BRUNO MAIOLI MARTINS

LUPA ADMINISTRADOR – FERRAMENTA WEB PARA GERENCIAMENTO DE CÓDIGO FONTE UTILIZANDO REPOSITÓRIOS CVS

ASSIS
2008

LUPA ADMINISTRADOR – FERRAMENTA WEB PARA GERENCIAMENTO DE CÓDIGO FONTE UTILIZANDO REPOSITÓRIOS CVS

BRUNO MAIOLI MARTINS

Trabalho de Conclusão de Curso apresentado ao Instituto Municipal de Ensino Superior de Assis, como requisito do Curso de Graduação, analisado pela seguinte comissão examinadora:

Orientador: _____

Analisador(1): _____

Analisador(2): _____

Assis
2008

BRUNO MAIOLI MARTINS

LUPA ADMINISTRADOR – FERRAMENTA WEB PARA GERENCIAMENTO DE CÓDIGO FONTE UTILIZANDO REPOSITÓRIOS CVS

Trabalho de Conclusão de Curso apresentado ao Instituto Municipal de Ensino Superior de Assis, como requisito do Curso de Graduação, analisado pela seguinte comissão examinadora:

Orientador: _____

Área de Concentração: _____

Assis
2008

DEDICATÓRIA

Dedico este trabalho a dois excepcionais mestres, meus pais que me ensinaram que a dignidade de um homem esta diretamente relacionada a sua sabedoria, e que acima de tudo deve-se haver humildade.

AGRADECIMENTOS

A professora, Diomara Martins Reigato Barros pela orientação e pelo constante estímulo transmitido durante o trabalho.

A Maximiliano Maioli Martins, irmão excepcional, que a todo o momento me aconselhou nas escolhas da vida. Uma frase “para ser um ótimo cientista, deve-se primeiro ser um bom agricultor”.

A Patrícia Francielle Perecim, namorada que se dedicou com todo o seu companheirismo, depositada em mim sua força. Uma frase “tudo que você quer, você vai conseguir”.

A Pollyanna Cristína Perecim, pelas conversas, sorrisos e toda a bagunça. Uma frase “vai lá cunhado”.

Aos familiares, que me ajudaram tanto para que a conclusão deste curso pudesse ser concretizada.

RESUMO

Este trabalho demonstrará a ferramenta Lupa Administrador, tal ferramenta trabalhará na rastreabilidade de código fonte, auxiliando a gerencia de projeto no ciclo de desenvolvimento, pois atualmente existe a grande necessidade de artefatos que possam dar base sólida para o gerenciamento de código fonte na área de produção. Como mecanismo utilizado, o sistema de controle de versões CVS, bem como a infra-estrutura da tecnologia Java com os frameWorks JSF e AJAX4JSF, tendo como função principal da ferramenta coletar dados e gerar informações para posteriores análises dos gerentes de projeto no desenvolvimento do código fonte.

Palavras-chave: CVS. Rastreabilidade. Análise.

ABSTRACT

This work demonstrate a tool Lupa Administrador, such tool work in traceability of the source, helping the project maneger on cycle of the development becouse currently exists big need of the artifacts tha give base solid for management of the source in the area production. How used mechanism, the system control version (CVS), and the infrastructure the Java technology with frameWork JSF and AJAX4JSF, major function of the tool collect data and generate information for further analysis by managers of the project in the development of the source.

Keywords: *CVS. Traceability. Analysis*.

LISTA DE ILUSTRAÇÕES

Figura 1. Modelo funcional. 15
Figura 2. Modelo lógico. 16
Figura 3. Organização das versões no repositório. 17
Figura 4. Os colaboradores efetuaram o download do mesmo documento do servidor. 18
Figura 5. Trabalhando simultaneamente, mais em locais diferentes. 18
Figura 6. Envia as alterações, servidor aceita e gera uma nova versão. 19
Figura 7. Identifica uma nova versão e informa ao colaborador. 19
Figura 8. Colaborador B atualiza a versão local e reenvia a mesma para o servidor gerando uma nova versão. 20
Figura 9. Cliente enviando requisição para o servidor por meio de um *browser*. 21
Figura 10. Clientes enviando solicitações – Servidor procurando operação correspondente da solicitação. 22
Figura 11. Enviando a resposta para o cliente. 22
Figura 12. Criando a versão 1.1. 25
Figura 13. Criando a versão 1.2. 25
Figura 14. Quinto mês de desenvolvimento obtém-se a versão 1.5. 26
Figura 15. Trabalhando paralelamente. 26
Figura 16. Sexto mês – as duas equipes submeteram seus códigos ao repositório, gerando as versões. 27
Figura 17. Rastreabilidade de código fonte e distinção entre versões. 28
Figura 18. Funcionamento básico da ferramenta. 31
Figura 19. Gerando arquivos de configuração. 33
Figura 20. Ajax manipulando dados. 34
Figura 21. Processamento interno de comandos. 35
Figura 22. Página inicial. 36
Figura 23. Administrativo. 37
Figura 24. Página Rastreabilidade / Analise. 37
Figura 25. Arquivo e revisões selecionados. 38
Figura 26. Revisão 1 – Arquivo. 38
Figura 27. Revisão 2 – Arquivo. 38
Figura 28. Diferença. 39
Figura 29. Disposição do projeto. 39

LISTA DE TABELAS

Tabela 1 . Comandos existentes dentro do arquivo de configuração. 35

LISTA DE ABREVIATURAS E SIGLAS

CVS	Sistema de Controle de Versões
SO	Sistema Operacional
AJAX	Asynchronous Javascript and XML
JSF	Java Server Faces
JSP	Java Server Pages
ISO	International Organization for Standardization
AJAX4JSF	Asynchronous Javascript and XML 4 Java Server Faces
MVC	Model View Control
SSH	Secure Shell

SUMÁRIO

1 INTRODUÇÃO 12
1.1 OBJETIVOS 12
1.2 JUSTIFICATIVAS E MOTIVAÇÃO 12
1.3 ESTRUTURA DO TRABALHO 13
2 SISTEMA DE CONTROLE DE VERSÕES (CVS) 14
2.1 O QUE É UM CVS? 14
2.2 FUNCIONAMENTO DO CVS 14
2.3 ORGANIZAÇÃO DO HISTÓRICO 17
2.4 TRABALHANDO EM EQUIPE 17
2.5 FUSÃO E CONFLITO 20
3 SERVIDOR WEB 21
3.1 O QUE É UM SERVIDOR WEB? 21
3.2 FUNCIONAMENTO BÁSICO 21
3.3 TOMCAT 23
4 PROPOSTA DO LUPA ADMINISTRADOR 23
4.1 PROPOSTA 24
5 IMPLEMENTAÇÃO DO LUPA ADMINISTRADOR 29
5.1 POR QUE A FERRAMENTA SER WEB? 29
5.2 TECNOLOGIAS UTILIZADAS 29
5.3 PRIMEIROS ESTUDOS 30
5.4 FUNCIONAMENTO BÁSICO 30
5.5 O PROBLEMA 31
5.6 LUPA ADMINISTRADOR 32
5.6.1 Pagina inicial 32
5.6.2 Administrativo 32
5.6.2.1 Funcionamento básico do Administrativo 32
5.6.3 Rastreabilidade / Análise 33
5.6.3.1 Funcionamento 34
5.6.4 Tecnologias utilizadas 36
5.6.5 Desenvolvido por 36
5.7 IMAGENS DO LUPA ADMINISTRADOR 36
5.8 EXEMPLO PRÁTICO 37
5.9 DISPOSIÇÃO DO PROJETO 39
6 CONSIDERAÇÕES FINAIS E PROJEÇÕES FUTURAS 42
BIBLIOGRAFIA 43

1 INTRODUÇÃO

Observando os aspectos nas empresas de criação de softwares, percebe que as mesmas não conseguem ter um alto controle e/ou auditoria para administrar o desenvolvimento de seus produtos, no que se diz respeito na área de produção de código fonte.

Os códigos fontes às vezes são produzidos de forma irregular, isto implica em produção de baixa qualidade, afetando diretamente no desempenho do software.

Outro fator é que as empresas estão adotando novas formas de trabalho, dando a flexibilidade para os profissionais trabalharem de suas casas, conectados apenas pela Internet.

Certamente existe vantagem em trabalhar remotamente, pois as empresas têm maior redução de custos, e os funcionários ganham com qualidade de vida.

Considerando tais aspectos, percebe-se a necessidade de alguns artefatos que possibilite dar bases sólidas para as empresas gerenciarem a produção.

1.1 OBJETIVOS

O objetivo deste trabalho é demonstrar a ferramenta web Lupa Administrador, o seu funcionamento e como a mesma foi desenvolvida para que possa efetuar a rastreabilidade de código fonte, interagindo com o sistema de controle de versões (CVS).

1.2 JUSTIFICATIVAS E MOTIVAÇÃO

A motivação maior por este trabalho é que possa agregar significativamente a carreira profissional à percepção do desenvolvimento WEB, além de proporcionar uma pequena pesquisa científica, no sentido de aprimorar a qualidade no desenvolvimento de projetos de software, e propiciar aos futuros pesquisadores, a desfrutarem de tal ferramenta, dando continuidade a este trabalho.

1.3 ESTRUTURA DO TRABALHO

Os conceitos abortados abrangem enorme gama, considerando este motivo, para a melhor organização deste trabalho, ele está dividido em 6 Capítulos, sendo que o primeiro é esta introdução.

O Capítulo 2 apresenta o Sistema de controle de versões (CVS), relatando o que é um CVS bem como o seu funcionamento e organização estrutural de armazenamento de dados.

O Capítulo 3 apresenta o Servidor Web com suas características, funcionamento básico e a pequena explicação do servidor TomCat que será utilizado.

O Capítulo 4 apresenta a Proposta do trabalho, relatando a razão pela qual este trabalho veio a ser proposto, explicando de forma simples o problema levantado e a sua solução.

O Capítulo 5 apresenta a implementação da proposta, demonstrando o como a ferramenta foi construída, bem como o seu funcionamento básico.

