento entre as instâncias de processos. Essas instâncias são tratadas sempre de maneira individualizada e independente umas das outras, o que não permite a

implementação de atividades de trabalho em grupo. O Openflow não permite que uma mesma atividade seja alocada a mais de uma pessoa. No Reactor, uma instância de atividade pode ser permitida a duas pessoas, mas após a primeira pessoa executar a atividade, ela é retirada da lista de atividades da segunda. Este tipo de atividade é especificada na máquina de *workflow* CEMT como sendo do tipo *Pool*. O Domino Workflow permite que uma instância de atividade seja alocada a uma equipe, mas esta instância de atividade pertence a apenas uma instância de processo. A especificação da máquina de *workflow* CEMT prevê a possibilidade de implantação de atividades em grupo. Estas instâncias de atividades em grupo são efetivamente executadas por participantes de instâncias diferentes da mesma definição de processo.

O Openflow permite que o usuário gere uma exceção em uma atividade. Não foi testada a situação de uma exceção ser provocada por algum comportamento anormal da aplicação. Depois o administrador pode tratar a exceção e redirecionar o fluxo da execução para qualquer atividade daquele ou de outro processo. Isso é feito especificamente na instância de processo com problema e funciona bem quando a exceção é localizada. No Domino Workflow, o proprietário da instância de processo pode intervir na sua execução. Já no Reactor, o administrador não tem como tomar qualquer ação para resolver um problema de execução anormal de uma instância. O tratamento de exceções da máquina de *workflow* CEMT ainda não foi especificado.

Openflow e Domino Workflow trabalham com papéis. Reactor não utiliza o conceito de papéis para atribuir permissões a grupos de usuários. Essa característica é importante pois permite maior flexibilidade em relação às mudanças que podem ocorrer no desempenho de funções na organização. Openflow aproveita os usuários e papéis definidos pelo Zope e isso pode ser bastante útil se o Zope já for um servidor bem utilizado na organização. Domino Workflow busca suas informações em um diretório da organização, podendo utilizar um específico para suas funções ou um diretório geral da organização. A especificação da máquina de *workflow* CEMT também prevê a utilização de papéis, sendo alguns pré-definidos e outros definidos pelo administrador da máquina.

Julgou-se que o aproveitamento do código desenvolvido no Openflow e no OFBIZ seria inviável. Openflow necessitaria uma compatibilização com o padrão da Interface I da WfMC e o OFBIZ apresenta muitas incorreções e encontra-se bastante incompleta. A máquina de *workflow* CEMT segue a idéia do OFBIZ de interpretar os arquivos XPDL e trasnportar suas informações para um banco de dados relacional. Isso foi possível com o desenvolvimento de um compilador que faz um trabalho de *parsing* no arquivo XPDL, verifica seus erros e possibilita a carga das informações de definições de processo em um banco de dados relacional. Embora o AW faça suas próprias verificações de erro, a máquina de workflow não fica isenta dessa verificação, uma vez que ela poderá receber definições de processo geradas por outras ferramentas.

Em suma, os estudos e testes feitos nessas máquinas de *workflow* existentes nortearam algumas decisões de projeto, como o uso da plataforma web, de um banco de dados relacional, de uma linguagem de *script* para servidor web e a criação de um serviço do sistema operacional. O uso de um banco de dados relacional permite a fácil recuperação de informações sobre os processos e as pessoas deles participantes, para fins estatísticos e gerenciais.

Foram implementadas ou especificadas as funcionalidades básicas de uma máquina de *workflow* de uso geral. O protótipo encontra-se disponível para testes, provisoriamente, na máquina cemt.inf.ufrgs.br, do Instituto de Informática da UFRGS. O protótipo atende aos padrões da WfMC e foram propostas algumas extensões que aumentam as possibilidades de controle de execução do *workflow*, sendo úteis a aplicações de uso geral e, em particular, a aplicações voltadas ao ensino a distância.

Este trabalho contribui para o projeto CEMT com a apresentação de um protótipo de uma máquina de *workflow* e segue a política do projeto de trabalhar com software livre. É importante ressaltar que este trabalho integra-se ao editor gráfico de *workflow* AW (TELECKEN et al., 2002), desenvolvido dentro do mesmo projeto. Isso foi possível graças ao uso dos padrões da WfMC, que permitem ainda a interação com vários softwares de *workflow* de terceiros. Este protótipo servirá como ponto de partida para outros projetos dentro do PPGC, podendo ser utilizado para testes e também pode ser aperfeiçoado com a inclusão de funcionalidades avançadas. Faz-se necessária, ainda, uma avaliação deste protótipo. É perfeitamente viável a evolução deste protótipo para uma máquina de *workflow* completa e isto poderá ser feito no Instituto de Informática da UFRGS por alunos do PPGC. Esta máquina terá aplicabilidade no suporte ao ensino a distância e em tarefas administrativas.

