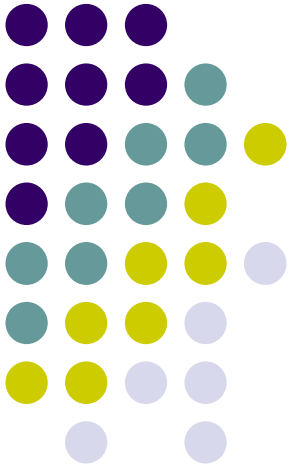
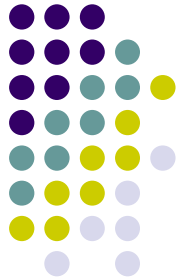
Sistema de *Help Desk* e Controle de Chamados Baseado em *Workflow*

Cristian Paulo Prigol

Marcel Hugo

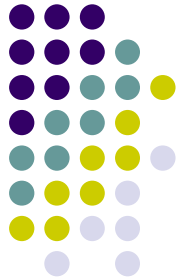


Seqüência da Apresentação



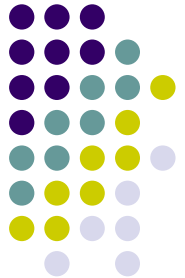
- Introdução e Objetivos
- *Workflow*
- *Sistema Antigo e Sistema implementado*
- *JBPM*
- *Especificação do sistema*
- *Implementação e Operacionalidade*
- *Conclusão e Extensões*

Introdução



- O computador é a principal ferramenta de trabalho e seu perfeito funcionamento é primordial
- Sistemas de *Help Desk* ajudam a coordenar os processos da equipes de suporte
- *Workflow* auxilia no gerenciamento de processos

Introdução



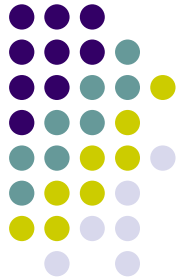
- *Java Business Process Management* (JBPM): *Framework* de *workflow* proposto pela JBoss

- Seção de Apoio ao usuário da FURB (APUS)

Objetivos

- **Desenvolver um sistema de *Help Desk* para o APUS utilizando a tecnologia de *workflow* .**
  - Possibilitar que o sistema gerencie o andamento dos chamados
  - Disponibilizar ao gestor de TI o acompanhamento dos chamados e do desempenho de seus técnicos
  - Melhorar o fluxo de trabalho no setor implantando o *workflow*
  - Desenvolver um sistema de *Help Desk* utilizando o *framework* JBPM

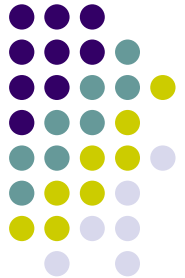
Workflow



- Workflow: automatização de um processo de negócio, durante o qual são passados documentos, informações ou tarefas entre participantes de acordo com regras processuais. (Hollingsworth, 1995, p. 06)