Finalmente o Capítulo 6 contém as Considerações finais e projeções futuras, explicando quais foram os resultados obtidos com a construção da ferramenta e propondo pesquisas futuras que possam vir a agregar utilidades para a mesma.

2 SISTEMA DE CONTROLE DE VERSÕES (CVS)

2.1 O QUE É UM CVS?

Um Sistema de Controle de Versões, CVS (do inglês *control version system*), tem como finalidade gerenciar diferentes versões de um determinado documento, garantindo a organização, flexibilidade e rastreabilidade de todo o documento. Sua utilização tem se dado destaque em desenvolvimento de softwares, área que requer alta demanda quando se trata de controle de versões, possibilitando que vários programadores possam estar desenvolvendo o mesmo projeto em conjunto, ganhando tempo na produção do mesmo.

Segundo Caetano (2004, p. 14) "O CVS é uma ferramenta *open source* que implementa as principais funções pertinentes ao processo de controle de versões.".

2.2 FUNCIONAMENTO DO CVS

Primeiramente é necessária a instalação de um servidor de controle de versões CVS. Para este trabalho será utilizado o servidor CVSNT, pois suas características são de âmbito de software livre. Mas vale ressaltar que existem diversos tipos de servidores que podem ser utilizados, inclusive de versões proprietárias.

Após a instalação e configuração do servidor, dá se o segundo passo, criando um repositório, local que será armazenado todos os documentos que compõem o projeto. Tal repositório será gerenciado com todas as potencialidades da ferramenta CVS (no caso utilizado o servidor CVSNT), servindo as versões para os membros que compõem o grupo de desenvolvedores.

Os desenvolvedores devem também, instalar um pequeno software em suas máquinas com características de Cliente, gerenciando a transferência dos arquivos de modo Servidor-Cliente e Cliente-Servidor.

Para que cada desenvolvedor trabalhe no projeto é necessário que seja feito o *download* (baixar, pegar) os arquivos referentes ao projeto do servidor, que foi previamente criado no repositório, fazendo assim uma cópia local na máquina do desenvolvedor que fez o *download* do mesmo, garantindo a minimização de conflito.

A Figura 1 demonstra o modelo funcional:

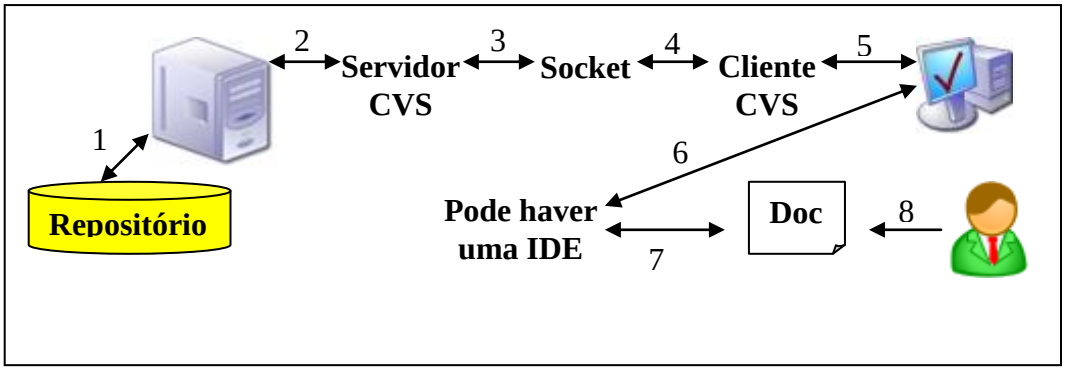

Figura 1. Modelo funcional.

Legenda:

1 – A máquina servidora manipula as informações do repositório no disco.
2 – O servidor CVS interage com o repositório.
3 e 4 – Pode haver um *socket* para a comunicação.
5 – Cliente CVS interage com o usuário.
6 e 7 – Pode haver uma IDE para a interação do documento com o usuário para facilitar a manipulação.
8 – Usuário trabalhando com o documento.

Desta forma, trabalhando-se com o projeto local, garante a minimização de conflitos no desenvolvimento, cada desenvolvedor terá a sua própria cópia, podendo efetuar todas as alterações que lhe achar necessárias.

Após a alteração ser feita, e concluir-se que aquela versão está estável, então ela pode ser enviada (*commit*) para o servidor, esta operação pode ser feita quantas vezes forem necessárias, gerando assim uma nova versão a cada envio. Aconselha-se a não enviar alterações com erros, podendo assim ser de extrema dificuldade para outros colaboradores alterarem ou dar continuidade ao trabalho, eles irão gastar muito tempo tentando achar os erros, ou mesmo, passarem por desapercebido e deixar todo o trabalho com falhas.

A Figura 2 demonstra o funcionamento lógico de envio para o servidor:

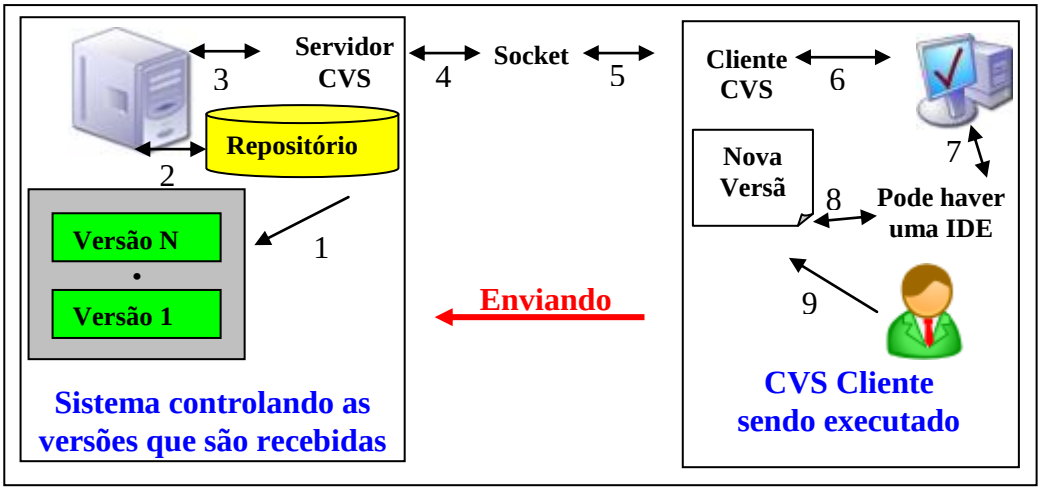

Figura 2. Modelo lógico.

Legenda:

1 – Versões existentes no repositório.
2 - Máquina servidora manipula as informações do repositório no disco.
3 - O servidor CVS interage com o repositório.
4 e 5 - Pode haver um *socket* para a comunicação.
6 – Cliente CVS interage com o usuário.
7 e 8 – Pode haver uma IDE para a interação do documento com o usuário para facilitar a manipulação.
9 – Usuário trabalhando com o documento.

Um servidor CVS pode controlar vários repositórios, é aconselhável utilizar um repositório para cada projeto para que o desenvolvedor não efetuar alterações indevidas em projetos distintos.

O sistema de controle de versões trabalha de forma inteligente no armazenamento das versões, não desperdiçando espaço, isso se deve por trabalhar da seguinte forma: é armazenado como um todo o projeto na primeira vez que é inserida ao repositório, nas demais versões que serão adicionadas, é apenas armazenado as suas diferenças entre a última versão, logo para que a última versão seja disponível ela é montada sobre todas as versões já existentes.

Exemplo abstrato: consideremos que para escrever a palavra CONTROLE DE PROGRAMA, utilizamos três versões, sendo que para escrever a palavra CONTROLE versão 1, DE versão 2 e PROGRAMA versão 3.

Para obter-se a versão 3 é necessária a concatenação de todas as versões, logo sendo: VF(versão final) e V(versão), dispondo matematicamente extrai se a função:

f(VF) = V1 + V2 + V3

onde:

f(V1) = CONTROLE

f(V2) = CONTROLE DE

f(V3 ou versão final VF) = CONTROLE DE PROGRAMA

A Figura 3 demonstra a organização das versões no repositório:

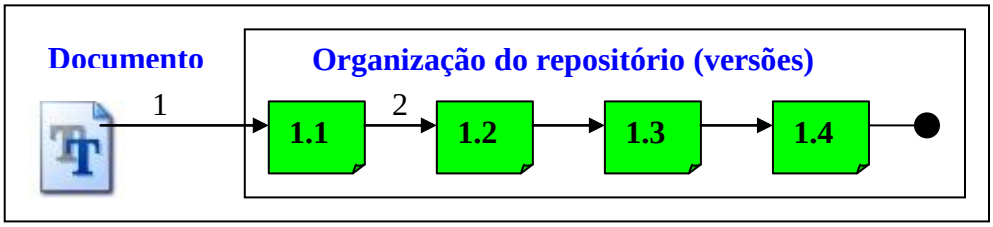

Figura 3. Organização das versões no repositório.

Legenda:

1 – Versão inicial do documento.
2 – Versões posteriores a inicial.

2.3 ORGANIZAÇÃO DO HISTÓRICO

A cada envio é registrado o *log* (identificação) do colaborador e a data de acesso, é obrigatório o comentário referente às alterações que foram efetuadas, facilitando assim a compreensão dos desenvolvedores que futuramente possam efetuando o *download* da nova atualização, dando se continuidade do projeto.

2.4 TRABALHANDO EM EQUIPE

A ferramenta CVS possibilita a permissão de trabalho simultâneo e assíncrono sobre o mesmo documento, isso é possível com seu mini sistema interno de controle de usuários, tornando o desenvolvimento de todo projeto com alta performance.

Exemplificando: dado um documento qualquer, vários colaboradores podem estar alterando o mesmo. O que realmente irá fazer a diferença será no envido das alterações para o servidor.