Quando ocorre uma mudança na definição de um processo que já esteja executando, o Openflow adota a política de migração para o *workflow* final. A exemplo do Domino Workflow e do Reactor, a máquina de *workflow* CEMT adota a concomitância para finalização. Outras políticas de migração podem ser implementadas no futuro.

Devido ao enorme volume de trabalho que seria conceber e implementar uma máquina de *workflow* completa, alguns aspectos não foram abordados neste trabalho. Dentre eles podem ser citados os aspectos de segurança e interface com o usuário.

A preocupação com a segurança deve começar no banco de dados para não permitir o acesso de pessoas não autorizadas. O banco de dados deveria ser acessível apenas através de uma API, onde, a cada chamada de operação, seria verificada a permissão do usuário autenticado. O banco de dados deveria contemplar a implementação de diversos papéis para os usuários, entre eles os papéis administrativos. O autocadastramento também é uma opção simplista, adequada a essa fase de teste do protótipo, e deverá ser substituído por uma requisição de cadastramento a ser confirmada por um administrador. Várias outras opções de administração de contas de usuários demandam uma implementação, como por exemplo o controle de tentativas de acesso não autorizado, expiração de senhas, trocas de senhas e verificação de senhas fracas.

A interface *web* foi desenvolvida com objetivo de demonstração do protótipo. A API recomendada pela WfMC na Interface 2 (WfMC: WORKFLOW CLIENT APPLICATION, 1997) é independente da interface *web*. Assim, outras interfaces *web*, que acessem essa API, poderão ser implementadas por terceiros e poderão ser aperfeiçoadas em relação à facilidade de uso e à separação entre funcionalidades acessíveis aos diversos papéis, como por exemplo aos usuários comuns e aos administradores.

Atualmente, a máquina de *workflow* utiliza apenas uma parte da saída do AW, que é a definição do processo de *workflow* escrita em XPDL. O arquivo em SVG gerado pelo

AW poderia também ser trabalhado para apresentar uma *worklist* em modo gráfico, combinando o arquivo SVG original com as informações sobre as instâncias de processos, de forma que as instâncias de atividades poderiam ser mostradas em cores correspondentes a cada estado.

REFERÊNCIAS

AHO, A. V. **Compilers**. Reading: Addison-Wesley, 1988.

BOURRET, R. **XML and Databases**. Disponível em: <http://www.rpbourret.com/xml/XMLAndDatabases.htm>. Acesso em: 07 jul. 2003.

CASATI, F. et al. Workflow Evolution. In: INTERNATIONAL CONFERENCE ON CONCEPTUAL MODELING, ER, 15., 1996, Cottbus. **Conceptual Modeling**: proceedings. Berlin: Springer-Verlag, 1996. (Lecture Notes in Computer Science, 1157).

CHAMPION, M. Storing XML in Databases. **EAI Journal**, [S.l.],p. 53-55, Oct. 2001. Disponível em: <http://www.eaijournal.com/PDF/StoringXMLChampion.pdf> . Acesso em: 07 jul. 2003.

DODDS, L. **XML and Databases? Follow your Nose**. Disponível em: <http://www.xml.com/lpt/a/2001/10/24/follow-yr-nose.html>. 2001. Acesso em: 18 jun. 2003.

HOLLINGSWORTH, D. **The Workflow Reference Model**. Document Number TC00-1003. 1995. Disponível em: <http://wfmc.org/standards/docs/tc003v11.pdf>. Acesso em: 23 fev. 2003.

LIMA, J. V. de; QUINT, V.; LAYAIDA, N.; EDELWEISS, N.; ZEVE, C. M. D.; PINHEIRO, M. K.; TELECKEN, T. L. The Conception of Cooperative Environment for Editing Multimedia Documents with Workflow Technology (CEMT). In: PROTEM-CC, 4., 2001, Rio de Janeiro. **Projects Evaluation Workshop**: international cooperation: proceedings.Brasília: CNPQ, 2001. p. 542-560.

