

**UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO  
DEPARTAMENTO DE INFORMÁTICA  
MESTRADO EM INFORMÁTICA**

**BRUNO CARREIRA COUTINHO SILVA**

**PROCESSOS E FERRAMENTAS PARA O  
DESENVOLVIMENTO DE SOFTWARE LIVRE: UM  
ESTUDO DE CASO**

VITÓRIA  
2006

# **Livros Grátis**

<http://www.livrosgratis.com.br>

Milhares de livros grátis para download.

**BRUNO CARREIRA COUTINHO SILVA**

**PROCESSOS E FERRAMENTAS PARA O  
DESENVOLVIMENTO DE SOFTWARE LIVRE: UM  
ESTUDO DE CASO**

Dissertação apresentada ao Programa de Mestrado em Informática da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Mestre em Informática.

Orientador: Prof. Dr. Ricardo de Almeida Falbo.

VITÓRIA  
2006

**BRUNO CARREIRA COUTINHO SILVA**

**PROCESSOS E FERRAMENTAS PARA O  
DESENVOLVIMENTO DE SOFTWARE LIVRE: UM  
ESTUDO DE CASO**

**COMISSÃO EXAMINADORA**

---

**Prof. Dr. Ricardo de Almeida Falbo, D. Sc.  
Orientador**

---

**Prof. Dr. Sérgio Antônio Andrade de Freitas, D.Sc.  
Universidade Federal do Espírito Santo**

---

**Carlos Alberto Marques Pietrobon, D.Sc.  
Pontifícia Universidade Católica de Minas Gerais**

**Vitória, 03 de outubro de 2006.**

## RESUMO

O movimento de Software Livre tem ganhado cada vez mais espaço e importância nos segmentos da comunidade de software (governo, academia, indústria etc), tanto em âmbito mundial quanto nacional, contando atualmente com a existência de diversos projetos dessa classe em andamento. Esse tipo de software não traz consigo somente inovações na forma de se desenvolver software, mas também proporciona à comunidade uma nova filosofia, afetando muitos dos atuais princípios da indústria de software.

Apesar de seu notório crescimento, na maioria das vezes, seu desenvolvimento não tem sido realizado segundo as melhores práticas da Engenharia de Software, incluindo nesse cenário a não utilização de processos de software bem definidos. A elaboração desses processos pode ser facilitada se assistida por normas e modelos de qualidade de processo de software adequados.

A aplicação dos processos definidos a uma organização se torna mais viável se auxiliados por um bom ambiente de apoio ao desenvolvimento de software. No caso do desenvolvimento de Software Livre, esse ambiente deve ser composto por ferramentas preferencialmente disponíveis pela Internet, dada a dispersão geográfica dos colaboradores participantes de projetos desse tipo.

Este trabalho tem por objetivo definir uma infra-estrutura para apoiar o desenvolvimento de software livre a ser aplicada ao Projeto ODE (*Ontology-based software Development Environment*), dando origem ao Projeto ODE Livre. O Projeto ODE visa ao desenvolvimento de um Ambiente de Desenvolvimento de Software Centrado em Processos e é o principal projeto em andamento no Laboratório de Engenharia de Software (LabES) da Universidade Federal do Espírito Santo. A infra-estrutura proposta inclui processos padrão para software livre, bem como a definição de requisitos para a construção de um ambiente de apoio aos processos elaborados – o Portal ODE Livre.

**Palavras-chave:** Software Livre, Qualidade de Software, Ambiente de Desenvolvimento de Software e Ferramentas CASE.

## ABSTRACT

Free Software is more and more earning space in software market. Nowadays, there are several projects of this kind in progress around the world. This new software development model brings along a new philosophy, affecting many of the software industry principles.

Despite of its importance and growth, in most cases, free software development is not being done according to the best practices of Software Engineering. In this scenario, many times software processes are not formally defined.

This paper discusses an effort for defining a standard process for free software projects at LabES/UFES. The initial goal of defining these processes is to apply it in ODE's Project, a project that aims to develop a software engineering environment as a free software

The goal of this work is to define an infrastructure to support free software projects at LabES/UFES, which includes standard software processes for open source software projects, as well as the definition of requirements for the development of an environment that is able to support the processes defined. This infrastructure is to be applied to ODE Project, a project that aims to develop the software engineering environment ODE (Ontology-based software Development Environment) as a free software, giving rise to the Free ODE Project. ODE Project aims to develop a Process Centered Software Development Environment and it is the main project in progress in the Software Engineering Laboratory of the Federal University of Espírito Santo (LabES/UFES).

**Key words:** Free Source, Software Quality, Software Development Environment and CASE Tools.

## LISTA DE FIGURAS

|  |    |
|--|----|
| Figura 2.1 – Modelo para Definição de Processos em Níveis.....   | 08 |
| Figura 2.2 – Estrutura da ISO/IEC 12207 (Emendas 1 e 2).....   | 12 |
| Figura 2.3 – Processos do Exemplo de Modelo de Avaliação de Processos da<br>ISO/IEC 15504.....   | 18 |
| Figura 2.4 – Componentes do Modelo CMMI.....   | 21 |
| Figura 2.5 – Processos do MPS.BR.....  | 24 |
| Figura 2.6 – Estrutura do Processo Padrão para Equipes Distribuídas.....   | 33 |
| Figura 4.1 – Definição de Processos em Níveis e o Projeto ODE Livre.....   | 53 |
| Figura 4.2 – Estrutura Original dos Processos Padrão do LabES.....   | 54 |
| Figura 4.3 – Estrutura Básica dos Processos Padrão do LabES Atualizada.....  | 55 |
| Figura 4.4 – Detalhamento da atividade “Definir Processo de Projeto” do Processo<br>de Gerência de Projetos do Processo Padrão LabES.....                        | 56 |
| Figura 4.5 – Detalhamento da atividade “Definir Processo de Projeto” do Processo<br>de Gerência de Projetos do Processo Padrão LabES para Software<br>Livre..... | 62 |
| Figura 5.1 – Formulário de Defeitos do Bugzilla.....   | 67 |
| Figura 5.2 – Tela inicial do Controle de Consulta do Bonsai.....   | 69 |
| Figura 5.3 – Tela principal do Tinderbox.....  | 70 |
| Figura 5.4 – Exemplo de página gerada pelo LXR.....  | 71 |
| Figura 5.5 – Página inicial do Wiki-Mozilla.....   | 71 |
| Figura 5.6 – Tela principal de um Fórum Apache.....  | 73 |
| Figura 5.7 – Exemplo de utilização do Difftool do BitKeeper.....   | 74 |
| Figura 5.8 – Tipos de Usuários.....  | 76 |
| Figura 5.9 – Diagrama de Pacotes do Portal ODE Livre.....  | 77 |
| Figura 5.10 – Diagrama de Casos de Uso do subsistema de Controle de Usuários...  | 78 |
| Figura 5.11 – Diagrama de Casos de Uso do subsistema de Controle de<br>Contribuições.....  | 79 |

## LISTA DE TABELAS

|   |    |
|---|----|
| Tabela 2.1 – Níveis de Maturidade e Áreas de Processos do CMMI-SE/SW..... | 22 |
| Tabela 2.2 – Níveis de Maturidade e Processos do MPS.BR.....              | 25 |
| Tabela 2.3 – Relação entre MPS.BR e CMMI.....                             | 28 |



## SUMÁRIO

|   |    |
|---|----|
| <b>Capítulo 1 – Introdução</b> .....                            | 01 |
| 1.1 Contexto e Motivação.....                                   | 02 |
| 1.2 Objetivos.....  | 02 |
| 1.3 Metodologia.....  | 04 |
| 1.4 Organização do Trabalho.....                                | 05 |
| <br>  |    |
| <b>Capítulo 2 – Processo de Software</b> .....                  | 06 |
| 2.1 O que é Processo de Software.....                           | 07 |
| 2.2 Níveis de Processo de Software.....                         | 07 |
| 2.3 Qualidade de Processo de Software.....                      | 09 |
| 2.3.1 ISO 9000:2000.....  | 09 |
| 2.3.2 ISO/IEC 12207.....  | 11 |
| 2.3.3 ISO/IEC 15504.....  | 15 |
| 2.3.4 CMMI.....   | 19 |
| 2.3.5 MPS.BR.....   | 22 |
| 2.3.6 IEEE 1219-1998.....                                       | 28 |
| 2.4 Processo Padrão para Equipes Geograficamente Dispersas..... | 30 |
| 2.5 Considerações Finais do Capítulo.....                       | 33 |
| <br>  |    |
| <b>Capítulo 3 – Software Livre</b> .....                        | 35 |
| 3.1 O que é Software Livre.....                                 | 36 |
| 3.2 Licenciamento.....  | 37 |
| 3.3 Características de Desenvolvimento de Software Livre.....   | 39 |
| 3.4 Projetos de Software Livre.....                             | 42 |
| 3.4.1 O Projeto Linux.....                                      | 43 |
| 3.4.2 O Projeto Mozilla.....                                    | 43 |
| 3.4.3 O Projeto Apache.....                                     | 44 |
| 3.5 Conclusões do Capítulo.....                                 | 45 |

|   |           |
|---|-----------|
| <b>Capítulo 4 – Processos Padrão para Software Livre.....</b>                     | <b>46</b> |
| 4.1    Ambientes de Desenvolvimento de Software.....                              | 47        |
| 4.2    O Projeto ODE.....   | 49        |
| 4.3    Em Direção a ODE Livre.....  | 51        |
| 4.4    Processos Padrão do LabES.....   | 54        |
| 4.5    Processos Padrão para Projetos de Software Livre.....                      | 57        |
| 4.6    Licença para Disponibilização de ODE como Software Livre.....              | 62        |
| 4.7    Considerações Finais do Capítulo.....                                      | 63        |
| <br>  |           |
| <b>Capítulo 5 – Requisitos de um Ambiente de Apoio ao Projeto ODE Livre.....</b>  | <b>64</b> |
| 5.1    Características de Ambientes de Apoio ao Desenvolvimento de Software Livre | 65        |
| 5.1.1    O Projeto Mozilla.....   | 65        |
| 5.1.2    O Projeto Apache.....  | 73        |
| 5.1.3    O Projeto Linux.....   | 74        |
| 5.2    Requisitos de um Ambiente de Apoio ao Projeto ODE Livre.....               | 75        |
| 5.2.1    Requisitos Funcionais Iniciais para o Portal ODE Livre.....              | 75        |
| 5.2.2    Outros Requisitos Gerais Desejados para o Portal ODE Livre.....          | 80        |
| 5.2.3    Utilizando ODE no Projeto ODE Livre.....                                 | 82        |
| 5.3    Considerações Finais do Capítulo.....                                      | 83        |
| <br>  |           |
| <b>Capítulo 6 – Considerações Finais.....</b>                                     | <b>84</b> |
| 6.1    Conclusões.....  | 84        |
| 6.2    Perspectivas Futuras.....  | 86        |
| <br>  |           |
| <b>Referências Bibliográficas .....</b>   | <b>88</b> |

# Capítulo 1

## Introdução

O modelo de Software Livre (SL) tem despertado interesse e suscitado reflexões nos mais diversos segmentos da comunidade de software (governo, academia, indústria etc) no Brasil e no exterior. O surgimento de uma rede virtual de desenvolvedores e usuários, complexa, auto-organizada, com motivações diversas e a existência de novas formas de licenciamento de software sinalizam a introdução de novas variáveis no setor de software (SOFTEX, 2005a).

A recente pesquisa realizada pelo Observatório Econômico da Softex e o Departamento de Política Científica e Tecnológica da UNICAMP, com o apoio do Ministério de Ciência e Tecnologia, indica que, apesar de não se tratar de uma ruptura tecnológica, o modelo de SL traz uma nova forma de desenvolver e licenciar software que está quebrando modelos tradicionais de apropriabilidade e de desenvolvimento tecnológico (SOFTEX, 2005a).

Há, atualmente, um grande número de projetos de SL em desenvolvimento e este número é crescente (SOURCEFORGE, 2006). Dentre os projetos existentes, alguns possuem qualidade considerável. Porém, tal qualidade, muitas vezes, não é obtida através da utilização das melhores práticas de engenharia de software. É amplamente reconhecido que a qualidade dos produtos de software depende da qualidade dos processos de software utilizados em seu desenvolvimento e manutenção, e há diversas normas e modelos de qualidade, reconhecidos internacionalmente, tratando da questão da qualidade do processo de software.

Seguindo essa tendência de busca da qualidade do produto pelo uso de processos de software de qualidade, este trabalho procura explorar o tema qualidade de software livre por meio da definição de processos para projetos de software livre.

## **1.1 Contexto e Motivação**

O Laboratório de Engenharia de Software (LabES) da Universidade Federal do Espírito Santo (UFES) é um laboratório de pesquisa e ensino em Engenharia de Software que preza pela busca da qualidade em seus processos de desenvolvimento com vistas à qualidade de seus produtos. Dentre os projetos em desenvolvimento no LabES, destaca-se o Projeto ODE (FALBO et al., 2003), que visa ao desenvolvimento de um Ambiente de Desenvolvimento de Software (ADS) Centrado em Processo.

ODE é o resultado do esforço conjunto de diversos alunos de graduação e mestrado pertencentes ao LabES e possui ferramentas importantes para apoiar diversas atividades do processo de software. Dado seu estado atual e, principalmente, motivado por sua origem acadêmica, uma evolução natural para o Projeto ODE é a sua disponibilização à comunidade de software como software livre (SL), permitindo, assim, que essa comunidade possa desfrutar de um ADS livre. É importante frisar que a disponibilização de ODE para a comunidade será feita sob uma ou mais licenças de software, visando proteger o trabalho já realizado e possibilitar sua evolução.

Apesar de existirem projetos de software livre relatados na literatura que adotam processos utilizando boas práticas de engenharia de software e buscando a qualidade de seus produtos (REIS, 2003), o fato de não se ter encontrado nenhum projeto de SL que seguisse um processo de software baseado em normas e modelos de qualidade foi uma das principais motivações para este trabalho.

## **1.2 Objetivos**

Este trabalho aborda a temática qualidade de software no desenvolvimento de software livre, tendo como objeto de estudo o Ambiente de Desenvolvimento de Software ODE, a ser disponibilizado como software livre. De modo geral, o objetivo deste trabalho é o estabelecimento de processos de software padrão para o Projeto ODE Livre, tomando por base características de software livre e normas e modelos de qualidade reconhecidos, e a definição de requisitos para um arcabouço de ferramentas

livres de apoio aos processos definidos. A seguir, esse objetivo é descrito com mais detalhes na forma de sub-objetivos:

- Definir um processo padrão de desenvolvimento para o Laboratório de Engenharia de Software (LabES) a ser aplicado no desenvolvimento de software livre, em especial no Projeto ODE Livre. Esse processo deve ser estabelecido com base em normas e modelos de qualidade de processo, sobretudo ISO/IEC 12207 e MPS.BR.
- Definir um processo padrão de manutenção para o Laboratório de Engenharia de Software (LabES) a ser aplicado na manutenção de software livre, mais especialmente no Projeto ODE Livre. Assim como o processo padrão para desenvolvimento de software livre, esse processo deve ser estabelecido com base em normas e modelos de qualidade de processo, com destaque para a norma IEEE 1219.
- Adaptar outros processos padrão (de apoio e organizacionais) para o desenvolvimento e manutenção de software livre, a serem aplicados no Projeto ODE Livre. Esses processos padrão devem ser estabelecidos com base em normas e modelos de qualidade de processo, sobretudo ISO/IEC 12207 e MPS.BR.
- Os processos definidos devem levar em conta, também, projetos de software livre de sucesso, buscando identificar as melhores práticas adotadas nesses projetos.
- Tanto os produtos resultantes dos processos quanto as ferramentas de apoio às atividades, bem como os processos em si, devem ser livres, de modo a se poder contar posteriormente com a colaboração de pesquisadores de outras universidades e de centros de pesquisa, bem como de profissionais interessados no Projeto ODE Livre.
- Estabelecer requisitos de ferramentas de apoio aos processos padrão definidos para software livre, buscando apoiar de forma automatizada os processos de software propostos. Tais ferramentas devem ser integradas em um ambiente na Web.

### **1.3 Metodologia**

Após a definição dos objetivos iniciais deste trabalho, uma profunda e detalhada revisão bibliográfica da literatura foi efetuada no que tange aos conceitos de qualidade e processo de software, abrangendo o estudo de diversas normas e modelos de qualidade, a saber: ISO 9000, ISO/IEC 12207, ISO/IEC 15504, CMMI, IEEE 1219 e MPS.BR. Em paralelo, foram realizados estudos referentes aos conceitos de Software Livre e características de seu desenvolvimento, apoiando-se, principalmente, em exemplos de projetos de Software Livre bem sucedidos.

A partir desse levantamento, o escopo do trabalho foi revisado, chegando aos objetivos descritos na seção anterior, destacando-se o foco voltado ao Projeto ODE Livre como base para a definição dos processos.

Paralelamente ao levantamento bibliográfico, os processos padrão foram sendo definidos. Inicialmente, o processo padrão já definido para o LabES foi reformulado aos moldes propostos pelo MPS.BR. A partir daí, o processo padrão de desenvolvimento de software livre para o LabES foi definido, seguido do processo padrão de manutenção do LabES (até então inexistente) e do processo padrão LabES de manutenção de software livre.

Após a definição dos processos padrão, um artigo referente a este trabalho foi produzido e submetido ao V Simpósio Brasileiro de Qualidade de Software, com o objetivo principal de se ter um sentimento sobre a aprovação da comunidade em relação ao trabalho produzido. O artigo (SILVA et al., 2006) foi aceito, publicado e apresentado no referido evento, sendo que algumas considerações feitas pelos revisores foram incorporadas ao trabalho.

Finalmente, com o intuito de prover auxílio aos processos definidos, requisitos de um ambiente de apoio automatizado aos processos definidos foram levantados, contando, inicialmente, com a integração de algumas ferramentas livres ao Portal ODE Livre, em desenvolvimento (SILVA, 2006).

## 1.4 Organização da Dissertação

Esta dissertação contém, além deste capítulo que apresenta a *Introdução*, mais cinco capítulos.

O Capítulo 2 – Processo de Software – aborda o tema processo de software, discutindo níveis de processo e qualidade de processos software através de normas e modelos de qualidade, dentre eles ISO 9001, ISO/IEC 12207, CMMI, ISO/IEC 15504 e MPS.BR-MR. Este capítulo apresenta, ainda, um processo padrão proposto para equipes geograficamente dispersas.

O Capítulo 3 – Software Livre – apresenta o movimento de Software Livre, características de desenvolvimento de SL, além de apresentar alguns projetos de SL bem sucedidos.

O Capítulo 4 – Processos Padrão para Software Livre – inicia apresentando o objeto de estudo ao qual este trabalho se aplica, o Ambiente de Desenvolvimento de Software ODE. A seguir, é feita uma discussão sobre os processos padrão para software livre desenvolvidos para o LabES, organização responsável pelo Projeto ODE Livre.

O Capítulo 5 – Requisitos de um Ambiente de Apoio ao Desenvolvimento de Software Livre – discute aspectos relacionados ao apoio automatizado para apoiar os processos definidos no capítulo anterior, apresentado os primeiros esforços no sentido de construir o Portal ODE Livre, onde o ambiente ODE será disponibilizado para a comunidade e no qual se encontrarão muitas ferramentas de apoio aos processos propostos.

O Capítulo 6 – Considerações Finais – apresenta as conclusões deste trabalho e perspectivas futuras.

## Capítulo 2

### Processo de Software

Uma importante contribuição da área de pesquisa de processo de software tem sido a conscientização de que o desenvolvimento de software é um processo complexo. Pesquisadores e profissionais têm percebido que desenvolver software não está circunscrito somente à criação de linguagens de programação e ferramentas efetivas. O desenvolvimento de software é um processo coletivo, complexo e criativo. Sendo assim, a qualidade de um produto de software depende fortemente das pessoas, da organização e de procedimentos utilizados para criá-lo e disponibilizá-lo (FUGGETTA, 2000).

Dada a complexidade envolvida na definição de processos de software, não é uma boa estratégia definir cada processo de projeto a partir do zero. Assim, apesar de cada projeto ter suas características próprias, é possível estabelecer conjuntos de elementos que devem estar presentes em todos os processos de uma organização. Esses elementos em comum possibilitam a formação dos processos padrão da organização, que, por sua vez, podem ser especializados para determinadas classes de projetos dessa organização. Processos padrão e especializados podem, então, ser instanciados em processos de projeto, em uma abordagem de definição de processos em níveis (ROCHA et al., 2001). Idealmente, os elementos de um processo de software devem ser definidos segundo normas e modelos de qualidade, objetivando obter processos de qualidade.

Este capítulo visa a dar uma visão geral da área de processos de software e está estruturado da seguinte forma: a seção 2.1 discute o que é processo de software; a seção 2.2 apresenta os diferentes níveis de processo de software; a seção 2.3 trata da qualidade de processo de software e apresenta sucintamente os principais modelos e normas de qualidade de processo de software; na seção 2.4 são abordadas características de um processo padrão para equipes geograficamente dispersas; e finalmente, na seção 2.5 são apresentadas as considerações finais deste capítulo.



## 2.1 O que é Processo de Software

Um processo de software pode ser definido como um conjunto coerente de atividades, políticas, estruturas organizacionais, tecnologias, procedimentos e artefatos necessários para conceber, desenvolver, dispor e manter um produto de software (FUGGETTA, 2000). Um processo eficaz deve, claramente, considerar as relações entre as atividades, os artefatos produzidos no desenvolvimento, as ferramentas e os procedimentos necessários e a habilidade, o treinamento e a motivação do pessoal envolvido (FALBO, 1998).

A escolha de um modelo de ciclo de vida (ou modelo de processo) é o ponto de partida para a definição de um processo de desenvolvimento de software. Um modelo de ciclo de vida organiza as macro-atividades básicas, estabelecendo precedência e dependência entre as mesmas (FALBO, 1998).

Processos de software definem o conjunto de atividades conduzidas no contexto do projeto, tais como análise de requisitos, projeto, codificação, teste, instalação etc, os recursos (*software*, *hardware* e pessoas) necessários e os procedimentos a serem adotados na realização de cada uma das atividades. Sendo que essas atividades são compostas por outras atividades e podem se comunicar entre si e operam sobre artefatos de entrada para produzir artefatos de saída (FALBO, 1998) (GRUHN, 2002).

Outra questão que envolve a elaboração de um processo de software é a sua adequação ao domínio de aplicação e ao projeto. A cada projeto, o processo de software deve ser ajustado às especificidades da aplicação, tecnologia a ser utilizada na sua construção, grupo de desenvolvimento e usuários finais (FALBO, 1998).

## 2.2 Níveis de Processo de Software

Apesar de cada projeto ter suas características próprias, é possível estabelecer conjuntos de ativos de processo (sub-processos, atividades, sub-atividades, artefatos, recursos e procedimentos) comuns para toda a organização. Esses conjuntos constituem os chamados processos padrão de uma organização. A partir deles, outros processos, ainda padrão, podem ser especializados, levando-se em consideração características como tipos de software, paradigmas ou domínios de aplicação. Finalmente, para cada projeto, pode-se instanciar um processo padrão, especializado ou não, buscando atender

às suas características próprias, definindo os chamados processos de projeto (ROCHA et al., 2001) (BERTOLLO et al., 2006).

Esse modelo de definição de processos em níveis é adotado no Laboratório de Engenharia de Software (LabES) da UFES, como mostra a Figura 2.1. Os processos padrão para o desenvolvimento e manutenção de software do LabES são definidos tomando por base modelos e normas de qualidade, principalmente as normas ISO/IEC 12207 (ISO/IEC, 1998) e IEEE 1219 (IEEE, 1998) e o modelo MPS.BR (SOFTEX, 2006).

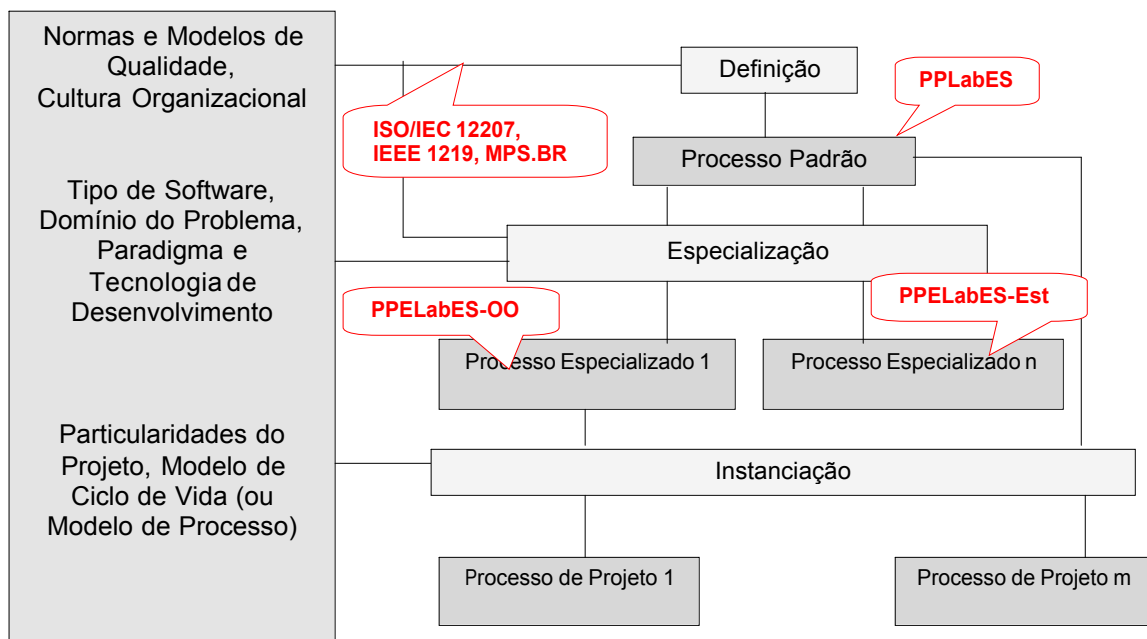


Figura 2.1 – Modelo para Definição de Processos em Níveis

A partir deles são especializados outros processos padrão de desenvolvimento e manutenção, tais como os Processos Especializados para o Desenvolvimento Orientado a Objetos (PPELabESOO) e para o Desenvolvimento segundo o Paradigma Estruturado (PPELabES-Est). Esses processos podem ser recursivamente especializados, criando outros níveis na hierarquia. Por exemplo, conforme discutido no capítulo 4 deste trabalho, a partir dos processos especializados para desenvolvimento e manutenção orientados a objetos, foram definidos processos especializados para desenvolvimento e manutenção de Software Livre no LabES, considerando a heterogeneidade dos grupos de trabalho, características do desenvolvimento de software com equipes geograficamente dispersas e práticas aplicadas a projetos de Software Livre bem sucedidos.

## **2.3 Qualidade de Processo de Software**

Quando se produz um software, assim como qualquer outro produto, é desejável que o mesmo possua elevada qualidade, que seja produzido de forma otimizada e dentro dos prazos estabelecidos previamente. No desenvolvimento de software, uma das principais maneiras de se garantir tais características para o produto final é através de um processo de software de qualidade. Ou seja, se um produto de software foi desenvolvido seguindo um processo de qualidade, as chances do mesmo ser um produto de qualidade são maiores.