Considere: ***A*** e ***B*** colaboradores – ***A*** e ***B*** efetuaram o *download* do mesmo documento em suas máquinas e estão trabalhando localmente, dado tempo, ***A*** terminou suas alterações e envia a

nova versão para o servidor, o servidor aceita as alterações e armazena-o no repositório. Em seguida ***B*** envia as alterações, mas o servidor identifica que existe uma nova versão (a versão que ***A*** enviou), então é solicitado que ***B*** faça o *download* da nova versão antes de enviar as suas alterações, por sua vez, as alterações de ***B*** não devem ser perdidas, então o Cliente CVS de ***B***, salva o documento localmente em uma base de *backup*, e informa a ***B*** que será efetuada uma fusão das alterações vindas do servidor com as suas alterações, se ***B*** autorizar a fusão, então o cliente primeiramente verifica se existe algum conflito, se não houver, então a fusão é feita, senão é informado o conflito.

Observe as Figuras 4 e 5 para uma melhor compreensão da situação:

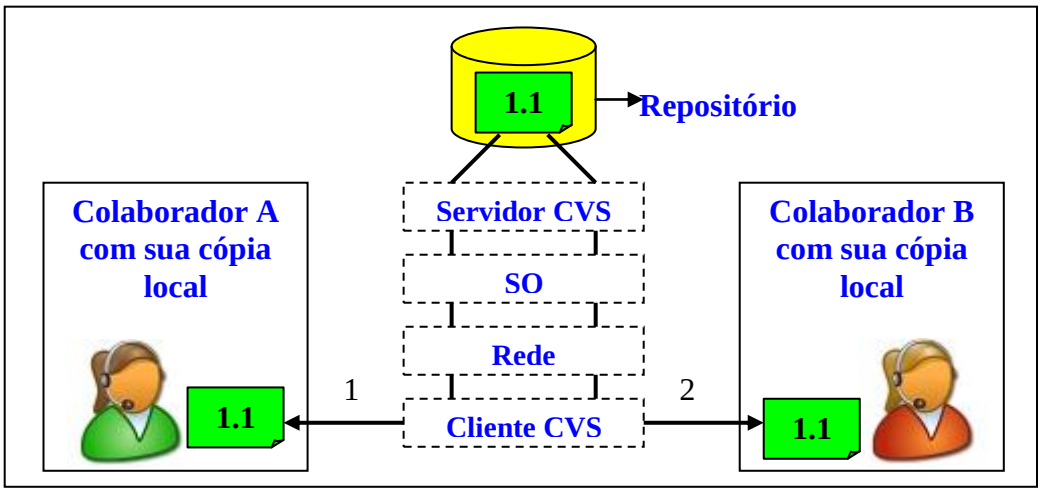

Figura 4. Os colaboradores efetuaram o download do mesmo documento do servidor.

Legenda:

1 e 2 – Colaboradores fazem o *download* do repositório.

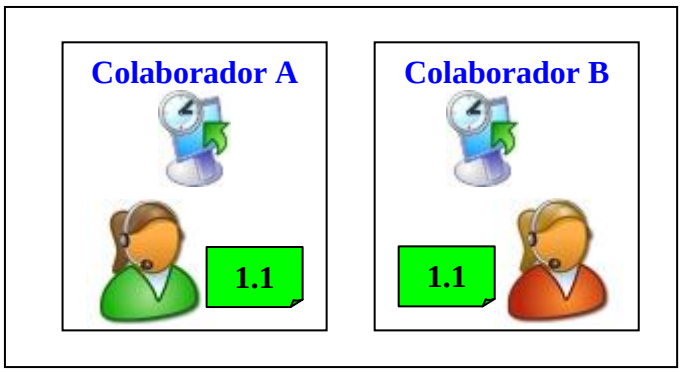

Figura 5. Trabalhando simultaneamente, mais em locais diferentes.

Colaborador ***A*** termina as suas alterações e envia a nova versão para o servidor. O servidor identifica que não existe nenhuma versão mais atualizada e aceita o envio, gerando a versão 1.2, logo a versão que o colaborador ***A*** tem localmente passa a ser a mesma do servidor.

Observe a Figura 6 para uma melhor compreensão da situação:

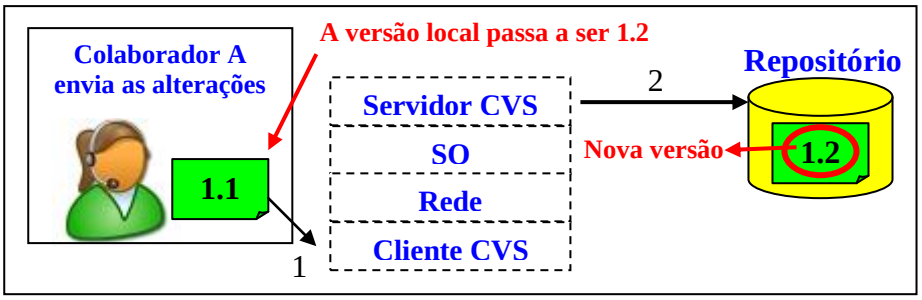

Figura 6. Envia as alterações, servidor aceita e gera uma nova versão.

Legenda:

1 - O colaborador ***A*** envia a sua versão para o repositório
2 - O repositório recebe a nova versão.

Colaborador ***B*** termina seu trabalho e envia a atualização para o servidor, mas o controlador de versões identifica que existe uma versão mais atualizada do que a de ***B***, então ***B*** é aconselhado efetuar o *download* da nova versão.

Observe a Figura 7 para uma melhor compreensão da situação:

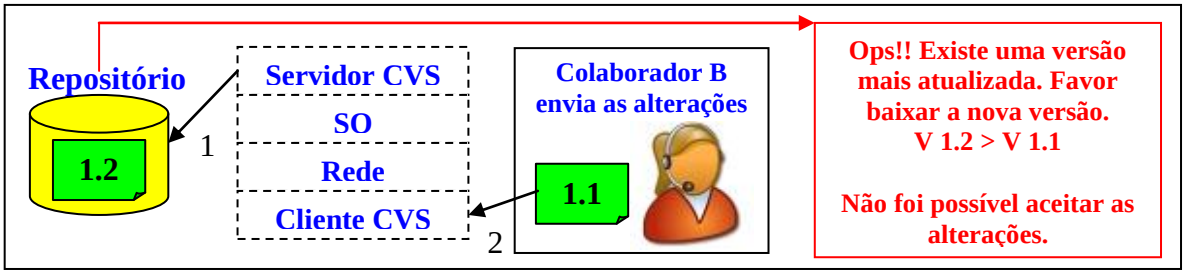

Figura 7. Identifica uma nova versão e informa ao colaborador.

Legenda:

1 – É repassado as atualizações para o repositório, mas o mesmo rejeita.
2 - O colaborador ***B*** envia a sua versão para o repositório

Após baixar a nova versão o próprio Cliente CVS faz uma cópia de *backup* (cópia de segurança) local nas alterações do desenvolvedor ***B***, verifica se existe algum conflito, se não houver, então é feita a fusão, e ***B*** passa agora a estar apto a enviar as suas alterações para o servidor CVS gerando uma nova versão.

Observe a Figura 8 para uma melhor compreensão da situação:

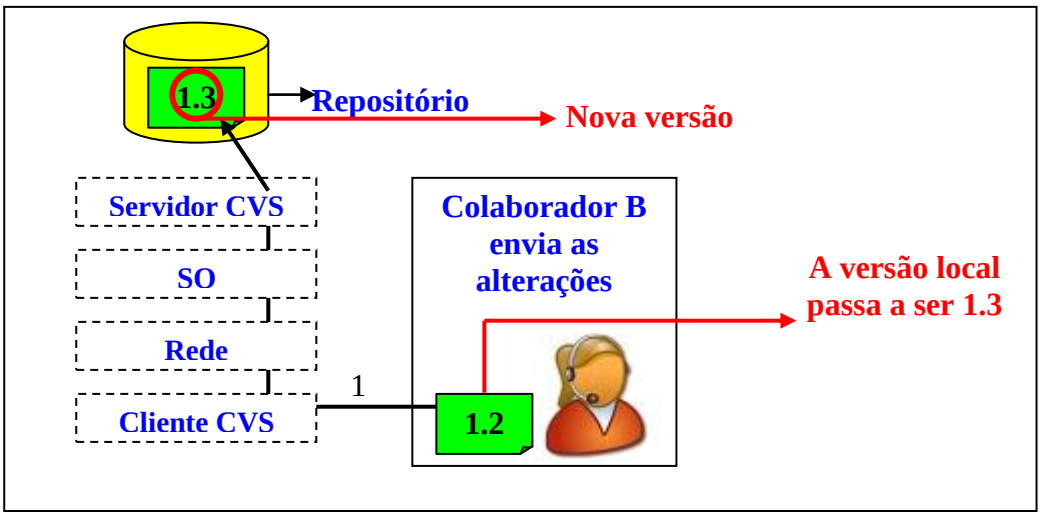

Figura 8. Colaborador B atualiza a versão local e reenvia a mesma para o servidor gerando uma nova versão.

Legenda:

1 – Repositório recebe a nova versão.

2.5 FUSÃO E CONFLITO

A fusão sempre ocorrerá no momento que dois ou mais indivíduos alterarem o mesmo documento e reenviam para o servidor, por sua vez o servidor identifica uma versão mais atualizada e solicita que o individuo faça o download desta versão. Com isso é feita a fusão do documento local com a do servidor.

Para tanto, pode haver conflitos no decorrer da fusão, pois considere que um documento possui 100 linhas, e os colaboradores que fizeram as alterações simultâneas, alteram, por exemplo, ambos a linha 50 do documento, logo se compreende que existe o conflito. Então a fusão deve ser feita manualmente.

3 SERVIDOR WEB

3.1 O QUE É UM SERVIDOR WEB?

É um software robusto capaz de atender requisições, processá-las e reenviá-las para o local de origem do pedido, fornecendo vários recursos de integração de negócios e/ou infra-estrutura, com qualidade para que todas as requisições sejam atendidas e disponibilizadas em um espaço de tempo relativamente pequeno, para tanto, conectar pessoas, processos e as demais informações ao meio.(IBM, 2008 – www.ibm.com.br).