MASON, T.; BROWN, D. **Lex & yacc**. [S.l.]: O'Reilly & Associates, 1990.

MYSQL Reference Manual for Version 4.0.14. Disponível em: <http://www.mysql.com/*downloads*/*download*.php?file=*Downloads*%2FManual%2Fmanual.tar.gz&pick=mirror>. Acesso em: 10 set. 2003.

NIELSEN S. P. et al. Using Domino Workflow. 2000. Disponível em: <http://www.redbooks.ibm.com/pubs/pdfs/redbooks/sg245963.pdf>. Acesso em: 09 mar. 2003.

OFBIZ – Open For Business Software. 2002. Disponível em: <http://www.ofbiz.org> . Acesso em: 09 mar. 2003.

OPENFLOW: an introduction. 2002. Disponível em: <http://www.openflow.it/Documentation/documentation/OpenFlowIntroduction>. Acesso em: 11 nov. 2002.

OPENFLOW. Disponível em: <http://www.openflow.it> . Acesso em: 23 fev. 2003.

REACTOR Evaluator's Guide. Disponível em: <http://www.oakgrovesystems.com/pdfs/R5_Evaluation_Guide.pdf>. Acesso em: 09 mar. 2003.

TELECKEN, T.L.; LIMA, J. V. de; ZEVE, C. M. D.; MACIEL, C.; BORGES, T. Modeling of Courses through Workflow Using the Standard SVG/XML. In: WORLD EDUCATIONAL MULTIMEDIA, HYPERMEDIA & TELECOMUNICATIONS, 2002, Denver. **Proceedings**...Norfolk:AACE, c 2002.p. 1940-1941.

VATTON, I. **Welcome to Amaya**. Disponível em: <http://www.w3.org/Amaya>. Acesso em: 18 fev. 2003.

WfMC: Workflow Process Definition Interface – XML Process Definition Language. Document Number WfMC-TC-1025. Document Status – 1.0 Final Draft. 2002. Disponível em: <http://www.wfmc.org/standards/docs/TC-1025_10_*XPDL*_102502.pdf>. Acesso em: 23 fev. 2003.

WfMC-Workflow Management Coalition. Disponível em: <http://wfmc.org>. Acesso em: 12 nov. 2002.

WfMC: Published Documents. 2003. Disponível em: <http://www.wfmc.org/standards/docs.htm>. Acesso em: 20 mar. 2003.

WfMC: Sample Workflow. 2003. Disponível em: <http://www.wfmc.org/standards/docs/*XPDL*_sample/>. Acesso em: 19 mar. 2003.

WfMC: *Workflow* Client Application (Interface 2). Applicaton Programming Interface (WAPI) Specification. Document Number WFMC-TC-1009. Version 2.0e (Beta). October/1997. Disponível em <http://www.wfmc.org/standards/docs/interface2-3.pdf> . Acesso em 22 out. 2003.

WfMC: Workflow Application Programer's Interface Naming Conventions. Document Number WFMC-TC-1013. 1997. Disponível em: <http://www.wfmc.org/standards/docs/tc013v14a.pdf> . Acesso em: 22 out. 2003.

ZOPE. Disponível em: <http://www.zope.org>. Acesso em: 23 fev. 2003.

APÊNDICE A UTILIZAÇÃO DE UMA MÁQUINA DE *WORKFLOW* PARA CONTROLAR A EXECUÇÃO DE CURSOS

Um *workflow* pode ser usado para controlar a execução de um curso presencial, um curso de auto-aprendizado ou um curso a distância. Para utilizar uma máquina de *workflow* em uma aplicação de ensino a distância, torna-se necessário levar em conta algumas características especiais, inerentes a esse tipo de utilização. No entanto, a máquina de *workflow* não poderá perder a capacidade de executar um *workflow* genérico. Para isso, uma aplicação cliente específica deve ser desenvolvida para controlar a execução de cursos, acessando as funcionalidades da máquina de *workflow*.