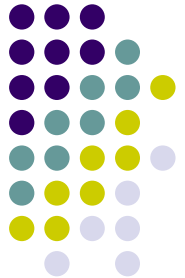
- Início das pesquisas na década de 70

Workflow



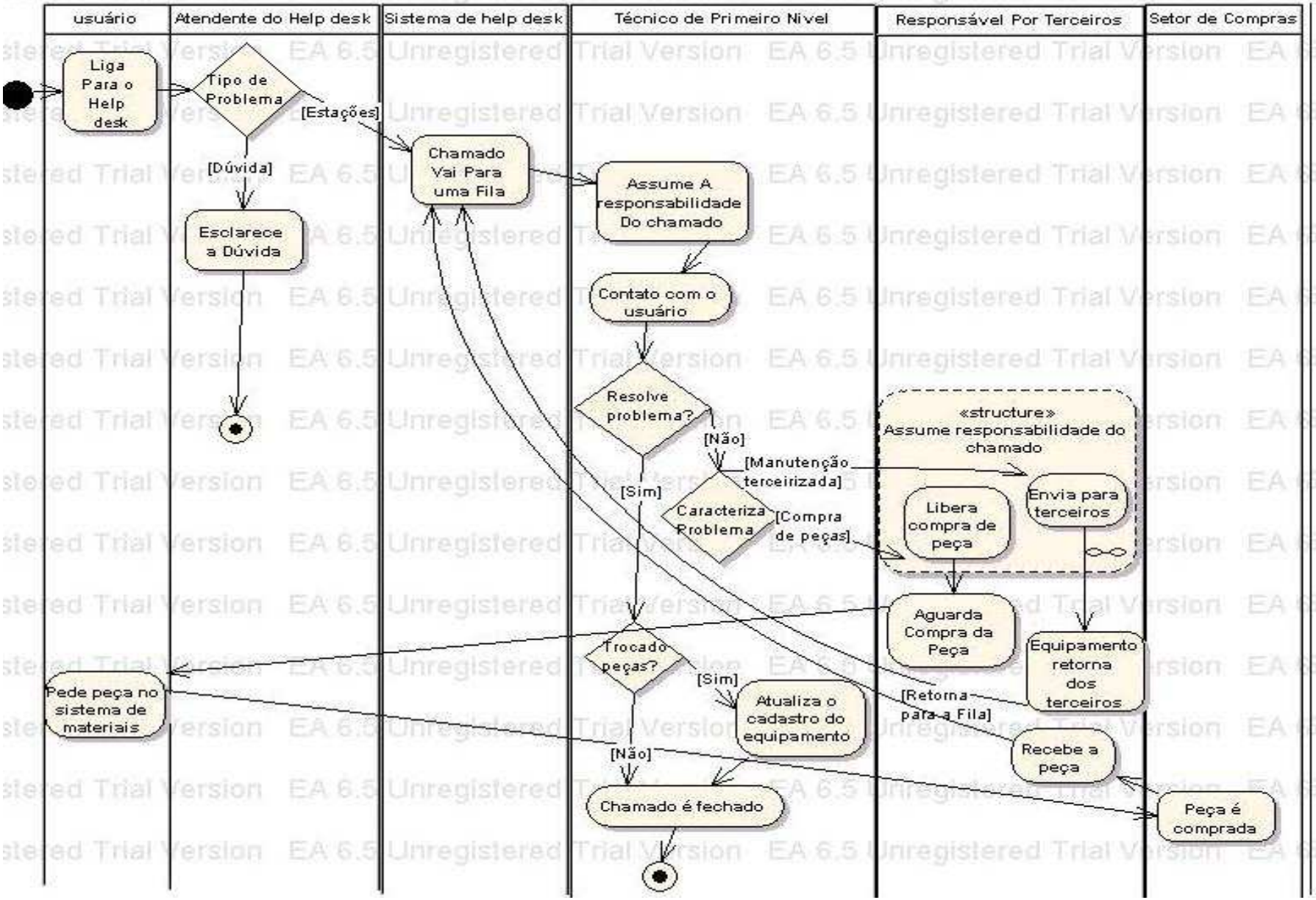
- Objetivo: aumentar a eficiência de processos de negócio
- Representa processos através fluxo de trabalho
- Aumento da produtividade e diminuição do ciclo do negócio

Sistema Antigo

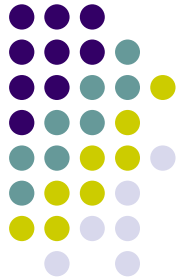


- Sistema desenvolvido por terceiros
- Somente um técnico por chamado
- Não existe cadastro de peças
- Não existe *feedback* ao/do usuário