3.2 FUNCIONAMENTO BÁSICO

Para que o Servidor Web seja acessado, é necessária a existência de infra-estrutura de máquina e rede, estando a todo o momento disponível para as solicitações, estabelecendo assim a relação Cliente-Servidor. É necessário também que um software Servidor Web seja instalado, e previamente configurado.

Do lado do Cliente deve existir instalado um navegador (programa que estabelece a relação Cliente-Servidor, gerindo comunicação) por meio de um protocolo bem definido como, por exemplo, SSH garantindo o envio e recebimento das informações.

Considerando que todos os pré-requisitos mencionados acima, sejam atendidos, procede-se da seguinte forma: o Cliente solicita a requisição, o Servidor localiza a operação que foi requerida, e repassa-a para tal, ao término é enviada a reposta para o Cliente.

Observe a Figura 9:

Figura 9. Cliente enviando requisição para o servidor por meio de um *browser*.

Legenda:

1 – Servidor comunica-ser através da rede (Internet).
2 – Cliente comunica-se através da rede (Internet).
3 – O *Browser* é o elo de comunicação com o usuário.
4 – Usuário interage com a ferramenta.

Após o estabelecimento da comunicação e envio da requisição, o servidor deve encontrar a operação requerida pela solicitação, deixando assim a mesma ser executada até ser completada e liberar o envio da resposta para o Cliente, e ao mesmo tempo estar atendendo as novas solicitações.

Observe as Figuras 10 e 11:

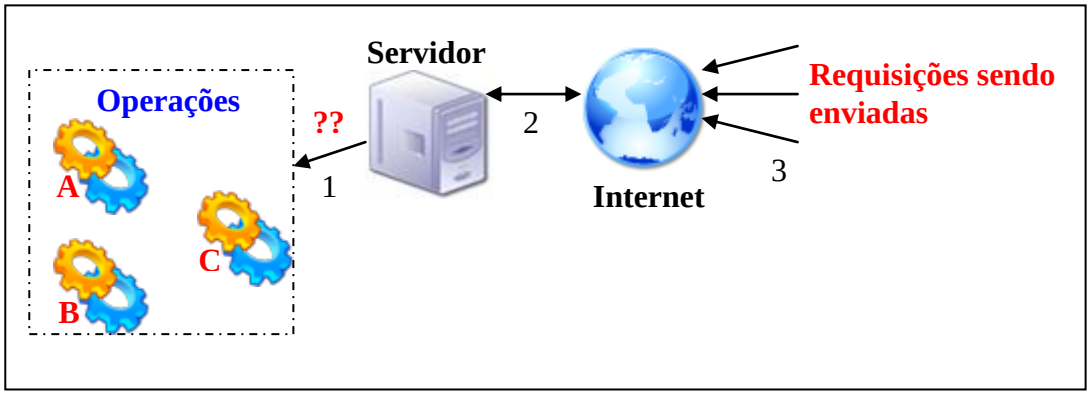

Figura 10. Clientes enviando solicitações – Servidor procurando operação correspondente da solicitação.

Legenda:

1 – Servidor procura qual operação deve ser executada.
2 – Servidor interage com a rede (Internet).
3 – Requisições são enviadas pela rede.

A operação ***C*** já executou e está sendo enviada para o Cliente que a solicitou, mas ao mesmo tempo o servidor continua recebendo requisições e está atendendo novos Clientes.

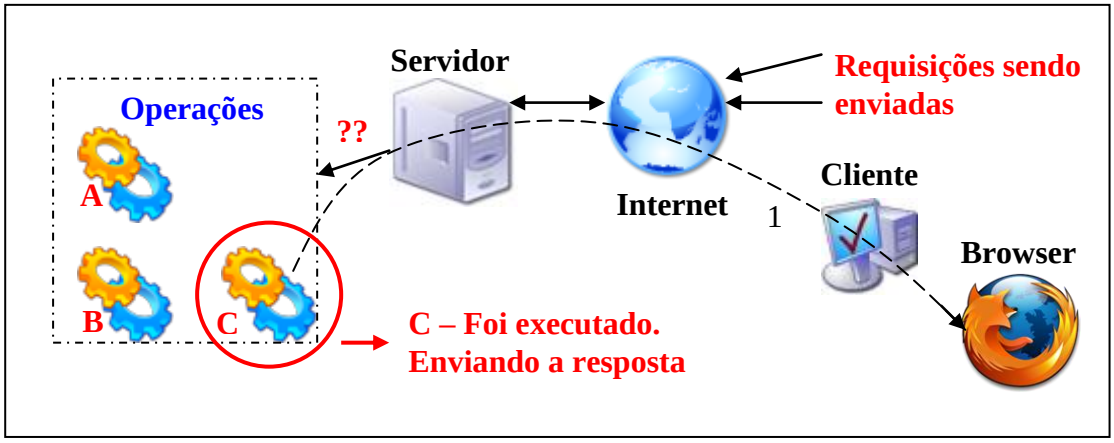

Figura 11. Enviando a resposta para o cliente.

Legenda:

1 – Envia a resposta para o cliente.

- Características do servidor web

  - ser robusto;

  - manipular múltiplos processos;

  - atender requisições e respondê-las;

  - efetuar autenticações (caso necessário).

- Alguns serviços disponíveis

  - paginas web;

  - correio eletrônico;

  - transferência de arquivos ou dados;

  - acesso remoto;

  - mensagens instantâneas.

3.3 TOMCAT

É um servidor de aplicações Java para web, totalmente gratuito, e para as finalidades deste trabalho será de grande importância na demonstração dos conceitos aplicados.

O surgimento do TomCat se deve a construção do projeto Apache Jakarta que foi apoiado pela Sun Microsystems, empresa na qual trabalha na implementação da tecnologias Java, no caso do TomCat se tratando de Java Servlet e Java Server Pages.(SUN MICROSYSTEMS, 2008)

Algo que chama atenção, ao trabalhar com este servidor, é que apesar de ser *open source*, ele possui uma grande estrutura, proporcionando eficiente desempenho na área de produção.

Quando acessada via interface Web, o TomCat disponibiliza suas diversas ferramentas administrativas para gerenciar os serviços que serão ou estão sendo executados, facilitando toda a movimentação, fluxo de dados e configurações possíveis para o emprego de sua utilização.

4 PROPOSTA DO LUPA ADMINISTRADOR

Com a necessidade de criar projetos extremamente grandes, várias empresas, sejam elas pequenas, médias ou grandes, encontram dificuldades para efetuar auditorias em seus produtos. Quando se trabalha em equipe, o processo de rastreabilidade de código fonte é algo que deve ser feito com a finalidade de se detectar erros ou ainda código mal intencionado. Partindo deste princípio, percebe-se a necessidade de se criar uma ferramenta, que auxilie tal processo, tornando o trabalho mais eficiente e seguro.

4.1 PROPOSTA

Na construção de um produto, os profissionais envolvidos, produzem o código fonte com o intuito de se chegar ao ponto alvo, que é o *deploy* (posicionar, distribuir) do software no cliente. Porém, pode haver no entremeio, idéias diferentes ou percepções sobre o projeto que não foram observadas no levantamento de requisitos e especificações, sendo assim, questionável a adição de novas utilidades para o produto, bem como melhorias, abrangido novos olhares, para a lucratividade sobre tal produto. Em contra partida, não se deve desviar a continuidade de um projeto, sem saber, se tais idéias podem realmente levar ao êxito.

O que fazer em situações como esta? Utilizando-se de um sistema de controle de versões concorrentes pode-se tornar possível a programação paralela do código fonte com a nova perspectiva disponibilizando outra equipe para tais fins, sem que se altere a continuidade do projeto original que seguirá normalmente com a equipe que esta totalmente envolvida desde o início.

Para exemplificar tal situação, consideremos que um software qualquer, em sua construção, terá um ciclo de desenvolvimento de 6 meses para a sua conclusão, e a cada mês será gerado uma nova versão minimamente estável. Estas versões serão todas armazenadas em um repositório controlado por um sistema de controle de versões (CVS).

Observe a Figura 12 após o primeiro mês de trabalho no código fonte, a equipe submete a primeira versão do produto para o repositório, criando assim a versão 1.1:

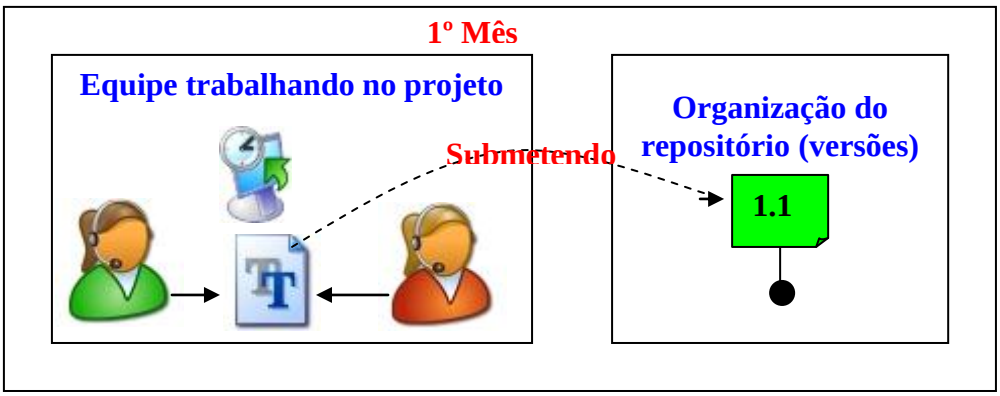

Figura 12. Criando a versão 1.1.

No segundo mês a equipe recupera do repositório a versão 1.1, procedendo assim à continuidade do desenvolvimento do código fonte, concluindo-se que as novas atualizações foram satisfatórias, submete-se o código fonte para o repositório, criando-se a versão 1.2 do projeto.

