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

Aluno: Roberto Reinert

Orientador: Everaldo A. Grahl

Roteiro de apresentação

- Introdução
  - Objetivos
- Fundamentação Teórica
  - Workflow
  - Processo de Software
  - Contexto do tema
- Desenvolvimento do trabalho
  - Requisitos
  - Especificação
  - Implementação
  - Resultados e discussão
- Conclusão
- Extensões

Introdução

- Desenvolver software com qualidade e dentro do prazo estabelecido, tem sido um dos grandes desafios para os gerentes de projetos de software atualmente.
- Uma das principais causas de problemas em projetos de software é a falta de um processo de desenvolvimento de software claramente definido, efetivo e controlado. Para que se possa ter um processo de software controlado é necessário que haja um ambiente integrado para definição destes processos, onde seja possível modelar as várias etapas que compõe o fluxo de trabalho entre as pessoas envolvidas no mesmo.

Objetivos do Trabalho

- O objetivo deste trabalho é desenvolver um sistema de *workflow* para modelagem e execução de processos de software. Os objetivos específicos são:
  - Possibilitar a criação de modelos de processo de software de forma gráfica;
  - Permitir aos envolvidos no processo a efetiva execução dos modelos gerados.

Fundamentação Teórica

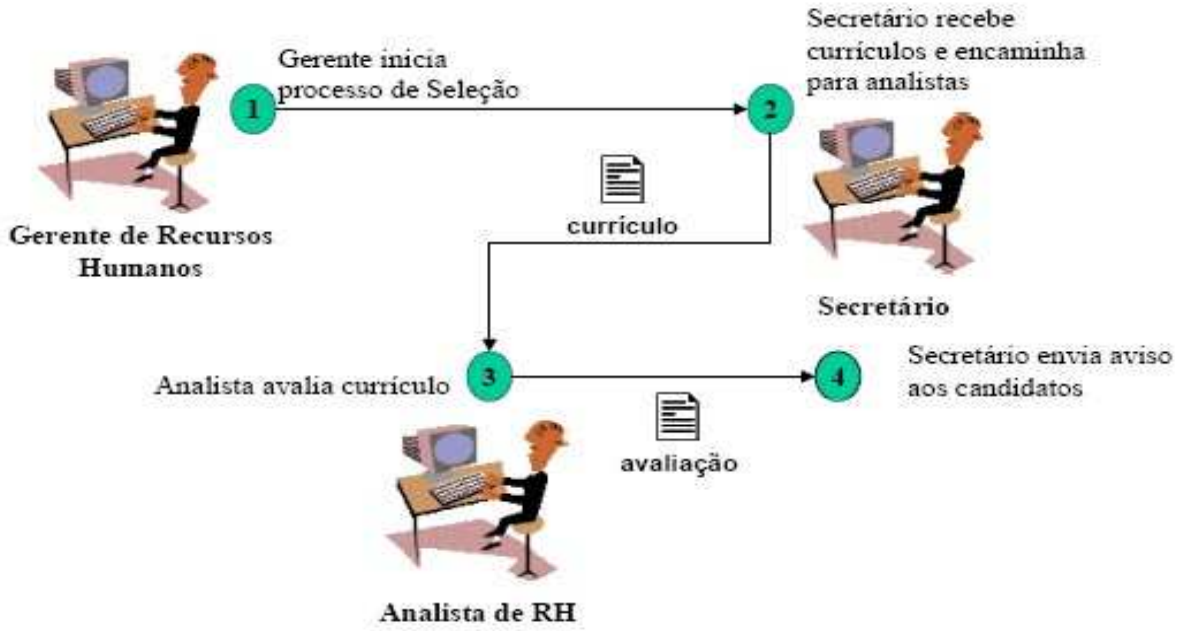
Workflow - Introdução

- 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.
- Com o intuito de fornecer um suporte à atividades que envolvem coordenação e comunicação entre pessoas em grupo de trabalho, surgiram os sistemas de *workflow*.

