SÁVIO GERALDO FERREIRA FRAGA

Orientador: Carlos Alberto Marques Pietrobon

EPROCESSOS: UM SISTEMA EDITOR DE PROCESSOS DE SOFTWARE

Ouro Preto

Novembro de 2010

Universidade Federal de Ouro Preto
Instituto de Ciências Exatas
Bacharelado em Ciência da Computação

EPROCESSOS: UM SISTEMA EDITOR DE PROCESSOS DE SOFTWARE

Monografia apresentada ao Curso de Bacharelado em Ciência da Computação da Universidade Federal de Ouro Preto como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação.

SÁVIO GERALDO FERREIRA FRAGA

Ouro Preto
Novembro de 2010

UNIVERSIDADE FEDERAL DE OURO PRETO

FOLHA DE APROVAÇÃO

EPROCESSOS: Um Sistema Editor de Processos de Software

SÁVIO GERALDO FERREIRA FRAGA

Monografia defendida e aprovada pela banca examinadora constituída por:

Dr. Carlos Alberto Marques Pietrobon – Orientador
PUC - Rio

M.Sc. Elton José da Silva
Universidade Estadual de Campinas

B.Sc. Sabir Ribas
Universidade Federal de Ouro Preto

Ouro Preto, Novembro de 2010

Resumo

Desenvolver software de qualidade e dentro do prazo estabelecido tem sido um dos grandes desafios nas empresas fabricantes de software. Neste contexto, as empresas mais competitivas tendem a ser aquelas que trabalham sob a ótica da melhoria contínua dos processos para aumentar a qualidade do processo de desenvolvimento e, consequentemente, aumentar a qualidade do produto final. Por esse motivo é tão importante a padronização dos processos como uma abordagem viável na introdução sistemática de qualidade. Uma das formas de se padronizar um determinado processo é através da sua descrição formal, a qual produz um modelo do processo de software. Sendo assim, este trabalho visa desenvolver um sistema web para a edição dos itens do processo de software, ou seja, criar e personalizar os atributos dos componentes do processo.

Palavras-chave: Processo de Software. Definição de Processo. Qualidade do Processo de Software. Sistema Web.

Abstract

Develop quality software within the prescribed period has been a major challenges in business software makers. In this context, firms more competitive tend to be those who work from the perspective of continuous process improvement to increase the quality of the development process and thereby increase the final product quality. For this reason it is so important to the standardization of processes as a viable approach in the introduction of systematic quality. One way to standardize a process is determined by its formal description, which produces a software process model. Thus, this work aims to develop a web system for editing the items in the software process, or create and customize the attributes of process components.

Keywords: Software Process. Process Definition. Quality Software Process. Web System

Dedico este trabalho a todos que de certa forma ajudaram para a realização do mesmo, principalmente aos meus pais, Celso e Célia, e irmãos que sempre me deram todo carinho e apoio ao qual necessitei.

Agradecimentos

A Deus, por estar sempre do meu lado me guiando pelo caminho certo.

À minha família que sempre esteve presente e me deu muita força para continuar lutando.

Aos amigos pelo apoio e gentilezas prestadas nas horas de necessidade.

Ao meu orientador por ter acreditado na conclusão deste trabalho e auxiliado para que isto aconteça.

Sumário

Lista de Figuras

Capítulo 1

Introdução

1.1 Contextualização

Desenvolver software de qualidade e dentro do prazo estabelecido tem sido um dos grandes desafios dentro das empresas fabricantes de software. Um caminho que contribui para que uma organização vença estes desafios é investir na melhoria da qualidade e da produtividade. Como a melhoria da qualidade do produto final é tipicamente obtida através da melhoria do próprio processo produtivo, melhorar o processo de software torna-se o grande desafio das empresas.

Assim, a partir da década de 1990, houve uma grande preocupação com a modelagem e melhorias no processo de software. Devido a isto, abordagens importantes como as normas de qualidade *ISO 9000* e a *ISO/IEC 12207*, o modelo *Capability Maturity Model (CMM)* e o modelo *Software Process Improvement and Capability Determination (SPICE)* foram criadas e revisadas. Tais aborbordagens sugerem que melhorando o processo de *software*, pode-se melhorar a qualidade dos produtos.

Uma forma de analisar e amadurecer o processo de desenvolvimento de *software* é através da sua definição, a qual consiste de um modelo de processo de *software*, esta descrição representa o processo de *software* sob a ótica de seu funcionamento, mas não apresentam como estes processos devem ser definidos, ficando esta responsabilidade para as empresas fabricantes de *software*.

A descrição formal de um processo de *software* é a atividade que permite que o mesmo seja analisado, compreendido e automatizado (executado). Para isso, o processo de *software* deve ser definido, ou seja, ter uma documentação para detalhar o que é feito (o produto), quando é feito (os passos), por quem é feito (os agentes), o que é usado (artefatos) e o que é produzido (os resultados) durante o desenvolvimento do produto. Com um processo definido, todos seguem um padrão e produzem produtos padronizados, e desta forma, qualquer um que conhecer o padrão consegue entender o processo.

A Norma *ISO/IEC 12207* estabelece uma estrutura comum, um padrão, para os processos

de ciclo de vida de *software* desde a concepção de idéias até a descontinuação do *software* com uma terminologia bem definida e é composta de processos, atividades e tarefas que servem para ser aplicada durante a aquisição de um sistema que contém *software*, de um produto de *software* independente ou de um serviço de *software*, e durante o fornecimento, desenvolvimento, operação e manutenção de produtos de *software*.

A dificuldade em definir processos encontra-se na ausência de um processo de *software* possível de ser genericamente aplicado. Os processos variam porque são diferentes os tipos de sistemas, os domínios de aplicação, as equipes, as organizações e as próprias restrições de negócio, tais como, cronograma, custo, qualidade e confiabilidade. Assim, é necessária uma abordagem flexível e configurável para a definição de processos, de modo a facilitar a adaptação de processos padrão às necessidades específicas de cada projeto. Nesse contexto, torna-se essencial prover ferramentas para automatizar as diversas tarefas que compõem o processo de desenvolvimento de *software* para que se possa obter um controle mais rígido e adaptado às necessidades da empresa.

1.2 Motivação e Justificativa

Muitas empresas não possuem, ou quando possuem é informal, um processo de *software* definido. Sendo assim, surge a necessidade de se desenvolver uma ferramenta para que seja feita a definição formal e de forma automatizada dos processos de desenvolvimento de *software*. Este trabalho propõe a implementação de um sistema editor de *processos de software*, com o qual é possível definir, isto é, criar modelos para estes.

A modelagem deve ser capaz de descrever alguns aspectos necessários para a coordenação das atividades envolvidas num processo de software. O modelo muitas vezes é utilizado como um guia para que seja possível acompanhar cada passo, identificar o estado do projeto e reorganizá-lo durante este acompanhamento. Em outros casos o modelo é utilizado apenas como documentação, isto é, como uma forma de explicitar o processo utilizado com o objetivo de conscientizar a equipe. Este último é o caso que será considerado neste trabalho.

A ferramenta proposta neste trabalho tem o intuito de trazer para uma empresa de desenvolvimento de *software* vantagens tais como melhoria da qualidade do produto (*software*), facilitar o entendimento e comunicação entre as pessoas envolvidas, fornecer orientações a equipe de projeto de forma automatizada e apoiar a gerencia e melhoria do processo de desenvolvimento de *software*. Para tal, o sistema é desenvolvido em linguagem *Java*, utilizando apenas ferramentas de código aberto e apresenta uma interface *Web* para os usuários.

1.3 Objetivos

1.3.1 Objetivo geral

O objetivo deste trabalho é desenvolver um sistema capaz de modelar processos de software, ou seja, criar e personalizar os atributos dos itens do processo. O principal objetivo é aumentar o nível de automação fornecido na reutilização de processos de software, apoiando a modelagem de processos padrão de uma organização. A ferramenta em desenvolvimento será uma Aplicação de *Internet* Rica (da sigla em inglês *RIA - Rich Internet Application*), ou seja, um aplicativo *web* que tem características e funcionalidades de *softwares* tradicionais do tipo *Desktop*.

1.3.2 Objetivos específicos

Os objetivos específicos a serem alcançados neste trabalho são os seguintes:

- Fazer uma revisão de literatura sobre processos de *software*;
- A partir da revisão feita, fazer o levantamento dos requisitos necessários para a implementação do sistema;
- Modelar o banco de dados para a aplicação de acordo com o levantamento de requisitos feitos no passo anterior;
- Implementar o sistema utilizando *Java* para *web*.

1.4 Organização do Trabalho

Para atingir esses objetivos neste trabalho há, além deste capítulo que apresenta a Introdução, mais cinco capítulos.

O Capítulo 2 - Conceitos Teóricos - aborda os conceitos de *Processos de Software*, com o objetivo de contextualizar a realidade em que o sistema desenvolvido está inserido.

No Capítulo 3 - Análise e Projeto - é detalhado questões referentes à modelagem do sistema aqui desenvolvido.

O Capítulo 4 - Implementação - detalha as ferramentas e técnicas escolhidas para a implementação do sistema em si, que deverá ser implementado utilizando software livre.

O Capítulo 5 - Funcionamento do Sistema - descreve os requisitos necessários para executar o sistema e descreve o funcionamento do sistema desenvolvido.

Finalmente, no Capítulo 6 - Conclusões e Trabalhos Futuros - são feitas as considerações finais sobre o trabalho aqui desenvolvido, apresentando suas contribuições e propostas para trabalhos futuros.

Capítulo 2

Conceitos Teóricos

Este Capítulo trata dos fundamentos conceituais sobre os quais o trabalho apresentado nesta monografia foi construído. Desta forma são apresentados a seguir os tópicos que orientam no embasamento para a formalização do estudo aqui aplicado.

2.1 Processos de Software

Sabe-se que um dos focos da *Engenharia de Software* é desenvolver sistemas com qualidade, dentro dos prazos estabelecidos e sem a necessidade de alocação de mais recursos. Para que tal objetivo seja alcançado, o foco não deve estar apenas nos produtos gerados, mas também no seu processo de desenvolvimento. Pressman (2009) cita que a engenharia de *software* é uma tecnologia em camadas, onde os processos de *software* formam a base para o controle gerencial de projetos de *software* e estabelecem o contexto no qual os métodos técnicos são aplicados, os produtos de trabalho são produzidos, os marcos são estabelecidos, a qualidade é assegurada e as modificações são adequadamente geridas.

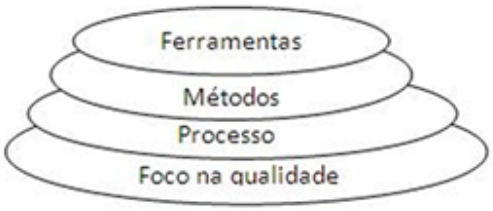

Figura 2.1: Engenharia de software em camadas.

Processo de Software pode ser definido como um conjunto de atividades realizadas para construir software, levando em consideração os produtos sendo construídos, as pessoas envolvidas, e as ferramentas com as quais trabalham. Desta forma pode-se dizer que um processo de desenvolvimento de *software* envolve atividades, métodos, técnicas e práticas com o objetivo

de orientar as pessoas envolvidas na produção de software, permitindo que um produto seja criado em conformidade com os padrões da organização.

Todo trabalho importante realizado nas empresas faz parte de algum processo (Graham e LeBaron, 1994). Mesmo nos casos em que os processos não foram definidos e mapeados, o trabalho segue uma série de passos que podem ser interpretados como processo. Da mesma forma, os processos existentes nas empresas sempre são orientados ao fornecimento de um produto ou serviço. Por isso, é importante para toda empresa possuir um processo claramente documentado, uma vez que, com um processo definido, todos seguem um padrão e produzem produtos padronizados, e desta forma, qualquer um que conhecer o padrão consegue entender o processo.

Não existe processo de *software* certo ou errado. Dependendo da sua aplicação, ambiente e objetivo, o uso de um processo específico pode ser vantajoso ou não. Elementos do projeto como, por exemplo, a tecnologia utilizada, os paradigmas adotados no desenvolvimento, o tamanho do produto, os métodos utilizados entre outros influenciam a aplicabilidade do processo de software ao caso específico.

Um processo padrão de uma organização é a definição de suas operações num âmbito organizacional que servirá como ponto de partida para a definição de processos específicos para os projetos de software (Bertollo e Falbo, 2003). O processo padrão pode ser aplicado a qualquer projeto da organização. Por isso, vale ressaltar que o processo padrão não se atem às características e detalhes inerentes a cada projeto. Sendo assim, é necessário instanciá-lo para cada projeto especificamente. Podemos dizer que um processo de software *padrão* visa atingir os seguintes objetivos:

Eficácia: Um processo eficaz deve ajudar a produzir o produto certo. Não importa quão elegante e bem escrito o *software*, nem a rapidez com que é produzido. Se não for o que o cliente queria, ou necessita, não é bom.

Manutenibilidade: Por melhor que seja o programador, as coisas ainda vão dar errado com o *software*. Requisitos mudam frequentemente. Em qualquer caso, podemos querer reutilizar elementos do *software* em outros produtos. Nada disto se torna mais fácil se, quando um problema é descoberto, e quem escreveu isso deixou a empresa na semana anterior, ou pior, "Alguém sabe quem escreveu esse código?". Uma das metas de um bom processo é expor aos envolvidos nos processos de tal forma que sua intenção seja clara. Então, pode-se rápida e facilmente encontrar e corrigir falhas ou descobrir onde fazer mudanças.

Previsibilidade: Qualquer desenvolvimento de produtos precisa ser planejado, e os planos são utilizados como base para a alocação de recursos: tempo e pessoas. É importante prever com precisão quanto tempo levará para desenvolver o produto. Isso significa estimar com precisão quanto tempo levará para a produção de cada parte dele. Um bom processo vai nos ajudar a fazer isso. O processo ajuda a estabelecer as etapas de desenvolvimento.

Repetibilidade: Se um processo é desenvolvido para o trabalho, ele que deve ser replicado

em projetos futuros. Processos *Ad-hoc* raramente são replicáveis, a menos que a mesma equipe esteja trabalhando no novo projeto. Mesmo que a a equipe não seja alterada de um projeto para outro é difícil manter o trabalho exatamente como foi feito em um projeto anterior. Uma questão intimamente relacionada, é o processo de reutilização. É um desperdício enorme para cada projeto produzir um processo do zero. É muito mais rápido e mais fácil de adaptar um processo existente.

