UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO – BACHARELADO

SISTEMA DE *WORKFLOW* PARA MODELAGEM E EXECUÇÃO DE PROCESSOS DE SOFTWARE

ROBERTO REINERT

BLUMENAU
2006

2006/1-37

ROBERTO REINERT

SISTEMA DE *WORKFLOW* PARA MODELAGEM E EXECUÇÃO DE PROCESSOS DE SOFTWARE

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Ciências da Computação — Bacharelado.

Prof. Everaldo Artur Grahl - Orientador

BLUMENAU
2006

2006/1-37

SISTEMA DE WORKFLOW PARA MODELAGEM E EXECUÇÃO DE PROCESSOS DE SOFTWARE

Por

ROBERTO REINERT

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

Presidente:	Everaldo Artur Grahl – Orientador, FURB
Membro:	Fabiane Barreto Vavassori Beniti – FURB
Membro:	Wilson Pedro Carli – FURB

Blumenau, 12 de julho de 2006.

Dedico este trabalho a todas as pessoas que direta ou indiretamente me auxiliaram na realização deste.

AGRADECIMENTOS

À Deus, por permitir eu estar presente hoje aqui.

Aos meus pais, por me proporcionarem um bom estudo.

Aos meus amigos, pelo apoio, dicas e sugestões.

Ao meu orientador, Everaldo Artur Grahl, pelo auxílio na realização deste trabalho.

O que lavra a terra com dedicação tem mais mérito religioso do que poderia obter com mil orações sem nada fazer.

Zoroastro

RESUMO

Um dos principais problemas relacionados à qualidade e produtividade em um projeto de software está na falta de utilização de processos de desenvolvimento efetivos às necessidades das organizações. Este trabalho propõe o desenvolvimento de uma ferramenta de *workflow* que auxilie o controle e execução das atividades que envolvem um processo de software. O objetivo principal é promover o fluxo organizado de troca de informações entre os envolvidos no processo, tendo como conseqüência um aumento na produtividade e melhoria na qualidade do processo de desenvolvimento. Como resultado conseguiu-se modelar processos básicos de software e executá-los de acordo com os modelos definidos.

Palavras-chave: Processo de software. *Workflow*.

ABSTRACT

One of the main problems related to quality and productivity in a software project is the lack of utilization of effective developing process to organizations necessities. This issue considers the development a workflow tool that helps the control of activities in a software process. The main goal is to promote an organized flow of information exchange between people that are involved in the process, having as consequence an increase on productivity and a quality of developing process improved. As result obtained was possible to model basic software process and execute them in accordance with the defined model.

Key-words: Software process. Workflow.

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

LISTA DE SIGLAS

IBM – *International Business Machines*

J2EE – *Java 2 Enterprise Edition*

J2ME – *Java 2 Micro Edition*

J2SE – *Java 2 Standard Edition*

JDBC – *Java Database Connectivity*

JSP – *Java Server Pages*

JVM – *Java Virtual Machine*

MA – Modelagem Ágil

MVC – *Model View Controller*

RF – Requisito Funcional

RNF – Requisito Não-Funcional

RUP – *Rational Unified Process*

SQL – *Structured Query Language*

UML – *Unified Modeling Language*

WFMC – *Workflow Management Coalition*

WTP – *Web Tools Platform*

WWW – *World Wide Web*

XML – *Extensible Markup Language*

XP – *Extreme Programming*

SUMÁRIO

1 INTRODUÇÃO

A tecnologia da informação vem desempenhando um papel importante em toda sociedade ao longo dos últimos anos. O software, como meio de disseminação da informação, deixou de ser um diferenciador de mercado para tornar-se a base de sustentação de diversas organizações. No entanto, à medida que ocorre este crescimento na importância da utilização do software, também cresce a necessidade e a dependência pelo desenvolvimento de softwares com maior agilidade e qualidade.

Segundo Ambler (2004, p. 21), a situação atual do desenvolvimento de software está aquém do ideal. Os sistemas são entregues com atraso ou orçamento estourado e por muitas vezes não atendem os requisitos do cliente, necessitando ser desenvolvidos novamente.

Portanto, para que um software tenha a qualidade esperada e alcance seu objetivo, é necessário que haja um processo formal de desenvolvimento que contemple todas as etapas do seu ciclo de vida. Uma das metodologias mais aceitas atualmente para a produção e gerenciamento de software é o *Rational Unified Process* (RUP), que é um processo configurável e customizável que usa a *Unified Modeling Language* (UML) e pode ser adequado tanto a empresas de pequeno porte quanto a empresas de grande porte (RATIONAL SOFTWARE, 2005).

Segundo Scott (2003, p.19), o RUP é um exemplo de versão especializada do processo unificado que adiciona elementos à estrutura genérica de um processo de software. No desenvolvimento de um projeto de um sistema, pode-se “saltar” algumas atividades que não agregam valor. Num processo de software customizado, inicia-se com os elementos básicos do núcleo e adicionam-se elementos na medida em que forem necessários. É isto que faz com que um processo de software possa ser adequado para empresas de portes variados.

Esta característica de se adicionar ou omitir atividades em um processo de software

pode ser obtida através da utilização de *workflows. Workflow* é a automatização de um processo de trabalho, por completo ou uma parte dele, durante o qual documentos, informações ou tarefas são passadas de um participante a outro para serem alvos de ações, de acordo com um conjunto de regras procedurais (WORKFLOW MANAGEMENT COALITION, 2005). Portanto, para possibilitar a customização de um processo de software podem ser utilizados *workflows* que dão suporte as várias etapas do ciclo de desenvolvimento de um software.

Para que estas metodologias e padrões de desenvolvimento de software sejam aplicados de forma efetiva e transparente, é necessário que haja ferramentas que disponibilizem um ambiente integrado para modelagem e definição destes processos. Com este intuito, pretende-se então, construir uma ferramenta que dê o suporte necessário à definição e execução de fluxos de trabalho, que esteja apta a interpretar as definições dos processos de software através de *workflows* e interagir com os envolvidos nas diversas etapas que compõem o processo de desenvolvimento de software.

1.1 OBJETIVOS DO TRABALHO

O objetivo deste trabalho é desenvolver um sistema de *workflow* para modelagem e execução de processos de software.

Os objetivos específicos do trabalho são:

a) possibilitar a criação de modelos de processo de software de forma gráfica;
b) permitir aos envolvidos no processo executarem as atividades definidas no modelo.

1.2 MOTIVAÇÃO

Atualmente, muitas empresas ainda adotam processos manuais e mal documentados. A troca de informações entre os envolvidos no desenvolvimento de um software muitas vezes esbarra na informalidade. Além de tempo, que significa dinheiro, processos informais também pecam quando se fala na qualidade do produto final.

Considerando-se a importância de se ter um processo de desenvolvimento de software muito bem definido e formalizado, propõe-se desenvolver neste trabalho uma ferramenta que permita modelar as várias atividades relacionadas a um processo de software.

Esta modelagem será efetuada de forma visual, o que facilita a visualização e entendimento do encadeamento das atividades que compõe o processo.

1.3 ESTRUTURA DO TRABALHO

No primeiro capítulo é feita uma contextualização do trabalho proposto, apresentando-se a introdução, objetivos e motivação do trabalho.

No capítulo dois é apresentada a fundamentação teórica do trabalho, através dos conceitos que envolvem o tema proposto, dando ênfase aos conceitos de processo de software e *workflow*.

No terceiro capítulo tem-se a especificação e a implementação da ferramenta em si, bem como a listagem dos requisitos do sistema e tecnologias utilizadas.

Por fim, o quarto capítulo apresenta as conclusões obtidas com o trabalho e sugestões de melhorias e extensões para o mesmo.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo aborda os assuntos referentes à revisão bibliográfica do trabalho. Destacam-se o *workflow* e processo de software.

2.1 WORKFLOW

Nos tópicos a seguir são abordados os principais assuntos referentes à tecnologia de *workflows,* seus conceitos e características.

2.1.1 Introdução

As várias etapas necessárias para o desenvolvimento de um software, desde a sua concepção até a sua implantação não são realizadas por uma única pessoa. Fatores como coordenação, cooperação e comunicação são fundamentais para a realização de trabalhos em grupo. Com o intuito de fornecer um suporte a estes fatores surgiram os sistemas de *workflow.* De acordo com Araújo e Borges (2001, p.1), os sistemas de *workflow* têm suas origens a partir de pesquisas em automação de escritórios na década de setenta. O foco principal destas pesquisas estava em oferecer uma solução para roteamento de documentos e formulários nas organizações, diminuindo a manipulação destes em papel.

Este conceito deixou de ser exclusivamente relacionado à manipulação de documentos, quando no início da década de oitenta o objetivo tornou-se unir as ilhas de trabalho e informação, com o auxílio de ferramentas para coordenação de trabalho em equipe.

Na década de noventa, a tecnologia de sistemas de *workflow* evoluiu significativamente, impulsionada pelo crescimento das infra-estruturas de redes e ambientes

para interação entre grupos. Questões relacionadas ao processamento distribuído e interoperabilidade de aplicações trouxeram novos desafios para os sistemas de workflow.

Araújo e Borges (2001, p.2), afirmam ainda que novos paradigmas de interação entre as organizações, baseados no potencial da *World Wide Web* (WWW) estão levando as pesquisas em *workflow* a um novo patamar voltado para a definição de arquiteturas distribuídas de execução de processos.

2.1.2 Definição

Workflow é a automatização de um processo de trabalho, por completo ou uma parte dele, durante o qual documentos, informações ou tarefas são passadas de um participante a outro para serem alvos de ações, de acordo com um conjunto de regras procedurais (WORKFLOW MANAGEMENT COALITION, 2005).

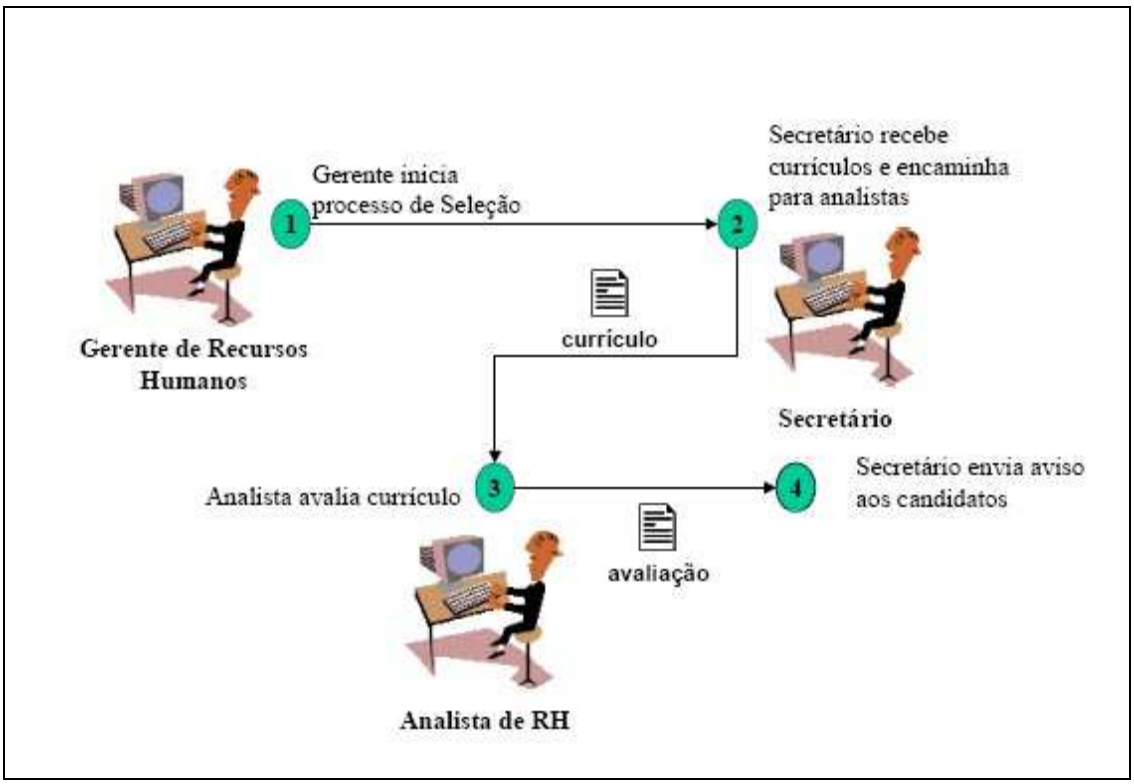
Na maioria das empresas é comum existirem etapas na execução de um processo. Em um determinado processo, cada funcionário executa uma atividade, um conjunto seqüencial de atividades vai sendo manipulado e repassado adiante, até o término do processo.

Um sistema de *workflow* corresponde a um conjunto de ferramentas que permitem o projeto e definição de fluxos de trabalho, sua instanciação e execução controlada dentro de um mesmo fluxo de trabalho (WORKFLOW MANAGEMENT COALITION, 2005).

Pereira e Casanova (2003, p.1), afirmam que considerando as definições e conceitos comuns em toda literatura pesquisada, um sistema de *workflow* pode ser definido como uma coleção de atividades organizadas para realizar um processo. Essas atividades podem ser realizadas por um ou mais sistemas de computador, por agentes humanos ou de software, ou ainda pela combinação destes.

2.1.3 Componentes de um *workflow*

Conforme descrito por Pereira e Casanova (2003, p.5), os componentes fundamentais de um fluxo de trabalho são as atividades. Estas, por sua vez, pressupõe que existam atores que as executam, interpretando seus papéis. Além de atores, devem existir documentos produzidos e consumidos durante a execução das atividades, bem como rotas devem definir os fluxos de destino da informação, conforme figura 1.



Fonte: Araújo e Borges(2001).

Figura 1 – Roteamento de informações e atividades entre membros de uma organização

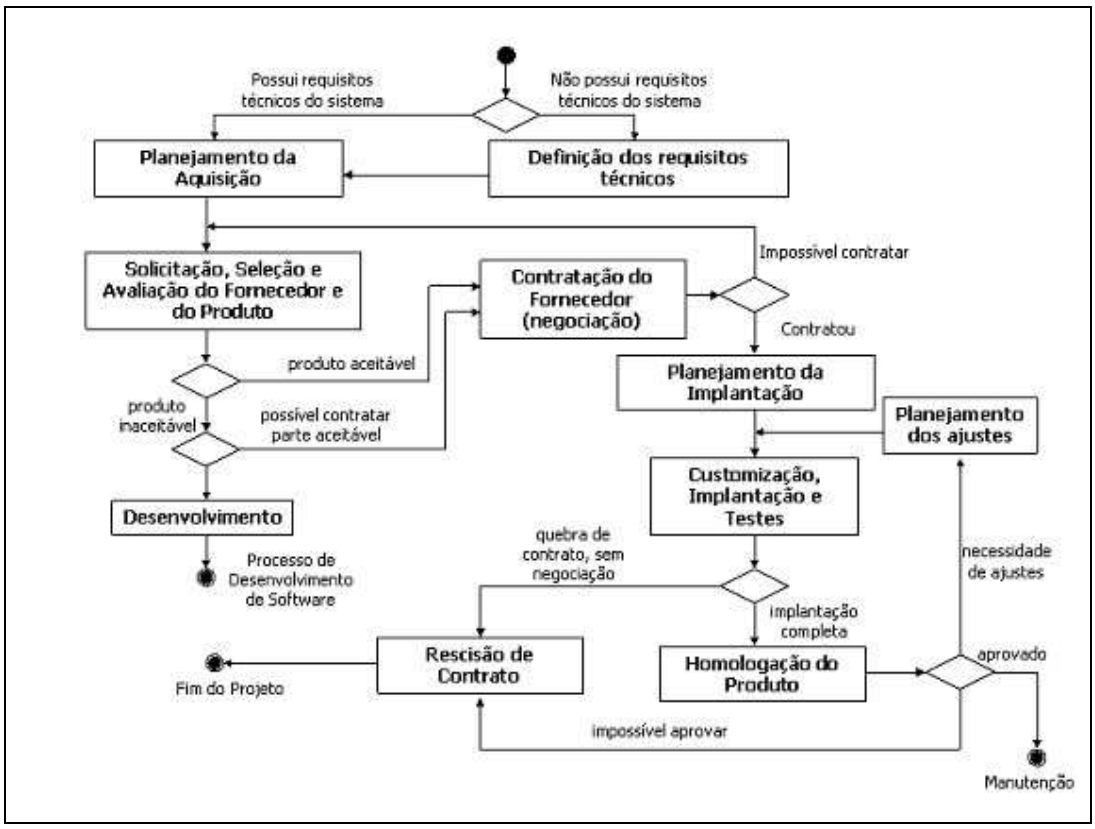
2.1.4 Modelagem de fluxos de trabalho

De acordo com Araújo e Borges (2001, p.5), um modelo de processo deve conter todos os dados necessários sobre os processos a serem executados pelo sistema de *workflow*. Estas informações incluem dados sobre as atividades que compõem os processos, condições de

início e finalização, usuários encarregados e documentos manipulados em cada atividade.

Pereira e Casanova(2003, p.9), afirmam que a definição de processo a ser automatizado em um sistema de *workflow* deve conter informações completas sobre quem tem quais responsabilidades, através de quais operações essas responsabilidades devem ser exercidas e qual a seqüência de execução das mesmas.

Na figura 2 é apresentado um exemplo de notação gráfica de um fluxograma de atividades para um fluxo de trabalho.



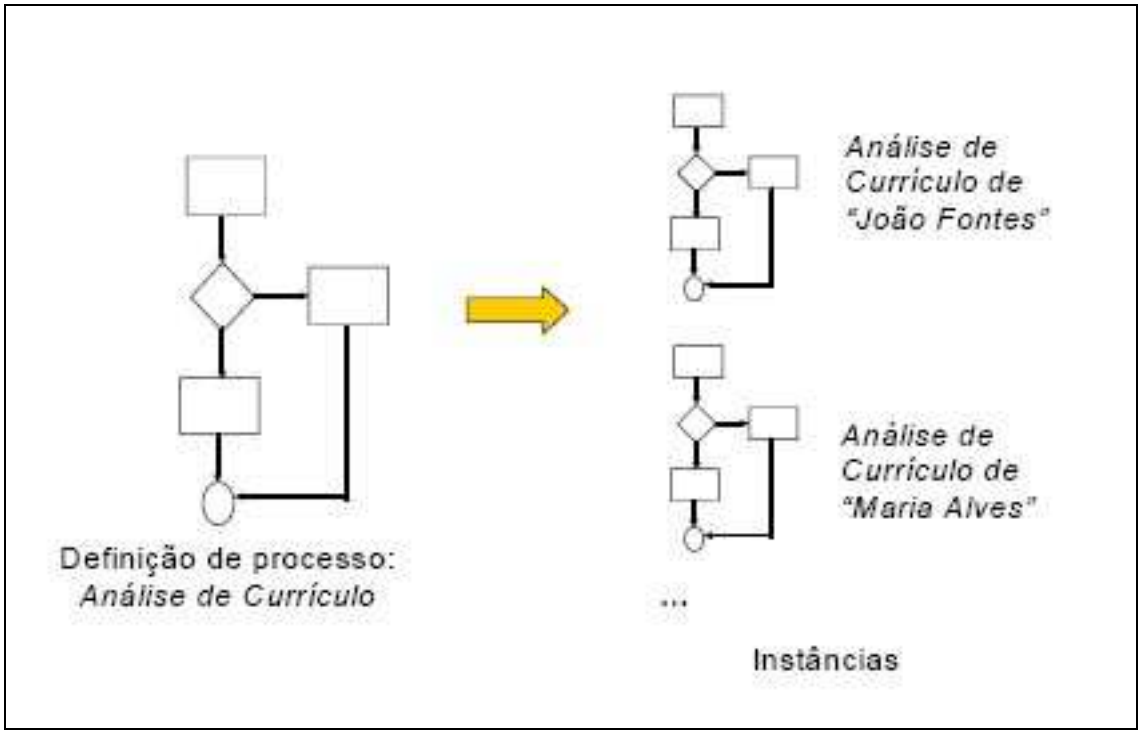
Fonte: Ito (2004).