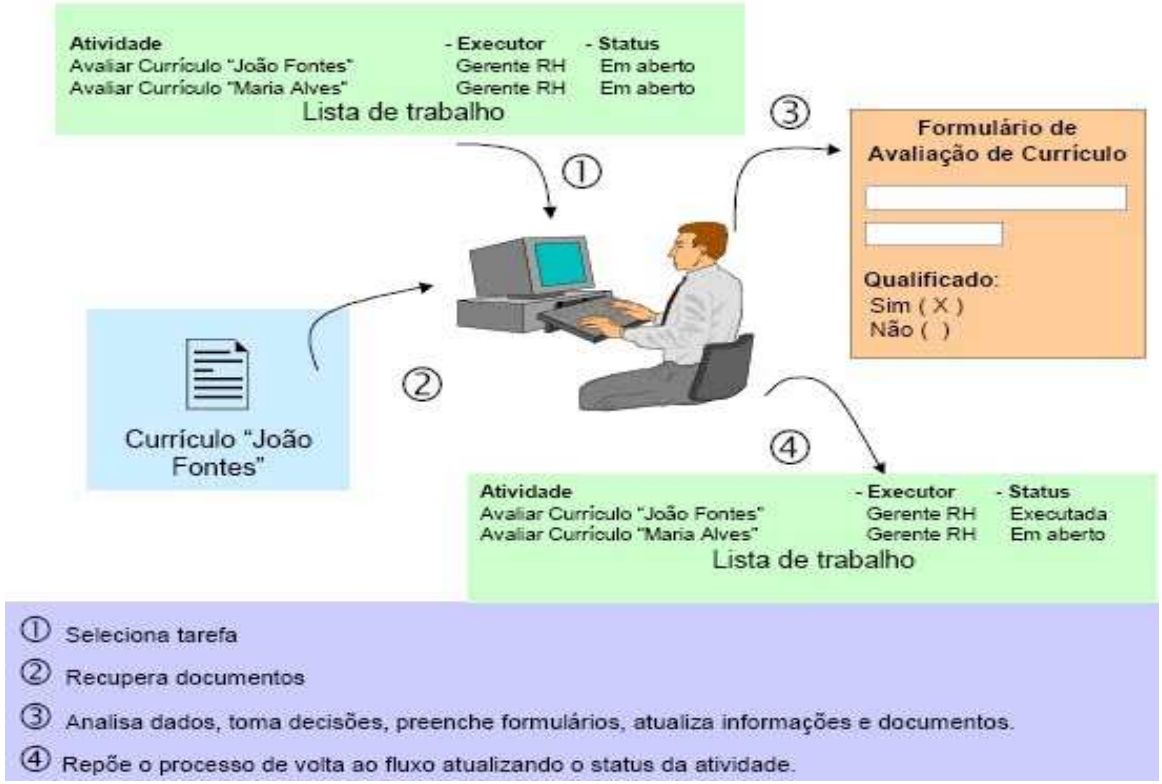
Workflow - Definição

- Segundo a Workflow Management Coalition (WFMC), *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.

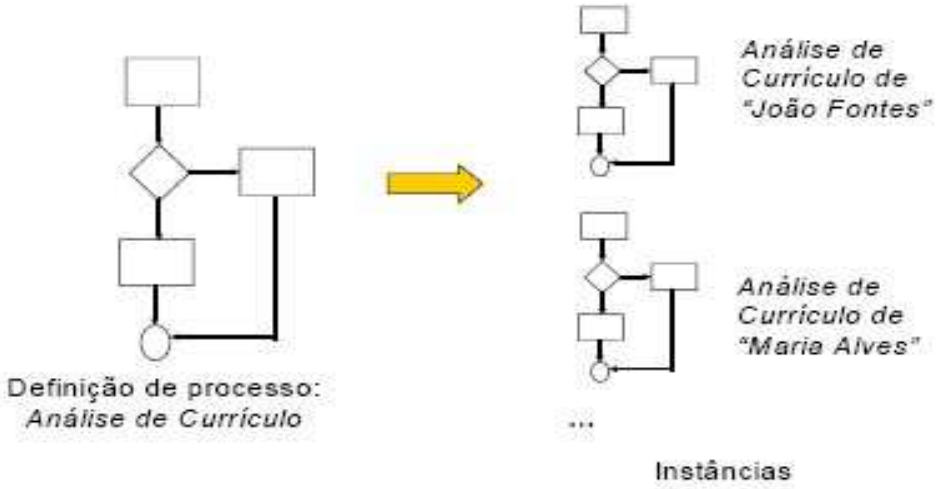
Atividades em um Workflow



Interação entre um usuário e o sistema de workflow



Instanciação de um modelo de processo



Processo de Software - Definição

- Segundo Scott (2004, 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.
- 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.

Processo de Software - Componentes

- Atividades;
- Artefatos;
- Trabalhadores.

Contexto do tema

- Nos trabalhos correlatos de Bork(2003) e Hauck(2004) são relatadas a customização e a implantação de um processo de desenvolvimento de software nas empresas Dynamix Software e VOID CAZ Sistemas respectivamente. Nestes trabalhos fica clara a dificuldade de se encontrar um modelo de processo de software pronto, que se enquadre nas necessidades específicas de pequenas empresas.

Desenvolvimento do trabalho

- A partir dos problemas abordados nos trabalhos correlatos e conceitos estudados iniciou-se a fase de desenvolvimento do trabalho.

Requisitos

- permitir o cadastramento de atividades, trabalhadores e artefatos (RF);
- possuir um editor gráfico de fluxogramas de atividades do processo (RF);
- possibilitar o envio de *e-mail* de notificação sobre o *status* das atividades (RF);
- viabilizar a criação de regras para roteamento das atividades (RF);
- possuir uma caixa de entrada onde serão listadas as atividades que determinado usuário do sistema possui (RF);
- possibilitar a criação de modelos que facilitam a reutilização e customização dos processos (RF);

Requisitos

- permitir a “instanciação” dos modelos (RF);
- ser implementado na linguagem Java, utilizando ambiente Eclipse (RNF)*;*
- utilizar arquitetura *Model View Controller* (MVC) (RNF);
- ser disponibilizado em ambiente *web*, através do servidor Apache Tomcat (RNF);
- utilizar banco de dados Apache Derby (RNF);
- utilizar o *framework* de persistência de objetos Hibernate (RNF);
- utilizar o *framework* JGraph para criação do editor de fluxogramas.

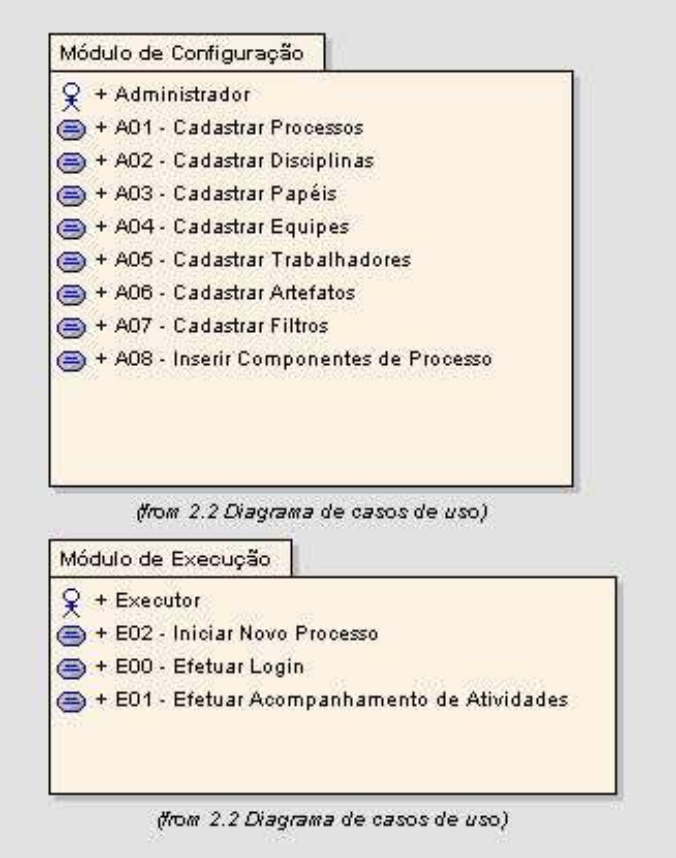
Especificação - Ferramentas

- Enterprise Architect, para criação dos casos de uso e diagramas;
- Macromedia Dreamweaver, para criação dos protótipos.

