

UNIVERSIDADE REGIONAL DE BLUMENAU
CENTRO DE CIÊNCIAS EXATAS E NATURAIS
CURSO DE CIÊNCIAS DA COMPUTAÇÃO – BACHARELADO

FERRAMENTA DE APOIO A GERÊNCIA DE
CONFIGURAÇÃO DE SOFTWARE

RODRIGO FURLANETO

BLUMENAU
2006

2006/2-26

RODRIGO FURLANETO

**FERRAMENTA DE APOIO A GERÊNCIA DE
CONFIGURAÇÃO DE SOFTWARE**

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

Prof. Everaldo Artur Grahl - Orientador

**BLUMENAU
2006**

2006/2-26

FERRAMENTA DE APOIO A GERÊNCIA DE CONFIGURAÇÃO DE SOFTWARE

Por

RODRIGO FURLANETO

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

Presidente: _____
Prof. Everaldo Artur Grahl - Orientador, FURB

Membro: _____
Prof. Fabiane B. V. Benitti, FURB

Membro: _____
Prof. Carlos E. Negrão Bizzotto, FURB

Blumenau, 13 de Dezembro de 2006

Dedico este trabalho a minha família, em especial ao meu Pai (Zenilto Furlaneto “in memoriam”). Por terem dado estímulo que me impulsionou a buscar uma vida melhor. Por terem se privado de muitos de seus sonhos, concedendo a mim a oportunidade de realizar o meu sonho.

AGRADECIMENTOS

Agradeço a realização deste trabalho a todos meus familiares e professores, em especial ao meu Orientador Everaldo Artur Grahl, que colaboraram para a realização deste trabalho, orientando e incentivando-me. E aos meus amigos, pelo apoio e motivação.

Que teu trabalho seja perfeito para que mesmo
depois da tua morte permaneça.

Leonardo da Vinci

RESUMO

Este trabalho apresenta uma ferramenta de apoio à gerência de configuração de software. A ferramenta criada foi baseada no estudo de outras ferramentas similares pesquisadas e nas diretrizes previstas no modelo MPS.BR no processo de gerência de configuração. As principais atividades suportadas pela ferramenta são controles de modificações e versões. A ferramenta criada se mostrou bem aderente aos resultados previstos no modelo MPS.BR.

Palavras-chave: Gerência de configuração, MPS.BR.

ABSTRACT

This work presents a tool of support to the management of software configuration. The developed tool was based on the study of other similar searched tools and on the lines of direction foreseen in model MPS.BR in the process of management of configuration. The main activities supported for the tool are controls of modifications and versions. The developed tool showed well adherent to the results foreseen in model MPS.BR.

Word-key: Management of configuration, MPS.BR.

LISTA DE ILUSTRAÇÕES

Figura 1 – Gestão de conteúdos na gestão de configuração de software	18
Figura 2 – Repositório de controle de versões	19
Figura 3 – Ramificações de um projeto.....	20
Figura 4 – Política “trava-modifica-destrava”	21
Figura 5 – Política “copia-modifica-resolve” quando arquivos são copiados.....	22
Figura 6 – Política “copia-modifica-resolve” quando arquivo é gravado	22
Figura 7 – Política “copia-modifica-resolve” quando é mesclada uma nova versão	23
Figura 8 – Componentes do MPS.BR	26
Figura 9 – Repositório do Subversion	29
Figura 10 – Cadastro de pedidos de mudança (<i>tickets</i>)	30
Figura 11 – Consulta de evolução dos projetos.....	31
Figura 12 – Caso de uso de Configuração.....	35
Figura 13 – Caso de uso de Execução	36
Figura 14 – Diagrama de atividades de um pedido de modificação.....	38
Figura 15 – Diagrama de classes da aplicação	39
Quadro 1 – Código fonte de uma classe de negócio.....	42
Quadro 2 – XML de mapeamento de uma classe de negócio	43
Quadro 3 – Código fonte de uma classe de persistência	43
Quadro 4 – Arquivo de versões completo	44
Figura 16 – Tela de cadastro de usuários	46
Figura 17 – Tela de cadastro de projetos.....	46
Figura 18 – Tela de linhas básicas e geração de pacotes.....	47
Figura 19 – Tela de cadastro de versão de configuração (linha básica)	47
Figura 20 – Tela de cadastro de tipo de itens de configuração.....	48
Figura 21 – Tela para importar itens de configuração.....	48
Figura 22 – Itens de configuração dentro do diretório do repositório.....	49
Figura 23 – Tela de ramificação de linhas básicas	50
Figura 24 – Tela de identificação de responsáveis	50
Figura 25 – Tela de pedidos de modificação.....	51
Figura 26 – Tela de registro de pedido de modificação	52
Figura 27 – Tela de registro de avaliação do pedido.....	53

Figura 28 – Tela de registro de decisão	54
Figura 29 – Tela de liberação de arquivos para auditoria.....	55
Figura 30 – Tela de registro de auditoria.....	55
Figura 31 – Tela de diferença de versões	56
Figura 32 – Relatório de acompanhamento das modificações	57
Figura 33 – Relatório de linhas básicas	57
Figura 34 – Relatório de auditorias	58
Figura 35 – Relatório de histórico dos itens de configuração	58
Quadro 5 – Análise das ferramentas de gerência de configuração no quesito controle de versões	60
Quadro 6 – Análise das ferramentas de gerência de configuração no quesito controle de mudanças	61
Quadro 7 – Quadro com os resultados esperados pelo modelo MPS.BR	62

LISTA DE SIGLAS

API – *Application Programming Interface*

APR – *Apache Portable Runtime*

CASE – *Computer Aided Software Engineering*

CMMI – *Capability Maturity Model Integration*

ICS – *Itens de Configuração de Software*

IEC – *International Engineering Consortium*

ISO – *International Standardization Organization*

MA-MPS – *Método de Avaliação para melhoria de Processo de Software*

MN-MPS – *Modelo de Negócio para Melhoria de Processo de Software*

MPS.BR – *Melhoria de Processo do Software Brasileiro*

MR-MPS – *Modelo de Referência para Melhoria de Processo de Software*

SQL – *Structured Query Language*

TCP/IP – *Transmission Control Protocol/Internet Protocol*

UML – *Unified Modeling Language*

XML – *eXtensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO.....	13
1.1 OBJETIVOS DO TRABALHO	14
1.2 ESTRUTURA DO TRABALHO	14
2 FUNDAMENTAÇÃO TEÓRICA	16
2.1 GERÊNCIA DE CONFIGURAÇÃO DE SOFTWARE.....	16
2.1.1 Na tarefa de Identificação	16
2.1.2 Controle de versão.....	18
2.1.3 Controle de modificação	23
2.1.4 Auditoria de configuração.....	24
2.1.5 Preparação de relatórios	24
2.1.6 Integração contínua	25
2.2 MELHORIA DE PROCESSO DO SOFTWARE BRASILEIRO (MPS.BR).....	25
2.2.1 Níveis de maturidade.....	26
2.2.2 Processo	27
2.2.3 Gerência de configuração de software no MPS.BR.....	27
2.3 FERRAMENTAS DE APOIO A GERÊNCIA DE CONFIGURAÇÃO	28
2.4 TRABALHOS CORRELATOS	31
3 DESENVOLVIMENTO DO TRABALHO	33
3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO.....	33
3.2 ESPECIFICAÇÃO	34
3.2.1 Diagrama de casos de uso	34
3.2.2 Diagrama de atividades	37
3.2.3 Diagrama de classes	39
3.3 IMPLEMENTAÇÃO	41
3.3.1 Técnicas e ferramentas utilizadas.....	41
3.3.2 Operacionalidade da implementação	45
3.4 RESULTADOS E DISCUSSÃO	59
3.4.1 Comparativo das ferramentas.....	59
3.4.2 Análise de aderência ao modelo MPS.BR	61
4 CONCLUSÕES	63
4.1 EXTENSÕES	64

REFERÊNCIAS BIBLIOGRÁFICAS	65
---	-----------

1 INTRODUÇÃO

Com o aumento da complexidade do desenvolvimento de software, crescem também os problemas no gerenciamento de alterações durante o processo de criação ou manutenção do mesmo. O problema agrava-se ainda mais quando as equipes não estão bem integradas ou não se usa um ambiente que permita um controle adequado das mudanças realizadas no software. Além disso, se exige um padrão para que as equipes consigam trabalhar corretamente em conjunto, mantendo a qualidade e os prazos estipulados.

De acordo com Sommerville (2003, p. 550), é necessário gerenciar os sistemas em desenvolvimento porque, à medida que eles são alterados, são criadas muitas versões diferentes de software. Para auxiliar as pessoas envolvidas em um determinado projeto a controlar bem seus processos é necessária uma ferramenta que auxilie na análise, construção, modificações, testes, entre outros quesitos. Muitas empresas ainda não se preocupam com isso, o que acaba tornando de baixa qualidade o serviço realizado. Uma ferramenta para ser útil a uma empresa deve atender um conjunto de regras e normas existentes na engenharia de software. Na engenharia de software existe a gerência de configuração, na qual se definem etapas para os trabalhos em equipe e controle dos artefatos no ciclo de vida do produto. Segundo Sommerville (2003, p. 550), o gerenciamento de configuração é o desenvolvimento e a aplicação de padrões e procedimentos para gerenciar um produto de sistema em evolução.

No mercado existem várias ferramentas que atendem a gerência de configuração, porém elevados custos de licença fazem com que pequenas e médias empresas não as utilizem de forma efetiva. Nos últimos anos surgiram algumas ferramentas *open-source* que dão apoio à gerência de configuração por um custo mais baixo, mesmo assim, é necessário um conjunto de ferramentas para atender os principais aspectos relacionados à gerência de configuração como o controle de versões de software, controle de mudanças e a integração contínua.

A gerência de configuração é abordada em vários modelos da qualidade de software tais como o CMMI (*Capability Maturity Model Integration*). Para esse trabalho foi escolhido o modelo de referência MPS.BR (Melhoria de Processo do Software Brasileiro), pois atende a necessidade de implantar os princípios de engenharia de software de forma adequada ao contexto das empresas brasileiras, estando alinhado com as principais abordagens internacionais para definição, avaliação e melhoria de processos de software. O modelo MPS.BR é baseado em sete níveis de maturidade que vão desde o nível mais alto (letra A), até o mais baixo (letra G). A gerência de configuração encontra-se no nível F. Este modelo é

muito recente e advém de estudos feitos no Brasil, onde predominam pequenas e médias empresas de software (ASSOCIAÇÃO PARA PROMOÇÃO DA EXCELÊNCIA DO SOFTWARE BRASILEIRO, 2006, p. 5). As orientações apresentadas neste modelo são mais adequadas para a maioria das organizações existentes no Brasil. Vale salientar que para as empresas que procuram melhorar seus processos de software ou ainda pretendem obter certificações internacionais é necessário ter implantado um processo consistente de gerência de configuração.

1.1 OBJETIVOS DO TRABALHO

O objetivo principal deste trabalho é desenvolver uma ferramenta que suporte o processo de gerência de configuração de software.

Os objetivos específicos são:

- a) definir o processo de gestão de configuração a ser suportado pela ferramenta;
- b) comparar a ferramenta criada com outras ferramentas *open-source* existentes no mercado;
- c) acrescentar novas funcionalidades nesta ferramenta não implementadas em trabalhos correlatos já desenvolvidos na FURB;
- d) incorporar funcionalidades na ferramenta para atender diretrizes previstas no processo de gerência de configuração do modelo MPS.BR.

1.2 ESTRUTURA DO TRABALHO

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

No segundo capítulo é apresentada a fundamentação teórica do trabalho, através dos conceitos que envolvem o tema proposto, dando ênfase aos conceitos gerência de configuração.

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

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

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo aborda os assuntos referentes à revisão bibliográfica do trabalho, incluindo a gerência de configuração de software e o modelo MPS.BR.

2.1 GERÊNCIA DE CONFIGURAÇÃO DE SOFTWARE

Segundo Dias (2006), a gerência de configuração é um conjunto de atividades de apoio ao desenvolvimento que permite que as mudanças inerentes ao mesmo sejam absorvidas pelo projeto de maneira controlada, mantendo a estabilidade na evolução do software. Conforme Paula Filho (2001, p. 513), mesmo um projeto pequeno gera diversos artefatos. De acordo com isso, é necessário ter um controle sistematizado sobre os artefatos de um sistema, para que as alterações sejam realizadas através de autorizações, permitindo gerar um histórico de todo o processo e impedindo que programadores desavisados eliminem funcionalidades criadas por outros nas atualizações corriqueiras do dia-a-dia. A gerência de configuração tem como objetivo identificar, controlar e garantir que a mudança seja implementada corretamente, e relatar a todos os envolvidos.

De acordo com Figueiredo, Santos e Rocha (2005, p. 3) o processo de gerência de configuração de software tem como objetivos principais:

- a) identificar todos os itens da configuração de software;
- b) gerir modificações em um ou mais itens;
- c) facilitar a construção de diferentes versões de uma aplicação;
- d) garantir que a qualidade do software seja mantida ao longo do seu ciclo de vida.

Para atender esses objetivos, a gerência de configuração dispõe de cinco tarefas, descritas a seguir: identificação, controle de versão, controle de modificação, auditoria de configuração e preparação de relatórios.

2.1.1 Tarefa de Identificação

Cada objeto tem um conjunto de características distintas que o identificam unicamente:

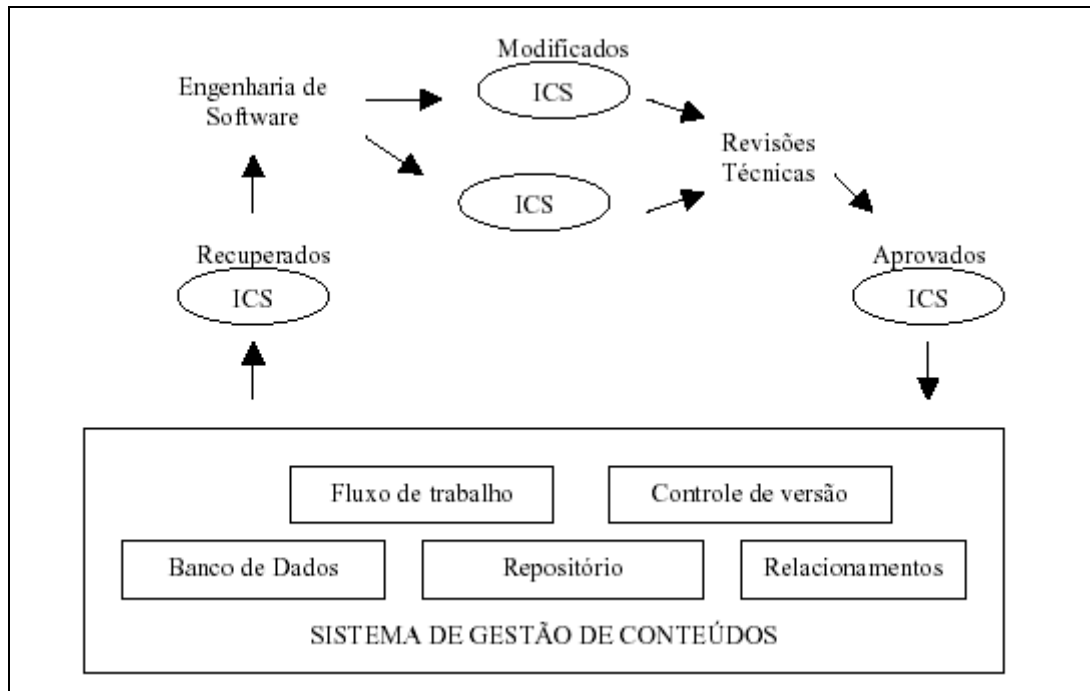
um nome, uma descrição, uma lista de recursos e uma “realização”. O nome do objeto é uma cadeia de caracteres que identifica o objeto de modo não-ambíguo. A descrição do objeto é uma lista de itens de dados que identifica: o tipo do item representado pelo objeto, um identificador de projeto, informação de modificação e/ou versão (PRESSMAN, 2006, p. 607).

Item de configuração é todo tipo de artefato que está ligado ao software em si, como código fonte, documentos de análise, compiladores, executáveis, etc. A quantidade de itens aumenta conforme o projeto avança. Em cada etapa do projeto são criados novos artefatos, e segue dessa maneira em todo o ciclo de vida do software.

De acordo com Parreiras e Bax (2003, p. 11), o volume crescente de itens de configuração e suas relações demandam um ambiente com as seguintes características:

- a) repositório: lugar onde todos os itens de configuração serão armazenados para futura recuperação;
- b) banco de dados: local de armazenamento dos metadados dos documentos, assim como outras informações do processo como o histórico de mudanças dos itens e informações de auditoria;
- c) relacionamento entre os itens: é necessário um mecanismo que permita ao usuário explicar as relações entre os itens, no intuito de facilitar a recuperação e agrupar os itens semanticamente;
- d) fluxo de trabalho: vários itens de configuração de software tramitam entre diversos envolvidos com a sua criação ou manutenção. Faz-se necessário a utilização de mecanismos de fluxo de trabalho com o objetivo de enviar os itens certos para as pessoas certas no momento certo;
- e) controle de versões e *releases*: grande parte dos itens de configuração de software sofre alterações, de modo a gerar novas versões destes itens. Contudo, se faz necessária a manutenção de outras versões na base de conhecimento. *Release* é uma versão de um sistema distribuída a um cliente.