Figura 2 – Fluxograma de atividades

2.1.5 Execução de fluxos de trabalho

Os fluxos de trabalho são executados através de instâncias de execução das atividades que os compõe. Essas atividades são definidas durante a fase de modelagem do processo e

neste trabalho estão sendo modeladas de acordo com a notação do diagrama de atividades da UML. Na figura 3 é ilustrado um exemplo de instanciação de uma atividade.

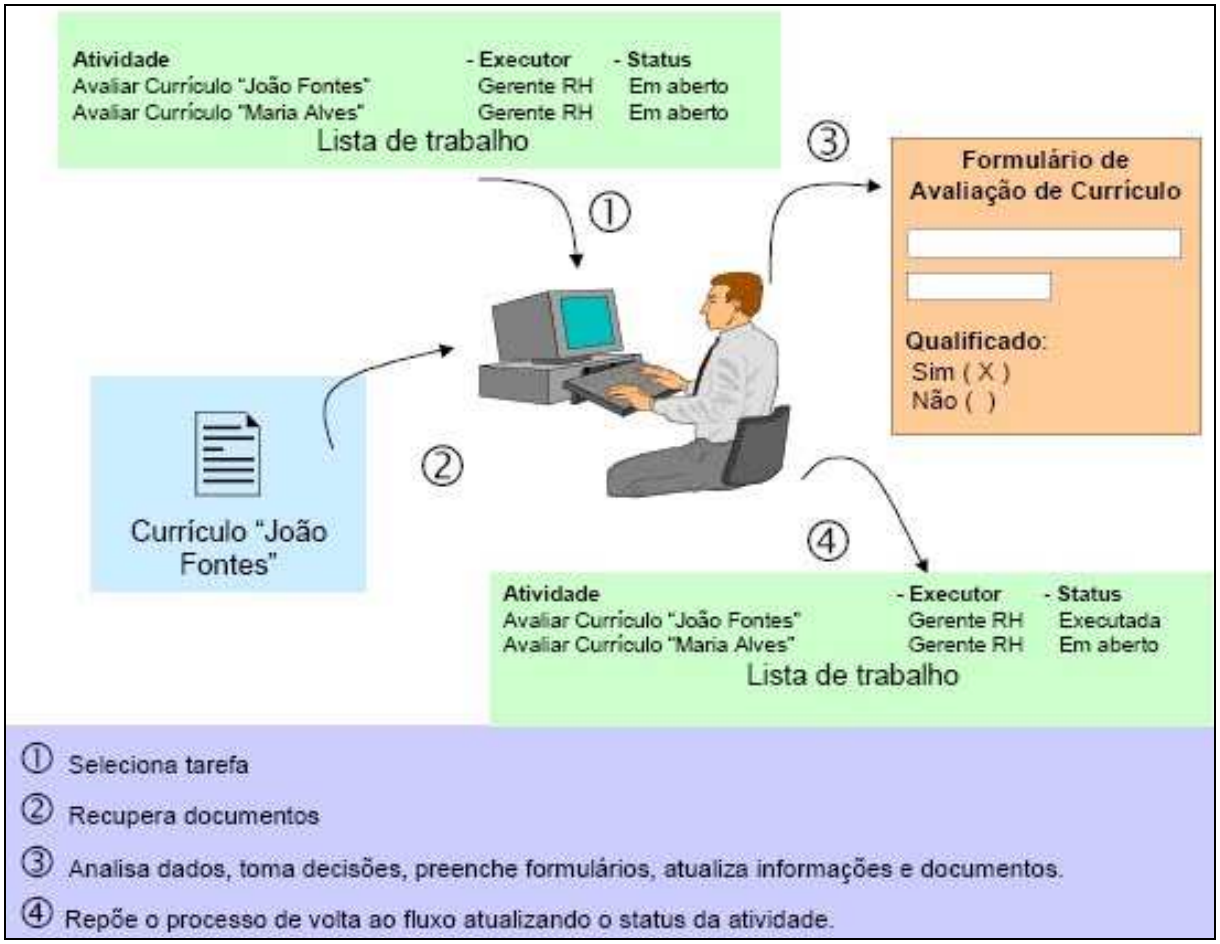


Fonte: Araújo e Borges(2001).

Figura 3 – Instanciação de atividades

2.1.6 Tipos de *workflow*

Os sistemas de *workflow* podem ser classificados de acordo com o seu modelo de execução e agentes relacionados com o mesmo. A seguir são relacionados os tipos de *workflow*: *Ad hoc*, administrativo e de produção. Na figura 4 é apresentada a interação entre usuários e um sistema de *workflow*.



Fonte: Araújo e Borges (2001).

Figura 4 – Interação entre usuários e o sistema de *workflow*

2.1.6.1 *Ad Hoc*

Conforme Oliveira (1996), *workflows ad Hoc* executam processos de negócio, tais como documentação de produtos onde não há um padrão pré-determinado de movimentação de informação entre pessoas. Tarefas do tipo *ad Hoc* envolvem coordenação humana, ou seja, não são automatizadas. Essa classe de *workflow* tipicamente envolve pequenos grupos de profissionais que desenvolvem atividades que requerem uma solução rápida.

2.1.6.2 Administrativo

Um *workflow* administrativo envolve processos repetidos com regras de coordenação de tarefas simples. A ordenação e coordenação das tarefas no *workflow* administrativo podem ser automatizadas. Esta classe de *workflow* não engloba um processamento complexo de informações e não requer acesso a sistemas de informação múltiplos. Devido a estas características, a ferramenta implementada neste trabalho enquadra-se nesta classificação, pois a ordenação das tarefas é automatizada, diferentemente de um *workflow ad hoc* e ao mesmo tempo não possui acesso a múltiplos sistemas externos, como num *workflow* de produção, descrito a seguir.

2.1.6.3 Produção

Um *workflow* de produção envolve processos de negócio repetitivos e previsíveis. Diferentemente dos *workflows* administrativos, os de produção englobam um processamento de informações complexas envolvendo acesso a múltiplos sistemas de informação.

2.2 PROCESSO DE SOFTWARE

Nos tópicos a seguir é apresentada a fundamentação teórica relacionada ao conceito de processos de software, abordando seus conceitos e objetivos.

2.2.1 Introdução

De acordo com Reis (2004, p.3), uma das áreas que mais influenciam o mundo

atualmente, dentre as áreas da ciência da computação é a engenharia de software. Apesar de inúmeros avanços recentes nesta área, muito ainda é discutido acerca da baixa produtividade da indústria mundial de software, o que reflete na insatisfação dos usuários e em prejuízos financeiros de enormes proporções.

Com o intuito de solucionar estes problemas, várias tecnologias vêm sendo experimentadas. Um dos esforços mais significativos corresponde à definição de metodologias voltadas a disciplinar o processo de desenvolvimento de software através do estabelecimento de etapas bem definidas, proporcionando desta forma, um mecanismo para o controle e maturidade destes processos.

2.2.2 Definição

Um processo de software pode ser visto como um manual que dita passo a passo quais os procedimentos que devem ser executados para que se obtenha um determinado resultado. Segundo Scott (2003, p.19), o processo de software é definido como um conjunto de atividades executadas para transformar um conjunto de requisitos do cliente em um sistema de software. De acordo com Wikipédia (2006), um processo de software é um conjunto de atividades ordenadas realizadas com a finalidade de obter um produto de software. São estudados pela Engenharia de Software e é considerado um dos principais mecanismos para obter software de qualidade.

Teles (2004), explica ainda que estas características são fáceis de compreender quando é feita uma analogia da construção de um software à montagem de um veículo. Para montar um automóvel, a indústria recebe um conjunto de matérias-primas que alimenta um processo de fabricação. Elas são transformadas ao longo do processo de modo de que, ao final, se tenha o automóvel desejado.

Assim como em um sistema de *workflow*, o processo de software envolve um conjunto de tarefas ou atividades logicamente ordenadas, que estão relacionados com pessoas, recursos e artefatos.

2.2.3 Objetivos

Conforme apresentado por Reis (2004, p.6), um processo de software tem como principais objetivos:

a) definir ferramentas para descrever os processos e acompanhar a sua execução, controlando a realização de atividades que ocorrem no desenvolvimento de software;
b) facilitar a adoção de uma estratégia de melhoria de maturidade dos processos de uma organização, a partir da prescrição, monitoração e registro dos eventos;
c) permitir o registro do conhecimento produzido acerca de processos bem sucedidos na organização, para ser reutilizado em contextos similares no futuro;
d) coletar métricas dos projetos da organização e torná-las disponíveis para consultas posteriores.

2.2.4 Fundamentos

Conforme Scott (2003, p.32), existem os seguintes elementos principais para um processo de software:

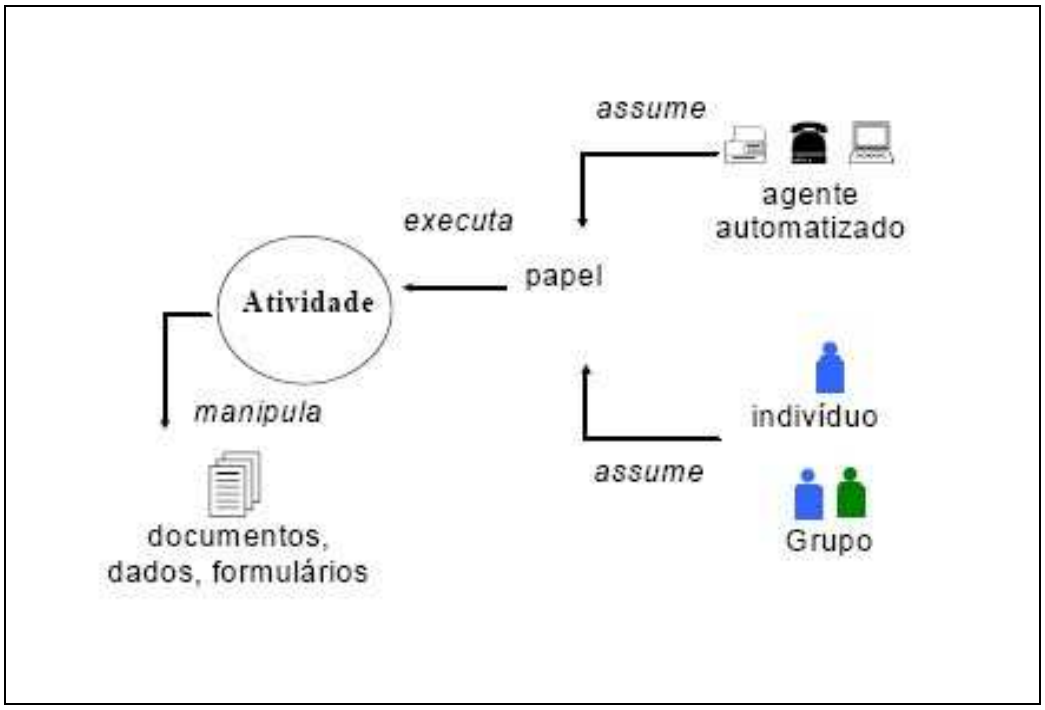
a) artefatos: um artefato é qualquer porção significativa de informação interna ou a ser fornecida a interessados externos que desempenhe um papel no desenvolvimento do sistema. É um produto criado ou modificado durante o

processo. Tal produto é resultado de uma atividade e pode ser utilizado posteriormente como matéria prima para a mesma ou para outra atividade a fim de gerar novos produtos;

b) trabalhadores: em um processo de software um trabalhador é definido como um papel que um indivíduo pode desempenhar no projeto. A diferença entre um trabalhador e um ator, por exemplo, é que o trabalhador tem uma visão mais interna do sistema, ao contrário do ator. Além disso, os atores têm um relacionamento operacional com o sistema, diferentemente do trabalhador que participa do desenvolvimento do sistema;

c) atividades: cada passo executado em processo é uma atividade. Uma atividade é um passo onde existem mudanças de estado visíveis no produto de software. Atividades incorporam procedimentos e regras, tendo como objetivo gerar ou modificar um conjunto de artefatos.

Na figura 5, é apresentado os elementos que compõem um fluxo de trabalho.



Fonte: Araújo e Broges (2001).

Figura 5 – Elementos de um fluxo de trabalho

2.3 TRABALHOS CORRELATOS

Em Bork (2003), é relatado o trabalho de customização e implantação de um processo de desenvolvimento de software baseado no RUP na empresa Dynamix Software. Este trabalho teve como resultado a criação de um protótipo para gerenciamento de projetos, uma especificação prática dentro da metodologia e formulários padrões que permitem o encaminhamento das solicitações de serviços dos clientes. Com este protótipo, foi possível definir o ciclo de vida do projeto, alocar recursos, apontar esforços dos colaboradores e analisar gerencialmente os projetos.

Em Hauck e Wangenheim (2004), é apresentado o trabalho de implantação de um processo de software na empresa Void Caz sistemas. Neste trabalho fica clara a dificuldade de se encontrar um modelo de processo de software pronto, que se enquadre nas necessidades de pequenas empresas. Neste trabalho verifica-se que a empresa inicia a adoção de um processo

de software a partir de metodologias e padrões já existentes, como a norma ISO 15504 e ISO12207 e vai customizando este processo de acordo com as necessidades específicas apontadas pela sua equipe de trabalho.

Já em Bosato, Resende e Bittes (2002), é demonstrado como sistemas de *workflow* podem auxiliar no gerenciamento de projetos de software. Pode-se verificar algumas ferramentas de *workflow* e suas principais deficiências e diferenciais. Outro aspecto interessante abordado refere-se à criação de equipes virtuais de desenvolvimento via *web*. É relatado o problema que existe atualmente em se agrupar bons profissionais em um único lugar. Com sistemas de *workflow* que suportem execução de tarefas via *web*, torna-se possível a criação de grupos de desenvolvimento e até mesmo empresas virtuais.

Estes trabalhos serviram de insumo para o desenvolvimento deste, bem como as documentações geradas serviram de apoio para a criação da ferramenta.

3 DESENVOLVIMENTO DO TRABALHO

Partindo-se dos conceitos e trabalhos estudados no decorrer deste trabalho foi iniciada a fase de levantamento de requisitos do sistema proposto. O sistema deveria possuir as características de um sistema de *workflow*, adicionando-se as características definidas para um processo de software. Como resultado obtido pelas fases de levantamento de requisitos, análise, projeto e construção, é apresentado os itens detalhados pelos tópicos a seguir.

3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

Os requisitos identificados para este sistema foram:

a) permitir o cadastramento de processos, atividades, recursos e artefatos (requisito funcional - RF);
b) possuir um editor gráfico de fluxogramas de atividades do processo (RF);
c) possibilitar o envio de *e-mail* de notificação sobre o *status* das atividades (RF);
d) viabilizar a criação de regras para roteamento das atividades (RF);
e) possuir uma caixa de entrada onde serão listadas as atividades que determinado usuário do sistema possui (RF);
f) possibilitar a criação de modelos que facilitam a reutilização e customização dos processos (RF);
g) permitir a “instanciação” dos modelos, ou seja, possibilitar a criação de instâncias para que possam ser executadas a partir de um modelo de processo existente. Cada modelo poderá gerar várias instâncias (RF);
h) ser implementado na linguagem Java, utilizando ambiente Eclipse (requisito não-funcional - RNF)*;*

i) utilizar arquitetura *Model View Controller* (MVC) (RNF);

j) ser disponibilizado em ambiente *web*, através do servidor Apache Tomcat (RNF);

k) utilizar banco de dados Apache Derby (RNF);

l) utilizar o *framework* de persistência de objetos Hibernate (RNF);

m) utilizar o *framework* JGraph para criação do editor de fluxogramas. Os fluxogramas seguirão a notação de um diagrama de atividades da UML (RNF).

3.2 ESPECIFICAÇÃO

Na figura 6, são apresentados os pacotes com os cenários correspondentes aos casos de uso especificados. Foram criados dois módulos para dividir os casos de uso.

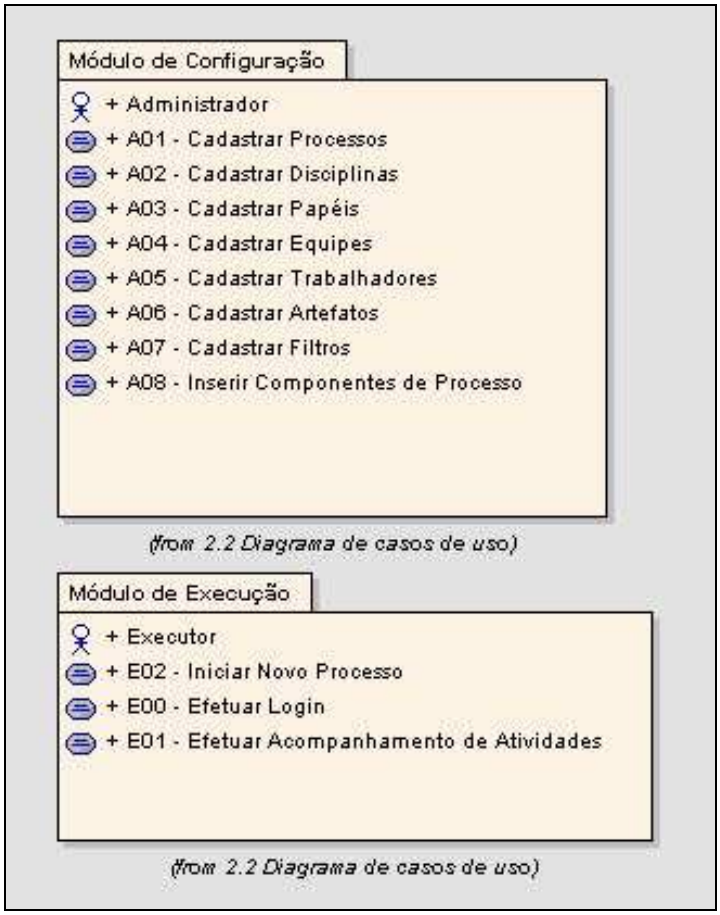


Figura 6 – Pacotes com os casos de uso

3.2.1 Módulo de Configuração

Neste módulo estão concentrados os casos de uso relacionados à configuração dos processos de software. Este módulo é responsável pelo cadastramento de disciplinas, papéis, equipes, trabalhadores, artefatos, filtros e atividades de operação, notificação e pontos de decisão. Este módulo conterá também o editor de fluxogramas, onde será possível interligar os componentes de um processo e ativá-lo para o módulo de execução. Os casos de uso correspondentes a este módulo são ilustrados pela figura 7.