Qualidade: Um dos objetivos de um processo definido é permitir que os engenheiros de *software* garantam um produto de alta qualidade. O processo deverá fornecer uma ligação clara entre os desejos de um cliente e o produto de um desenvolvedor.

Melhoria: Ninguém espera que seu processo alcance a perfeição e não necessite de aperfeiçoamento próprio. Mesmo se fôssemos tão bons quanto poderíamos ser, ambos produtos de ambientes de desenvolvimento e requisições de produtos estão mudando tão rapidamente que nossos processos estarão sempre correndo para alcançá-los. A meta do processo deve ser identificar e prototipar possibilidades para a melhoria no processo em si.

Acompanhamento: Um processo definido deve permitir o gestor, desenvolvedores e clientes acompanharem o status do projeto. Acompanhamento é o outro lado da previsibilidade. Ele mantém a par de como as nossas previsões são boas e, portanto, como melhorá-los.

2.2 Elementos do Processo de Software

Nesta seção serão apresentados e detalhados os principais elementos que compõe um processo de software.

2.2.1 Atividades

Atividades definem "como as coisas são feitas" ou "o que será feito" e para isso incorporam procedimentos, regras, políticas, agentes, recursos, papéis, artefatos (consumidos e produzidos). Cada atividade deve definir claramente seu início e final. Normalmente, atividades são organizadas em uma rede de duas dimensões: horizontal e vertical. A dimensão horizontal define os relacionamentos entre as atividades e a dimensão vertical decompõe a atividade em diversos níveis, ou seja, uma atividade composta por diversas outras sub-atividades. Exemplo de sub-atividades seria a decomposição da atividade gerenciamento de projeto nas sub-atividades: estimativas de tamanho, estimativas de custo, controle e acompanhamento.

2.2.2 Recursos

Recursos são entidades estáticas necessárias para executar uma atividade. A não utilização ou indisponibilidade de recursos necessários pode causar falha na execução de uma atividade ou até mesmo impedir sua execução. Recursos relacionam-se com diversos componentes do processo de software, por exemplo, técnicas, métodos e ferramentas. Deste modo, uma atividade

pode utilizar diversos recursos e um recurso pode ser utilizado por diversas atividades ou, um agente pode utilizar diversos recursos e um recurso pode ser utilizado por diversos agentes. Artefatos, ferramentas, equipamentos, ambientes (sala de reuniões, laboratórios, entre outros) são considerados como tipos especializados (derivados) de recursos.

2.2.3 Artefatos

Artefato é um tipo de recurso produzido ou consumido em uma atividade. Nesse contexto, um artefato pode ser utilizado como uma entrada (insumo) para uma determinada atividade ou como uma saída de uma atividade, resultado da execução de uma atividade (produto). Um artefato pode estar associado a diversas atividades e uma atividade pode estar associada a diversos artefatos. Artefatos, necessariamente, não são tangíveis. Em um contexto de projeto de software, alguns artefatos existem apenas na forma de um arquivo lógico (arquivo executável, por exemplo). Geralmente, artefatos são persistentes e as possíveis modificações sofridas por um artefato podem produzir versões de um mesmo artefato (versionados). Artefatos podem ser representados como um agregado de sub-artefatos (módulos de código-fonte) ou artefatos mais complexos podem ser decompostos em sub-artefatos (Projeto Procedimental). Exemplos de artefatos são, entre outros: código-fonte, código executável, manual de padrões, relatório de resultados, documento de requisitos, plano de trabalho.

2.2.4 Agentes

Agentes ou atores são as entidades que executam atividades por intermédio de um papel. Essa entidade pode ser um pessoa ou grupo de pessoas. Geralmente, agentes estão relacionados a papéis e atividades, assim, um agente pode executar diversos papéis e um papel pode estar relacionado a diversos agentes. Atividades envolvem diversos agentes e um agente pode participar de diversas atividades. Com isso diferentes agentes terão percepções diferentes do que acontece durante o processo de software.

2.2.5 Papéis

Papéis representam um conjunto de responsabilidades, obrigações, permissões e habilidades necessárias para executar uma atividade ou sub-atividade. Geralmente, papéis são desempenhados por agentes humanos. Um sinônimo de papel seria cargo ou função. Uma atividade pode exigir diversos papéis para ser executada e um papel pode ser aplicado em diversas atividades. Agentes podem assumir diversos papéis durante o desenvolvimento de software e um papel pode ser assumido por diversos agentes durante o desenvolvimento de software. Programador, analista de sistemas, gerente de projeto, testador entre outros são alguns exemplos de papéis que podem ser associados a um determinado processo de software.

2.3 Definição de Processos

A definição de processo de software ganhou importância com a criação de normas e modelos que auxiliam na definição e na avaliação de processos de software de uma organização. Os modelos e normas são importantes, pois agregam às organizações as melhores práticas de Engenharia de Software. Além disso, os modelos e normas servem como base para avaliar organizações que buscam a melhoria contínua de seus processos.

Abordagens importantes como as normas de qualidade *ISO 9000* e a *ISO/IEC 12207*, o modelo *Capability Maturity Model (CMM)* e o modelo *Software Process Improvement and Capability Determination (SPICE)* sugerem que melhorando o processo de *software*, pode-se melhorar a qualidade dos produtos (Bertollo e Falbo, 2003). A Norma *ISO/IEC 12207* estabelece uma estrutura comum para os processos de ciclo de vida de *software* desde a concepção de idéias até a descontinuação do *software* com uma terminologia bem definida e é composta de processos, atividades e tarefas que servem para ser aplicada durante a aquisição de um sistema que contém software, de um produto de software independente ou de um serviço de software, e durante o fornecimento, desenvolvimento, operação e manutenção de produtos de software.

O objetivo de se definir um processo de software é favorecer a produção de sistemas de alta qualidade, atingindo as necessidades dos usuários finais, dentro de um cronograma e um orçamento previsíveis (Falbo, 2006). Para isso é preciso criar uma representação do mesmo e depois descrevê-lo de forma que as pessoas possam se orientar na execução das atividades.

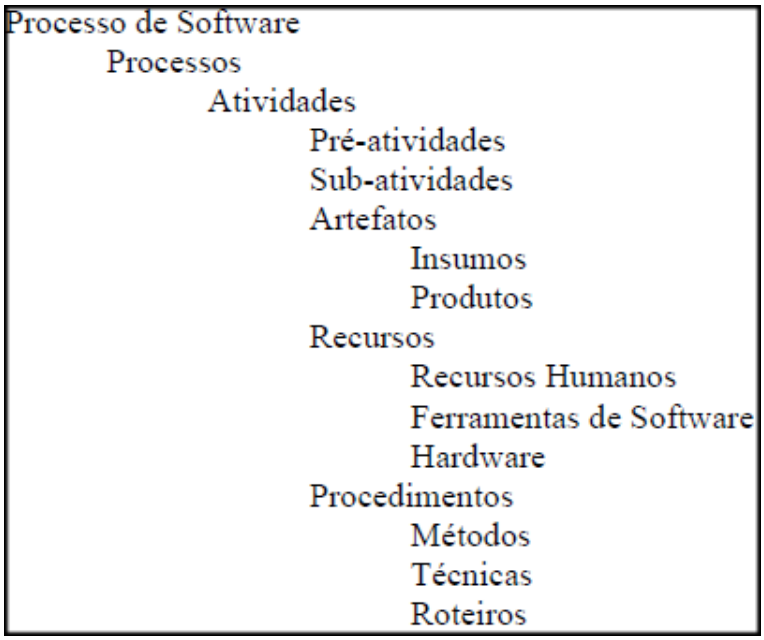

Figura 2.2: Itens do processo de software.

Em Falbo (2006) os itens do processo de software estão representados de forma esquemática na Figura 2.2. Este modelo faz uma adaptação ao modelo proposto pela norma *ISO/IEC 12207*, uma vez que, esta divide as atividades em tarefas, e na abordagem de Falbo (2006) as atividades são divididas em sub-atividades. A estrutura de processo de software a ser utilizada neste trabalho é a que foi descrita na Figura 2.2.

Estudos e experiência levaram à construção de um modelo para definição de processos de software que consiste em três etapas: definição do processo padrão, especialização do processo padrão e instanciação para projetos específicos. Como resultados de cada etapa temos processos em diferentes níveis de abstração (Rocha et al., 2001). A Figura 2.3 apresenta os diferentes níveis de definição de processos, bem como os fatores que influenciam a definição de processos em diferentes níveis.

O primeiro nível trata da definição de um processo padrão para a organização, neste nível deve ser contemplado apenas os ativos de processo essenciais que devem ser incorporados a quaisquer processos da organização. Idealmente, o processo padrão da organização deve ser definido considerando padrões e modelos de qualidade, como *CMMI* e *ISO/IEC 12207*. Um processo definido para um projeto de software descreve o conjunto de atividades que serão executadas no contexto do projeto, os recursos (hardware, software e humano) necessários, os artefatos (insumos e produtos) e os procedimentos (métodos, técnicas, roteiros e normas) a serem adotados na realização de cada uma das atividades (Falbo, 1998).

O segundo nível de definição trata dos processos especializados. Os processos especializados baseiam-se no processo padrão da organização. Porém, características específicas do desenvolvimento, tais como paradigma adotado, tipo de *software* ou, ainda, domínio do pro-

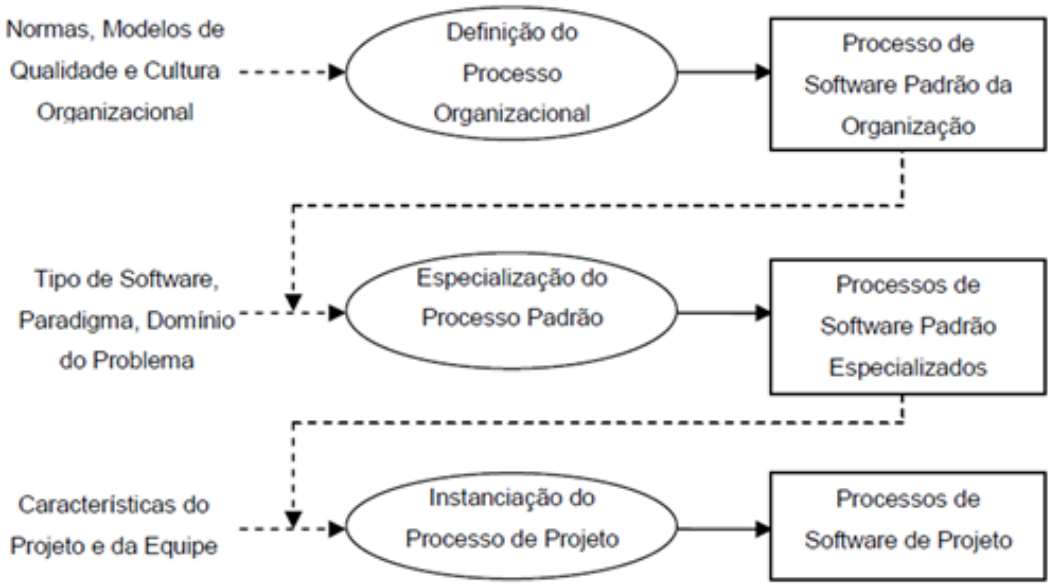

Figura 2.3: Etapas do processo de software.

blema, são definidas.

O último nível de definição trata da instanciação de um processo padrão ou especializado para um projeto específico. Nesse nível, o gerente de projeto pode selecionar um processo padrão ou especializado e a partir daí instanciar um processo de projeto. Durante a definição do processo, busca-se também eleger quais os indicadores devem ser coletados e quais as ferramentas de suporte devem ser utilizadas. Com o processo definido, e preferencialmente documentado, inicia-se a sua implementação.

Para que a cultura da organização não se perca, o ponto de partida é criar uma representação de como o processo é executado na prática, ou seja, modelar o processo de forma descritiva. Para isso, num primeiro passo procura-se identificar as principais atividades que são executadas durante o processo, a sequência de execução e os desvios condicionais existentes. O próximo passo é identificar os papéis e as responsabilidades relacionados a cada atividade. No terceiro passo o objetivo é identificar os artefatos que são produzidos ou consumidos na execução de atividade, e caso existam também seus respectivos templates. Na sequência é preciso identificar as medidas que devem ser utilizadas durante a execução de uma atividade, pois estabelecem os dados quantitativos e qualitativos que deverão ser coletados para a gerência de projetos, a melhoria e a garantia de qualidade do software. Para completar a descrição do processo atual, é preciso ainda levantar outras informações importantes tais como, critérios de entrada e de saída, métodos e recursos.

A modelagem do processo provê apenas uma representação inicial do processo, tornando explícitos os principais elementos que o compõe. Essa representação precisa ser documentada e revisada pelos participantes do(s) processo(s), com o objetivo de validá-la. A revisão busca

detectar inconsistências, erros, elementos importantes que estão faltando, ambiguidades, entre outras. Baseado nessa revisão, a documentação inicial vai sendo atualizada e completada, até que todos os revisores a aprovem.

2.4 Conclusões do Capítulo

Com este capítulo, conclui-se que processo é um conjunto de ações, tarefas, atividades ordenadas visando um objetivo, no caso um produto de *software* que atenda as necessidades dos envolvidos.

As empresas de TI encontram cada vez mais dificuldades ao desenvolver *softwares* mais complexos, em ambientes de desenvolvimentos mais complexos, atendendo os prazos e custos estipulados, com grande qualidade e satisfazendo todos os envolvidos. Para resolver esses problemas, os processos na Engenharia de Software fornecem estabilidade, controle e organização, padronizando os produtos.

Um processo tem que se atentar a eficácia, manutenibilidade, previsibilidade, repetibilidade, qualidade, melhoria e acompanhamento. E um processo tem que ser flexível (ou seja, um processo pode satisfazer o projeto "A" mas não satisfazer o projeto "B") e o mais simples possível para cada projeto, para que ele não se torne um problema no desenvolvimento do produto.