Além disso, para que a organização que produz software tenha sua qualidade reconhecida em todo o mundo, seus processos devem respeitar padrões de qualidade definidos pela comunidade de software.

Desde a década de 1980, iniciaram-se esforços para melhoria de processos de software, com o objetivo de melhorar a qualidade, aumentar a produtividade e diminuir os custos. Diferentes modelos são utilizados dependendo do mercado alvo das organizações de software (ROCHA et al., 2001). As principais normas e modelos de qualidade difundidos atualmente são: ISO 9000 (ISO, 2000), ISO/IEC 12207 (ISO/IEC, 1998), ISO/IEC 15504 (ISO/IEC, 2003), CMMI (SEI, 2001) e MPS.BR (SOFTEX, 2006), sucintamente apresentados na seqüência juntamente com a norma IEEE 1219 (IEEE, 1998), que é voltada exclusivamente para manutenção.

### **2.3.1 ISO 9000:2000**

As normas da família NBR ISO 9000:2000 (ISO, 2000) foram desenvolvidas para apoiar organizações de todos os tipos e tamanhos (não só as de software), na implementação e operação de sistemas de gestão da qualidade eficazes.

A norma ISO 9000 descreve os fundamentos de sistemas de gestão da qualidade, que constituem o objeto da família NBR ISO 9000, e define os termos a ela relacionados (ISO, 2000).

Segundo XAVIER (2001), a família NBR ISO 9000 é composta de uma série de normas, mas somente as normas ISO 9001, 9002 e 9003 podem ser utilizadas como requisitos entre clientes e fornecedores. A ISO 9003 abrange somente requisitos para a garantia da qualidade em inspeção e ensaios finais. A ISO 9002 abrange requisitos para

a garantia da qualidade em produção, instalação e serviços associados. Já a ISO 9001 é a mais completa, abrangendo todo o ciclo de vida de um produto ou serviço, cobrindo os requisitos para a garantia da qualidade em projetos, desenvolvimento, produção, instalação e serviços associados (MUTAFELIJA et al., 2003).

Com a chegada da versão 2000 da ISO 9000, os requisitos da ISO 9001 foram reestruturados e as normas ISO 9002 e 9003 deverão ser canceladas (WEBER, 2001).

XAVIER (2001) considera que a grande modificação introduzida pela versão 2000 da ISO 9001 está na mudança do objetivo dos sistemas de gestão da qualidade, passando do atendimento aos requisitos especificados pelo cliente, para a satisfação do cliente, ou seja, para o atendimento não somente dos requisitos explícitos, especificados pelo cliente, mas também dos requisitos implícitos (WEBER, 2001).

A ISO 9000:2000 é baseada em um conjunto de princípios de gerenciamento de qualidade, definidos a partir da experiência de várias organizações (ISO, 2000):

- **Princípio 1 - Foco no cliente:** Dado que as organizações dependem de seus clientes, é recomendável que atendam às suas necessidades atuais e futuras e aos seus requisitos e procurem exceder as suas expectativas.
- **Princípio 2 - Liderança:** Líderes estabelecem a unidade de propósito e o rumo da organização. Convém que criem e mantenham um ambiente onde as pessoas estejam totalmente envolvidas no propósito de atingir os objetivos da organização.
- **Princípio 3 - Envolvimento de pessoas:** Pessoas de todos os níveis são a essência de uma organização. Seu envolvimento possibilita o aproveitamento de suas habilidades por toda a organização.
- **Princípio 4 - Abordagem de processo:** Um resultado desejado é alcançado mais eficientemente quando suas atividades e recursos são gerenciados como um processo.
- **Princípio 5 - Abordagem sistêmica para a gestão:** Identificar, entender e gerenciar os processos inter-relacionados como um sistema contribui para a eficácia e eficiência da organização no sentido de atingir os seus objetivos.

- **Princípio 6 - *Melhoria contínua***: Convém que a melhoria contínua do desempenho global da organização seja seu objetivo permanente.
- **Princípio 7 - *Abordagem factual para tomada de decisão***: Decisões eficazes são baseadas na análise de dados e informações.
- **Princípio 8 - *Benefícios mútuos nas relações com os fornecedores***: Pelo fato de organização e fornecedores serem interdependentes, uma relação de benefícios mútuos aumenta a capacidade de ambos em agregar valor.

Para que as organizações funcionem de forma eficaz, elas têm que identificar e gerenciar processos inter-relacionados e interativos. Frequentemente, a saída de um processo resultará diretamente na entrada do processo seguinte. A identificação sistemática e a gestão dos processos empregados na organização e, particularmente, as interações entre tais processos são conhecidos como “abordagem de processos”. Assim, a intenção desta norma é encorajar a adoção da abordagem de processo para a gerência da qualidade em uma organização (ISO, 2000).

### 2.3.2 ISO/IEC 12207

A norma internacional NBR ISO/IEC 12207 – Tecnologia da Informação – Processos de Ciclo de Vida de Software (ISO/IEC, 1998) estabelece uma estrutura comum para os processos de ciclo de vida de software, com terminologia bem definida, podendo ser referenciada pela indústria de software. Sua estrutura é composta de processos, atividades e tarefas, que podem ser aplicados durante a aquisição de um sistema que contém software, de um produto de software independente ou de um serviço de software, e também durante o fornecimento, desenvolvimento, operação e manutenção de produtos de software.

Recentemente foram feitas duas emendas à ISO/IEC 12207. A Emenda 1 (ISO/IEC, 2002) foi publicada em 2002 visando estabelecer um conjunto coordenado de informações de processo de software que pudesse ser utilizado para a definição, avaliação e melhoria de processos. A Emenda 1 resolveu o problema de granularidade relacionado ao uso da ISO/IEC 12207 para avaliação de processo e definiu propósitos e resultados esperados de um processo para estabelecer um Modelo de Referência de Processo, de acordo com os requisitos da norma ISO/IEC 15504-2 (ISO/IEC, 2003).

Um Modelo de Referência de Processo fornece definições de processos num ciclo de vida, descritas em termos de propósitos e resultados do processo, juntamente com uma arquitetura descrevendo a relação entre eles.

A utilização da Emenda 1 para avaliação de processos revelou defeitos técnicos e problemas editoriais em certos processos do Modelo de Referência de Processo. Os defeitos notados tiveram impacto sobre o desenvolvimento do modelo de exemplo de avaliação da ISO/IEC 15504-5 (ISO/IEC, 2004). A Emenda 2 resolve essas deficiências e fornece aos usuários do Modelo de Referência de Processo e aos desenvolvedores dos modelos de avaliação uma base melhorada para o seu trabalho (ISO/IEC, 2004).

Os processos do ciclo de vida do software estão agrupados em três classes nessa norma: Processos Fundamentais, Processos de Apoio e Processos Organizacionais, como mostra a figura 2.2. Os processos são compostos por atividades e estas são decompostas em tarefas.

| <b>Processos Fundamentais</b>    |                       | <b>Processos de Apoio</b>           |  | <b>Adaptação</b> |
|----------------------------------|-----------------------|-------------------------------------|--|------------------|
| Aquisição                        |                       | Documentação                        |  |                  |
| Fornecimento                     |                       | Gerência de Configuração            |  |                  |
| Desenvolvimento                  | Operação              | Garantia da Qualidade               |  |                  |
|                                  |                       | Verificação                         |  |                  |
|                                  |                       | Validação                           |  |                  |
|                                  |                       | Revisão                             |  |                  |
|                                  | Manutenção            | Auditoria                           |  |                  |
|                                  |                       | Usabilidade                         |  |                  |
|                                  |                       | Gerência de Resolução de Problemas  |  |                  |
|                                  |                       | Gerência de Solicitação de Mudanças |  |                  |
| Avaliação do Produto             |                       |                                     |  |                  |
| <b>Processos Organizacionais</b> |                       |                                     |  |                  |
| Gerência                         | Engenharia de Domínio | Melhoria                            |  |                  |
| Gestão de Ativos                 | Infra-estrutura       |                                     |  |                  |
| Gestão de Programa de Reúso      | Recursos Humanos      |                                     |  |                  |

Figura 2.2 – Estrutura da ISO/IEC 12207 (Emendas 1 e 2)

A seguir, os processos da norma, já contendo as alterações das emendas 1 e 2, são descritos segundo seu propósito (ISO/IEC, 2002) (ISO/IEC, 2004):

### **Processos Fundamentais**

1. *Processo de aquisição* – Obter um produto e/ou serviço que satisfaça a necessidade expressa pelo cliente. O processo inicia com a identificação de

uma necessidade do cliente e encerra com a aceitação do produto e/ou serviço.

2. *Processo de fornecimento* – Fornecer um produto ou serviço ao cliente que atenda aos requisitos acordados.
3. *Processo de desenvolvimento* – Transformar um conjunto de requisitos em um produto de software ou um sistema baseado em software que atenda às necessidades explicitadas pelo cliente.
4. *Processo de operação* – Operar o produto de software no seu ambiente e fornecer suporte aos clientes desse produto.
5. *Processo de manutenção* – Modificar um produto de software/sistema após sua entrega para corrigir falhas, melhorar o desempenho ou outros atributos, ou adaptá-lo a mudanças no ambiente.

### **Processos de apoio**

1. *Processo de documentação* – Desenvolver e manter registradas as informações do software produzidas por um processo.
2. *Processo de gerência de configuração* – Estabelecer e manter a integridade de todos os produtos de trabalho de um processo ou projeto e disponibilizá-los a todos os envolvidos.
3. *Processo de garantia da qualidade* – Fornecer garantia de que os produtos de trabalho e processos estão em conformidade com os planos e condições pré-definidos.
4. *Processo de verificação* – Confirmar que cada produto de trabalho de software e/ou serviço de um processo ou projeto reflete apropriadamente os requisitos especificados.
5. *Processo de validação* – Confirmar que são atendidos os requisitos de um uso específico pretendido para o produto de trabalho de software.
6. *Processo de revisão* – Manter um entendimento comum com os envolvidos (*stakeholders*) a respeito do progresso obtido em relação aos objetivos acordados e ao que deveria ser feito. Isso é realizado para ajudar a assegurar o desenvolvimento de um produto que satisfaça os envolvidos. As revisões conjuntas ocorrem nos níveis gerencial e técnico e são conduzidas durante o ciclo de vida do projeto.

7. *Processo de auditoria* – Determinar, de forma independente, a conformidade dos produtos e processos selecionados com os requisitos, planos e contratos, quando apropriado.
8. *Processo de usabilidade* – Garantir que sejam considerados os interesses e necessidades dos envolvidos de forma a proporcionar otimização do suporte e do treinamento, aumento da produtividade e da qualidade do trabalho, melhoria das condições para o trabalho humano e redução das chances de rejeição do sistema por parte do usuário.
9. *Processo de gerência de resolução de problemas* – Assegurar que todos os problemas identificados são analisados e resolvidos e que as tendências são identificadas.
10. *Processo de gerência de solicitação de mudanças* – Garantir que pedidos de mudanças são gerenciados, seguidos e controlados.
11. *Processo de avaliação de produto* – Garantir, por meio de exames e medições sistemáticos, que o produto atinge as necessidades indicadas e implícitas dos usuários do produto.

### **Processos organizacionais**

1. *Processo de gerência* – Organizar, monitorar e controlar a iniciação e a execução de qualquer processo de forma a atingir as suas metas, tendo por base as metas de negócio da organização.
2. *Processo de infra-estrutura* – Manter uma infra-estrutura estável e confiável, necessária para apoiar a execução de qualquer outro processo. A infra-estrutura pode incluir hardware, software, métodos, ferramentas, técnicas, padrões e instalações para o desenvolvimento, a operação ou a manutenção.
3. *Processo de melhoria* – Estabelecer, avaliar, medir, controlar e melhorar um processo de ciclo de vida de software.
4. *Processo de recursos humanos* – Fornecer à organização os recursos humanos adequados e manter as suas competências consistentes com as necessidades do negócio.
5. *Gerência de ativos* – Gerenciar a vida dos ativos reutilizáveis desde a sua concepção até a sua descontinuação.



6. *Gerência do programa de reutilização* – Planejar, estabelecer, gerenciar, controlar e monitorar esse programa em uma organização e sistematicamente explorar as oportunidades de reúso.
7. *Processo de Engenharia de Domínio* – Desenvolver e manter modelos, arquiteturas e ativos de domínio.

### **Processo de adaptação**

O Processo de adaptação possui atividades que permitem adaptar a norma para sua aplicação na organização ou em projetos. A adaptação deve ser executada baseando-se em alguns fatores que diferenciam uma organização ou projeto de outros, dentre eles a estratégia de aquisição, modelos de ciclo de vida, características de sistemas e software e cultura organizacional. Este processo permite que a norma seja adaptável a qualquer projeto, organização, modelo de ciclo de vida, cultura e técnica de desenvolvimento (ROCHA et al., 2001).

### **2.3.3 ISO/IEC 15504**

A norma internacional ISO/IEC 15504 (ISO/IEC, 2003) estabelece uma estrutura para a avaliação de processos. Essa estrutura pode ser usada por organizações envolvidas com planejamento, gerenciamento, monitoração, controle e melhoria de aquisição, fornecimento, desenvolvimento, operação, evolução e suporte de software. São propósitos dessa norma:

- ajudar organizações a compreender o estado de seus próprios processos com vistas à melhoria;
- ajudar organizações a determinar a adequação de seus processos para um requisito particular ou classe de requisitos;
- ajudar organizações a determinar a adequação de processos de uma outra organização para um contrato particular ou classe de contratos.

A norma consiste das seguintes partes, sob o título geral Tecnologia de Informação – Avaliação de Processo de Software (ISO/IEC, 2003):

- Parte 1: Conceitos e vocabulário (informativo): provê uma introdução geral aos conceitos de avaliação de processos e um glossário de termos relacionados a avaliação.
- Parte 2: Realizando uma avaliação (normativa): define os requisitos mínimos para a realização de uma avaliação.
- Um modelo de referência para processos e capacidade de processos
- Parte 3: Guia para a realização de avaliações (informativa): provê orientações para interpretar os requisitos para a realização de uma avaliação.
- Parte 4: Guia para uso na melhoria de processo e na determinação da capacidade de processo (informativa): identifica a avaliação de processo como uma atividade que pode ser realizada tanto como parte de uma iniciativa de melhoria de processo quanto parte de uma abordagem de determinação da capacidade.
- Parte 5: Um Exemplo de modelo de avaliação de processo baseado na ISO/IEC 12207 e suas Emendas 1 e 2 (informativa): contém um exemplo de modelo de avaliação de processo que é baseado no modelo de processo de referência definido na ISO/IEC 12207 e suas emendas 1 e 2.

A parte 5 da norma ISO/IEC 15504 desperta maior interesse para os interessados na definição de processo, por oferecer um modelo de referência disposto a guiar a definição de um processo de software de qualidade. De fato, o objetivo dessa parte da ISO/IEC 15504 é definir um exemplo de um Modelo de Avaliação de Processo que satisfaz os requisitos da ISO/IEC 15504-2 e que apóia a realização de uma avaliação, provendo indicadores para guiar a interpretação dos propósitos de processo, conforme definidos nas emendas 1 e 2 da ISO/IEC 12207, e dos atributos de processo definidos ISO/IEC 15504-2.

O Modelo de Avaliação de Processo nesta parte da ISO/IEC 15504 é dirigido a patrocinadores de avaliações e avaliadores competentes que desejem escolher um modelo para avaliação, seja para determinação da capacidade seja para melhoria de processo. Adicionalmente, esse modelo pode ser útil para os desenvolvedores de

modelos de avaliação, provendo exemplos de boas práticas de gerência e engenharia de software.

A Figura 2.3 lista os processos das emendas 1 e 2 da ISO/IEC 12207 que foram incluídos na dimensão de processos do exemplo de Modelo de Avaliação de Processo e mostra sua classificação em categorias e grupos de processos.

| Processos Primários  | Processos Organizacionais  |
|--|--|
| <b>Grupo de Processos de Aquisição</b><br>Preparação para Aquisição<br>Seleção de Fornecedor<br>Acordo de Contrato<br>Monitoração de Fornecedor<br>Aceite do Cliente   | <b>Grupo de Processos de Gerência</b><br>Alinhamento Organizacional<br>Gerência Organizacional<br>Gerência de Projeto<br>Gerência da Qualidade<br>Gerência de Risco<br>Medição |
| <b>Grupo de Processos de Fornecimento</b><br>Proposta do Fornecedor<br>Entrega do Produto<br>Suporte ao Aceite do Produto  | <b>Grupo de Processos de Melhoria de Processos</b><br>Estabelecimento de Processos<br>Avaliação de Processos<br>Melhoria de Processos  |
| <b>Grupo de Processos de Engenharia</b><br>Levantamento de Requisitos<br>Análise de Requisitos do Sistema<br>Projeto de Arquitetura do Sistema<br>Análise de Requisitos de Software<br>Projeto de Software<br>Construção de Software<br>Integração de Software<br>Testes de Software<br>Integração de Sistema<br>Testes de Sistema<br>Instalação de Software<br>Manutenção de Sistema e Software | <b>Grupo de Processos de Recursos e Infra-estrutura</b><br>Gerência de Recursos Humanos<br>Treinamento<br>Gerência de Conhecimento<br>Infra-estrutura                          |
| <b>Grupo de Processos de Operação</b><br>Uso Operacional<br>Suporte ao Cliente   | <b>Grupo de Processos de Reúso</b><br>Gerência de Ativos<br>Gerência do Programa de Reutilização<br>Engenharia de Domínio  |
| Processos de Apoio   |  |
| <b>Grupo de Processos de Controle de Configuração</b><br>Documentação<br>Gerência de Configuração<br>Gerência de Resolução de Problemas<br>Gerência de Solicitação de Mudanças   | <b>Grupo de Processos de Garantia da Qualidade</b><br>Garantia da Qualidade<br>Verificação<br>Validação<br>Revisão Conjunta<br>Auditoria<br>Avaliação do Produto               |

Figura 2.3 – Processos do Exemplo de Modelo de Avaliação de Processos da ISO/IEC 15504.

### 2.3.4 CMMI

O projeto CMMI (*Capability Maturity Model Integration*) (SEI, 2002) foi concebido para solucionar o problema do uso de múltiplos modelos de maturidade de capacitação, tendo como foco três modelos principais: SW-CMM (*Capability Maturity Model for Software*), SECM (*System Engineering Capability Model*) e IPD-CMM (*Integrated Product Development Capability Maturity Model*) (CHRISISS et al., 2003). É um projeto patrocinado pelo Departamento de Defesa Americano, em conjunto com a indústria, por meio do Comitê de Engenharia de Sistemas na Associação Industrial de Defesa Nacional, e o Instituto de Engenharia de Software (*Software Engineering Institute* – SEI). O projeto CMMI envolveu um grande número de pessoas de diferentes organizações de todo o mundo, interessadas nos benefícios do desenvolvimento de uma estrutura integrada em prol da melhoria de processos.

Apesar de prover um novo modelo, o CMMI procura preservar ao máximo os investimentos feitos em melhoria de processos baseadas no SW-CMM.

O objetivo do CMMI é fornecer direcionamentos para melhorar os processos de uma organização e sua capacidade de gerenciar o desenvolvimento, aquisição e manutenção de produtos e serviços. O CMMI provê abordagens comprovadas em uma estrutura que ajuda organizações a avaliar sua maturidade ou capacidade em determinada área de processo, estabelecer prioridades para melhoria e implementar essas melhorias.

É bom lembrar que o CMMI é direcionado para a melhoria de processos de organizações de qualquer tipo. Além disso, como outros modelos de maturidade de capacitação, os modelos do CMMI fornecem orientações a serem utilizadas no desenvolvimento de processos. Ou seja, esses modelos não são processos ou descrições de processos. Os processos reais utilizados em uma organização dependem de vários fatores, incluindo domínio de aplicação e tamanho e estrutura da organização. Assim, normalmente, as áreas de processo do modelo CMMI não podem ser mapeadas uma a uma em processos utilizados em uma organização (SEI, 2002).

O CMMI tem duas representações: *em estágio* e *contínua*. Ambas as representações contém áreas de processo comuns. Porém, na representação em estágio, as áreas de processo estão agrupadas em níveis de maturidade e na representação contínua, a mesma área de processo está dividida em categorias.

A representação contínua permite selecionar a seqüência de melhorias que melhor atende aos objetivos de negócio e reduz as áreas de risco da organização. Também permite comparações entre organizações em uma área de processo pela comparação de resultados utilizando estágios equivalentes (SEI, 2002).

A representação em estágios oferece uma seqüência comprovada de melhorias, começando com práticas básicas de gerência e progredindo por um caminho pré-definido e comprovado de níveis sucessivos, cada um servindo de base para o próximo. Além disso, permite comparações entre organizações pelo uso de níveis de maturidade (SEI, 2002).

Mesmo não sendo objetivo principal da representação contínua a classificação em um determinado nível de maturidade, e sim o desenvolvimento de determinadas áreas de processos, um determinado nível de maturidade pode ser atingido por quem usa a representação contínua, se todas as áreas de processo relevantes para tal nível tiverem atingido a capacidade mínima para o nível de maturidade.

Os componentes do Modelo CMMI incluem níveis de maturidade (*Maturity Levels*), áreas de processo (*Process Areas - PAs*), metas genéricas (*Generic Goals - GG*), metas específicas (*Specific Goals - SG*), práticas genéricas (*Generic Practices - GP*) e práticas específicas (*Specific Practices - SP*), como ilustra a Figura 2.4.

Uma área de processo é um grupo de práticas relacionadas em uma área que, quando executadas de forma coletiva, satisfazem um conjunto de metas consideradas importantes para trazer uma melhoria significativa naquela área. As áreas de processos do CMMI são as mesmas para ambas as representações (contínua e em estágios). Na representação em estágios, as áreas de processo estão organizadas por níveis de maturidade.

As metas específicas se aplicam a uma área de processo e contemplam características únicas que descrevem o que deve ser implementado para satisfazer tal área. Metas específicas são componentes exigidos do modelo e são utilizadas nas avaliações para auxiliar a determinar se a área de processo está sendo satisfeita.

Práticas específicas são atividades consideradas importantes na satisfação de suas metas específicas associadas. Descrevem atividades focadas no atendimento de metas específicas de uma área de processo. As práticas específicas são componentes esperados do modelo.

As metas genéricas são chamadas de “genéricas”, pois a mesma declaração de meta é encontrada em diversas áreas de processos. Na representação em estágios, cada

área de processo tem somente uma meta genérica. A satisfação de uma meta genérica em uma área de processo significa um controle melhorado do planejamento e implementação dos processos associados com aquela área de processo, indicando, portanto, se esses processos parecem ser eficientes, repetíveis e duráveis. As metas genéricas são componentes exigidos do modelo e são utilizadas em avaliações para determinar a satisfação de uma área de processo.

As práticas genéricas oferecem uma institucionalização que garante que os processos associados com a área de processo em questão serão eficientes, repetíveis e duráveis. As práticas genéricas são categorizadas pelas metas genéricas e características comuns e são componentes esperados em modelos CMMI.

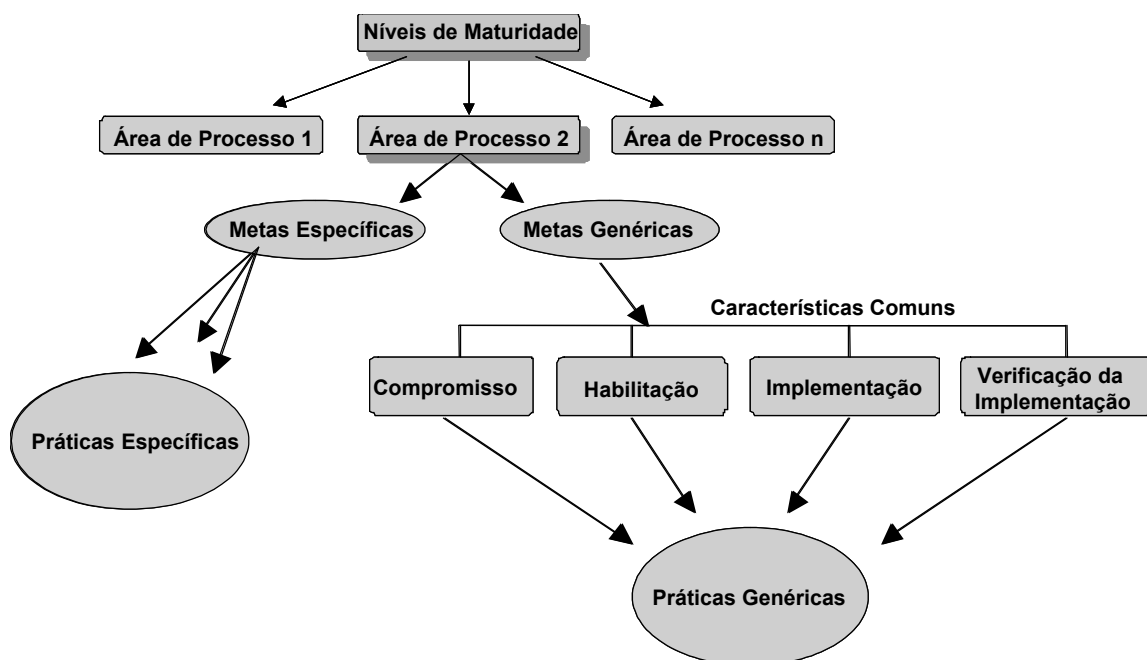


Figura 2.4 – Componentes do Modelo CMMI.

O nível de maturidade de uma organização é uma maneira de prever o futuro desempenho da mesma dentro de dada disciplina ou conjunto delas. A experiência mostra que as organizações funcionam melhor quando concentram seus esforços de melhoria de processos em um número controlado de áreas de processos que exigem esforço cada vez mais sofisticado à medida que a organização melhora.

Cada nível de maturidade, além de representar uma etapa evolucionária definida da melhoria de processos, estabiliza uma parte importante dos processos da organização.

Nos modelos CMMI com representação em estágios, existem cinco níveis de maturidade, cada um representando uma camada da base da melhoria de processos, definidos pelos números de 1 a 5. A Tabela 2.1 apresenta as áreas de processo do CMMI para Engenharia de Sistemas / Engenharia de Software, organizados em níveis de maturidade.