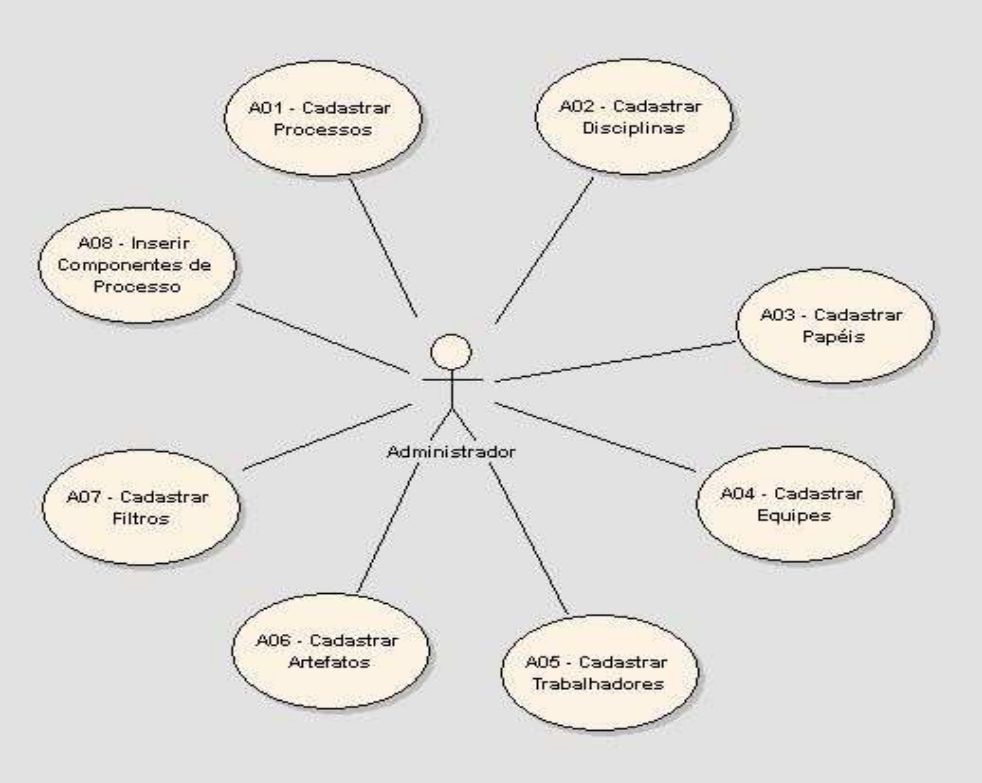
Especificação – Casos de Uso

- Os casos de uso desenvolvidos neste trabalho foram divididos nos seguintes pacotes:
  - Módulo Configurador;
  - Módulo Executor.

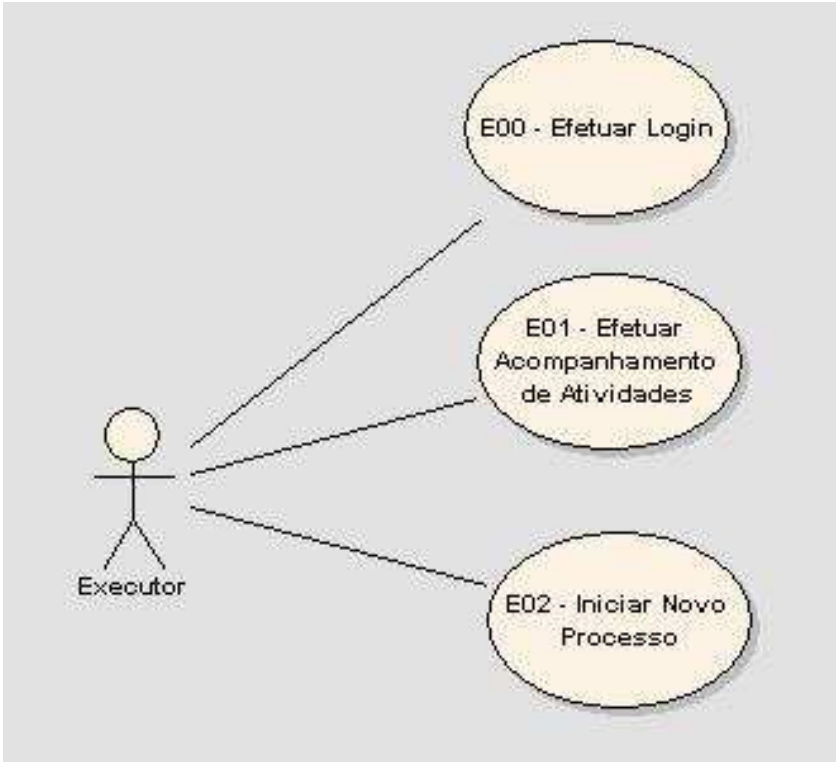
Pacotes


Módulo de Configuração
+ Administrador
+ A01 - Cadastrar Processos
+ A02 - Cadastrar Disciplinas
+ A03 - Cadastrar Papéis
+ A04 - Cadastrar Equipes
+ A05 - Cadastrar Trabalhadores
+ A06 - Cadastrar Artefatos
+ A07 - Cadastrar Filtros
+ A08 - Inserir Componentes de Processo
(from 2.2 Diagrama de casos de uso)
Módulo de Execução
+ Executor
+ E02 - Iniciar Novo Processo
+ E00 - Efetuar Login
+ E01 - Efetuar Acompanhamento de Atividades
(from 2.2 Diagrama de casos de uso)