A partir da definição do processo padrão da organização, este é utilizado a cada novo projeto, para a geração do processo definido do projeto que é adaptado para abranger as características específicas do projeto. O plano de desenvolvimento de software deve ser feito a partir do processo definido do projeto e deverá descrever as atividades que serão executadas, implementadas e controladas.

Capítulo 3

Análise e Projeto do Sistema

Após a revisão bibliográfica feita sobre processos de software foi possível fazer o levantamento dos requisitos necessários para o desenvolvimento da ferramenta proposta neste trabalho.

Este capítulo apresentará questões relativas à análise e ao projeto do EPROCESSOS, um sistema editor de processos de software, o qual permite definir, isto é, criar modelos para estes. A modelagem deve ser capaz de descrever alguns aspectos necessários para a coordenação das atividades envolvidas num processo de software.

3.1 Levantamento de Requisitos

O modelo muitas vezes é utilizado como um guia para que seja possível acompanhar cada passo, identificar o estado do projeto e reorganizá-lo durante este acompanhamento. Em outros casos o modelo é utilizado apenas como documentação, isto é, como uma forma de explicitar o processo utilizado com o objetivo de conscientizar a equipe. Este último é o caso considerado neste trabalho.

Na ferramenta aqui desenvolvida é permitida a definição dos processos padrão, além de permitir a modelagem de conteúdos reutilizáveis. Isto possibilita a definição de diferentes processos utilizando o mesmo conjunto de artefatos, recursos, papéis e procedimentos. A partir dos estudos feitos verificou-se que a ferramenta desenvolvida deve possuir os requisitos funcionais listados a seguir:

- Definir / Editar / Excluir um processo;
- Cadastrar / Editar / Excluir atividades do processo;
- Selecionar dentre as atividades já cadastradas as sub-atividades;
- Selecionar dentre as atividades já cadastradas as pré-atividades;
- Cadastrar / Editar / Excluir artefatos;

- Associar os artefatos cadastrados a uma atividade como insumo ou produto desta;
- Cadastrar / Editar / Excluir recursos;
- Associar recursos cadastrados às atividades;
- Cadastrar / Editar / Excluir procedimentos;
- Associar procedimentos cadastrados às atividades;
- Cadastrar / Editar / Excluir papéis;
- Associar um recurso humano a um papel em uma atividade, gerar agente.

3.2 Diagrama de Classes

De acordo com os requisitos levantados na seção anterior foi desenvolvido o modelo conceitual apresentado na Figura 3.1.

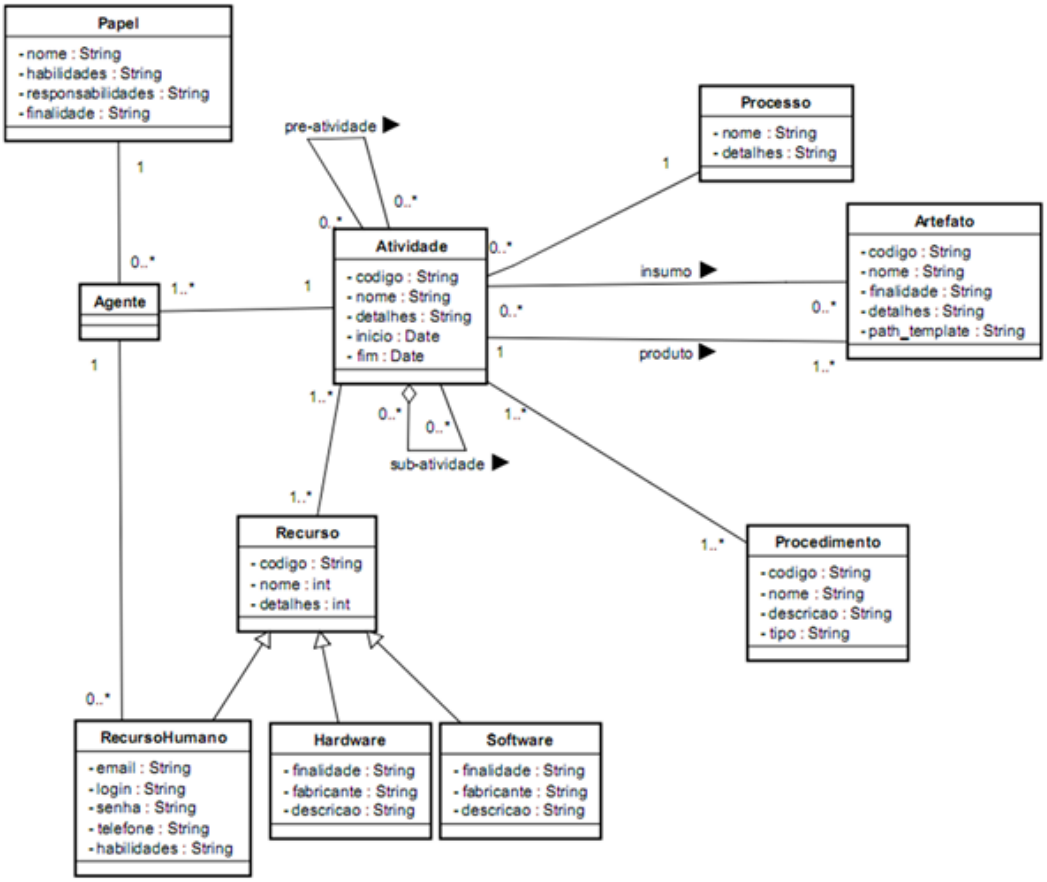

Figura 3.1: Diagrama de Classes

Para esta tarefa foi utilizada a ferramenta JUDE, uma *Integrated Development Enviroment* (IDE) para modelagem de dados *Unified Modeling Language* (UML), criada com *Java* e de uso fácil e intuítivo. Com a IDE JUDE é possível realizar uma modelagem de dados complexa, apresentar os dados para o usuário de forma clara e ainda possuir a vantagem de seu *layout* ser bem intuítivo.

Este diagrama será útil tanto para gerar as classes do sistema quanto para auxiliar na criação do modelo de banco de dados.

3.3 Modelagem do Banco de Dados

O modelo conceitual apresentado anteriormente foi utilizado para apoiar o desenvolvimento da modelagem do banco de dados utilizado no sistema. Este modelo está representado na Figura 3.2.

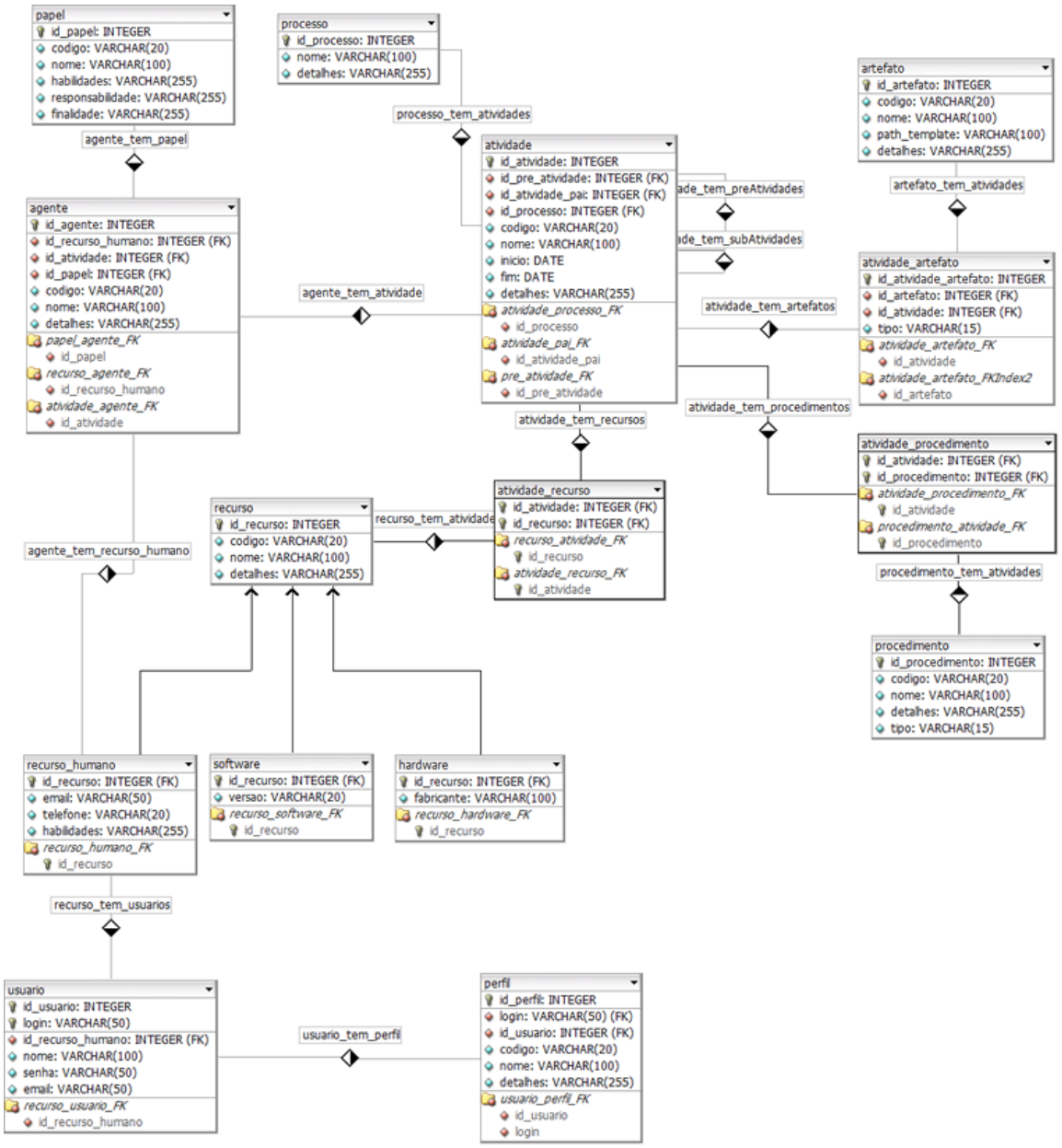

Figura 3.2: Modelo de dados relacional

Para esta tarefa foi utilizada a ferramenta *DBDesginer Fork*, um editor visual para criação de banco de dados que integra criação, modelagem, desenvolvimento e manutenção dos bancos em um ambiente gráfico simples. Possui uma função que permite exportar tabelas ou bancos

inteiros em diversos formatos ou em formato texto (apenas as sintaxes *Structured Query Language* - SQL). A Vantagem do *DBDesigner Fork* sobre o *DBDesigner 4* é que além do suporte ao *MySQL* ele também oferece suporte a *Firebird/InterBase*, *PostgreSQL* e *Oracle*.

3.4 Diagrama de Pacotes

A Figura 3.3 representa o diagrama de pacotes do sistema a ser desenvolvido, através desse diagrama é possível ver, de maneira abstrata, como as classes da aplicação se itegram.

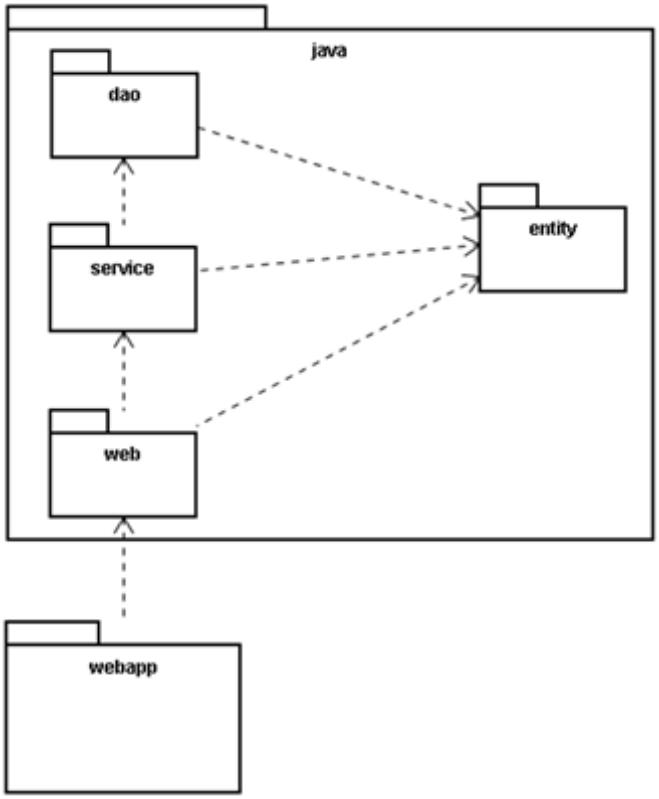

Figura 3.3: Diagrama de pacotes

No pacote **dao (data access object)** estão as classes responsáveis pela comunicação com o *Sistema de Gerenciamento de Banco de Dados* (SGBD), ou seja, todas as funcionalidades de bancos de dados, tais como obter as conexões e executar comandos *SQL*, são feitas por classes contidas neste pacote.

No pacote **entity** estão as classes responsáveis por mapear objetos do modelo de dados, sendo que foi gerada uma classe para cada tabela com suas propriedades. Sendo assim, estas classes incorporam objetos representativos do domínio da aplicação, incluindo aspectos de dados. Os objetos desta camada são usualmente persistentes, ou seja, são armazenados em

memória não volátil, tipicamente em bancos de dados relacionais. As classes desse pacote devem ser o mais independente possível do esquema de persistência adotado.

No pacote **service** estão as classes responsáveis pelas regras de negócio da aplicação, regras tais como listar, adicionar, remover e incluir um registro no banco de dados. Essas classes acessam e utilizam métodos implementados nas classes dos pacotes **dao** e **entity** para disponibilizar serviços da aplicação.

O pacote **web** é responsável por publicar os serviços e o modelo de dados, os quais estão representados nos pacotes **service** e **entity** para a camada de apresentação.

O pacote **webapp** contém os recursos necessários para a geração da interface gráfica para a interação do usuário com a aplicação, ou seja, este pacote representa a camada de visualização da aplicação. Esta interação é feita através do pacote web que é responsável por publicar os serviços.