A figura 1 exhibe a gestão de conteúdos na gestão de configuração de software.



Fonte: Parreiras e Bax (2003).

Figura 1 – Gestão de conteúdos na gestão de configuração de software

2.1.2 Controle de versão

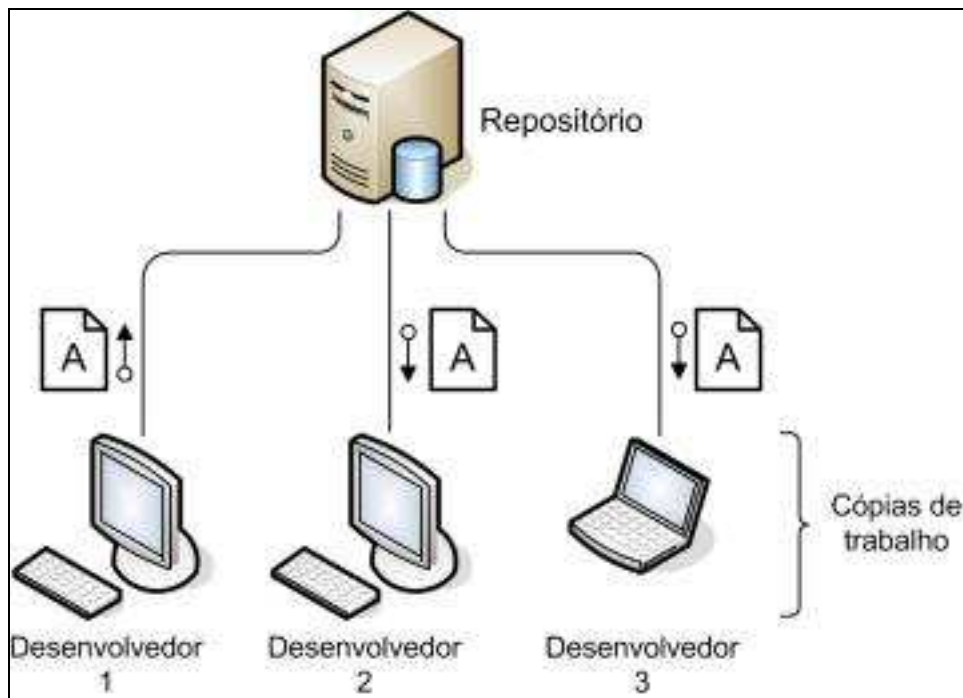
Segundo Sommerville (2003, p. 556), “o gerenciamento de versões e *releases* consiste no processo de identificar e acompanhar o desenvolvimento de diferentes versões e *releases* de um sistema”. Uma versão de sistema é uma instância de um projeto que difere, de muitas maneiras, de outras instâncias. Novas versões podem ter diferentes funcionalidades e desempenhos ou podem resolver defeitos no sistema. Uma *release* de sistema é uma versão que é distribuída para os clientes. Cada *release* deve incluir nova funcionalidade ou se destinar a uma diferente plataforma de hardware. De acordo com isso, a atividade de controle de versão tem como finalidade o armazenamento e gerenciamento das versões dos itens de configuração durante o ciclo de vida do software.

De acordo com Caetano (2004, p. 14), as principais funções pertinentes ao controle de versões podem ser resumidas da seguinte forma:

- a) recuperar versões anteriores;
- b) auditar as modificações realizadas: quem, quando, o quê;
- c) automatizar o rastreamento de arquivos;
- d) estabelecer meios para obter a situação de um projeto em determinado ponto do

- tempo;
- e) prevenir conflitos entre desenvolvedores;
- f) permitir o desenvolvimento paralelo.

O lugar onde os itens de configuração são armazenados é chamado de repositório. A figura 2 exhibe um exemplo de repositório.

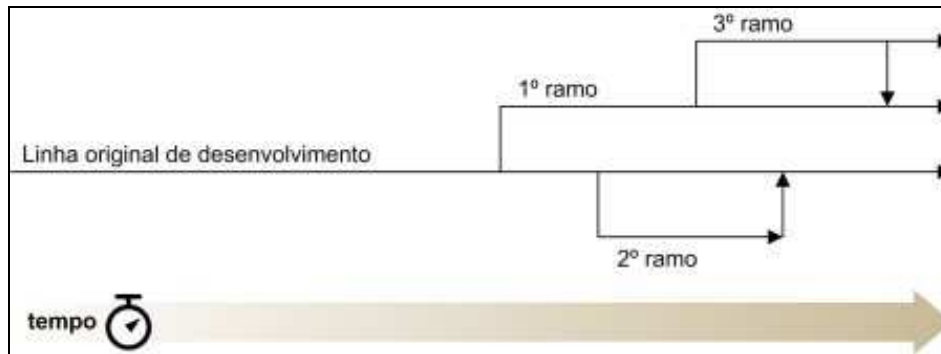


Fonte: Dias (2006).

Figura 2 – Repositório de controle de versões

O repositório tem como objetivo manter a integridade dos itens. Conforme vão surgindo alterações nos artefatos, o repositório vai guardando as múltiplas versões dos arquivos, ou seja, nenhuma versão será perdida ou sobreposta.

Existem algumas funcionalidades essenciais como é o caso de ramificação de versões de um projeto. Em algumas situações específicas, ou por necessidade de algum cliente, é necessário que uma versão antiga seja alterada. Como realizar essa tarefa foge dos padrões da disciplina, é então criada uma versão paralela para que se resolva o problema desejado. A figura 3 ilustra uma ramificação de um projeto.



Fonte: Dias (2006).

Figura 3 – Ramificações de um projeto

As mudanças no ciclo de vida de um software é algo muito comum. Pois, por melhor que tenham sido definidos os requisitos do sistema em sua construção, com o tempo vão surgindo novas necessidades, no qual o sistema terá que adaptar-se.

Para ajudar a controlar as mudanças, existem as linhas básicas (*baselines*), que segundo Pressman (2006, p. 602) é um conceito de gerenciamento de configuração de software que nos ajuda a controlar as mudanças, sem impedir seriamente as mudanças justificáveis. Para uma linha básica ser gerada é necessário que passe por todas as etapas da atividade de controle de mudanças. Para isso será necessária uma nova solicitação de mudança e a mesma deverá ser aprovada. A partir desse momento, um programador autorizado poderá buscar o objeto do repositório e realizar as alterações solicitadas e encaminhar para a auditoria. Sendo aprovado na auditoria, a linha básica será adicionada ao repositório de controle de versões.

Existem algumas funcionalidades essenciais como ramificações em arquivos. Seria o caso do *branch*, que é uma linha de desenvolvimento de um módulo, onde alterações em um *branch* podem ser incorporadas a outros *branches* a qualquer momento, dessa forma permite que vários usuários alterem de forma concorrente um mesmo projeto. Quando alguém confirma uma modificação, o *merge* grava as alterações realizadas no repositório, porém pode acontecer o caso do *merge* não realizar sua tarefa, isso acontece quando mais de um cliente tenta executar alterações na mesma linha do documento, o sistema apenas executa a primeira alteração, e informa o usuário da segunda alteração que será necessário uma intervenção humana (FARIAS, 2004, p. 8).

No desenvolvimento de projetos é comum que várias pessoas necessitem modificar os mesmos arquivos. Para resolver esse problema existem algumas políticas de bloqueio que impedem que arquivos sejam atualizados sobrescrevendo alterações de outros, tais como a “trava-modifica-destrava” e a “copia-modifica-resolve”. A política “trava-modifica-destrava”

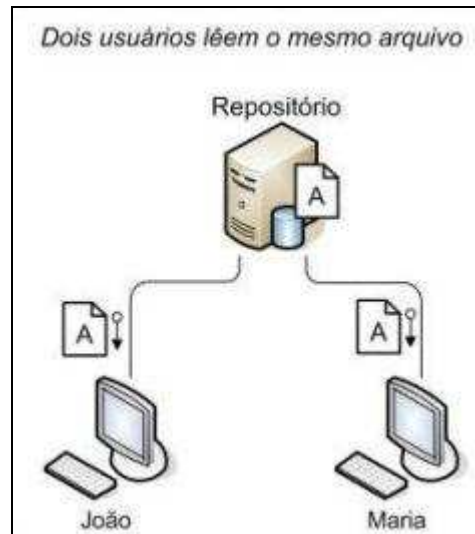
possui uma regra bem simples, pois permite que apenas uma pessoa por vez altere o arquivo, enquanto que a “cópia-modifica-resolve” não existe travamento de arquivo. Neste modelo os desenvolvedores fazem uma cópia local dos objetos e alteram livremente, sendo que no final as alterações são mescladas no repositório formando uma nova versão. Segundo Dias (2006) a política “trava-modifica-destrava” é restrita e frequentemente atrapalha o trabalho dos usuários. Enquanto que a outra solução apesar de parecer caótica, funciona bem na prática. Conflitos são raros e são causados basicamente pela falta de comunicação entre desenvolvedores. Na grande maioria dos casos, as alterações não se sobrepõem e são mescladas automaticamente pelo sistema de controle de versão. A figura 4 exhibe a política “trava modifica-destrava”.



Fonte: Dias (2006).

Figura 4 – Política “trava-modifica-destrava”

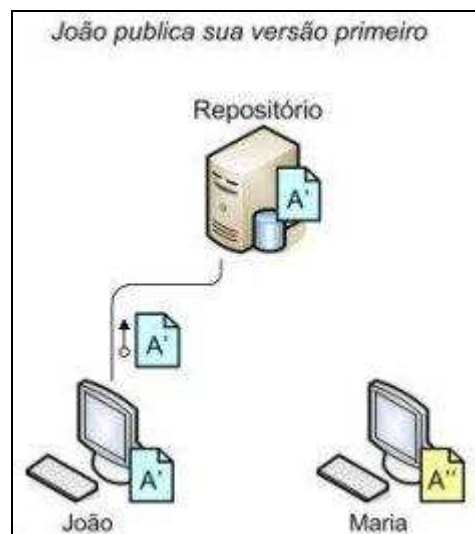
A figura 4 exhibe o comportamento do repositório no momento que um usuário faz a cópia de um arquivo para edição. Nesta situação, o usuário Maria só conseguirá buscar o objeto para alterar quando o João terminar suas alterações e liberá-lo para o repositório.



Fonte: Dias (2006).

Figura 5 – Política “cópia-modifica-resolve” quando arquivos são copiados

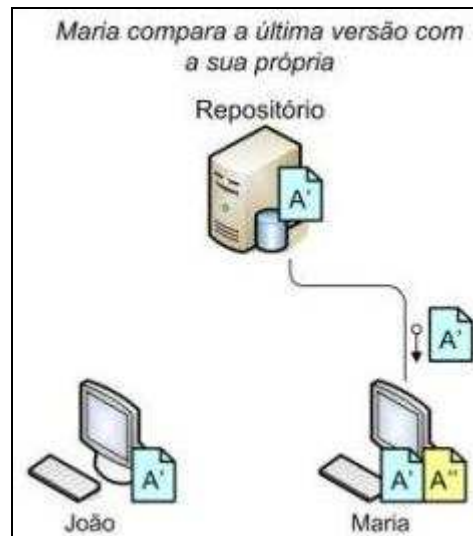
A figura 5 mostra dois usuários buscando o mesmo arquivo para realizar modificações. O repositório não bloqueia o arquivo, permitindo assim que diversos usuários busquem esse arquivo. Após buscar o arquivo, os usuários podem começar a editá-lo.



Fonte: Dias (2006).

Figura 6 – Política “cópia-modifica-resolve” quando arquivo é gravado

A figura 6 exibe o usuário João adicionando no repositório uma nova versão do arquivo, nesse momento o arquivo que Maria tem estará desatualizado em relação à versão no repositório. Para não ocorrer problemas de sobrepor alterações do usuário João, Maria deverá sincronizar seu arquivo com o que está no repositório. A figura 7 exibe o usuário Maria mesclando sua versão com a versão do repositório, gerando assim uma nova versão do arquivo.



Fonte: Dias (2006).

Figura 7 – Política “copia-modifica-resolve” quando é mesclada uma nova versão

2.1.3 Controle de modificação

O foco dessa atividade é de permitir que as mudanças dos itens de configuração sejam propostas, avaliadas, aceitas e aplicadas. Segundo Sommerville (2003, p. 554), os procedimentos de gerenciamento de mudanças devem ser concebidos para assegurar que os custos e os benefícios das mudanças sejam adequadamente analisados e as mudanças em um sistema sejam feitas de maneira controlada.

Uma nova modificação deve ser iniciada com uma proposta de modificação. As propostas deverão contar apenas com as alterações ou correções desejadas, sem detalhes técnicos. A partir disso, uma equipe de responsáveis técnicos, avalia a viabilidade de realizarem essas propostas. As propostas aprovadas serão transformadas em solicitações de mudanças, que são o mesmo que uma ordem de serviço para realização da tarefa. Uma solicitação pode corresponder a uma ou a várias propostas. A partir de uma solicitação, responsáveis técnicos preparam uma descrição de projeto de modificação, que descreve como a solicitação será implementada, identificando os arquivos que serão alterados. Após isso, a descrição de projeto é encaminhada para uma aprovação, se a mesma for aprovada, será encaminhada para uma equipe que será responsável para buscar os objetos do repositório e realizar as modificações e testes necessários. Terminado essa etapa, a atividade é encaminhada para a auditoria para ser aprovada. Se for aprovado, os objetos serão submetidos

ao repositório novamente, assim, gerando uma nova versão e chegando ao fim do processo de controle de modificação.

2.1.4 Auditoria de configuração

Esta atividade visa assegurar que as alterações tenham sido implementadas corretamente. Essas auditorias são conduzidas de acordo com processos bem definidos que se constituem de vários papéis e responsabilidades de auditores. Logo, cada auditoria deve ser planejada cuidadosamente. Uma auditoria pode necessitar de certo número de indivíduos para realizar uma variedade de tarefas durante um período razoavelmente curto (FIGUEIREDO, SANTOS, ROCHA, 2005, p. 5).

Dois tipos de avaliação podem ser feitos: funcional e física. A avaliação funcional investiga os aspectos lógicos dos arquivos. Por outro lado, a avaliação física consiste em verificar se a configuração a ser congelada está composta da versão mais recente dos itens de configuração para a fase específica do ciclo de vida e se os procedimentos e padrões foram realizados corretamente. A avaliação física é executada no fim de cada fase do ciclo de vida do software (FIGUEIREDO, SANTOS, ROCHA, 2005, p. 5).

2.1.5 Preparação de relatórios

Cada vez que um item de configuração recebe uma identificação nova ou atualizada, uma entrada no relatório de estado da configuração é feita. Cada vez que uma auditoria de configuração é conduzida, os resultados são relatados como parte da tarefa de relatório de estado da configuração. A saída pode ser colocada em um banco de dados *on-line*, de modo que os desenvolvedores ou mantenedores de software possam ter acesso à informação de modificação por categoria de palavra-chave. Além disso, um relatório é gerado regularmente e se destina a manter a gerência e os profissionais informados de modificações importantes (PRESSMAN, 2006 p.612).

2.1.6 Integração contínua

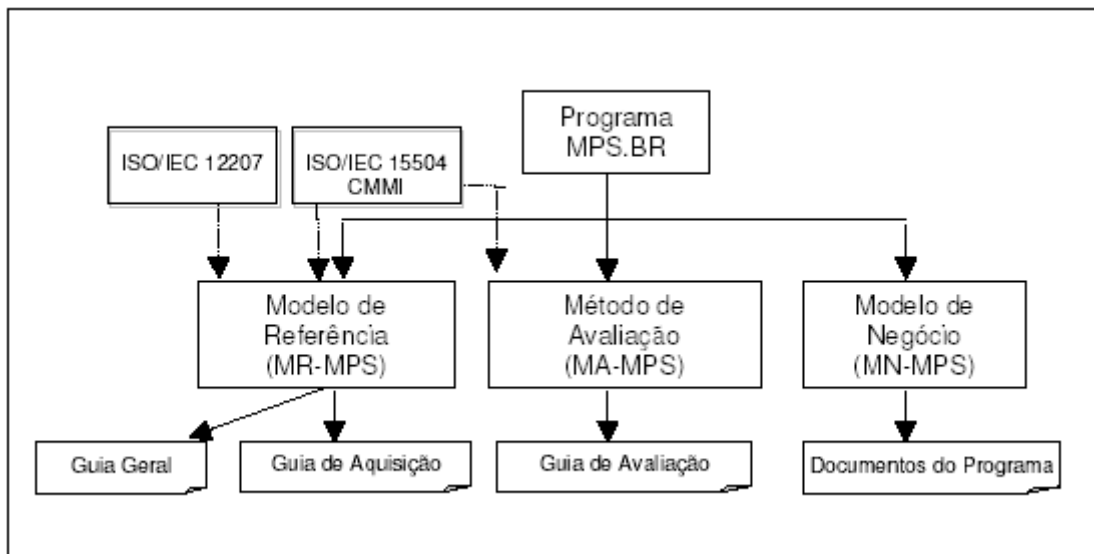
Para as necessidades da gerência de configuração, bastaria um controle de construção de software que cuidassem da identificação, empacotamento e preparação de uma linha básica (*baseline*) para a entrega a um cliente externo ou interno, tornando-a um *release* ou uma *build* respectivamente. A idéia de utilizar uma integração contínua, entretanto, vai um pouco mais além. O objetivo é garantir que as mudanças no projeto são construídas, testadas e relatadas tão logo quanto possível depois de serem introduzidas. Em projetos de software, a construção do software é feita pela recuperação da configuração correta do sistema de controle de versão e a construção dos arquivos executáveis e de instalação do produto. Este processo é executado geralmente após cada mudança publicada no sistema de controle de versão ou em intervalos de tempo pré-definidos (DIAS, 2006).