Tabela 2.1 – Níveis de Maturidade e Áreas de Processos do CMMI-SE/SW.

| Nível  | Área de Processo                      |
|--|---------------------------------------|
| 5 (mais alto)                                  | Inovação e Implantação na Organização |
|  | Análise de Causas e Resolução         |
| 4  | Desempenho do Processo Organizacional |
|  | Gerência Quantitativa do Projeto      |
| 3  | Análise de Decisão e Resolução        |
|  | Gerência de Riscos                    |
|  | Desenvolvimento de Requisitos         |
|  | Solução Técnica                       |
|  | Integração do Produto                 |
|  | Verificação                           |
|  | Validação                             |
|  | Treinamento Organizacional            |
|  | Foco no Processo Organizacional       |
|  | Definição do Processo Organizacional  |
|  | Gerência de Projeto Integrada         |
|  | 2                                     |
| Gerência de Configuração                       |                                       |
| Gerência de Acordo com Fornecedores            |                                       |
| Garantia da Qualidade do Processo e do Produto |                                       |
| Gerência de Requisitos                         |                                       |
| Planejamento de Projeto                        |                                       |
| Monitoração e Controle de Projeto              |                                       |

### 2.3.5 MPS.BR

O MPS.BR (SOFTEX, 2006) é um programa para Melhoria de Processo do Software Brasileiro, que está em desenvolvimento desde dezembro de 2003, sendo coordenado pela Associação para Promoção da Excelência do Software Brasileiro (SOFTEX) e com o apoio do Ministério da Ciência e Tecnologia (MCT),



Os objetivos principais do MPS.BR são (SOFTEX, 2006): (i) definir e aprimorar um modelo de melhoria e avaliação de processos de software, com foco preferencialmente nas micro, pequenas e médias empresas, de forma a atender suas necessidades de negócio e (ii) ser reconhecido nacional e internacionalmente como um modelo aplicável à indústria de software. Ele é aderente a modelos e normas internacionais, estando fortemente baseado, sobretudo, nas normas ISO/IEC 12207 e suas emendas 1 e 2, ISO/IEC 15504 e no CMMI-SE/SW. Além disso, o MPS.BR define regras para a sua implementação e avaliação, dando sustentação e garantia de que o modelo está sendo empregado de forma coerente com as suas definições (SOFTEX, 2006).

O MPS.BR baseia-se nos conceitos de maturidade e capacidade de processo para a avaliação e melhoria da qualidade e produtividade de produtos de software e serviços correlatos. Dentro desse contexto, o MPS.BR é decomposto em três partes: Modelo de Referência (MR-MPS), Método de Avaliação (MA-MPS) e Modelo de Negócio (MN-MPS) (SOFTEX, 2006).

O ponto inicial para a definição do MR-MPS (WEBER et al., 2004) foi a análise das empresas brasileiras, as normas ISO/IEC 12207 e ISO/IEC 15504-2 e o modelo CMMI.

A definição dos processos de software do MR-MPS segue a forma apresentada na emenda 1 da ISO/IEC 12207, declarando o propósito e os resultados esperados de sua execução. Isso permite avaliar e atribuir graus de efetividade na execução dos processos. Da mesma forma que a norma ISO/IEC 12207, o MR-MPS define processos fundamentais, processos de apoio e os processos organizacionais (mostrados na Figura 2.5) e um processo de adaptação. Cada empresa interessada em implantar o MR-MPS deve selecionar os processos pertinentes a partir desse conjunto, de acordo com o processo de adaptação, e deve definir as atividades e tarefas necessárias para atender ao propósito e aos resultados esperados de cada processo (SOFTEX, 2006).

O MR-MPS define sete níveis de maturidade: A (Em Otimização), B (Gerenciado Quantitativamente), C (Definido), D (Largamente Definido), E (Parcialmente Definido), F (Gerenciado) e G (Parcialmente Gerenciado). A escala de maturidade se inicia no nível G e progride até o nível A. Para cada um desses sete níveis de maturidade foi atribuído um perfil de processos e de capacidade de processos que indica onde a organização tem que colocar esforços para melhoria, de forma a atender aos seus objetivos de negócio (SOFTEX, 2006).

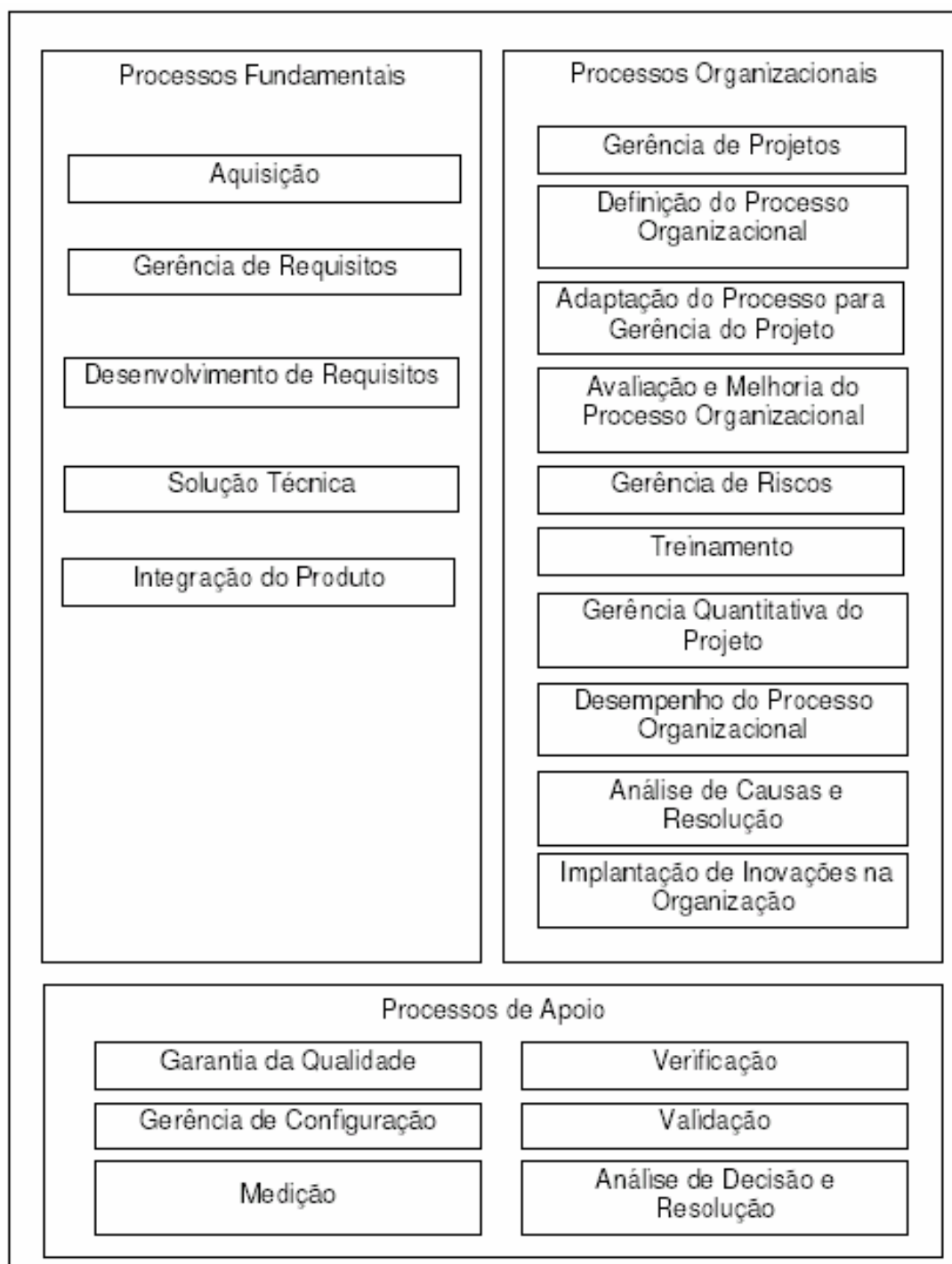


Figura 2.5 – Processos do MPS.BR

Os processos no MR-MPS são descritos em termos de seu propósito, resultados e informações adicionais. O propósito é o objetivo geral que deve ser atingido na execução do processo em questão. Os resultados esperados estabelecem os resultados a serem atingidos com a implementação efetiva do processo. As evidências desses resultados podem ser tanto artefatos produzidos quanto mudanças significativas na

execução do processo. As informações adicionais são referências que podem ajudar na definição do processo organizacional, fornecendo descrições de atividades, tarefas e melhores práticas, que podem apoiar a definição e implementação do processo nas organizações. A Tabela 2.2 mostra os processos correspondentes a cada nível de maturidade do MPS.BR e seus propósitos.

Tabela 2.2 – Níveis de Maturidade e Processos do MPS.BR.

| Nível                | Processos                               | Propósito de Processo   |
|----------------------|---|---|
| <b>A (mais alto)</b> | Implantação de Inovações na Organização | Selecionar e implantar melhorias incrementais e inovadoras que, de forma mensurada, melhorem os processos e as tecnologias da organização.  |
|                      | Análise de Causas e Resolução           | Identificar causas de defeitos e de outros problemas e tomar ações para prevenir suas ocorrências no futuro.  |
| <b>B</b>             | Desempenho do Processo Organizacional   | Estabelecer e manter um entendimento quantitativo do desempenho dos processos-padrão da organização para apoiar os objetivos para qualidade e para o desempenho dos processos. Fornecer dados, linhas-básicas ( <i>baselines</i> ) e modelos para gerenciar quantitativamente os projetos da organização. |
|                      | Gerência Quantitativa do Projeto        | Gerenciar quantitativamente o processo definido para o projeto de forma a alcançar os objetivos para qualidade e para o desempenho do processo estabelecido para o projeto.   |
| <b>C</b>             | Análise de Decisão e Resolução          | Analisar possíveis decisões usando um processo formal da avaliação das alternativas identificadas em relação a critérios estabelecidos.   |
|                      | Gerência de Riscos                      | Identificar, gerenciar e reduzir continuamente os riscos em nível organizacional e de projeto.  |

Tabela 2.2 – Níveis de Maturidade e Processos do MPS.BR (Continuação).

|          |   |  |
|----------|---|--|
| <b>D</b> | Desenvolvimento de Requisitos                   | Estabelecer os requisitos dos componentes do produto, do produto e do cliente.   |
|          | Solução Técnica                                 | Projetar, desenvolver e implementar soluções para atender aos requisitos.  |
|          | Integração do Produto                           | Compor os componentes do produto, chegando a um produto integrado consistente com o projeto ( <i>design</i> ), e demonstrar que os requisitos funcionais e não-funcionais são satisfeitos para o ambiente alvo ou equivalente. |
|          | Verificação                                     | Confirmar que cada serviço e/ou produto de trabalho do processo ou do projeto reflete apropriadamente os requisitos especificados.   |
|          | Validação                                       | Confirmar que um produto ou seu componente atenderá a seu uso pretendido quando colocado no ambiente para o qual foi desenvolvido.   |
| <b>E</b> | Treinamento                                     | Prover a organização e os projetos com profissionais que possuam os conhecimentos e as habilidades necessárias para executar suas funções de forma efetiva.  |
|          | Definição do Processo Organizacional            | Estabelecer e manter um conjunto de ativos dos processos organizacionais usáveis e aplicáveis às necessidades de negócio da organização.   |
|          | Avaliação e Melhoria do Processo Organizacional | Determinar o quanto os processos-padrão da organização contribuem para a organização a planejar e implementar melhorias contínuas nos processos com base no entendimento de seus pontos fortes e fracos.                       |
|          | Adaptação do Processo para Gerência do Projeto  | Estabelecer e gerenciar o projeto e envolver os interessados de acordo com o processo definido e integrado que é adaptado do conjunto de processos-padrão da organização.  |

Tabela 2.2 – Níveis de Maturidade e Processos do MPS.BR (Continuação).

|          |                          |  |
|----------|--------------------------|--|
| <b>F</b> | Medição                  | Coletar e analisar os dados relativos aos produtos desenvolvidos e aos processos implementados na organização e em seus projetos, de forma a apoiar os objetivos organizacionais.                  |
|          | Gerência de Configuração | Estabelecer e manter a integridade de todos os produtos de trabalho de um processo ou projeto e disponibilizá-los a todos os envolvidos.   |
|          | Aquisição                | Obter um produto e/ou serviço que satisfaça a necessidade do cliente.  |
|          | Garantia da Qualidade    | Garantir que os produtos de trabalho e a execução dos processos estão em conformidade com os planos e recursos predefinidos.   |
| <b>G</b> | Gerência de Requisitos   | Gerenciar os requisitos dos produtos e componentes do produto do projeto e identificar inconsistências entre esses requisitos e os planos e produtos de trabalho do projeto.                       |
|          | Gerência do Projeto      | Identificar, estabelecer, coordenar e monitorar as atividades, tarefas e recursos que um projeto necessita para produzir um produto e/ou serviço, com base nos requisitos e restrições do projeto. |

A Tabela 2.3 mostra a relação entre os processos do MPS.BR e as áreas de processo do CMMI, além de relacionar seus níveis de maturidade. O mapeamento obtido só é possível pelo fato do modelo MPS.BR ser fortemente baseado no CMMI.

Tabela 2.3 – Relação entre MPS.BR e CMMI.

| MPS.BR        |   | CMMI-SE/SW   |
|---------------|---|--|
| Nível         | Processo  | Área de Processo   |
| A (mais alto) | Inovação e Implantação na Organização           | Inovação e Implantação na Organização (CMMI-5)                                 |
|               | Análise de Causas e Resolução                   | Análise de Causas e Resolução (CMMI-5)   |
| B             | Desempenho do Processo Organizacional           | Desempenho do Processo Organizacional (CMMI-4)                                 |
|               | Gerência Quantitativa do Processo               | Gerência Quantitativa do Projeto (CMMI-4)                                      |
| C             | Análise de Decisão e Resolução                  | Análise de Decisão e Resolução (CMMI-3)  |
|               | Gerência de Riscos                              | Gerência de Riscos (CMMI-3)  |
| D             | Desenvolvimento de Requisitos                   | Desenvolvimento de Requisitos (CMMI-3)   |
|               | Solução Técnica                                 | Solução Técnica (CMMI-3)   |
|               | Integração do Produto                           | Integração do Produto (CMMI-3)   |
|               | Verificação                                     | Verificação (CMMI-3)   |
| E             | Validação                                       | Validação (CMMI-3)   |
|               | Treinamento                                     | Treinamento Organizacional (CMMI-3)  |
|               | Avaliação e Melhoria do Processo Organizacional | Foco no Processo Organizacional (CMMI-3)                                       |
|               | Definição do Processo Organizacional            | Definição do Processo Organizacional (CMMI-3)                                  |
| F             | Adaptação do Projeto para Gerência de Projetos  | Gerência de Projeto Integrada (CMMI-3)   |
|               | Medição   | Medição e Análise (CMMI-2)   |
|               | Gerência de Configuração                        | Gerência de Configuração (CMMI-2)  |
|               | Aquisição                                       | Gerência de Acordo com Fornecedores (CMMI-2)                                   |
| G             | Garantia da Qualidade                           | Garantia da Qualidade do Processo e do Produto (CMMI-2)                        |
|               | Gerência de Requisitos                          | Gerência de Requisitos (CMMI-2)  |
|               | Gerência de Projeto                             | Planejamento de Projeto (CMMI-2)<br>Monitoração e Controle de Projeto (CMMI-2) |

### 2.3.6 IEEE 1219-1998

A norma IEEE 1219 (IEEE, 1998) descreve um processo iterativo para gerenciar e executar atividades de manutenção de software, sendo que sua utilização não é restringida por tamanho, complexidade, nível crítico ou aplicação do produto de

software. Ela prescreve requisitos para o processo, o controle e a gerência do planejamento, da execução e da documentação das atividades de manutenção de software.

O modelo de processo básico descreve as fases do processo de manutenção, suas entradas, saídas e controles necessários. Vale realçar que ele não pressupõe o uso de qualquer modelo de ciclo de vida particular (ex. cascata, espiral etc.).

Segundo essa norma, um processo de manutenção de software deve incluir as seguintes fases:

- Identificação, classificação, e priorização do problema / modificação: as modificações de software são identificadas, classificadas e recebem uma posição inicial no ranking de prioridades. Cada solicitação de alteração deve ser avaliada para que se determine sua classificação e prioridade;
- Análise: nesta fase, informações do repositório e da solicitação de alteração validada na fase anterior, junto aos documentos do sistema e do projeto, são utilizadas para estudar a viabilidade e o escopo da modificação e para definir um plano preliminar para o projeto, implementação, teste e entrega. Além disso, uma análise detalhada é feita para definir os requisitos da modificação e para identificar e alterar os elementos necessários;
- Projeto: toda a documentação corrente do sistema e do projeto, software e bancos de dados existentes, bem como os resultados obtidos na fase anterior, devem ser utilizados para projetar a modificação no sistema;
- Implementação: nesta fase, os resultados da fase de projeto, o código fonte corrente e a documentação do sistema devem ser usados para guiar os esforços de implementação;
- Testes de regressão / sistema: testes de sistema devem ser realizados no sistema modificado. Testes de Regressão fazem parte dos testes de sistema e devem ser realizados para validar o código modificado quanto à introdução de falhas que não existiam antes da atividade de manutenção;
- Testes de aceitação: devem ser conduzidos em um sistema totalmente integrado. Testes de aceitação devem ser realizados pelo cliente, usuários do pacote de modificação ou uma terceira parte designada pelo cliente;
- Entrega: refere-se à entrega do sistema de software modificado e procedimentos associados.

## 2.4 Processo Padrão para Equipes Geograficamente Dispersas

O crescimento rápido e contínuo da Internet vem possibilitando uma forma de desenvolvimento de software antes não explorada: o desenvolvimento por equipes dispersas geograficamente. Esse tipo de desenvolvimento se mostra muito produtivo por impedir que a distância seja uma barreira para o desenvolvimento, dado que a comunicação, essencial para o desenvolvimento de software por equipes, tem um ótimo parceiro – a rede mundial de computadores.

Neste contexto, as atividades de software podem ser classificadas como *atividades assíncronas*, quando os desenvolvedores trabalham em diferentes horários com a mesma informação ou trocam mensagens e dados para coordenar suas tarefas; ou *atividades síncronas*, quando algumas tarefas, tais como reuniões, requerem discussões, em uma determinada hora, entre profissionais que podem ser responsáveis por diferentes aspectos do software (MAIDANTCHIK, 1999).

Pelo fato dos sistemas de computação serem desenvolvidos por várias equipes, normalmente, a interação e a comunicação entre esses grupos pode determinar o fracasso ou sucesso de um projeto. Uma equipe deve ser conduzida por um coordenador local, enquanto a gerência de projeto deve supervisionar todo o processo.

Os problemas encontrados no desenvolvimento de software por equipes distribuídas são comuns a outros contextos de desenvolvimento, porém são agravados quando os grupos de trabalho não se encontram num mesmo lugar. Tais problemas podem ser divididos nas seguintes classes (MAIDANTCHIK, 1999):

- **Coordenação de equipes:** refere-se à multidisciplinariedade e heterogeneidade das equipes;
- **Coordenação das atividades:** refere-se aos problemas de identificação de atividades concorrentes e formas de resolução de problemas de sincronismo;
- **Controle de artefatos:** envolve os problemas de integração de componentes, autoria distribuída, visibilidade dos resultados, notificação de mudanças e gerência de configuração;
- **Apoio à comunicação:** aplica-se à resolução de problemas, notificação de decisões, registro, acesso e compartilhamento de dados do software e publicação de informações que auxiliam o acompanhamento de atividades.



A qualidade dos produtos de software desenvolvidos por equipes geograficamente dispersas é fortemente dependente de uma comunicação efetiva, da coordenação dos grupos distribuídos, do rastreamento sistemático das atividades e artefatos, e da disponibilidade das informações do processo de software. Dadas essas características, os processos de software para equipes geograficamente dispersas possuem requisitos específicos, tais como (MAIDANTCHIK, 1999):

- Determinar a capacidade das equipes - O processo deve incorporar atividades para identificar o conhecimento, experiência e os recursos humanos, tecnológicos e econômicos de cada equipe. A determinação da capacidade dos grupos de trabalho deve ser executada antes de se iniciar um projeto de software, permitindo definir os papéis de cada membro da equipe e atribuir tarefas, de forma adequada, às diferentes equipes.
- Apoiar a coordenação distribuída - Por trabalhar em diferentes localidades, cada equipe deve ser supervisionada, localmente, por um coordenador, o qual responde diretamente à gerência do projeto. Tal requisito exige a rastreabilidade do processo e o acompanhamento das atividades técnicas pela coordenação local.
- Apoiar a gerência distribuída do projeto - É responsabilidade da gerência de uma organização monitorar a execução de todo o processo de software, através da supervisão das atividades relatadas pelos coordenadores locais. A gerência deve identificar tarefas concorrentes, determinar trabalhos cooperativos entre as diversas equipes e definir os pontos de controle do projeto.
- Apoiar o controle de artefatos - O processo deve incorporar atividades para acompanhar o desenvolvimento de artefatos, garantir que a sua produção siga os procedimentos pré-definidos pela gerência do projeto e administrar a integração com outros artefatos produzidos pelas diversas equipes.
- Apoiar a comunicação entre as equipes - É responsabilidade da organização identificar os recursos disponíveis para transmitir e receber mensagens e implantar mecanismos tradicionais ou eletrônicos para permitir a comunicação entre as equipes de software.
- Apoiar a publicação e o compartilhamento de informações - É responsabilidade da organização instalar repositórios e mecanismos de

acesso a todas as informações do software, tais como idéias, decisões, restrições e o estado do trabalho em execução. O processo deve incorporar procedimentos para informar mudanças realizadas por uma determinada equipe e que possam influenciar o trabalho de outras equipes.

- Apoiar a resolução de problemas - A análise e a resolução de problemas em ambientes distribuídos requer que o processo de software incorpore atividades para notificar a existência de um problema, divulgar a sua resolução e identificar as possíveis implicações no trabalho das outras equipes.
- Incorporar um repositório de terminologias - É responsabilidade da organização identificar, padronizar e documentar um glossário sobre o domínio da aplicação do projeto, garantindo a compreensão única dos termos pelas várias equipes.
- Apoiar a redefinição de atividades do software - É responsabilidade da organização identificar, redefinir e controlar a execução das atividades do processo de software, considerando as características e restrições das equipes dispersas. A organização deve estabelecer os procedimentos e a infraestrutura necessária para a realização de tarefas de forma distribuída, tais como gerência de configuração, especificação, projeto, codificação e testes, autoria e documentação, validações, verificações e revisões.

Seguindo a tendência de definição de processos em níveis, MAIDANTCHIK (1999) propôs um processo de software padrão definindo uma estrutura única a ser seguida por todos os grupos de trabalho em um desenvolvimento com equipes geograficamente dispersas que visa a atender aos requisitos citados anteriormente. Como base para construção desse processo, foi utilizada a norma ISO/IEC 12207.

Pelo fato de ser um processo destinado às organizações formadas por grupos geograficamente dispersos, o processo de aquisição foi substituído pelo processo de definição do problema e o processo de contratação, pelo processo de planejamento. Além disso, atividades específicas para o desenvolvimento distribuído foram adicionados ao processo padrão (MAIDANTCHIK, 1999).

De acordo com os requisitos citados anteriormente, é necessário definir um repositório com informações sobre as equipes de trabalho. Tal repositório deve conter avaliações das capacidades e características dos membros de cada grupo, e coleta de

dados e observações de projetos anteriormente realizados. A organização também deve estabelecer procedimentos para a gerência do projeto, coordenação de atividades das equipes, controle de artefatos, comunicação entre os profissionais e instalar a infraestrutura necessária à execução do processo de software (MAIDANTCHIK, 1999).

A Figura 2.6 mostra a estrutura do processo padrão para equipes geograficamente distribuídas proposto em (MAIDANTCHIK, 1999).

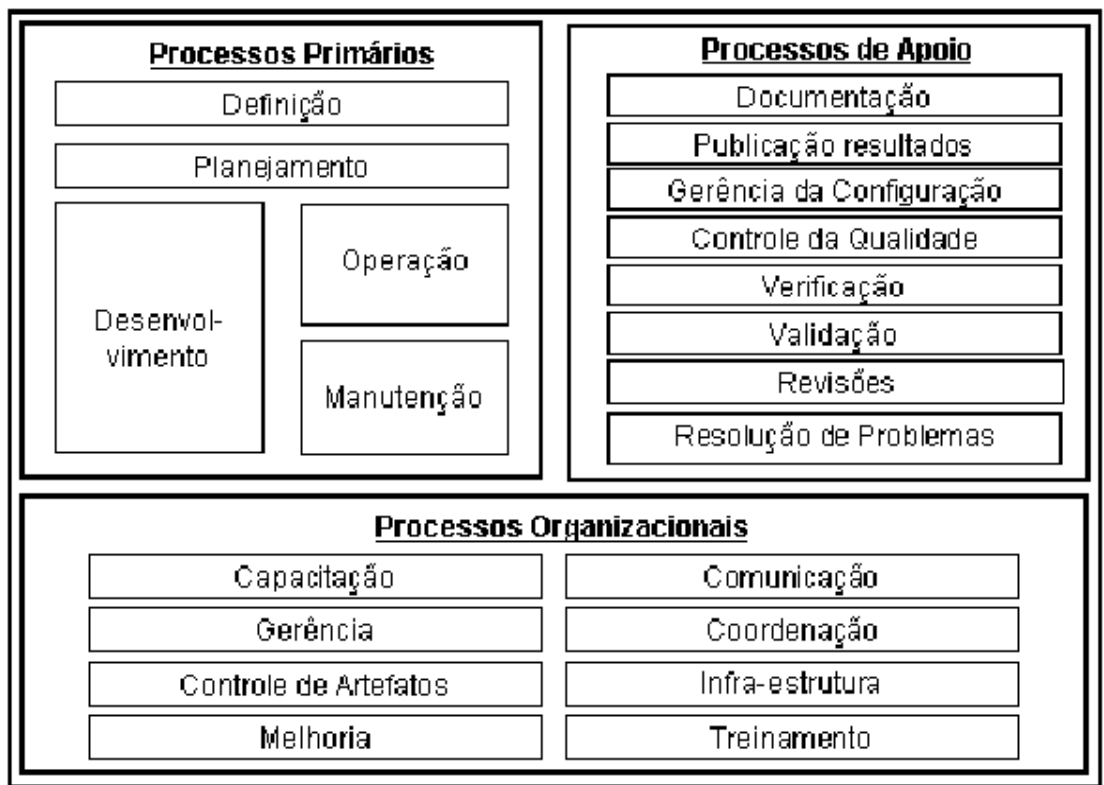


Figura 2.6 – Estrutura do Processo Padrão para Equipes Distribuídas (MAIDANTCHIK, 1999).

## 2.5 Considerações Finais do Capítulo

Por ser um dos principais objetivos deste trabalho a definição de processos padrão de qualidade para o Projeto ODE Livre, neste capítulo, procurou-se conceituar o que é processo de software e como esses processos podem ser definidos em níveis.

Vale destacar que este trabalho está inserido no contexto do Laboratório de Engenharia de Software (LabES) da UFES, que possui processos padrão e especializados já definidos, o que conduziu à especialização em outros níveis de processos padrão com o objetivo de possibilitar a criação de projetos de software livre

por essa organização. A partir desses processos especializados, instâncias deverão ser criadas por cada organização ou indivíduo que queira colaborar com o Projeto ODE Livre.

Pelo fato da qualidade ser um requisito fundamental para os processos a serem definidos, foram estudadas as principais normas e modelos de qualidade de processo vigentes. Além disso, outro ponto importante levantado neste capítulo foi a caracterização de um processo padrão para equipes geograficamente dispersas. Esse processo é uma referência importante para a definição de processos padrão para software livre por introduzir características comuns a esse tipo de software, tal como a dispersão geográfica dos desenvolvedores do sistema.

Assim, este capítulo apresentou o referencial teórico referente a processos de software utilizado como base para o desenvolvimento deste trabalho. A essa base, foram adicionadas características do desenvolvimento de software livre, descritas no próximo capítulo.