3.5 Conclusões do Capítulo

Este capítulo teve por objetivo, com base em uma análise de requisitos, definir as características do sistema, isto é, o modelo conceitual. Este modelo conceitual foi utilizado para especificar as características do software. No capítulo seguinte, os conceitos e funcionalidades do modelo conceitual são utilizados na definição e implementação da arquitetura e componentes do sistema desenvolvido neste trabalho.

Capítulo 4

Implementação do Sistema

4.1 Introdução

Hoje a *Web* já se tornou uma plataforma sólida onde aplicações podem ser feitas de forma rápida e eficiente. Com o aumento da velocidade da *Internet* e o acesso banda larga, as tecnologias *web* continuam evoluindo para proporcionar novas experiências aos usuários e aumentar a utilidade e qualidade dessas aplicações (Loosley, 2006).

Devido a esse grande potencial para se criar sistemas *Web* foi escolhido implementar o sistema proposto neste trabalho, o EPROCESSOS, como sendo uma Aplicação de *Internet Rica* (da sigla em inglês *RIA - Rich Internet Application*), ou seja, um aplicativo *Web* que tem características e funcionalidades de softwares tradicionais do tipo *Desktop*.

4.2 Tecnologias Utilizadas

4.2.1 J2EE

A plataforma *Java Enterprise Edition* (J2EE), cuja versão adotada para este trabalho é a mais recente, *J2EE 6*, publicada em dezembro de 2009. A *J2EE* é um conjunto de especificações para acesso a diversos serviços de infraestrutura, tais como segurança, distribuição de processamento, controle de transações, comunicação entre camadas e outros (Júnior, 2003). Sendo assim, pode-se dizer que *J2EE* é uma plataforma *Java* voltada para redes, *internet*, *intranet* e semelhantes, a qual contém bibliotecas especialmente desenvolvidas para o acesso a servidores, a banco de dados entre outras características. Graças a essas características, o *J2EE* provê suporte a uma grande quantidade de usuários ao mesmo tempo. A grande vantagem da utilização desta plataforma é que ela permite ao desenvolvedor focar apenas na construção da lógica de negócio de suas aplicações, deixando serviços de infraestrutura sob a responsabilidade da plataforma. O modelo da arquitetura da plataforma J2EE pode ser visto na Figura 4.1.

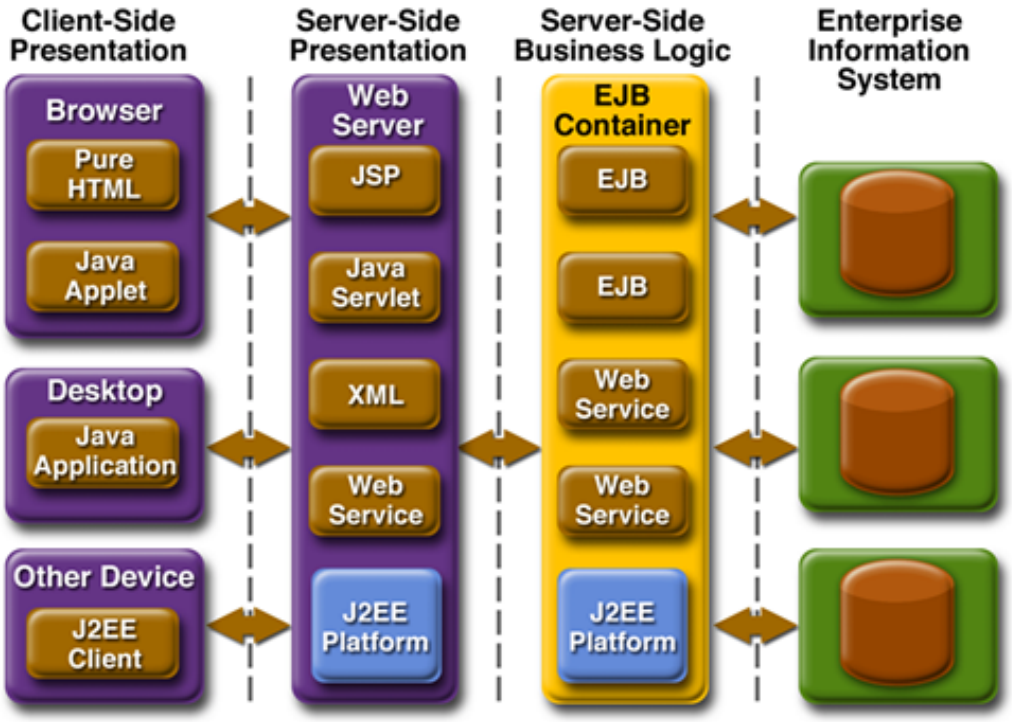

Figura 4.1: Arquitetura J2EE.

4.2.2 JSP

Java Server Pages (JSP) é uma tecnologia orientada a criar páginas web com programação em Java. Com JSP é possível criar aplicações *web* que se executam em vários servidores *web*, de múltiplas plataformas, já que Java é em essência uma linguagem multiplataforma. As páginas *JSP* estão compostas de código *HTML/XML* misturado com etiquetas especiais para programar scripts de servidor em sintaxe *Java*. Portanto, poderemos escrever as *JSP* com nosso editor *HTML/XML* habitual. A versão adotada neste projeto foi o *JSP 2.1*.

4.2.3 Servlets

São classes *Java* desenvolvidas de acordo com uma estrutura bem definida, e que, quando instaladas junto a um Servidor que implemente um *Servlet Container* (um servidor que permita a execução de *Servlets*, muitas vezes chamado de Servidor de Aplicações *Java*), podem tratar requisições recebidas de clientes (Temple et al., 2004). Portanto, os *Servlets* permitem a geração de conteúdo dinâmico nos sites.

4.2.4 JPA

A *Java Persistence API* (JPA), cuja versão adotada para este trabalho é a mais recente, *JPA 2.0* publicada em dezembro de 2009 juntamente com o *J2EE 6*.

A *JPA* é utilizada nas aplicações *Java* para permitir a gravação/leitura de objetos em bancos de dados relacionais de forma transparente. Essa técnica é conhecida como *Object Relational Mapping*(ORM).

Para fazer consultas com *JPA*, podemos usar a *JPA Query Language* (JPQL). Nessa linguagem, as instruções de consulta são escritas de forma muito semelhante ao padrão *SQL*. A implementação *JPA* utilizada transforma essas instruções *JPQL* em instruções *SQL* para que possam ser executadas no banco de dados.

A implementação da *JPA* utilizada neste trabalho foi o framework *Hibernate*, na sua versão 3.5.0-Final. O *Hibernate* é uma ferramenta de mapeamento objeto/relacional para *Java*. Ela transforma os dados tabulares de um banco de dados em um grafo de objetos definido pelo desenvolvedor. A idéia principal é criar uma camada entre a aplicação e o banco de dados de modo que o desenvolvedor não tenha que se preocupar com instruções *SQL* ou com o banco de dados na qual a aplicação irá persistir seus dados, algo que não é feito utilizando apenas o *JDBC*. Pelo fato de o *Hibernate* utilizar uma linguagem própria para realizar a persistência de dados, a *Hibernate Query Language* (HQL), ele traz como vantagem a possibilidade da mudança do banco de dados (devem ser bancos de dados compatíveis com o padrão *SQL*) utilizado a qualquer momento sem grandes alterações no código já escrito.

4.2.5 JAAS

O *Java Authentication and Authorization Service* (JAAS) é um conjunto de *API's* que permite as aplicações *Java* ter um controle de autenticação e de acesso. O *JAAS* implementa uma versão *Java* do *framework* padrão *Pluggable Authentication Module (PAM)*, e suporta autorização baseada em usuário. Isso permite que aplicações fiquem independentes desse controle de segurança. Serve para controlar permissões de vários tipos de recursos: arquivos, diretórios, conteúdos, *URLs*. Quando se usa este padrão de segurança, este módulo está em nível de servidor de aplicação e não de aplicação, ou seja, este módulo de autenticação será executado pelo servidor de aplicação, antes mesmo de acessar a aplicação.

4.2.6 JavaScript

JavaScript também conhecido por *"Mocha", "LiveScript", "JScript" e "ECMAScript"*, é uma das linguagens de programação mais populares do mundo. Praticamente todos os computadores pessoais do ao redor do globo, têm pelo menos um interpretador *JavaScript* instalado e em uso. Essa popularidade é devida inteiramente ao seu papel como "linguagem de script para o ambiente *Web*" (Crockford, 2001). Com *JavaScript* é possível criar efeitos especiais nas

páginas e definir interatividades com o usuário. O navegador do cliente é o encarregado de interpretar as instruções *JavaScript* e executá-las para realizar estes efeitos e interatividades, de modo que o maior recurso, e talvez o único, com que conta esta linguagem é o próprio navegador. É uma linguagem de programação bastante simples e pensada para fazer as coisas com rapidez. Inclusive as pessoas que não possuam uma experiência prévia na programação poderão aprender esta linguagem com facilidade e utilizá-la em toda sua potência com somente um pouco de prática. Entre as ações típicas que se podem realizar em *JavaScript* temos duas vertentes. Por um lado os efeitos especiais sobre páginas *Web*, para criar conteúdos dinâmicos e elementos da página que tenham movimento, mudem de cor ou qualquer outro dinamismo. Por outro lado, JavaScript nos permite executar instruções como resposta às ações do usuário, com o que podemos criar páginas interativas com programas como calculadoras, agendas, ou tabelas de cálculo (W3C, 2010).

4.2.7 JSON

JavaScript Object Notation (JSON) é um formato mais leve para transferências de dados entre cliente e servidor, ele é considerado um subconjunto da notação de objeto JavaScript, mas seu uso não requer JavaScript exclusivamente. Também é independente de linguagem, mas que usa convenções semelhantes as da família C. O JSON se baseia em duas estruturas de dados que tem suporte em praticamente todas as linguagens de programação modernas, as estruturas são as seguintes:

a) Um conjunto de pares nome/valor. Nas linguagens de programação modernas eles são considerados com um objeto, registro, dicionário;

b) Uma lista ordenada de valores, que geralmente é considerada um array, vetor, lista. Já que essas estruturas têm suporte em tantas linguagens de programação, o JSON é uma opção ideal para intercâmbio de dados entre sistemas diferentes. Além disso, já que o JSON se baseia em um subconjunto do JavaScript padrão, deve ser compatível com todos os navegadores *Web* modernos (Asleson e Schttuta, 2006).

4.2.8 AJAX

O *Asynchronous JavaScript And XML* (AJAX), segundo (Garret, 2005), na realidade não é uma tecnologia, é um conjunto de várias tecnologias, cada uma progredindo de forma independente, e que se juntaram de forma a poder explorar formas de melhorar a interação com os utilizadores em aplicações *Web*. O *AJAX* também pode ser visto como uma metodologia de desenvolvimento que utiliza diversas tecnologias disponíveis nos *browsers* atuais para melhorar a interface de aplicações Web (Silveira, 2005). Com a utilização em conjunto de várias tecnologias, que são *JavaScript*, *XML*, *XMLHttpRequest*, *Cascading Style Sheets* (CSS), *XHTML* e JSON, o AJAX maximiza a interação entre os usuários e as aplicações *Web*, tornando-as mais simplificadas, com ganho em desempenho.

4.2.9 EXTJS

O *EXTJS* é um *framework JavaScript* criado originalmente como uma extensão do *YUI (Yahoo! User Interface)*. Na época de sua criação, chamava-se *yui-ext*, funcionando junto ao *YUI*, que era a base. Por ser uma extensão do *YUI*, o *yui-ext* não funcionava sem ele (Rosa, 2009). Com ele é possível criar interfaces que se parecem muito com aplicações *desktop*. Ele disponibiliza muitos componentes e funções que facilitam e muito o desenvolvimento de interfaces. A criação de uma tela em *EXTJS* é feita criando, adicionando e alinhando os itens da tela, tudo dentro de containers e *layouts*, mas com a facilidade de que se pode facilmente alterar seus *CSS* para mudar algo. O *EXTJS* trabalha muito com *AJAX*, de várias formas ele pode solicitar e enviar informações para um servidor web através de *AJAX*. Todo o *EXTJS* trabalha com *JSON* (mas não só *JSON*), tanto para montar os objetos e componentes da tela, como para enviar dados para o servidor, e trabalhar com *JSON* em tudo, primeiro padroniza todo o ambiente, e segundo, facilita muito o desenvolvimento, pois o *JSON* é simples e muito completo. A licença do *EXTJS* é *open-source* somente para projetos *open-source*, de resto se tem um preço para utilizá-lo. Neste trabalho a versão utilizada foi a 3.2.1.

4.2.10 DWR

O *Direct Web Remoting* (DWR) é uma biblioteca desenvolvida em *Java* que permite que o código *Java* no servidor e o código *JavaScript* em um navegador possam interagir e chamar uns aos outros tão simples quanto possível (DWR, 2010).Com ele, a estrutura do código *Java* fica acessível ao cliente via *JavaScript*, não existindo uma distinção entre lado cliente e servidor do ponto de vista do desenvolvedor. O *DWR* é divido em dois componentes:

1. *Java Servlet*, que processa as requisições vindas do cliente e devolve para o navegador. *JavaScript*, que envia as requisições para o servidor e atualiza a página dinamicamente.

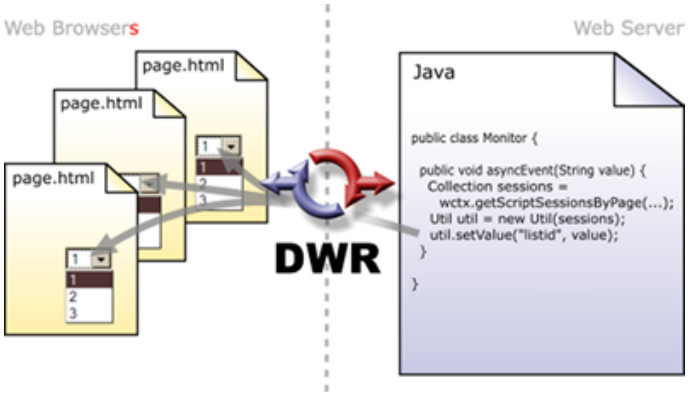

Figura 4.2: Chamada *Javascript* ao servidor *Java*.