2.2 MELHORIA DE PROCESSO DO SOFTWARE BRASILEIRO (MPS.BR)

Existem vários modelos e normas da qualidade que abordam o processo de gerência de configuração entre os quais inclui-se o MPS.BR. Este modelo visa definir e aprimorar um modelo de melhoria e avaliação de processo de software, visando preferencialmente as micro, pequenas e médias empresas, de forma a atender as suas necessidades de negócio e ser reconhecido nacional e internacionalmente como um modelo aplicável à indústria de software. O MPS.BR também estabelece um processo e um método de avaliação, o que dá sustentação e garante que o MPS.BR está sendo empregado de forma coerente com as suas definições (ASSOCIAÇÃO PARA PROMOÇÃO DA EXCELÊNCIA DO SOFTWARE BRASILEIRO, 2006, p. 5).

A base técnica para a construção e aprimoramento deste modelo é composta pelas normas NBR ISO/IEC 12207 – Processo de Ciclo de Vida de Software, pelas emendas 1 e 2 da norma Internacional ISO/IEC 12207 e pela ISO/IEC 15504 – Avaliação de Processo. Este modelo também cobre o conteúdo do CMMI/SW, através da inclusão de processos e resultados esperados além dos estabelecidos na Norma ISO/IEC 12207. O MPS.BR está dividido em três componentes: Modelo de Referência (MR-MPS), Método de Avaliação (MA-MPS) e Modelo de Negócio (MN-MPS) (ASSOCIAÇÃO PARA PROMOÇÃO DA EXCELÊNCIA DO SOFTWARE BRASILEIRO, 2006, p. 11). Cada componente é descrito

por meio de guias e/ou documentos do MPS.BR. A figura 8 exibe os componentes do MPS.BR.



Fonte: Associação para promoção da excelência do software brasileiro (2006).

Figura 8 – Componentes do MPS.BR

O modelo de referência MR-MPS contém as definições dos níveis de maturidade, processos e atributos do processo.

2.2.1 Níveis de maturidade

Os níveis de maturidade estabelecem patamares de evolução de processos, caracterizando estágios de melhoria da implementação de processos na organização. O MR-MPS define sete níveis de maturidade. Cada nível tem um perfil de processo, que indicam onde a organização deve colocar o esforço de melhoria (ASSOCIAÇÃO PARA PROMOÇÃO DA EXCELÊNCIA DO SOFTWARE BRASILEIRO, 2006, p. 12). Os níveis são:

- a) nível A – em otimização: tem como foco principal implantar melhorias incrementais e inovadoras de forma a melhorar os processos e as tecnologias da organização;
- b) nível B – gerenciado quantitativamente: tem como o objetivo gerenciar quantitativamente o processo definido para o projeto de forma a alcançar os objetivos de qualidade e de desempenho do processo estabelecidos para o projeto;
- c) nível C – definido: neste nível, o propósito principal é diminuir os riscos nos níveis

organizacionais e de projeto;

- d) nível D – largamente definido: através de um conjunto de processos é garantido que o produto desenvolvido atende as necessidades do cliente;
- e) nível E – parcialmente definido: tem como foco garantir que todos os profissionais tenham o conhecimento necessário para realizar suas tarefas;
- f) nível F – gerenciado: tem como propósito principal manter a integridade de todos os produtos e disponibilizá-los a todos os envolvidos;
- g) nível G – parcialmente gerenciado: tem como objetivo controlar as atividades, tarefas e recursos, possibilitando a construção de um produto, assim como gerenciar os requisitos do mesmo.

Os níveis de maturidade vão do G até o A, sendo que para atender cada nível superior é necessário atender todos os níveis inferiores.

2.2.2 Processo

Os processos são descritos em termos de propósitos e resultados. O propósito descreve o objetivo geral a ser atingido durante a execução do processo. Os resultados esperados do processo estabelecem os resultados a serem obtidos com a efetiva implementação do processo. Estes resultados podem ser evidenciados por um artefato produzido ou uma herança significativa de estado ao se executar o processo (ASSOCIAÇÃO PARA PROMOÇÃO DA EXCELÊNCIA DO SOFTWARE BRASILEIRO, 2006, p. 15).

2.2.3 Gerência de configuração de software no MPS.BR

A gerência de configuração de software no MPS.BR se encontra no nível F (gerenciado). Os resultados esperados são:

- a) os itens de configuração são identificados;
- b) os itens de configuração gerados pelo projeto são definidos e colocados sob uma linha básica;
- c) é estabelecido e mantido um sistema de gerência de configuração;
- d) as modificações e liberações dos itens de configuração são controladas;

- e) as modificações e liberações são disponibilizadas para todos os envolvidos;
- f) a situação dos itens de configuração e as solicitações de mudanças são registradas, relatadas e o seu impacto é analisado;
- g) a completeza e a consistência dos itens de configuração são asseguradas;
- h) o armazenamento, o manuseio e a entrega dos produtos de trabalho são controlados;
- i) a integridade das linhas básicas (*baselines*) é estabelecida e mantida, através de auditoria da configuração e de registros da gerência de configuração.

2.3 FERRAMENTAS DE APOIO A GERÊNCIA DE CONFIGURAÇÃO

Foram pesquisadas duas ferramentas de apoio à gerência de configuração para o desenvolvimento deste trabalho: Subversion e o Trac.

O Subversion (SUBVERSION, 2006) é uma ferramenta de controle de versão. Esta ferramenta não só controla a versão do conteúdo dos arquivos, mas também de diretórios, cópias, renomeações e meta-dados. Algumas das principais funcionalidades do Subversion estão listadas a seguir:

- a) controle de versões de arquivos texto e binário: utiliza um algoritmo para diferenciação binário que funciona de modo idêntico para arquivos textos e binários;
- b) ramificações de projetos: possui mecanismos para criação de ramos em projetos, permitindo desenvolvimentos paralelos;
- c) acesso ao repositório: pode ser acessado através de protocolos de rede ou disco local. Sua localização sempre é determinada através de uma URL.

O Subversion não possui interface gráfica, porém executa em qualquer sistema operacional que o servidor Apache reconhece, pois utiliza uma biblioteca de portabilidade chamada de APR (Apache Portable Runtime). A figura 9 exibe arquivos dentro de um repositório do Subversion.

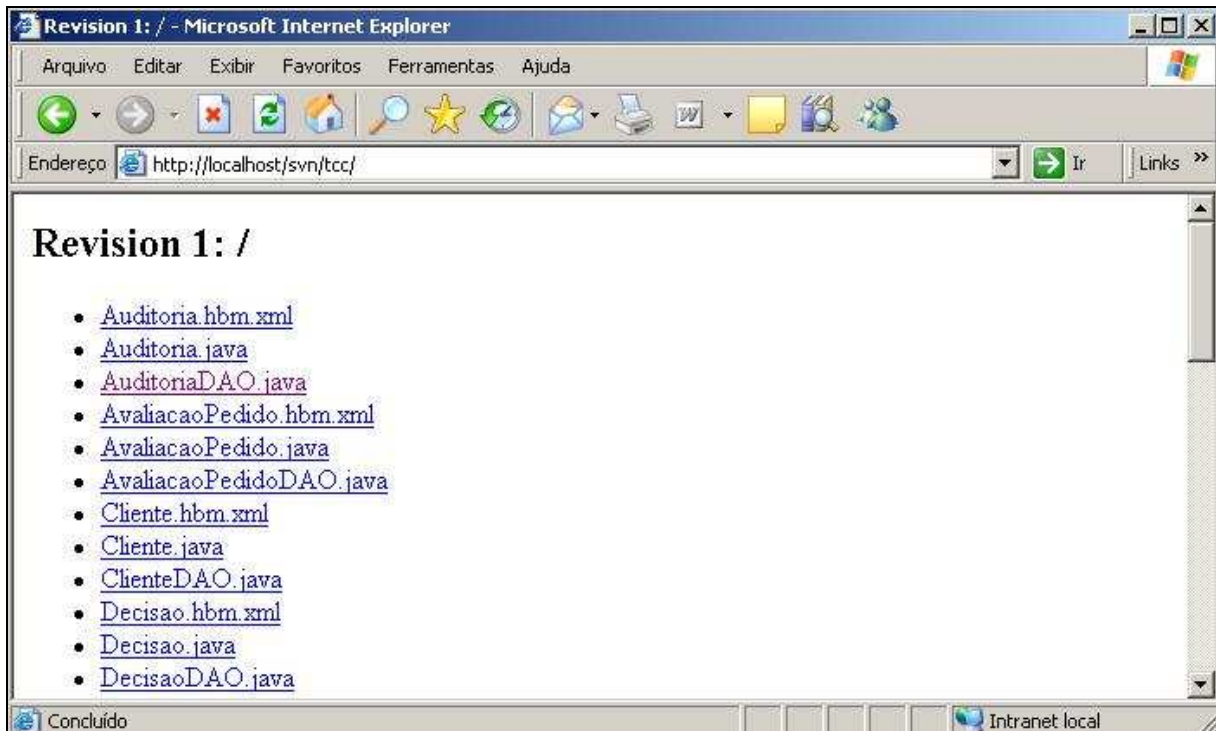


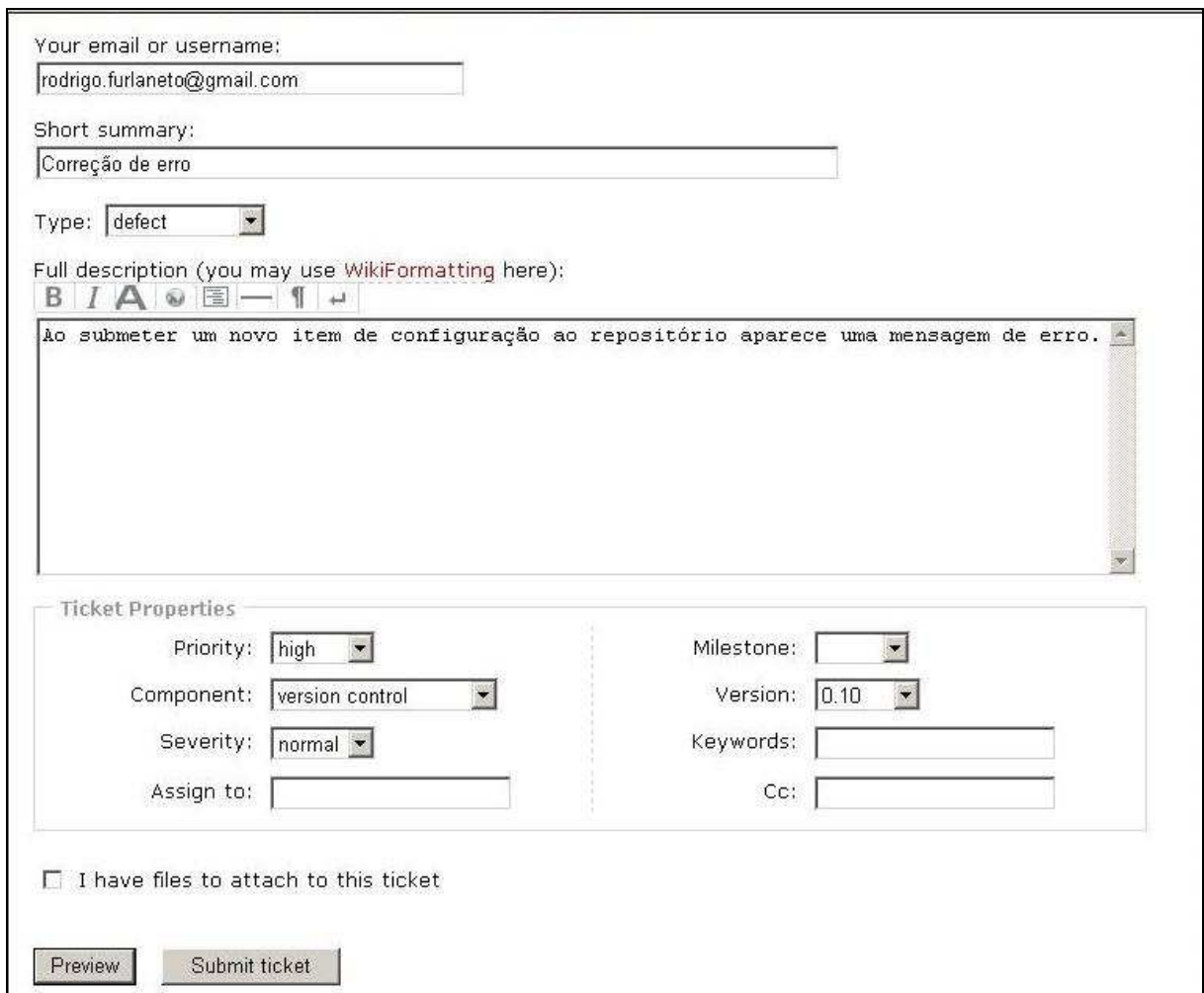
Figura 9 – Repositório do Subversion

O Trac (TRAC, 2006) é uma ferramenta para rastreamento de mudança em projetos de desenvolvimento de software e executa em um ambiente *web*. É desenvolvido e mantido pela empresa Edgewall software e por colaboradores da comunidade *open source*. Algumas de suas funcionalidades estão listadas a seguir.

- controle de modificações: permite registrar os pedidos de modificação, no Trac os pedidos são chamados de *tickets*;
- acompanhamento de evolução de projetos: permite acompanhar a evolução dos projetos através de análises dos pedidos;
- anexação de arquivos: permite anexar arquivos aos pedidos de mudanças. Isto é bastante útil, como em casos de correções de erro, onde se pode anexar à imagem de uma tela, por exemplo;
- notificações: envia mensagens por e-mails para todos os envolvidos;
- integração com o Subversion: permite acessar o repositório através de um *browse* permitindo a visualização dos conteúdos do projeto.

Essas duas ferramentas integradas atendem bem a gerência de configuração. Como o Trac é uma ferramenta *open source* e está em constante evolução, utiliza-se ele próprio para gerenciar o projeto. A figura 10 exibe o cadastro de pedidos de modificação do Track. O

usuário pode informar atributos importantes como a prioridade, componente, versão entre outros.



The image shows a web form for creating a ticket. It includes the following elements:

- Your email or username:** A text input field containing "rodrigo.furlaneto@gmail.com".
- Short summary:** A text input field containing "Correção de erro".
- Type:** A dropdown menu with "defect" selected.
- Full description (you may use WikiFormatting here):** A rich text editor with a toolbar (bold, italic, link, unlink, list, indent, outdent) and a text area containing "Ao submeter um novo item de configuração ao repositório aparece uma mensagem de erro."
- Ticket Properties:** A section with two columns of fields:
 - Priority: dropdown menu with "high" selected.
 - Component: dropdown menu with "version control" selected.
 - Severity: dropdown menu with "normal" selected.
 - Assign to: empty text input field.
 - Milestone: empty dropdown menu.
 - Version: dropdown menu with "0.10" selected.
 - Keywords: empty text input field.
 - Cc: empty text input field.
- I have files to attach to this ticket
- Preview** and **Submit ticket** buttons.

Figura 10 – Cadastro de pedidos de mudança (*tickets*)

A figura 11 exibe uma consulta que mostra uma relação dos pedidos resolvidos e os ainda ativos para cada marco de um projeto (*milestone*).