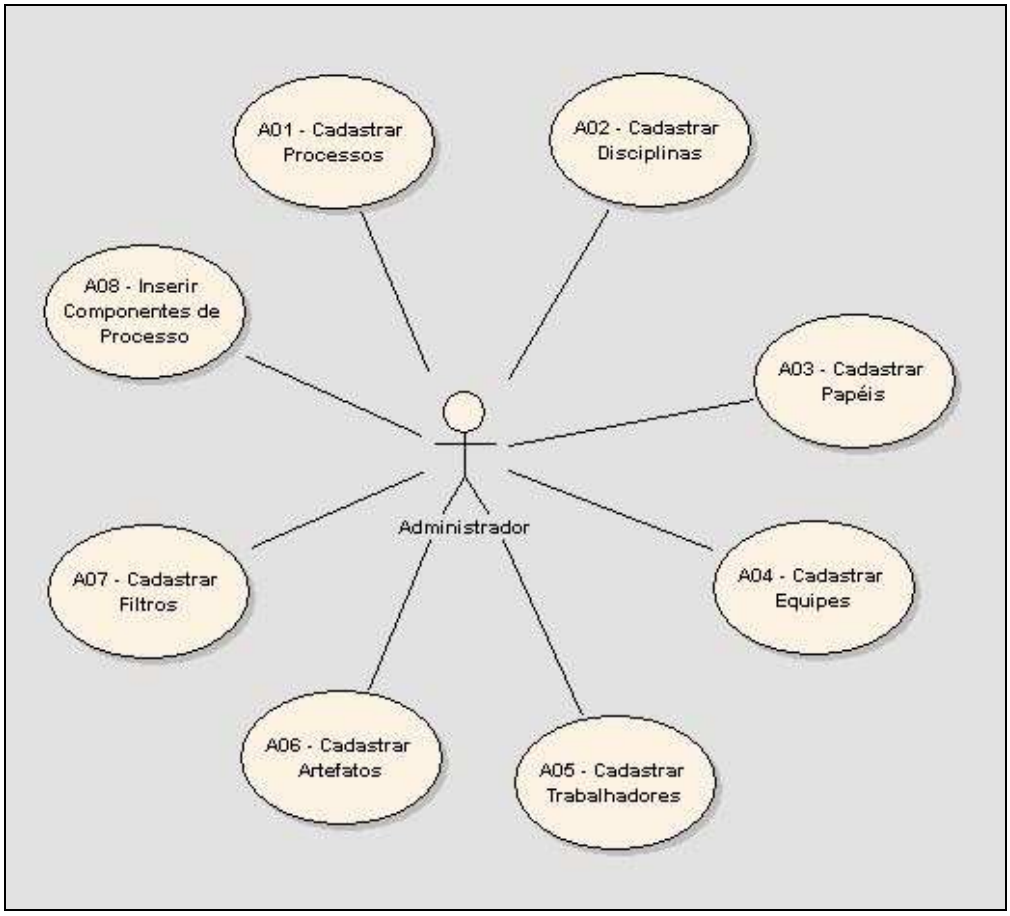


Figura 7 – Casos de uso do módulo de Configuração

3.2.1.1 Cadastros básicos

Os cadastros básicos compreendem a inserção, edição, exclusão e exibição das

entidades no sistema. A definição de cada entidade e o relacionamento entre as mesmas é descrito no capítulo 3.2.3 - Entidades e diagrama de classes de negócio.

3.2.1.2 Inserir componentes de processo

Após ter devidamente cadastrado as entidades básicas no repositório, o administrador poderá criar os modelos que irão conter os componentes. Este caso de uso contempla a criação de novos modelos, abertura de modelos existentes e ativação dos modelos para o módulo de execução.

No cenário principal “Novo Modelo” tem-se:

a) administrador seleciona a opção "Arquivo->Novo";
b) sistema cria novo modelo de processo em branco;
c) o caso de uso é encerrado.

No cenário alternativo “Abrir Modelo” tem-se:

a) administrador seleciona a opção "Arquivo->Abrir";
b) sistema exibe tela de seleção de arquivos, filtrando apenas os arquivos com formato XML;
c) administrador seleciona o arquivo desejado;
d) sistema abre o arquivo correspondente e lista os componentes no editor;
e) o caso de uso é encerrado.

No cenário alternativo “Salvar Modelo” tem-se:

a) administrador seleciona a opção "Arquivo->Salvar";
b) sistema exibe tela de seleção de arquivos;
c) administrador seleciona o arquivo desejado;
d) sistema salva o arquivo correspondente no formato XML;

e) o caso de uso é encerrado.

No cenário alternativo “Ativar Modelo” tem-se:

a) administrador seleciona a opção "Processo->Ativar";

b) sistema verifica a precedência das atividades e persiste em banco;

c) o caso de uso é encerrado.

3.2.1.3 Inserir componentes de processo

Uma vez criado o processo, o administrador poderá selecionar entidades que serão componentes do processo a ser modelado. Haverá no sistema a opção de inserção de atividades do tipo operação, ponto de decisão e notificação. Para cada componente do processo será possível interligar os mesmos através de conectores, setas que definirão a precedência entre as atividades.

O cenário principal para este caso de uso é descrito a seguir:

a) administrador seleciona um componente de processo;

b) sistema insere o componente selecionado no centro do editor;

c) o caso de uso é encerrado.

3.2.2 Módulo de Execução

Neste módulo estão concentrados os casos de uso relacionados à execução do processo em si. Este módulo é responsável pela listagem das atividades pendentes de cada usuário do sistema, pela instanciação de novas execuções de processo, pela edição dos atributos artefatos e pelo “roteamento” das atividades entre os participantes, conforme figura 8.

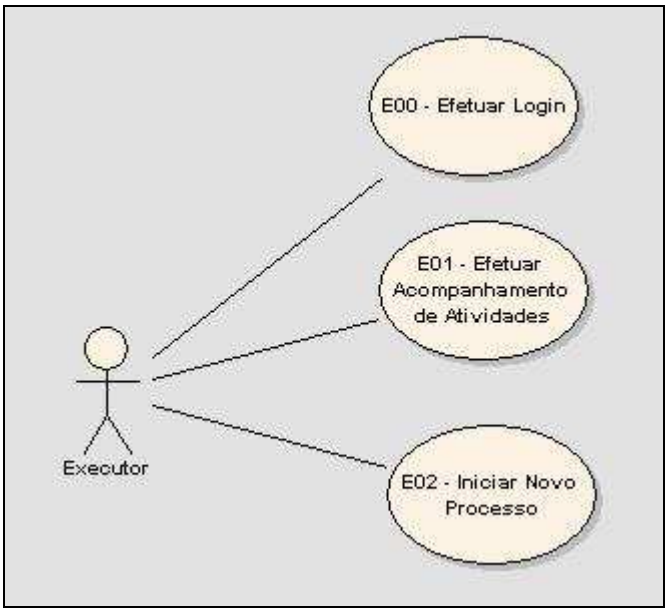


Figura 8 – Casos de uso do módulo executor

3.2.2.1 Efetuar login

Este caso de uso permite a autenticação do usuário no módulo executor. O cenário principal para este caso de uso é descrito a seguir:

a) executor informa login e senha;

b) sistema verifica a existência do mesmo na base e autentica;

c) o caso de uso é encerrado.

3.2.2.2 Efetuar acompanhamento de atividades

Este caso de uso compreende a exibição das atividades que se encontram sob a responsabilidade do usuário autenticado no módulo executor. Estas atividades serão listadas no formato de uma “caixa de entrada”, onde o usuário poderá selecionar uma atividade específica e escolher entre as opções de encaminhamento ou edição dos artefatos. Também neste caso de uso estará disponibilizada opção de geração de relatórios de atividades, onde

será possível efetuar a listagem de todas as atividades executadas no sistema, de acordo com os parâmetros informados pelo usuário.

O cenário principal para este caso de uso é descrito a seguir:

a) após efetuado o login, o sistema exibe as atividades que estão pendentes para o usuário logado;
b) usuário seleciona atividade e seleciona a opção editar artefatos;
c) sistema exibe os artefatos atrelados à atividade;
d) usuário seleciona artefato que deseja efetuar alterações;
e) sistema exibe artefato;
f) usuário preenche os atributos do artefato e salva;
g) o caso de uso é encerrado.

No cenário alternativo “Encaminhar atividade”, caso no passo “b” do cenário principal o usuário selecione a opção encaminhar atividade:

a) sistema encaminha a atividade;
b) o caso de uso é encerrado.

No cenário alternativo “Gerar relatório de atividades”, caso no passo “b” do cenário principal o usuário selecione a opção gerar relatório de atividades:

a) sistema exibe a tela de parâmetros de relatórios;
b) usuário preenche os parâmetros;
c) sistema exibe a listagem das atividades conforme parâmetros informados;
d) o caso de uso é encerrado.

3.2.2.3 Iniciar novo processo

Este caso de uso permite a instanciação de um modelo de processo que foi ativado pelo

módulo configurador.

O cenário principal para este caso de uso é descrito a seguir:

a) usuário seleciona a opção iniciar novo processo;
b) sistema exibe uma lista com todos os processos ativados pelo módulo configurador;
c) usuário seleciona o processo desejado e seleciona a opção continuar;
d) sistema instancia o modelo de processo selecionado, executando a primeira atividade do fluxo de precedência;
e) o caso de uso é encerrado.

3.2.3 Entidades e diagrama de classes de negócio

Como resultado obtido no levantamento dos requisitos, análise e projeto deste trabalho chegou-se às seguintes entidades:

a) disciplinas: as disciplinas compõe um conjunto de atividades com um mesmo objetivo em determinado momento da fase de desenvolvimento de um software.
b) trabalhadores: os trabalhadores serão os usuários propriamente ditos do sistema. São as pessoas que estarão envolvidas no processo, gerando artefatos ao longo das atividades;
c) equipes: conjunto de pessoas, trabalhadores em um determinado grupo ou empresa;
d) papéis: dentro de uma equipe, um determinado trabalhador, pode assumir um papel, como por exemplo ser um testador, programador ou analista;
e) artefatos: os artefatos são os produtos que serão gerados com o decorrer das atividades. Cada artefato irá possuir um conjunto de atributos que serão disponibilizados para inserção de dados pelos trabalhadores. Estes dados inseridos

servirão de insumo para os filtros, descritos a seguir;

f) filtros: conjunto de verificações efetuadas sobre os atributos de um artefato. A partir dos filtros pode-se definir, por exemplo, se um artefato possui itens a serem revistos. Os filtros servirão de insumo para os pontos de decisão;

g) atividades de operação: são as atividades onde o usuário irá efetuar alguma alteração em um artefato;

h) pontos de decisão: onde o sistema irá avaliar os filtros relacionados e definir para qual atividade o fluxo deve prosseguir;

i) atividades de notificação: onde o sistema irá enviar uma notificação por e-mail ao trabalhador selecionado.

Na figura 9 é apresentado o diagrama de classes de negócio da ferramenta desenvolvida neste trabalho, com seus atributos, operações e relacionamentos.

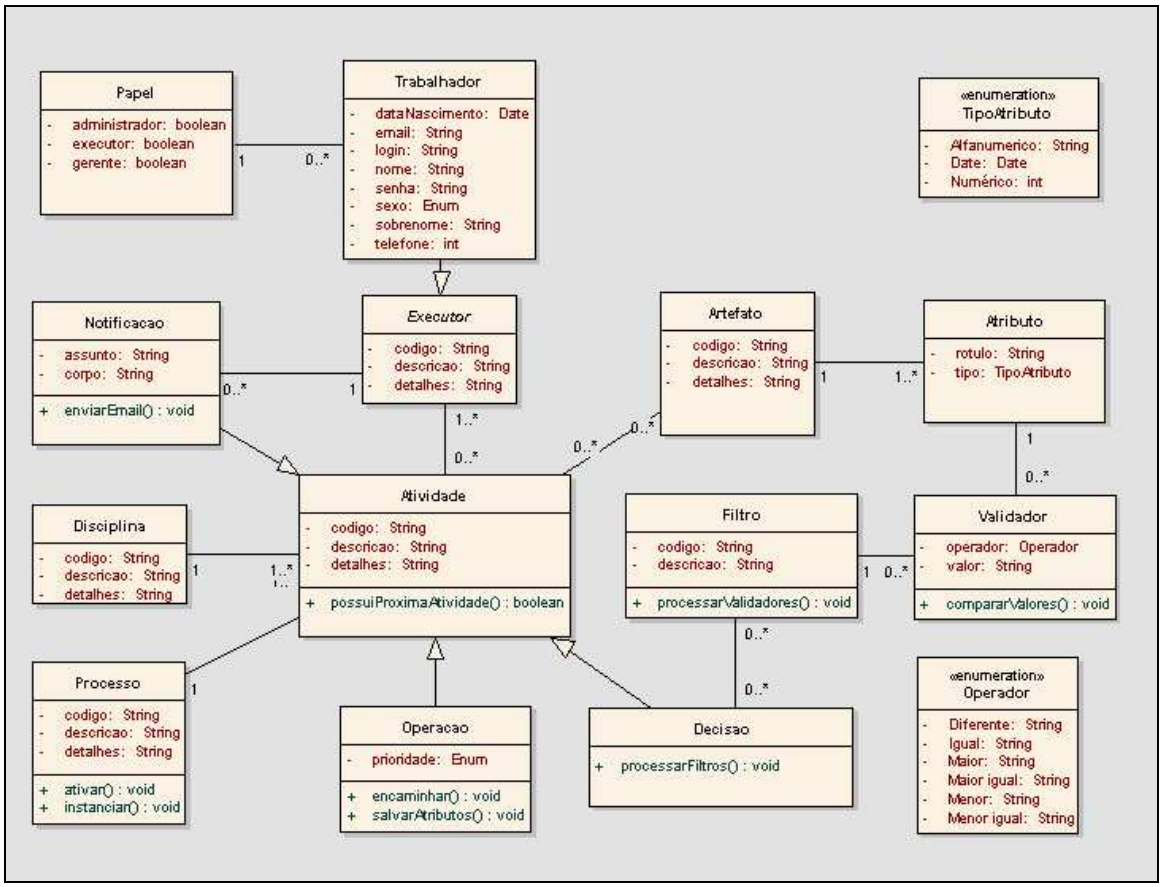


Figura 9 – Diagrama de classes da aplicação.

Os principais atributos contidos neste diagrama são:

a) código, descrição e detalhes :estes atributos estão presentes na maioria das classes e servem para identificar e detalhar informações genéricas da entidade que está sendo manipulada;
b) nome, sobrenome, e-mail, entre outros: atributos da classe Trabalhador que descrevem informações relevantes dos usuários do sistema;
c) administrador, executor e gerente: atributos booleanos da classe Papel que servem para indicar o tipo de papel cadastrado;
d) rotulo e tipo: atributos da classe “Atributo” que servem para informar qual rótulo deve ser exibido no módulo de execução para cada atributo de um artefato, bem como qual tipo de dado estará contido no mesmo;
e) prioridade: atributo que descreve qual a prioridade que a tarefa irá possuir no módulo de execução;
f) assunto, corpo e destinatário: atributos da classe “Notificação” que serão utilizados para o envio de *e-mail*.

As principais operações contidas neste diagrama são:

a) ativar() operação responsável pela ativação e disponibilização de um processo recém criado pelo módulo de configuração para o módulo de execução;
b) instanciar() esta operação tem a função de criar uma nova instância de um processo no módulo de execução;
c) possuiProximaAtividade() operação responsável pela verificação da existência de uma próxima atividade no processo;
d) enviarEmail() operação que tem por função possibilitar o envio de *e-mails* pelo sistema;
e) encaminhar() esta operação é responsável pelo encaminhamento da atividade para

o próximo executor no sistema;

f) salvarAtributos() esta operação pertence à classe Operação e é responsável pela persistência das informações inseridas no decorrer da execução das atividades no processo;

g) processarFiltros() operação responsável pela processamento dos filtros, ou seja, a verificação dos retornos de cada filtro em uma atividade do tipo Decisão, com o intuito de verificar a próxima atividade a ser executada;

h) processarValidadores() esta operação compreende o processamento do retorno dos validadores de um determinado filtro. Esta operação utiliza-se da operação compararValores();

i) compararValores() esta operação tem a finalidade de verificar o conteúdo de cada atributo de um artefato e compará-lo com o valor especificado no validador.

3.3 IMPLEMENTAÇÃO

Seguem nos próximos tópicos as principais ferramentas utilizadas nas fases de desenvolvimento do projeto, bem como as tecnologias, arquiteturas e *frameworks* utilizados. Também são abordados a operacionalidade e os resultados obtidos.

3.3.1 Técnicas utilizadas

Nos tópicos a seguir são apresentadas as principais técnicas utilizadas no desenvolvimento deste trabalho, destacam-se a plataforma Java e a arquitetura *Model View Controller* (MVC).

3.3.1.1 Arquitetura MVC

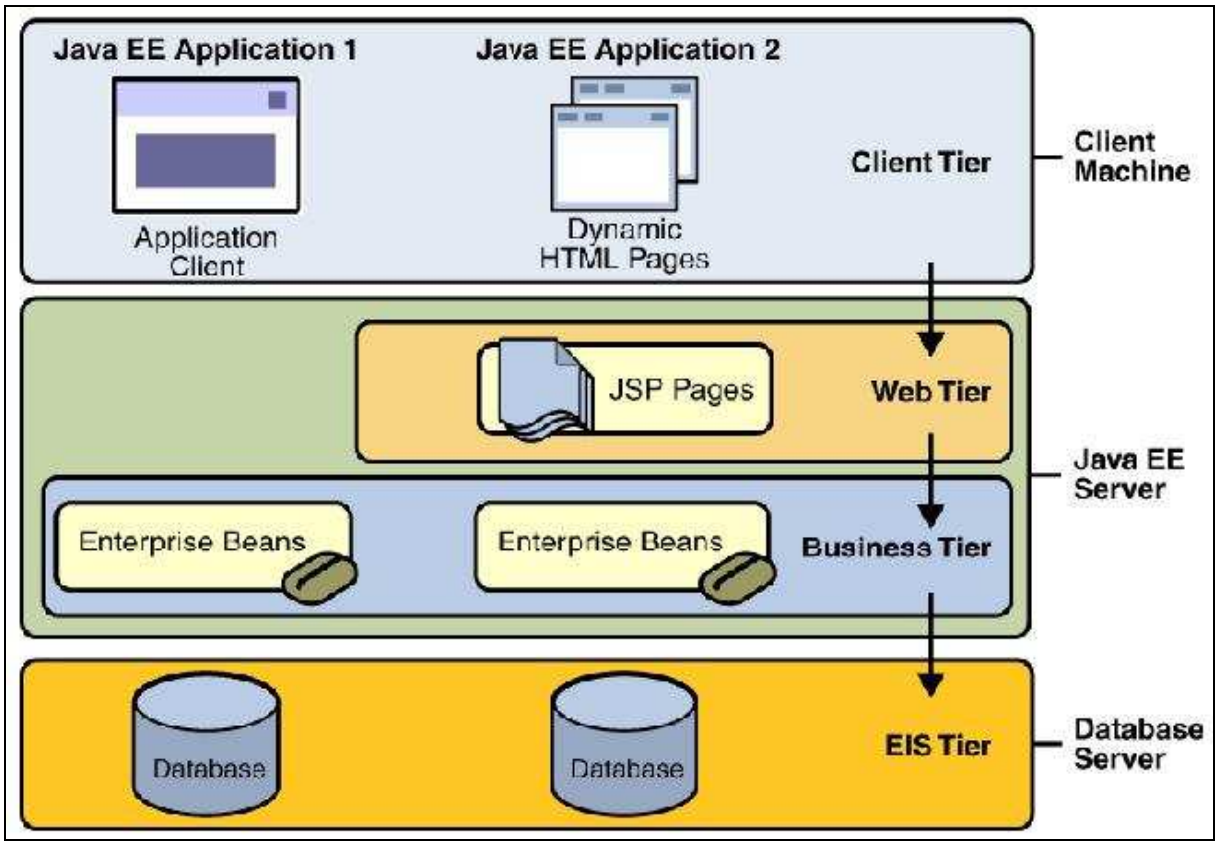
Segundo Kurniawan (2002), MVC é uma arquitetura que separa geração de conteúdo de apresentação de conteúdo. Os aplicativos que utilizam esta arquitetura são mais flexíveis e mais fáceis de manter e de estender, devido à presença de um controlador que controla o fluxo da aplicação.