Casos de Uso do Módulo de Configuração



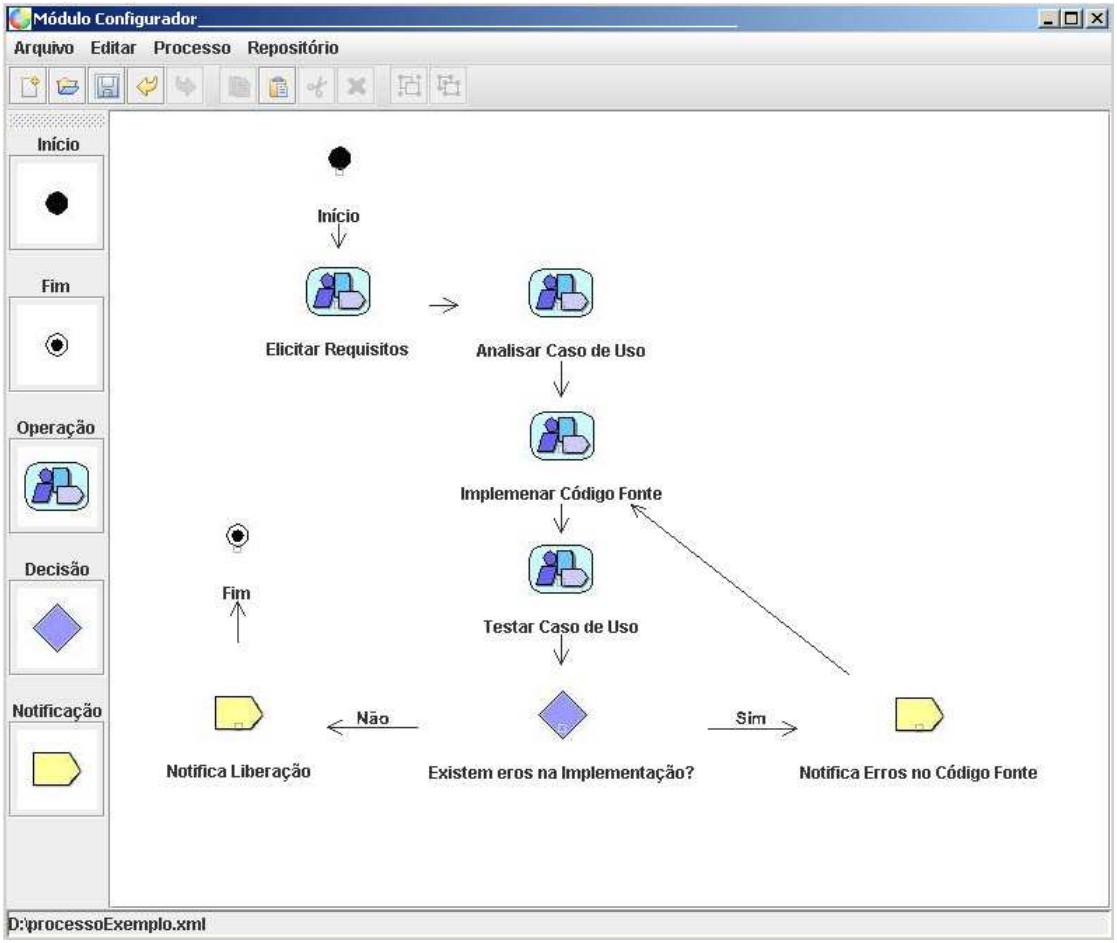
Casos de Uso do Módulo de Execução



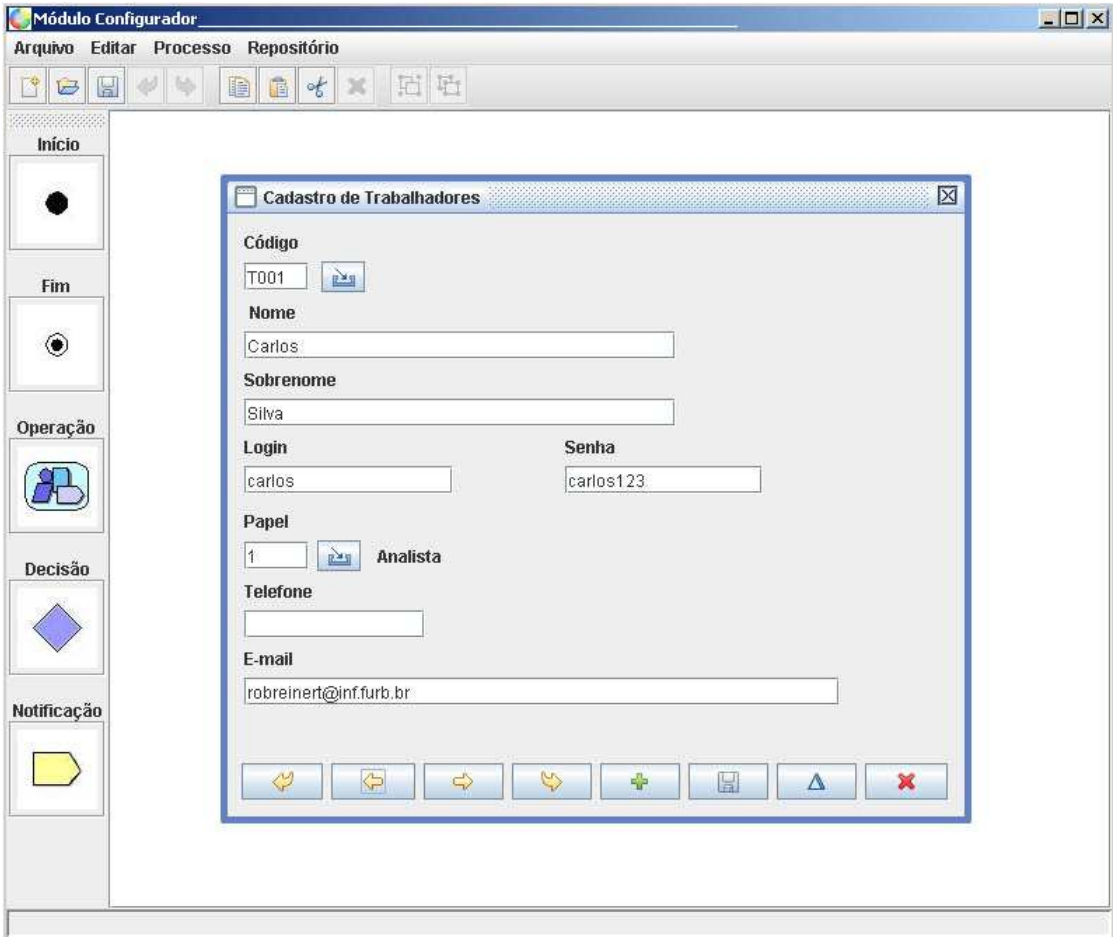
Módulo de Configuração

- Concentra os casos de uso relacionados à configuração dos processos de software;
- É responsável pelo cadastramento dos trabalhadores, artefatos e atividades;
- Contém o editor de fluxogramas, onde será possível interligar os componentes de um processo e ativá-lo para o módulo de execução.

Tela - Editor


Módulo Configurador
Arquivo Editar Processo Repositório
Início
Fim
Operação
Decisão
Notificação
Início
Elicitar Requisitos
Analisar Caso de Uso
Implemenar Código Fonte
Testar Caso de Uso
Fim
Não
Sim
Notifica Liberação
Existem eros na Implementação?
Notifica Erros no Código Fonte
D:\processoExemplo.xml