Observe a Figura 13:

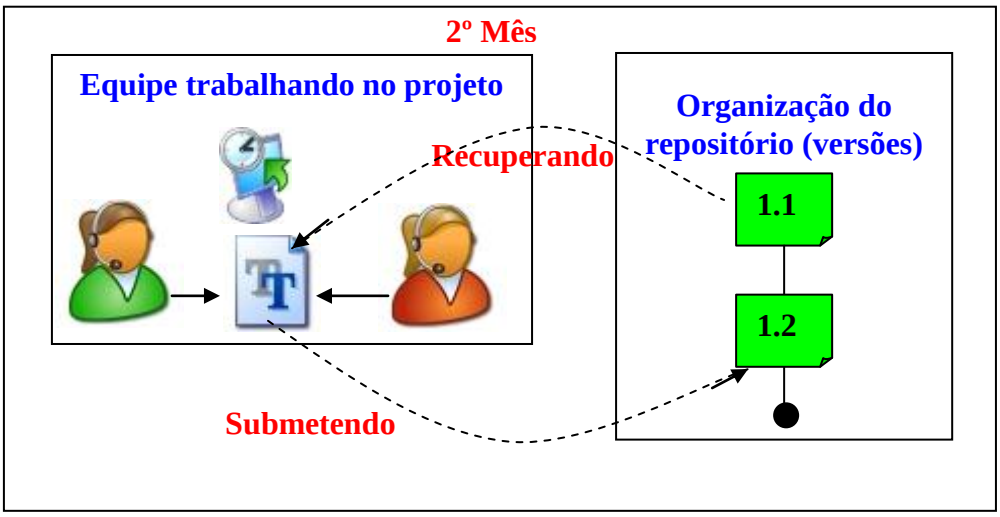

Figura 13. Criando a versão 1.2.

Seguindo o mesmo raciocínio, conclui-se que os demais meses que faltam para o término do clico de desenvolvimento sejam igual ao processo anterior, obtém-se o quinto mês, a seguinte Figura 14:

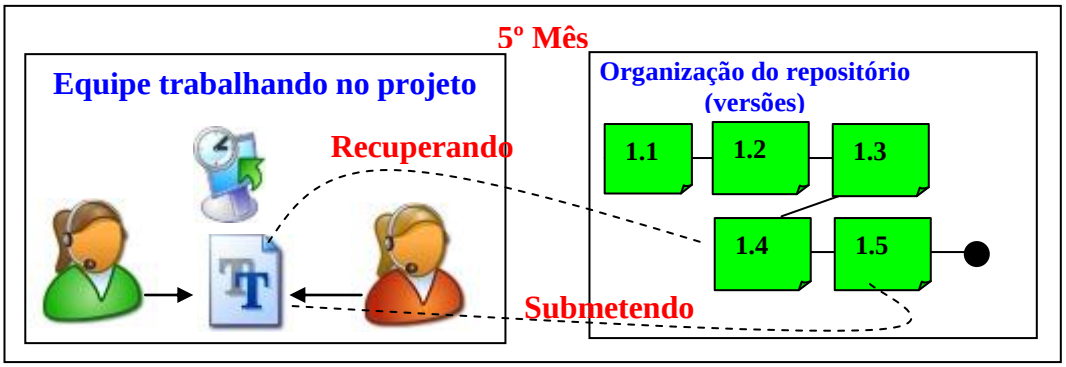

Figura 14. Quinto mês de desenvolvimento obtém-se a versão 1.5.

Após cinco meses de desenvolvimento, a equipe ou mesmo o gerente de projeto percebe que pode ser criada nova funcionalidade para o produto, porém falta apenas um mês para a sua finalização, e não se deve tomar outro caminho no desenvolvimento.

Observando a situação acima, e considerando que a equipe já usufrui de um sistema de controle de versões, conclui-se, para a resolução de tal tarefa é extremamente fácil, pois, basta criar outra equipe de desenvolvimento que trabalhará no desvio do projeto a partir da versão 1.5 enquanto a equipe atual termina o produto iniciado.

Como a nova equipe criada estará partindo da versão 1.5, e também, a equipe inicial estará trabalhando com a mesma versão, não haverá conflitos dos códigos, o CVS, com apenas um comando se encarregará de criar uma cópia da versão 1.5 para a segunda equipe, deste modo o desenvolvimento não será afetada em ambas as partes. Observe a Figura 15:

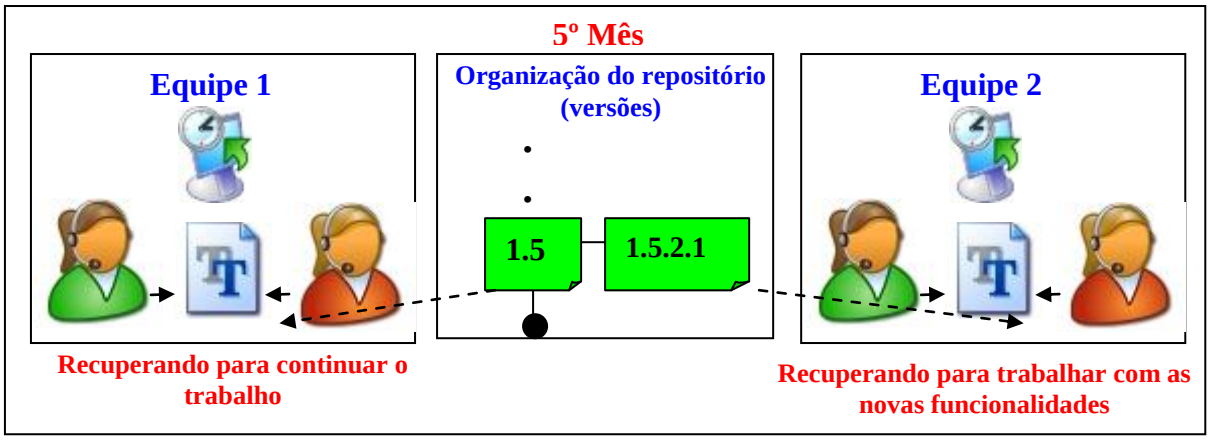

Figura 15. Trabalhando paralelamente.

A construção do produto seguirá adiante com as duas equipes trabalhando paralelamente no mesmo projeto, com a peculiaridade de perspectivas diferentes para a agregação de funcionalidades do produto.

Designando-se do contexto acima, a imagem correspondente após um mês decorrido, compreende-se com a representação da Figura 16:

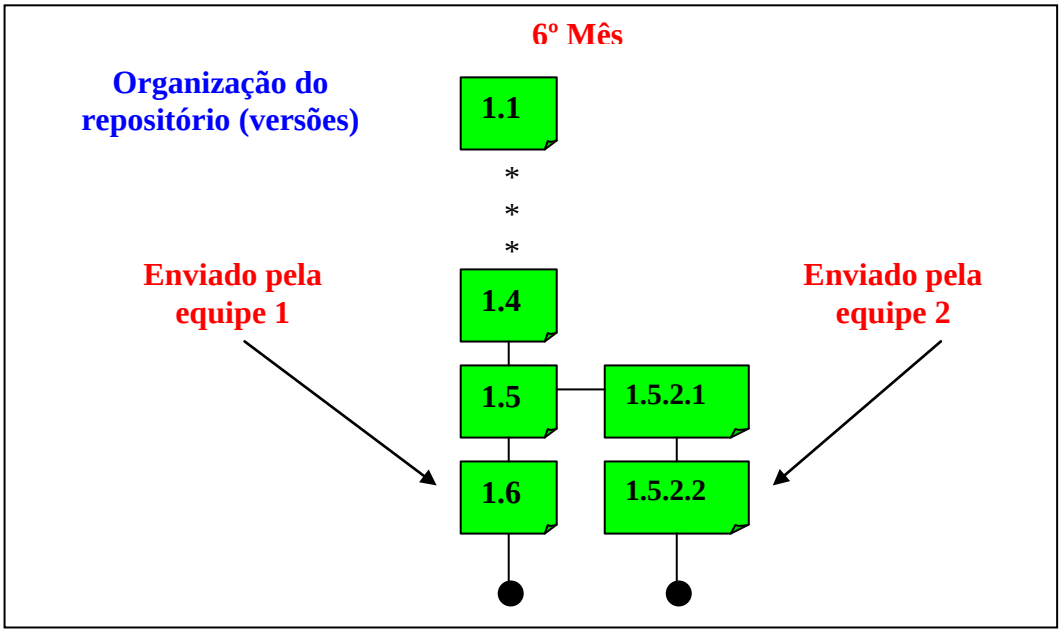

Figura 16. Sexto mês – as duas equipes submeteram seus códigos ao repositório, gerando as versões.

Caso subentenda que a construção do código fonte desenvolvida pela equipe 2 esteja realmente satisfatório, a empresa pode torná-lo novo produto e/ou versão, ou até mesmo uma fusão entre o código fonte original com a nova perspectiva, sendo um produto único.

Abordando tal situação, percebe-se que se houvesse determinada ferramenta que proporcionasse a auditoria sobre o código fonte de um projeto qualquer, tornar-se-ia possível a administração e controle do que realmente foi feito e está sendo produzido na concepção do código paralelo ao código original.

Tomando como base o exemplo citado acima, a ferramenta Lupa Administrador é voltada para efetuar auditorias em códigos fontes em qualquer lugar do globo, proporcionando flexibilidade no que se diz respeito ao ciclo de desenvolvimento, de forma que, a ferramenta interaja com o repositório que esteja sobre a gerência do CVS.

Como funcionalidade a ferramenta efetuará a rastreabilidade e distinção sobre os códigos fontes em qualquer ciclo de desenvolvimento ou versão, expondo de forma objetiva as informações para que se possa iniciar uma futura análise pela gerencia de projeto.