Os componentes de uma arquitetura MVC dividem em:

a) *model:* camada que contém a lógica de negócios, os objetos que serão manipulados;
b) view: camada que possui a interface com o usuário, responsável pela exibição do conteúdo ao usuário;
c) controller: camada que controla o fluxo da aplicação, fica entre a *model* e a *view*.

Na figura 10 é apresentada a estrutura de camadas aplicada neste trabalho. Na camada do cliente (*Client Machine*) está a aplicação *Java EE Application 1*, que se aplica ao módulo configurador (aplicação cliente), e a aplicação *Java EE Application 2*, que se aplica ao módulo executor (aplicação *web*) da ferramenta implementada.

Na camada Java EE Server temos os componentes que estarão no servidor. Fazem parte desta camada as páginas *Java Server Pages* (JSP) e classes de negócio representados pelos quadros 2 e 3 respectivamente.



Fonte: adaptado de Sun Microsystems. (2006).

Figura 10 – Arquitetura MVC

Nos quadros 1, 2 e 3 é apresentado trechos do código fonte da implementação correspondentes às camadas persistência, negócio e página JSP da camada de apresentação respectivamente.

```
package persistencia.atividade;

import java.lang.reflect.ParameterizedType;

public class NotificacaoDAO<E> extends BaseDAO<Notificacao> {
    private static NotificacaoDAO instance;

    public static NotificacaoDAO getInstance() {
        if (instance == null) instance = new NotificacaoDAO();
        return instance;
    }

    protected Notificacao buscar(Criterion... criterion) {
        Criteria crit = makeCriteria(criterion);
        Notificacao bean = (Notificacao) crit.uniqueResult();
        return bean;
    }

    public E salvar(E bean) {
        Session s = getSession();
        try {
            s.saveOrUpdate(bean);
        } catch (HibernateException e) {
            rollback();
            throw e;
        }
        return bean;
    }
```

Quadro 1 – Código fonte de uma classe de persistência.

```
package negocio.atividade;

import negocio.executor.Executor;

/**
 * @hibernate.joined-subclass table="Notificacao" lazy="false"
 * @hibernate.joined-subclass-key column="ATIVIDADE_ID"
 */
public class Notificacao extends Atividade {

    private String assunto;

    private String corpo;

    private Executor destinatario;

    /** @hibernate.property not-null="false" */
    public String getAssunto() {
        return this.assunto;
    }

    /** @hibernate.property not-null="false" */
    public String getCorpo() {
        return this.corpo;
    }

    /**
     * @hibernate.many-to-one not-null="false"
     */
    public Executor getDestinatario() {
        return this.destinatario;
    }
```

Quadro 2 – Código fonte de uma classe de negócio.

```
<table width="30%" border=0 align="center" cellSpacing=0
    borderColor=#c0c0c0 borderColorDark=#ffffff bgColor=#ffffff>
    <tr>
        <td class="textocorpo" title="Identificação do usuário no sistema"
            style="CURSOR: help"><b>Usuário:<b></td>
        <td><mtw:input name="usuario" size="25" /> <mtw:hasError>
            <font color="red"><mtw:error field="usuario" /></font>
        </mtw:hasError></td>
    </tr>
    <tr>
        <td class="textocorpo" title="Senha do usuário"
            style="CURSOR: help"><b>Senha:</b></td>
        <td><mtw:input type="password" name="senha" size="15" />
            <mtw:hasError>
            <font color="red"><mtw:error field="senha" /></font>
        </mtw:hasError></td>
    </tr>
    <tr>
        <td align=center colspan=2 bgcolor="#FFFFFF"><input type="submit"
            value="OK" class="botao"> <input type="button"
            style="width:'20%'" value="Limpar"
            onclick="javascript:limpar();"
            class="botao"></td>
    </tr>
</table>


<table width="100%" height="14" cellpadding="0" cellspacing="0"
    class="textorodape">
```

Quadro 3 – Código fonte de um JSP de apresentação.

A utilização deste tipo de arquitetura foi utilizada neste projeto pelo fato de permitir uma maior clareza e consistência na divisão das camadas que compõe o projeto. Com a arquitetura MVC o programador pode concentra-se unicamente na camada que está desenvolvendo, seja ela a camada de apresentação negócio ou persistência.

3.3.1.2 Plataforma Java

A linguagem Java, desenvolvida pela Sun Microsystems, consiste de um conjunto de edições de desenvolvimento que atendem as mais diversas necessidades do mercado. De acordo com Sun Microsystems (2005), Java 2 é a nova versão da plataforma Java para desenvolvimento. Estas edições dividem-se em:

a) Java 2 *Standard Edition* (J2SE), destinada a aplicações para *desktops*;

b) Java 2 *Enterprise Edition* (J2EE), destinada a aplicações multicamadas;

c) Java 2 *Micro Edition* (J2ME), destinada a aplicações móveis.

A razão da popularidade da plataforma Java, dentre várias outras vantagens, reside no fato de que as aplicações geradas nesta linguagem podem ser executadas em diferentes sistemas operacionais. Isto se torna possível, devido ao fato de a plataforma Java fornecer uma máquina virtual chamada *Java Virtual Machine (JVM).* Cada sistema operacional deve possuir uma JVM específica, que fará a tradução do *Byte Code* para linguagem de máquina, compreensível a cada sistema operacional.

O motivo de este trabalho ter sido implementado na linguagem Java, além das razões citadas anteriormente, reside no fato de a mesma ser de amplo domínio da comunidade de desenvolvimento de software mundial e pela legibilidade e facilidade de implementação que a mesma proporciona.

Neste trabalho foi utilizada a edição J2EE do Java, que oferece suporte a aplicações *web* multicamadas, escalonáveis e robustas.

3.3.1.3 *Framework* JGraph

Para atender ao requisito de “possuir um editor gráfico de fluxogramas de atividades do processo” foi desenvolvido um editor utilizando-se como base um *framework* com funcionalidades de edição gráfica. Estas funcionalidades facilitam o desenvolvimento da nova ferramenta.

Conforme JGRAPH LTD (2006), entre as principais características do JGraph destacam-se:

a) biblioteca de classes e métodos altamente documentada;

b) utiliza 100% linguagem Java;

c) baseado no padrão MVC;

d) possui licença livre.

O JGraph consiste de um conjunto de classes Java que permitem ao programador a criação de objetos gráficos bidimensionais. Para cada objeto gráfico criado na tela existem alguns atributos que são tratados pelo próprio JGraph como posição X e Y na tela, tamanho do objeto, cor e imagem além de referências à outros objetos que estejam ligados à ele.

Para que estes atributos não se percam a cada execução é necessário que os mesmos sejam persistidos de alguma forma. Para prover esta funcionalidade o JGraph permite que os mesmos sejam salvos no formato XML (Extensible Markup Language). Dessa forma é possível recuperar o modelo com os objetos gráficos salvos apenas informando o nome do arquivo XML que contém os atributos dos objetos que se pretende trabalhar.

3.3.1.4 *Framework web* Mentawai

Para o desenvolvimento do módulo executor deste trabalho atender ao requisito de "ser disponibilizado em ambiente *web*" foi utilizado o *framework* Mentawai. Conforme Oliveira (2006), o Mentawai tem por objetivo ser um *framework web* simples, flexível, prazeroso e produtivo. Possui uma arquitetura voltada para o paradigma de ações ou "a*ctions*". Cada ação é mapeada em uma classe Java, ao contrário de outros *frameworks web,* que se utiliza de arquivos XML para definição sobre qual página *web* deve ser exibida em cada ação do usuário.

O *framework* Mentawai possibilitou a utilização da arquitetura MVC neste trabalho, atuando na camada "*Controller*", interligando as páginas JSP da camada de apresentação (*View*) e negócio (*Model*), através de suas *actions*, citadas anteriormente.

3.3.1.5 *Framework* de persistência Hibernate e geração de código com XDoclet

Um dos maiores esforços no desenvolvimento de aplicações orientadas a objetos encontra-se na camada de persistência. O “mapeamento” das classes feitas em Java para tabelas no banco de dados pode ser uma tarefa bastante complicada e lenta. Com o intuito de facilitar este trabalho, foram utilizados o *framework* de persistência Hibernate em conjunto com o gerador de código XDoclet.

O Hibernate é uma solução de alta performance para persistência de objetos relacionais. Com o Hibernate é possível recuperar objetos do banco de dados de forma transparente e eficiente, pois as classes Java são mapeadas para arquivos XML correspondentes aos atributos das tabelas do banco de dados (JBOSS INC, 2006).

Para facilitar ainda mais o trabalho, é possível criar *doclets*, comentários nas classes Java que em conjunto com o Hibernate já geram todas as tabelas do banco de dados sem que o desenvolvedor execute nenhum comando *Structured Query Language* (SQL).

3.3.1.6 Banco de dados Apache Derby

Em relação à persistência dos dados da ferramenta produzida neste trabalho, no que refere-se à banco de dados, foi utilizado o banco Apache Derby.

O Apache Derby, um subprojeto do projeto Apache, é um banco de dados relacional implementado totalmente em Java (APACHE FOUNDATION, 2006). Destacam-se as seguintes características:

a) é baseado nos padrões Java, JDBC (*Java Database Connectivity*) e SQL;
b) suporta os modos de utilização “embacardo” (dentro da aplicação) e cliente/servidor;

c) é de fácil instalação, distribuição e uso;

d) possui licença livre.

3.3.2 Ferramentas utilizadas

A seguir encontram-se as principais ferramentas utilizadas no desenvolvimento deste trabalho, desde a fase de análise, passando pelo desenvolvimento e testes.

3.3.2.1 Enterprise Architect

Nas fases de requisitos, análise e projeto foi utilizada a ferramenta Enterprise Architect para definição dos casos de uso, especificação de requisitos e regras de negócio, bem como a especificação dos fluxos e criação de diagramas de classes e atividades.

3.3.2.2 Eclipse e JUnit

Para as fases de implementação e testes foram utilizados o editor Eclipse 3.2 WTP (*Web Tool Platform*) em conjunto com um "*plug-in*" para o eclipse, o JUnit, que viabiliza a criação de testes unitários no código fonte implementado.

Com a utilização da versão WTP do eclipse é possível executar a aplicação diretamente pelo editor, sem ter a necessidade de se executar nenhum aplicativo externo ao mesmo, como por exemplo, um navegador *web*.

3.4 OPERACIONALIDADE DA IMPLEMENTAÇÃO

Nos tópicos a seguir, é apresentada a operacionalidade da ferramenta desenvolvida neste trabalho, contemplando desde a configuração do processo até sua execução. Na figura 11 é ilustrado o diagrama de atividades envolvendo passos executados pelos usuários e o sistema durante a modelagem e execução de um processo na ferramenta.

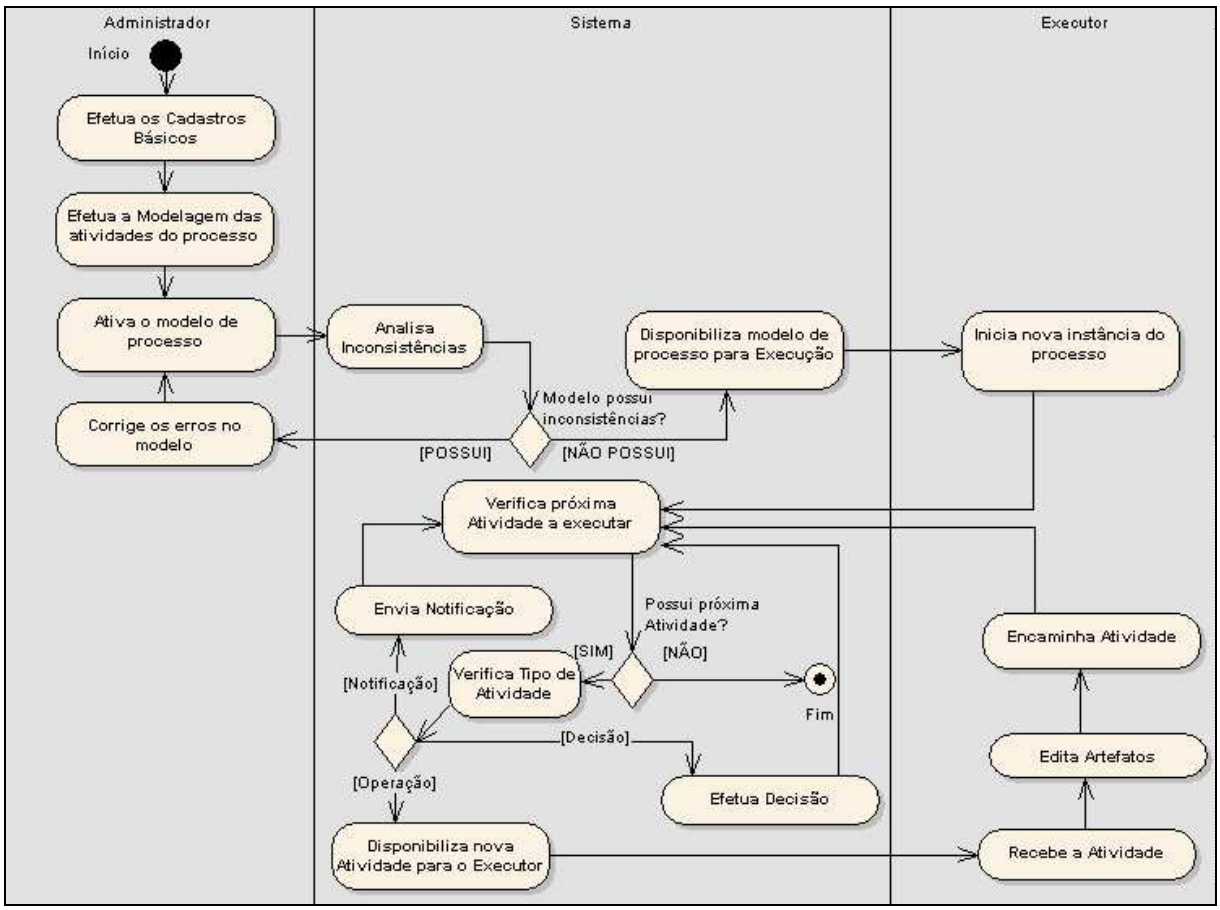


Figura 11 – Diagrama de atividades envolvendo os usuários e o sistema

3.4.1 O processo exemplo

Para ilustrar a operacionalidade da implementação desenvolvida, foi criado um "processo exemplo", um modelo reduzido que consiste de quatro macro atividades básicas: "elicitar requisitos", "analisar e projetar sistema", "implementar código fonte" e "testar

sistema”. Para cada uma destas atividades foi definido um trabalhador responsável pela execução da mesma, bem como os artefatos que serão manipulados em cada atividade. No decorrer de algumas atividades deste processo, notificações são enviadas aos trabalhadores envolvidos. Na figura 12 é apresentada a modelagem do processo.

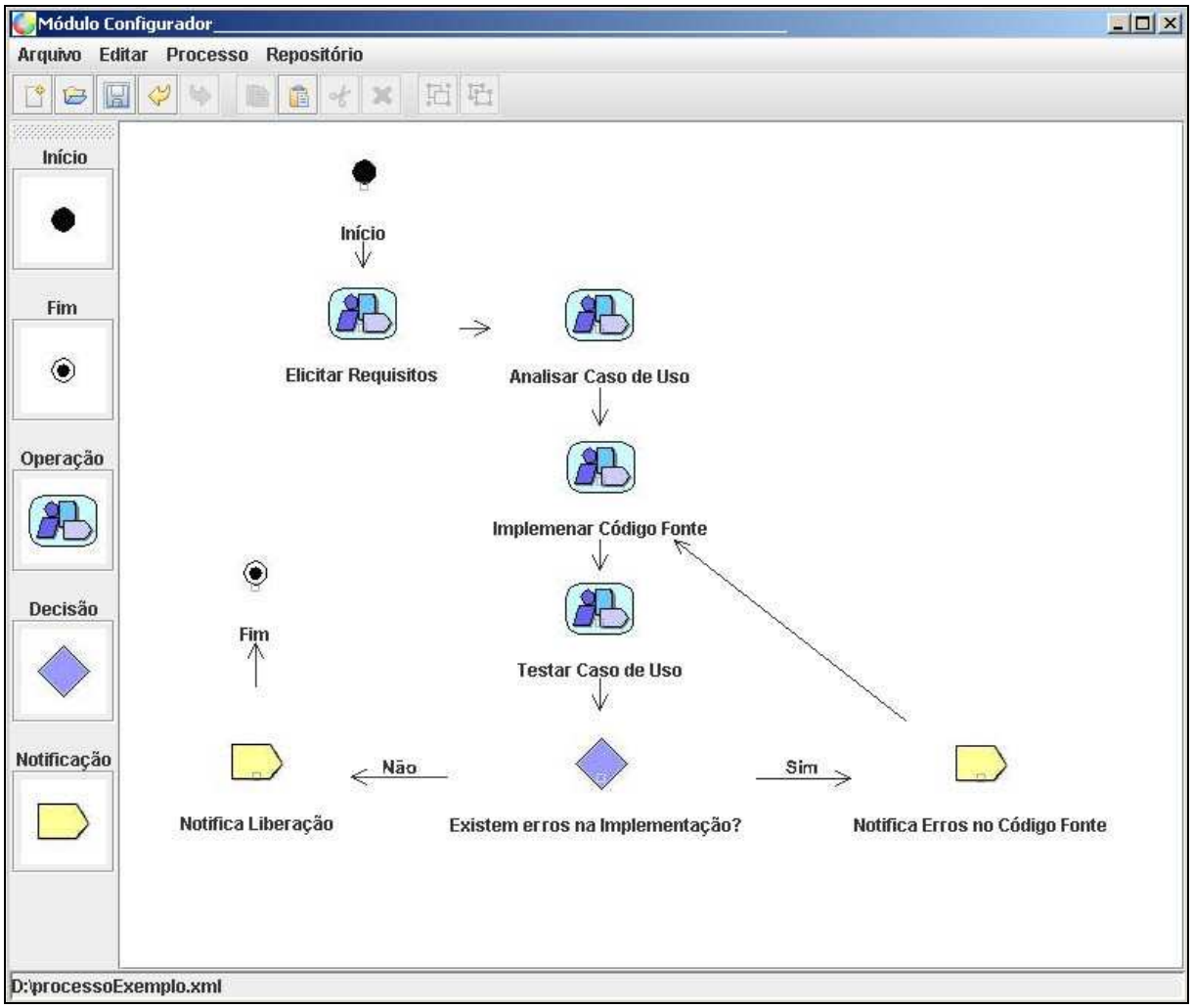


Figura 12 – Modelagem gráfica

3.4.1.1 Cadastros básicos

Como primeiro passo a ser seguido para a configuração de um processo de software na ferramenta, o usuário necessita efetuar o cadastramento de alguns dados básicos no sistema.

Para possibilitar a modelagem do processo na ferramenta é necessário que sejam cadastradas as informações nas telas correspondentes aos itens de menu: “Artefatos”, “Notificações”, “Operações”, “Filtros”, “Pontos de decisão”, “Processos” e “Trabalhadores”. Todas as telas de cadastro possuem uma barra inferior com alguns botões padronizados, onde é possível realizar a manutenção e navegação dos registros. Na figura 13 é apresentado o menu com os cadastros básicos que podem ser realizados no repositório, que servirá para reutilização posterior no momento da modelagem dos processos.