Fluxo Antigo de trabalho

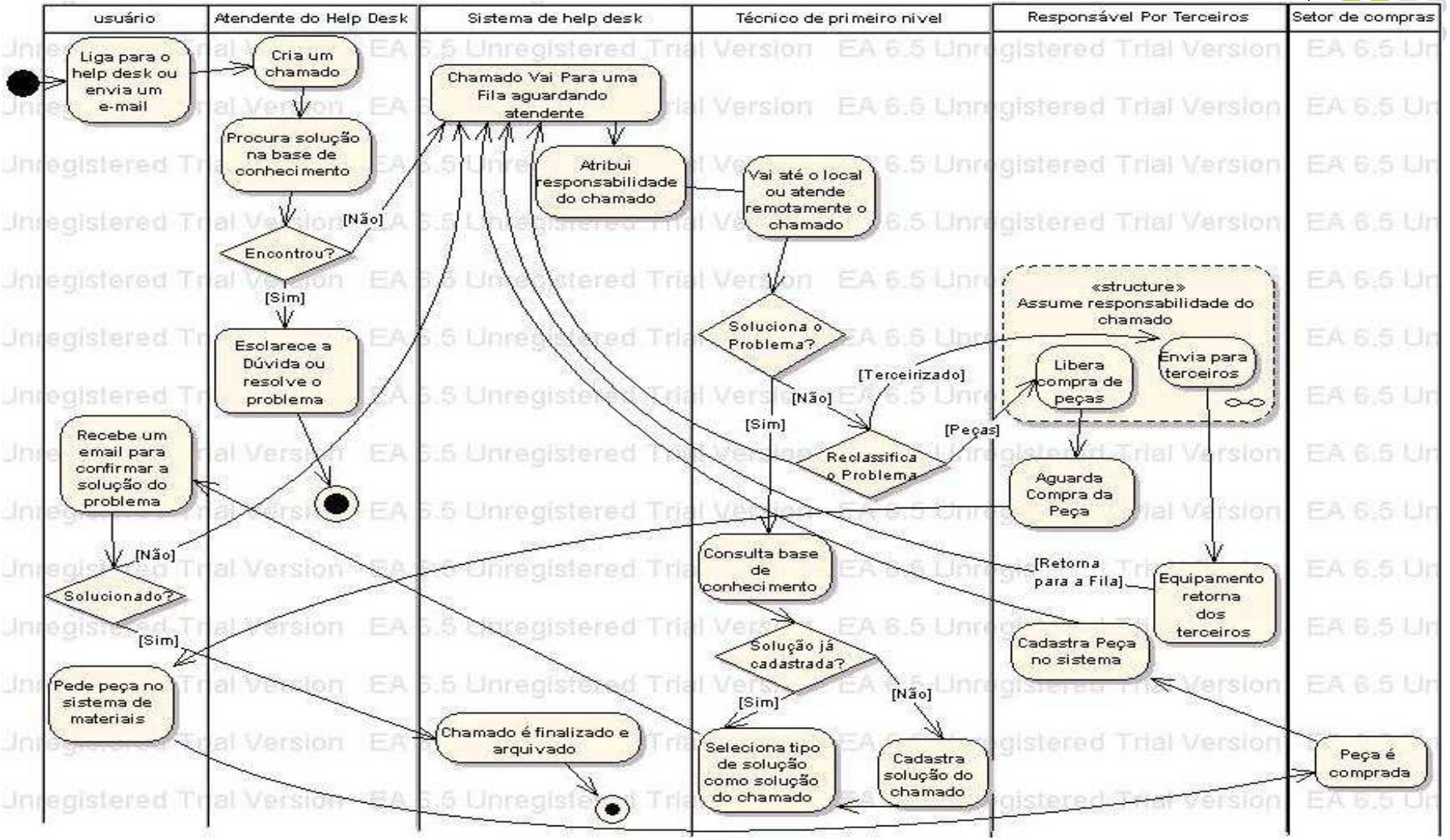


Sistema Implementado

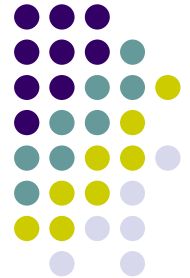


- Chamado classificado como processo e dividido em tarefas
- Tarefas vinculadas automaticamente às pessoas através de e-mail
- Fluxo segue rotas do processo
- Base de usuários vinculada à base principal da FURB
- Implementado cadastro de hardwares e peças

Novo Fluxo de trabalho

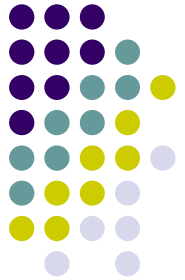


Principais Requisitos Funcionais



- O sistema deverá enviar um e-mail atribuindo o próximo chamado da fila ao técnico que estiver disponível.
- Enviar um e-mail para que o usuário confirme a resolução do chamado.
- Permitir consulta do status dos seus chamados.
- Permitir ao técnico manter um cadastro de hardware.
- Possibilitar a classificação do chamado como problema de hardware, software, equipamento terceirizado, servidores.
- O sistema deverá permitir ao administrador manter cadastro de privilégios dos usuários.
- O sistema deverá possibilitar cadastro de chamados pelo atendente.

Requisitos Não Funcionais



- Deverá ser implementado na linguagem Java padrão J2EE
- Deverá utilizar a base de usuários já existente nos sistemas da FURB.
- Deverá dividir os chamados em tarefas.
- Deverá utilizar o *workflow Engine JBPM.*
- Deverá utilizar o banco de dados *Oracle.*