A Figura 4.2 mostra basicamente o funcionamento da arquitetura do *DWR*, onde existem o *browser* e o servidor *Web*, ambos se comunicando. O *DWR* utiliza *XML* e o *XMLHttpRequest* para trafegar automaticamente os dados do lado cliente para o servidor. Isso tudo sem nenhuma necessidade de um navegador especial ou *plugins* específicos. Além disso, não sendo necessário também construir *Servlets* para tratar as requisições vindas do cliente pelo *JavaScript*, basta apenas criar classes simples com métodos de controle e chamá-lo pelo *JavaScript*. O *DWR* possui dois arquivos *JavaScript*. O primeiro é chamado de **"util.js"**. Ele contém um número de funções utilitárias que ajudam atualizar as páginas *Web* com os dados do *JavaScript* (tais como pode ser retornado ao usuário). O segundo é o **"engine.js"**, que é responsável por preparar e tratar as chamadas do *JavaScript*.

Essa biblioteca permite também mapear as classes modelo em classes *JavaScript* facilitando a manipulação no navegador.

Outra grande vantagem do *DWR* é que após toda essa configuração, é possível verificar se tudo foi feito corretamente através de uma ferramenta de teste que o *DWR* fornece a qual pode ser acessada na aplicação pelo endereço, adicionando /**dwr** após o contexto da aplicação, como pode ser visto na Figura 4.3.

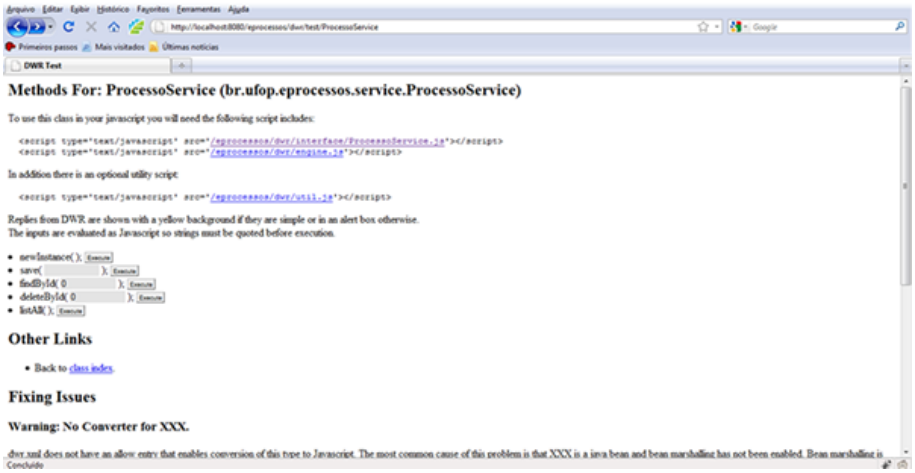

Figura 4.3: Página de teste do DWR.

4.2.11 Guice

O *Guice* (Guice, 2010) é um *framework* leve para injeção de dependências. Sua implementação é bastante simples e as dependências podem ser configuradas através de métodos *Java* ao invés de contextos em *XML*, facilitando testes e modularidade. Sendo assim, podemos dizer que o *Guice* fornece um meio para construção de objetos, reduzindo o acoplamento entre componentes, privilegiando a facilidade de manutenção e garantindo a testabilidade do código.

4.3 Arquitetura Proposta

O sistema *EPROCESSOS* é uma aplicação *Java EE* de interface *Web*. A plataforma *Java EE* foi escolhida por ser uma tecnologia portável e dispor de soluções para aspectos inerentes às aplicações servidoras multicamadas, transacionais e distribuídas. As tecnologias especificadas na plataforma *Java EE* são realizadas através da utilização dos containers *Java EE* que implementam as especificações dos vários componentes que compõem uma aplicação corporativa.

A solução para acesso dos clientes da aplicação será a utilização de clientes "magros"[1] através de navegadores *Web*, acrescentando quando necessário funcionalidades *AJAX* com objetivo de melhorar a usabilidade e interatividade do usuário final com a aplicação. Dessa forma as soluções tecnológicas necessárias para o *EPROCESSOS* estão centralizadas nos containers *Web*. O container *Java EE* selecionado para o projeto foi o *Tomcat*. A Figura 4.4 representa a arquitetura adotada para o desenvolvimento do sistema proposto neste trabalho.

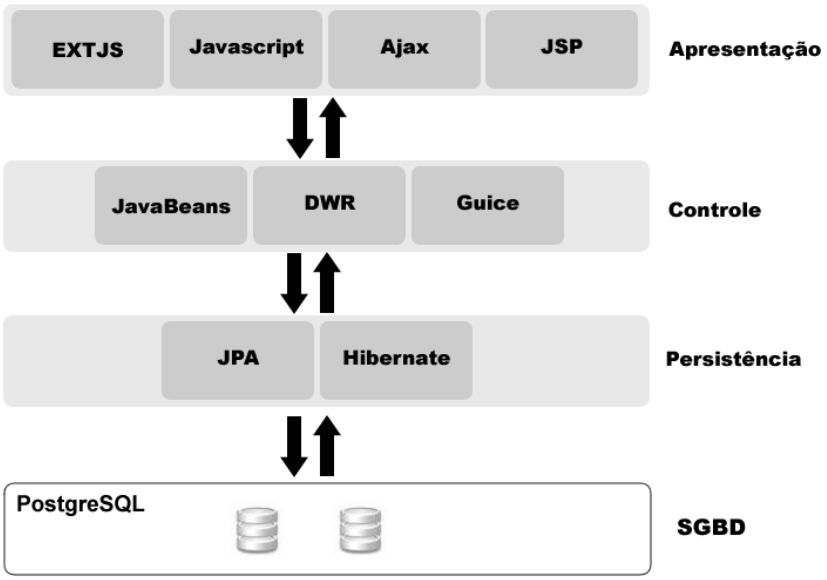

Figura 4.4: Arquitetura do sistema.

Na camada de apresentação foram utilizadas tecnologias complementares ao controle existente em *DWR*, como páginas descritas com *JSP* e *XHTML*, complementadas por funcionalidades *AJAX* inseridas pela fácil integração com o *framework EXTJS*. O visual da aplicação envolve a utilização padronizada de folhas de estilo *(CSS)* e componentes do *EXTJS*. Os componentes de apresentação acessam a camada de negócio via objetos *Javascript* gerados pelo *DWR* que fica responsável por fazer a comunicação com as classes *Java (JavaBeans)* de dentro do próprio *JavaScript(JSP/JS* $\leftrightarrow$ *JAVA)*, permitindo o controle de transação.

[1] Em um modelo cliente magro, todo o processamento de aplicação e o gerenciamento de dados é realizado no servidor. O cliente é responsável, simplesmentes por executar o software de apresentação.

A camada de controle implementa a lógica da aplicação, expondo esta lógica para a camada de apresentação. A alternativa de implementação escolhida para este trabalho foi separar a lógica em classes designadas por JavaBeans, classes estas que são acessadas pela camada de apresentação através de serviços *DWR*.

Por último a persistência de dados é realizada através de mapeamento objeto-relacional utilizando a especificação *JPA* e sua implementação com *Hibernate*. O mapeamento dos objetos seguiu uma abordagem *up-down*, onde os objetos de negócio (abstrações chave) foram identificados na fase de especificações, para que sejam mapeados em tabelas do banco de dados alvo. Para a manipulação e gerenciamento do banco de dados é necessário a utilização de um SGBD e para este trabalho foi escolhido o *PostgreSQL*, entretanto pelo mecanismo de persistência utilizado a aplicação é portável em relação ao banco de dados usado.

4.3.1 Fluxo Básico de Funcionamento do Sistema

O fluxo começa quando o usuário, através da interface, faz uma requisição ao servidor. Dentre as requisições implementadas estão: definir um novo processo, cadastrar nova atividade, cadastrar/remover itens do processo (artefatos, papéis, recursos e procedimentos), adicionar/remover uma sub-atividade, adicionar/remover pre-atividade e adicionar/remover agente. A requisição é feita por uma chamada *JavaScript* ao servidor. Nela contém as variáveis e objetos necessários para a utilização do serviço. O componente *DWR* trata a passagem do ambiente da interface para o ambiente do servidor, mantendo a compatibilidade, ou seja, os tipos e valores das variáveis passadas pelo *JavaScript* são reconhecidas como tipos e valores do *Java*. Ao entrar no ambiente do servidor, a requisição é direcionada para a camada de apresentação que serve apenas para fazer a comunicação entre a interface do usuário e a camada inferior. Ela é responsável também, se necessário, por instanciar objetos que serão utilizados pela camada inferior. Então, a partir daí a requisição é enviada para a próxima camada, que tratará de toda a lógica de negócios da requisição solicitada. A camada de controle receberá a requisição e realizará as transformações necessárias de acordo com o serviço solicitado. Isso é feito de tal forma que a requisição possa ser passada para a próxima camada, responsável pela persistência no banco. A camada persistência receberá a requisição e a persistirá no banco, retornando um valor ou objeto. Esse valor percorrerá de volta para as camadas de controle e apresentação, chegando na Interface do Usuário. Essa resposta pode ser apenas uma confirmação de que, por exemplo, um processo foi cadastrado com sucesso.

4.4 Ambiente de Desenvolvimento

Para a construção do *EPROCESSOS* o ambiente de desenvolvimento utilizou as ferramentas estão descritas nas seções seguintes.

4.4.1 Eclipse + WTP

O *Eclipse* é um *IDE* para codificação e implantação rápida de aplicações em desenvolvimento. A versão utilizada neste trabalho foi o *Eclipse Java EE IDE for Web Developers 3.6 - Helios release*. Esta versão do *Eclipse* já vem instalada com o plugin *Web Tools Platform* (WTP) que tem por finalidade prover um ambiente de desenvolvimento padrão para *web* sob a plataforma *Eclipse*, sendo formado por um conjunto de *APIs* para *J2EE* e aplicações *web*, além de ferramentas para suporte de implantação e testes de aplicações.

4.4.2 JDK

O *Java SE Development Kit* (JDK) é um *Kit* de Desenvolvimento *Java*, ou seja, um conjunto de utilitários que permitem criar sistemas de software para plataforma *Java*. Ele contém todo o ambiente necessário para a criação e execução de aplicações *Java*, incluindo a máquina virtual Java (JVM), o compilador *Java*, *APIs* do *Java* e outras ferramentas utilitárias. A versão utilizada neste trabalho é a JDK 6.20.

4.4.3 Apache Maven

O *Maven* é um projeto de código livre, mantido pela *Apache Software Foundation*, criado originalmente para gerenciar o complexo processo de criação do projeto *Jakarta Turbine* (Filho, 2008). É uma poderosa ferramenta para construção e implantação das aplicações durante o desenvolvimento e para a integração contínua.

4.4.4 Tomcat

O Tomcat, um *Container Java EE*, surgiu dentro do conceituado projeto *Apache Jakarta*. Sua principal característica é estar focado na linguagem de programação *Java*, em especial nas tecnologias *Servlets* e *JSP*. O *Tomcat* pode se comportar como um servidor *web (HTTP)*. Neste trabalho foi utilizado o *Apache Tomcat* versão 6.0 que implementa a *Servlet 2.5* e *JavaServer Pages 2.1* especificações do *Java Community Process*, e inclui muitos recursos adicionais que a tornam uma plataforma útil para o desenvolvimento e implantação de aplicações *web* e *web services* (Tomcat, 2010). Para utilizar o *Tomcat* com uma performance satisfatória é recomendável ter no mínimo *1GB de memória RAM*.

4.4.5 PostgreSQL

É um sistema gerenciador de banco de dados relacional, de código aberto e gratuito para qualquer uso. O *PostgreSQL* apresenta o melhor otimizador de consulta dentre todos os bancos de dados livres. Com todas as qualidades listadas acima, o *PostgreSQL* oferece uma utilização muito boa principalmente se tratando de desenvolvimento sistemas web para empresas. Neste trabalho foi utilizado o *PostgreSQL* versão 8.3.

4.4.6 Firebug

É uma extensão para o navegador *Mozilla Firefox* que adiciona ao navegador inúmeras ferramentas para facilitar a tarefa de desenvolvimento de páginas *web*. Ele possibilita a identificação e eliminação de erros de programação, edição e também o monitoramento de *CSS*, *HTML* e *JavaScript* presentes em qualquer página da *internet*.

4.5 Conclusões do Capítulo

Este capítulo teve como objetivo descrever as tecnologias e ferramentas utilizadas para a implementação do EPROCESSOS. Outro ponto abordado foi a arquitetura adotada na implementação, isto é, como as tecnologias utilizadas se integram. Além disso, também foi descrito o fluxo de funcionamento da aplicação.

No capítulo seguinte serão mostradas as telas do sistema implementado e também será detalhado como cada uma destas telas funciona.

Capítulo 5

Funcionamento do Sistema

5.1 Introdução

Este capítulo apresenta o funcionamento do sistema, bem como os requisitos necessários para sua execução.

5.2 Tecnologias Requeridas

Para que o usuário (cliente) execute o sistema basta a ele ter instalado em seu computador um *browser*(navegador de internet) com permissão para rodar *Javascript*.

Para o servidor é recomendável que o computador possua pelo menos 1GB de memória RAM, necessário para rodar o *Apache Tomcat* com uma boa performance. Além disso, este computador deve ter instalado as seguintes ferramentas:

- O *SGBD PostgreSQL* na versão 8.25;
- A *JDK* na versão 6.20;
- O servidor *Apache Tomcat* na versão 6.0.

5.3 Funcionamento do Sistema

5.3.1 Login

Para acessar o sistema o usuário deverá estar autenticado e para isso ele deve fornecer o nome de usuário e sua senha.