## Capítulo 3

### Software Livre

Alavancado, principalmente pela crescente utilização do Sistema Operacional livre Linux, o conceito de Software Livre (SL) ganhou espaço tanto no meio acadêmico quanto no mercado em geral. Uma das principais características de desenvolvimento deste tipo de software é a dispersão geográfica dos profissionais que o produzem. Além disso, produtos de Software Livre são caracterizados pela distribuição de seus códigos fonte correspondentes e pela utilização, na maioria das vezes, de mão-de-obra voluntária para seu desenvolvimento. Esses fatores podem fortalecer a desconfiança dos usuários desses produtos, necessitando, assim, que seu processo de desenvolvimento possua qualidade reconhecida em busca de maior credibilidade.

Conforme discutido no capítulo anterior, a qualidade do produto é fortemente dependente da qualidade do processo utilizado para seu desenvolvimento. Assim, o caminho pela busca da qualidade de SL passa por processos de software livre de qualidade. Como a definição de processos de software deve levar em conta as características dos produtos a serem desenvolvidos, é necessário considerar atentamente as características específicas de SL, sobretudo no que diferem em relação a produtos de software desenvolvidos de maneira tradicional.

O objetivo deste capítulo é explorar os conceitos envolvidos em Software Livre, licenças aplicáveis a tais produtos de software, características específicas de seu desenvolvimento e apresentar exemplos de projetos de Software Livre que têm alcançado sucesso.

O capítulo está estruturado da seguinte forma: a seção 3.1 define o que é Software Livre; a seção 3.2 mostra a importância do licenciamento para produtos de software deste tipo e apresenta as principais licenças aplicadas; na seção 3.3 características próprias do desenvolvimento de Software Livre são exploradas; a seção 3.4 descreve brevemente alguns dos principais projetos de Software Livre; e, finalmente, na seção 3.5 são apresentadas as considerações finais deste capítulo.

### 3.1 O que é Software Livre

Software Livre - SL (*Free Software*) é o software disponível com a permissão para qualquer um usá-lo, copiá-lo e distribuí-lo, seja na sua forma original ou com modificações, seja gratuitamente ou com custo. Em especial, a possibilidade de modificações implica que o código fonte esteja disponível (HEXSEL, 2002).

SL é uma questão de liberdade, não de preço. Mais precisamente, se refere a quatro liberdades para seus usuários, a saber (FREE SOFTWARE FOUNDATION, 2006):

0. Liberdade de executar o programa, para qualquer propósito;
1. Liberdade de estudar como o mesmo funciona e adaptá-lo para as suas necessidades;
2. Liberdade de redistribuir cópias de modo a ajudar ao próximo;
3. Liberdade de aperfeiçoar o programa e liberar os seus aperfeiçoamentos, de modo que toda a comunidade se beneficie.

As liberdades 1 e 3 implicam na necessidade de se ter, pelo menos, acesso ao código-fonte, apontado como a principal característica de produtos de software livres.

Na década de 1980, a idéia de SL se popularizou, com a criação da *Free Software Foundation – FSF*, por Richard Stallman. O movimento de SL nasceu de uma contestação aos mercados proprietários mais poderosos da indústria de software, revelando todo um apelo político, institucional e emocional (SOFTEX, 2005a), e é hoje um movimento sólido.

Produtos de software livres apresentam algumas características vantajosas peculiares, muitas vezes, resultantes do próprio processo de desenvolvimento dessa classe de produtos de software (NAKAGAWA, 2002), incluindo estabilidade, portabilidade para uma variedade de plataformas, acesso ao código-fonte, suporte aos usuários por parte dos desenvolvedores, baixo custo e evolução contínua, com versões mais novas do software sendo lançadas com maior frequência e defeitos sendo corrigidos em um tempo menor do que em softwares proprietários (DAVIS et al., 2000).

Pelo seu caráter cooperativo, propiciado principalmente pela distribuição do código fonte, o desenvolvimento de SL é feito por colaboradores dispersos

geograficamente, muitas vezes, ao redor do mundo. Por todas essas características, SL exige uma abordagem de desenvolvimento diferente da tradicionalmente empregada.

### 3.2 Licenciamento

O software pode ser considerado um produto como qualquer outro e, portanto, a única forma de garantir sua proteção é através do *copyright*. Uma pessoa que escreve um programa de computador detém o *copyright* sobre o mesmo. Isso permite que o autor controle a cópia, o uso e a adaptação do mesmo. Para disponibilizar o software para terceiros, o autor deve dar autorização a eles para certas operações. Isso é chamado de licença. Dependendo dos desejos do autor, diferentes licenças padrão podem ser adotadas, ainda que o autor possa também escrever sua própria licença (ENGELFRIET, 2003). Assim, a licença de software é um documento que estabelece a forma como o proprietário do *copyright* permite a distribuição e cópia de um software.

É importante frisar que todo software livre deve ser registrado sob uma licença de software para que sua distribuição seja corretamente controlada e suas liberdades sejam protegidas, não ganhando, assim, características de um software de domínio público e evitando que abusos legais sejam cometidos. Existem várias licenças disponíveis para o registro desses softwares, tendo cada uma delas características próprias de distribuição (HEXSEL, 2002), a saber:

- **GPL:** A Licença Pública Geral GNU (*GNU General Public License - GPL*) é a licença que acompanha os pacotes distribuídos pelo Projeto GNU e mais uma gama de produtos de software, incluindo o núcleo do sistema operacional Linux. A formulação da GPL é tal que, ao invés de limitar a distribuição do software por ela protegido, ela, de fato, impede que esse software seja integrado a software proprietário. A GPL é baseada na legislação internacional de *copyright*, o que garante cobertura legal para o software licenciado com a GPL.
- **Debian:** A licença Debian (*Debian Free Software Guidelines - DFSG*) é parte do contrato social celebrado entre o Projeto Debian e a comunidade de usuários de software livre. O Projeto Debian é responsável por uma distribuição de um Sistema Operacional Livre que contém o Linux como *kernel*, possuindo diversos pacotes próprios incorporados. Em essência, essa

licença contém critérios para a distribuição que incluem, além da exigência da publicação do código fonte, os seguintes: (a) a redistribuição deve ser livre; (b) o código fonte deve ser incluído e deve poder ser redistribuído; (c) trabalhos derivados devem poder ser redistribuídos sob a mesma licença do original; (d) pode haver restrições quanto à redistribuição do código fonte, se o original foi modificado; (e) a licença não pode discriminar qualquer pessoa ou grupo de pessoas, nem quanto a formas de utilização do software; (f) os direitos outorgados não podem depender da distribuição onde o software se encontra; e (g) a licença não pode 'contaminar' outro software.

- **Open Source:** A licença do *Open Source Initiative* é derivada da Licença Debian, com as menções à Debian removidas.
- **BSD:** A licença BSD cobre as distribuições de software da *Berkeley Software Distribution*, além de outros programas. Essa é uma licença considerada 'permissiva', porque impõe poucas restrições sobre a forma de uso, alterações e redistribuição do software licenciado. O software pode ser vendido e não há obrigações quanto à inclusão do código fonte, podendo, até mesmo, o mesmo ser incluído em software proprietário. Essa licença garante o crédito aos autores do software, mas não tenta garantir que trabalhos derivados permaneçam como software livre.
- **Creative Commons:** É um projeto que disponibiliza opções flexíveis de licenças que garantem proteção e liberdade para artistas e autores de obras intelectuais, não se limitando a software. O *Creative Commons* define um espectro de possibilidades entre o direito autoral total – *todos os direitos reservados* – e o domínio público – *nenhum direito reservado*, ou seja, permite ao autor manter seu direito autoral e, ao mesmo tempo, permite certos usos de sua obra – um direito autoral de “alguns direitos reservados”. Existem seis licenças principais nesse projeto, sendo que a mais restritiva impede a distribuição comercial e a modificação da obra, e a menos restritiva exige somente que sejam atribuídos os devidos créditos ao autor da obra (CREATIVE COMMONS, 2006).



### 3.3 Características de Desenvolvimento de Software Livre

Pelo seu caráter cooperativo, propiciado principalmente pela distribuição do código fonte dos programas, o desenvolvimento de Software Livre é feito por colaboradores dispersos geograficamente, na maioria das vezes, ao redor do mundo. Essa forma de desenvolvimento exige uma abordagem de desenvolvimento diferenciada.

O modelo de desenvolvimento de software tradicionalmente utilizado para produzir software proprietário ou acadêmico é chamado por RAYMOND (1999) de Modelo Catedral. Sua característica principal é sua forma centralizada e controlada de desenvolver o software e a necessidade de uma pessoa (ou grupo de pessoas) que centralize o processo de desenvolvimento.

Quando pensamos em software livre, o Modelo Catedral não é adequado para seu desenvolvimento, principalmente pela dispersão geográfica dos inúmeros desenvolvedores. Devido a esse fato, RAYMOND (1999) propôs a adoção de um modelo denominado Modelo Bazar para o desenvolvimento de SL, incluindo características peculiares, tais como: disponibilização de código fonte; envolvimento de diversos desenvolvedores, muitas vezes, voluntários; dispersão geográfica dos mesmos em um processo colaborativo; tempo curto de desenvolvimento; e liberdade para os desenvolvedores escolherem o trabalho que desejam realizar. Vale apontar que esse modelo requer uma pessoa (ou grupo de pessoas) centralizando o processo.

O Modelo Bazar tem sido aplicado com sucesso em diversos projetos, tais como os projetos do Linux e do Apache, apresentados na seção 3.4. Observa-se, contudo, que para se obter sucesso na sua utilização, é necessário, ainda, ter (NAKAGAWA, 2002): um objetivo bem definido do que será desenvolvido; uma motivação fácil de ser compreendida; um bom líder que mantenha os desenvolvedores motivados e o objetivo em mente; uma comunidade de participantes que trabalhe com entusiasmo e de forma descentralizada; e uma tecnologia que possibilite a comunicação de forma eficiente. Assim, o sucesso do Modelo Bazar não depende somente da disponibilização do código fonte, sendo necessário organizar e coordenar todo o processo de forma apropriada.

Apesar de interessante e muito útil, o Modelo Bazar é apenas uma referência inicial para um processo de software livre, principalmente quando confrontado com normas e modelos de qualidade, tal como a ISO/IEC 12207. Sob a ótica dessas normas e modelos, um processo de software pode ser definido como um conjunto de atividades,

métodos, práticas, estruturas organizacionais, tecnologias e artefatos que são necessários para conceber, desenvolver, entregar e manter um produto de software (FUGGETA, 2000). Assim, um processo eficaz deve considerar claramente as relações entre as atividades, os artefatos produzidos no desenvolvimento, as ferramentas e os procedimentos necessários e a habilidade, o treinamento e a motivação do pessoal envolvido.

Claramente, o Modelo Bazar não chega a esse nível de definição. Na verdade, RAYMOND (1999) estava focado nas estruturas organizacionais e características de um projeto de SL, não estando diretamente preocupado com atividades, artefatos e procedimentos que um processo deveria seguir. Até porque, processos têm de ser definidos caso a caso, levando em consideração características específicas do projeto em questão, incluindo as tecnologias envolvidas, o tipo de software a ser desenvolvido, o domínio da aplicação e a equipe de desenvolvimento (ROCHA et al., 2001).

REIS (2003) procurou levantar e quantificar aspectos essenciais do processo de software utilizado nos projetos de software livre, discutindo questões como: De que forma os projetos de software livre trabalham para produzir software? Existe um processo comum entre os projetos de software livre? De que forma esses processos diferem do processo de software convencional documentado na literatura de engenharia de software? Dentre as conclusões a que ele chegou em sua pesquisa, podem-se destacar:

- C1. Na maioria dos projetos de SL, a equipe trabalha de maneira descentralizada, envolvendo, muitas vezes, desenvolvedores que não se conhecem pessoalmente. Vale destacar, ainda, que nos últimos anos estão ocorrendo mudanças neste perfil, com organizações investindo na contratação de desenvolvedores para projetos de software livre.
- C2. Em grande parte dos projetos de SL, os desenvolvedores são também usuários do produto em desenvolvimento e contribuem efetivamente para determinar sua funcionalidade.
- C3. Existe, na maior parte dos projetos de SL, um sistema de liderança, sendo as formas mais utilizadas: líder com delegação informal, líder com delegação formal e comitê.

- C4. Há um uso amplo de ferramentas para viabilizar a comunicação entre membros da equipe geograficamente dispersa, com destaque para ferramentas de controle de versão e listas de correio eletrônico, além de sistemas de acompanhamento de alteração / defeitos, serviço de hospedagem de projetos e ferramentas de comunicação em tempo real. No que tange a ferramentas de desenvolvimento, sua utilização aparece majoritariamente centrada em ferramentas de implementação.
- C5. A maior parte dos projetos de SL possui equipe pequena.
- C6. Há indícios fortes do uso de conhecimento pré-existente para estabelecer os requisitos do projeto, seja por meio da reutilização de levantamentos anteriores feitos por outras equipes, através de padrões estabelecidos ou documentados, seja por replicar de maneira significativa um outro produto de software.
- C7. Parece existir muito pouco processo sistemático em relação à qualidade, incluindo testes. Não há evidências de uma política forte de controle e revisão relacionada à aceitação, integração e auditoria de contribuições. Contudo, ter seu trabalho constantemente sob olhar público, parece forçar os desenvolvedores a encararem sua tarefa com atenção e padrão de qualidade superiores. Assim, mesmo que o código não seja revisado e auditado com frequência, o fato de poder ser avaliado – e publicamente criticado – parece influenciar significativamente a atitude dos desenvolvedores na construção de SL.
- C8. Pouca atenção é dada à usabilidade do software, o que, muitas vezes, leva produtos de software livres a ter uma reputação de complexo ou difícil de usar.

Além das conclusões de Reis sobre características de projetos de software livre, vale a pena destacar outras considerações relatadas na literatura:

- C9. À primeira vista não é dada muita ênfase à especificação de requisitos (SCACCHI, 2002), seja porque a maioria dos projetos de software livre replica um software já existente ou por partirem de uma motivação pessoal do autor, trazendo requisitos implícitos (RAYMOUND, 1999). O conceito

de um “documento de especificação de requisitos” parece raro entre a comunidade de software livre, embora, existam especificações formais descrevendo partes restritas de alguns sistemas mais complexos (REIS, 2003).

- C10. Na fase de projeto, os modelos são de mais alto nível, não sendo possível identificar uma arquitetura bem definida do sistema (VIXIE, 1999). Segundo REIS (2003), existe pouca documentação formal de projeto e é provável que grande parte do trabalho de descrever o projeto fique a cargo do próprio código-fonte.
- C11. A melhor estratégia de lançamento de software livre é disponibilizá-lo o quanto antes, de modo que ocorra um crescimento evolutivo por meio de sugestões e críticas dos usuários. Contudo, para que o software seja lançado, deve apresentar funcionalidade interessante o suficiente para atrair atenção (RAYMOUND, 1999).
- C12. Para que se alcance o sucesso, os participantes de projetos de software livre precisam ter sua contribuição reconhecida.
- C13. Projetos de software livre, geralmente, não são alvo de pressões referentes a prazos, propiciando maior tranquilidade aos colaboradores e trabalhos mais elaborados. Além disso, prazos não podem ser estabelecidos, pois é um trabalho voluntário.
- C14. A escolha, por parte dos desenvolvedores, do trabalho que desejam realizar, pode proporcionar sua motivação.

As características citadas acima são o resultado de um apanhado geral em torno dos principais projetos de software livre disponíveis na *Web*. Alguns desses projetos são sucintamente apresentados na próxima seção.

### **3.4 Projetos de Software Livre**

Existem milhares de projetos de software livre espalhados pelo mundo, sendo alguns ativos e outros inativos, alguns bem sucedidos e outros não. Esta seção descreve

alguns dos projetos de software livre que se destacam pela qualidade, grande número de usuários e colaboradores, a saber os projetos Linux (LINUX, 2006), Mozilla (MOZILLA, 2006) e Apache (APACHE, 2006). Contudo há muitos outros projetos importantes, tais como os projetos relacionados ao desenvolvimento dos bancos de dados MySQL (MYSQL, 2006) e PostgreSQL (POSTGRESQL, 2006) e das linguagens de programação PHP (PHP, 2006) e Perl (PERL, 2006)

### **3.4.1 – O Projeto Linux**

O Linux é um sistema operacional que foi inicialmente criado como um hobby por um estudante jovem, Linus Torvalds, na universidade de Helsinque na Finlândia. Linus tinha interesse no Minix, um pequeno sistema UNIX, e decidiu desenvolver um sistema que excedia os padrões do Minix. Começou seu trabalho em 1991 e em 1994 liberou a versão 1.0 do *Kernel* do Linux. O *Kernel*, coração de todos os sistemas Linux, é desenvolvido e liberado sob a licença GNU-GPL, tornando seu código fonte disponível livremente a qualquer um. Atualmente existem centenas de empresas, organizações e indivíduos que disponibilizam suas próprias versões de sistemas operacionais baseados no *Kernel* do Linux. A versão em que o Linux se encontra atualmente é a de número 2.6 e o desenvolvimento continua (LINUX, 2006).

Além do fato de ser distribuído gratuitamente, a funcionalidade, adaptabilidade e robustez do Linux têm feito dele a principal alternativa aos sistemas operacionais proprietários da Microsoft e UNIX. A IBM, a HP e outros gigantes da computação têm adotado o Linux e têm dado suporte ao seu desenvolvimento.

### **3.4.2 – O Projeto Mozilla**

O Projeto Mozilla é um projeto livre dedicado ao desenvolvimento de ferramentas relacionadas ao navegador *Web* Mozilla. Desde a sua criação em 1998, o projeto atraiu milhares de participantes e possui uma das maiores comunidades colaboradoras de um projeto de software livre atualmente (MOZILLA, 2006). Apesar do produto principal ser um navegador, o Projeto Mozilla possui vários subprojetos relacionados.

O projeto é desenvolvido e acompanhado por uma organização própria que é responsável pela gerência, planejamento e suporte para o desenvolvimento do mesmo.

Em linhas gerais, essa organização responde pelo processo de desenvolvimento do Mozilla. Os membros que compõem essa organização são voluntários e profissionais de organizações diferentes, sendo que os mais assíduos e melhor preparados possuem privilégios e posições de mais destaque na organização (REIS, 2003).

O fato de ter sido originado de um projeto proprietário trouxe algumas características importantes para a evolução necessária no processo de desenvolvimento. Uma das principais contribuições dessa evolução foi a criação e utilização de um conjunto de ferramentas livres para o apoio do processo adotado, conforme discutido com mais detalhes no Capítulo 5.

### **3.4.3 – O Projeto Apache**

O Projeto Apache, dedicado ao desenvolvimento e manutenção de um servidor http livre, seguro e eficiente para sistemas operacionais modernos, foi iniciado em fevereiro de 1995, resultado de um esforço combinado para coordenar manutenções no programa de httpd NCSA desenvolvido por Rob McCool. Depois de passarem vários meses adicionando funcionalidades e corrigindo pequenos erros, os desenvolvedores do Apache substituíram o código do servidor antigo em julho de 1995 por uma nova arquitetura projetada por Robert Thau. Depois disso, todas as antigas e novas funcionalidades foram traduzidas para a nova arquitetura e foram disponibilizadas para testes, resultando na disponibilização formal do Apache httpd 1.0 em janeiro de 1996.

O processo de desenvolvimento de software do Apache é resultado tanto da natureza do projeto quanto do conhecimento dos líderes de projeto. O Projeto Apache já começou com uma tentativa consciente de resolver os problemas de processo primeiro, antes mesmo de iniciar o desenvolvimento, porque estava claro, desde o início, que conjuntos de desenvolvedores geograficamente dispersos, sem nenhum vínculo organizacional, necessitariam de um único processo de desenvolvimento para poderem tomar decisões (MOCKUS et al., 2002).

Um estudo realizado em (MOCKUS et al., 2002) comprovou algumas das hipóteses relacionadas às características de desenvolvimento no Projeto Apache, por meio de estudos de casos. Dentre elas, destaca-se a hipótese de que o desenvolvimento de software livre envolve um grupo principal de desenvolvedores, que controla a base de código. Esse grupo possui, no máximo, de 10 a 15 pessoas e cria aproximadamente 80% ou mais das novas funcionalidades do software. Outra hipótese relacionada à

anterior é a de que, se o projeto é tão grande que o número de pessoas do grupo principal não é capaz de desenvolver 80% do código num espaço de tempo razoável, então novos grupos devem participar do desenvolvimento. Porém, o trabalho deve ser subdividido em vários projetos. Por fim, é hipótese desse estudo, também, o fato de que em projetos de software livre bem sucedidos, um grupo maior que o grupo principal em uma ordem de magnitude corrige os erros, e um grupo maior ainda (por outra ordem de magnitude) reporta problemas.

### **3.5 Conclusões do Capítulo**

Software Livre possui conceitos bem particulares quando confrontado com produtos de software tradicionais, tanto em termos políticos, econômicos e legais, mas principalmente culturais. Dentre as inovações trazidas à comunidade de software, o Software Livre trouxe novas formas de se licenciar software, dentre elas, a Licença Pública Geral GNU.

Como não poderia ser diferente, dadas as mudanças conceituais, Software Livre traz novos desafios à comunidade no que se refere à forma de desenvolver software, trazendo características e necessidades, antes não evidenciadas.

Neste capítulo foram apresentados os conceitos envolvidos no desenvolvimento de SL e características dos principais projetos de SL, que foram fundamentais para a realização deste trabalho.

Assim, as informações relatadas neste capítulo, reunidas aos conceitos de qualidade discutidos no capítulo anterior foram a base para a definição de processos padrão de desenvolvimento e manutenção de software livre apresentados no próximo capítulo.

## Capítulo 4

### Processos Padrão para Software Livre

O Laboratório de Engenharia de Software (LabES) da Universidade Federal do Espírito Santo (UFES) vem conduzindo desde 1999 o Projeto ODE (FALBO et al., 2003), inicialmente denominado Projeto ADS (MIAN et al., 2001), que visa ao desenvolvimento de um Ambiente de Desenvolvimento de Software (ADS) Centrado em Processo.

Desde então, o ambiente vem amadurecendo e no final de 2004 foi implantado experimentalmente em uma organização de software parceira do LabES para avaliação de sua adequação a organizações de software reais. Dado seu estágio atual em termos de funcionalidades, aliado à sua origem acadêmica, um caminho de evolução natural para o Projeto ODE é a sua liberação como software livre (SL), permitindo, assim, que a comunidade de software possa se beneficiar com o uso de um ADS livre, dado que existem poucos ambientes dessa natureza disponíveis e que a grande maioria dos ADSs disponíveis no mercado têm suas licenças disponibilizadas por preços muito altos, o que inviabiliza sua compra pela maioria das organizações de software de pequeno e médio porte. Surge, assim, o Projeto ODE Livre.

Pelas razões expostas acima, a disponibilização de ODE como software livre tende a chamar a atenção da comunidade de Engenharia de Software, tanto para sua utilização como para a colaboração. Mesmo que haja algum custo para sua obtenção, tal custo tende a ser muito baixo, dada a possibilidade de fácil distribuição a custo zero proporcionada pela política do software livre.

Uma questão importante a ser observada nessa nova etapa do Projeto ODE está relacionada a como garantir que a qualidade do ambiente não será afetada por essa nova modalidade de desenvolvimento. Visando tratar essa questão, decidiu-se definir processos de software padrão a serem adotados no Projeto ODE Livre. A idéia é que cada contribuição feita ao ambiente seja tratada como um projeto que deve instanciar, a partir dos processos padrão definidos para o Projeto ODE Livre, um processo de software adequado às suas características.



A base teórica para a definição desses processos padrão foi levantada nos capítulos anteriores e o foco deste capítulo é a apresentação dos processos padrão definidos para software livre a serem aplicados no Projeto ODE Livre. Assim, este capítulo está estruturado da seguinte forma: a seção 4.1 aborda brevemente o tema Ambientes de Desenvolvimento de Software; a seção 4.2 apresenta o Projeto ODE em linhas gerais; na seção 4.3 é apresentada a abordagem adotada para a transformação de ODE em um ADS Livre; na seção 4.4 os processos padrão do LabES são brevemente apresentados; na seção 4.5 são descritas características dos processos padrão definidos para apoiar o Projeto ODE Livre; a seção 4.6 trata da definição da licença sob a qual o ambiente ODE será disponibilizado; e, finalmente, na seção 4.7 são apresentadas as considerações finais do capítulo.

## **4.1 Ambientes de Desenvolvimento de Software**

Um dos grandes desafios da Engenharia de Software é a automatização de partes do processo de software. Com este objetivo, várias ferramentas para apoiar o engenheiro de software na construção dos seus produtos já foram e continuam sendo desenvolvidas. Contudo, é importante que essas ferramentas sejam capazes de se comunicar para que possam fornecer apoio integrado às várias atividades do processo de software.

Essa necessidade de integração fez surgir os Ambientes de Desenvolvimento de Software (ADSs). Um ADS pode ser visto como uma coleção de ferramentas integradas que facilita as atividades da engenharia de software, durante todo o ciclo de vida do software ou pelo menos em porções significativas dele. ADSs possuem o objetivo de interferir positivamente no tempo de desenvolvimento, no custo e na qualidade de um projeto (HARRISON et al., 2000).

Os ADSs combinam diferentes ferramentas, com diferentes informações e possibilitam a comunicação entre elas, objetivando a troca de informações e evitando inconsistências. Entre os benefícios de um ADS, podem ser citados (PRESSMAN, 2002):

- A transferência harmoniosa de informações entre ferramentas e entre etapas de engenharia de software;
- Redução do esforço exigido para realizar certas atividades, tal como o gerenciamento de configuração de software;

- Apoio à produção de documentação;
- Aumento no controle do projeto, uma vez que é possível haver melhor planejamento, monitoração e comunicação;
- Melhor coordenação entre os membros de uma equipe que estejam trabalhando em um grande projeto de software.

Existem vários níveis de integração de ferramentas. Dentre eles, podem ser citados (TRAVASSOS, 1994) (FALBO, 1998) (PFLEEGER, 2001):

- *Integração de Dados*: A chave para integrar ferramentas é a habilidade de compartilhar informações de projeto.
- *Integração de Apresentação*: A integração de apresentação tem como objetivo tornar as interfaces do ADS consistentes, permitindo que o usuário utilize as ferramentas, alternando de uma para outra, sem sofrer um impacto na interface. Os componentes utilizados devem ser os mesmos, com as mesmas funcionalidades, não havendo a necessidade de aprender como utilizar cada ferramenta. A partir do momento que se aprende uma, consegue-se trabalhar mais facilmente com as outras.
- *Integração de Processo*: Para integrar um conjunto de ferramentas, é preciso ter um foco forte no gerenciamento do processo. Engenheiros de software têm se tornado mais orientados a processos, usando métodos, técnicas e ferramentas para controlar e guiar seu trabalho.
- *Integração de Controle*: As ferramentas devem ser capazes de notificar umas às outras sobre eventos, ativar outras ferramentas e compartilhar funções. Mecanismos de integração de controle incluem a passagem explícita de mensagens, gatilhos ativados por tempo ou acesso, e servidores de mensagens.
- *Integração de Conhecimento*: Com o aumento da complexidade dos processos de software, faz-se necessário considerar informações de natureza semântica, sem as quais a integração não é plenamente obtida. O conhecimento, assim como os dados, deve estar disponível e representado de forma única no ambiente, de forma a poder ser acessado por todas as ferramentas que dele necessitarem, evitando redundância e inconsistências.
- *Integração de Plataforma*: Diz respeito à capacidade do ADS funcionar em

diferentes plataformas de software e de hardware. A prática de desenvolvimento de software é independente de plataforma e, portanto, é interessante que as ferramentas utilizadas no desenvolvimento também sejam.