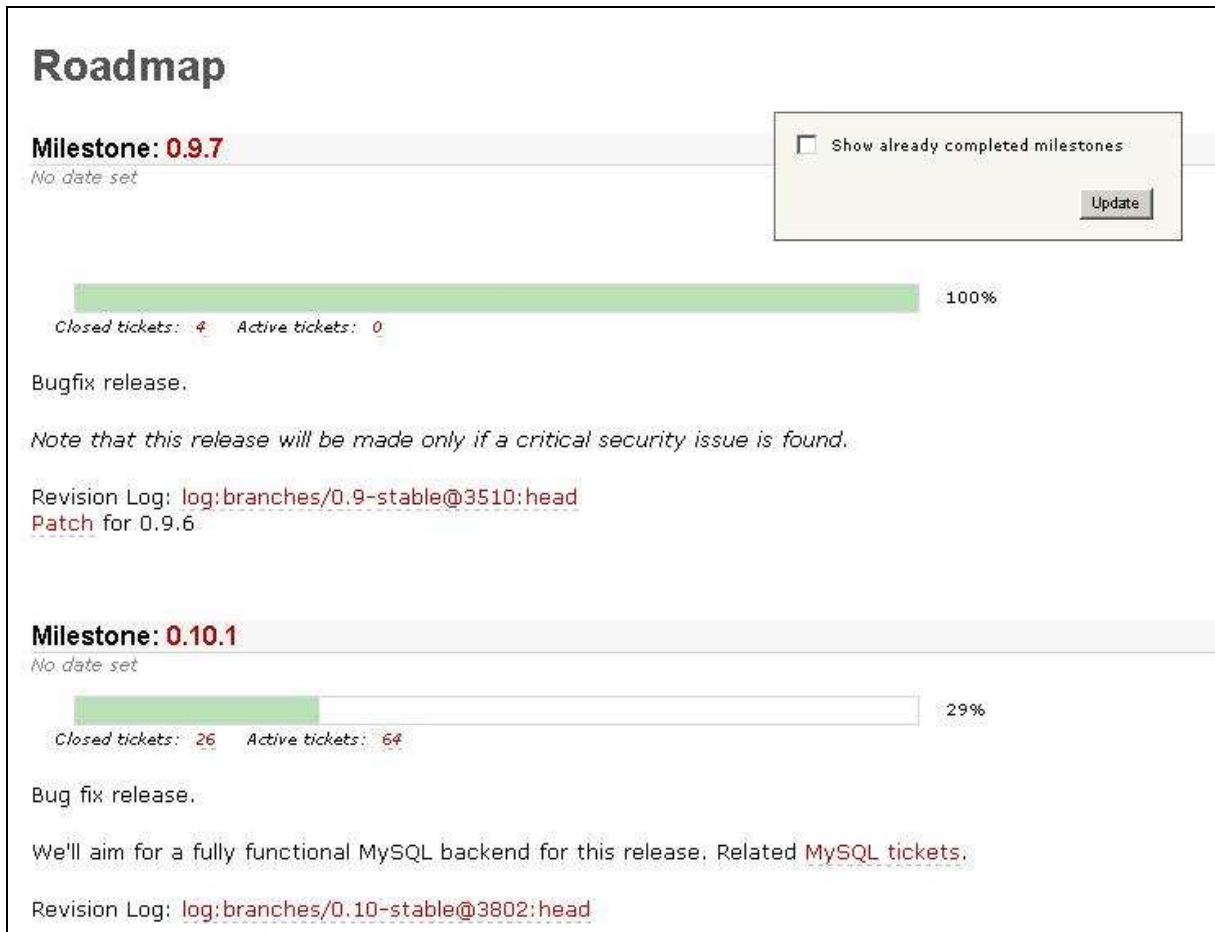


Figura 11 – Consulta de evolução dos projetos

2.4 TRABALHOS CORRELATOS

Em Barbaresco (2000) são apresentadas de uma forma bem detalhada as atividades relacionadas ao processo de gerência de configuração. O ponto forte deste trabalho é o estudo de modelos e normas da qualidade. Porém, a ferramenta, em que foi implementada em Delphi, não incorpora de maneira satisfatória todas as funcionalidades exigidas pelos modelos estudados, como por exemplo o controle de versões e a integração contínua, porém atende aos pedidos de modificações dos itens de configuração, juntamente com histórico e relatórios das alterações dos artefatos.

Em Bohn (2005) é apresentado o desenvolvimento de uma ferramenta utilizando a plataforma Delphi com o banco de dados MySQL que serve de apoio à gerência de configuração de software baseado no modelo CMMI. Nesta ferramenta é possível controlar as mudanças nos itens de configuração por processos bem definidos, no qual cada usuário tem

uma responsabilidade. Foi realizada também uma comparação da ferramenta desenvolvida com algumas ferramentas existentes no mercado. Foram comparados itens importantes como o controle de acesso, registro de rastreamento, gerenciamento de múltiplas versões, relato da situação da configuração, geração de versões e capacidade de arquivamento.

3 DESENVOLVIMENTO DO TRABALHO

Com base nos estudos realizados, foram definidos os requisitos do sistema proposto. Nas próximas seções são apresentados os requisitos da ferramenta e o desenvolvimento da mesma.

3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

Foi definido um processo de gestão de configuração com base nos resultados esperados pelo modelo MPS.BR. Este processo permite identificar responsáveis pelas atividades básicas que gerenciam um projeto, tais como abrir um pedido, analisá-lo, decidir se será realizado, realizá-lo e auditá-lo. Além disso, permitirá ter um histórico de todas as alterações realizadas. Os trabalhos correlatos e as ferramentas pesquisadas também contribuíram para a definição dos requisitos. A seguir são apresentados os requisitos funcionais e não funcionais da ferramenta:

- a) permitir o cadastramento de usuários, projetos, clientes e tipos de itens de configuração (Requisito Funcional - RF);
- b) registrar os itens de configuração (RF);
- c) buscar itens de configuração do repositório (RF);
- d) bloquear e desbloquear os itens de configuração (RF);
- e) permitir o rastreamento dos artefatos (RF);
- f) criar ramificação nos itens de configuração (RF);
- g) listar diferenças encontradas nos itens de configuração (RF);
- h) controlar as versões dos itens de configuração (RF);
- i) controlar as linhas básicas¹ (RF);
- j) registrar os pedidos de modificação (RF);
- k) registrar a avaliação, decisão e auditoria de um pedido de modificação (RF);
- l) registrar histórico de versões no sistema (RF);
- m) empacotar linhas básicas de projetos (RF);

¹ uma configuração formalmente aprovada para servir de referência para o desenvolvimento posterior do sistema.

- n) gerar relatórios de acompanhamento das modificações e liberações dos itens de configuração, informações sobre linhas básicas de um determinado projeto, auditorias realizadas e evolução dos itens de configuração (RF);
- o) ser implementado na linguagem Java, utilizando o ambiente Eclipse 3.2 (Requisito Não-Funcional - RNF);
- p) utilizar banco de dados MySQL 4.1 (RNF);
- q) utilizar o *framework* de persistência de objetos *Hibernate* (RNF);
- r) atender diretrizes previstas no modelo de referência MPS.BR (RNF).

3.2 ESPECIFICAÇÃO

A especificação da ferramenta é apresentada através de diagramas de casos de uso, diagrama de classes e diagrama de atividades. Os diagramas foram elaborados na ferramenta CASE Enterprise Architect 6.0.

3.2.1 Diagrama de casos de uso

Para os diagramas de casos de uso foram definidos cinco atores, que representam os responsáveis pelo processo de gerência de configuração de software:

- a) solicitante: responsável por abrir os pedidos de modificação;
- b) analista de sistemas: responsável para realizar a análise dos pedidos de modificação;
- c) gerente de projetos: responsável por permitir a realização de modificações e dos cadastros necessários;
- d) programador: responsável para realizar a implementação nos itens de configuração;
- e) auditor: responsável pela auditoria de uma implementação.

Nas figuras 12 e 13 são representados os diagramas de casos de uso de configuração e de execução.

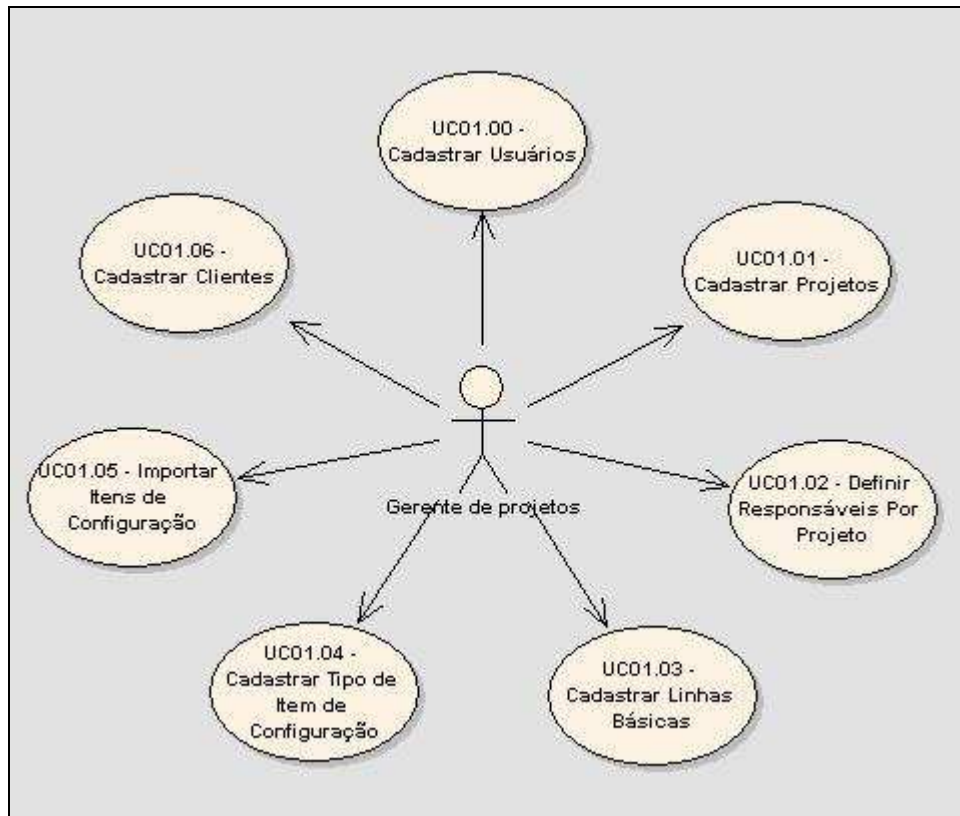


Figura 12 – Caso de uso de Configuração

A figura 12 envolve todos os pré-cadastros necessários para a execução da ferramenta. A seguir são descritos os casos de uso de configuração:

- a) cadastrar usuários: o gerente de projeto deve cadastrar todos os usuários da ferramenta. Deve ser informado o nome, login, senha, diretório de trabalho e tipo de usuário. Existem cinco tipos de usuários, que são: solicitante, analista, programador, auditor e gerente. Sendo que cada tipo tem acesso a determinadas funcionalidades do sistema. Os usuários podem ser incluídos, excluídos, alterados e consultados;
- b) cadastrar clientes: o gerente de projetos deve cadastrar os clientes dos projetos, informando o nome, tipo de pessoa, endereço, cidade, estado, telefone e fax. Os clientes podem ser incluídos, excluídos, alterados e consultados;
- c) cadastrar projetos: o gerente de projetos deve registrar os projetos que ele irá gerenciar, informando o nome e uma descrição sobre o projeto. Os projetos podem ser incluídos, excluídos, alterados e consultados;
- d) definir responsáveis por projeto: para cada projeto cadastrado o gerente de projetos terá que selecionar as pessoas que farão parte dos processos da gerência de

- configuração do projeto. Deve ser selecionados programadores, analistas e auditores;
- e) cadastrar linhas básicas: o gerente de projetos deve cadastrar as linhas básicas que compõem um projeto, informando o identificador da versão e uma descrição. As linhas básicas podem ser incluídas, excluídas, alteradas e consultadas;
 - f) cadastrar tipo de item de configuração: o gerente de projetos deve cadastrar os tipos de itens de configuração informando o nome, e o diretório que irá guardar os itens de configuração. Os tipo de item podem ser incluídos, excluídos, alterados e consultados;
 - g) importar itens de configuração: o gerente de projetos deverá importar todos os itens de configuração que fazem parte de uma linha básica, informando o tipo de item e uma descrição. Os itens de configuração podem ser incluídos, alterados, excluídos e consultados.

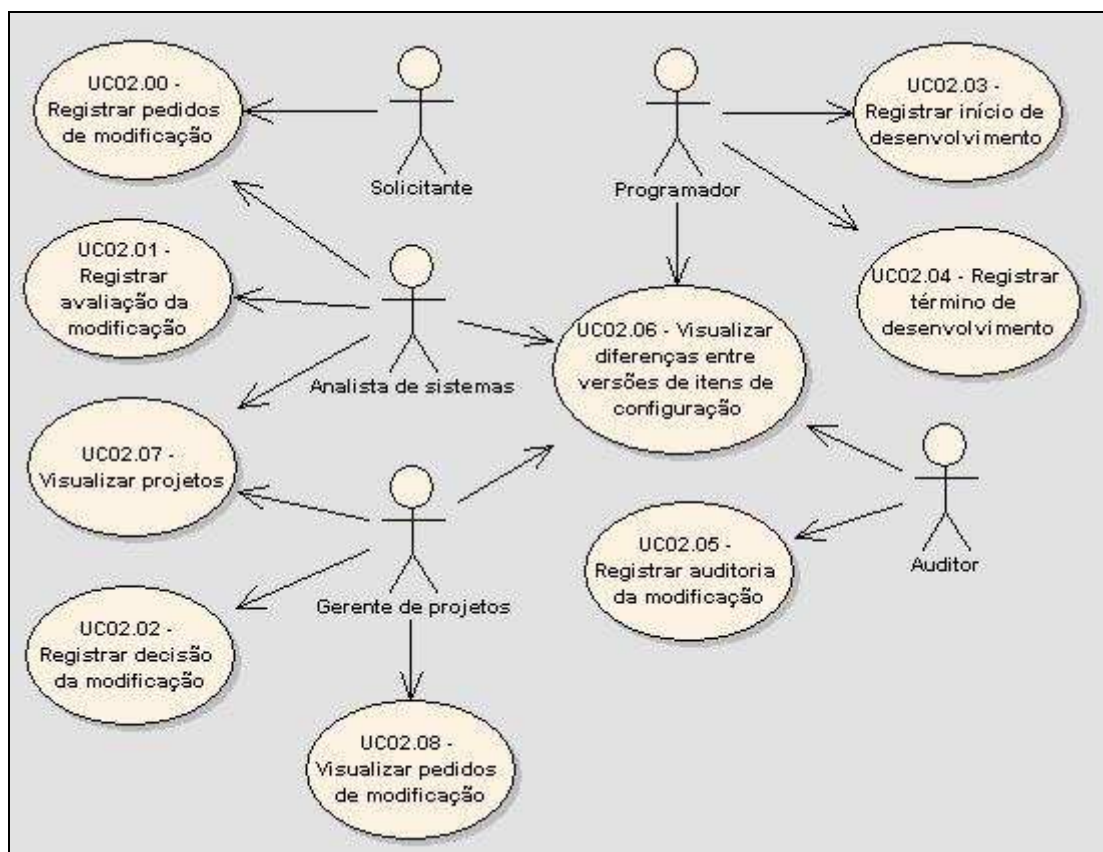


Figura 13 – Caso de uso de Execução

A figura 13 envolve todos os processos da gerência de configuração da ferramenta. A seguir são descritos os casos de uso de execução:

- a) registrar pedidos de modificação: o solicitante deve registrar o pedido de

modificação informando o cliente, o analista, o tipo e a descrição da modificação. Os pedidos podem ser incluídos, excluídos, alterados e consultados;

- b) registrar avaliação da modificação: o analista de sistemas deve registrar a avaliação da modificação informando uma descrição detalhada da modificação, os itens que serão alterados, qual linha básica que vai incorporar essa modificação, prioridade e tempo de esforço;
- c) registrar decisão da modificação: o gerente de projetos deve registrar a decisão, informando o parecer e uma descrição. Caso o parecer for de aprovado, o gerente deve informar o programador e o auditor responsáveis pelo pedido de modificação;
- d) registrar início de desenvolvimento: o programador deve registrar o início de desenvolvimento, neste momento os itens de configuração que devem ser alterados serão copiados do repositório, para a área de trabalho do programador e ficarão bloqueados para as outras modificações;
- e) registrar término de desenvolvimento: o programador responsável deve registrar o término de desenvolvimento, assim, os objetos alterados serão liberados para auditoria, para cada objeto deve ser informado uma observação do que foi alterado;
- f) registrar auditoria da modificação: o auditor responsável deve registrar a auditoria informando se foi aprovado ou não, e uma descrição detalhando a auditoria. Se a auditoria for aprovada os itens alterados serão submetidos ao repositório e liberados para novas modificações, caso contrário o pedido volta para a situação de desenvolvimento para que o programador possa fazer as devidas correções;
- g) visualizar projetos: o gerente de projetos ou analista de sistemas podem visualizar todos os dados de um projeto, verificando suas linhas básicas e o histórico de modificações;
- h) visualizar pedidos de modificação: o gerente de projetos pode verificar o andamento de todos os pedidos de modificação de um determinado projeto;
- i) visualizar diferenças entre versões de itens de configuração: os usuários podem verificar diferenças entre as versões dos arquivos, mas apenas para arquivos textos.

3.2.2 Diagrama de atividades

Na figura 14 é representado o diagrama de atividades, que detalha o fluxo de

atividades de um pedido de modificação.