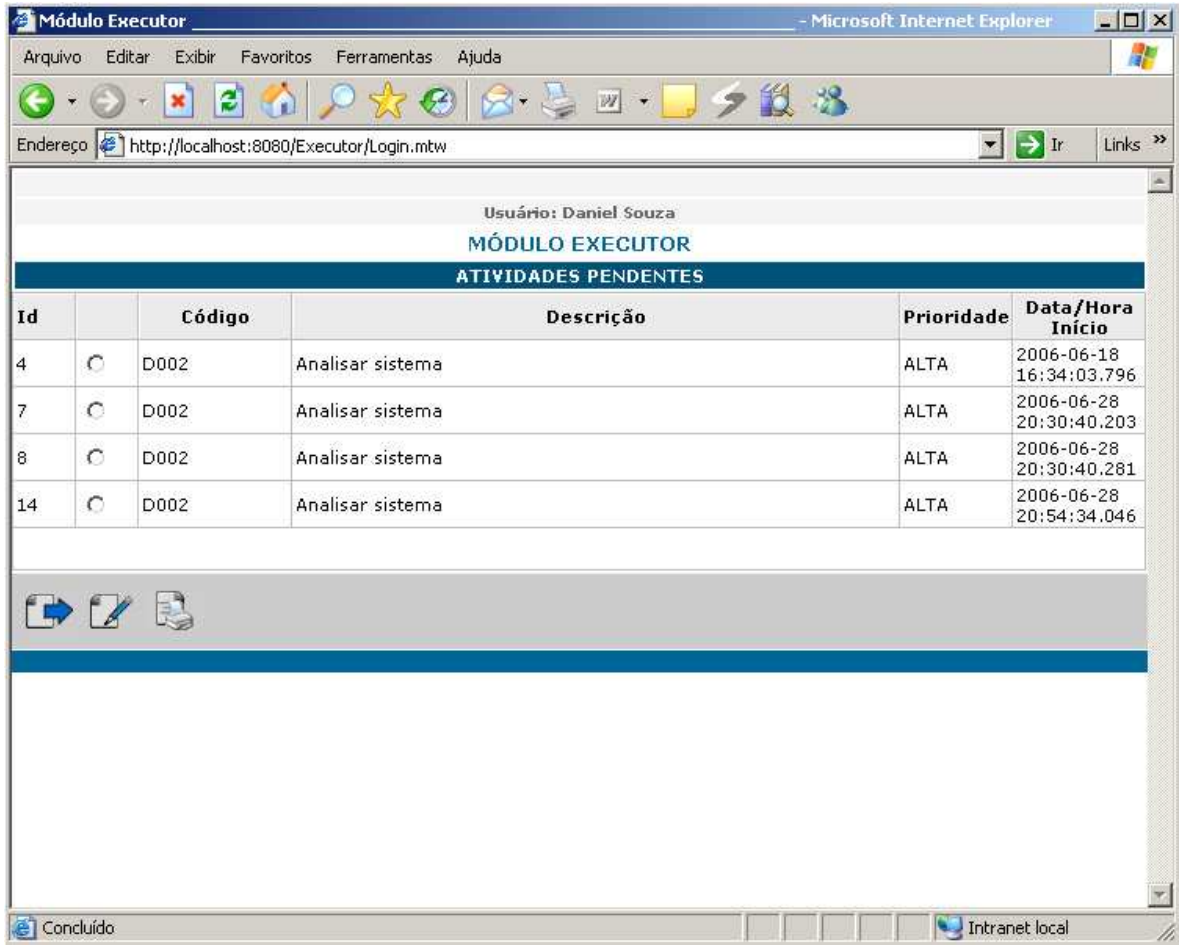
Tela – Cadastro de Trabalhadores



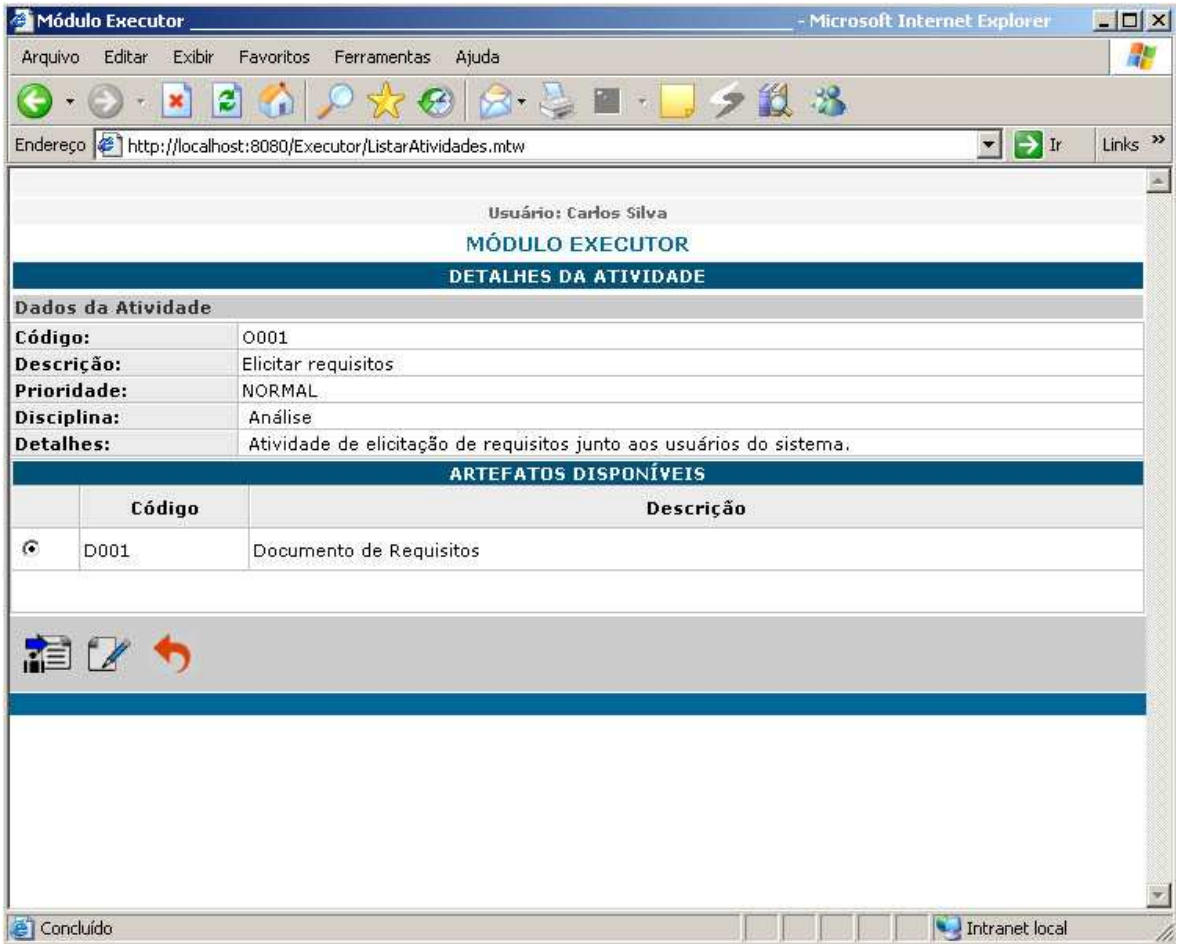
Módulo de Execução

- Concentra os casos de uso relacionados à execução do processo em si;
- É responsável pela listagem das atividades pendentes de cada usuário do sistema
- Instancia novas execuções de processo;
- Edição dos atributos dos artefatos.