Diagrama de Casos de Uso

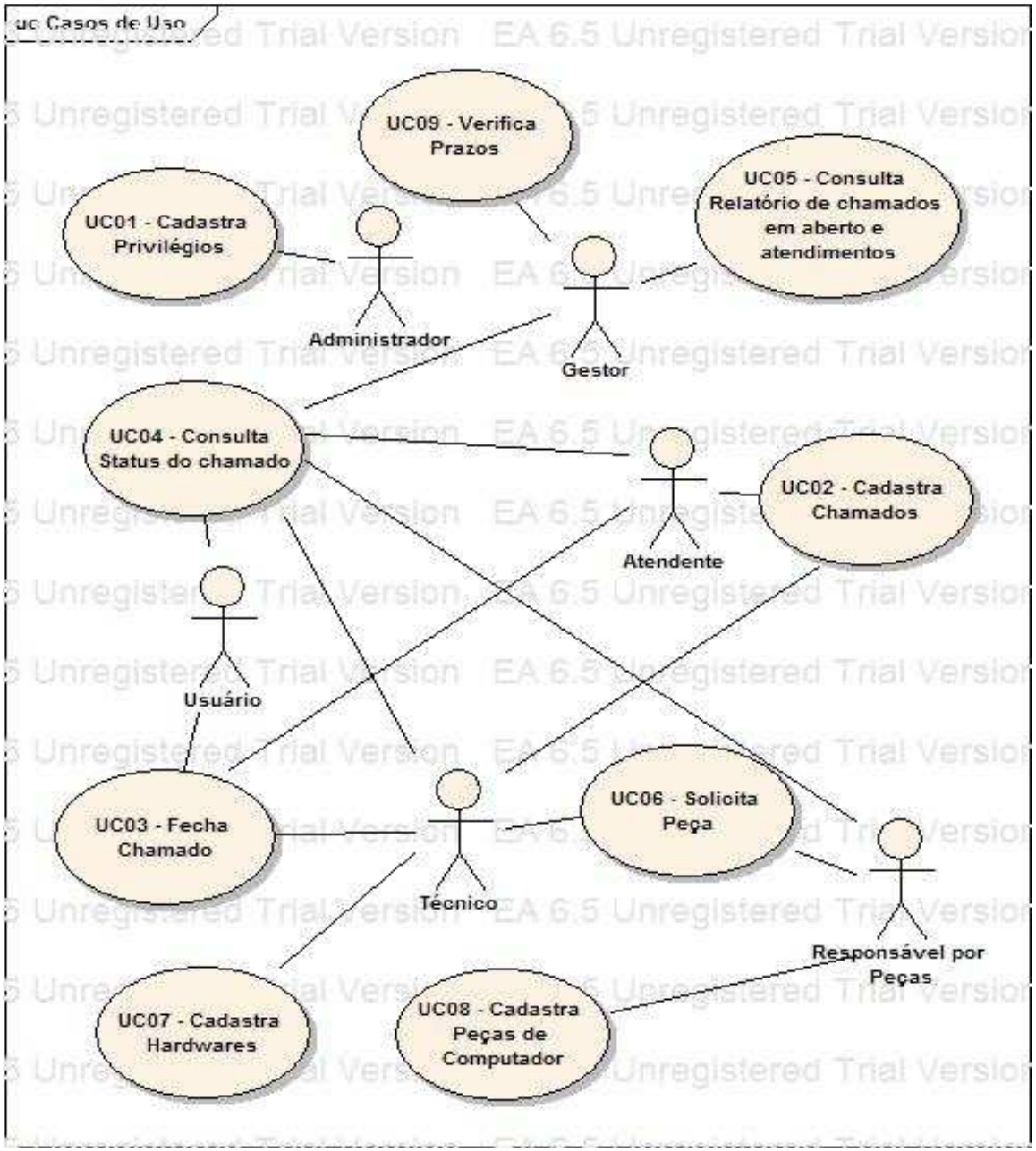