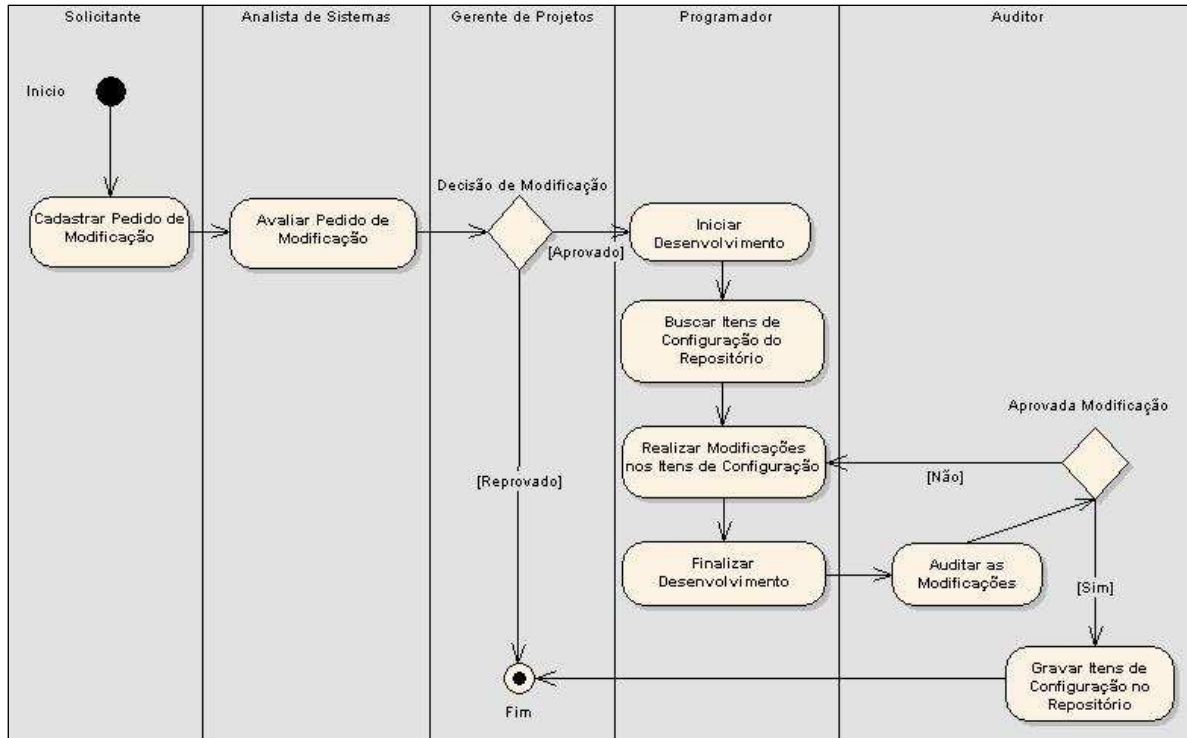


Figura 14 – Diagrama de atividades de um pedido de modificação

Um solicitante registra o pedido de modificação, descrevendo qual a necessidade desejada. O mesmo deverá selecionar o cliente interessado, e o analista que irá realizar a avaliação desse pedido. O analista de sistemas responsável avaliará os pontos de impacto no sistema, identificando os itens que deverão ser alterados, e em qual linha básica do projeto deverá ser liberado. Depois de realizada a avaliação, o gerente de projetos decidirá se tal modificação deve ser realizada. Se for aprovado, o programador identificado como responsável pela modificação iniciará o desenvolvimento. Neste momento os itens de configuração identificados pelo analista serão copiados para um diretório de trabalho do programador.

Após terminar todas as modificações necessárias, o programador pode liberar o pedido para auditoria finalizando seu desenvolvimento. Neste momento o sistema enviará os arquivos para um diretório no servidor, que ficará aguardando pela auditoria. O auditor responsável irá baixar os arquivos para seu diretório de trabalho, e realizará os testes e verificações necessárias nos itens de configuração. Em seguida, o auditor registrará o resultado da auditoria informando se foi aprovado ou reprovado com uma breve descrição. Se reprovado, o pedido retorna para o programador, para que o mesmo corrija caso contrário, os arquivos são gravados no repositório gerando uma nova versão para cada item alterado.

3.2.3 Diagrama de classes

Na figura 15 é apresentado o diagrama de classes de negócio da ferramenta desenvolvida.

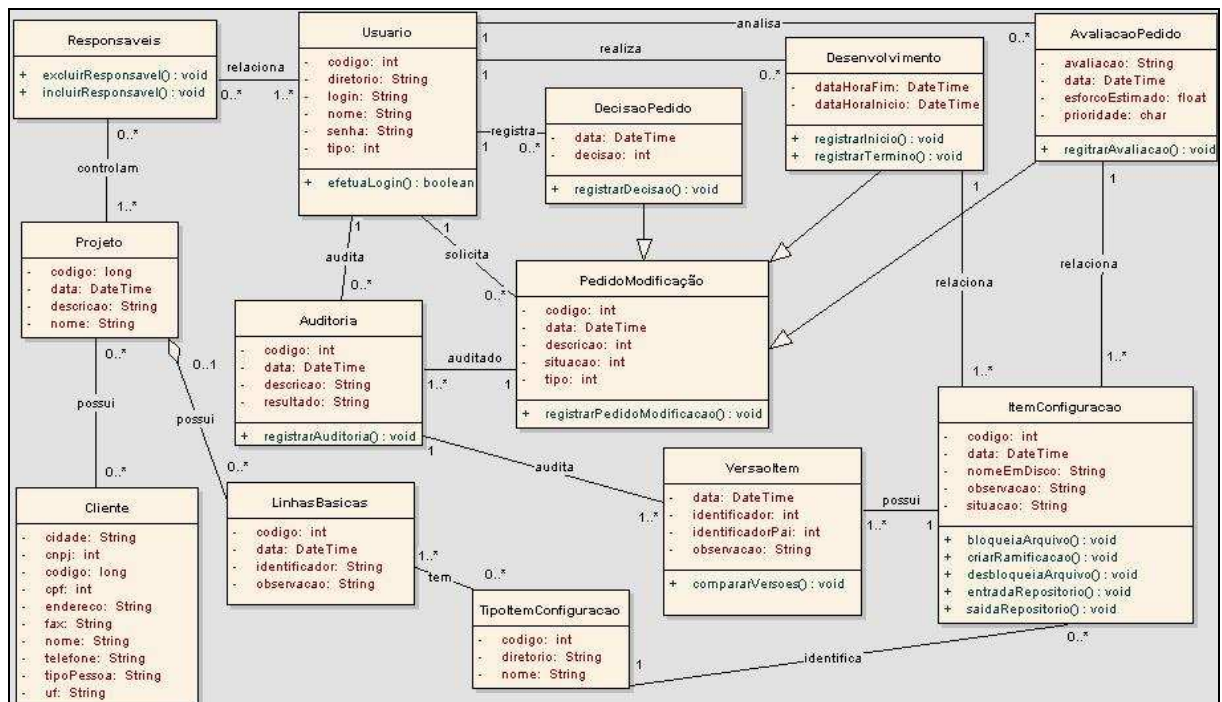


Figura 15 – Diagrama de classes da aplicação

Os principais atributos contidos neste diagrama são:

- código e descrição: estes atributos estão presentes na maioria das classes e servem para identificar e detalhar informações genéricas da entidade que está sendo manipulada;
- nome, senha, login, tipo entre outros: atributos da classe “Usuario” que descrevem informações relevantes dos usuários do sistema;
- cnpj, cpf, endereço, telefone entre outros: atributos da classe “Cliente” que descrevem informações relevantes dos clientes cadastrados no sistema;
- situacao e tipo: atributos da classe “PedidoModificacao” no qual definem em que etapa do processo está o pedido e qual sua finalidade;
- prioridade e esforcoEstimado: atributos da classe “AvaliacaoPedido” que definem a urgência do pedido, e o tempo estimado para realização da tarefa;
- decisão: atributo da classe “Decisão” que define a decisão do gerente de projetos

em permitir a implementação do pedido;

- g) resultado: atributo da classe “Auditoria” que serve para definir se a implementação do pedido foi aprovada ou reprovada;
- h) situacao: atributo da classe “ItemConfiguracao” que identifica se o item de configuração está liberado ou bloqueado para novas alterações;
- i) identificador: atributo da classe “LinhasBasicas” que identifica a versão da configuração do projeto;
- j) identificadorPai: atributo da classe “VersaoItem” que é utilizado quando criado um ramo para o item, identificando qual arquivo está sendo ramificado.

As principais operações contidas neste diagrama são:

- a) registrarPedidoModificacao(): operação para gravar um novo pedido de modificação;
- b) registrarAvaliacao(): operação para registrar a avaliação de um pedido, colocando-o em situação de aguardando decisão;
- c) registrarDecisao(): operação que registra a decisão do gerente, finalizando o pedido ou colocando o mesmo sob a responsabilidade de um programador;
- d) registrarInicio(): operação que muda a situação do pedido para “em desenvolvimento”, e copia os itens de configuração a serem alterados para o diretório do responsável. Este método utiliza-se da operação saidaRepositorio();
- e) registrarTermino(): operação que finaliza o desenvolvimento, mudando sua situação para “em auditoria”;
- f) registrarAuditoria(): operação para registrar a auditoria do desenvolvimento, se aprovado os itens de configuração serão gravados no repositório, este método utiliza-se da operação entradaRepositorio();
- g) saidaRepositorio(): operação para retirar um arquivo do repositório e copiar para o diretório do responsável. Este método utiliza-se da operação bloqueiaArquivo();
- h) entradaRepositorio(): operação para gravar um arquivo ou uma nova versão no repositório. Este método utiliza-se da operação desbloqueiaArquivo();
- i) bloqueiaArquivo(): operação para bloquear o arquivo para edição;
- j) desbloqueiaArquivo(): operação para desbloquear o arquivo para edição;
- k) criarRamificacao(): operação que permite criar uma nova modificação em qualquer versão de algum item de configuração, criando o desenvolvimento em uma linha paralela;
- l) compararVersoes(): operação que carrega as diferenças entre os históricos dos itens

de configuração.

3.3 IMPLEMENTAÇÃO

Os próximos tópicos abordam as técnicas e tecnologias utilizadas, operacionalidade da ferramenta e resultados obtidos.

3.3.1 Técnicas e ferramentas utilizadas

Este trabalho utilizou a plataforma Java 5.0, a IDE Eclipse 3.2 e o banco de dados MySQL 4.1. Os diagramas foram feitos no Enterprise Architect 6.0. Entre as principais tecnologias utilizadas neste trabalho, destacam-se o *framework* Hibernate e a arquitetura cliente-servidor.

Para a interface com o usuário foi utilizado o conjunto de ferramentas de interface gráfica do Java Swing. Este componente é completo, e permite desenvolver qualquer tipo de interface, além de permitir criar painéis, setar layouts, utilizar heranças, e vários outros recursos.

O sistema é executado através de duas aplicações, uma cliente e outra servidor. A aplicação cliente possui a interface gráfica com o usuário, enquanto que a outra aplicação possui diversas operações tais como a comunicação com o banco de dados, entrada e saída de objetos do repositório dentre outras funcionalidades que só são executadas por requisições de algum cliente. Para realizar essa comunicação cliente/servidor foi utilizada a tecnologia soquete.

Soquete é um objeto que encapsula uma conexão TCP/IP. Para se comunicar com a outra extremidade de conexão, são utilizados fluxos de entrada e saída anexados ao soquete (HORSTMANN, 2004, p. 816). O programa servidor quando carregado, fica na escuta em uma porta de comunicação apenas esperando requisição de algum cliente. Quando receber uma requisição é iniciado uma nova *Thread* onde a mesma vai executar as operações necessárias para realizar a requisição. Ao término, o sistema responderá ao cliente o resultado da operação.

O Hibernate é um *framework* de persistência de objeto relacional. Esta ferramenta é

um projeto *open source* que foi desenvolvido para facilitar o mapeamento das classes feitas em Java para tabelas no banco de dados. Utilizando esta ferramenta, o desenvolvedor não precisa se preocupar com os comandos *Structured Query Language* (SQL). Para realizar o mapeamento das classes com as tabelas correspondentes no banco de dados, é necessário que seja criado um arquivo XML para cada classe, onde a mesma relacionará os atributos da classe com a tabela do banco de dados (JBOSS INC, 2006).

Nos quadros 1, 2 e 3 são mostrados trechos de códigos de uma classe de negócio, um arquivo XML necessário para mapear essa classe com o banco de dados, e um código fonte de persistência, onde utiliza métodos do Hibernate para persistir os dados.

```
package componentes;

import java.io.Serializable;

public class Usuario implements Serializable {
    private int codigo;

    private String nome;

    private String login;

    private String senha;

    private String diretorio;

    private int tipo;

    public int getCodigo() {
        return codigo;
    }

    public String getNome() {
        return nome;
    }

    public String getLogin() {
        return login;
    }
}
```

Quadro 1 – Código fonte de uma classe de negócio

```

<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
    "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
    "C:\hibernate-2.1\src\net\sf\hibernate\hibernate-mapping-3.0.dtd">
<hibernate-mapping package="componentes">
    <class name="Usuario" table="usuario">
        <id name="codigo" column="codigo" type="int">
            <generator class="increment"></generator>
        </id>
        <property name="nome" column="nome" type="string" />
        <property name="tipo" column="tipo" type="int" />
        <property name="login" column="login" type="string" />
        <property name="senha" column="senha" type="string" />
        <property name="diretorio" column="diretorio" type="string" />
    </class>
</hibernate-mapping>

```

Quadro 2 – XML de mapeamento de uma classe de negócio

```

package componentes;

import persistencia.PersistenciaDAO;

public class UsuarioDAO extends PersistenciaDAO {

    public UsuarioDAO() throws Exception {
        super();
    }

    public void insert(Usuario usuario) throws Exception {
        Session session = factory.openSession();
        session.save(usuario);
        session.flush();
        session.close();
    }

    public void update(Usuario usuario) throws Exception {
        Session session = factory.openSession();
        session.update(usuario);
        session.flush();
        session.close();
    }

    public void delete(Usuario usuario) throws Exception {
        Session session = factory.openSession();
        session.delete(usuario);
        session.flush();
        session.close();
    }
}

```

Quadro 3 – Código fonte de uma classe de persistência

Em relação à funcionalidade de controle de versões, existem dois tipos de arquivos que a ferramenta deverá controlar as versões, que são arquivos de texto plano e arquivos binários.

Para os arquivos de texto plano foi adotada a mesma estrutura de gravação do trabalho de Bohn (2005), que utiliza um conjunto de *tags* para guardar a versão atual do arquivo e suas diferenças para as versões anteriores. Essas *tags* têm o seguinte formato:

- a) <ADD> - adiciona uma linha no arquivo;
- b) <TROCA> <Lx> *arg* - troca o conteúdo de *arg1* da linha *x*;
- c) <DELE> <Lx> - exclui a linha *x*;
- d) <HIST_VERSAO=*x*> - exibe o histórico das versões do arquivo, onde *x* é o identificador da versão;
- e) [VERSÃO=*x*] - define início de informações de uma versão, onde *x* é o identificador da versão;
- f) [AUTOR=*x*] - nome do responsável pela alteração;
- g) [DATA=*x*] - data de alteração do arquivo;
- h) [DESCRICAÇÃO=*x*] – descrição da alteração realizada.

O quadro 4 exibe um arquivo de versões, contendo todas as *tags*:

```
<HIST_VERSAO=1.1>
<HIST_VERSAO=1.0>

[VERSÃO=1.0]
[AUTOR=Iam Sommerville]
[DATA=01/12/2006]
[DESCRICAÇÃO=Adicionado no projeto]

[@BOF]
<TROCA> <L2> FURB
<DELE> <L4>
<ADD> Gerência de configuração de software
[@EOF]

[VERSÃO=1.1]
[AUTOR=Iam Sommerville]
[DATA=29/12/2006]
[DESCRICAÇÃO=Modificação para entrega do TCC]

[@BOF]
Trabalho de conclusão de curso
Centro de ciências exatas
BCC
Prof Everaldo
Aluno Rodrigo Furlaneto
Ferramenta de apoio à gerência de configuração
[@EOF]
```

Quadro 4 – Arquivo de versões completo

Porém para arquivos do tipo binário, a utilização de *tags* pode ocasionar problemas, pois estes arquivos podem ser danificados nas operações de entrega e retirada do repositório. Então para armazenar os arquivos binários, será necessário duplicá-los no repositório. Mas para não ocupar muito espaço, estes arquivos são compactados no formato ZIP. Para realizar a compactação foi utilizado a API Zip do Java. Esta API possui um conjunto de métodos para compactar e descompactar arquivos.

Para geração de relatórios foi utilizada a API *open source* iText. Com essa API é possível gerar documentos contendo textos, tabelas, e imagens, e apresenta diversos tipos de fontes. Pode ser usada em diversos tipos de aplicações, com suporte a geração de código de barras (ITEXT HOMEPAGE, 2006).

3.3.2 Operacionalidade da implementação

Esta ferramenta tem o objetivo de apoiar as atividades de gerência de configuração de software, definindo papéis para cada usuário, responsáveis pelos projetos, e controle de todo fluxo de modificações e liberações.

O gerente de projetos é o principal usuário da ferramenta, pois o mesmo tem acesso a todas as telas, sendo responsável pelos pré-cadastros necessários para o uso da ferramenta. Para ilustrar os cadastros básicos, parte-se do princípio que já exista um gerente de projetos cadastrado no sistema. O primeiro passo é o gerente efetuar o *login* para autenticar-se no sistema. Depois é preciso cadastrar todos os usuários necessários, dentre eles solicitantes, analistas, gerentes, programadores e auditores. Para isso deve acessar a tela de cadastro de usuários pelo menu principal arquivos e na sub-opção cadastrar usuários. A figura 16 mostra a tela de cadastro de usuários.