Para uma melhor explicação observe a Figura 17:

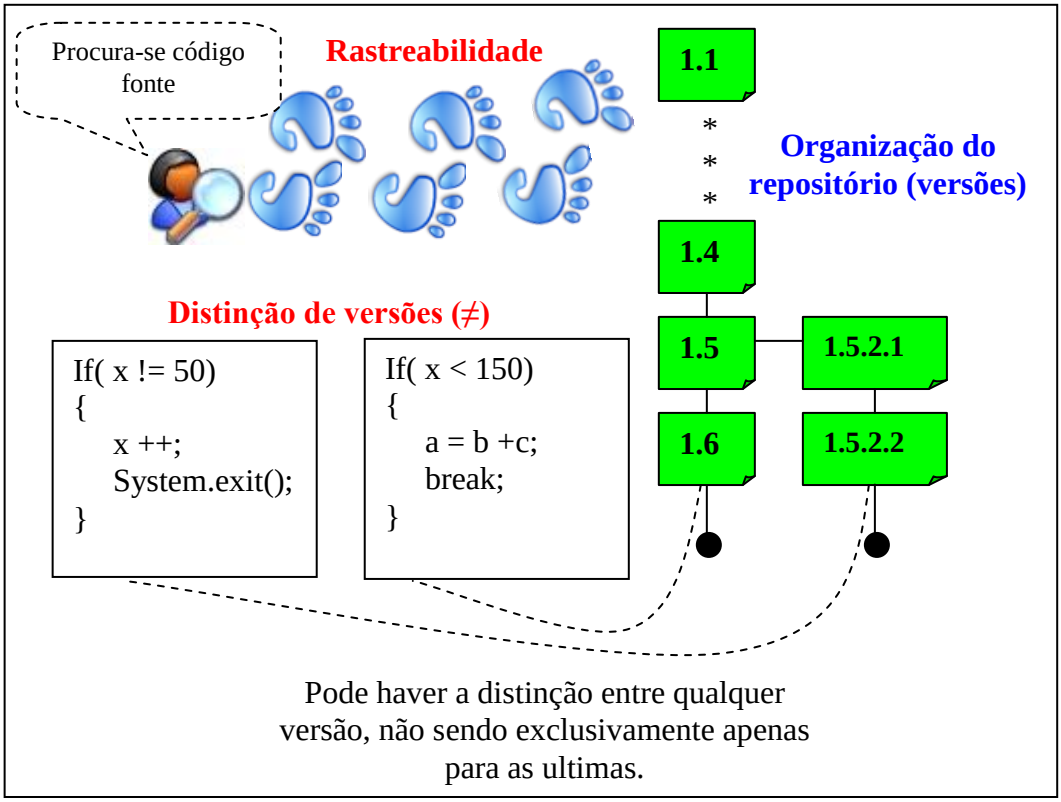

Figura 17. Rastreabilidade de código fonte e distinção entre versões.

Segundo os padrões internacionais ISO 8402, "rastreabilidade é definida como a habilidade de descrever a história, aplicação, processos ou eventos e localização de um produto, a uma determinada organização, por meios de registros e identificação."

Nota: Vale ressaltar que o exemplo do software com ciclo de 6 meses apresentado acima foi meramente explicativo. Tal conceito pode ser aplicado em qualquer fase de desenvolvimento, não se aplicando apenas para os últimos meses.

5 IMPLEMENTAÇÃO DO LUPA ADMINISTRADOR

Com o estudo dos componentes utilizados, bem como as soluções possíveis para consolidar a proposta desse trabalho, neste momento inicia-se a explicação e métodos das funcionalidades da ferramenta proposta.

5.1 POR QUE A FERRAMENTA SER WEB?

Ao observar o mercado de trabalho, percebe-se a existência de uma grande procura por profissionais que tenham conhecimento suficiente para os desenvolvimentos de trabalhos que sejam voltados para Web e/ou aplicações distribuídas. Para as grandes organizações, compreende-se a necessidade de ter seus dados disponibilizados a todo o momento, isso está diretamente relacionado com a sua lucratividade, já que os sistemas que trabalham localmente não são capazes de dar flexibilidade total de acesso. As informações devem ser manipuladas, não importa a sua localização no globo, proporcionando crescimento e rentabilidade para as organizações.

5.2 TECNOLOGIAS UTILIZADAS

Com o crescimento da utilização da tecnologia Java no Brasil e também nos demais países, considerando que o Java tem sua grande comunidade de desenvolvedores localizados em nosso território, optou-se por criar tal ferramenta usufruindo desta tecnologia, pois caso haja dúvidas sobre a sua utilização, será de fácil a obtenção de ajuda.

Foram utilizadas a seguintes tecnologias derivadas do Java, bem como *Design Patterns*:

- *Java Server Faces – FrameWork* que possibilita agilizar o desenvolvimento Web.
- *Java Server Page* – Possibilita a adição de código Java dentro de paginas HTML, tornando páginas Web dinâmicas (SUN MICROSYSTEMS, 2008 – www.sun.com).
- *Servlet* – Classes Java que trata das requisições e respostas que são geradas entre Cliente e Servidor.

- *Java Beans* – são componentes de softwares reutilizáveis que pode se desenvolver e reunir facilmente para criar aplicações sofisticadas (SUN MICROSYSTEMS, 2008 – www.sun.com)
- MVC *(Model View Control)* – *Design Pattern* que trata de forma inteligente as informações, separando as mesmas de forma clara e objetiva, distinguindo quais são as partes de modelo, visão e controle.
- Ajax4Jsf – *FrameWork* que trata de forma assíncrona os dados que são gerados tanto pelo Cliente como o Servidor, economizando o tráfego de dados pela rede, tornando a disposição dos dados com alta eficiência.

O objetivo deste trabalho não é a explicação a fundo de tais tecnologias utilizadas, mas sim a união das mesmas que foram capazes de compor a ferramenta correlata.

Para que futuros pesquisados, caso haja interesse pessoal é recomendado que estudem as tecnologias citadas acima para dar continuidade nesta linha de pesquisa.

5.3 PRIMEIROS ESTUDOS

Com o problema formulado e traçado o objetivo da construção da ferramenta, parte-se para a análise, surgindo várias perguntas: Como proceder? Por onde começar?
Para responder tais perguntas, deu-se início ao estudo de cada componente utilizado, bem como o software CVS que é o ícone da aplicação.

5.4 FUNCIONAMENTO BÁSICO

A ferramenta quando inicializada deve carregar todos os arquivos que estão dispostos no repositório para o *browser*. O usuário seleciona um dado arquivo para efetuar a análise e/ou rastreabilidade, automaticamente a ferramenta se encarrega de buscar quais versões estão relacionadas com o arquivo especificado. Desta forma é possível que seja apontada quais versões serão analisadas, e finalmente pode ser submetido o comando de execução, fazendo com que a ferramenta passe os comandos e retorne as saídas de análises obtidas, conforme apresentado na Figura 18.

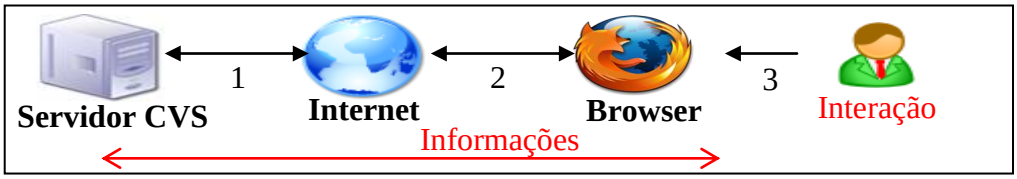

Figura 18. Funcionamento básico da ferramenta.

Legenda:

1 – Servidor comunica-ser através da rede (Internet).
2 – O Browser comunica-se com a rede (Internet).
3 – Usuário interage com a ferramenta.

5.5 O PROBLEMA

A construção da ferramenta, por mais complicada que seja, ao trabalhar com vários componentes, deparou-se com a seguinte questão: Como fazer um programa externo (CVS) receber instruções de outro programa feito em Java e ainda coletar informações?

Após várias semanas de estudo, obteve-se apenas uma forma de resolver tal situação, concluindo-se então o seguinte caminho:

- O CVS é capaz de receber informações através do *Shell* (programa de linha de comando que estabelece comunicação entre o sistema operacional e dado programa requerido);
- A linguagem Java é capaz de criar arquivos de lote e alocar processos que são delegados para o sistema operacional e recuperar as informações de saída.

Logo, observou-se a oportunidade de criar pequenos arquivos de comunicação com o *Shell* e ordenar que o Java delega-se os arquivos para o mesmo em forma de processo, deste modo criou-se à primeira comunicação entre dois programas externos, e o problema foi solucionado.

Obs: Trabalhando com arquivos de lote, não é possível que haja a comunicação de vários usuários, pois o SO tentará ler o arquivo de configuração que já está sendo utilizado, e o mesmo lançará uma exceção, pois múltiplos usuários estão requerendo a leitura do mesmo arquivo. A ferramenta não está no momento tratando leituras de arquivo, para tanto existe tal deficiência, deste modo à mesma não esta pronta para a utilização em larga escala, gerando uma nova linha de pesquisa para a resolução de tal problema.

5.6 LUPA ADMINISTRADOR

Lupa Administrador é o nome aderido pela ferramenta, tal nome se deve a característica de procurar, observar e/ou rastrear determinada versão de código fonte que tenha a mesma alocada no repositório que o CVS administre.
O Lupa Administrador compreende-se em:

- Pagina inicial;
- Administrativo;
- Rastreabilidade / Análise;
- Tecnologias utilizadas;
- Desenvolvido por.

5.6.1 Pagina inicial

A página inicial tem por objetivo disponibilizar o acesso aos menus, bem como dar uma breve explicação da necessidade de tal ferramenta ter tais características, quais as suas necessidades e o objetivo esperado.

5.6.2 Administrativo

Para que a ferramenta possa funcionar corretamente, deve-se haver a pré-configuração, sendo de obrigatoriedade o preenchimento de três campos:

- Repositório: diretório do repositório que o CVS esteja administrando;
- Nome da pasta do projeto: nome que indica qual pasta a ferramenta deve trabalhar, pois em um mesmo repositório pode haver vários projetos que o CVS administre;
- Variável de ambiente: nome da variável de ambiente que foi previamente configurada na instalação do CVS, de suma importância, pois a mesma é responsável por fazer com que a ferramenta possa trocar comandos com o CVS.

5.6.2.1 Funcionamento básico do Administrativo

Com os campos obrigatórios preenchidos, e pressionando o botão **Gravar Configuração,** a ferramenta coleta as informações do *browser*, e inicia-se o processo de geração do arquivo de configuração, tal arquivo esta localizado por definição no diretório: c:\LupaTCC\conf\.