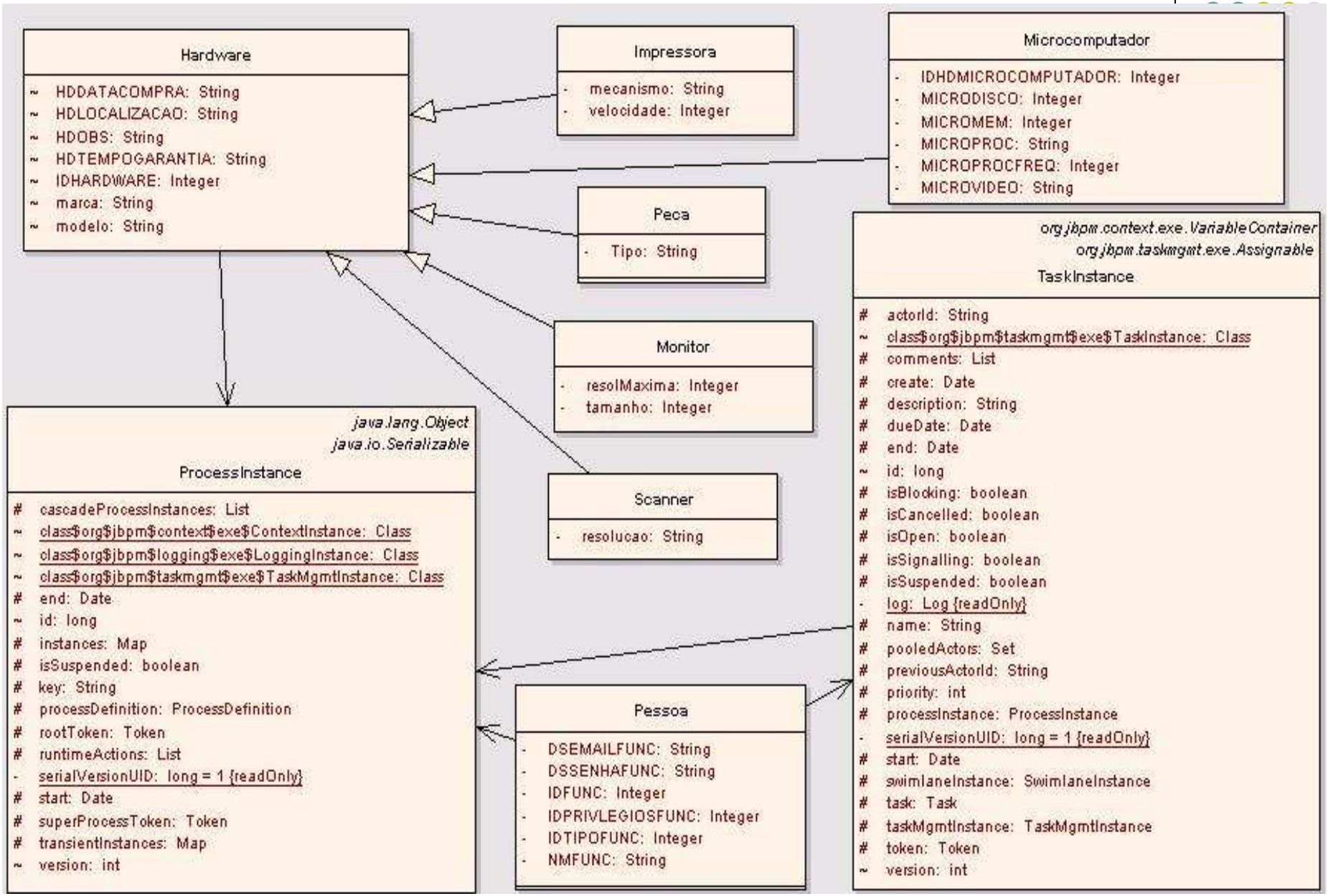