Usuários

Código
2

Nome
Rodrigo Furlaneto

Login
rodrigo

Senha

Tipo
Programador

Diretório de trabalho
C:\Rodrigo

Figura 16 – Tela de cadastro de usuários

A seguir é necessário que o gerente cadastre os projetos acessando a tela de cadastro de projetos pelo menu arquivos e sub-opção cadastrar projetos. A figura 17 mostra a tela de cadastro de projetos.

Projetos

Código
1

Nome
Ferram. de Apoio a Gerência de Conf. de Software

Descrição
Tem por objetivo gerenciar os pedidos de modificações, controlar versões dos itens de configuração e definir responsáveis pelas atividades.

Gerente de projeto
Everaldo Artur Grahl

Figura 17 – Tela de cadastro de projetos

Agora é necessário cadastrar a primeira linha básica do projeto entrando na tela de criação de linhas básicas pelo menu arquivos e sub-opção linhas básicas. Nesta tela deverão

ser cadastrados os tipos de itens de configuração e importados os itens de configuração de cada tipo. A figura 18 mostra a tela de criação de linhas básicas e geração de pacotes.

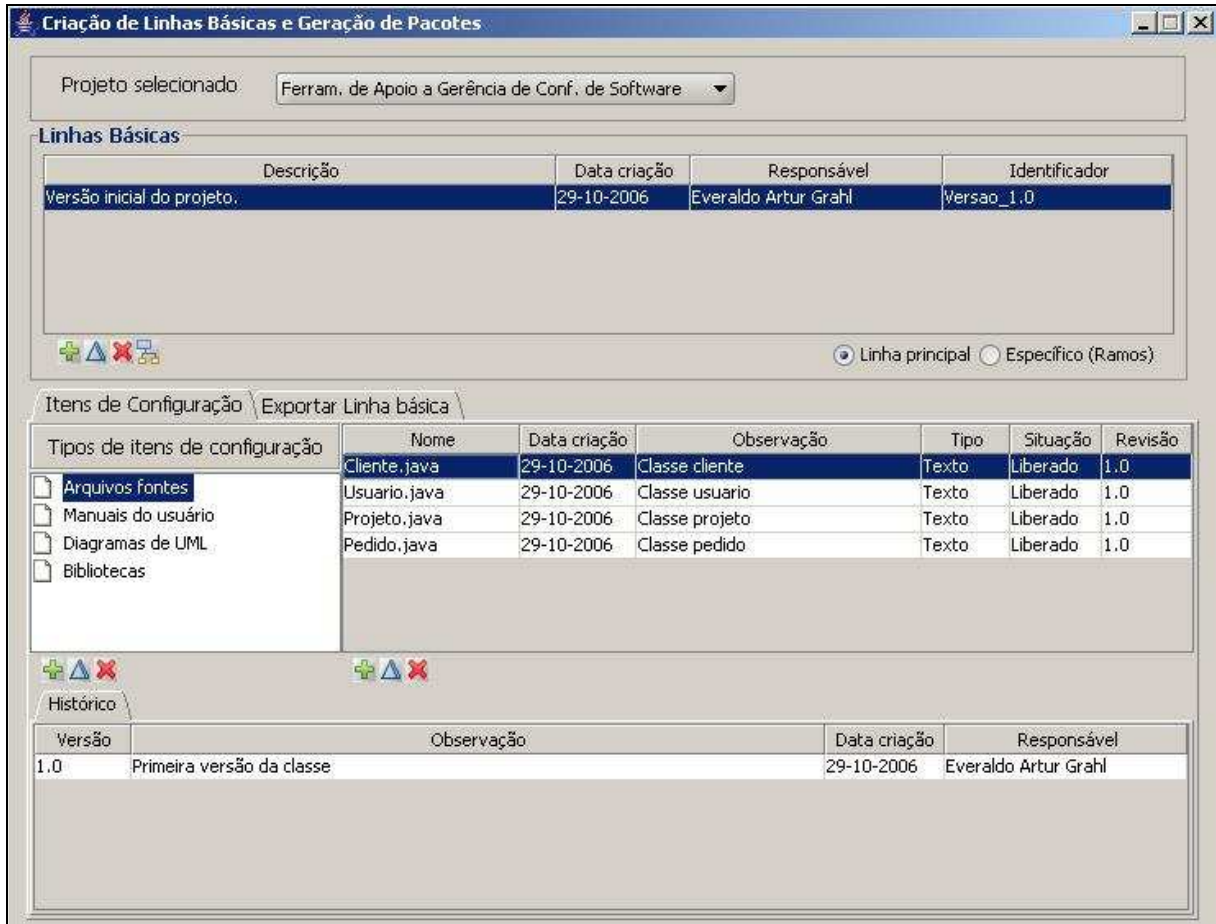


Figura 18 – Tela de linhas básicas e geração de pacotes

Para cadastrar a linha básica é necessário pressionar o botão com símbolo de adição abaixo da tabela de linhas básicas. A figura 19 mostra a tela que será aberta.

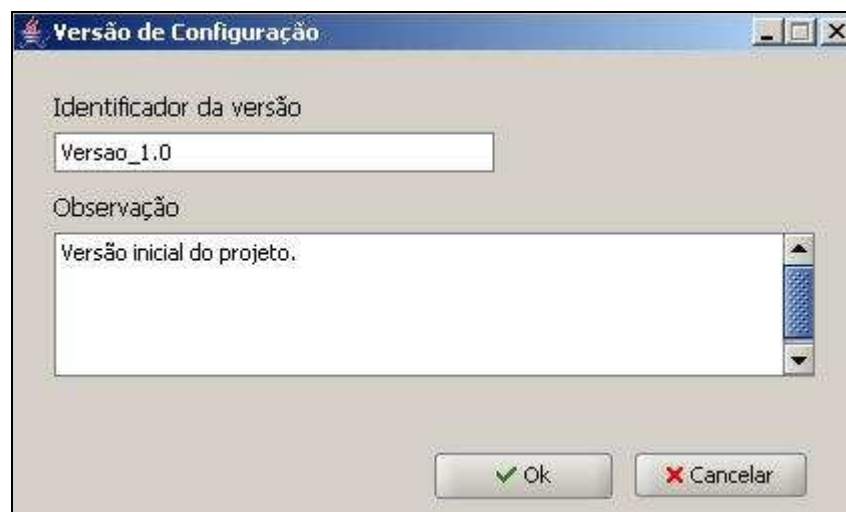


Figura 19 – Tela de cadastro de versão de configuração (linha básica)

Com a linha básica cadastrada é necessário importar os itens de configuração que farão parte dessa linha de desenvolvimento, mas antes é necessário cadastrar os tipos de itens através do botão de adição abaixo da tabela de tipos de itens de configuração. A figura 20 mostra a tela de cadastro de tipo de itens.

Figura 20 – Tela de cadastro de tipo de itens de configuração

Os tipos de itens determinam em qual diretório serão guardados os itens de configuração no repositório. A próxima etapa é importar os itens de configuração pressionando o botão de adição que se encontra abaixo da tabela de itens de configuração. A figura 21 mostra a tela que será exibida.

Nome do Arquivo	Formato	Comentário
Cliente.java	Texto	Classe cliente
Projeto.java	Texto	Classe projeto
Usuario.java	Texto	

Figura 21 – Tela para importar itens de configuração

Na tela da figura 21 podem ser importados vários arquivos simultaneamente, basta selecioná-los no diretório onde os arquivos se encontram. Antes de confirmar a importação é necessário selecionar o formato do arquivo e colocar um breve comentário para cada um deles. Após a confirmação de importação, os mesmos serão copiados para o diretório referente ao tipo de item selecionado pelo gerente. A figura 22 mostra os itens importados.

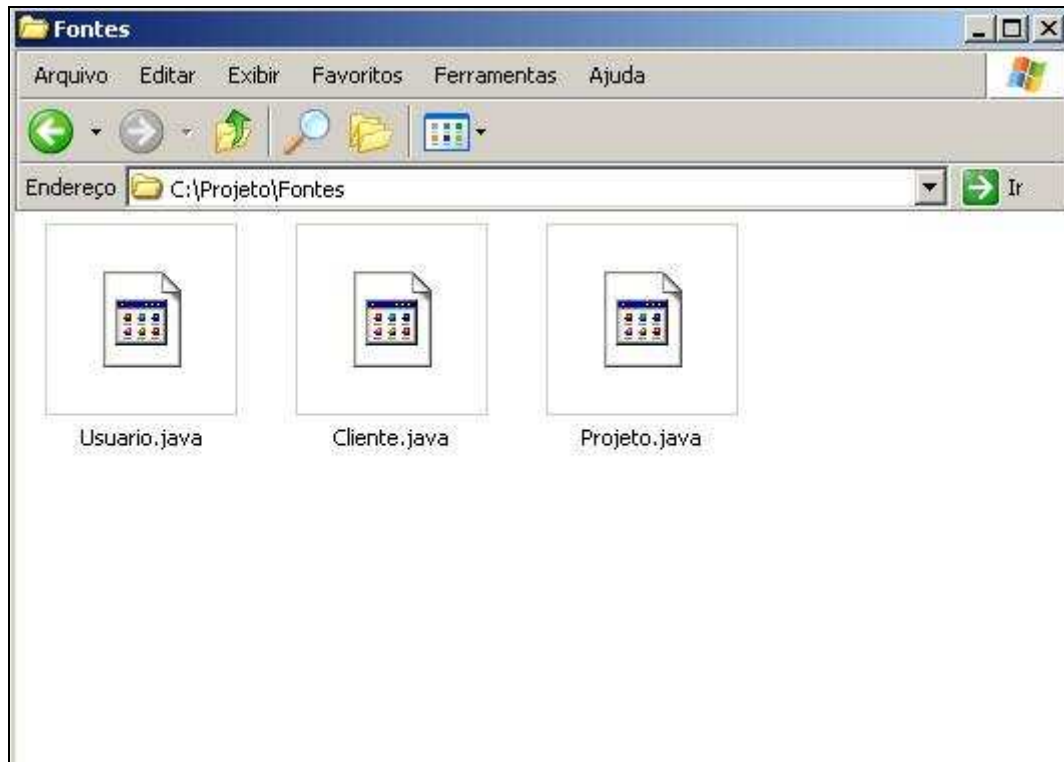


Figura 22 – Itens de configuração dentro do diretório do repositório

Ainda na tela de linhas básicas, existe uma funcionalidade muito importante quando se trata de desenvolvimento paralelo. Existe a possibilidade de criar ramificações nas linhas básicas selecionando a linha que se deseja criar um ramo e pressionar o botão “criar ramo” que se encontra abaixo da tabela de linhas básicas. A figura 23 mostra a tela de criação de ramos.

Ramificação de linhas básicas

Identificador da versão
Versao_1.0

Nome do ramo
Ramo_Versao_1.0

Observação
Correção de erros não encontrados na versão inicial do projeto.

Ok Cancelar

Figura 23 – Tela de ramificação de linhas básicas

Terminado essa etapa, o gerente de projetos deve definir as pessoas que serão responsáveis por esse projeto. Para isso o gerente deve acessar a tela de identificar responsáveis pelo menu arquivo e sub-opção identificar responsáveis. A figura 24 mostra a tela de identificar responsáveis.

Identificar Responsáveis

Projeto
Ferram. de Apoio a Gerência de Conf. de Software

Gerente
Everaldo Artur Grahl

Responsáveis

Nome	Tipo
Rodrigo Furlaneto	Programador
José C. Maldonado	Analista
Roger S. Pressman	Auditor

+ X

Figura 24 – Tela de identificação de responsáveis

A finalidade da tela da figura 24 é definir as pessoas que executarão as atividades relacionadas à gerência de configuração do projeto selecionado. Para selecionar os usuários pressiona-se o botão com o símbolo de adição.

Para finalizar os cadastros iniciais do sistema, o gerente de projetos deve cadastrar os clientes dos projetos pelo menu arquivo e sub-opção cadastrar clientes. Neste cadastro são informados o nome, endereço, tipo de pessoa, telefone e fax.

Realizados os cadastros anteriores, pode-se começar a registrar os pedidos de modificação. Para isso deve-se entrar no menu arquivo e sub-opção pedido de modificação. A figura 25 exibe a tela de pedidos de modificação.

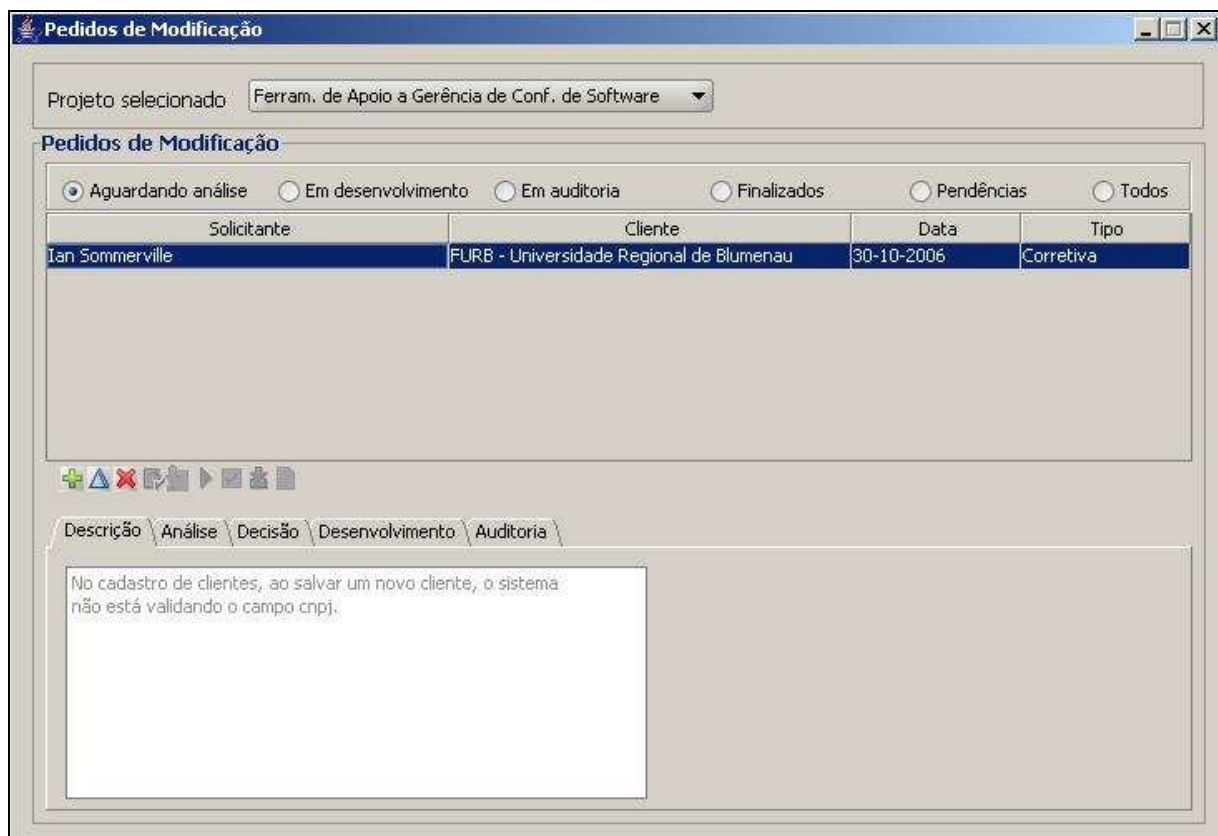


Figura 25 – Tela de pedidos de modificação

Esta é a principal tela do sistema. É a única tela que todos os usuários têm acesso. É possível filtrar os pedidos por aqueles que estão aguardando análise, em desenvolvimento, em auditoria, todos e pendências. A opção pendências mostra apenas os pedidos que estão sob responsabilidade do usuário logado, no qual deverá realizar alguma ação. Para registrar um novo pedido de modificação, o usuário deve ser do tipo solicitante, ou analista de sistemas. Pressiona-se o botão com símbolo de adição para registrar um novo pedido de modificação. A figura 22 mostra a tela de registro de pedido de modificação.

Pedido de Modificação

Solicitante
Ian Sommerville

Cliente
FURB - Universidade Regional de Blumenau

Analista responsável
José C. Maldonado

Tipo
 Corretiva
 Adaptativa
 Nova Funcionalidade

Descrição
No cadastro de clientes, ao salvar um novo cliente, o sistema não está validando o campo cnpj.

✓ Ok ✗ Cancelar

Figura 26 – Tela de registro de pedido de modificação

Na tela da figura 26 é informado o cliente que solicitou a modificação, um analista responsável pelo projeto, o tipo de modificação, que pode ser corretiva, adaptativa ou uma nova funcionalidade e uma breve descrição do que deve ser feito. Depois de registrado o pedido, o mesmo cai nas pendências do analista selecionado. Agora este analista precisa registrar a avaliação deste pedido pressionando o botão com o símbolo de uma folha azul na tela da figura 25. A figura 27 mostra a tela de registro de avaliação do pedido.

Projeto: Ferram. de Apoio a Gerência de Conf. de Software

Analista de sistemas: José C. Maldonado

Pedido: No cadastro de clientes, ao salvar um novo cliente, o sistema não está validando o campo cnpj.

Descrição da avaliação: Modificação relativamente simples de implementar. Deve ser alterado o método validarObjeto() da classe cliente.