Ao termino da geração do arquivo, a ferramenta faz a cópia dos arquivos que estão no repositório informando para o diretório c:\LupaTCC\coptrab (cópia de trabalho), isto para que não haja nenhuma alteração nos arquivos e possíveis erros que possam vir a acontecer por motivos inesperados, garantido total conformidade das informações dentro do repositório. Outro motivo se deve as propriedades do CVS que não admitem que determinados comandos administrativos possam ser executados dentro do repositório, obrigando assim que seja criada a cópia de trabalho com os mesmo arquivos pertencentes ao repositório.

Para a melhor compreensão veja a Figura 19:

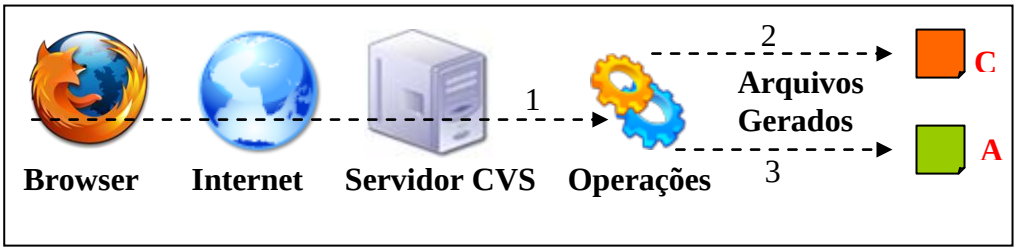

Figura 19. Gerando arquivos de configuração.

Legenda:

1 – Servidor recebe comandos, e delega para as operações correspondentes.
2 e 3 – Gerando arquivos.

5.6.3 Rastreabilidade / Análise

Após a pré-configuração da ferramenta, a mesma está apta a efetuar futuras análises sobre as versões alocadas na cópia de trabalho, para tanto a opção **Rastreabilidade / Análise** tem por finalidade disponibilizar á ação de toda a proposta de trabalho que foi relatada nos capítulos acima.

Ao clicar na opção **Rastreabilidade / Análise** o *browser* redirecionará para a página de análise. A mesma é composta pelo layout de:

- Três *comboBox*:
  - Arquivo: armazena todos os arquivos encontrados dentro do repositório especificado;
  - Revisão 1: exibe as versões existentes para dado arquivo selecionado;

- Revisão 2: idem a Revisão 1;

- Três *textArea*:

  - Revisão 1 – arquivo: exibe o código fonte do arquivo selecionado no primeiro *comboBox* (**Arquivo**) referente a **Revisão 1**;

  - Revisão 2 – arquivo: exibe o código fonte do arquivo selecionado no primeiro *comboBox*(**Arquivo)** referente a **Revisão 2;**

  - Diferença: exibe a diferença encontra pela análise entre as **Revisão 1** e **Revisão 2**.

5.6.3.1 Funcionamento

Ao carregar a pagina de **Rastreabilidade / Análise,** a ferramenta se encarrega de efetuar a varredura sobre os arquivos que estão dentro do diretório de trabalho (c:\LupaTCC\coptrab) e carregar os mesmo para o primeiro *comboBox* intitulado pelo nome **Arquivo** da pagina exibida.

Quando selecionado determinado arquivo do *comboBox* **Arquivo**, automaticamente a ferramenta envia uma requisição via Ajax para o servidor requerendo que o mesmo faça uma segunda busca, para encontrar quais versões estão disponíveis para o arquivo selecionado, ao encontrá-las o servidor envia os dados referentes as versões encontradas para o *browser* para que o segundo e terceiro *comboBox* sejam preenchidos com as versões encontradas.

Tais requisições foram tratadas via Ajax para garantir que a rede não fique com gargalos de dados, enviando apenas a carga útil entre servidor e cliente.

Para a melhor compreensão observe a Figura 20:

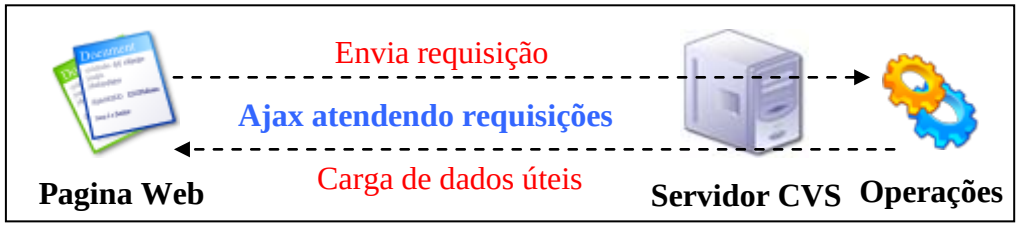

Figura 20. Ajax manipulando dados.

Ao pressionar o botão **Análise** a ferramenta, via Ajax, envia para o servidor os dados que foram selecionados nos *comboBox's*, contendo o nome do arquivo, revisão 1 e a revisão 2, o servidor verifica que deve ser executado a análise, então o mesmo gera o arquivo de lote (Shell) com as informações recebidas.

Tal arquivo de lote para ser gerado, primeiramente dever coletar as informações que estão dentro do arquivo que foi criado na primeira etapa do menu **Administrador** onde se encontra dentro do diretório c:\LupaTcc\conf, e associá-los com os dados recebidos do *browser*, gerando um arquivo único que será alocado no diretório c:\LupaTcc\execute. Tomando como um exemplo real, internamente neste arquivo obtém-se o seguinte conteúdo:

Linhas	**Comandos**
1	cd C:\LupaTCC\coptrab
2	set cvsroot=Variável_de_Ambiente
3	cvs diff –rRevisao1 –rRevisao2 Nome_Pasta_Projeto\Arquivo

Tabela 1 . Comandos existentes dentro do arquivo de configuração.

Linha 1: mostra para o CVS em qual diretório os comandos deveram ser submetidos.

Linha 2: especifica para o CVS qual é a variável de ambiente que foi configurada na instalação do servidor.

Linha 3: comando CVS que encontra as diferenças entre as revisões especificadas juntamente com o arquivo.

Com o arquivo formulado a linguagem Java repassado o mesmo para o *Shell* do sistema operacional, o SO efetua a leitura do arquivo e o interpreta cada linha em forma de processo para que os comandos contidos dentro do arquivo sejam delegados para o CVS, por sua vez o CVS recebe os comandos e os executa, ao termino, é devolvida a informação de saída para o sistema operacional, então a linguagem Java coleta as informações do SO, modelando-as para que possam ser exibidas de forma clara, e repassa as mesmas para o navegador via Ajax.

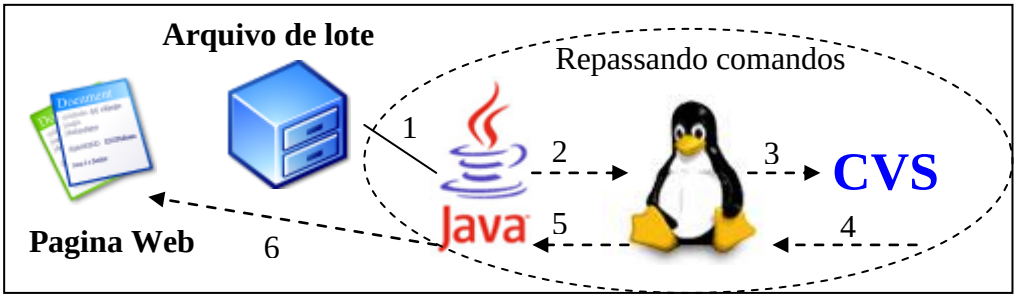

Figura 21. Processamento interno de comandos.

Legenda:

1 – Gera o arquivo de lote.
2 – Repassa o arquivo de lote para o SO.
3 – CVS recebe comandos do SO.
4 – O SO recebe a saída do CVS.

5 – O Java coleta as informações do SO com threads.
6 – As informações já modeladas são repassadas para a página web.

5.6.4 Tecnologias utilizadas

Relata quais tecnologias, *Design Patterns*, *FrameWorks* foram utilizados e suas referencias, em quais locais encontrá-las e breve explicação das mesmas.

5.6.5 Desenvolvido por

Descreve por quem a ferramenta foi desenvolvida, os dados pessoais, quais as dificuldades encontradas e a satisfação obtida em desenvolver tal trabalho de pesquisa. Propondo também novas idéias para o melhoramento da ferramenta.

5.7 IMAGENS DO LUPA ADMINISTRADOR

Para a melhor compreensão da ferramenta observe atentamente as imagens a seguir do Lupa Administrador em seu pleno funcionamento.

A Figura 22 exibe a página inicial que dá acesso às opções da ferramenta.

Figura 22. Página inicial.

A Figura 13 demonstra a área Administrativo para a pré-configuração da ferramenta, a mesma é necessária para que se possa efetuar as posteriores análises.

Figura 23. Administrativo.

Neste momento a Figura 24 exibe a funcionalidade de Rastreabilidade / Análise que ocorre entre a interação da ferramenta com o repositório CVS.

Figura 24. Página Rastreabilidade / Analise.

5.8 EXEMPLO PRÁTICO

As imagens a seguir demonstram um exemplo prático do funcionamento da ferramenta, efetuando a análise entre duas versões encontradas dentro do repositório com o seu respectivo arquivo de código fonte que está sob a gerência do CVS.