Tela – Atividades Pendentes



Tela – Detalhes da Atividade



Implementação

Técnicas e Ferramentas Utilizadas

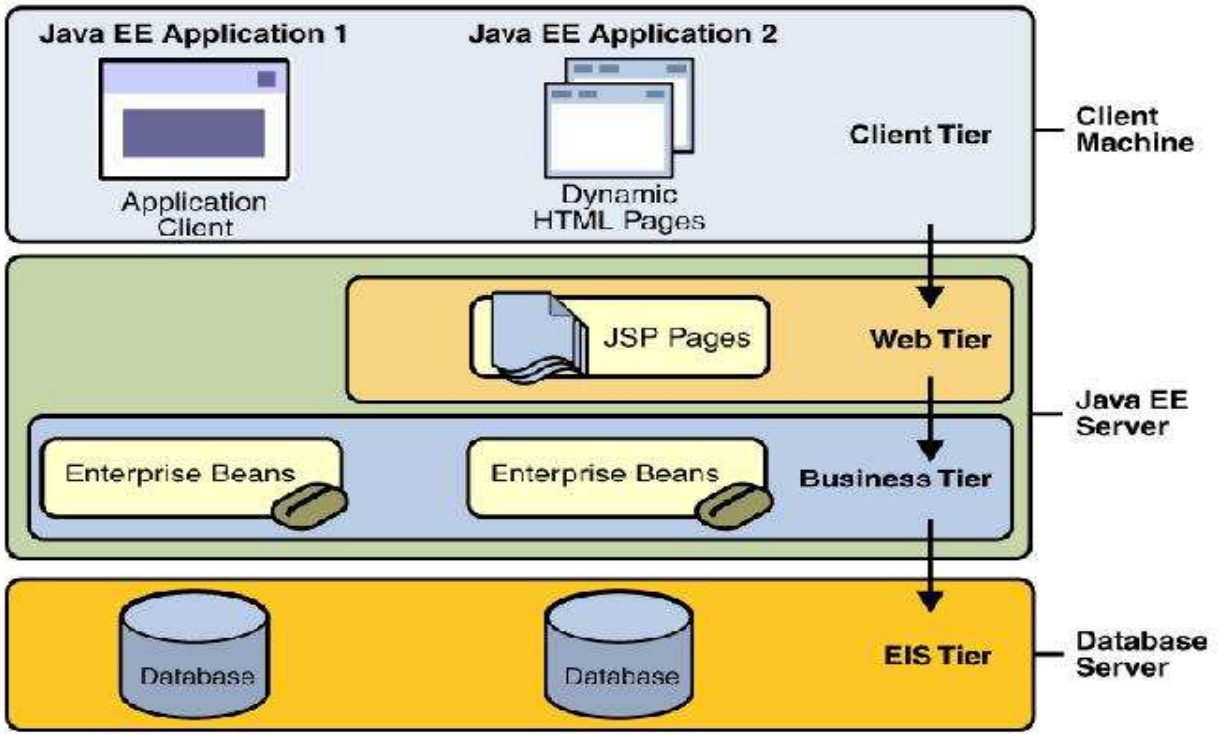
Arquitetura MVC

- Separa geração de conteúdo da apresentação do conteúdo;
- Mais flexíveis e fáceis de manter e de estender;

Arquitetura MVC

- *Model:* Camada que contém a lógica de negócios, os objetos que serão manipulados;
- *View:* Camada que possui a interface com o usuário, responsável pela exibição do conteúdo ao usuário;
- *Controller:* Camada que controla o fluxo da aplicação, fica entre a *model* e a *view* – Mentawai.

Arquitetura MVC



Camada de Apresentação

```
<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">
```

Camada de Negócio

```java
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;
    }
```

Camada de Persistência

```
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;
    }
```

Plataforma Java

- Desenvolvida pela Sun Microsystems;
- As aplicações podem ser executadas em diferentes sistemas operacionais;
- Versão utilizada: Java 5.

Framework JGraph

- manipulação de objetos gráficos;
- biblioteca de classes e métodos altamente documentada;
- utiliza 100% linguagem Java;
- baseado no padrão MVC;
- possui licença livre.

Framework *web* Mentawai

- *Framework web* simples, flexível 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.

Framework de persistência Hibernate e geração de código com XDoclet

- solução para persistência de objetos relacionais;
- possibilita a recuperação de 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.
- para facilitar ainda mais o trabalho, foram utilizados os Xd*oclets,* comentários nas classes Java que em conjunto com o Hibernate geram todas as tabelas do banco de dados sem que o desenvolvedor execute nenhum comando SQL (*Structured Query Language*).

Banco de dados Apache Derby

- subprojeto do projeto Apache, é um banco de dados relacional implementado totalmente em Java;
- é baseado nos padrões Java, JDBC e SQL;
- suporta os modos de utilização “embacardo” (dentro da aplicação) e cliente/servidor;
- é de fácil instalação, distribuição e uso;
- possui licença livre.