As dimensões de integração anteriormente citadas são importantes para o desenvolvimento de quaisquer ADSs, seguindo qualquer modelo de desenvolvimento. Porém, merecem atenção especial quando se fala em desenvolvimento de um ADS livre. Isso se deve ao fato dos colaboradores de projetos de software livre estarem dispersos geograficamente, podendo estar localizados em organizações com metodologias e culturas diferentes. Dadas tais dificuldades, a comunicação e o processo devem ser capazes de favorecer que padrões de integração sejam respeitados por quaisquer colaboradores ao redor do mundo, uma vez que uma violação a esses padrões pode acarretar a não aceitação de uma colaboração para um projeto.

Como exemplo, pode-se citar a importância da integração de apresentação para um ADS. Caso um colaborador não siga os padrões de interface determinados para o mesmo, sua colaboração tende a ser descartada pelo grupo coordenador do projeto de tal ADS.

## **4.2 O Projeto ODE**

Conforme discutido anteriormente, em um ADS as ferramentas precisam estar bem integradas para que haja um compartilhamento de informações eficiente. Para tal, é importante que as ferramentas tenham um entendimento comum a respeito dos principais conceitos envolvidos em Engenharia de Software. A conceituação comum necessária para que esse compartilhamento possa ser alcançado em um ambiente pode ser obtida através do uso de ontologias. ODE (*Ontology-based software Development Environment*) (FALBO et al., 2004) é um ADS Centrado em Processo que tem sua fundamentação em ontologias.

Um ADS Centrado em Processo é um ambiente que suporta a definição de processos de software, utilizando-se desta para estabelecer uma ligação explícita entre ferramentas do ambiente e os processos definidos (integração de processo) (BERTOLLO et al., 2002).

A interação das ferramentas em ODE é facilitada pelo fato destas serem construídas tomando por base as mesmas ontologias. Isso significa que os principais conceitos manipulados pelas ferramentas estão bem definidos, dado que a mesma ontologia pode ser utilizada para construir diferentes ferramentas dando apoio a atividades relacionadas da Engenharia de Software.

O Projeto ODE está em desenvolvimento no Laboratório de Engenharia de Software da Universidade Federal do Espírito Santo (LabES/UFES) desde 1999, motivado pela existência de poucos ambientes desse tipo disponíveis. ODE vem sendo desenvolvido no meio acadêmico, sendo o resultado do esforço conjunto de vários alunos de graduação e mestrado. Ele é implementado em Java, com persistência em banco de dados relacional PostgreSQL.

Com o amadurecimento do trabalho, em 2004, foi possível experimentar sua aplicação prática. Para tal, foi feita uma parceria com uma organização de software visando testá-lo num ambiente corporativo. Dessa interação universidade-empresa, diversas sugestões foram dadas e uma nova versão de ODE foi desenvolvida e implantada em março de 2006 nessa organização. Nessa última versão, ODE possui ferramentas para apoiar a definição do escopo de um projeto de software (ARANTES, 2006), a definição de processos (incluindo processos padrão, especializados e de projeto) (BERTOLLO, 2006), a alocação de recursos (SCHWAMBACH, 2002), a realização de estimativas (ARANTES, 2006), a gerência de riscos (FALBO et al., 2004), a gerência de requisitos (FREITAS et al., 2006) e o acompanhamento de projetos (DAL MORO et al., 2005). Outras ferramentas, já desenvolvidas ou em desenvolvimento, estão sendo integradas a ODE e há previsão para a distribuição de uma nova versão do ambiente em outubro de 2006, incluindo melhorias nas ferramentas citadas, uma ferramenta de apoio à modelagem UML e um sistema de gerência de conhecimento.

Dado o objetivo de transformar ODE em um software livre e a importância da comunicação entre colaboradores do Projeto pela Internet, seria ideal que algumas ferramentas de ODE pudessem ser disponibilizadas na Internet para que apoiassem o desenvolvimento de ODE. Ou seja, usando os processos padrão de desenvolvimento e manutenção de ODE Livre, as próprias ferramentas de ODE poderiam ser usadas para apoiar o processo definido para o projeto do colaborador.

### 4.3 Em Direção a ODE Livre

O LabES é um laboratório de pesquisa e ensino em Engenharia de Software, no qual há vários projetos de software em andamento, sobretudo no contexto de projetos de pesquisa, ainda que alguns projetos sejam contratados por organizações externas.

Visando colocar em prática o que é ensinado, o LabES adota uma política de desenvolvimento de software com qualidade, seguindo o preceito de que a qualidade do produto pode ser mais facilmente obtida se o mesmo for desenvolvido segundo um processo de qualidade. Assim, tomando por base modelos e normas de qualidade, principalmente a norma ISO/IEC 12207 (ISO, 1998) e o modelo de referência do MPS.BR (SOFTEX, 2006b), foi definido um processo padrão para o LabES. Além desse processo, seguindo uma abordagem de definição de processos em níveis (ROCHA et al., 2001), foram definidos processos especializados para os paradigmas orientado a objetos e estruturado, seguindo a abordagem discutida no Capítulo 2 e mostrada na Figura 2.1. Essa abordagem se caracteriza pela existência de um processo padrão da organização, processos padrão especializados a partir do processo padrão da organização e processos instanciados ou de projeto. No caso do LabES, o processo raiz é o Processo Padrão LabES (PPLabES). A partir dele, são especializados os Processos Padrão LabES Orientado a Objetos (PPELabES-OO) e Estruturado (PPELabES-Est), a partir dos quais são instanciados os processos de projeto.

O principal projeto desenvolvido pelo LabES é o Projeto ODE, caracterizado na seção anterior. Pela variada gama de ferramentas existentes em ODE, pode-se notar que o projeto de um ADS não é uma tarefa simples. Há muitas ferramentas potencialmente úteis a serem desenvolvidas e, além disso, organizações diferentes podem necessitar customizar algumas dessas ferramentas para melhor adequá-las à sua forma de trabalho. Assim, o modelo de software livre passou a ser considerado potencialmente interessante para a evolução do projeto, dando origem ao Projeto ODE Livre. Pensando ODE como software livre, a comunidade poderia se privilegiar de um ADS poderoso e, em contrapartida, o LabES contaria com a ajuda da comunidade para torná-lo um ambiente mais completo e que atendesse melhor às necessidades de várias organizações.

Conforme discutido no capítulo anterior, muitas das características requeridas para o sucesso de um projeto de software livre se aplicam ao Projeto ODE, tais como: (i) usuários do produto podem contribuir efetivamente para determinar sua funcionalidade,

podendo vir a se interessar em se tornar também desenvolvedores; (ii) a liderança do projeto pode ser exercida pelo LabES, de forma compartilhada por seus membros, ficando um membro ou um grupo de pessoas responsável por analisar as contribuições enviadas para uma ferramenta ou funcionalidade do sistema; (iii) o ambiente ODE já tem funcionalidade interessante o suficiente para atrair a atenção da comunidade de desenvolvimento de software, como pode ser comprovado pela sua implantação em uma organização de desenvolvimento de software; (iv) o LabES continuará fornecendo desenvolvedores para o projeto, sendo importante força de trabalho para o projeto, mas novas instituições, sejam de ensino, sejam de mercado, poderão se engajar ao projeto. Por exemplo, muitos ex-integrantes do LabES podem levar o ambiente para suas novas organizações, tornando-se colaboradores (SILVA et al., 2006).

Contudo, algumas características do Projeto ODE Livre são um pouco diferentes da maioria dos projetos de software livre (SILVA et al., 2006). Primeiro, como há poucos ambientes deste tipo disponíveis, geralmente de difícil acesso, bem como não há padrões documentados para a maior parte das ferramentas de um ADS, atenção especial tem de ser dada à atividade de levantamento de requisitos, inclusive no que tange à elaboração de um documento formal de especificação de requisitos.

Segundo, usabilidade é uma característica de qualidade essencial para um ADS. Assim, esse aspecto tem que ser cuidadosamente tratado. De fato, por um ADS estar muito em linha com o desenvolvimento de software de qualidade, é natural que o Projeto ODE tenha um cuidado especial em relação à qualidade e, portanto, padrões devem ser estabelecidos e deverá ser verificada a conformidade das contribuições a esses padrões.

Por fim, dado que um ADS é um sistema muito complexo, em que a integração de ferramentas é uma questão fundamental, a documentação de sua arquitetura e do projeto de suas ferramentas é vital para a evolução do ambiente.

Tendo em vista as características que aproximam e afastam o Projeto ODE da prática corrente do movimento de software livre, procurou-se chegar a uma abordagem própria para o desenvolvimento do Projeto ODE Livre. Essa abordagem consiste na definição de processos padrão para software livre a serem seguidos pelos colaboradores do projeto. Esses processos foram definidos tomando por base os processos padrão do LabES, seguindo a abordagem de definição de processos em níveis, como ilustra a Figura 4.1.

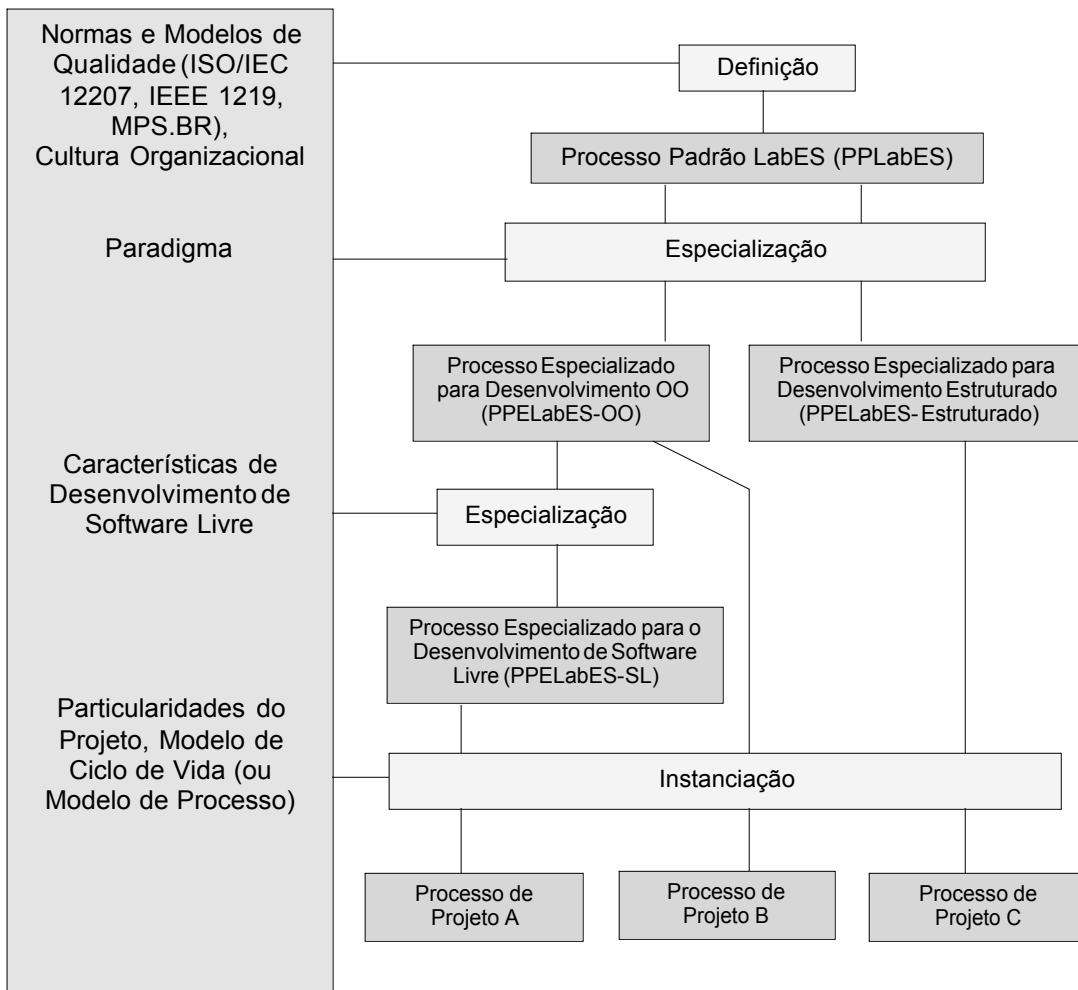


Figura 4.1 – Definição de Processos em Níveis e o Projeto ODE Livre.

Como o Projeto ODE é desenvolvido segundo o paradigma orientado a objetos, os processos padrão para software livre foram definidos como especializações dos processos já especializados para esse paradigma (PPELabES-OO). A ideia é que, para que uma contribuição seja incorporada ao ambiente ODE, ela terá de ser desenvolvida segundo um processo em conformidade com os processos padrão para software livre do LabES. Ou seja, processos têm que ser instanciados a partir dele, para cada sub-projeto de contribuição, levando-se em conta as características desse sub-projeto e da organização colaboradora.

Antes de apresentar os processos padrão para software livre definidos, é importante apresentar a estrutura de processos padrão do LabES, uma vez que muitos elementos são diretamente reutilizados desses processos.

## 4.4 Processos Padrão do LabES

O Processo Padrão do LabES (PPLabES) era originalmente estruturado em duas categorias de processos – Processos Fundamentais e Processos de Apoio – e continha quatro sub-processos (Processo de Desenvolvimento de Software, Processo de Gerência de Projetos, Processo de Garantia da Qualidade e Processo de Gerência de Configuração), como representado na Figura 4.2. Suas atividades eram descritas com base em cinco itens:

1. Entradas: artefatos de insumo para a atividade.
2. Recursos Humanos: papéis das pessoas envolvidas na atividade.
3. Saídas: artefatos produzidos pela atividade.
4. Ferramentas: apoio ferramental à atividade.
5. Procedimentos: métodos, técnicas, modelos de documento, roteiros e outros procedimentos adotados na realização da atividade.

O processo padrão organizacional tinha duas especializações: Processo Padrão LabES Especializado para o Desenvolvimento Orientado a Objetos (PPELabESOO) e Processo Padrão LabES Especializado para o Desenvolvimento Estruturado (PPELabES-Estruturado). Esses processos especializados refinavam as atividades de análise de requisitos e projeto para os respectivos paradigmas.

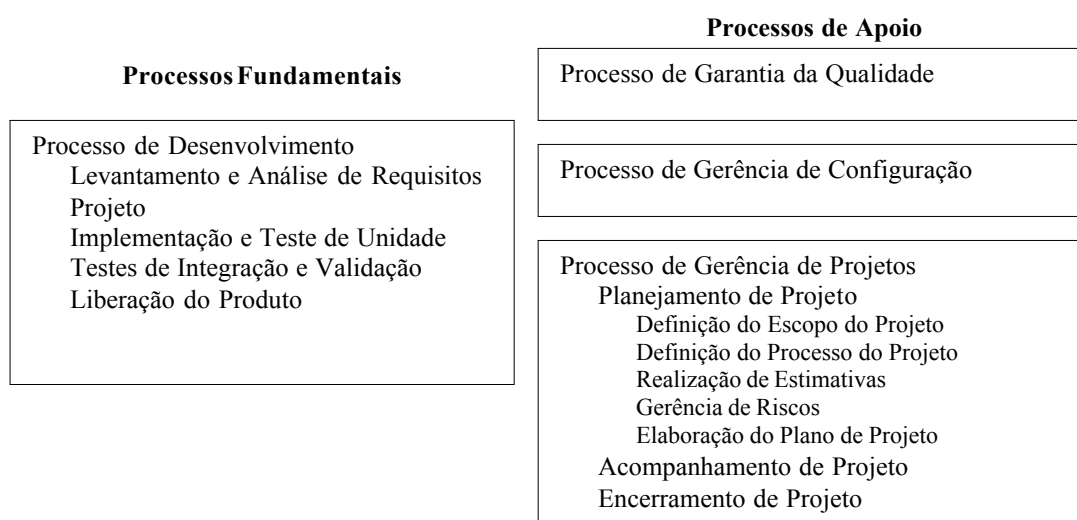


Figura 4.2 – Estrutura Original dos Processos Padrão do LabES.



Durante a elaboração deste trabalho, o PPLabES foi revisado, melhorado e a descrição de suas atividades passou a ser feita nos moldes definidos no MPS.BR. Esta decisão foi tomada depois de um estudo feito sobre o MPS.BR-MR, pois se deseja uma aderência ao mesmo.

Vale destacar que durante a atualização do Processo Padrão LabES, buscando aderência ao MPS.BR, os processos foram agrupados em três categorias: Processos Fundamentais, incluindo os processos de desenvolvimento e manutenção, Processos de Apoio, englobando os processos de garantia da qualidade e gerência de configuração, e Processos Organizacionais, que inclui o processo de gerência de projetos, como mostra a Figura 4.3.

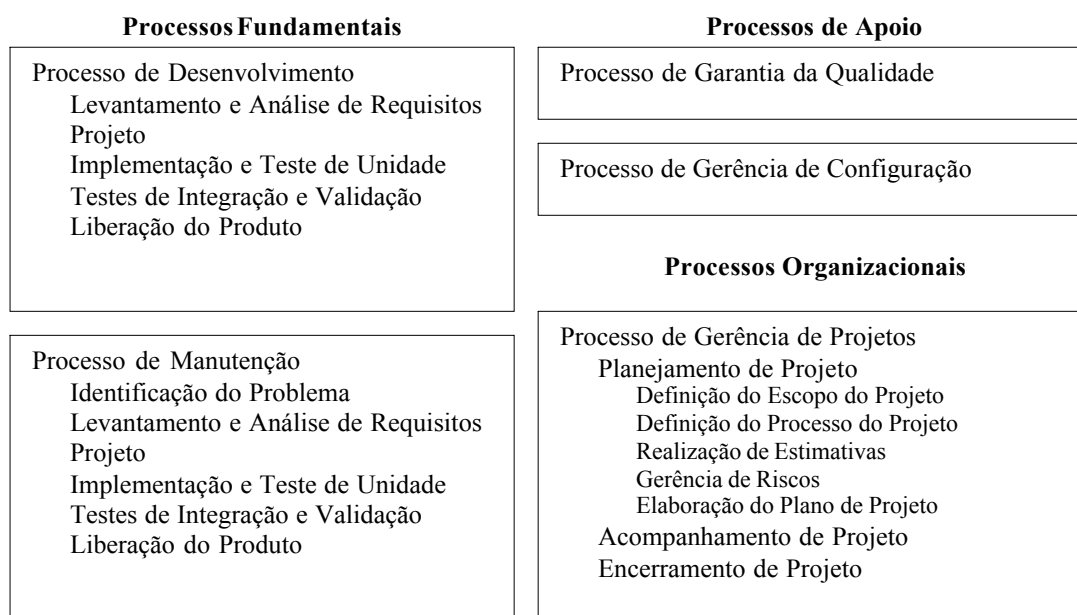


Figura 4.3 – Estrutura Básica dos Processos Padrão do LabES Atualizada.

Esses processos estão definidos em termos de atividades e sub-atividades e, para cada atividade, são definidos, ainda: nome, descrição, critérios de entrada e saída, responsáveis, participantes, artefatos requeridos e produzidos, pré e pós-atividades, ferramentas, procedimentos, resultados esperados (em termos dos resultados esperados definidos no MPS.BR), observações e um indicador da necessidade de execução (obrigatório, fortemente recomendado, recomendado, desejável). A Figura 4.4 mostra o detalhamento para a atividade “Definir Processo de Projeto” do Processo de Gerência de Projetos.

|                              |  |
|------------------------------|--|
| <b>Nome</b>                  | Definir Processo de Projeto  |
| <b>Descrição</b>             | Um dos processos padrão do LabES deve ser instanciado para o projeto em questão e adequado a ele.  |
| <b>Critérios</b>             |  |
| <b>Entrada</b>               | Escopo definido  |
| <b>Saída</b>                 | Processo de Projeto  |
| <b>Responsável(is)</b>       |  |
| <b>Organização</b>           | LabES  |
| <b>Papel(is)</b>             | Gerente de Projeto   |
| <b>Participantes</b>         |  |
| <b>Organização</b>           | LabES  |
| <b>Papel(is)</b>             | Analista de Sistemas   |
| <b>Artefatos Requeridos</b>  | Escopo do Projeto, Características do Projeto e Processo Padrão LabES ou alguma de suas especializações  |
| <b>Artefatos Gerados</b>     | Processo de Projeto  |
| <b>Pré-atividade</b>         | Definir Escopo do Projeto  |
| <b>Pós-atividade</b>         | Realizar Estimativas   |
| <b>Sub-atividades</b>        |  |
| <b>Ferramentas</b>           | Ferramenta de Apoio à Definição de Processos   |
| <b>Procedimentos</b>         |  |
| <b>Indicador de Execução</b> | Obrigatório  |
| <b>Resultados Esperados</b>  | DFP (Definição do Processo Organizacional) 5 - São desenvolvidas estratégias para adaptação do processo-padrão de acordo com as necessidades dos projetos. |
| <b>Observações</b>           |  |

Figura 4.4 – Detalhamento da atividade “Definir Processo de Projeto” do Processo de Gerência de Projetos do Processo Padrão LabES.

Além da revisão dos processos existentes, neste trabalho foi elaborado o Processo Padrão de Manutenção do LabES. Esse processo está descrito também nos moldes do MPS.BR e seus processos e atividades são aderentes à norma IEEE 1219 (IEEE, 1998). Esse processo foi inserido na categoria de Processos Fundamentais, como mostra a Figura 4.3.

Na seção que se segue, são discutidas as principais características do processo padrão especializado para software livre definido e os fatores que motivaram a inclusão dessas características no processo.

## 4.5 Processos Padrão para Projetos de Software Livre

O estudo para a definição do Processo Padrão Especializado LabES para Software Livre (PPELabES-SL) começou por um exame de características relevantes de produtos livres e por uma avaliação de quais processos do MPS.BR se aplicariam a um processo desse tipo. A seguir, procuraram-se encontrar no Processo Padrão do LabES (PPLabES) os elementos de processo correspondentes, visando a compatibilidade. Tendo em vista que é um objetivo do LabES manter seus processos em linha com as diretrizes do MPS.BR, a definição do PPELabES-SL serviu de base, também, para a melhoria do PPLabES, que passou a incorporar algumas das recomendações do MPS.BR ainda não tratadas.

Conforme discutido no Capítulo 2, o modelo de referência do MPS.BR (SOFTEX, 2006b) define 21 processos, agrupados em três categorias:

- Processos Fundamentais: Aquisição, Gerência de Requisitos, Desenvolvimento de Requisitos, Solução Técnica e Integração do Produto.
- Processos de Apoio: Garantia da Qualidade, Gerência de Configuração, Validação, Verificação, Medição e Análise de Decisão e Resolução.
- Processos Organizacionais: Gerência de Projeto, Definição do Processo Organizacional, Adaptação do Processo para Gerência de Projeto, Avaliação e Melhoria do Processo Organizacional, Gerência de Riscos, Treinamento, Gerência Quantitativa do Projeto, Desempenho do Processo Organizacional, Análise de Causas e Resolução e Implantação de Inovações na Organização.

Esses processos são organizados em sete níveis crescentes de maturidade (G a A), sendo que, inicialmente, foram considerados apenas os processos dos quatro primeiros níveis (G a D), uma vez que esses níveis já proporcionam um elevado nível de qualidade. Assim, foi avaliada a aplicabilidade a software livre dos seguintes processos: Gerência de Projeto e Gerência de Requisitos (nível G), Garantia da Qualidade, Aquisição, Gerência de Configuração e Medição (nível F), Adaptação do Processo para Gerência de Projeto, Definição do Processo Organizacional, Avaliação e Melhoria do Processo Organizacional e Treinamento (nível E) e Desenvolvimento de Requisitos, Solução Técnica, Integração do Produto, Verificação e Validação (nível D).

Todos esses processos foram considerados aplicáveis e suas atividades foram classificadas, segundo a importância de seus resultados esperados, em: obrigatória, fortemente recomendável, recomendável e desejável. Atividades obrigatórias devem ser executadas e os artefatos por elas produzidos têm de ser entregues como parte da contribuição para que a mesma seja aceita. Para facilitar sua realização, ferramentas de apoio deverão ser providas. Atividades fortemente recomendáveis, ainda que não obrigatórias, também são muito importantes e será incentivada a sua realização, provendo-se ferramentas de apoio. Para as demais atividades, inicialmente, não será provida nenhuma facilidade nem será exigida nenhuma evidência de sua realização, ainda que as mesmas estejam definidas no processo.

Algumas das atividades do processo ficarão sob responsabilidade do colaborador, ou seja, do indivíduo ou organização que esteja trabalhando uma contribuição, e outras ficarão sob responsabilidade da organização que detém o controle do projeto, no caso, o LabES. Assim, informações sobre a responsabilidade de execução do processo / atividade também foram definidas. Visando tornar o processo mais enxuto para os colaboradores, processos que são integralmente de responsabilidade do LabES não foram incorporados ao PPELabES-SL. Este é o caso dos processos de Definição do Processo Organizacional e Avaliação e Melhoria do Processo Organizacional.

No que tange ao processo de aquisição, vale destacar que, nos sub-projetos do Projeto ODE Livre, um novo projeto de desenvolvimento ou manutenção se iniciará a partir da demanda de um usuário de ODE que registrará sua crítica ou sugestão (*bug* no sistema, oportunidade de melhoria, sugestão de nova funcionalidade etc) no Portal do Projeto. Algumas vezes, esse usuário também vai ser o colaborador responsável pelo desenvolvimento de sua própria requisição, mas acredita-se que, na maioria das vezes, outro colaborador desenvolverá um novo projeto baseado nos requisitos deixados pelo usuário idealizador. Esses fatores comprovam a falta de uma identificação clara da relação cliente/fornecedor para projetos de software livre, levando à simplificação deste processo. Apesar disso, critérios para aceitação e seleção de contribuições são fortemente recomendáveis.

Pelas características de software livre, o Processo de Treinamento, ainda que considerado desejável, não foi definido, uma vez que cabe a cada organização definir como tratar essa questão. Tutoriais sobre Engenharia de Software e sobre como realizar atividades do processo podem ser disponibilizados no Portal do Projeto. Essa porção do Processo de Treinamento, contudo, ficaria a cargo do LabES e, por isso, não foi incluída

no processo. No caso do treinamento de usuários, material de apoio também deve ser produzido e disponibilizado no Portal do Projeto. Estuda-se a criação de um *wiki* (WIKI, 2006) voltado ao Projeto ODE e conhecimentos relacionados a Engenharia de Software importantes para o uso de ODE.