Release de liberação:

- Nova release
- Especifico (ramificação)

Prioridade:

- Alta
- Média
- Baixa

Esforço: horas 5

Item de Configuração	Revisão
Cliente.java	1.0

Ok Cancelar

Figura 27 – Tela de registro de avaliação do pedido

Na tela da figura 27 o analista de sistemas vai informar uma descrição detalhada de como realizar a implementação, selecionar qual a *release* de liberação que ficará disponível essa modificação, a prioridade, o tempo de esforço, e os itens de configuração que serão alterados pelo programador. Em relação a *release* de liberação, se o analista escolher a opção “nova *release*” a modificação ficará disponível para a última linha básica cadastrada para o projeto. Caso a opção for “específico”, será habilitado o campo abaixo, onde listará os ramos existentes no projeto. Esta opção é para desenvolvimentos que precisam alterar versões de configuração antigas, no qual já foram finalizadas, mas que por uma necessidade seja necessário realizar a alteração. Como uma versão antiga não é permitida alteração, é possível criar ramos para suprir a necessidade para um determinado cliente. Finalizando o registro de avaliação do pedido, a próxima etapa será de responsabilidade do gerente do projeto que deverá registrar a decisão do pedido, se o mesmo será realizado ou não. Para isso deve-se pressionar o botão com uma flecha amarela na tela da figura 25. A figura 28 ilustra a tela de registro de decisão.

Registro de Decisão

Projeto: Ferram. de Apoio a Gerência de Conf. de Software

Gerente de Projeto: Everaldo Artur Grahl

Parecer: Aprovado

Descrição: Modificação aprovada, por se tratar de um erro que deve ser corrigido.

Responsáveis:

Programador: Rodrigo Furlaneto

Auditor: Roger S. Pressman

Ok Cancelar

Figura 28 – Tela de registro de decisão

Na tela da figura 28 o gerente de projetos informará um parecer e uma breve descrição. Se o parecer for “aprovado”, o gerente deverá informar os próximos responsáveis pelo pedido, que são o programador e o auditor. Agora a responsabilidade é do programador e o pedido aparecerá como sua pendência. Quando ele realizar essa atividade é necessário que registre o início do desenvolvimento pressionando a flecha azul na tela da figura 25. O sistema pedirá uma confirmação. Ao confirmar, todos os itens de configuração definidos pelo analista serão copiados do repositório para sua área de trabalho local. Estes itens de configuração ficarão bloqueados para outras modificações. Quando terminadas as modificações é o momento de finalizar o desenvolvimento. Para isso o programador pressiona o botão com um *check* azul na tela da figura 25. Neste momento abrirá uma tela onde deverá ser informada uma descrição do que foi realizado em cada objeto. A figura 29 mostra a tela.

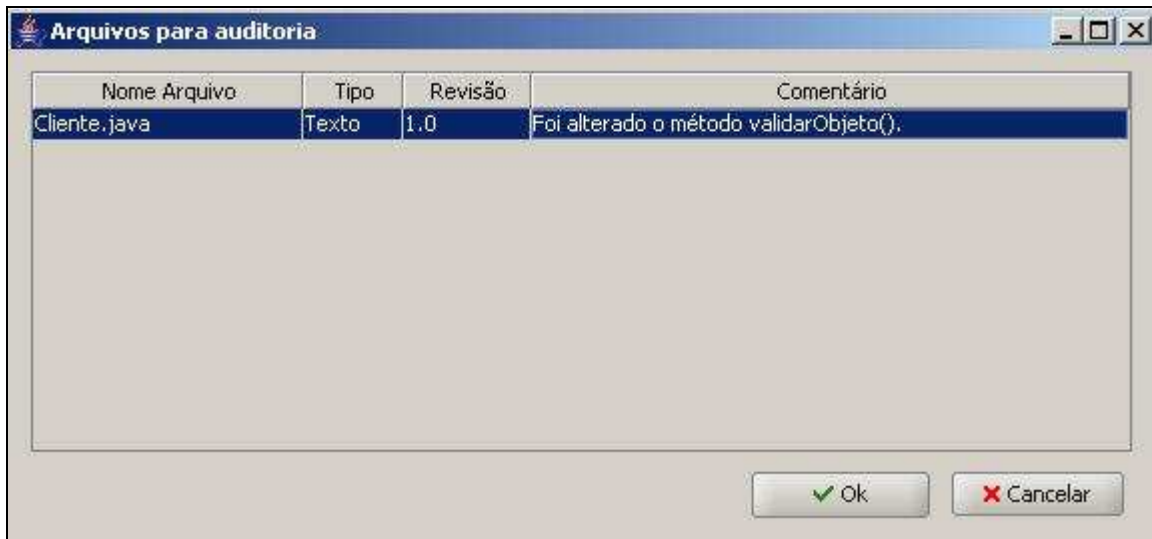


Figura 29 – Tela de liberação de arquivos para auditoria

Após confirmar os arquivos serão enviados para o servidor, mas ainda não serão colocados em uma linha base. Neste momento o pedido está na última etapa, que é de auditoria. O auditor poderá baixar para sua máquina os arquivos alterados através do botão com uma flecha apontando para baixo na tela da figura 25. Após realizar os testes necessários o auditor poderá registrar a auditoria pressionando o botão com uma folha branca na tela da figura 25. A figura 30 ilustra a tela de registro de auditoria.



Figura 30 – Tela de registro de auditoria

Na tela da figura 30 o auditor deverá informar o resultado da auditoria e uma descrição. Se o resultado for reprovado, o pedido volta para o programador corrigir os

problemas. Caso a auditoria seja aprovada os itens de configuração serão guardados no repositório gerando uma nova versão, e liberando o item de configuração para novas modificações. Esta nova versão será colocada na linha básica que foi definida pelo analista. Após todas essas etapas concluídas, o gerente de projetos pode gerar um pacote com todos os itens de configuração de qualquer versão do projeto. Para isso deverá acessar a tela de linhas básicas, entrar na página “exportar linha básica”. Nesta página deverá selecionar um diretório e pressionar o botão exportar. Neste momento será gerado um arquivo, com todos os itens de configuração compactados.

Assim, o fluxo de atividades termina. A ferramenta apoiou os processos em todas as etapas. Algumas situações devem ser tomadas pelas pessoas envolvidas, tais como a criação de ramos, liberação dos pacotes para entrega, entre outros. Porém nas etapas básicas a ferramenta apóia de maneira satisfatória.

Para visualizar as diferenças entre versões o usuário deverá acessar a tela de linhas básicas (Figura 18), selecionar o item de configuração e os seus históricos, depois pressionar o botão com duas folhas sobrepostas abaixo da tabela de itens. Neste momento será exibida uma tela com as duas versões selecionadas uma ao lado da outra, as linhas que diferem são destacadas por uma seta. A Figura 31 exibe a tela de diferenças de arquivos.

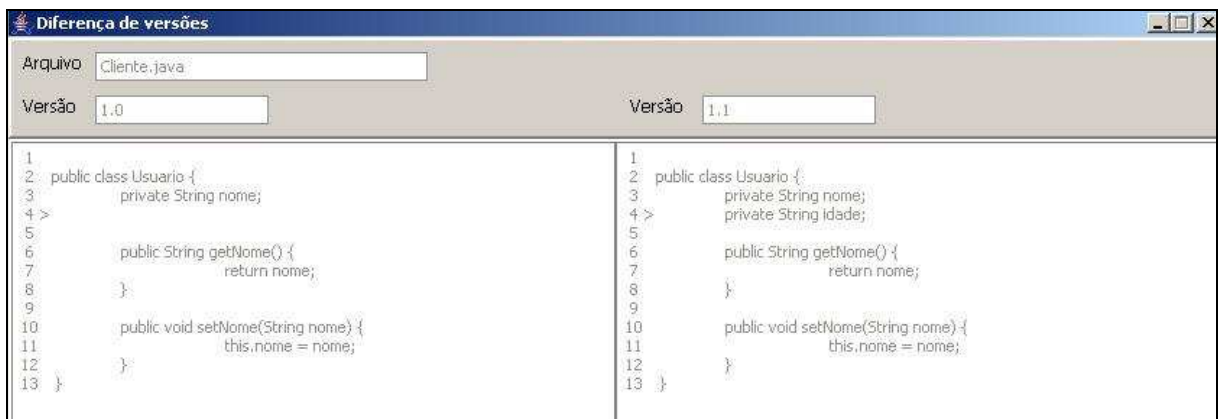


Figura 31 – Tela de diferença de versões

Para visualizar o andamento das modificações seleciona-se o menu relatórios e sub-opção acompanhamento das modificações. Neste relatório é possível visualizar todos os pedidos de modificações abertos por projeto (figura 32).

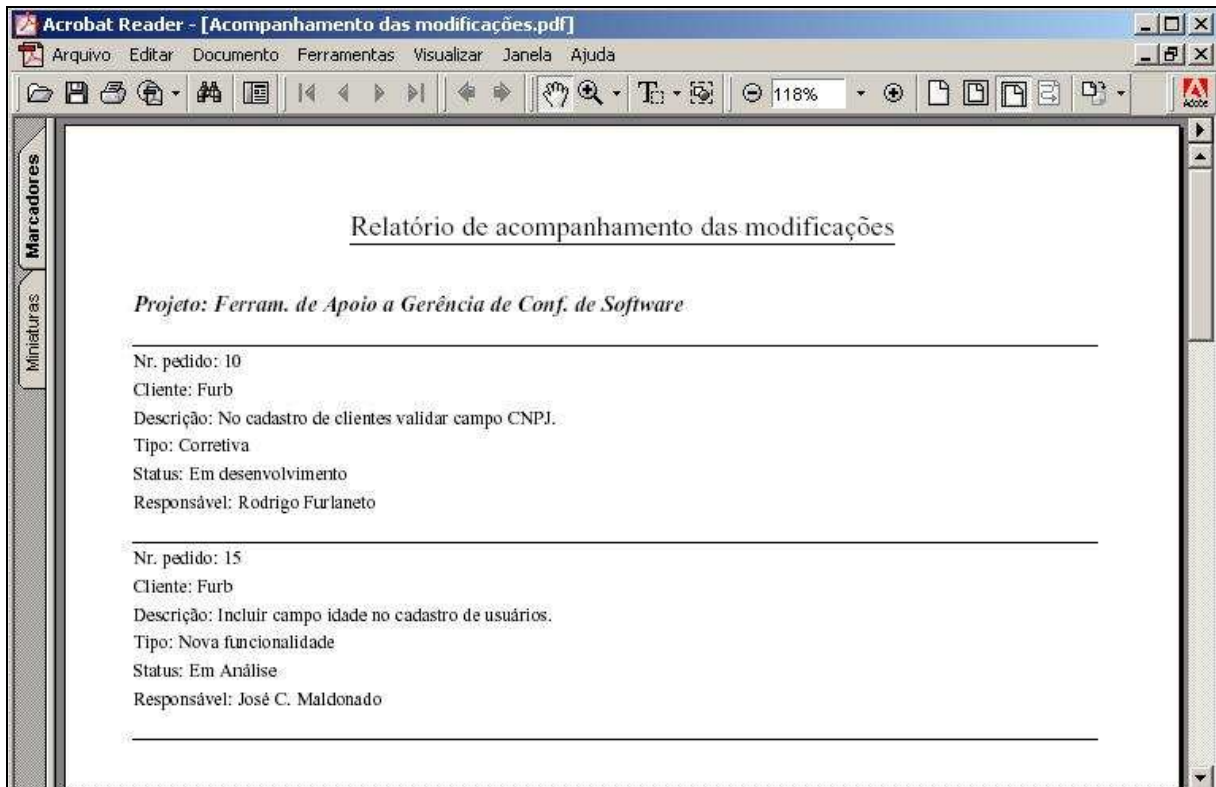


Figura 32 – Relatório de acompanhamento das modificações

A figura 32 apresenta o relatório de linhas básicas, que pode ser acessado pelo menu relatórios e sub-opção linhas básicas.

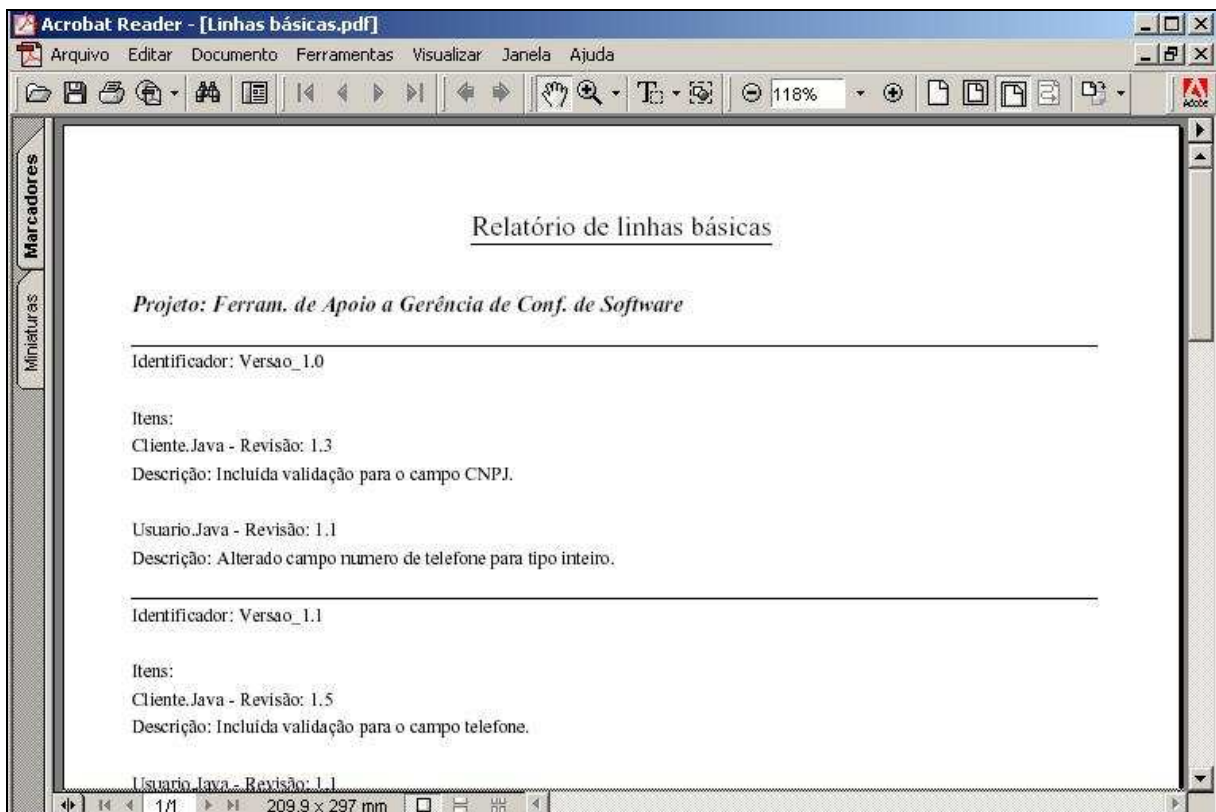


Figura 33 – Relatório de linhas básicas

O relatório da figura 33 exibe todas as linhas básicas que um projeto possui, detalhando as versões dos itens que o compõe. A figura 34 exibe o relatório de auditorias que pode ser acessado pelo menu relatórios e sub-opção auditorias, onde poderão ser visualizadas todas as auditorias que foram realizadas em um projeto.

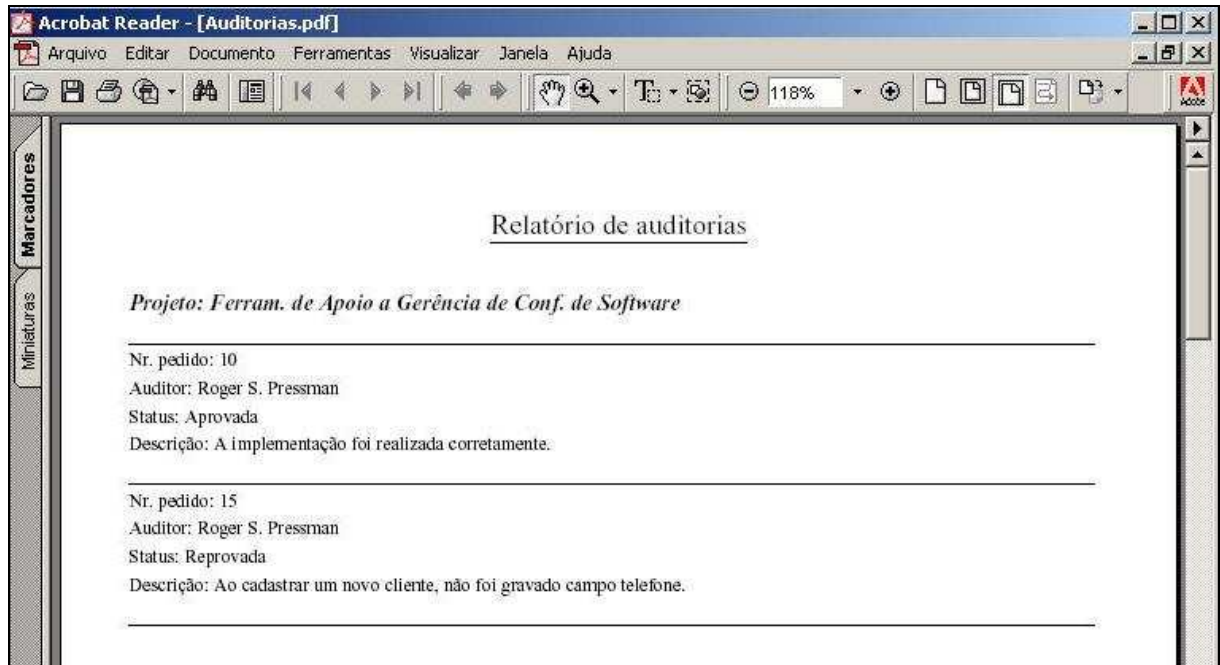


Figura 34 – Relatório de auditorias

Para visualizar os históricos dos itens de configuração que compõe um projeto pode-se acessar o menu relatórios e sub-opção históricos. A figura 35 exibe o relatório de histórico dos itens.