Num *workflow* administrativo o foco principal é a conclusão dos processos. Por exemplo, um banco pode oferecer um financiamento, cujo processo de aprovação pode ser controlado por um *workflow*. Cada funcionário do banco pode estar envolvido com vários processos de financiamento. Se o gerente encarregado de aprovar um processo de financiamento estiver de férias, outro gerente ou substituto poderá fazê-lo. O importante é que o processo de financiamento seja analisado com fins de ser aprovado ou reprovado. Já na execução de um curso a distância, o "quem" está executando uma atividade torna-se tão importante quanto o "que" está sendo feito. Afinal, um mesmo curso é executado por vários alunos. Uma afirmação do tipo "O curso de Inglês foi concluído" é uma informação incompleta. Numa aplicação de execução de um curso, tão importante quanto o fato de que as atividades desse curso tenham sido feitas é o fato de que elas tenham sido feitas por um aluno específico. Quando uma definição de processo corresponde a um curso, o significado semântico de instanciar essa definição é matricular-se no curso. Portanto, a aplicação deve impedir que um aluno crie mais de uma instância de um curso, quando a primeira instância ainda está em execução, uma vez que não faz sentido o aluno estar matriculado duas vezes no mesmo curso, ao mesmo tempo. A aplicação de controle de execução de cursos pode utilizar a máquina de *workflow* da seguinte forma:

- Os alunos são cadastrados como usuários da máquina de *workflow* e colocados em grupos, de acordo com os cursos que lhe são permitidos. Isso corresponde a uma matrícula na escola;
- Os professores também são cadastrados como usuários da máquina de *workflow*. Eles preparam as definições de processo correspondentes aos cursos e carregam-nas na máquina de *workflow*;
- O administrador da escola muda o estado das definições de processo destes cursos entre os estados Ativo ou Inativo. Isso corresponde a abrir ou fechar a matrícula para os cursos;

- Os alunos instanciam as definições de processo que lhes interessam e que lhes estejam disponíveis. Isso corresponde à matrícula no curso.
- Os alunos poderão executar atividades em grupo.

A seguir são abordadas duas questões importantes em uma aplicação de *workflow* para cursos: a execução de atividades individuais e a execução de avaliações.

A.1 Atividades Individuais

A CEMT Workflow pode controlar a execução de um curso, criando uma única definição de processo para cada curso e uma instância de processo para cada aluno matriculado naquele curso. Este aluno torna-se o responsável por aquela instância. Ao contrário do outro exemplo de *workflow* administrativo, um aluno não pode delegar suas atividades do curso a outro aluno, pois não apenas é importante que as atividades do curso sejam feitas, mas também que elas sejam feitas pelo aluno responsável por ela. Como uma mesma definição de processo de *workflow* de curso deverá ser seguida por vários alunos, não é possível atribuir cada aluno como participante nas atividades, na definição de processo. A exemplo do que foi implementado no Reactor, foi criado um papel pré-definido, chamado ProcessCreator, que corresponde ao usuário que cria a instância de processo. Assim, para que todas as atividades de uma definição de processo sejam executadas por um mesmo aluno, é necessário criar um participante do tipo ROLE e nome igual a ProcessCreator e colocá-lo como *performer* de todas as atividades do processo. Em tempo de execução, o aluno irá ver as definições de processo, ou cursos, disponíveis e irá criar uma instância de um processo. O aluno é, então, colocado como ProcessCreator do processo e participante de todas as atividades que têm como *performer* um participante do tipo ROLE e nome igual a ProcessCreator.

A.2 Atividades com data e horário marcados

Embora um curso a distância ofereça uma flexibilidade aos alunos quanto ao horário de executar muitas de suas atividades, ainda restam atividades que exigem uma data e horário específicos para serem executadas. Exemplos são as videoconferências e avalições presenciais.

Assim, pode-se conceber um sistema de ensino e avaliação, onde o ensino seja a distância e a avaliação seja presencial. Isto implica em custos diferentes para cada tipo de atividade. Uma vez desenvolvido o material didático, a sua disponibilização via *web* tem um custo mais baixo comparando-se ao custo da aplicação de uma avaliação, que depende também da alocação de fiscais, sala e/ou máquinas. A forma mais natural de diminuir esse custo é agrupar alunos para fazerem a avaliação simultaneamente e em um mesmo local.

Um projetista de um curso poderá definir se para executar uma atividade de avaliação outras atividades obrigatoriamente devem ser executadas, ou seja, se essas outras atividades são pré-requisitos, como pode ser visto na figura A.1.

Figura A.1: Avaliação executada sempre após o término da lição.