5.3.2 Tela Inicial do Sistema

Após fazer o *login* corretamente o usuário terá a tela inicial do sistema, a qual pode ser vista na Figura 5.1:

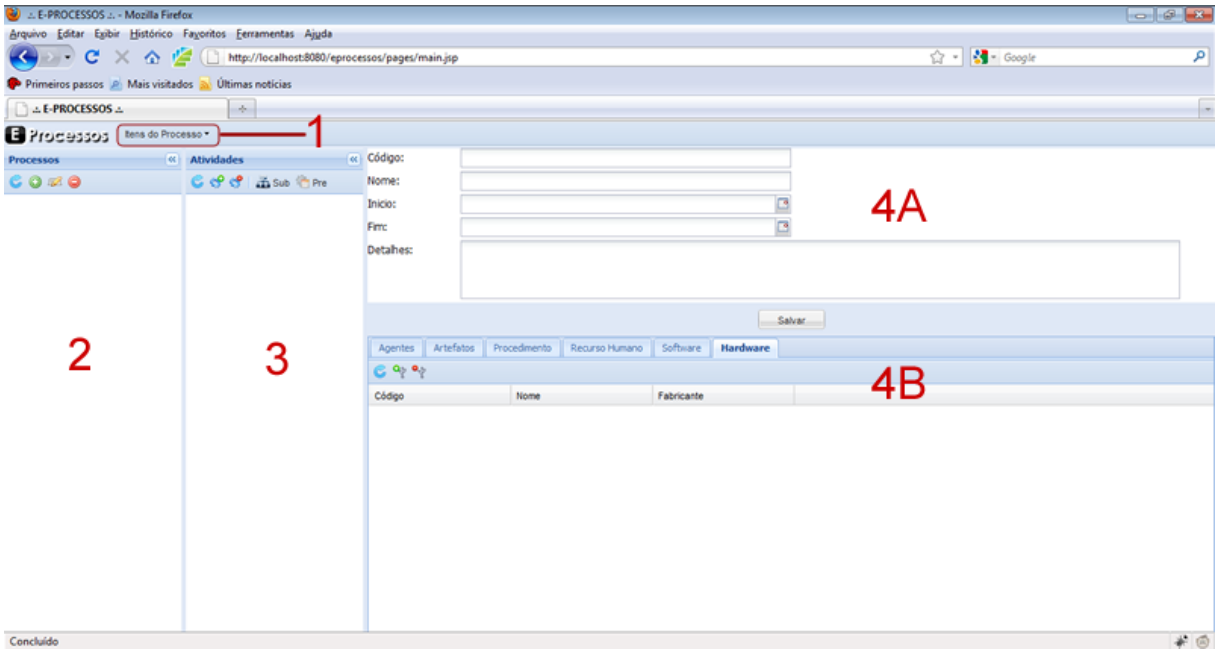

Figura 5.1: Tela Inicial do Sistema.

Na parte "1" temos o menu "Itens do processo" onde é possível visualizar, adicionar e remover os itens que compõem o processo.

Abaixo da barra superior temos a tela divida em três partes.

Na parte "2" temos uma lista com os processos cadastrados no sistema. Como pode ser visto, inicialmente esta lista apresenta-se vazia, uma vez que não há nenhum processo criado ainda.

Na parte "3" temos uma outra lista, sendo esta utilizada para mostrar as atividades que compõe cada processo listado na parte "1". Esta lista também aparece vazia inicialmente, uma vez que para visualizar as atividades que compõe um processo primeiro deve ser dado um duplo clique em um dos processos que estiver listado na parte "2".

A parte "4", relativa à edição das ativadades do processo, é subdividida em duas partes: a parte "4A" e a parte "4B". A parte "4A" é composta por um formulário com os campos para a descrição da atividade. Tal formulário é útil para editar a descrição de uma atividade. Já a parte "4B" é composta por 6 abas, cada uma listando os itens do processo que estão sendo utilizados em uma atividade.

5.3.3 Menu Itens do Processo

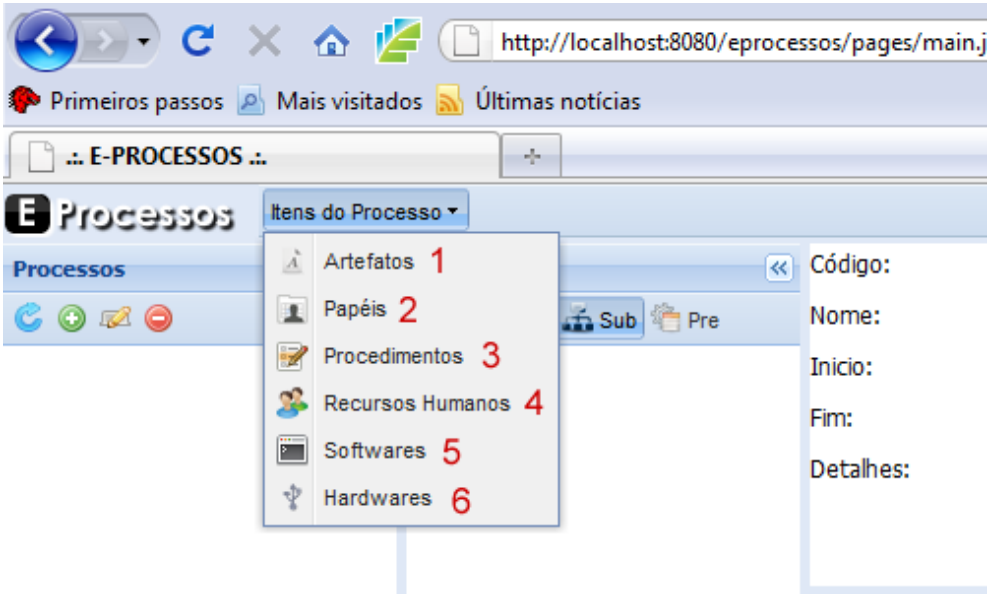

Figura 5.2: Tela Menu Itens do Processo

A Figura 5.2 apresenta o menu "Itens do Processo", sendo este composto por seis botões. Ao se clicar em um destes botões abre-se uma nova janela, a qual apresenta uma lista dos itens cadastrados e um menu para adicionar,remover e editar estes itens.

Como todas as telas dos itens do processo apresentam um mesmo padrão foi decidido apresentar de forma detalha apenas o item "1", de forma que o usuário entendendo o funcionamento desta tela ele entenderá facilmente o funcionamento das telas equivalentes dos intes "2" a "6".

5.4 Cadastro de Artefatos

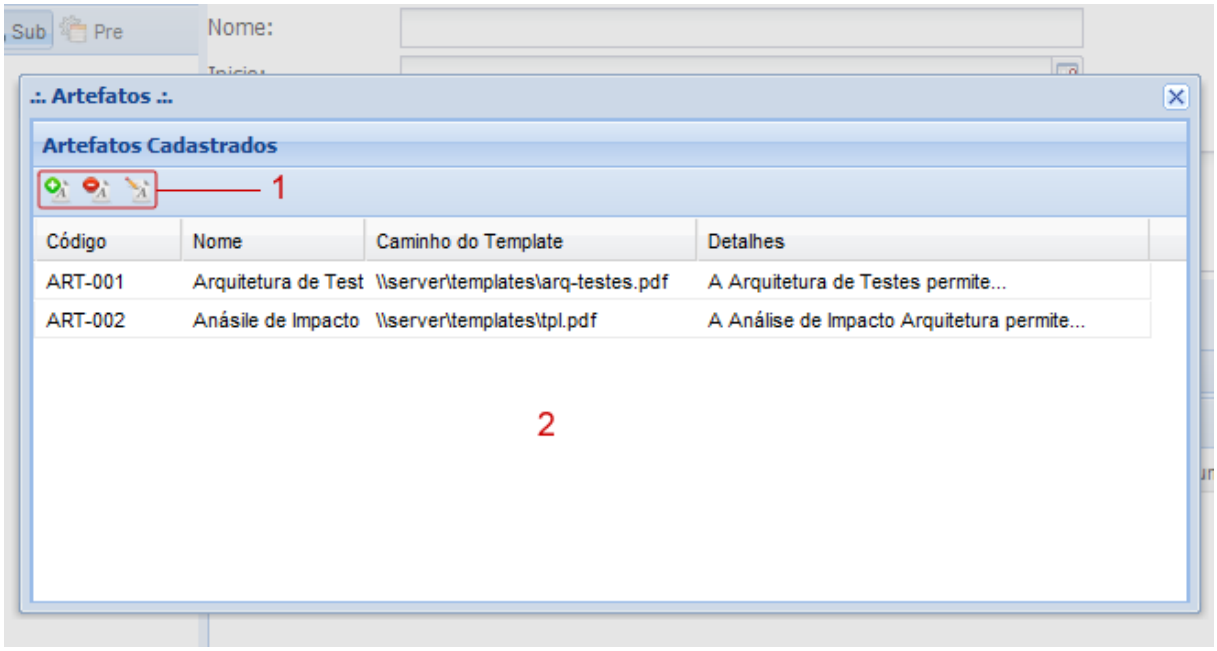

Figura 5.3: Tela Lista de Artefatos Cadastrados

Na parte "1" tem-se um menu composto por três botões. O primeiro, da esquerda para a direita, é o botão para a função adicionar um novo artefato. Ao clicar neste botão abre-se o formulário, como o que está apresentado na Figura 5.4. O segundo é o botão para função deletar um artefato, sendo necessário selecionar previamente um artefato listado na parte "2". Se nenhum artefato estiver selecionado o sistema retorna um aviso ao usuário informando que nenhum artefato foi selecionado. O terceiro e último botão é o de edição do artefato. Ao clicar nele abre-se o mesmo formulário visto na figura 5.4, porém com os campos já preenchidos com os dados do artefato a ser editado. Para ação de editar também é necessário selecionar previamente um artefato listado na parte "2", caso contrário o sistema retorna um aviso ao usuário.

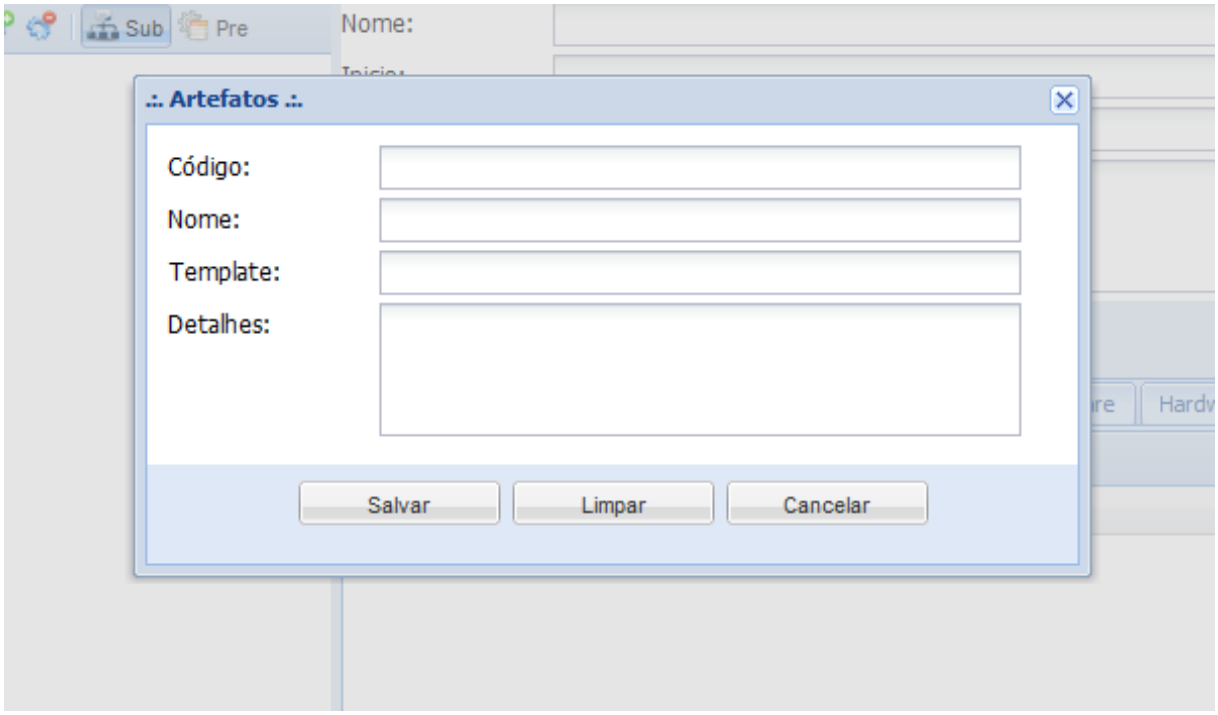

Figura 5.4: Tela Cadastro de Artefato

A tela para cadastros de artefato apresentada na Figura 5.4 é composta por um campo código, sendo este usado para uma identificação interna da empresa de desenvolvimento de software. Sendo assim, dentro da empresa já deverá existir um padrão de código para artefatos. Um exemplo de código é 'ART-001'. Em seguida, temos o campo nome que tem a função de atribuir um nome ao artefato com o objetivo de dar a este uma identificação mais detalhada do que o código. O próximo campo é o template, neste campo deve ser fornecido o endereço físico de um modelo, se existir, que representa o artefato a ser cadastrado. E por fim, temos o campo detalhes que é um campo livre para que o usuário descreva alguma informação relevante do artefato.

5.5 Cadastro de Processos

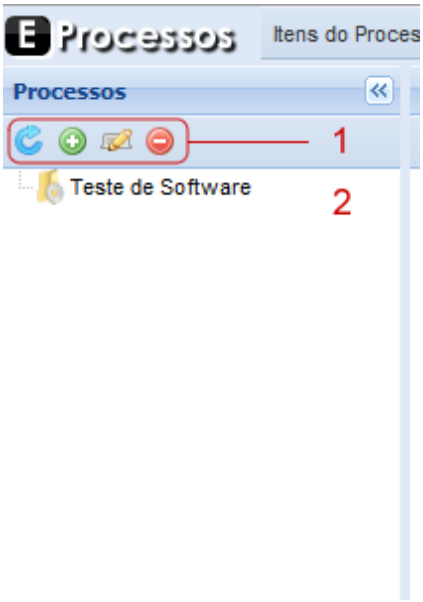

Figura 5.5: Tela Lista de Processos Cadastrados

A Figura 5.5 mostra o componente que lista e permite definir um novo processo, editar e excluir. Ele é composto por duas partes: na parte "1" temos um menu composto por quatro botões. O primeiro botão, da esquerda para a direita, é o botão para atualizar a listagem dos processos cadastrados que estão dispostos na parte "2".

O segundo é o botão para a ação de definir um novo processo, ao clicar neste botão uma janela como a apresentada na Figura 5.6 é aberta.