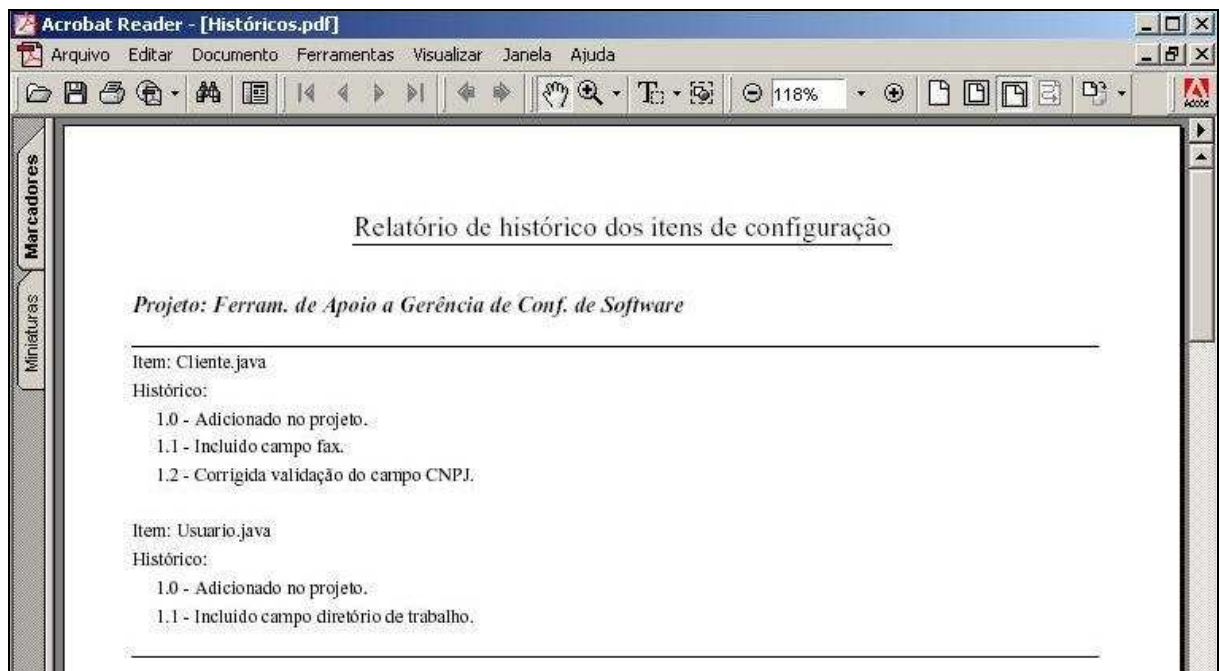


Figura 35 – Relatório de histórico dos itens de configuração

3.4 RESULTADOS E DISCUSSÃO

No trabalho de Bohn (2005) foi desenvolvida uma ferramenta de apoio à gerência de configuração seguindo o modelo CMMI. Esta ferramenta possui algumas limitações quanto à definição de responsáveis por projeto. Apresenta apenas três tipos de usuários, que são analista, gerente e auditor, ou seja, um processo inicia direto em um analista, ou gerente, pois a ferramenta não permite outros tipos de usuários. Outra limitação é quando um pedido de mudança é registrado por um analista e aprovado por um gerente, o responsável pela implementação é o próprio analista que registra o pedido. É o analista de sistemas que registra o pedido, que o analisa e que implementa, não podendo fluir estas atividades para outros usuários. Outro problema é em relação às versões dos arquivos de configuração, pois a mesma só armazena arquivos textos no repositório.

A ferramenta desenvolvida neste trabalho apresentou algumas evoluções. Nela é possível definir todos os usuários que serão responsáveis por um projeto, entre eles o gerente, os analistas, programadores e auditores. Os pedidos podem ser registrados por um analista ou por um usuário solicitante, que pode ser de um setor de *helpdesk* ou um cliente diretamente. O restante do fluxo pode ser conferido no tópico 3.2. Em relação aos itens de configuração, o repositório guarda arquivos do tipo binário e texto, ou seja, pode-se manter o controle de todos os itens de configuração do projeto como manuais, diagramas, arquivos compilados, não sendo limitado a código fonte. Outro ponto interessante é que o sistema utiliza um protocolo de rede para trafegar os arquivos do repositório do servidor para uma estação cliente e vice-versa, isto mantém um nível de segurança maior em relação às cópias dos arquivos. A ramificação de projetos também é algo essencial que foi desenvolvido neste trabalho e não foi verificado em trabalhos correlatos.

3.4.1 Comparativo das ferramentas

Subversion e Trac, são ferramentas *open source* desenvolvidas para auxiliar no apoio a gerência de configuração. Essas duas ferramentas atendem processos distintos dentro da área estudada.

A principal característica do Subversion de forma resumida é manter, registrar, rastrear e armazenar os históricos dos itens de configuração. Esta ferramenta dispõe de muitas

funcionalidades avançadas para criação de ramos, operações de controle de versionamento para qualquer tipo de arquivo, seja ele binário ou texto. Mas para a gerência de configuração, apenas essas funcionalidades não bastam. É necessário manter um controle das modificações realizadas pelos usuários da ferramenta, e para isso existe uma outra ferramenta que pode ser utilizada em conjunto.

Esta outra ferramenta é o Trac, que tem como principal funcionalidade o controle de mudanças. Uma vantagem é que ela pode ser integrada com o Subversion. Possui funcionalidade de rastreamento de mudanças, acompanhamento da evolução de projetos e é claro o registro de pedido de modificação.

Estas ferramentas são boas opções para empresas de pequeno, médio e até grande porte, por serem bem completas, e em constante evolução. Mas a principal vantagem é que são gratuitas.

Em relação à ferramenta desenvolvida, apesar dela ser um protótipo, atende adequadamente as etapas da gerência de configuração. A mesma atende funcionalidades de controle de versões em conjunto com o controle de mudanças.

Em relação à comparação da ferramenta desenvolvida com as ferramentas *open source* estudadas, foram definidos dois critérios para avaliação, uma para o controle de versões e outra para o controle de mudanças. Como o Subversion e o Trac atendem atividades distintas, a comparação será diferente para cada uma. Através do estudo desenvolvido foram definidos alguns requisitos que são importantes para o controle de versões e controle de mudanças. Os quadros 5 e 6 apresentam os resultados obtidos nas ferramentas estudadas com a desenvolvida.

	Subversion	Ferramenta desenvolvida
Registro de rastreamento	Sim	Sim
Definições e gerenciamento de múltiplas versões	Sim	Sim
Geração de versões	Sim	Sim
Capacidade de arquivamento	Sim	Sim
Ramificação de arquivos	Sim	Sim

Quadro 5 – Análise das ferramentas de gerência de configuração no quesito controle de versões

	Trac	Ferramenta desenvolvida
Acompanhamento do ciclo de vida do pedido de mudança	Sim	Sim
Rastreamento da mudança	Sim	Sim
Anexação de arquivos ao pedido	Sim	Não
Configuração do fluxo de trabalho	Não	Não
Notificações para acompanhamento da evolução do pedido	Sim	Não

Quadro 6 – Análise das ferramentas de gerência de configuração no quesito controle de mudanças

3.4.2 Análise de aderência ao modelo MPS.BR

Para atender a gerência de configuração em relação ao MPS.BR é necessário suportar os nove resultados esperados. O quadro 7 mostra os resultados esperados e o grau de atendimento da ferramenta. As escalas utilizadas foram: atende, atende parcialmente e não atende.

Resultados esperados	Ferramenta desenvolvida
GCO 1. Os itens de configuração são identificados	Atende
GCO 2. Os itens de configuração gerados pelo projeto são definidos e colocados sob uma linha básica	Atende
GCO 3. É estabelecido e mantido um sistema de gerência de configuração	Atende Parcialmente
GCO 4. As modificações e liberações dos itens de configuração são controladas	Atende
GCO 5. As modificações e liberações são disponibilizadas para todos os envolvidos;	Atende
GCO 6. A situação dos itens de configuração e as solicitações de mudanças são registradas, relatadas e o seu impacto é analisado	Atende
GCO 7. A completeza e a consistência dos itens de configuração são	Atende

asseguradas	
GCO 8. O armazenamento, o manuseio e a entrega dos produtos de trabalho são controlados	Atende Parcialmente
GCO 9. A integridade das linhas básicas é estabelecida e mantida, através de auditoria da configuração e de registros da gerência de configuração	Atende

Quadro 7 – Quadro com os resultados esperados pelo modelo MPS.BR

A seguir é realizada uma breve justificativa em relação a cada resultado esperado.

- a) GC01: a ferramenta permite registrar itens de configuração, através da tela de linhas básicas(figura 18);
- b) GC02: a ferramenta mantém todos os itens de configuração em linhas básicas;
- c) GC03: a ferramenta permite definir responsáveis por projeto, onde cada um é responsável por uma atividade distinta, porém o fluxo de trabalho é fixo tornando essa tarefa limitada;
- d) GC04: a ferramenta só permite alterações e liberações dos itens de configuração, através de um pedido de modificação aprovado por um gerente;
- e) GC05: a ferramenta permite verificação das modificações e liberações para todos os usuários envolvidos;
- f) GC06: a ferramenta permite registrar pedidos de modificação através da tela de pedidos (figura 25), assim como sua análise na tela de avaliação do pedido (figura 27);
- g) GC07: a ferramenta garante a completeza e a consistência dos itens, pois os mesmos só poderão ser modificados através de processos definidos, mantendo um histórico de alterações com seus respectivos responsáveis;
- h) GC08: a ferramenta controla o armazenamento e o manuseio dos produtos, porém a entrega não é formalizada;
- i) GC09: a ferramenta registra novas versões de linhas básicas com aprovações de registro de auditoria através da tela de auditoria (figura 29).

Dos nove resultados esperados a ferramenta atende sete e atende parcialmente duas. Esta análise mostra que a ferramenta está bem aderente as diretrizes do modelo MPS.BR.

4 CONCLUSÕES

As ferramentas de apoio à gerência de configuração são fundamentais para corporações que pretendem melhorar o desenvolvimento de seus projetos e organização de suas equipes de desenvolvimento. A utilização de um modelo brasileiro de processos e qualidade de software como o MPS.BR contribui muito para estas empresas conseguirem se adaptar da melhor forma a modelos reconhecidos mundialmente.

Em relação aos objetivos definidos para esse trabalho, a ferramenta atendeu bem os resultados esperados pelo modelo MPS.BR. Com isso, conclui-se que é possível que pequenas e médias empresas melhorem a qualidade de seus processos, atividades, e com isso o produto final, sem muito investimento em ferramentas de trabalho. Mas para isso é necessário que as pessoas respeitem as regras da gerência de configuração.

O comparativo com ferramentas *open source* existentes no mercado deixa claro que a ferramenta atende aspectos importantes tanto no processo de controle de versões como o controle de mudanças. Além disso, permitiu uma melhor visão das necessidades para essa atividade além do que é proposto pelo MPS.BR.

A evolução deste trabalho em relação a trabalhos correlatos da Furb não ficou restrita só a nova plataforma e uma arquitetura diferente. Nesta ferramenta é possível guardar históricos de qualquer tipo de item de configuração. Além do mais, as atividades da gerência de configuração são distribuídos aos usuários da ferramenta através de um fluxo de trabalho.

A definição do processo de gestão de configuração foi elaborada com estudos nas bibliografias encontradas e se adaptou bem as necessidades da gerência de configuração de software. Neste processo é possível executar as atividades básicas da gerência de configuração como registros de pedidos de mudanças, controle de versões dos objetos de um projeto, e definir os usuários que executarão cada atividade.

As tecnologias utilizadas contribuíram para este desenvolvimento. Algumas foram estudadas no decorrer do trabalho, como o Hibernate e o iText. A arquitetura cliente-servidor permite que os usuários da ferramenta possam trabalhar em equipe, mesmo em localidades diferentes. Existem algumas limitações como algoritmos apropriados para controlar versões de arquivos binários. Outra limitação é a carência de desenvolvimentos concorrentes, onde cada vez que fosse atualizada uma versão de um arquivo, o sistema atualizasse o arquivo do outro usuário, tornando o trabalho em equipe sincronizado. A principal limitação do projeto é em relação à definição do processo de gestão de configuração, onde o fluxo de atividades é

fixo. Deveria haver a possibilidade de modelar esse fluxo, facilitando ainda mais sua implantação em corporações.

Esse trabalho foi desenvolvido com bases de modelos de referência e literatura de engenharia de software. A sua definição é simples e pode ser estendida para realização de outros trabalhos. Por fim, conclui-se que a ferramenta atingiu o proposto, e atendeu as principais atividades de gerência de configuração de software.

4.1 EXTENSÕES

Como possíveis extensões do trabalho citam-se:

- a) desenvolver um algoritmo específico para armazenar arquivos binários;
- b) permitir anexar arquivos nos pedidos de modificação;
- c) permitir modelar o fluxo de atividades;
- d) implementar a política de bloqueio “copia-modifica-resolve”;
- e) implementar novas funcionalidades para atender 100% os resultados esperados pelo processo de gerência de configuração do modelo MPS.BR.

REFERÊNCIAS BIBLIOGRÁFICAS

ASSOCIAÇÃO PARA PROMOÇÃO DA EXCELÊNCIA DO SOFTWARE BRASILEIRO. **MPS.BR**: melhoria de processo do software brasileiro. São Paulo: Softex, 2006. 56 p. Disponível em: <<http://www.softex.br>>. Acesso em: 15 abr. 2006.

BARBARESCO, Eduardo Alexandre. **Software de apoio ao processo de gerência da configuração segundo normas e modelos da qualidade**. 2000. 64 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

BOHN, André. **Ferramenta de apoio à gerência de configuração de software baseado no modelo CMMI**. 2005. 83 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

CAETANO, Cristiano. **Controle de versões e desenvolvimento colaborativo de software**. São Paulo: Novatec, 2004. 146 f.

DIAS, André Felipe. **Gêrencia de configuração**. BRA, [2006?]. Disponível em: <<http://www.pronus.eng.br>>. Acesso em: 10 out. 2006.

FARIAS, Alexsandro José de Melo. **Gerência de configurações de software padrões e ferramentas**. 2004. 64 f. Trabalho de Conclusão de Curso (Graduação em Ciências da Computação) – Centro de Informática, Universidade Federal de Pernambuco, Pernambuco.

FIGUEIREDO, Sávio; SANTOS, Gleison; ROCHA, Ana Regina. **Gerência de configuração em ambientes de desenvolvimento de software orientados a organização**. 2005, 14 f. Programa de Engenharia de Sistemas e Computação, Rio de Janeiro.

HORSTMANN, Cay. **Big Java**. São Paulo: Bookman, 2004, 1125 f.

ITEXT HOMEPAGE, **iText**. USA, [2006?]. Disponível em: <<http://www.lowagie.com/iText/>>. Acesso em: 20 out. 2006.

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

PARREIRAS, Fernando S.; BAX, Marcello P. **A gestão de conteúdos no apoio a engenharia de software**. 2003. 15 f. Programa de Pós-Graduação em Ciências da Informação, Escola de Ciência da Informação, Universidade Federal de Minas Gerais, Minas Gerais.

PAULA FILHO, Wilson de Pádua. **Engenharia de software: fundamentos, métodos e padrões**. Rio de Janeiro: Livros Técnicos e Científicos, 2001. 581 p.

PRESSMAN, Roger S. **Engenharia de software**. Tradução José Carlos Barbosa dos Santos. Rio de Janeiro: McGraw-Hill, 2006. 1056 p.

ROCHA, Ana Regina Cavalcanti; MALDONADO, José Carlos; WEBER, Kival Chaves. **Qualidade de software: teoria e prática**. São Paulo: Pretice Hall, 2001. 301 p.

SOMMERVILLE, Ian. **Engenharia de software**. 6. ed. Tradução Maurício de Andrade. São Paulo: Addison Wesley, 2003. 592 p.

SUBVERSION. **Version control system**. USA, [2006?]. Disponível em: <<http://subversion.tigris.org/>>. Acesso em: 20 out. 2006.

TRAC. **Integrated SCM & project management**. USA, [2006?]. Disponível em: <<http://trac.edgewall.org/>>. Acesso em: 20 out. 2006.