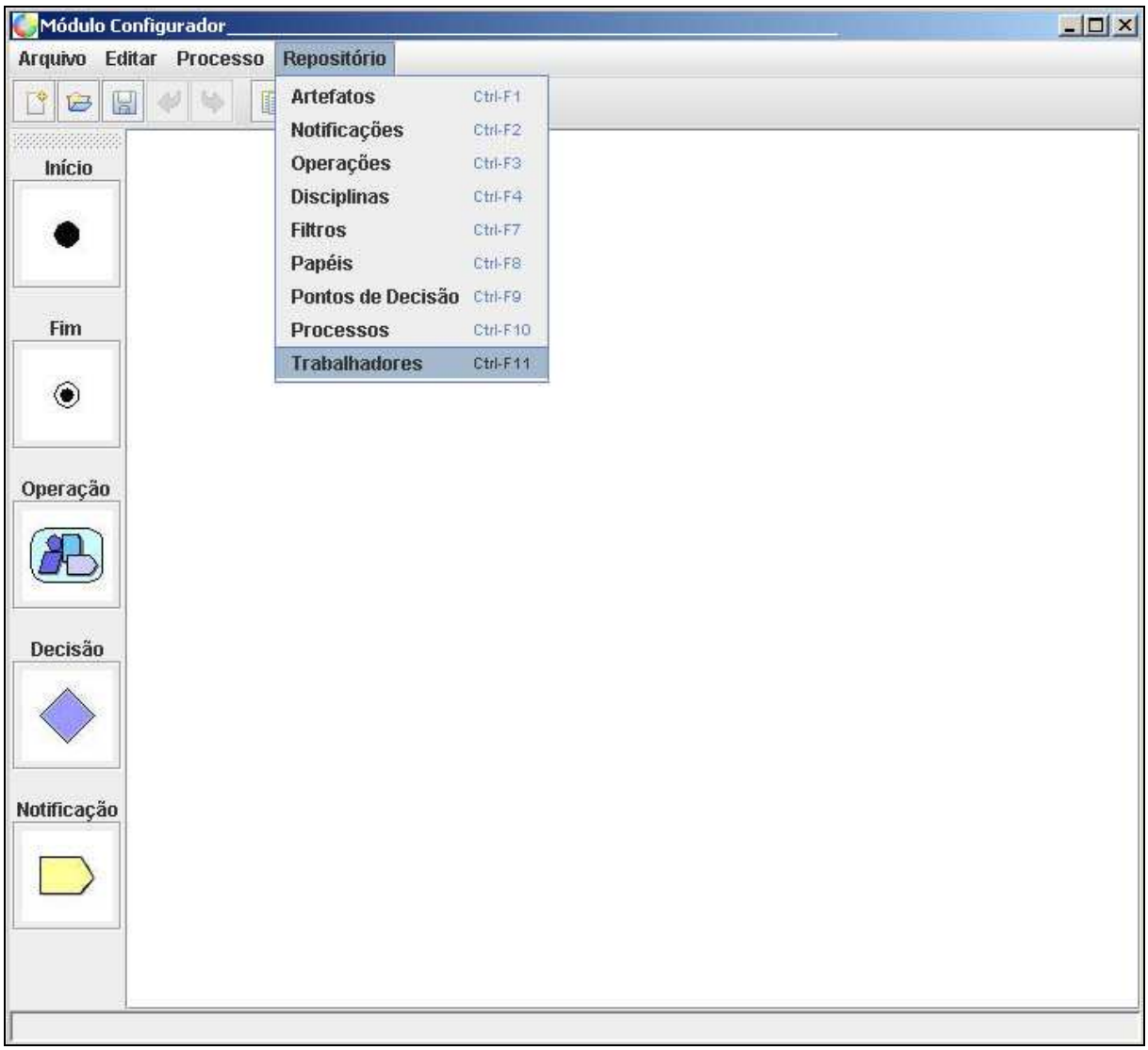


Figura 13 – Cadastros básicos

3.4.1.2 Cadastro de trabalhadores

O cadastro de trabalhadores, ilustrado pela figura 14, é responsável pela manutenção dos dados dos usuários do sistema no módulo executor. Cada trabalhador terá algumas informações mantidas no sistema como nome, sobrenome, papel, usuário, senha e e-mail. Para este processo exemplo, foram cadastrados quatro trabalhadores, onde cada um será o responsável por uma atividade do processo.

Os trabalhadores a serem cadastrados são:

a) “Carlos Silva” que irá assumir o papel de “analista” e será responsável pela atividade de levantamento de requisitos;
b) “Daniel Souza”: irá assumir o papel de “projetista”, sendo responsável pela atividade de análise e projeto;
c) “Rafael Pereira”: este trabalhador terá o papel “programador” e responde pela atividade de “implementação”;
d) “Guilherme Oliveira”: trabalhador que irá assumir papel de “testador” e será o responsável pela atividade de testes.

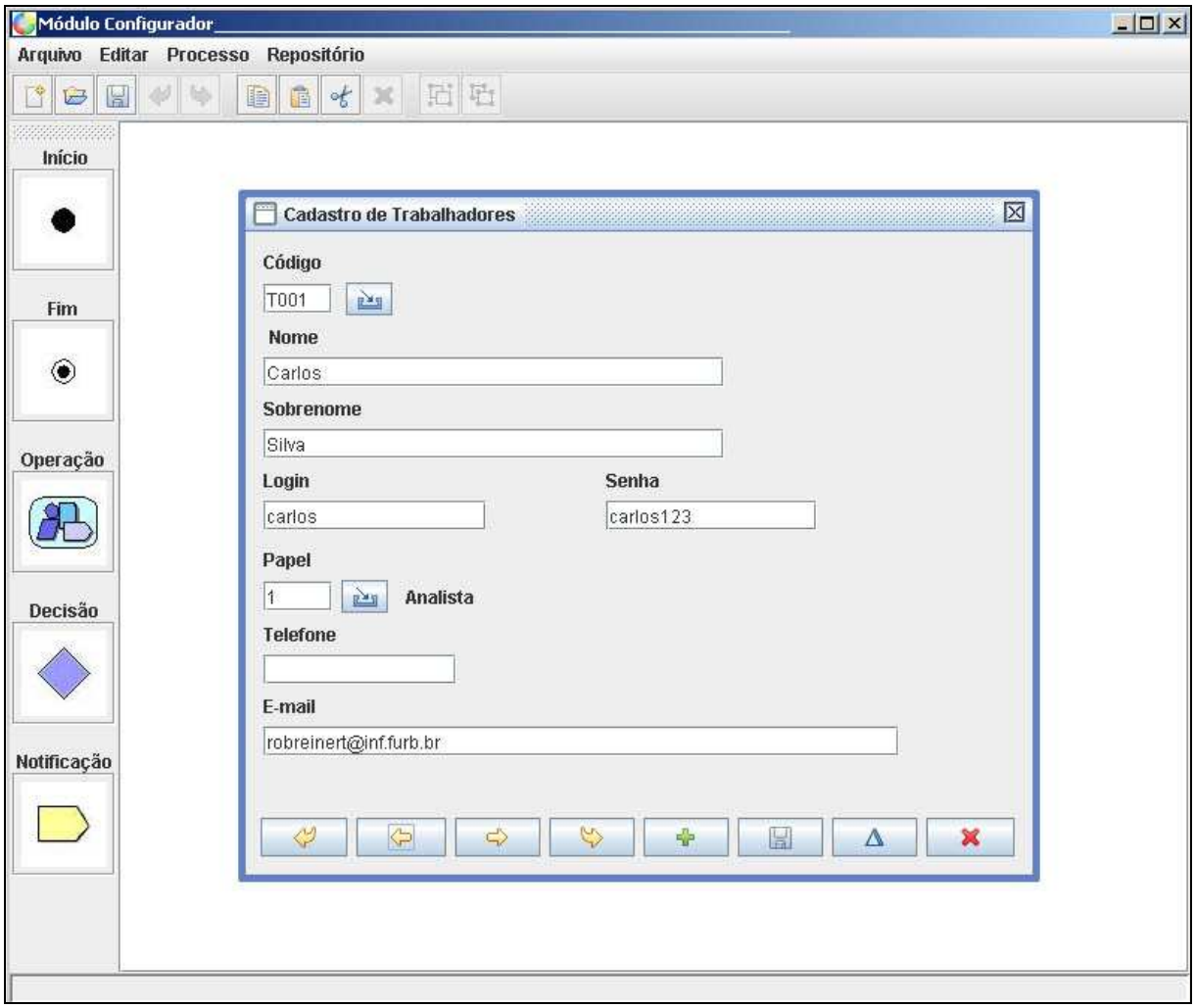


Figura 14 – Cadastro de trabalhadores

3.4.1.3 Cadastro de artefatos

Assim como o cadastro de trabalhadores, também se faz necessário o cadastramento de informações relacionadas aos artefatos que serão gerados pelos trabalhadores nas atividades. Cada artefato poderá conter um conjunto de atributos que descrevem informações relacionadas ao mesmo.

Para este processo exemplo serão criados os seguintes artefatos:

a) documento de requisitos: documento que será preenchido com as necessidades dos usuários. O documento de requisitos irá conter atributos como “data da elicitação”,

"usuário responsável" e "detalhes dos requisitos";

b) especificação dos casos de uso: este documento será preenchido com os fluxos principais e secundários de um caso de uso. Este documento irá conter atributos como "nome do caso de uso", "total de pontos de função", "requisitos associados", "fluxo básico", "fluxos alternativos" e "fluxos de erros";
c) código fonte: este documento será preenchido com dados referentes à construção de um caso de uso. Os atributos "data de início da construção", "data fim da construção" e "total de horas", entre outros, irão compor este documento;
d) *check-list* de teste: este documento irá conter os dados provenientes dos testes efetuados sobre a construção realizada. Este documento terá atributos como "data início do teste", "data fim do teste" e "total de erros".

Na imagem 15 é ilustrada a tela correspondente a este cadastro.

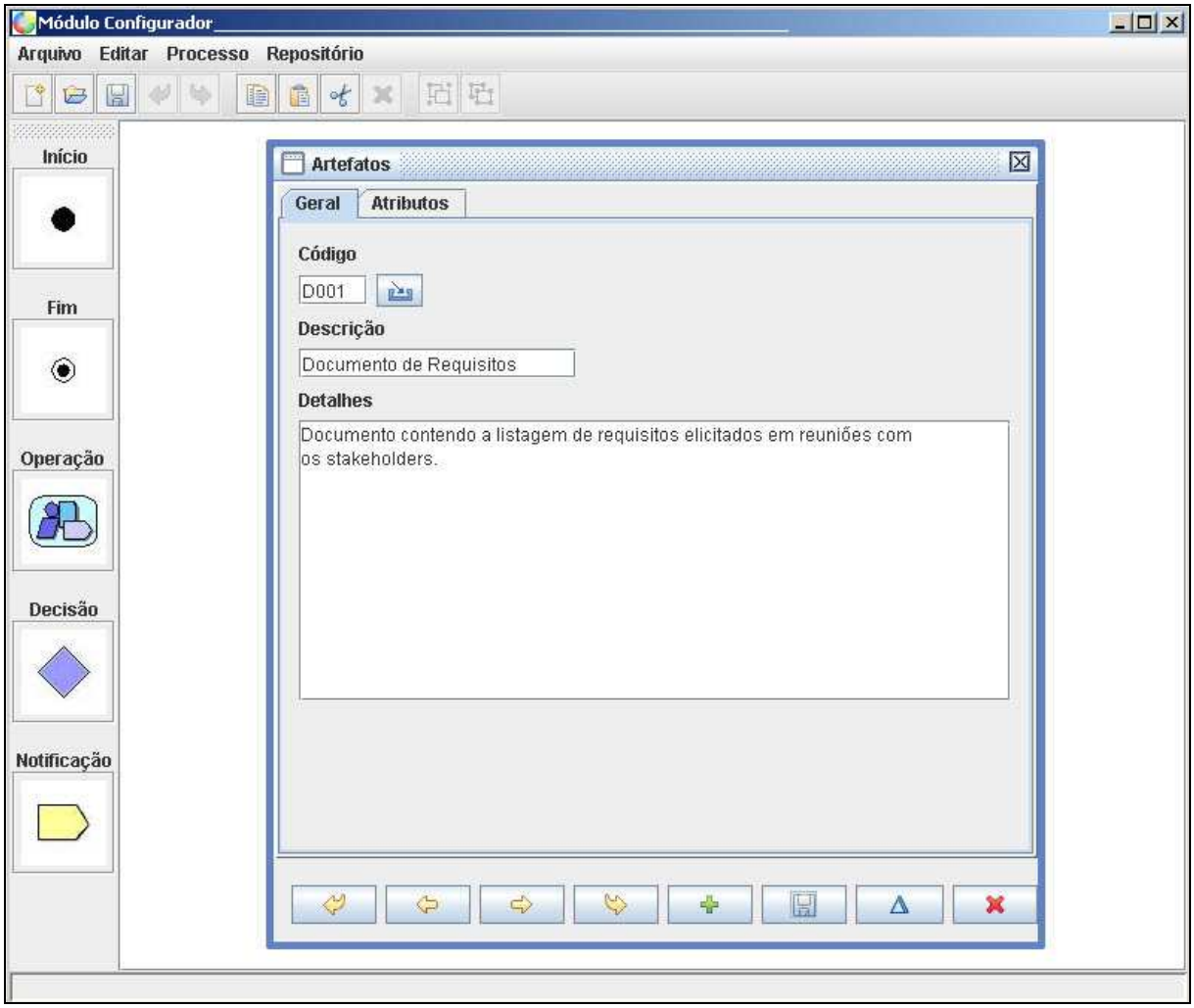


Figura 15 – Cadastro de artefatos

3.4.1.4 Cadastro de Operações

O cadastro de operações corresponde às atividades que serão realizadas pelos trabalhadores com o objetivo de gerar um artefato. Na tela de cadastro de operações existem informações referentes ao proprietário da atividade, ou seja, para quem será aberta uma atividade no módulo de execução e também informações sobre que artefatos serão gerados ou utilizados pelo trabalhador na atividade, bem como à qual disciplina pertence a operação, conforme figura 16.

Conforme citado anteriormente são necessárias as seguintes atividades para este

processo exemplo:

a) elicitar requisitos: atividade onde o trabalhador com papel de “analista de requisitos” irá gerar o artefato “documento de requisitos”;
b) analisar sistema: atividade onde o trabalhador com papel de “analista projetista” irá utilizar o “documento de requisitos” e a partir deste irá gerar o artefato “especificação do caso de uso”;
c) implementar código fonte: nesta atividade o trabalhador que possui o papel “programador” irá construir o caso de uso. Nesta atividade o programador recebe o artefato “especificação dos casos de uso” e irá gerar o artefato “código fonte”;
d) testar sistema: esta é a atividade onde o trabalhador com papel “testador” irá receber a “especificação dos casos de uso” e “código fonte” implementado e irá gerar o artefato “check-list de testes”.

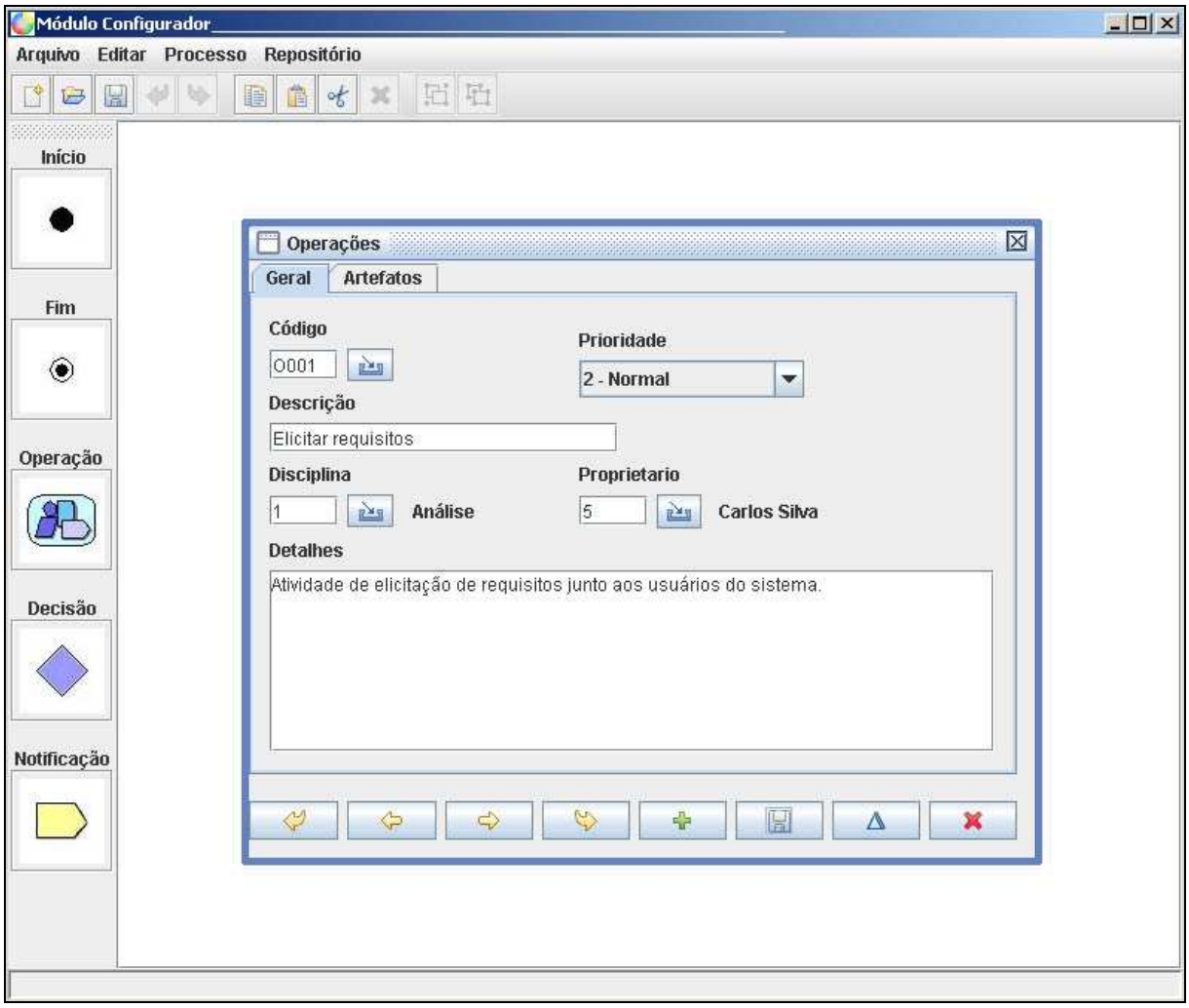


Figura 16 – Cadastro de atividades

3.4.1.5 Cadastro de Notificações

O próximo cadastro que pode ser efetuado no sistema são as notificações. As notificações correspondem a informativos emitidos pelo sistema através de e-mails. As informações necessárias neste cadastro são basicamente o destinatário para a mensagem, um assunto e o corpo do e-mail, conforme ilustrado na figura 17.