O terceiro botão permite editar um processo cadastrado, mas para isso deve-se primeiramente selecionar um dos processos que estão listados na parte "2". Caso não haja um processo selecionado o sistema retorna um aviso ao usuário informando que nenhum processo foi selecionado. Ao clicar neste botão abra-se uma janela como a vista na Figura 5.6 , mas com os campos já preenchidos com as informações do processo selecionado.

O quarto é o botão para excluir um processo cadastrado, mas para isso deve-se primeiramente selecionar um dos processos que estão listados na parte "2". Caso não haja um processo selecionado o sistema retorna um aviso ao usuário informando que nenhum processo foi selecionado.

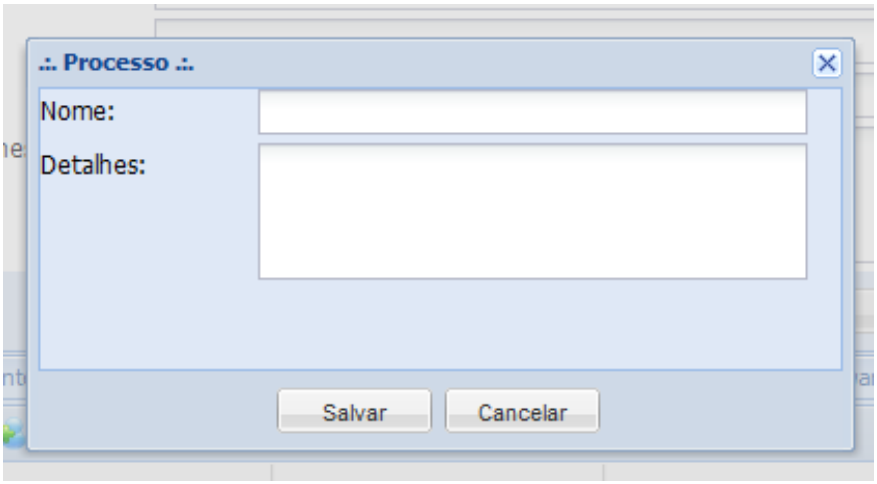

Figura 5.6: Tela Cadastro de Processo

A Figura 5.6 apresenta a janela para o cadastro de um novo processo. Nela há um formulário com dois campos e dois botões. O primeiro campo é o referente ao nome do processo, como exemplo podemos citar o processo "Teste de Software".

O segundo campo é o referente aos detalhes do processo, nele o usuário pode descrever informações importantes que devem ser levadas em consideração no processo que a ser cadastrado.

O primeiro botão é o "Salvar". Ao clicar nele os dados contidos nos dois campos são enviados para o servidor e este fica responsável por salvar os dados no banco de dados.

O segundo botão é o "Cancelar". Ao clicar nele a janela é fechada sem que nenhuma alteração nos dados seja feita, isto é, nenhuma informação é enviada ao servidor mesmo que os campos estejam preenchidos.

A janela apresentada na Figura 5.6 apenas cadastra as informações básicas do processo, sendo que este é composto além destes dados básicos de atividades. O cadastro de atividades de um processo se dará como o exemplificado na seção seguinte.

5.6 Cadastro de Atividades

A Figura 5.7 mostra o componente responsável por listar todas as atividades, este esta dividido em três partes.

Na parte "1" temos três botões. O primeiro, da esquerda para a direita, é o botão responsável por atualizar a listagem das atividades cadastras em um processo, tal listagem está disposta na parte "3".

O segundo é o botão para adicionar uma nova atividade. Para isto, primeiro deve-se selecionar qual o processo que se deseja adicionar a atividade, esta seleção é feita dando um **duplo clique** em um dos processos listados na parte "2" da Figura 5.5. Há difentes maneiras para se adicionar uma atividade. A primeira delas é quando a atividade a ser cadastrada não

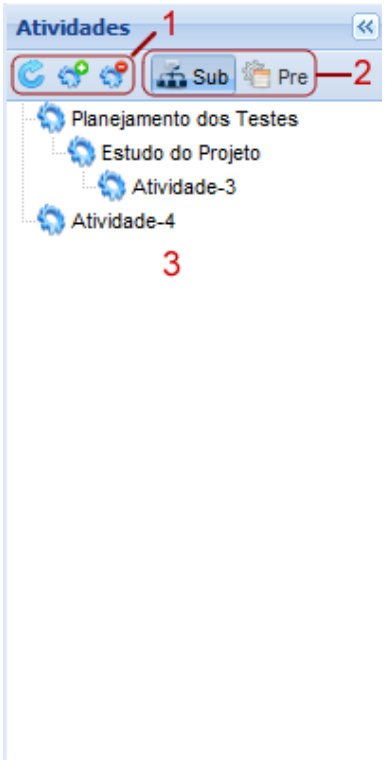

Figura 5.7: Lista de Atividades Cadastradas na Visão de Sub-Atividades

é subatividade e nem pré ativiade de nenhuma outra atividade, sendo assim basta clicar no botão que a atividade será cadastrada. A outra maneira é quando a atividade a ser cadastrada deve ser uma subatividade de alguma atividade, neste caso é preciso selecionar a visão de subatividades clicando no primeiro botão da parte "2", dar um **duplo clique** na atividade a qual se deseja adicionar a subatividade e então clicar no botão para adicionar a atividade. A terceira e última maneira para se adicionar uma atividade é quando se deseja que ela seja uma pré atividade de alguma atividade, neste caso é preciso selecionar a visão de pré atividades clicando no segundo botão da parte "2", dar um **duplo clique** na atividade a qual se deseja adicionar a pré atividade e então clicar no botão adicionar atividade. Ao se adicionar uma atividade os valores de código da atividade e nome da atividade já vem cadastrados seguindo um padrão, sendo que o código segue o padrão "AT-"+ o número de atividades cadastras e o nome da atividade segue o padrão "Atividade-" + número de atividades cadastradas. Os dados da atividade podem ser editados na tela vista na Figura 5.9. Para editar uma atividade basta dar um **duplo clique** na atividade que se deseja editar.

O terceiro é o botão para deletar as atividades cadastradas, para isso é preciso selecionar uma das atividades listadas na parte "3", caso contrário, o sistema exibirá um aviso ao usuário informando que nenhuma atividade foi selecionada.

Na parte "2" temos dois botões, responsáveis por selecionar a maneira com a qual se deseja

visualizar as atividades listadas na parte "3". O primeiro botão ativa a visão de "subatividades", esta é a visão que é apresentada na Figura 5.7. Nesta visão as atividades são ordenadas hierarquicamente de forma que a atividade mais externa é a atividade principal e as mais internas são as suas subatividades. Por exemplo, a atividade "Planejamento dos Testes" possuí as subatividades "Estudo do Projeto" e "Atividade 3", esta última é uma subatividade da atividade "Estudo do Projeto". O segundo botão ativa a visão de "pre-atividades", esta visão é apresentada na Figura 5.8. Nesta visão as atividades estão ordenadas hierarquicamente de forma que a atividade mais externa é a atividade principal e a mais interna é a pré-atividade. Por exemplo, na Figura 5.8, temos que atividade "Planejamento dos Testes" possui como pré-atividade a "Atividade 4". Isto é, para que a atividade "Planejamento dos Testes" começe a ser executada, primeiro a atividade "Atividade 4" deve ser concluída. As atividades "Estudo do Projeto" e "Atividade 3" não possuem pré-atividades.

Na parte "3" temos a listagem das atividades cadastradas, esta pode ser vista de duas maneiras de acordo com a seleção feita na parte "2".

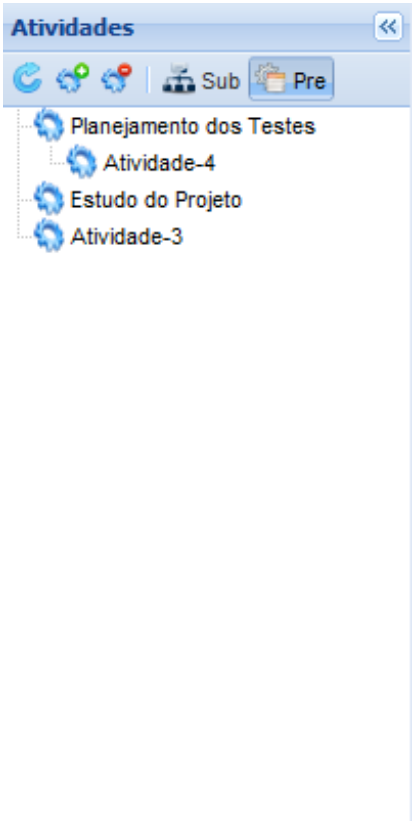

Figura 5.8: Lista de Atividades Cadastradas na Visão de Pré-Atividades

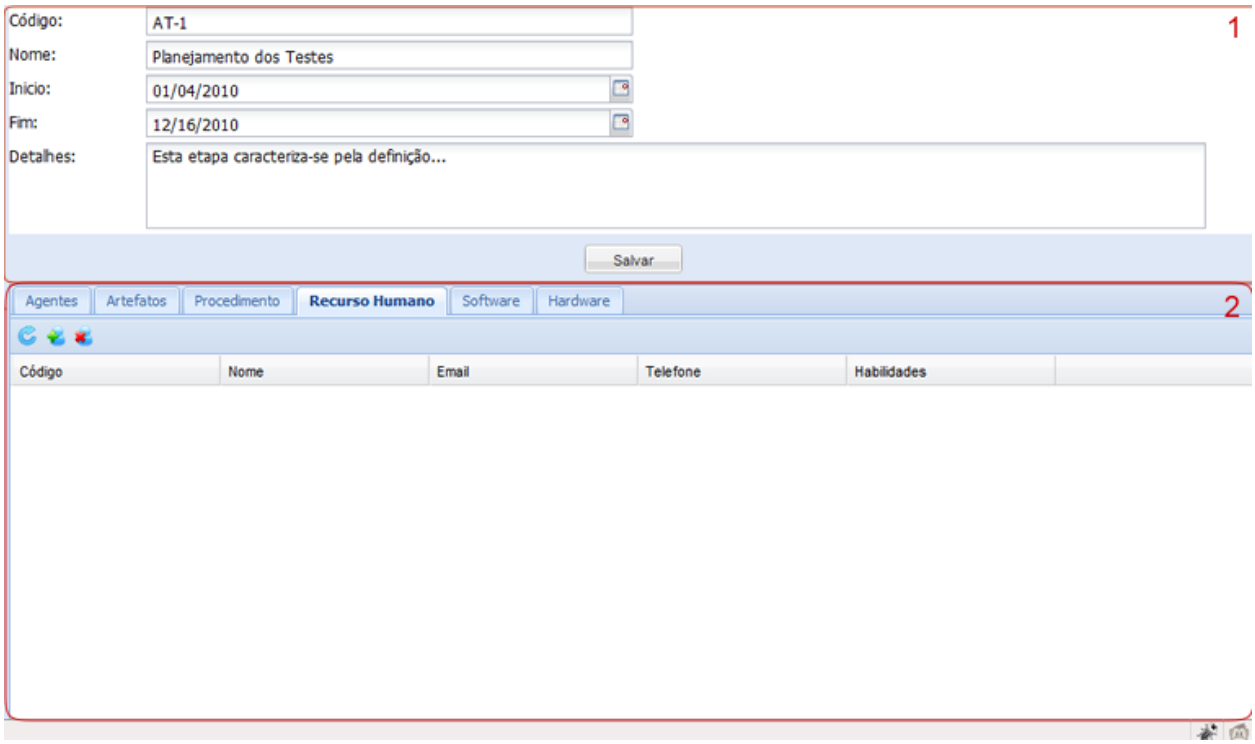

Figura 5.9: Tela Editar Atividade

A Figura 5.9 apresenta a tela para a edição de atividades. Esta tela esta dividida em duas partes.

Na parte "1" está o formulário com cinco campos e um botão. O primeiro campo é o código usado para uma identificação interna da empresa de desenvolvimento de software. Sendo assim, dentro da empresa já deverá existir um padrão de código para atividades. O segundo campo é o nome da atividade atribuir um nome ao artefato com o objetivo de dar a este uma identificação mais detalhada do que o código. O terceiro campo é a data de início da atividade, que visa informar a data a a atividade deverá começar a ser executada. O quarto campo é a data de termino da atividade, que visa informar a data máxima em que a atividade deve ser concluída. O último campo é o campo descrição, útil para detalhar informações que serão relevantes na execução da atividade. O botão "Salvar" envia os dados dos campos para o servidor e este os armazena no banco de dados, após isto o sistema exibe um aviso ao usuário informando que os dados foram salvos com sucesso. Na parte "1" estão apenas os dados básicos de uma atividade, sendo que uma atividade utiliza para sua execução itens do processo que já deverão estar cadastrados, cadastros estes que podem ser feito através do menu "Itens do Processo" que é apresentado na parte "1" da Figura 5.1 e na Figura 5.2.

A parte "2" é composta por seis abas, cada uma contendo um menu e lista para adicionar, editar e remover os itens que serão utilizados na execução da atividade. O cadastro de Agentes será detalhado na seção 5.7 e o cadastro de Artefatos será detalhado na seção 5.8. Os demais

cadastros não serão detalhados, uma vez que eles seguem os mesmos padrões dos cadastros que serão detalhados nas próximas seções.