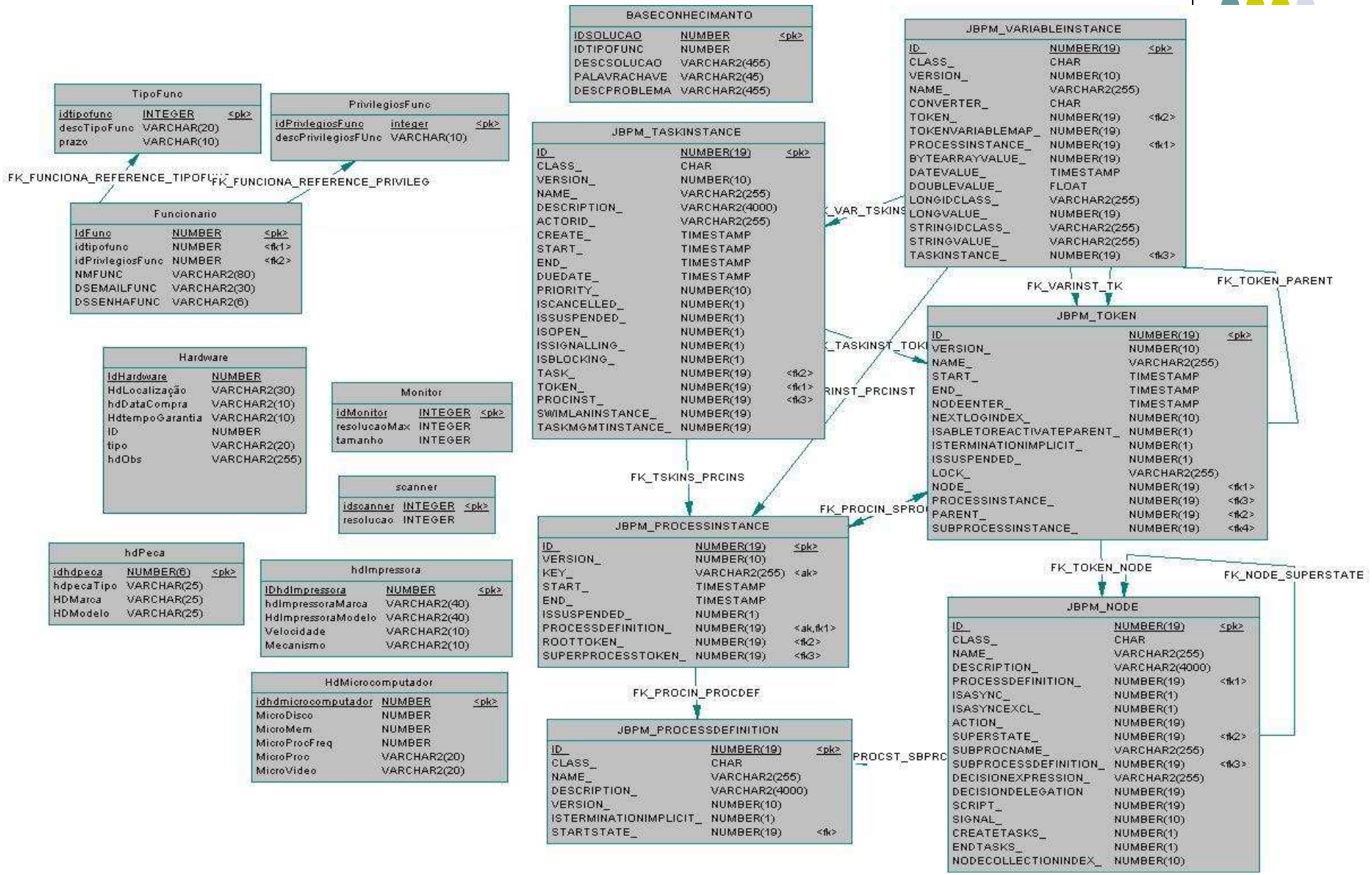