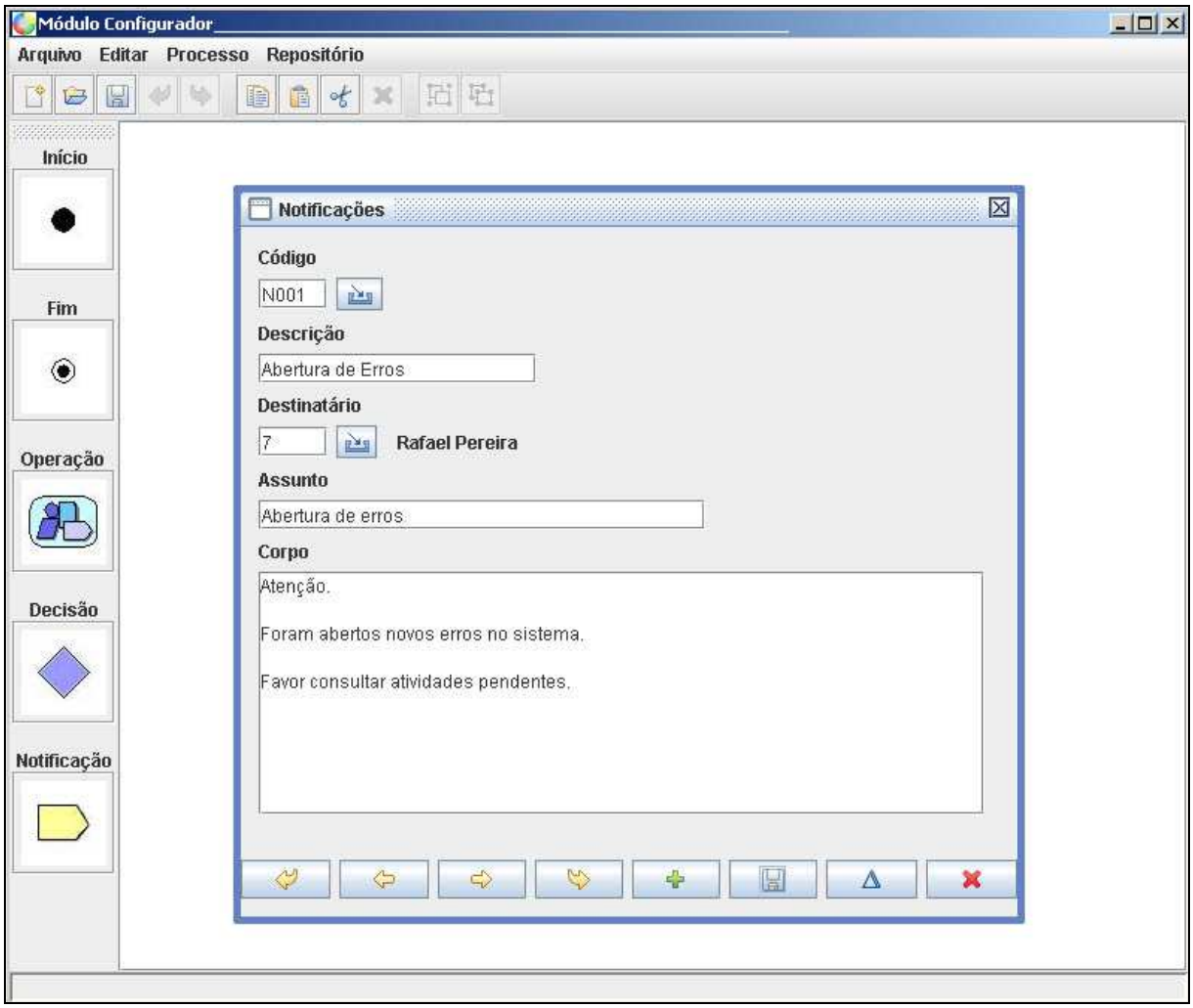


Figura 17 – Cadastro de Notificações

Para este processo exemplo será cadastrada uma notificação que será enviada ao "programador" caso existam erros no artefato "código fonte" gerado na atividade de implementação. Caso não sejam encontrados erros no sistema, uma notificação de liberação do sistema é enviada ao analista de requisitos. No próximo tópico é explicado como o sistema irá detectar esta existência de erros através de filtros.

3.4.1.6 Cadastro de filtros

Para possibilitar o "roteamento" de uma atividade, ou seja, definir a próxima atividade a ser executada no sistema, existem os filtros que são as verificações realizadas sobre os

atributos dos artefatos. Na figura 18 é apresentada a tela de cadastro de filtros.

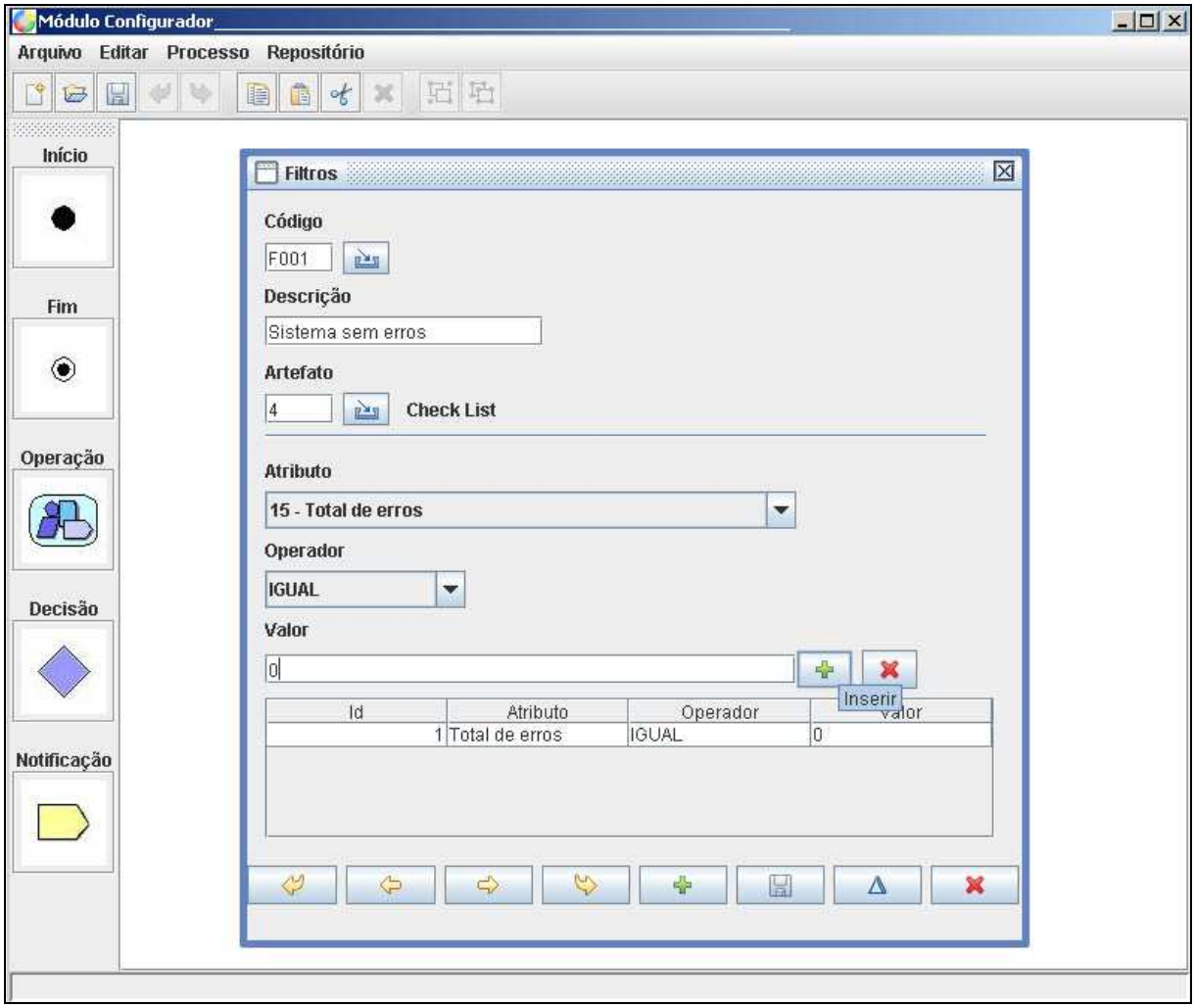


Figura 18 – Cadastro de filtros

Para este processo que está sendo utilizado como exemplo, será criado um único filtro, que se aplicará ao artefato "*check-list* de testes". O conteúdo do atributo "total de erros" será utilizado para realizar a verificação do número de erros existentes na implementação do caso de uso. Caso o número de erros seja igual à zero, o filtro irá retornar um resultado positivo, caso contrário o resultado será negativo. Os resultados deste filtro serão utilizados nos pontos de decisão descritos a seguir.

3.4.1.7 Cadastro de pontos de decisão

Os pontos de decisão possuem a finalidade de efetuar a tomada de decisão sobre qual atividade deve ser executada no sistema. Para possibilitar isto, os pontos de decisão utilizam-se dos filtros, descritos no tópico anterior. Cada filtro pode retornar dois resultados: verdadeiro ou falso. Caso os filtros retornem um resultado verdadeiro, a atividade definida no campo “Atividade Verdadeiro” é executada, caso contrário a outra atividade referenciada na modelagem do processo é executada. Na figura 19 é ilustrado cadastro de pontos de decisão.

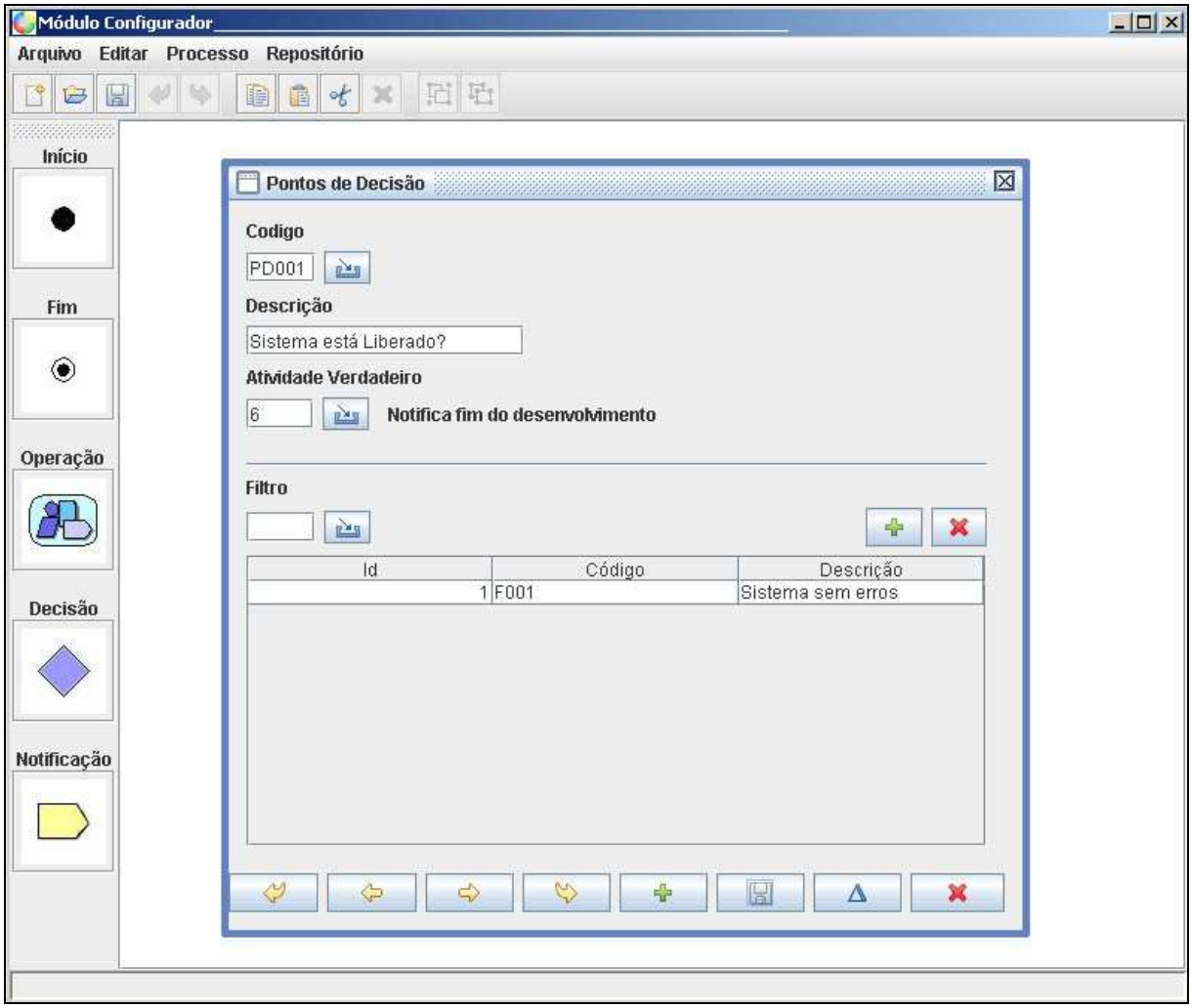


Figura 19 – Cadastro de pontos de decisão

Neste processo exemplo, será cadastrado um único ponto de decisão, utilizando do filtro apresentado anteriormente.

3.4.2 Modelagem do processo

Após efetuados os cadastros básicos no sistema, é possível iniciar a fase de modelagem do processo propriamente dita. A modelagem do processo consiste basicamente em efetuar o relacionamento entre os componentes cadastrados anteriormente.

Este relacionamento é feito de forma gráfica, através dos botões existentes na parte esquerda da tela do sistema. Os componentes gráficos disponíveis para a modelagem são descritos de acordo com a ordem que aparecem na figura 19, respectivamente:

a) início: este botão indica o ponto de início do processo. Ao selecionar este botão será possível inserir um ponto de início na tela. Ao efetuar duplo clique este componente na tela é possível indicar qual o processo cadastro está sendo referenciado;
b) fim: o botão fim é utilizado para se definir quando um processo não possui mais atividades subseqüentes e deve ser finalizado;
c) operação: o botão operação corresponde às atividades de operação efetuadas pelos trabalhadores. Ao se efetuar duplo clique sobre este componente na tela é aberta a tela de cadastro de operações, onde é possível indicar qual a atividade correspondente;
d) decisão: o botão decisão é utilizado para inserção de atividades de ponto de decisão no modelo. Assim como nas atividades de operação, também pode-se efetuar duplo clique neste componente, sendo que neste caso será exibida a tela de cadastramento de pontos de decisão;
e) notificação: este botão possibilita a inserção de uma atividade de notificação no modelo do processo. Ao ser efetuado um duplo clique sobre este componente é exibida a tela de cadastro de notificações, para que se possa relacionar este item

com uma notificação cadastrada.

Para interligar os componentes do modelo podem-se utilizar os conectores, que são as setas direcionais. Estas setas indicam o caminho pelo qual será seguido o fluxo de execução das atividades. Cada componente inserido na tela possui no centro um ponto, onde é possível iniciar a interligação entre os componentes.

Para liberar um modelo de processo para o módulo de execução, tem-se o item de menu “Processo->Ativar”. Este procedimento irá efetuar algumas consistências no modelo criado, como a existência de dados cadastrados para as entidades modeladas e em seguida gerar uma estrutura em banco de dados que é disponibilizada para o módulo de execução.

3.4.3 Execução do processo

Após modelado e ativado um processo de software no módulo de configuração, entra em ação o módulo de execução. Este módulo, que é disponibilizado em ambiente *web*, fica disponível aos usuários do sistema acessando um endereço definido pelo administrador do sistema em um navegador *web*.

Nos tópicos a seguir são apresentados os passos necessários para a instanciação dos modelos de processo definidos anteriormente no módulo de configuração, bem como as operações que podem ser efetuadas sobre as atividades pendentes para cada usuário.

3.4.3.1 Login

Para acessar o módulo de execução um usuário necessita estar cadastrado no módulo de configuração com um usuário e senha apropriados. Na figura 21 é ilustrada a tela de login do módulo de execução para o usuário “Carlos”, cadastrado no módulo de configuração.

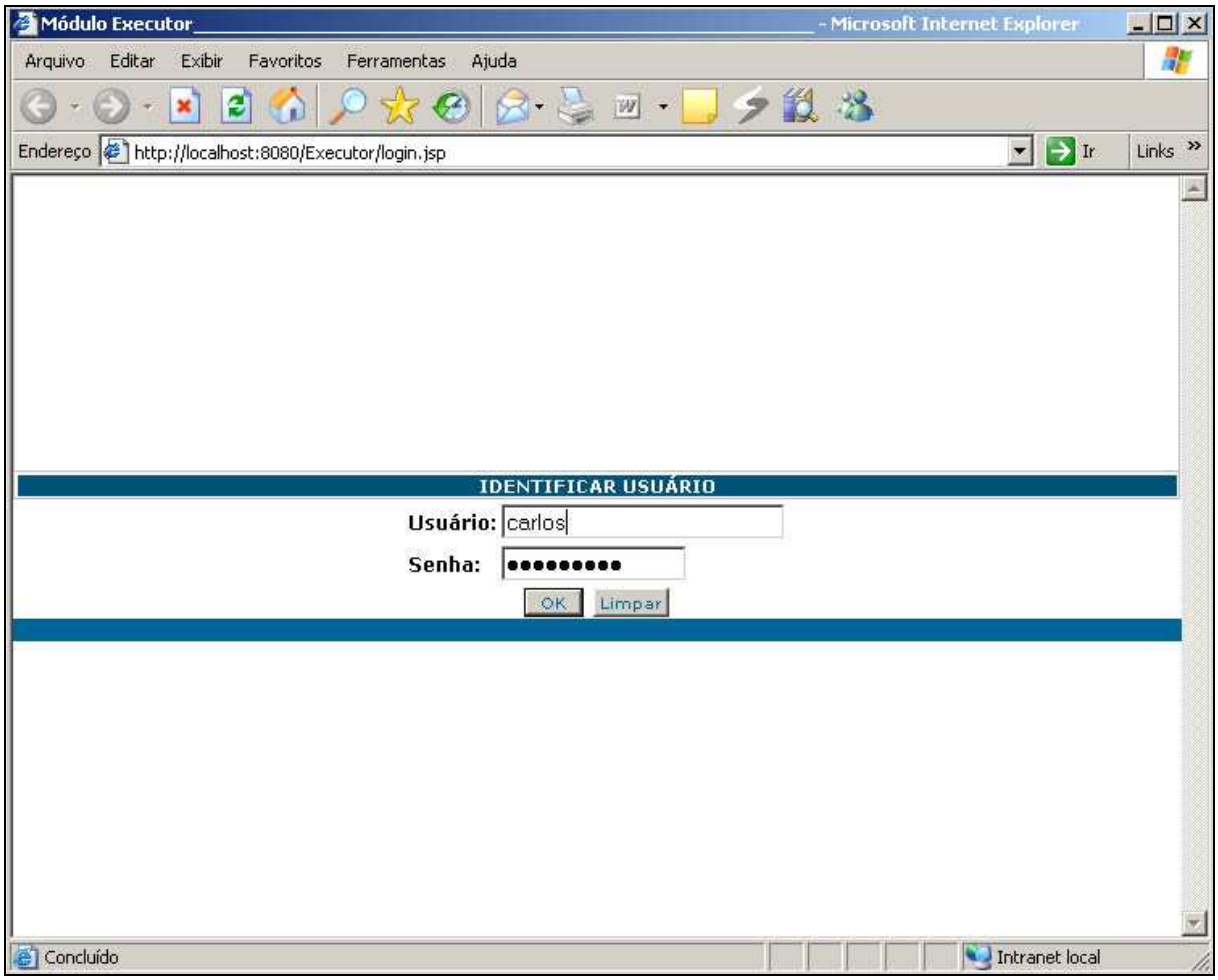


Figura 20 – Tela de login

3.4.3.2 Caixa de Entrada – Iniciar novo processo

Após efetuado o login no sistema, é exibida para o usuário a tela de caixa de entrada. Nesta tela qualquer usuário poderá iniciar (instanciar) um novo processo, acompanhar as atividades pendentes e emitir relatório de execução de atividades. Caso o usuário selecione a opção “Novo Processo” o sistema exibe para o usuário uma listagem com todos os processos liberados pelo módulo configurador. Nesta tela o usuário poderá selecionar um processo novo que deseja iniciar. No exemplo abaixo, o usuário “Carlos” irá instanciar o processo que foi cadastrado anteriormente no módulo de configuração, conforme figura 21.