Os Processos de Gerência de Projeto e Adaptação do Processo para Gerência de Projeto foram tratados como um processo único, denominado Processo de Gerência de Projetos, composto das seguintes atividades, tomando por base o processo padrão LabES: Planejamento de Projeto, Acompanhamento de Projeto e Encerramento de Projeto. A atividade de Planejamento de Projeto é decomposta nas seguintes sub-atividades: Definição do Escopo do Projeto (obrigatória), Definição do Processo de Projeto (obrigatória), Realização de Estimativas (fortemente recomendável), incluindo estimativas de tamanho e esforço, alocação de recursos e elaboração de cronograma, Gerência de Riscos (desejável) e Elaboração do Plano de Projeto (obrigatória). No acompanhamento, as atividades do planejamento são realizadas novamente, sendo que um relatório de acompanhamento deve ser elaborado. O encerramento inclui análises *post-mortem*. Todas essas atividades são realizadas pelo colaborador, com exceção do encerramento que pode envolver também membros do LabES.

Ainda no que se refere à Gerência de Projetos, julgou-se que, geralmente, não se aplica a software livre atividades relacionadas a planejamento e acompanhamento de custos, tendo em vista que, por princípio, o desenvolvimento de software livre tende a não prever custo financeiro. Vale destacar, ainda, que a realização de estimativas é considerada apenas fortemente recomendável, e não obrigatória, pela inexistência de prazos fixos para um projeto de software livre. Cabe ao colaborador gerenciar seu tempo, ainda que estimativas de tamanho, esforço e tempo, juntamente com o cronograma do projeto, facilitem a organização e a viabilidade do projeto.

No Processo de Medição, toda a parte de definição de métricas deve ficar a cargo do LabES, bem como a definição das atividades de medição e a análise dos resultados. Caberia às organizações colaboradoras apenas a coleta dos dados, bem como o uso dessas informações para apoiar suas decisões. Para tal, é muito importante prover ferramentas de apoio para que esse processo se torne viável, incluindo um sistema de gerência de conhecimento. Assim, optou-se por postergar a introdução desse processo para uma futura melhoria do PPELabES-SL. De fato, esse processo não foi incorporado ao PPLabES.

Os demais processos foram todos considerados muito importantes e incorporados ao PPELabES-SL.

Visando à simplicidade e para manter a compatibilidade com o PPLabES, os processos de Garantia da Qualidade, Verificação e Validação foram agrupados em um único processo, denominado Processo de Garantia da Qualidade. Assim, o processo de garantia da qualidade proposto envolve atividades de verificação e validação, incluindo a garantia da conformidade a padrões e revisão conjunta. Vale destacar que no âmbito da definição deste processo, diversos modelos de documento foram propostos, tais como Modelo de Plano de Projeto, Modelo de Especificação de Requisitos e Padrão de Codificação, e, portanto, verificar a aderência a esses padrões é fundamental. Assim, as atividades desses processos deverão ser realizadas tanto pelos colaboradores quanto pelos membros do LabES, durante a avaliação de uma contribuição.

O Processo de Gerência de Configuração é de extrema importância para os projetos de software livre. Os itens de configuração devem ser muito bem controlados, pois disso depende o sucesso do projeto de software livre. Os colaboradores devem ter acesso aos itens para poderem colaborar efetivamente e as versões de cada item devem ser controladas e documentadas de modo a evitar retrabalho e para permitir um bom entendimento por parte dos colaboradores.

Os demais processos – Gerência de Requisitos, Desenvolvimento de Requisitos, Solução Técnica e Integração do Produto – foram todos incorporados a um processo de engenharia. Na verdade, seguindo o PPLabES, foram definidos dois processos de engenharia: um Processo de Desenvolvimento e outro de Manutenção. Esses processos têm um esqueleto muito similar, embora o primeiro esteja focado no desenvolvimento de novos artefatos, enquanto o foco do segundo é a manutenção de artefatos previamente desenvolvidos (ainda que novos artefatos possam ser criados). Esse esqueleto é composto das seguintes atividades: Levantamento e Análise de Requisitos, Projeto, Implementação e Teste de Unidade, Testes de Integração e Validação, e Liberação do Produto. Todas essas atividades, com exceção da Liberação do Produto, são de responsabilidade do colaborador. Contudo, atividades de Integração e Testes de Integração e Validação são realizadas também por membros do LabES. Por fim, cabe aos membros do LabES decidir quando liberar uma nova versão do ambiente, devendo descrever, ainda, as modificações efetuadas e os responsáveis por elas.

É importante destacar, ainda, algumas características do Processo de Manutenção. Ele tem como objetivo principal a solução de Solicitações de Alteração

feitas por usuários. Esse processo, assim como o Processo Padrão de Manutenção do LabES, é fortemente baseado na norma IEEE 1219 (IEEE, 1998). Assim, além das atividades comuns ao processo de desenvolvimento, há uma atividade inicial de Identificação do Problema, que envolve as seguintes sub-atividades:

- Submeter Solicitação de Alteração: O processo é iniciado quando um usuário de ODE submete uma solicitação de alteração pelo Portal do Projeto. A solicitação é numerada automaticamente e armazenada;
- Analisar Validade e Classificar Solicitação de Alteração: Membros de LabES são designados para analisar a validade da Solicitação de Alteração registrada e, caso seja verificada sua real necessidade de execução, a mesma será classificada segundo tipo (corretiva, adaptativa, perfectiva ou preventiva) e área do projeto;
- Efetuar Estimativa Preliminar de Tamanho / Magnitude da Manutenção: Depois de classificada, o tamanho e a magnitude da Solicitação devem ser estimados. Essas estimativas são feitas por membros do LabES e são importantes por auxiliarem a decisão de colaboradores, indicando recursos necessários para a sua realização;
- Priorizar Solicitação de Alteração: As solicitações devem ser agrupadas por classe e áreas do projeto e submetidas a uma análise, efetuada em reunião entre membros do LabES, na qual são priorizadas e posteriormente publicadas como um novo sub-projeto no Portal do Projeto, estando disponível para os colaboradores. As reuniões são periódicas.

A partir deste ponto o colaborador é o agente ativo na solução de Pacotes de Solicitações de Alteração. Através do Portal do Projeto, o colaborador se propõe a solucionar um pacote de solicitações e obtém os artefatos necessários para tal. Uma vez selecionado um sub-projeto de manutenção por um colaborador, um processo semelhante ao processo de desenvolvimento é realizado, incluindo as interações com os processos de apoio.

É importante frisar que a estrutura do Processo Padrão LabES para Software Livre é a mesma que a do Processo Padrão LabES, apresentada na Figura 4.3, assim como o detalhamento das atividades. A Figura 4.5 mostra o detalhamento da atividade

“Definir Processo de Projeto” do Processo de Gerência de Projetos, agora no contexto do Processo Padrão LabES Especializado para Software Livre.

|                              |  |
|------------------------------|--|
| <b>Nome</b>                  | Definir Processo de Projeto  |
| <b>Descrição</b>             | O processo padrão LabES para Software Livre deve ser instanciado para o projeto em questão e adequado a ele.   |
| <b>Crítérios</b>             |  |
| <b>Entrada</b>               | Escopo definido  |
| <b>Saída</b>                 | Processo de Projeto  |
| <b>Responsável(is)</b>       |  |
| <b>Organização</b>           | Colaborador  |
| <b>Papel(is)</b>             | Gerente de Projeto   |
| <b>Participantes</b>         |  |
| <b>Organização</b>           | Colaborador  |
| <b>Papel(is)</b>             | Analista de Sistemas   |
| <b>Artefatos Requeridos</b>  | Escopo do Projeto, Características do Projeto e Processo Padrão LabES Livre  |
| <b>Artefatos Gerados</b>     | Processo de Projeto  |
| <b>Pré-atividade</b>         | Definir Escopo do Projeto  |
| <b>Pós-atividade</b>         | Realizar Estimativas   |
| <b>Sub-atividades</b>        |  |
| <b>Ferramentas</b>           | Ferramenta de Apoio à Definição de Processos, Ferramenta de <i>Groupware</i> (Portal).   |
| <b>Procedimentos</b>         |  |
| <b>Indicador de Execução</b> | Obrigatório  |
| <b>Resultados Esperados</b>  | DFP (Definição do Processo Organizacional) 5 - São desenvolvidas estratégias para adaptação do processo-padrão de acordo com as necessidades dos projetos. |
| <b>Observações</b>           |  |

Figura 4.5 – Detalhamento da atividade “Definir Processo de Projeto” do Processo de Gerência de Projetos do Processo Padrão LabES para Software Livre.

Os processos padrão do LabES, incluindo suas especializações para software livre, estão publicados na íntegra no Portal do LabES e podem ser acessados a partir do site do [www.inf.ufes.br/~labes](http://www.inf.ufes.br/~labes).

#### 4.6 Licença para Disponibilização de ODE como Software Livre

Um requisito fundamental para a transformação de ODE em um SL é que se decida sob qual licença o software será disponibilizado. Essa escolha é fundamental,



pois espelha os objetivos da organização (no caso o LabES) ao disponibilizar seu produto como SL. Portanto, se mal escolhida, pode representar o fracasso de todo um planejamento de um projeto.

Neste trabalho foram realizadas pesquisas sobre as várias licenças de SL existentes, buscando confrontar cada licença estudada às características do Projeto ODE. Havia a possibilidade de se criar uma licença própria para ODE Livre, porém essa opção foi desconsiderada, tanto por não haver, por parte dos participantes do projeto, uma pessoa que domine a legislação correspondente, quanto por ser uma escolha perigosa no sentido de estar sujeita a incompatibilidades entre as licenças já existentes.

Das licenças pesquisadas, as que mais se destacavam eram a FreeBSD (FREEBSD, 2006) e a GNU-GPL (GNU-GPL, 2006), sendo que foram encontrados, inclusive, documentos que indicavam a adoção de licenças do estilo BSD para pesquisadores. Porém, dada a origem acadêmica do Projeto ODE (sem fins lucrativos) e pelo fato da licença GNU-GPL estar mais de acordo com os conceitos reais definidos para Software Livre, decidiu-se adotar essa licença para o Projeto ODE Livre. Vale destacar que essa decisão foi auxiliada por uma consulta feita à FSF (*Free Software Foundation*) (FREE SOFTWARE FOUNDATION, 2006), por meio de mensagem eletrônica .

#### **4.7 Considerações Finais do Capítulo**

Com o objetivo de produzir um Ambiente de Desenvolvimento de Software livre, o Projeto ODE Livre foi criado. Neste capítulo foram apresentadas as motivações para a transformação do Projeto ODE em um projeto de software livre e as novas características que essa nova modalidade de desenvolvimento traria para o processo de software.

Para dar o apoio necessário a essa transformação, foi definido um novo nível na estrutura de processos de software do LabES, onde foram inseridos os processos padrão especializados para software livre.

Contudo, para tornar viável a utilização dos processos definidos, é necessário que se crie um ambiente que os apóie. O próximo capítulo discute características, requisitos e ferramentas que tal ambiente deve possuir.

## Capítulo 5

# Requisitos de um Ambiente de Apoio ao Projeto ODE Livre

Durante a definição do processo que uma organização deve seguir, é desejável que sejam disponibilizadas ferramentas de apoio, visando facilitar a realização das atividades pertencentes ao mesmo.

No caso do desenvolvimento de Software Livre, o apoio ferramental se torna essencial, principalmente, devido à dispersão geográfica dos colaboradores. Esse fato também exige que as ferramentas sejam disponibilizadas pela Internet, ou que os artefatos produzidos o sejam.

Dentro deste contexto, este capítulo se inicia com um levantamento de características e requisitos de um ambiente de apoio ao desenvolvimento de Software Livre. Esse levantamento foi feito por meio de pesquisas a alguns projetos de Software Livre existentes, buscando identificar funcionalidades de apoio necessárias ao Projeto ODE Livre.

Após essa pesquisa, foram definidos requisitos para o Portal ODE Livre, portal que concentrará todo o apoio ferramental necessário para o desenvolvimento de ODE seguindo o processo padrão definido no Capítulo 4.

Este capítulo está estruturado da seguinte forma: na seção 5.1 são levantadas características ambientes de apoio ao desenvolvimento de Software Livre (SL) levando em conta projetos de SL bem sucedidos; a seção 5.2 descreve os requisitos de um ambiente de apoio ao Projeto ODE Livre com base nas características levantadas e a estrutura geral e as funcionalidades que se pretende disponibilizar no Portal ODE Livre; finalmente, na seção 5.3 são apresentadas as considerações finais do capítulo.

## **5.1 Características de Ambientes de Apoio ao Desenvolvimento de Software Livre**

Tendo como base as características de desenvolvimento de Software Livre levantadas no Capítulo 3 desta dissertação, chega-se à conclusão de que a característica fundamental e básica de um ambiente de apoio ao desenvolvimento de SL é que a grande maioria de suas funcionalidades esteja disponível pela Internet. Essa característica é facilmente observada pelo fato de que os colaboradores de um projeto de SL estão, na grande maioria das vezes, dispersos geograficamente, exigindo que o apoio ferramental forneça suporte ao desenvolvimento colaborativo e acesso simultâneo a um grande volume de informações.

Contudo, é importante observar outras características importantes, sobretudo funcionais. Pelo destaque adquirido mundialmente, as características levantadas neste capítulo têm como base, principalmente, três dos mais importantes projetos de SL existentes – Projeto Mozilla, Projeto Apache e Projeto Linux – cujos ambientes de apoio são sucintamente apresentados a seguir.

### **5.1.1 – O Projeto Mozilla**

A maioria das ferramentas de apoio ao Mozilla pode ser acessada por um navegador *Web* e todas elas estão disponíveis sob licenças de Software Livre, de modo que possam ser utilizadas em qualquer projeto, seja livre ou proprietário. Dentre elas destacam-se: CVS, Bugzilla, Bonsai, Tinderbox, LXR e ferramentas de comunicação.

#### **CVS (*Concurrent Version System*)**

É um sistema de controle de versão, um componente importante da Gerência de Configuração de Software. Com sua utilização, é possível registrar o histórico de evolução do código fonte e dos documentos de um projeto (FREE SOFTWARE FOUNDATION, 2006). Segundo REIS (2002), são aspectos interessantes que dessa ferramenta:

- Repositório Central: o CVS opera no modo cliente-servidor. Um repositório central armazena o código fonte e informações da versão, e os clientes requisitam o código e recebem cópias do mesmo.
- Controle de Versões: é a funcionalidade básica de um sistema de gerência de configuração. O CVS permite que versões de arquivos anteriores possam ser recuperadas.
- Ramos (*Branches*): o CVS suporta múltiplos ramos de desenvolvimento, que podem coexistir independentemente. Essa característica é importante na integração de novas funcionalidades com grande impacto.
- Desenvolvimento Concorrente: o CVS não bloqueia o artefato (código ou documento) em um *checkout* (registro de que uma alteração no artefato está sendo feita). Em vez disso, cada desenvolvedor faz um *checkout* do artefato, obtendo uma cópia local, modifica o mesmo, e quando estiver pronto, submete de volta ao repositório. Conflitos de alterações no artefato são resolvidos no lado do cliente, e não no repositório.
- Suporte de Rede: o CVS funciona bem em redes de qualquer tamanho, que é um pré-requisito para o desenvolvimento de SL.
- Funcionalidades Oferecidas: o CVS é implementado como uma ferramenta que apóia as várias tarefas envolvidas no controle de versões, fazendo *checkout* do código, atualizando uma cópia local, imprimindo diferenças entre versões e submetendo mudanças.

## **Bugzilla**

Bugzilla é um sistema de controle de defeitos que permite que indivíduos ou grupos de programadores acompanhem relatórios de erros ou pedidos de melhoria, sendo atualmente a principal ferramenta existente para gestão de erros (BUGZILLA, 2006). Bugzilla é a ferramenta de apoio de maior destaque do Projeto Mozilla, tanto que é utilizada pelos principais projetos de Software Livre existentes.

Bugzilla possui um grande número de funcionalidades de apoio ao processo de desenvolvimento do Projeto Mozilla, oferecendo suporte a diferentes atividades e agindo como um eixo central para comunicação e revisão de código entre membros da comunidade do projeto (REIS, 2002). Os desenvolvedores o utilizam diariamente para

registrar pacotes e pedir e fornecer revisão; os membros do grupo de garantia da qualidade o utilizam para reportar e controlar defeitos (*bugs*); gerentes o utilizam para alocar desenvolvedores e controlar o progresso.

Entre as características do Bugzilla, destacam-se (REIS, 2002):

- Conta de Usuário: cada usuário tem uma conta, identificada por seu endereço eletrônico (e-mail). O processo de criação e validação da conta é muito simples, o que reduz a barreira de adesão e encoraja a participação.
- Atributos de Defeito (*Bug*): os defeitos têm propriedades que combinam muito bem com os requisitos de processo do Mozilla e permitem um bom controle sobre a responsabilidade, o planejamento, dependências e estado. A Figura 5.1 mostra os vários atributos de defeitos.

**Bugzilla Bug 97966** Query JS nuked old selections.

Bug List: (0 of 14) [First](#) [Last](#) [Prev](#) [Next](#) [Show list](#) [Query page](#) [Enter new bug](#)

Bug#: [97966](#)

Product: Bugzilla

Component: Query/Bug List

Status: RESOLVED

Resolution: FIXED

Assigned To: [kiko@async.com.br](#) (Christian Reis)

QA Contact: [matty@chariot.net.au](#)

URL:

Summary: Query JS nuked old selections.

Status:

Whiteboard:

Keywords: patch, review

Platform: All

OS: All

Version: 2.15

Priority: P1

Severity: blocker

Target: Bugzilla 2.16

Milestone:

Reporter: [matty@chariot.net.au](#) (Matthew Tuck)

Add CC:

CC: [caillon@returnzero.com](#), [gerv@mozilla.org](#), [justclave@synclcomm.com](#), [myk@mozilla.org](#)

Remove selected CCs

| Attachment   | Type  | Modified       | Status                        | Actions              |
|--|-------|----------------|-------------------------------|----------------------|
| <a href="#">patch_v1: first hack, can probably be improved on.</a>   | patch | 02/02/02 14:53 | none                          | <a href="#">Edit</a> |
| <a href="#">kiko_v1: fix bug, no JS warnings.</a>                    | patch | 02/02/02 14:53 | none                          | <a href="#">Edit</a> |
| <a href="#">kiko_v2: fix per caillon's comments, no JS warnings.</a> | patch | 02/05/02 04:35 | first-review                  | <a href="#">Edit</a> |
| <a href="#">kiko_v3: fix tiny comment bug</a>                        | patch | 02/05/02 04:35 | first-review<br>second-review | <a href="#">Edit</a> |

Create a New Attachment (proposed patch, testcase, etc.) [View All](#)

Bug 97966 depends on: [122154](#)  [Show dependency tree](#)

Bug 97966 blocks:  [Show dependency graph](#)

Votes: 0 [Show votes for this bug](#) [Vote for this bug](#)

Figura 5.1 - Formulário de Defeitos do Bugzilla.

- Registro de Comentários: pelo fato de cada defeito manter uma lista seqüencial de comentários de usuários, o Bugzilla trabalha muito bem como um fórum de discussão focado.
- Controle de Anexos: defeitos podem ter anexos, que são arquivos carregados pelo usuário e ligados a um defeito específico. A maioria dos anexos são complementos para o defeito, mas também podem ser casos de teste, capturas de telas e especificações. O controle de anexos também fornece apoio para revisão de código e tem uma interface especial para isso.
- Interface de Busca: Bugzilla registra centenas de milhares de defeitos. Para apoiar as consultas a essa base de dados, há uma função de busca que permite especificar que atributos definem o defeito sendo pesquisado.
- Integração de *E-mail*: as mudanças geram mensagens eletrônicas para todas as partes envolvidas em um defeito, permitindo que as pessoas sejam notificadas das requisições e do progresso do mesmo.
- Modelo Conceitual Simples: Bugzilla é basicamente um registrador de defeitos e embora possua uma gama extensa de funcionalidades, os conceitos são simples de entender: produtos, componentes (que são subdivisões de um produto), anexos e defeitos (*bugs*).

## Bonsai

Bonsai é uma interface de busca para o repositório do CVS, que permite a realização de diversas consultas no contexto de um arquivo de CVS, tais como: obter uma lista de *checkins* (registros de realização de alteração), ver que *checkins* foram realizados por uma determinada pessoa, ou num dado ramo (*branch*) de CVS, ou num determinado período de tempo. Também inclui ferramentas para visualizar registros (*logs*) de *checkins* (e comentários), exibir diferenças entre versões de um arquivo e descobrir que pessoa é responsável por alterar determinada linha de código (BONSAI, 2006).

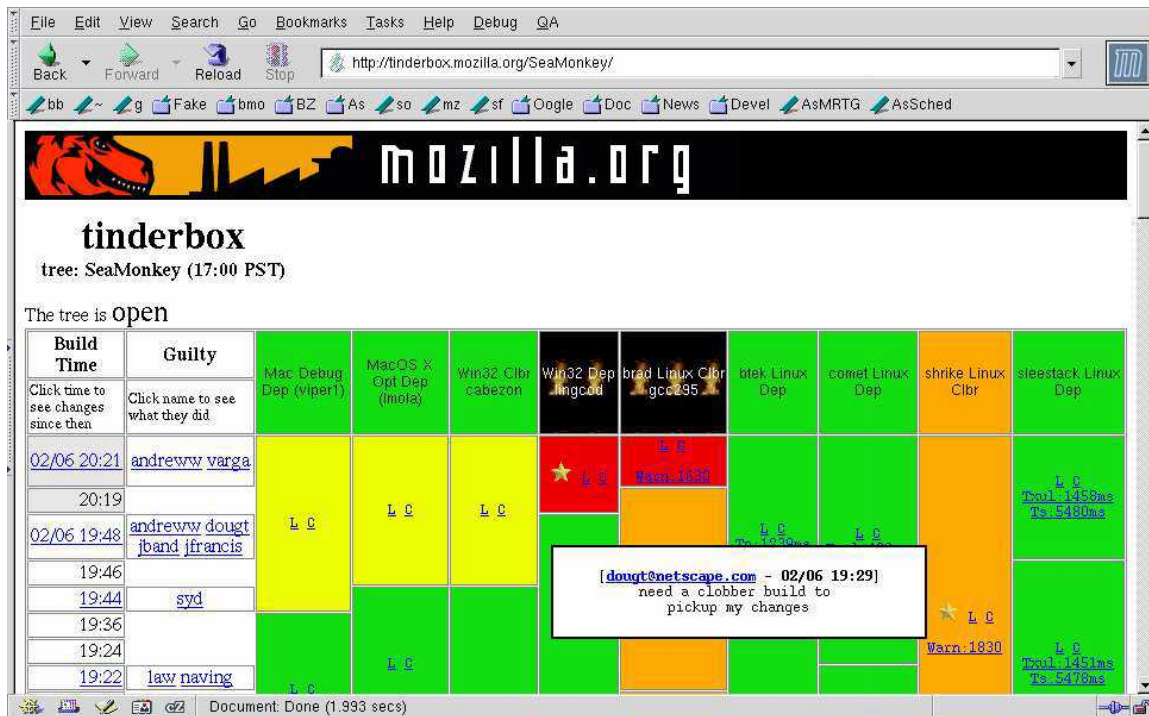
A Figura 5.2 mostra a tela inicial de controle de consulta do Bonsai.



Figura 5.2 – Tela inicial do Controle de Consulta do Bonsai.

## Tinderbox

Tinderbox é um sistema automatizado que controla compilações e testes. É uma ferramenta cliente-servidor: as máquinas do cliente de variadas arquiteturas e sistemas operacionais formam um conjunto. A tarefa dessas máquinas é compilar, testar e reportar de volta os resultados ao servidor Tinderbox. A principal característica visível ao usuário é uma página exibindo os resultados da compilação associada à máquina individual no conjunto e as mudanças de código que foram integradas ao repositório ao mesmo tempo, como mostra a Figura 5.3. Cada máquina do conjunto de compilação é representada por uma coluna, que mostra as várias construções que aconteceram durante um período de tempo. As construções mais recentes aparecem nas colunas de cima. A cor de cada seção das colunas indica um resultado diferente da compilação (REIS, 2002).



5.3 – Tela principal do Tinderbox.

## LXR

É uma ferramenta de hipertexto que indexa código fonte e gera páginas HTML, como mostra a Figura 5.4. Foi originalmente desenvolvida como uma ferramenta para o estudo do *Kernel* do Linux, mas foi adaptada ao Mozilla pela comunidade. Exibe arquivos de código como páginas, com *links* pra cada linha e para cada identificador, isto é, funções, classes e variáveis são ligadas por *hyperlinks*. Assim, é possível acessar a declaração e a implementação de uma função que está sendo chamada em um certo arquivo (LXR, 2006).



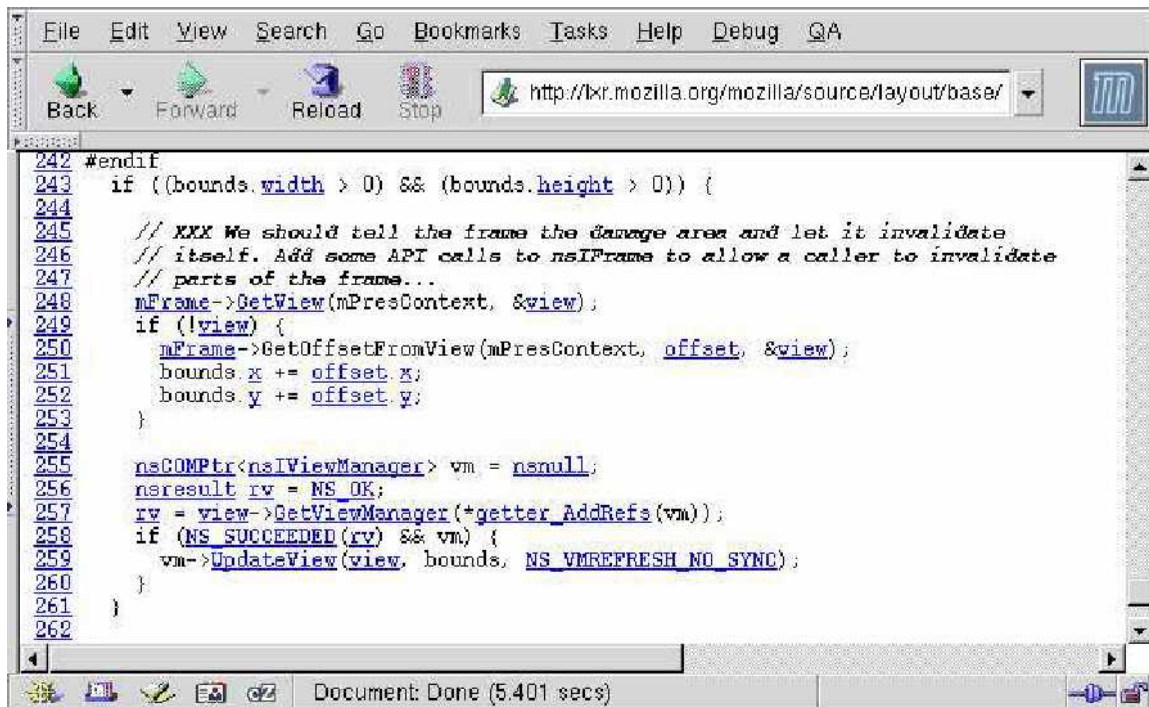


Figura 5.4 – Exemplo de página gerada pelo LXR.