5.7 Adicionar Agentes a uma Atividade

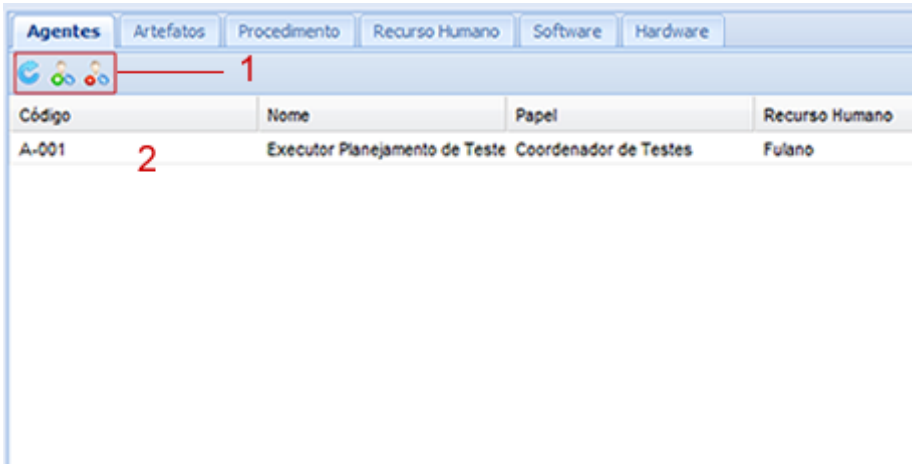

Figura 5.10: Tela agentes utilizados em uma atividade

A Figura 5.10 mostra a tela de cadastro de agentes. Na parte "1" encontra-se um menu com três botões. O primeiro botão tem a função de atualizar a lista de agentes cadastrados na parte "2". O segundo botão tem a função de adicionar um novo agente à atividade. Ao clicar neste botão será aberta uma janela como a que é apresentada na Figura 5.11. O terceiro botão tem a função excluir um agente, para isso um agente deve ser previamente selecionado na parte "2", se nenhum agente for selecionado e o botão for clicado o sistema exibirá um aviso ao usuário informando que nenhum agente foi selecionado.

Na parte "2" tem-se o componente responsável por listar todos os agentes que a atividade utiliza.

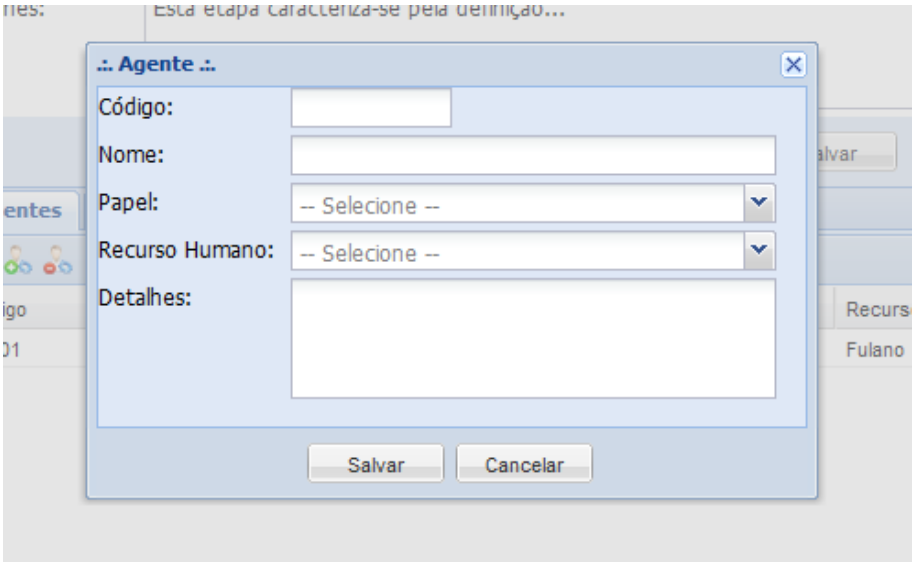

Figura 5.11: Tela Cadastrar Agente

Lembrando que, como foi dito na seção 2.2.4, um agente é um recurso humano que executa uma atividade por intermédio de um papel.

A Figura 5.11 apresenta a tela para adicionar um agente a uma atividade. A tela apresenta um formulário com cinco campos e dois botões. O primeiro campo é o código, sendo este usado para uma identificação interna da empresa de desenvolvimento de software. Sendo assim, dentro da empresa já deverá existir um padrão de código para agentes. O segundo campo é o nome que tem o objetivo de atribuir um nome ao agente com o intuito de dar a este uma identificação mais detalhada do que o código. O terceiro campo é o papel, este campo possui uma lista com todos os papéis cadastrados no sistema para que o usuário escolha um. O papel escolhido será o papel que o recurso humano, selecionado no quinto campo, irá executar na atividade. O quinto campo é o Recurso Humano, este campo possui uma lista com todos os recursos humanos cadastrados no sistema para que o usuário escolha um. O último campo é o Detalhes que tem por objetivo detalhar informações específicas do agente.

Ao clicar no botão "Salvar" os dados contidos nos campos serão enviados ao servidor e armazenados na base de dados. Após isto, o agente adicionado estará listado no componente apresentado na parte "2" da Figura 5.10.

Ao clicar no botão "Cancelar" a janela será fechada e os dados contidos nos campos não serão salvos.

5.8 Adicionar Artefatos a Uma Atividade

A Figura 5.12 mostra a tela de cadastro de artefatos utilizadas em uma atividade. Na parte "1" encontra-se um menu com três botões. O primeiro botão tem a função de atualizar a lista

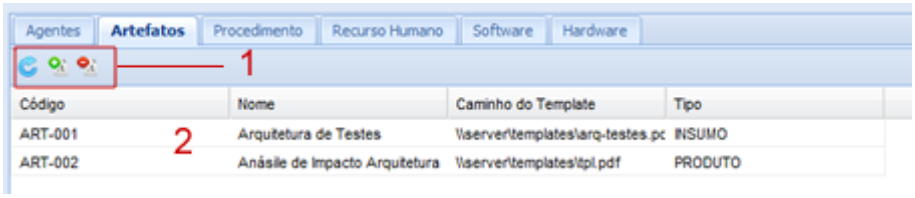

Figura 5.12: Tela lista de artefatos utilizados em uma atividade

de artefatos cadastrados na parte "2". O segundo botão tem a função de adicionar um novo artefato à atividade. Ao clicar neste botão será aberta uma janela como a que é apresentada na Figura 5.13. O terceiro botão tem a função de excluir um artefato, para isso um artefato deve ser previamente selecionado na parte "2", se nenhum artefato for selecionado e o botão for clicado o sistema exibirá um aviso ao usuário informando que nenhum agente foi selecionado.

Na parte "2" tem-se o componente responsável por listar todos os agentes que a atividade utiliza em sua execução.

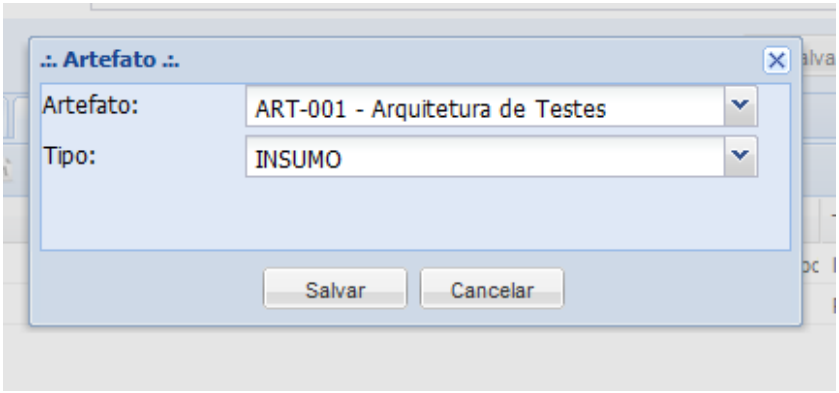

Figura 5.13: Tela adicionar artefato a uma atividade

A Figura 5.13 mostra a janela com o formulário para adicionar um artefato a uma tarefa. O campo Artefato possui uma lista com todos os artefatos cadastrados no sistema, sendo que o usuário deve escolher um. O campo Tipo possui uma lista com dois valores "INSUMO" e "PRODUTO", ao selecionar o primeiro valor o usuário está indicando que o artefato será consumido pela atividade e ao selecionar o segundo valor o usuário está indicando que o artefato será gerado pela atividade.

Ao clicar no botão "Salvar" os dados contidos nos campos serão enviados ao servidor e armazenados na base de dados. Após isto, o artefato adicionado estará listado no componente de listagem de artefatos utilizados na atividade.

Ao clicar no botão "Cancelar" a janela será fechada e os dados contidos nos campos não serão salvos.

5.9 Conclusões do Capítulo

Este capítulo apresentou os principais pontos de funcionamento do sistema, exemplificando e ilustrando as opções de navegação disponíveis ao usuário do sistema. Com este capítulo é possível instruir ao usuário a forma que ele poderá utilizar o sistema, deixando claras as opções de navegação para que o usuário entenda o sistema.

Capítulo 6

Conclusões e Trabalhos Futuros

6.1 Conclusões

Ter um processo de desenvolvimento bem definido facilita o desenvolvimento de sistemas complexos. Com o processo de desenvolvimento definido é possível que o mesmo seja analisado e como resultado desta análise, ele possa sofrer evoluções que o tornarão mais maduro e influenciarão na qualidade do produto final.

Este trabalho apresentou um sistema web para modelagem de processos de desenvolvimento de software. O sistema possibilita a definição de processos de desenvolvimento permitindo que sejam criados ou explicitados os padrões e processos utilizados no desenvolvimento de software. Possui recursos para definição dos itens componentes do processo e a partir da composição destes elementos é possível estruturar todo o processo, disponibilizando informações tais como: data de início e fim para cada uma das atividades, nome e email dos agentes que irão representar os papéis existentes no processo e, o caminho para o template e os recursos que devem ser utilizados para geração de cada artefato.

A utilização do sistema aqui implementado traz benefícios, como melhor comunicação entre as pessoas envolvidas no desenvolvimento, suporte à reutilização de modelos dos itens de processo e utilização de recursos.

Espera-se que, com o uso desta ferramenta, um gerente de projetos consiga modelar um processo de software de forma integrada, além de acompanhar todo o processo estando sempre ciente dos acontecimentos mesmo sem estar fisicamente junto dos demais integrantes da equipe. Espera-se também que os atores do processo tenham acesso ao conjunto de informações necessárias para execução de suas tarefas e com isso atendam as expectativas do gerente em relação a prazos e produtos gerados.

6.2 Trabalhos Futuros

Como todo produto de software, muito ainda pode ser feito em termos de evolução e melhorias. Novas necessidades podem surgir, bem como muitas outras já existem para ser supridas. Dentre as necessidades existentes que podem ser supridas em versões futuras estão:

- levar em conta questões relativas à usabilidade deste. Sendo assim é necessário melhorar as interfaces com o usuário e também disponibilizar no sistema telas de ajuda ao usuário;
- criar um módulo para cadastro de usuários do sistema. Este módulo deverá cadastrar os membros da equipe de desenvolvimento e dar estes acesso ao sistema. Dessa forma, cada membro poderá ter um perfil com permissões de acesso ao sistema, tais como, um usuário somente poderá visualizar, outro, poderá visualizar e editar os itens do processo.
- e o mais importante utilização do sistema em um caso de uso real para verificar os reais benefícios que ele pode trazer a uma empresa de desenvolvimento de software.

Referências Bibliográficas

Asleson, R. e Schttuta, N. T. (2006). *Fundamentos do Ajax*. Alta Books, 1 edição.

Bertollo, G. e Falbo, R. A. (2003). Apoio automatizado para definição de processos em níveis. *II Simpósio Brasileiro de Qualidade de Software, Fortaleza, Brasil*, 2:1–6.

Crockford, D. (2001). Javascript: A menos entendida linguagem de programação do globo! Disponível em: <http://javascript.crockford.com/pt/javascript.html>. Acessado em: 25 out. 2010.

DWR (2010). Direct web remoting - site oficial. <http://directwebremoting.org/dwr/index.html>. Acessado em: 25 out. 2010.

Falbo, R. A. (1998). *Integração de Conhecimento em um Ambiente de Desenvolvimento de Software*. PhD thesis, COPPE/UFRJ, Rio de Janeiro.

Falbo, R. A. (2006). Engenharia de software. Notas de Aula - 2005. Disponível em: <http://www.inf.ufes.br/falbo/download/aulas/es-g/2005-1/NotasDeAula.pdf > Acessado em: 15 out. 2010.

Filho, W. (2008). *Introdução ao Apache Maven*. Eteg Tecnologia da Informação - Disponível em: <www.eteg.com.br/treinamento/material/Eteg/underlineMaven0108.pdf> Acessado em: 15 out. 2010.

Garret, J. J. (2005). Ajax: A new approach to web applications. Adaptive Path Publications. Disponível em: <http://www.adaptivepath.com/publications /essays/archives/000385.php>. Acessado em: 31 set. 2010.

Graham, M. e LeBaron, M. (1994). *The horizontal revolution: reengineering your organization through teams*. Jossey Bass Managent Series.

Guice (2010). Google guice - documentation. Disponível em: <http://code.google.com/p/google-guice>. Acessado em: 15 out. 2010.

Júnior, V. N. P. (2003). Estratégias para a utilização da tecnologia j2ee com a arquitetura de cinco camadas. Master's thesis, UFMG, Belo Horizonte.

Loosley, C. (2006). Rich internet applications: Design, measurement, and management challenges. *Keynote Systems*, 15:1–15. Disponível em: <www.keynote.com/docs/whitepapers/RichInternet/underline5.pdf> Acessado em: 16 out. 2010.

Pressman, R. S. (2009). *Engenharia de Software*. MCGRAW-HILL BRASIL TECNICOS, 6 edição.

Rocha, A.; Maldonado, J. C. e Weber, K. C. (2001). *Qualidade de Software: Teoria e Prática*. Prentice Hall.

Rosa, E. (2009). Extjs: Um excelente framework de javascript. Documento Eletrônico Disponível em: <http://www.vivaolinux.com.br/artigo/ExtJS-Um-excelente-framework-de-JavaScript>. Acessado em: 25 out. 2010.

Silveira, R. W. R. (2005). Ajax: Aumentando a interatividade em aplicações web. Documento eletrônico disponível em: <http://www.dca.ufrn.br/raphaela/nataljavaday/apresentacoes/palestra7.pdf>. Acessado em 27 set. 2010.

Temple, A.; Mello, R. F.; Calegari, D. T. e Schiezaro, M. (2004). Jsp, servlets e j2ee. Documento eletrônico disponível em: <http://www.sifap.com.br/download/pooII/2503Jsp_Servlets-J2ee.pdf> Acessado em: 28 out. 2010.

Tomcat (2010). Apache tomcat 6.0 - documentation. Disponível em: <http://tomcat.apache.org/tomcat-6.0-doc/index.html>. Acessado em: 10 out. 2010.

W3C (2010). World wide web consortium. Disponível em: <http://www.w3c.org/>. Acessado em: 15 set. 2010.