Eclipse e JUnit

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

Operacionalidade da Implementação

Diagrama de Classes

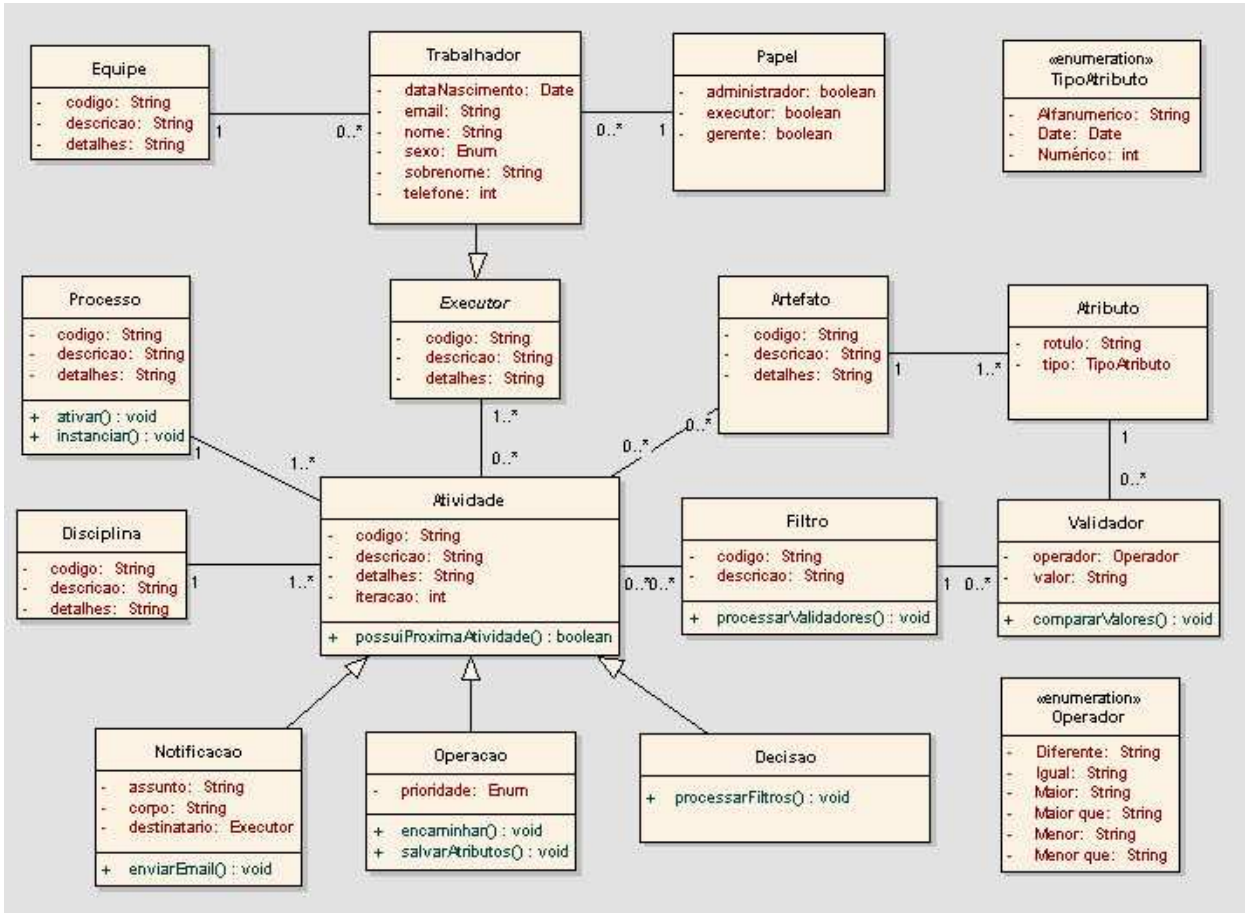
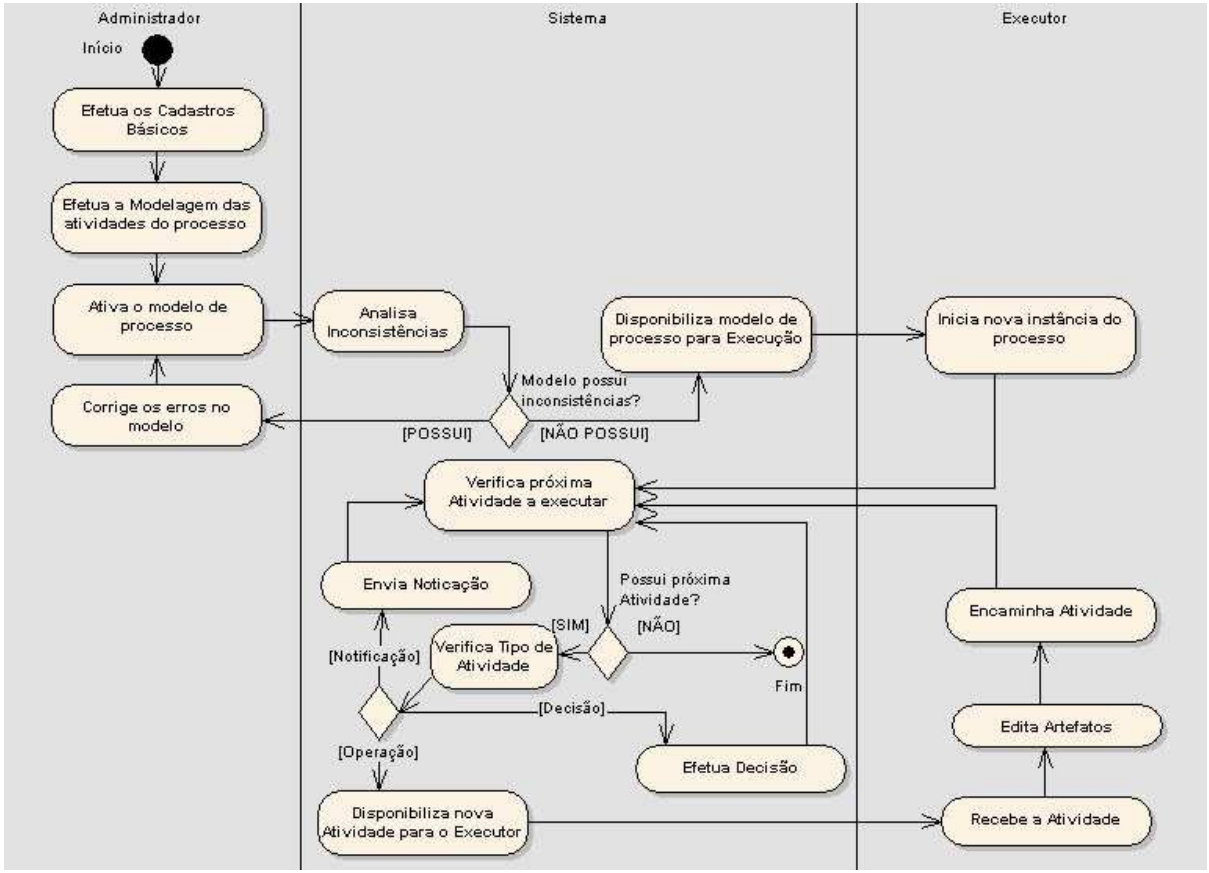


Diagrama de Atividades



Resultados e discussão

- No trabalho desenvolvido por Bork (2003) e Hauck(2004), foi apresentada a implantação e customização de um processo de software baseado em processos e normas existentes. Esta implantação, no entanto, não envolve um processo automatizado de *workflow*, mas sim a utilização de formulários e documentos 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.
- Não houveram testes em situações reais para obtenção de dados comparativos entre a processos manuais e processos implantados com a utilização desta ferramenta.

Conclusão

- A utilização de sistemas de *workflow* em processos de software apresenta-se como um item fundamental para a organização do fluxo de trabalho;
- Os objetivos definidos na proposta deste trabalho foram alcançados, foi possível efetuar a criação de modelos e executá-los da ferramenta;

Conclusão

- Muitas das tecnologias utilizadas tiveram que ser estudadas e pesquisadas durante o decorrer deste trabalho, sendo que e as mesmas auxiliaram e facilitaram bastante o seu desenvolvimento;
- A aderência das ferramentas e tecnologias utilizadas no projeto foi total. Foram utilizadas tecnologias altamente conhecidas da comunidade de software livre e que permitem a continuidade deste trabalho.

Extensões

Como possíveis extensões ao trabalho desenvolvido até o presente momento temos:

- possibilitar a criação de sub-processos no módulo configurador para permitir um melhor acoplamento entre os processos;
- permitir a inserção de anexos aos artefatos, tornando-os mais flexíveis com as necessidades de um sistema de *workflow*;
- 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;

Demonstração da implementação

Obrigado!