## Wiki-Mozilla

Uma ferramenta vem ganhando cada vez mais adeptos para o auxílio na gerência de conhecimento de projetos de software: o Wiki. O Projeto Mozilla tem o seu Wiki, o Wiki-Mozilla, cuja página inicial é exibida na Figura 5.5 (WIKI-MOZILLA, 2006).

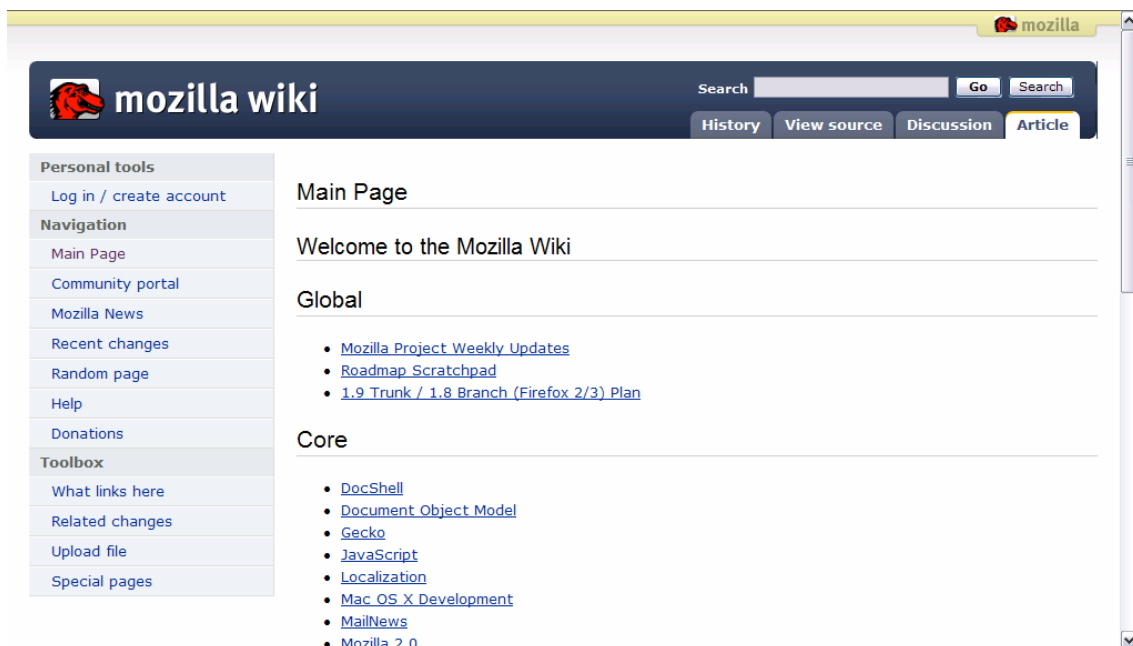


Figura 5.5 – Página inicial do Wiki-Mozilla

Wiki é uma ferramenta que permite que usuários criem conteúdo livremente e editem conteúdo na *Web* usando qualquer tipo de navegador *Web*. Oferece suporte a *hiperlinks* e possui uma sintaxe simples para criar novas páginas e referências a outros documentos internos (WIKI, 2006).

Esta ferramenta possibilita que usuários cadastrados tenham a liberdade de criar e editar páginas *Web* a qualquer momento, encorajando a utilização democrática da *Web* e promovendo a composição do conteúdo por usuários não técnicos.

Wikis podem ser facilmente utilizados no apoio à Gerência de Conhecimento de Projetos de Software Livre por armazenarem e disponibilizarem a experiência dos mais diversos colaboradores e usuários.

### **Ferramentas de Comunicação**

Motivado pela dispersão geográfica dos colaboradores, ferramentas de comunicação via *Web* são fundamentais para a viabilidade de projetos de Software Livre. Apesar da comunidade do Mozilla utilizar, muitas vezes, o Bugzilla como uma ferramenta de apoio à comunicação, listas de discussão via e-mail e IRCs (*Internet Relay Chats*) são mais viáveis para tal propósito.

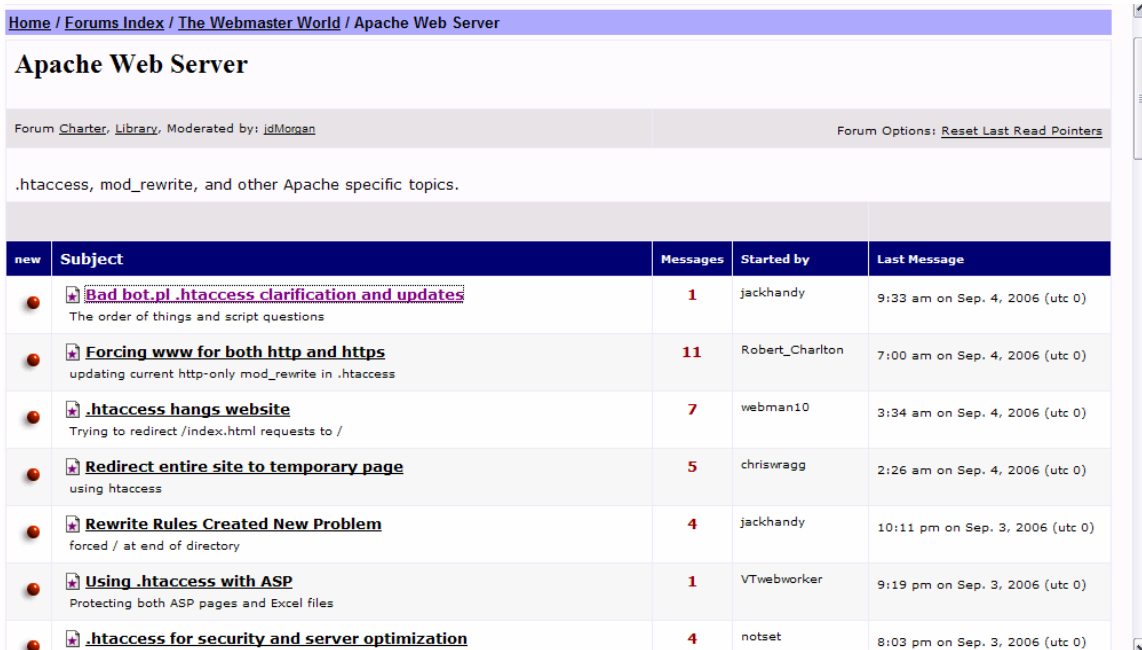
Um IRC é um sistema de comunicação em tempo real que aloca seus participantes em canais. Apresenta um cenário interessante para um ambiente de desenvolvimento distribuído: embora funcione em tempo real, permite comunicação seletiva. Portanto, pode ser usado como um substituto razoável para um telefone quando questões técnicas estão sendo discutidas (REIS, 2002).

Em relação às listas de discussão, o fato delas apoiarem a troca de experiências na comunidade tem como efeito colateral a possibilidade de se arquivar essa comunicação para análise futura. Normalmente esses arquivos *online* são indexados por serviços de busca na *Web* e são utilizados como uma fonte importante de documentação para os usuários, já que frequentemente apresentam problemas recorrentes, já discutidos no passado por outros (REIS, 2002).

## 5.1.2 – O Projeto Apache

De forma análoga ao Mozilla, o Projeto Apache tem a gerência de configuração apoiada pelo CVS, possui seu próprio Wiki para auxílio à gerência de conhecimento (WIKI-APACHE, 2006), sua comunicação é fortemente baseada em listas de discussão classificadas de acordo com o papel do colaborador e há uma ferramenta de controle de defeitos, inicialmente BUGDB e atualmente o Bugzilla (APACHE, 2006).

O Projeto Apache utiliza, também, uma ferramenta de comunicação responsável por armazenar informações para análise futura: o Web-fórum. O Web-fórum é uma ferramenta que disponibiliza um espaço de discussão no qual seus membros podem inserir comentários a qualquer momento sobre um assunto abordado, caracterizando-se como uma ferramenta de comunicação assíncrona. A Figura 5.6 exemplifica um fórum criado para o Projeto Apache.



| new | Subject  | Messages | Started by      | Last Message                     |
|-----|--|----------|-----------------|----------------------------------|
|     | <a href="#">Bad bot.pl .htaccess clarification and updates</a><br>The order of things and script questions | 1        | jackhandy       | 9:33 am on Sep. 4, 2006 (utc 0)  |
|     | <a href="#">Forcing www for both http and https</a><br>updating current http-only mod_rewrite in .htaccess | 11       | Robert_Charlton | 7:00 am on Sep. 4, 2006 (utc 0)  |
|     | <a href="#">.htaccess hangs website</a><br>Trying to redirect /index.html requests to /                    | 7        | webman10        | 3:34 am on Sep. 4, 2006 (utc 0)  |
|     | <a href="#">Redirect entire site to temporary page</a><br>using htaccess                                   | 5        | chriswragg      | 2:26 am on Sep. 4, 2006 (utc 0)  |
|     | <a href="#">Rewrite Rules Created New Problem</a><br>forced / at end of directory                          | 4        | jackhandy       | 10:11 pm on Sep. 3, 2006 (utc 0) |
|     | <a href="#">Using .htaccess with ASP</a><br>Protecting both ASP pages and Excel files                      | 1        | VTwebworker     | 9:19 pm on Sep. 3, 2006 (utc 0)  |
|     | <a href="#">.htaccess for security and server optimization</a>   | 4        | notset          | 8:03 pm on Sep. 3, 2006 (utc 0)  |

Figura 5.6 – Tela principal de um Fórum Apache.

### 5.1.3 – O Projeto Linux

Apesar de ser um dos principais projetos de Software Livre existentes, o Projeto Linux é um dos projetos mais minimalistas em relação à infra-estrutura de desenvolvimento. Por muito tempo os desenvolvedores do *Kernel* do Linux possuíam somente o apoio de e-mails e listas de discussão para o desenvolvimento e manutenção do código fonte. Apesar de ainda não ser uma unanimidade, o Bitkeeper (BITKEEPER, 2006) está sendo usado como ferramenta para controle de versões. A Figura 5.7 apresenta um exemplo de utilização da ferramenta para analisar diferenças entre versões de código fonte (REIS, 2003). Além disso, também recentemente, o Bugzilla tem sido utilizado para apoiar o Projeto Linux em seu controle de defeitos e evoluções e wikis também são utilizados no apoio à gerência de conhecimento (WIKI-LINUX, 2006).

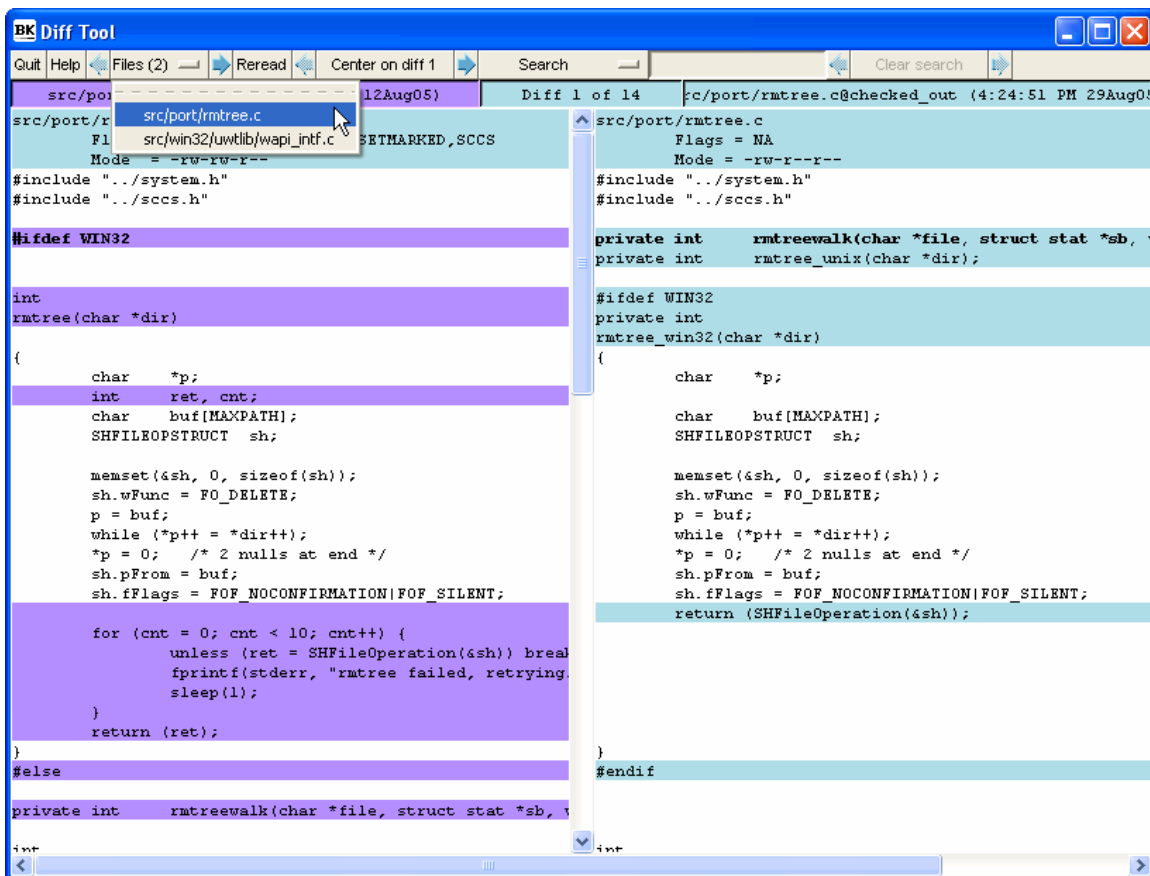


Figura 5.7 – Exemplo de utilização do Difftool do BitKeeper.

## **5.2 Requisitos de um Ambiente de Apoio ao Projeto ODE Livre**

A partir do levantamento feito sobre as características de apoio ferramental para o desenvolvimento de software dos principais projetos de Software Livre, procurou-se descrever requisitos mínimos de um ambiente de apoio ao Projeto ODE Livre.

Tendo como base o principal requisito de um ambiente de apoio ao desenvolvimento de Software Livre – possuir suas ferramentas disponíveis pela Internet – foi definido neste trabalho que todas as funcionalidades do ambiente de apoio ao Projeto ODE Livre deveriam ser passíveis de acesso a partir de um Portal *Web*, denominado Portal ODE Livre.

### **5.2.1 – Requisitos Funcionais Iniciais para o Portal ODE Livre**

Com base no levantamento realizado na seção anterior sobre as ferramentas utilizadas em projetos de SL e nas características específicas do projeto ODE, alguns requisitos foram definidos para a construção do Portal ODE Livre. Esses requisitos estão descritos em detalhes no Projeto de Graduação de Julierme Silva (SILVA, 2007), que foi fortemente baseado nas características e conceitos levantados neste trabalho.

Para a primeira versão do portal, foram identificadas funcionalidades de apoio ao controle de usuários e contribuições, sendo importante frisar que os requisitos levantados não obrigam que a construção do Portal do ODE Livre se dê por meio do desenvolvimento de um novo produto de software. Ou seja, nada impede que seu desenvolvimento se dê pela incorporação de ferramentas livres que atendam aos requisitos especificados para o Portal ODE Livre.

O gerenciamento e o acesso ao Portal do Projeto ODE Livre serão feitos de acordo com o tipo de usuário. Para tal os usuários são agrupados nas seguintes categorias: Internauta, Usuário Padrão, Membro LabES, Colaborador, Professor e Administrador, como mostra a Figura 5.8.

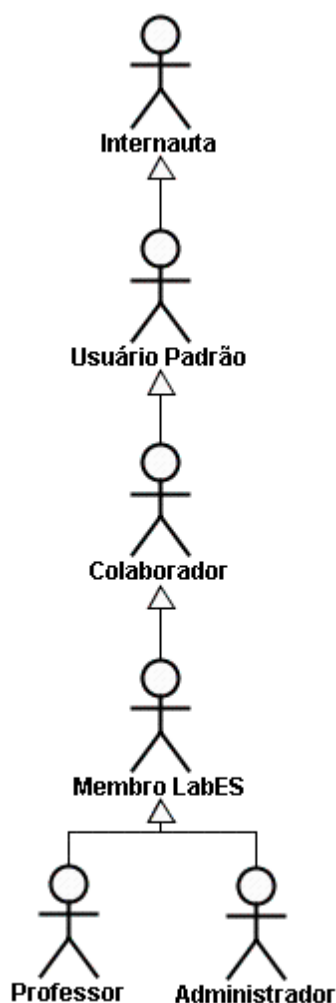


Figura 5.8 - Tipos de Usuários

O Portal ODE Livre é um portal direcionado ao apoio ao desenvolvimento de ODE como Software Livre, um projeto desenvolvido no LabES. Porém, existe também o Portal LabES, uma porta de entrada na Internet para o LabES, cujo propósito é facilitar a interação do LabES com o público interessado na área de Engenharia de Software. Sendo assim, o Portal do ODE Livre está relacionado ao Portal do LabES e os tipos de usuários acima exibidos visam tratar o escopo de ambos os portais.

O *Internauta* representa qualquer pessoa que esteja navegando na Internet. Esse tipo de usuário pode se cadastrar como um *Usuário Padrão*, tem acesso às funcionalidades de visualização de qualquer material ou informação disponível e pode efetuar *download* de publicações e materiais do Portal do LabES.

Um *Usuário Padrão* representa usuários cadastrados no Portal do LabES / Portal do ODE Livre. Esse tipo de usuário pode alterar seus dados e sua senha, e terá de se autenticar no sistema para ter acesso às funcionalidades específicas dessa classe de

usuários. Apenas usuários padrão poderão fazer o *download* do ambiente ODE. Além disso, poderão relatar falhas do ambiente e propor novas funcionalidades para o mesmo.

O *Colaborador* representa um usuário padrão que se dispôs a colaborar com o Projeto ODE Livre, corrigindo alguma falha ou desenvolvendo uma nova funcionalidade para o ambiente.

A classes de usuários *Membros de LabES* agrupa todos os usuários que atuam dentro do laboratório. Esse tipo de usuário pode consultar os dados de qualquer outro membro do LabES e pode registrar materiais a serem disponibilizados no portal.

*Professor* representa os professores associados ao LabES. Esse tipo de usuário pode disponibilizar publicações e projetos, cadastrar áreas de pesquisa, além de poder consultar os dados dos membros do LabES e registrar materiais, como qualquer membro do LabES.

Por fim, a classe de usuários *Administrador* representa os administradores do sistema, que têm permissão para cadastrar novas funcionalidades, tipos de usuários, membros do LabES, professores e outros administradores. De fato, um administrador tem acesso a todas as funcionalidades do sistema.

No que se refere ao Portal do ODE Livre, neste levantamento inicial, dois subsistemas foram definidos, como mostra a Figura 5.9:

- **Controle de Usuários:** envolve toda a funcionalidade relacionada com o controle de usuários do Portal ODE Livre / Portal do LabES, abrangendo controle de funcionalidades, tipos de usuários e usuários.
- **Controle de Contribuições:** disponibiliza funcionalidades que permitem gerenciar as falhas (*bugs*) relatadas e as novas funcionalidades que são sugeridas pelos colaboradores do projeto.

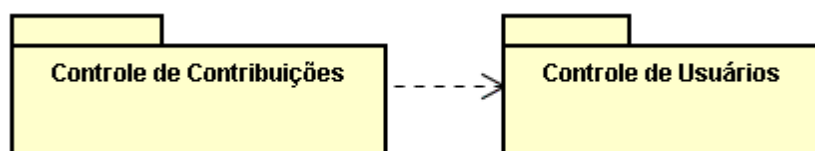


Figura 5.9 – Diagrama de Pacotes do Portal ODE Livre

Para cada um desses subsistemas, um conjunto de funcionalidades inicial foi identificado e modelado na forma de casos de uso. A Figura 5.10 mostra o diagrama de casos de uso do subsistema Controle de Usuários, cujas funcionalidades são (FIORIO, 2005):

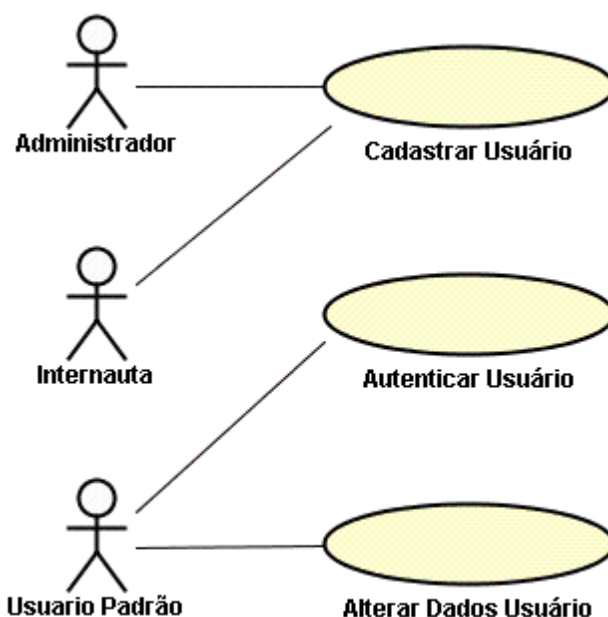


Figura 5.10: Diagrama de Casos de Uso do subsistema de Controle de Usuários.

- **Cadastrar Usuário:** este caso de uso é responsável pelo controle de usuários, abrangendo a criação de um novo usuário, consulta e exclusão de usuários existentes.
- **Autenticar Usuário:** este caso de uso é responsável pela autenticação do usuário no sistema, abrangendo a efetuação de *login*, *logout* e envio de senha.
- **Alterar Dados Usuário:** Este caso de uso permite que um usuário padrão efetue a alteração de seus dados pessoais e senha.

A Figura 5.11 mostra o diagrama de casos de uso referente ao subsistema de Controle de Contribuições. Nesse subsistema atuam os atores Membro LabES e Colaborador, para os quais são disponibilizadas funcionalidades relacionadas ao controle de contribuições feitas por colaboradores ao Projeto ODE Livre. Uma descrição sucinta de cada caso de uso é apresentada a seguir. Para maiores detalhes, vide (SILVA, 2007).



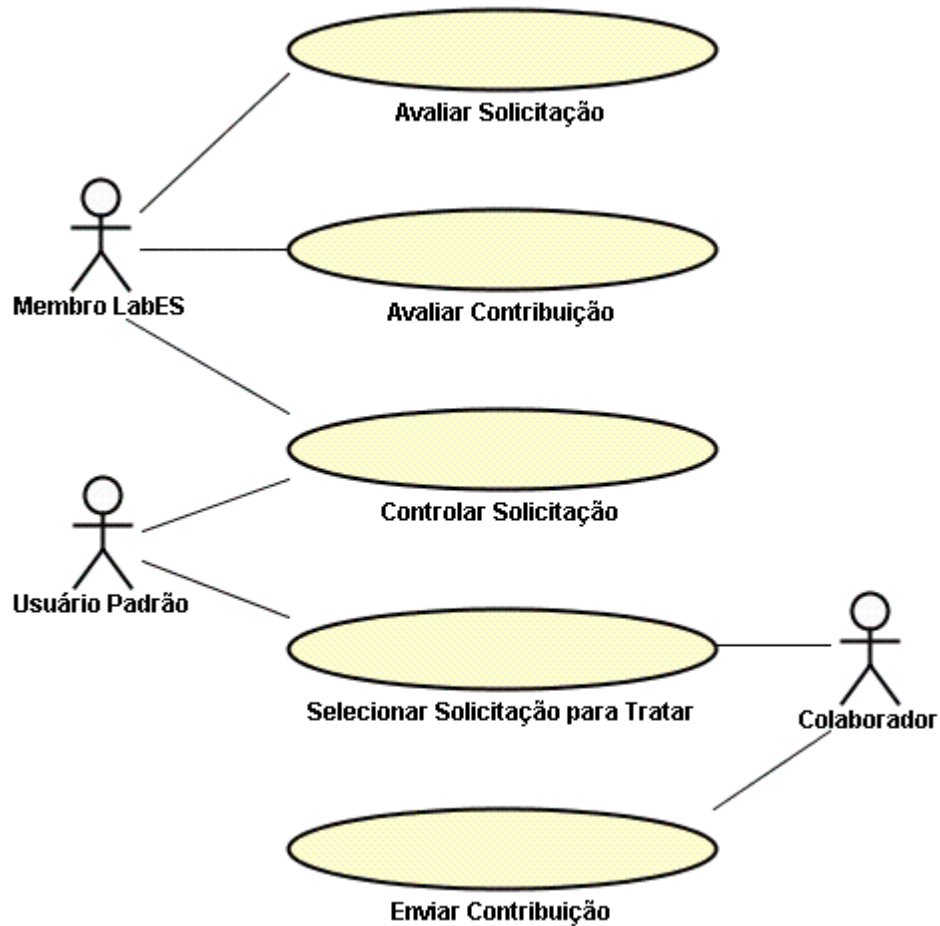


Figura 5.11 – Diagrama de Casos de Uso do subsistema de Controle de Contribuições.

- **Controlar Solicitação:** este caso de uso é responsável pelo controle de solicitações feitas pelos usuários de ODE. Por meio dele, um usuário padrão pode solicitar a correção de uma falha por ele detectada no ambiente ou pode sugerir uma nova funcionalidade para o ambiente, dentre outros.
- **Avaliar Solicitação:** este caso de uso trata da avaliação de solicitações. Por meio dele, membros do LabES consultam solicitações e registram avaliações das mesmas, aprovando ou rejeitando solicitações feitas pelos usuários. Solicitações aprovadas são preparadas para serem colocadas disponíveis para colaboradores no Portal.

- **Selecionar Solicitação para Tratar:** este caso de uso permite que um Colaborador selecione uma solicitação, se responsabilizando por tratá-la, isto é, por desenvolver uma solução para a solicitação.
- **Enviar Contribuição:** este caso de uso permite que um colaborador envie uma contribuição ao Projeto ODE Livre
- **Avaliar Contribuição:** este caso de uso permite que um membro do LabES registre a avaliação de uma contribuição enviada por um usuário ao Projeto ODE Livre.

É importante frisar que, dados os requisitos funcionais descritos anteriormente na forma de modelos de casos de uso, nada impede que o Portal ODE Livre seja construído pela integração da ferramenta livre de acompanhamento de alterações Bugzilla, visto que tal ferramenta abrange todos os requisitos citados (BUGZILLA, 2006). Porém, a escolha de uma opção de construção de uma ferramenta própria de controle de contribuições e alterações seria justificada por algumas particularidades:

- Para uma melhor integração do Portal ao próprio Projeto ODE;
- Criação de novas funcionalidades particulares ao projeto;
- Automatização do processo de distribuição de pedidos de apoio (envio de solicitações e contribuições) para colaboradores (desenvolvedores e avaliadores), utilizando-se mecanismos de gerência de conhecimento para tal.

Como dito no início desta seção, os requisitos aqui definidos não obrigam a construção própria de um novo produto de software.