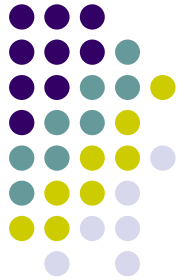
Diagrama de Classes



Modelo Entidade Relacionamento



Ferramentas Utilizadas

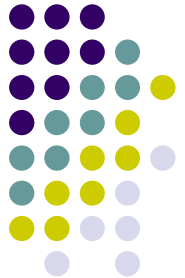


- *Framework* JBPM
- *Framework* Hibernate
- Linguagem Java padrão J2EE
- Servidor de aplicações JBoss
- Banco de dados Oracle
- Ambiente Eclipse

JBPM

- Ferramenta para modelagem de processos de negócio.
- Utiliza padrões de workflow desenvolvidos em pesquisas acadêmicas do Prof. Wil Van der Aalst - Universidade de Tecnologia de Eindhoven na Holanda.
- Classes Java para o gerenciamento de processos.

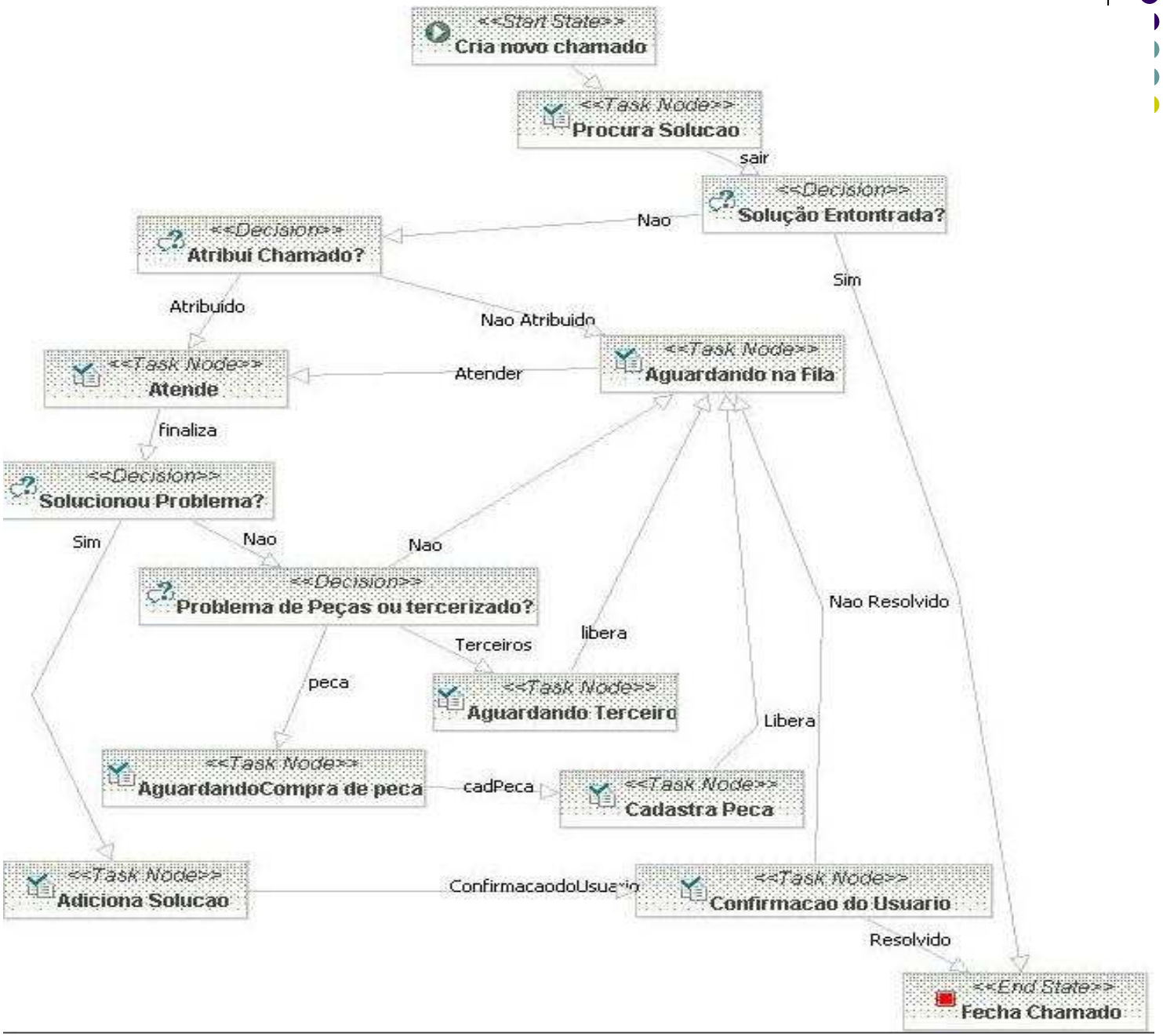
JBPM



- Modelagem feita no Eclipse.

- Gera um arquivo JPDL com a definição de processo.