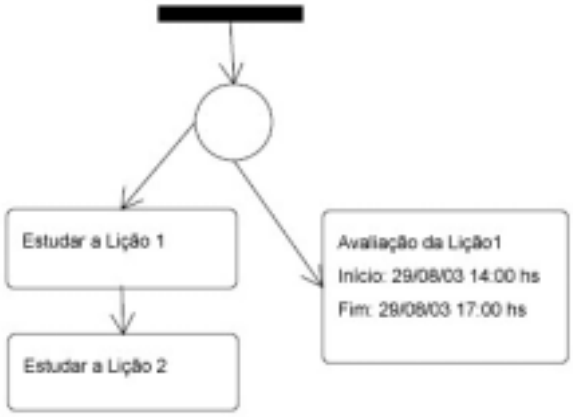


Figura A.2: Avaliação com data marcada.

Dessa forma, cada aluno faria a avaliação da lição 1 a um tempo, já que provavelmente terminariam de estudar a lição 1 em momentos diferentes. Haveria um grande custo relativo à alocação de um fiscal para cada avaliação feita. Outra alternativa seria estabelecer uma data e horário para início e fim da avaliação, como na figura A.2. Nesse caso, o administrador pode querer que todos os alunos façam a avaliação, independente do fato de terem ou não concluído o estudo da lição. Os alunos que terminarem o estudo da lição 1 poderão seguir estudando a lição 2, enquanto aguardam a data da avaliação. O custo para a aplicação da avaliação seria menor, já que poderia ser alocado um fiscal para cada grupo de alunos que façam a avaliação em uma sala. Mas os alunos que estiverem em atraso com o estudo da lição 1 ficariam prejudicados, pois teriam que fazer a avaliação antes de estudarem toda a lição. Uma combinação das duas opções pode ainda ser implementada, obrigando o aluno a cumprir um pré-requisito e estipulando uma data para a avaliação (fig. A.3).

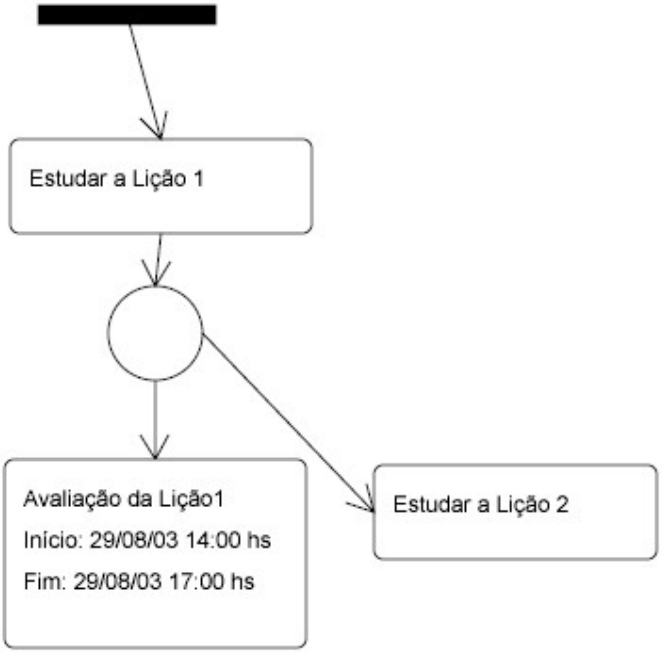


Figura A.3: Avaliação com data marcada e executada após o término da lição.

Após terminarem o estudo da lição 1, os alunos poderão estudar a lição 2, enquanto aguardam a data da avaliação da lição 1. Entretanto, aqueles alunos que atrasarem a conclusão da lição 1 poderão perder a data da avaliação e terem a continuidade do seu curso comprometida. Se a política da escola tende a ser mais rígida, a solução pode estar de acordo. Mas se a escola tem uma postura mais flexível, outra solução precisa ser adotada. Uma variação da opção da figura A.3 é a aplicação de avaliações em uma certa periodicidade, pré-estabelecida ou não. Por exemplo, pode haver uma avaliação da lição 1 toda sexta-feira de manhã. Dessa forma, após concluir o estudo da lição 1, o aluno poderá fazer a avaliação na primeira data disponível. Outros alunos que demorarem mais para finalizar a lição poderão aguardar até a próxima data.

Uma avaliação pode também ser criada sem uma data pré-estabelecida. Neste caso, a atividade de avaliação deverá ficar inicialmente bloqueada. Quando o professor julgar conveniente, ele pode informar a data em que a avaliação deverá ser executada e desbloquear a atividade de avaliação. Toda vez que há uma alteração na data de início da atividade, uma notificação é automaticamente enviada aos participantes de todas as instâncias daquele processo, que ainda não iniciaram a atividade.