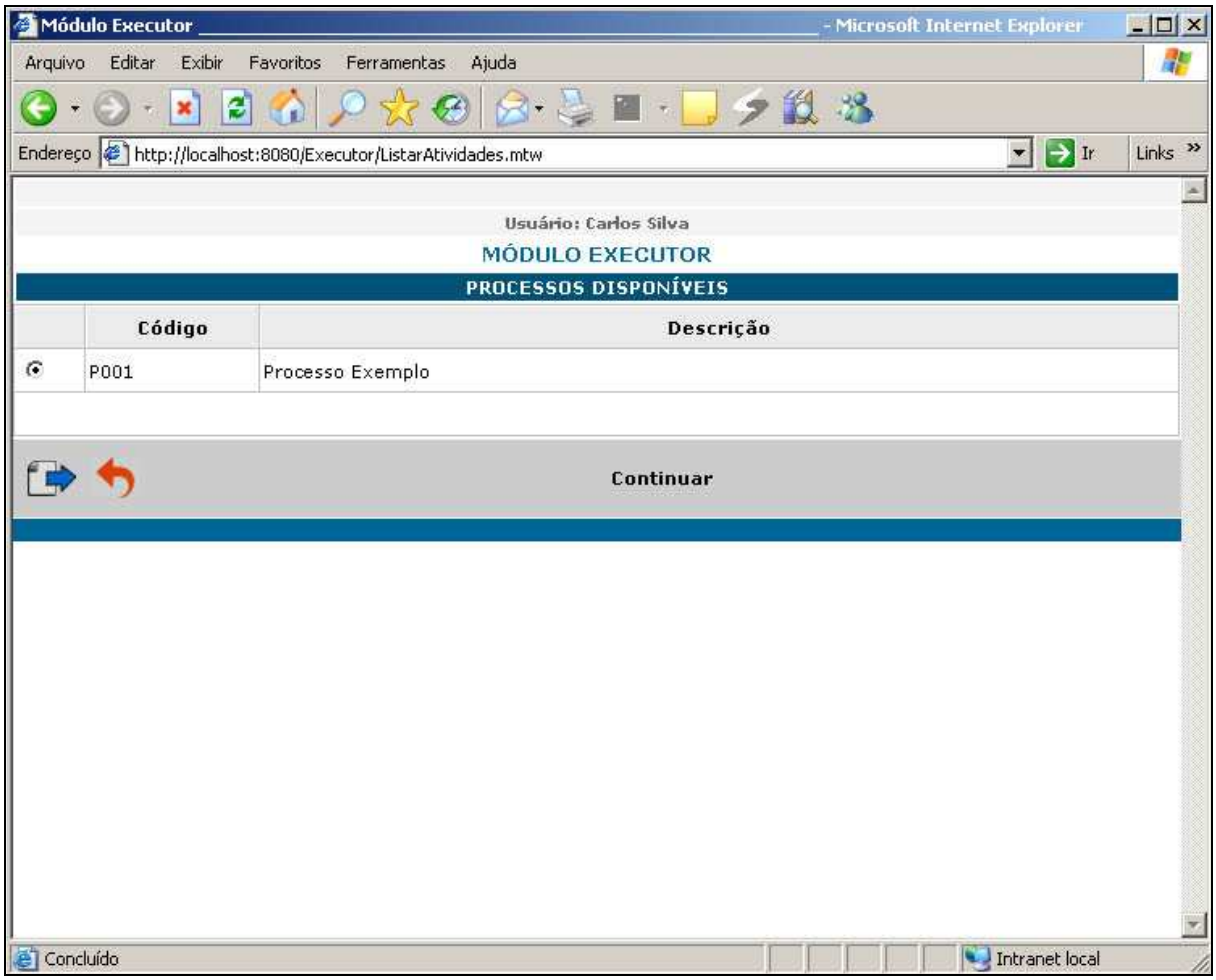


Figura 21 – Iniciar novo processo

3.4.3.3 Caixa de Entrada – Atividades Pendentes

Após ter sido instanciado um processo no sistema, será iniciado a execução das atividades conforme modelado no módulo de configuração. Neste processo exemplo, a primeira atividade a ser executada após a instanciação do processo no sistema é a “elicitação de requisitos” que possui como responsável o usuário “Carlos Silva”. Portanto ao entrar na tela de atividades pendentes, conforme figura 22, é apresentada uma nova atividade para o usuário, no caso a atividade de “Elicitar Requisitos”.

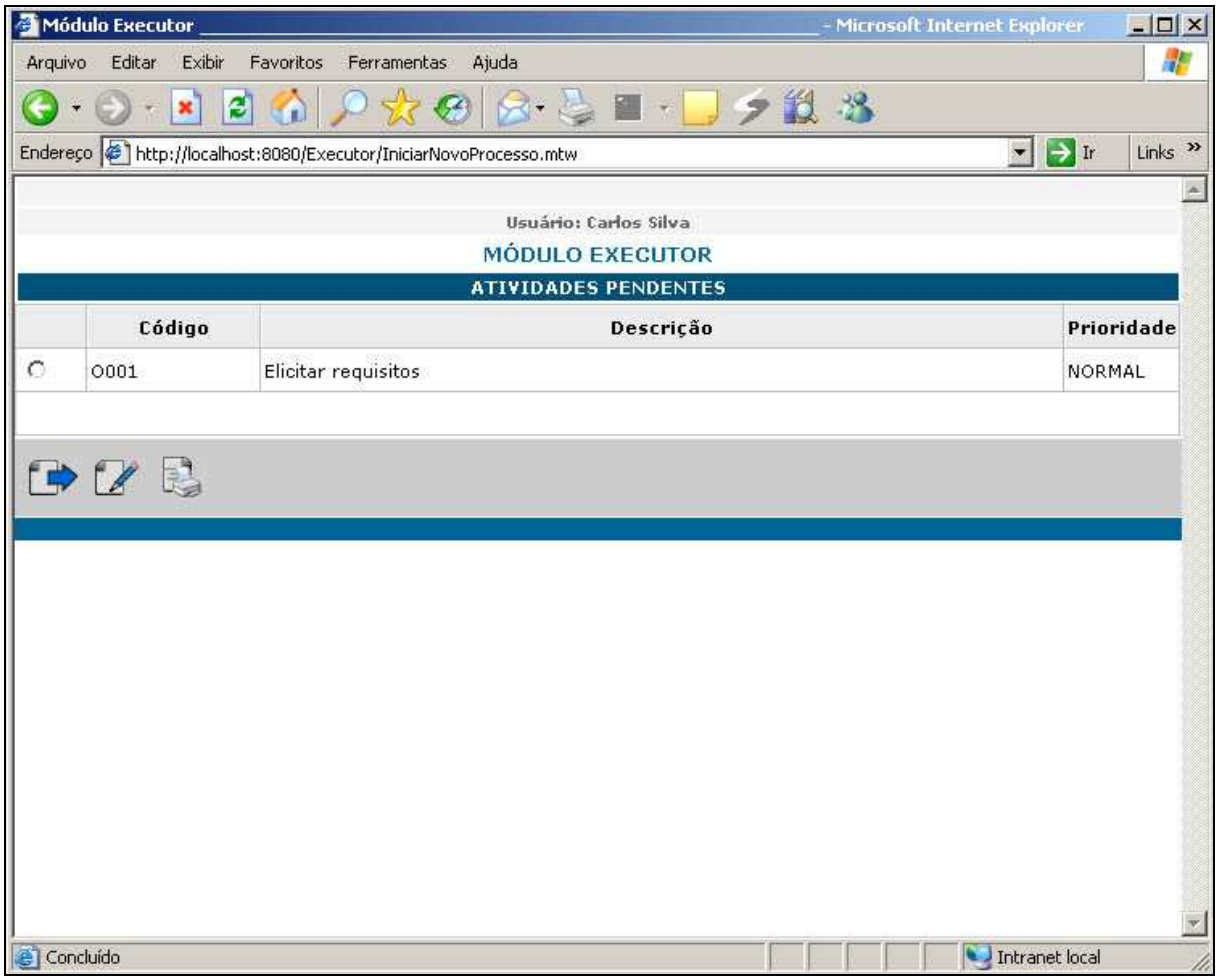


Figura 22 – Caixa de entrada de atividades pendentes.

3.4.3.4 Detalhar Atividade

Caso o usuário possua alguma atividade pendente na lista de atividades da caixa de entrada, o mesmo poderá selecionar uma delas e mostrar os detalhes da mesma, selecionando a opção “Detalhar atividade”. Após selecionada esta opção o sistema exibe a tela com as informações referentes à atividade pendente para o usuário. Entre estas informações há também uma lista de artefatos que podem ser manipulados pelo usuário na atividade em questão. Serão disponibilizadas para o usuário as opções de “Encaminhar Atividade” e “Editar Artefato”. Para o exemplo utilizado, é apresentado na atividade “Elicitar Requisitos” o artefato “Documento de Requisitos”, que foi relacionado a esta atividade no módulo de

Configuração. Na figura 23 é apresentada a tela correspondente.

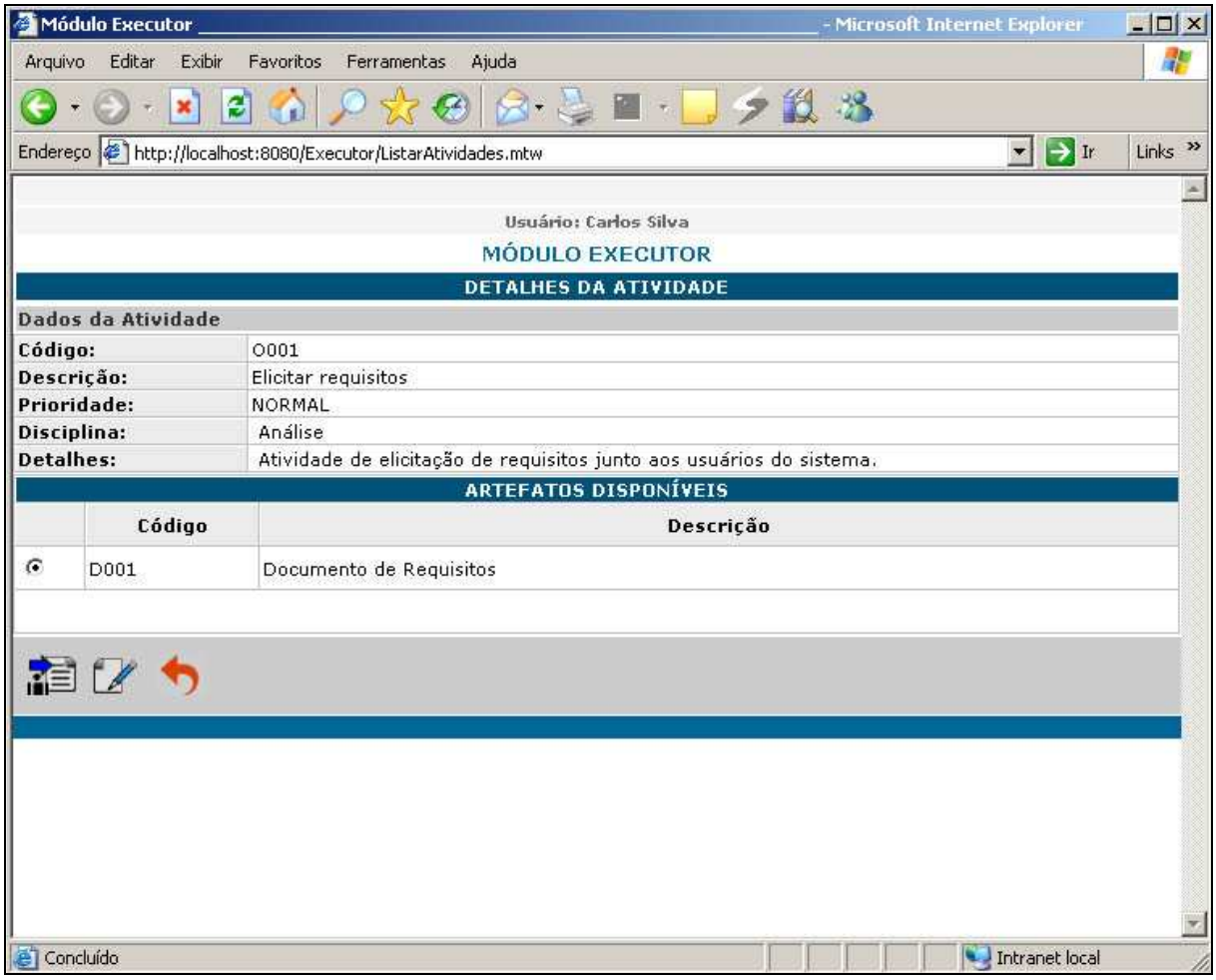


Figura 23 – Detalhes da atividade

3.4.3.5 Detalhar Artefato

Caso o usuário selecione um artefato e em seguida a opção “Editar Artefato” será exibido a tela de detalhes do artefato. Nesta tela o usuário poderá preencher ou visualizar as informações contidas nos atributos do artefato selecionado. Uma vez preenchidas as informações necessárias referentes ao artefato, o usuário poderá salvá-las, ou simplesmente voltar à tela anterior. Para este exemplo, são disponibilizados para o usuário os atributos “Data da Elicitação”, “Usuário Responsável” e “Detalhes dos Requisitos”, que são os atributos cadastrados para o artefato “Documento de Requisitos” selecionado anteriormente.

Na figura 24 é apresentada a tela de detalhamento de artefatos.

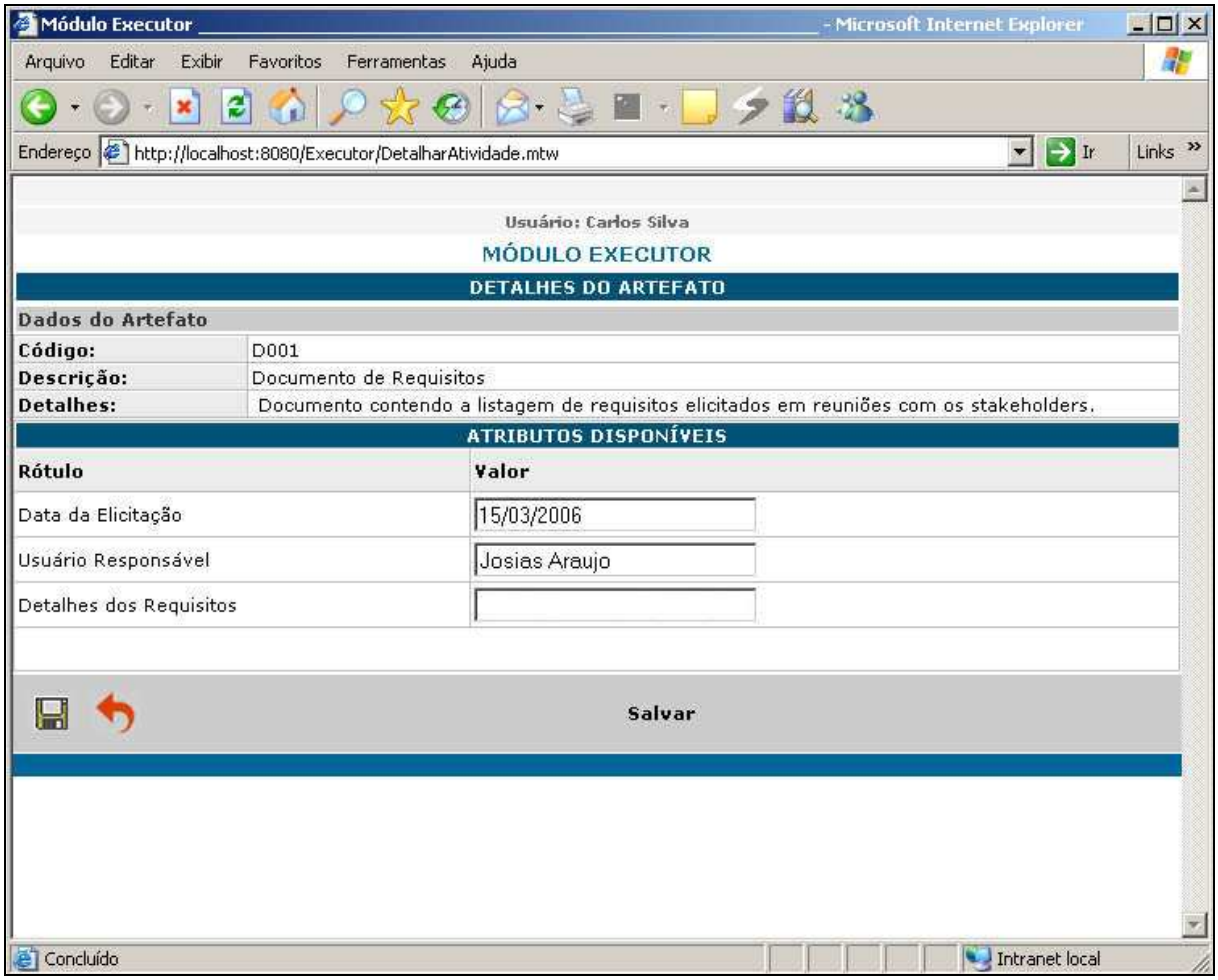


Figura 24 – Detalhes do artefato

3.4.3.6 Continuação do fluxo de execução

Após editados os artefatos necessários, o usuário poderá encaminhar a atividade para o próximo executor. Para este processo exemplo, após ter sido encaminha a atividade “Elicitar Requisitos” será iniciada a atividade “Analisar Casos de Uso”. A atividade que estava pendente para o usuário “Carlos” deixa de ser listada na sua lista de atividades pendentes e passa agora a ser listada para o usuário “Daniel Souza”, conforme figura 25.