Workflow no JBPM

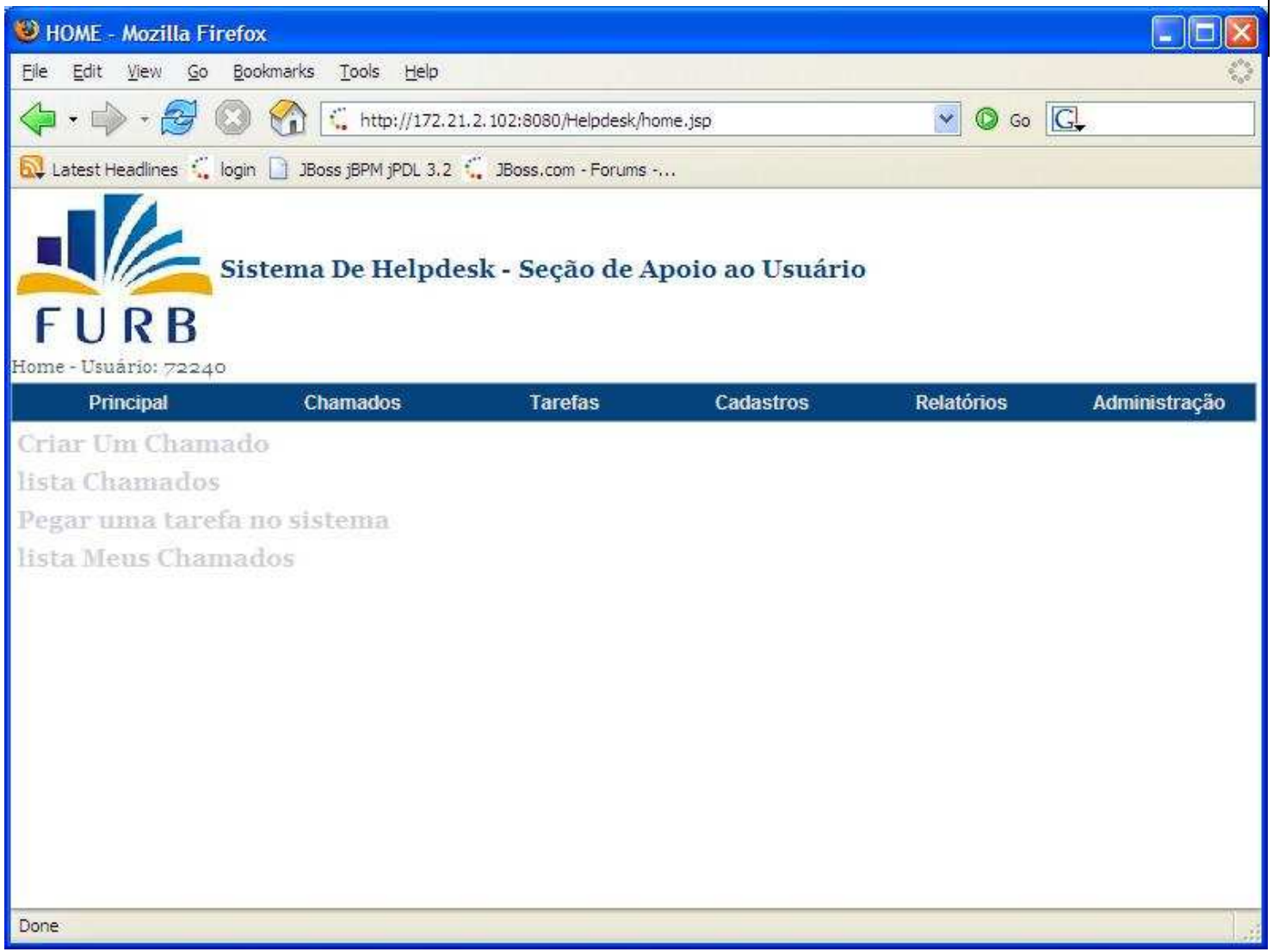


Exemplo criação de chamado

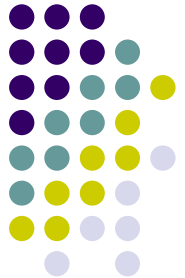
```
ProcessDefinition pd = gs.findLatestProcessDefinition("Chamado");
ProcessInstance pi = new ProcessInstance(pd);

TaskInstance taskInstance = pi.getTaskMgmtInstance()
        .createStartTaskInstance();
/*
 * Cria um map com as variaveis que vão fazer parte do processo As
 * variaveis estão na sessão do usuário
 */
Map taskVariables = new HashMap();
taskVariables.put("idSolicitante", request.getParameter("idPessoa"));
taskVariables.put("idSetor", request.getParameter("idSetor"));
taskVariables.put("idEquipamento", request
        .getParameter("idEquipamento"));
taskVariables.put("idChtipo", request.getParameter("idChtipo"));
taskVariables.put("Obs", request.getParameter("Obs"));
taskVariables.put("atendente", request.getParameter(""));
taskInstance.addVariables(taskVariables);
/*
 * O metodo close do jbpmContext presiste no banco de dados as
 * informações
 */
taskInstance.end();
jbpmContext.save(taskInstance);
```

Operacionalidade da implementação

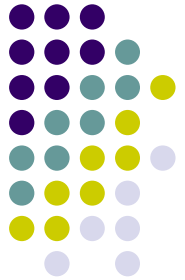


Resultados e discussão



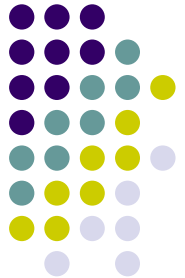
- Sistema não foi implantado devido a implantação de outro sistema
- Foi utilizado o fluxo de trabalho desenvolvido no novo sistema

Conclusões



- Permitiu o estudo do JBPM
- Permitiu o estudo de *workflow*
- Fluxo pré-determinado facilita o controle de chamados
- Permitiu aplicação dos conceitos estudados no curso
- Objetivo principal foi atingido

Extensões



- Sugestão de adequar o sistema às boas práticas do ITIL