### **5.2.2 – Outros Requisitos Gerais Desejados para o Portal ODE Livre**

Além dos requisitos funcionais especificados para o Portal ODE Livre, para que este se torne um ambiente adequado para o Projeto ODE Livre, há, ainda, outros requisitos gerais a serem considerados, tipicamente tratados em projetos de SL e levantados a partir de colocações feitas em (NAKAGAWA, 2002) e (REIS, 2003), dentre outros. Esses serviços devem ser disponibilizados via Portal do projeto e incluem:

- **Endereço de correio eletrônico do Projeto:** Um endereço eletrônico do projeto deve ser disponibilizado, permitindo aos usuários o contato direto com o coordenador do projeto ou com os responsáveis pela integração (membros do LabES).
- **Lista de discussão:** O projeto deve apresentar uma série de listas, cada uma para um público-alvo. Decidiu-se criar dois tipos de listas, sendo que cada um desses tipos é subdividido em outras listas. As listas mais importantes no sentido de promover o desenvolvimento são:
  - **Lista de usuários:** Nessa lista, circulam mensagens referentes ao uso e operação do ambiente. Essa lista é importante por criar uma comunidade ao redor do software e por facilitar a comunicação referente a pedidos de alterações e problemas. Em geral, nessa lista concentram-se usuários avançados, que respondem às perguntas mais básicas, e alguns desenvolvedores, que captam mensagens que têm impacto no desenvolvimento e respondem às questões mais complexas. Este tipo de lista pode ser subdividido em listas para cada ferramenta de ODE, incluindo o Portal;
  - **Lista de Desenvolvedores:** Nessa lista são discutidas decisões de projeto. Em geral, essa lista tem alto tráfego e apresenta discussões abordando questões mais elaboradas. Esse tipo de lista pode ser subdividido em listas para cada ferramenta de ODE ou até mesmo pelo nível hierárquico dos colaboradores;
- **Fóruns e Chats:** Listas de discussão são importantes ferramentas de comunicação, porém, por ser um projeto de SL, é muito importante a utilização de outras ferramentas de comunicação, tais como fóruns e *chats*. A integração de uma ferramenta livre de fórum ao Portal ODE Livre pode permitir uma forma de comunicação assíncrona mais eficiente e organizada dos colaboradores, por meio da subdivisão de tópicos de discussão e visibilidade controlada. *Chats* integrados, por sua vez, possibilitam a existência de comunicação síncrona, ideal para colaborações dinâmicas.

- **Repositório de software:** Possibilita que versões (mais recentes ou não) do ambiente possam ser obtidas. Esse repositório deve ser disponibilizado no Portal ODE Livre;
- **Repositório de controle de versão:** É um requisito fundamental para que o desenvolvimento distribuído seja possível e agilizado por meio do controle e análise das diversas versões do software. Atualmente, o LabES controla a versão de seus produtos utilizando a ferramenta Subversion (SUBVERSION, 2006). Assim, deve-se estudar a utilização desta ferramenta de forma integrada ao Portal ODE Livre .
- **Wiki:** Esta ferramenta possibilita a inserção de conhecimento para todos os envolvidos no projeto a partir da criação de páginas *Web* livres. Vários Wikis referentes às ferramentas de ODE podem ser criados no intuito de fornecer conhecimento aos usuários e colaboradores do projeto. Os wikis são importantes por impedir que o conhecimento fique retido com determinados membros colaboradores, disseminando-o com os demais.

### 5.2.3 – Utilizando ODE no Projeto ODE Livre

Pelo fato de ODE ser um Ambiente de Desenvolvimento de Software (ADS), cujo objetivo é fornecer apoio ao longo de todo o processo de software ou em porções significativas dele, seria um “desperdício” deixar de utilizar as diversas ferramentas providas por ODE para auxiliar o seu próprio desenvolvimento como Software Livre.

Conforme discutido anteriormente, idealmente, seria interessante que todas as ferramentas de apoio fossem disponibilizadas pela *Web* para que os colaboradores do Projeto ODE Livre pudessem utilizá-las. Assim, o próprio ambiente ODE e todas as suas ferramentas teriam de ser aplicações *Web*, o que não ocorre.

Tendo esta limitação em mente, foi estabelecida uma nova funcionalidade para ODE: a importação e exportação de projetos no formato XML. Dessa forma, este requisito possibilita que um colaborador utilize qualquer ferramenta de ODE para produzir um artefato, de forma *off-line*, gere um arquivo XML correspondente ao projeto – contendo um ou mais artefatos produzidos – e submeta este arquivo ao Projeto ODE Livre através do

Portal ODE Livre, utilizando a funcionalidade de Controle de Contribuições. De forma similar, colaboradores que queiram utilizar ou alterar os artefatos produzidos por outros poderiam fazer o *download* dos arquivos XML correspondentes e importá-los para a versão de ODE local em sua máquina. Desta forma, consegue-se, parcialmente, a integração de ODE ao Portal, tornando-o mais completo e permitindo a colaboração em vários artefatos de projeto.

### **5.3 Considerações Finais do Capítulo**

Definido o processo padrão a ser seguido para o desenvolvimento e manutenção de ODE Livre, é preciso fornecer apoio ferramental para viabilizar a implantação e uso de tal processo.

Neste capítulo, foram levantadas características de apoio ferramental a projetos de SL através de pesquisas feitas, principalmente, a três dos projetos de maior destaque em âmbito mundial: Projeto Mozilla, Projeto Apache e Projeto Linux.

A partir do levantamento realizado, requisitos foram estabelecidos para a elaboração do Portal ODE Livre. Todos os requisitos levantados têm como necessidade básica o acesso a suas funcionalidades pela Internet. Parte desses requisitos deverá ser incorporada ao projeto do Portal ODE Livre, já em andamento, que em sua versão atual possui funcionalidades definidas para controle de usuários e controle de contribuições.

Finalmente, este capítulo apresentou uma forma de utilizar ODE para apoiar seu próprio desenvolvimento, por meio de mecanismos de importação e exportação de arquivos XML.

# Capítulo 6

## Considerações Finais

Os principais objetivos deste trabalho foram definir processos padrão de qualidade para o desenvolvimento de Software Livre e estabelecer uma estrutura computacional de apoio adequada para tal, tendo como estudo de caso o Projeto ODE Livre. Só foi possível alcançar este objetivo através da realização de um levantamento detalhado sobre os dois conceitos principais deste trabalho: qualidade de software, incluindo normas e modelos de qualidade, e características de projetos de software livre bem sucedidos, incluindo ferramentas de apoio ao seu desenvolvimento.

Este capítulo procura destacar as contribuições deste trabalho para a comunidade de software em geral e, mais especificamente, para os envolvidos no desenvolvimento de Software Livre. Apesar de ter alcançado o principal objetivo previamente imaginado, algumas questões ficaram em aberto e novas perspectivas surgiram para trabalhos futuros relacionados.

### 6.1 Conclusões

O desenvolvimento de Software Livre (SL) tem evoluído bastante, proporcionando o surgimento de um número cada vez maior de projetos de SL e adquirindo importância considerável no cenário mundial de desenvolvimento de software.

O SL trouxe à comunidade de software, uma nova forma de se desenvolver software, diferente da tradicional, acarretando mudanças culturais, políticas e econômicas. Sua principal característica é a liberdade, ou seja, a distribuição dos arquivos binários de um sistema livre deve ser feita junto ao código fonte, permitindo o estudo e alteração a qualquer pessoa e, dependendo da licença de software utilizada, a nova liberação após alterações também deve respeitar as mesmas regras. Em relação ao desenvolvimento, é

caracterizado, na maior parte das vezes, pelo trabalho voluntário e pela distribuição geográfica dos envolvidos.

A partir de uma análise sobre o cenário atual do desenvolvimento de software livre, constatou-se não haver nenhum projeto de SL desenvolvido com a utilização de processos baseados em normas e modelos de qualidade mundialmente reconhecidos. Assim, o objetivo principal deste trabalho foi reforçado e decidiu-se elaborar processos padrão para o desenvolvimento de software livre no Laboratório de Engenharia de Software (LabES) da Universidade Federal do Espírito Santo, associados a requisitos de um ambiente de apoio ferramental a esses processos.

Após considerável estudo bibliográfico sobre qualidade de software e características de desenvolvimento de SL, foram elaborados os processos padrão para desenvolvimento e manutenção de Software Livre no LabES. Visando adequar uniformemente todos os processos padrão do LabES (livres e não livres), decidiu-se reformular o processo padrão de desenvolvimento LabES nos moldes sugeridos pelo MPS.BR e criou-se o processo padrão de manutenção LabES com base no padrão IEEE 1219.

A partir da elaboração dos processos padrão, era necessário definir o ambiente ferramental de apoio aos processos. Devido a limitações de prazos, decidiu-se definir os requisitos necessários para o ambiente proposto – o Portal ODE Livre, contando com a ajuda de Julierme Silva (SILVA, 2007) para tal.

Assim, foram contribuições deste trabalho:

- Elaboração do Processo Padrão LabES de Desenvolvimento de SL nos moldes sugeridos pelo MPS.BR;
- Elaboração do Processo Padrão LabES de Manutenção de SL nos moldes sugeridos pelo MPS.BR e com base no padrão IEEE 1219;
- Elaboração do Processo Padrão LabES de Manutenção nos moldes sugeridos pelo MPS.BR e com base no padrão IEEE 1219;
- Revisão do Processo Padrão LabES de Desenvolvimento nos moldes sugeridos pelo MPS.BR;
- Definição da GNU-GPL como licença livre sob a qual ODE Livre deverá ser disponibilizado;
- Definição dos Requisitos para a construção do Portal ODE Livre.

Durante o desenvolvimento deste trabalho, algumas limitações foram identificadas. Dentre elas, pode-se citar a necessidade de se aplicar os processos propostos em projetos piloto no intuito de avaliá-los e melhorá-los. Para que se pudesse efetivar essa aplicação, seria necessário um tempo maior que, comumente, não é disponível à elaboração de uma dissertação de mestrado. Outra limitação enfrentada neste trabalho foi a impossibilidade de se construir o Portal ODE Livre para averiguar se os requisitos levantados são suficientes para atender ao objetivo do projeto.

## **6.2 Perspectivas Futuras**

Conforme discutido na seção anterior, uma vez definidos os processos padrão para o Projeto ODE Livre, é preciso aplicá-los em projetos piloto para que sejam feitos os ajustes necessários, aperfeiçoando-os a ponto de serem realmente disponibilizados para apoiarem o desenvolvimento de ODE como Software Livre. Inicialmente, espera-se disponibilizar ODE para parceiros do LabES, criando instâncias dos processos padrão elaborados para cada projeto desenvolvido por uma organização colaboradora, de acordo com suas peculiaridades. Após feitos os ajustes necessários, ODE será efetivamente disponibilizado como SL.

Dada a necessidade de se fornecer apoio ferramental aos colaboradores para facilitar a aplicação dos processos definidos, é importante não só concluir a construção do Portal ODE Livre com os requisitos levantados no Capítulo 5, mas também procurar definir novas ferramentas para melhorar a qualidade do apoio fornecido. Neste sentido, é importante evoluir as pesquisas relacionadas o como utilizar ODE no desenvolvimento de ODE. Há uma expectativa de que os benefícios decorrentes do uso de ferramentas de apoio ao desenvolvimento de ODE Livre seriam muito maiores se o próprio ambiente ODE pudesse ser usado integrado às demais ferramentas do Portal. Uma das formas de atingir tal integração seria através da disponibilização de ODE pela Internet, ou seja, transformando-o numa aplicação *Web*.



O objetivo de transformar ODE em Software Livre tem sido visto como muito promissor pelo LabES, principalmente, pelo fato de se poder vislumbrar a evolução de ODE contando com o apoio da comunidade de software brasileira. A crença neste interesse é depositada na considerável quantidade de funcionalidades oferecidas por ODE, um ADS centrado em processos.

## Referências Bibliográficas

- APACHE, Apache Software Foundation. Disponível em: <http://www.apache.org/>. Acesso em: 13/09/2006.
- ARANTES, L. O., **Apoio Automatizado à Gerência de Projetos**, Projeto de Graduação, Curso de Ciência da Computação, UFES, 2006.
- BERTOLLO, G., **Definição de Processos em um Ambiente de Desenvolvimento de Software**, Dissertação de Mestrado, Mestrado em Informática, Universidade Federal do Espírito Santo, Maio de 2006.
- BERTOLLO, G.; RUY, F.B.; MIAN, P.G.; PEZZIN, J.; SHWAMBACH, M.M.; NATALI, A.C.C.; FALBO, R.A., **ODE: Um ambiente de Desenvolvimento de Software Baseado em Ontologias**. Anais do XVI Simpósio Brasileiro de Engenharia de Software - SBES'2002. Caderno de Ferramentas, pp. 438-443, Gramado - RS, Brasil, Outubro 2002.
- BERTOLLO, G.; FALBO, R. A., **Apoio Automatizado à Definição de Processos em Níveis**. Anais do II Simpósio Brasileiro de Qualidade de Software, p. 77-91, Fortaleza, Brasil, Setembro 2003.
- BERTOLLO, G., SEGRINI, B., FALBO, R.A., **Definição de Processos de Software em um Ambiente de Desenvolvimento de Software Baseado em Ontologias**. Anais do V Simpósio Brasileiro de Qualidade de Software, p. 61 – 75, Vila Velha, Brasil, Maio 2006.
- BITKEEPER, The Scalable Distributed Software Configuration Management System. Disponível em: <http://www.bitkeeper.com/>. Acesso em: 13/09/2006.
- BONSAI, The Mozilla Organization. Bonsai. 2000. Disponível em: <http://www.mozilla.org/bonsai.html>. Acesso em: 13/09/2006.
- BUGZILLA, Bugzilla Bug Tracking System. Disponível em: <http://www.bugzilla.org/>. Acesso em: 13/09/2006.
- CHRISISS, M. B.; KONRAD, M.; SHRUM, S., **Knowledge Management of Software Risks**, Journal of Universal Computer Science, Volume 9, Number 7, 670-681, 2003.

- CREATIVE COMMONS, Home Page. Disponível em: <http://creativecommons.org>. Acesso em: 11/11/2006.
- DAL MORO, R.; NARDI, J. C.; FALBO, R. A., **ControlPro: Uma Ferramenta de Acompanhamento de Projetos Integrada a um Ambiente de Desenvolvimento de Software**, Anais da Sessão de Ferramentas do XIX Simpósio Brasileiro de Engenharia de Software – SBES'2005, Uberlândia - MG, Brasil, Outubro 2005.
- DAVIS, M., O'DONOVAN, W., FRITZ, J. **Linux and open source in the academic enterprise**. In: Proc. of the 28th SIGUCCS Conference on User Services, Richmond, 2000.
- ENGELFRIET, A. Choosing a Software License, 2003 <http://www.iusmentis.com/computerprograms/licenses/pcactive0203/>. Capturado em: 13/09/2006.
- FALBO, R. A. **Integração de Conhecimento em um Ambiente de Desenvolvimento de Software**. Tese de Doutorado, COPPE/UFRJ, RJ, Dezembro, 1998.
- FALBO, R.A.; NATALI, A.C.C.; MIAN, P.G.; BERTOLLO, G.; RUY, F.B. **ODE: Ontology-based software Development Environment**, IX Congreso Argentino de Ciencias de la Computación, p. 1124-1135, La Plata, Argentina, outubro 2003.
- FALBO, R.A., RUY, F.B., PEZZIN, J., DAL MORO R. **Ontologias e Ambientes de Desenvolvimento de Software Semânticos**. Actas de las IV Jornadas Iberoamericanas de Ingeniería del Software e Ingeniería Del Conocimiento, JIISIC'2004, Volumen I, pp. 277-292, Madrid, España, Noviembre 2004.
- FIORIO, G. C., **Desenvolvimento de um Portal para o Laboratório de Engenharia de Software**, Projeto de Graduação, Curso de Ciência da Computação, UFES, 2005.
- FREEBSD, The 4.4BSD Copyright. Disponível em: <http://www.freebsd.org/copyright/license.html>. Acesso em: 13/09/2006.
- FREE SOFTWARE FOUNDATION, O que é software livre?. Disponível em: <http://www.gnu.org/philosophy/free-sw.pt.html>. Acesso em: 13/09/2006.
- FREITAS, A. M.; NARDI, J. C.; FALBO, R. A., **ReqODE: Uma Ferramenta de Apoio à Engenharia de Requisitos Integrada ao Ambiente ODE**, Anais da Sessão de

- Ferramentas do XX Simpósio Brasileiro de Engenharia de Software – SBES’2006, Florianópolis- SC, Brasil, Outubro 2006.
- FUGGETTA, A., **Software Process: A Roadmap**, In: Proc. of The Future of Software Engineering, ICSE’2000, Limerick, Ireland, 2000, pp. 25-34.
- GPL, GNU General Public License. Disponível em: <http://www.gnu.org/copyleft/gpl.html>. Acesso em: 13/09/2006.
- GRUHN, V., **Process-Centered Software Engineering Environments: A Brief History and Future Challenges**, In: Annals of Software Engineering 14, 2002, pp. 363-382.
- HARRISON, W., OSSHER, H., TARR, P. **Software Engineering Tools and Environments: A Roadmap**. Proceedings of The Future of Software Engineering (ICSE’2000). Limerick, Ireland, 2000.
- HEXSEL, R. A., **Software Livre – Propostas de Ações do Governo para Incentivar o Uso de Software Livre**. Relatório Técnico – RT-DINF 004/2002, Universidade Federal do Paraná, PR, 2002.
- IEEE Standards Boards, IEEE Std 1219-1998: **Standard for Software Maintenance**, jun. 1998.
- ISO/IEC 12207, **Information Technology - Software life cycle processes**, 1995.
- ISO/IEC 12207, **Information Technology - Software life cycle processes, Amendment 1**, 2002.
- ISO/IEC 12207, **Information Technology - Software life cycle processes, Amendment 2**, 2004.
- ISO/IEC 15504 **Information Technology – Process Assessment – Part 1: Concepts and Vocabulary**, 2003.
- ISO/IEC 15504 **Information Technology – Process Assessment – Part 1: An exemplar Process Assessment Model**, 2004.
- ISO 9000:2000, **Sistemas de gestão da qualidade – Fundamentos e vocabulário**, 2000.
- LINUX, The Linux Home Page at Linux Online. Disponível em: <http://www.linux.org/>. Acesso em: 13/09/2006.

- LXR, Mozilla Cross-Reference. Disponível em: <http://lxr.mozilla.org/>. Acesso em: 11/11/2006.
- MAIDANTCHIK, C. L. L. **Gerência de Processos de Software para Equipes Geograficamente Dispersas**. Tese de Doutorado, COPPE/UFRJ, RJ, Junho, 1999.
- MIAN, P.G.; NATALI, A. C. C.; FALBO, R. A., **Ambientes de Desenvolvimento de Software e o Projeto ADS**, Revista Engenharia, Ciência e Tecnologia, Volume 04, Número 04, ISSN 1414-8692, pp. 3 - 10, Vitória, Brasil, Julho/Agosto 2001.
- MOCKUS, A., Fielding, R. T., Herbsleb, J. **Two case studies of open source software development: Apache and Mozilla**. ACM Transactions on Software Engineering and Methodology, v. 11, n. 3, 2002.
- MUTAFELIJA, B., STROMBERG, H. **Systematic Process Improvement Using ISO 9001:2000 And CMMI**, Boston, London: Artech House Publishers, 2003.
- MYSQL, Home Page. Disponível em: <http://www.mysql.com> . Acesso em: 13/09/2006.
- NAKAGAWA, E. Y. **Software Livre: Processo e Produto Livres no Desenvolvimento de Aplicações Web**, Exame de Qualificação ao Doutorado, Instituto de Ciências Matemáticas e de Computação – ICMC/USP, São Carlos, 2002.
- POSTGRESQL, Home Page. Disponível em: <http://www.postgresql.org> . Acesso em: 13/09/2006.
- PHP, Home Page. Disponível em: <http://www.php.net> . Acesso em: 13/09/2006.
- PERL, Home Page. Disponível em: <http://www.perl.com> . Acesso em: 13/09/2006.
- PFLEEGER, S.L., **Software Engineering: Theory and Practice**, 2<sup>nd</sup> Edition, New Jersey: Prentice Hall, 2001.
- PRESSMAN, R.S., **Software Engineering: A Practitioner's Approach**, 5th Edition, New York: McGraw-Hill, 2000.
- PRESSMAN, R. S., **Engenharia de Software**. 5ª edição. Rio de Janeiro: McGrawHill, 2002.
- RAYMOUND, E., **The cathedral and the bazaar**. O'Reilly & Associates, 1999.
- REIS, C. R., **Caracterização de um Processo de Software para Projetos de Software Livre**. Dissertação de Mestrado, ICMC/USP, São Carlos, 2003.

- REIS, C. R., FORTES, R. P. M., **An Overview of the Software Process and Tools in the Mozilla Project**. Proceedings of the Open Source Software Development Workshop, p. 155-175, Newcastle Upon Tyne, United Kingdom, 2002.
- ROCHA, A. R. C.; MALDONADO, J. C.; WEBER K. C. **Qualidade de Software: Teoria e Prática**. 1ª. Edição. Editora Prentice Hall. São Paulo. 2001.
- SCACCHI, W., **Understanding the requirements for developing open source software systems**. In: IEE Proceedings – Software Engineering, 149(1), p. 24-39, 2002.
- SCHWAMBACH, M. M., **Apoio Automatizado ao Planejamento e Acompanhamento de Projetos de Software**, Projeto de Graduação, Curso de Ciência da Computação, UFES, 2002.
- SEI, **CMMI-SE/SW, versão 1.1 – Representação em Estágios**, 2002.
- SEI, **Standard CMMI Appraisal Method for Process Improvement (SCAMPI), Version 1.1: Method Definition Document**, CMU/SEI-2001-HB-001, 2001.
- SILVA B. C. C., FALBO R. A., 2006, **Definição de um Processo Padrão para Software Livre, SBQS 2006, V Simpósio Brasileiro de Qualidade de Software**, Anais do V Simpósio Brasileiro de Qualidade de Software, p. 159-173, Vila Velha, Brasil, Maio 2006.
- SILVA, J. L., **Portal do Projeto ODE Livre**, Projeto de Graduação, Curso de Ciência da Computação, UFES, 2007 (em andamento, conclusão prevista para Julho de 2007).
- SOFTEX, “O impacto do software livre e de código aberto na indústria de software do Brasil”, Softex Campinas, 2005a [on line]. Disponível: [www.softex.br](http://www.softex.br) [capturado em Abril 2006].
- SOFTEX, MPS.BR – Melhoria de Processo do Software Brasileiro: Guia Geral, Versão 1.0, 2005b [on line]. Disponível: [www.softex.br/mpsbr](http://www.softex.br/mpsbr) [capturado em Abril 2006].
- SOFTEX, MPS.BR – Melhoria de Processo do Software Brasileiro: Guia Geral, Versão 1.1, 2006 [on line]. Disponível: [www.softex.br/mpsbr](http://www.softex.br/mpsbr) [capturado em Agosto 2006].
- SOURCEFORGE, Home Page. Disponível em: <http://sourceforge.net/index.php>. Acesso em: 12/11/2006.
- SUBVERSION, Home Page. Disponível em: <http://subversion.tigris.org/>. Acesso em: 13/09/2006.

- TRAVASSOS, G.H. **O Modelo de Integração de Ferramentas da Estação TABA**. Tese de D.Sc., Programa de Engenharia de Sistemas e Computação, COPPE/UFRJ, Rio de Janeiro, RJ, Brasil, 1994.
- VIXIE, P., **Software Engineering, In: Open sources: Voices of the open source revolution**, 1<sup>st</sup> edition, O'Reilly & Associates, p. 91–100, 1999.
- WEBER, C. K, ROCHA, A. R; ALVES, A, AYALA, A. M, GONÇALVES, A, PARET, B, SALVIANO, C; MACHADO, C. F, SCALET, D, PETIT, D, ARAÚJO, E, BARROSO, M. G, OLIVEIRA, K, OLIVEIRA, L. C. A, AMARAL, M. P, CAMPELO, R. E. C; MACIEL, T., **Modelo de Referência para Melhoria de Processo de Software: uma abordagem brasileira**, XXX Conferencia Latino-americana de Informática, Arequipa, Peru, 2004.
- WEBER, K.C., **Mudanças na norma ISO 9000**, In: Qualidade de Software: Teoria e Prática, Eds. A.R.C. Rocha, J.C. Maldonado, K. Weber, Prentice Hall, 2001.
- WIKI, What is Wiki. Disponível em: <http://www.wiki.org/wiki.cgi?WhatIsWiki>. Acesso em: 13/09/2006.
- WIKI-APACHE, Home Page. Disponível em: <http://wiki.apache.org> . Acesso em: 13/09/2006.
- WIKI-LINUX, WIKI Linux-NTFS Home Page. Disponível em: <http://wiki.linux-ntfs.org>. Acesso em: 13/09/2006.
- WIKI-MOZILLA, Home Page. Disponível em: <http://wiki.mozilla.org>. Acesso em: 13/09/2006.
- XAVIER, J.H.F. **ISO 9000 aplicada ao software**, In: Qualidade e Produtividade em Software, 4<sup>a</sup> edição renovada, Organizadores:WEBER, K.C., ROCHA, A.R.C. e NASCIMENTO, C.J., Makron Books, 2001.

# Livros Grátis

( <http://www.livrosgratis.com.br> )

Milhares de Livros para Download:

[Baixar livros de Administração](#)

[Baixar livros de Agronomia](#)

[Baixar livros de Arquitetura](#)

[Baixar livros de Artes](#)

[Baixar livros de Astronomia](#)

[Baixar livros de Biologia Geral](#)

[Baixar livros de Ciência da Computação](#)

[Baixar livros de Ciência da Informação](#)

[Baixar livros de Ciência Política](#)

[Baixar livros de Ciências da Saúde](#)

[Baixar livros de Comunicação](#)

[Baixar livros do Conselho Nacional de Educação - CNE](#)

[Baixar livros de Defesa civil](#)

[Baixar livros de Direito](#)

[Baixar livros de Direitos humanos](#)

[Baixar livros de Economia](#)

[Baixar livros de Economia Doméstica](#)

[Baixar livros de Educação](#)

[Baixar livros de Educação - Trânsito](#)

[Baixar livros de Educação Física](#)

[Baixar livros de Engenharia Aeroespacial](#)

[Baixar livros de Farmácia](#)

[Baixar livros de Filosofia](#)

[Baixar livros de Física](#)

[Baixar livros de Geociências](#)

[Baixar livros de Geografia](#)

[Baixar livros de História](#)

[Baixar livros de Línguas](#)



[Baixar livros de Literatura](#)  
[Baixar livros de Literatura de Cordel](#)  
[Baixar livros de Literatura Infantil](#)  
[Baixar livros de Matemática](#)  
[Baixar livros de Medicina](#)  
[Baixar livros de Medicina Veterinária](#)  
[Baixar livros de Meio Ambiente](#)  
[Baixar livros de Meteorologia](#)  
[Baixar Monografias e TCC](#)  
[Baixar livros Multidisciplinar](#)  
[Baixar livros de Música](#)  
[Baixar livros de Psicologia](#)  
[Baixar livros de Química](#)  
[Baixar livros de Saúde Coletiva](#)  
[Baixar livros de Serviço Social](#)  
[Baixar livros de Sociologia](#)  
[Baixar livros de Teologia](#)  
[Baixar livros de Trabalho](#)  
[Baixar livros de Turismo](#)