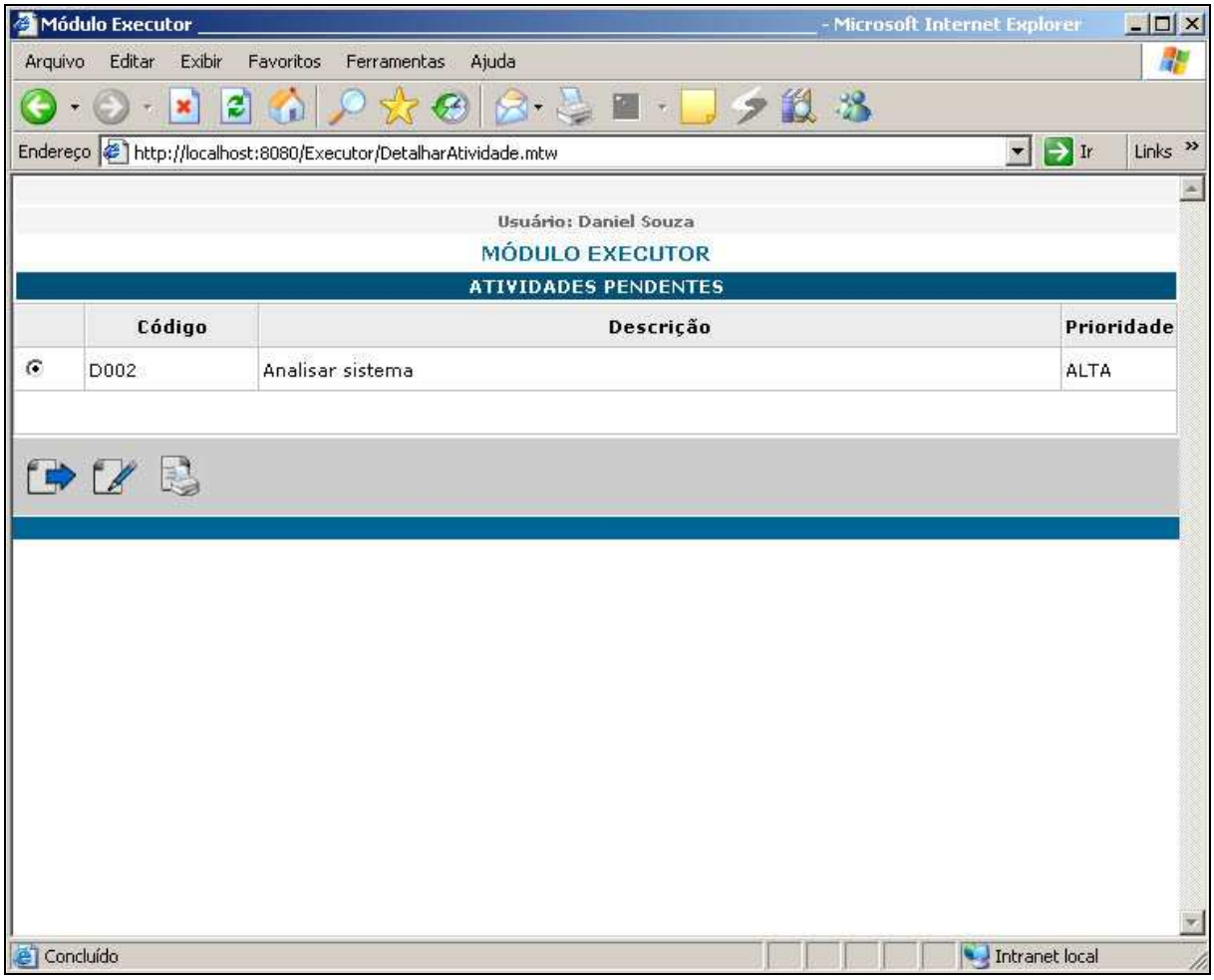


Figura 25 – Continuação do fluxo de execução – Lista de atividades

O fluxo de execução das atividades continua normalmente, repassando de usuário para usuário até o ponto onde ocorre a atividade “Testar Caso de Uso”. Nesta atividade o usuário “Guilherme Oliveira” irá preencher o campo “Total de Erros”. Conforme definido na modelagem do processo, caso o usuário informe um valor diferente de zero a próxima atividade a ser executada será novamente a atividade “Implementar Código Fonte”. Como conseqüência, será novamente listada na tela de atividades pendentes do usuário “Rafael Pereira” a atividade correspondente à implementação do código fonte.

3.4.3.7 Gerar Relatórios de Atividades

Caso o usuário selecione a opção “Gerar Relatório de Atividades” será exibido a tela de parâmetros de relatórios. Nesta tela o usuário irá informar os parâmetros para a filtragem dos dados referentes aos processos e atividades executadas, conforme figura 26.

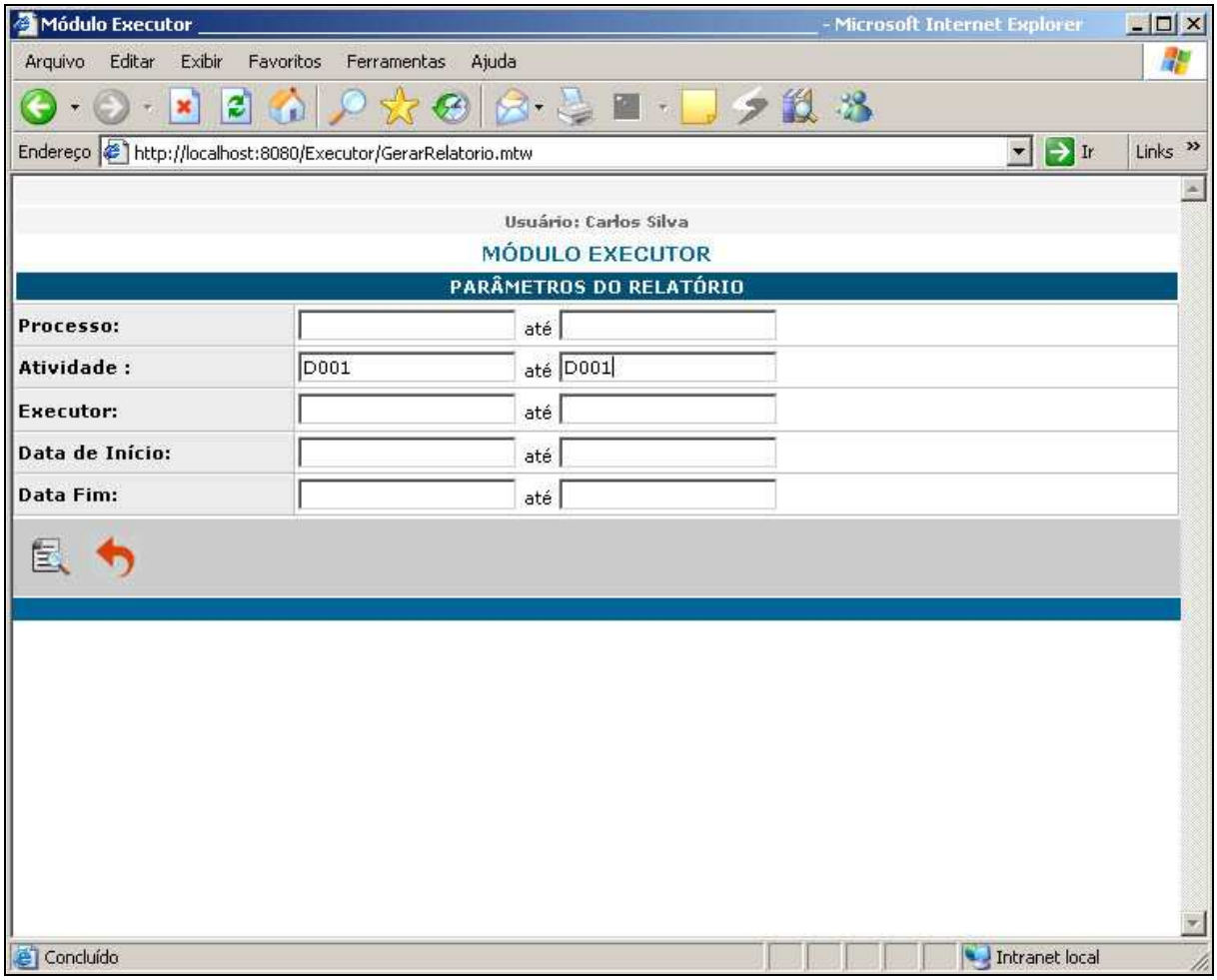


Figura 26 – Parâmetros de Relatórios

Após informados os parâmetros o usuário solicita a geração do relatório e o sistema exibe os dados de execução das atividades, conforme figura 27.

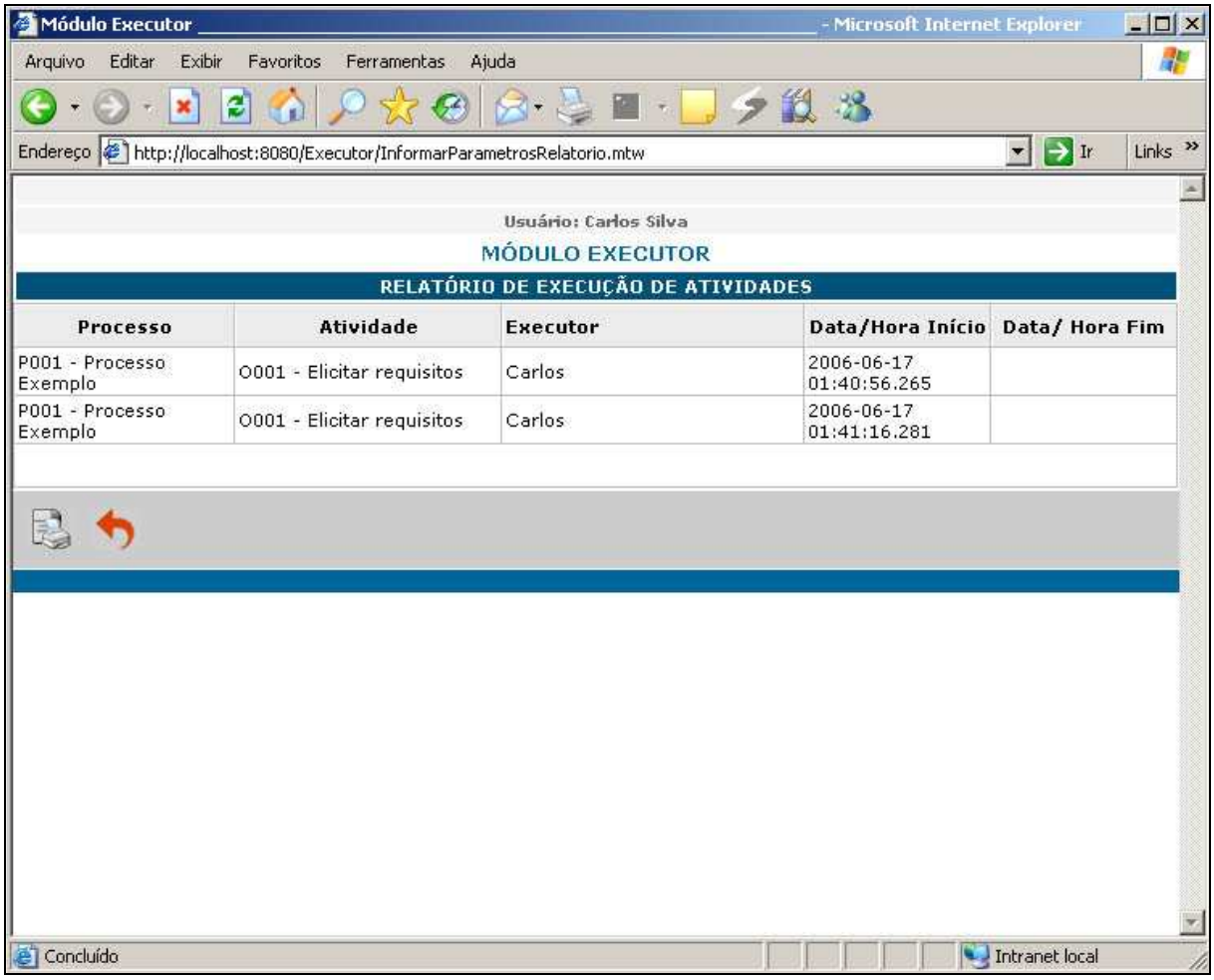


Figura 27 – Relatório de Execução de Atividades

3.5 RESULTADOS E DISCUSSÃO

No trabalho desenvolvido por Bork (2003), foi apresentada a implantação e customização de um processo de software baseado no RUP. Esta implantação, no entanto, não envolve um processo automatizado de *workflow* para o encaminhamento e tomada de decisões nas atividades que compõe o processo de software, mas sim a utilização de formulários padrões que permitiam o encaminhamento de solicitações de serviços aos responsáveis na empresa. Com esta ferramenta de *workflow* desenvolvida, podem-se gerar formulários customizáveis através do cadastro de artefatos e seus respectivos atributos e com a vantagem de estes atributos servirem de insumo para o mecanismo de controle de tomada de decisão.

Este trabalho possui características que o classificam como sendo um sistema de *workflow*. O fluxo de trabalho é automatizado pela ferramenta, onde cada atividade é listada para cada usuário do sistema na medida em que são feitas as execuções no processo. Neste trabalho foi implementado o *workflow* do tipo administrativo, devido à existência de regras de coordenação simples e pelo fato de não haver a troca de informações com outros sistemas.

Não foram feitos testes reais em empresas ou instituições que utilizam algum processo de software para obtenção de resultados comparativos entre a implantação de processos em papel *versus* a implantação através deste sistema de *workflow*. Grande parte dos testes foi efetuada com o orientador deste trabalho em simulação de processos básicos.

4 CONCLUSÕES

A utilização de sistemas de *workflow* para modelagem e gerenciamento de processos de software apresenta-se como um item fundamental para a organização do fluxo de trabalho em instituições que pretendem adotar um processo de software efetivo e eficiente. A adoção de documentações é bem-vinda, porém sem um mecanismo “real” de execução para esta documentação dificilmente consegue-se implantar uma “cultura” de comprometimento com os procedimentos e regras da instituição.

Em relação aos objetivos definidos neste trabalho, chega-se à conclusão de que os mesmos foram alcançados, pois foi possível modelar processos básicos de software graficamente e executá-los conforme modelado. Com a utilização do *framework* JGraph foi possível criar um editor gráfico, o que permitiu a criação de fluxogramas para modelar os processos. No que diz respeito à execução das atividades, a utilização do *framework web* Mentawai em conjunto com o servidor Apache Tomcat permitiu que se conseguisse realizar esta execução em um ambiente web.

Algumas limitações foram observadas ao final do desenvolvimento, como a falta da possibilidade de criar processos em vários subníveis, que traria um melhor acoplamento entre os processos. Na modelagem e execução das atividades, também observa-se que não existe a possibilidade da utilização de paralelismo entre as atividades, o que em alguns processos mais complexos torna-se necessário.

Ao final deste trabalho concluiu-se que a ferramenta desenvolvida possui potencial para fazer a modelagem de processos genéricos de negócio, não se restringindo apenas a processos de software.

Quanto às ferramentas e tecnologias utilizadas, a aderência das mesmas às necessidades do projeto foi total. Muitas das tecnologias utilizadas tiveram que ser estudadas

e pesquisadas durante o decorrer deste trabalho, sendo que as mesmas auxiliaram e facilitaram bastante o seu desenvolvimento.

Por fim, chega-se a conclusão de que este trabalho ainda pode ser evoluído em vários pontos, conforme tópico a seguir e que a implementação produzida até o momento possui tecnologias amplamente utilizadas atualmente na indústria de software e de forma livre, o que permite a evolução desta ferramenta para futuros trabalhos.

4.1 EXTENSÕES

Como possíveis extensões ao trabalho desenvolvido cita-se:

a) possibilitar a criação de sub-processos no módulo configurador para permitir um melhor acoplamento entre os processos;
b) permitir a inserção de anexos aos artefatos, tornando-os mais flexíveis com as necessidades de um sistema de *workflow*;
c) possibilitar a criação de variáveis que retornem informações sobre o sistema, como por exemplo, a data atual e permitir que as mesmas sejam utilizadas nos filtros para roteamento das atividades.

REFERÊNCIAS BIBLIOGRÁFICAS

AMBLER, S. W. **Modelagem ágil.** Tradução Acauan Fernandes. Porto Alegre: Bookman, 2004.

APACHE FOUNDATION. **Apache Derby database.** USA, [2006?]. Disponível em: <http://db.apache.org/derby>. Acesso em: 03 mar. 2006.

ARAUJO, R. M.; BORGES, M. R. **Sistemas de workflow**. In: CONGRESSO DA SOCIEDADE BRASILEIRA DA COMPUTAÇÃO, 2001, Fortaleza. Disponível em: <http:// chord.nce.ufrj.br/cursos/teesi/textos/apostilaJai2001div.pdf>. Acesso em: 15 abr. 2006.

BORK, C. J. **Customização e implantação de um processo de desenvolvimento de software baseado no Rational Unified Process (RUP).** 2003. 98 f. Trabalho de Estágio Supervisionado (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

BOSATO, A.; REZENDE, E.; BITTES, J. M. **Sistemas de workflow apoiando processos de engenharia de software.** Rio de Janeiro: Núcleo de Computação e Eletrônica – UFRJ, 2002.

HAUCK, J. R.; WANGENHEIM, C. G. **Modelando o processo de software em uma pequena empresa – o caso Void Caz**. In: SIMPÓSIO INTERNACIONAL DE MELHORIA DE PROCESSO DE SOFTWARE, 2004, São Paulo. Disponível em: <http://ssooweb04.univali.br/lqps/anexos/simpros2004-voidcaz-vref.pdf>. Acesso em: 30 mai. 2006.

ITO, Marcia. **Processo de aquisição de produto de software dirigido por casos de uso**. São Paulo, [2004]. Disponível em: <http://www.choose.com.br/infochoose/ artigos/46art01.htm>. Acesso em: 16 abr. 2006.

JBOSS INC. **Hibernate Relational persistence for Java and .NET.** USA, [2006?]. Disponível em: < http://www.hibernate.org>. Acesso em: 20 abr. 2006.

JGRAPH LTD. **JGraph library.** UK, [2006?]. Disponível em: <http://www.jgraph.com>. Acesso em: 22 mar. 2006.

KURNIAWAN, B. **Java para web com servlets, jsp e ejb.** Tradução Savannah Hartmann. Rio de Janeiro: Ciência Moderna, 2002.

OLIVEIRA, J. **Caracterizando sistemas de workflow**. Read, Porto Alegre, 1996. Disponível em: <http:// inf.ufrgs.br/~palazzo/docs/read/workflow.htm/>. Acesso em: 20 fev 2006.

OLIVEIRA, S. **Mentawai web framework**. Porto Alegre, [2006?]. Disponível em: < http://www.mentaframework.org>. Acesso em: 25 mar. 2006.

PEREIRA, L. M.; CASANOVA, M. A. **Sistemas de gerência de workflows:** características, distribuição e exceções**.** Rio de Janeiro, [mar. 2003]. Disponível em: <ftp://ftp.inf.puc-rio.br/pub/docs/techreports/03_11_pereira.pdf/>. Acesso em: 18 maio 2006.

RATIONAL SOFTWARE. **Rational unified process.** USA, [2005?]. Disponível em: <http://www.rational.com>. Acesso em: 3 mar. 2006.

REIS, C. L. **Introdução à modelagem de processos de software.** Belém, [2004]. Disponível em: < http://www.cultura.ufpa.br/clima/mod_proc/apostila.pdf/>. Acesso em: 17 abr. 2006

SCOTT, K. **O processo unificado explicado.** Tradução Ana M. de Alencar Price. Porto Alegre: Bookman, 2003.

SUN MICROSYSTEMS. **Java 2 platform, enterprise edition.** [S.1.], set. 2005. Disponível em: <http://java.sun.com/j2ee/>. Acesso em: 17 mar. 2006.

TELES, V. M. **Extreme programming.** São Paulo: Novatec, 2004.

WIKIPÉDIA. **Processo de desenvolvimento de software**. [S.l.], [2006]. Disponível em: <http://pt.wikipedia.org/wiki/Processo_de_desenvolvimento_de_software>. Acesso em: 19 mar. 2006

WORKFLOW MANAGEMENT COALITION. **The enterprise content management association.** UK, [2005?]. Disponível em: <htpp://www.aiim.org/wfmc>. Acesso em: 10 abr. 2006.