Para a melhor compreensão a página de Rastreabilidade / Análise está divida em 4 figuras para a explicação:

A Figura 25 exibe o arquivo e as revisões (versões) que foram indicadas para este exemplo:

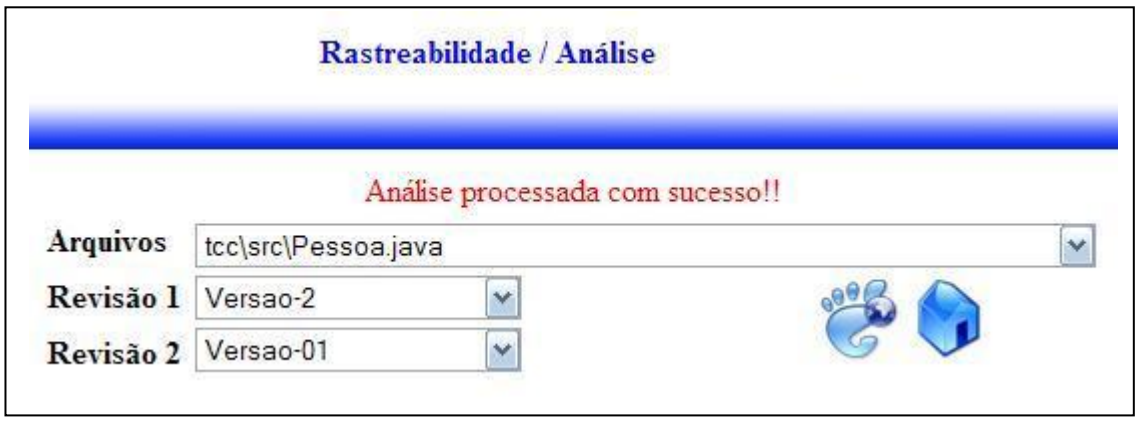

Figura 25. Arquivo e revisões selecionados.

Depois de acionado a análise, encontra-se como saída para a Revisão 1 a Figura 26 exibindo o código fonte encontrado:

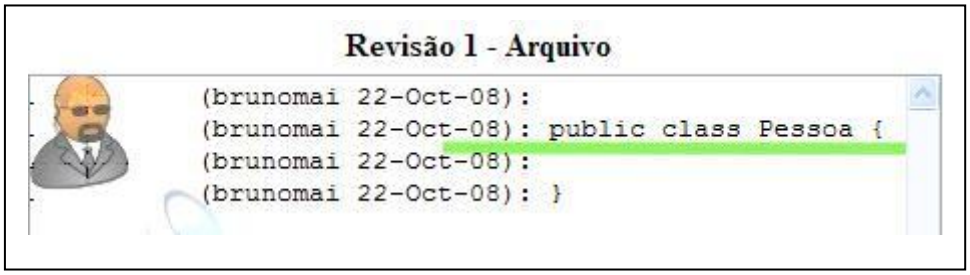

Figura 26. Revisão 1 – Arquivo.

De mesmo modo a Figura 27 demonstra a saída do código fonte encontrado referente à Revisão 2:

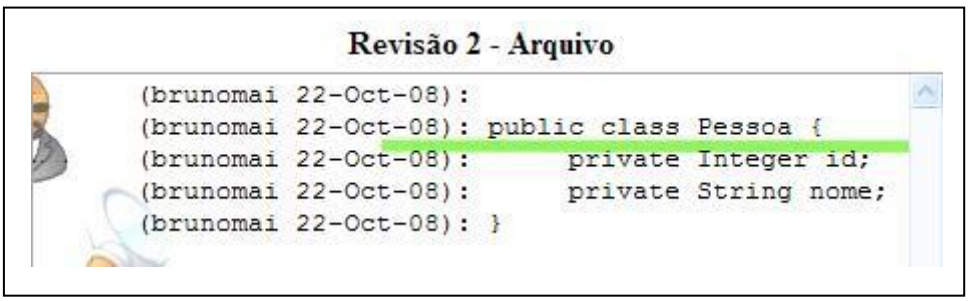

Figura 27. Revisão 2 – Arquivo.

Sem mais, para finalizar é exibido a diferença entre a Revisão 1 e Revisão 2. Observe a Figura 28, a mesma é auto-explicativa.

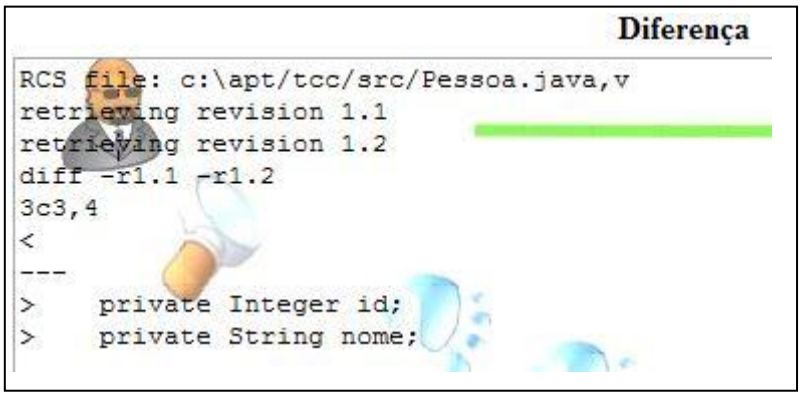

Figura 28. Diferença.

5.9 DISPOSIÇÃO DO PROJETO

Na codificação da ferramenta, foram adotadas algumas praticas para a melhor organização de seu desenvolvimento, de modo que a Figura 29 demonstra a disposição do projeto.

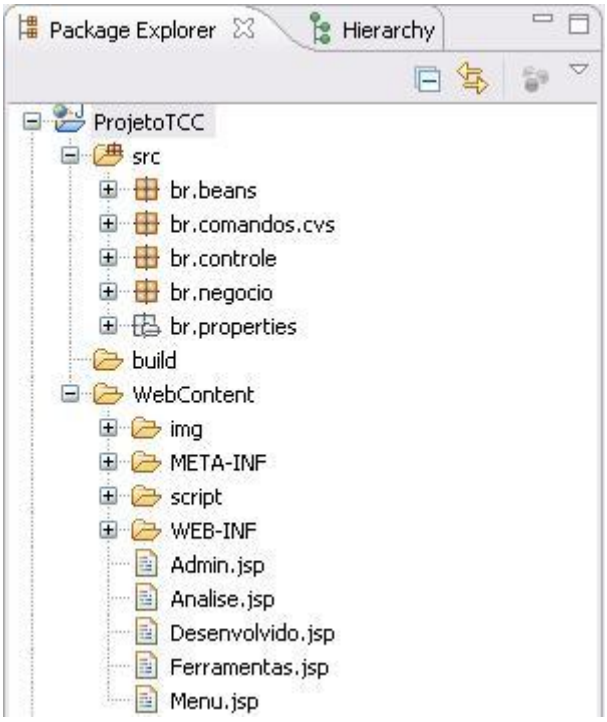

Figura 29. Disposição do projeto.

Observe a pasta **src**, a mesma é responsável por armazenar todos os pacotes referentes à aplicação, estando distribuídos da seguinte forma:

- br.beans – contém todas as classes beans utilizadas pela ferramenta em sua manipulação de objetos;
- br.comandos.cvs – este pacote possui todas as classes com os comandos que podem ser passados para o CVS;
- br.controle – classes que controlam o fluxo que esta vindo do navegador, delegando ações para as classes de negócios;
- br.negocio – neste ponto existem as classes que irão efetuar toda a parte de negócio da aplicação;
- br.properties – contém o arquivo de internacionalização.

A pasta build é responsável por armazenar todas as classes compiladas que serão utilizadas no momento de execução da ferramenta.

Os demais arquivos e pastas são referentes à parte web, onde o usuário estará interagindo com a ferramenta.

Observe o pequeno trecho do método da classe Differenca.java que é responsável por delegar para o SO as operações de diff.

```
public String diff(String str) {
	Process prcSistema;
	String fonte = "";
	try {
		prcSistema = Runtime.getRuntime().exec(str);
		BufferedReader buffer = new BufferedReader(new
		InputStreamReader(prcSistema.getInputStream()));
		fonte = format(buffer);
	} catch (IOException e) {
		// TODO Auto-generated catch block
		e.printStackTrace();
	}
	return fonte;
}
```

Perceba que o método diff recebe como argumento uma string que é referente ao local de onde está o arquivo de lote criado anteriormente pela ferramenta, de tal forma que o mesmo será executado pela classe Runtime, que delega para o SO o arquivo de lote para ser processado pelo CVS. Todas as comunicações que ocorrem com o SO seguem a mesma nomenclatura citada acima.

Depois de processado pelo CVS o SO devolve as informações obtidas para o método diff que por sua vez é passado para o método format. Tal método é responsável por tratar as informações para que as mesmas sejam disponibilizadas de forma clara e concisa. Observe o código abaixo:

```
private static String format(BufferedReader buffer) {
	String linha = "";
	String fonte = "";
	try {
		while ((linha = buffer.readLine()) != null) {
			if (linha.indexOf("=") == 0) {
				break;
			}
		}
		linha = "";
		while ((linha = buffer.readLine()) != null) {
			fonte = fonte + linha + "\n";
		}
	} catch (IOException e) {
		// TODO Auto-generated catch block
		e.printStackTrace();
	}
	return fonte;
}
```

6 CONSIDERAÇÕES FINAIS E PROJEÇÕES FUTURAS

Ao se trabalhar com o software CVS percebe-se quão e tão o mesmo é fantástico por proporcionar a manipulação de código fonte de forma inteligente. Existem algumas discussões indicando que certas versões proprietárias não são tão eficientes quanto ao CVS devido a sua forma de manipular o código fonte no repositório.

Voltando para a concepção da ferramenta construída, obteve-se o resultado superior do que era esperado quando se confirmou à proposta de trabalho, a mesma neste momento é capaz de realizar análises de código fonte de projetos em desenvolvimento.

Obteve-se o enorme agregamento de conhecimento sobre todo o material de estudo, bem como grande satisfação em realizar a linha de pesquisa e a partir da mesma concluir a construção da pequena ferramenta.

Compreende-se que é necessário o melhoramento da mesma para que seja utilizada em larga escala, mas o objeto deste trabalho não é criar tal ferramenta por completo, isto foi apenas uma pequena pesquisa que abriu o leque para futuros pesquisadores, demonstrando que existe a grande necessidade de ferramentas para o meio computacional.

Para que a ferramenta obtenha a melhor performance, é indicado que futuros pesquisadores estudem meios de ligar a ferramenta com bancos de dados, desta forma a mesma pode diminuir expressivamente o número de processamentos de hardware e disponibilizar os dados de forma mais ágil, bem como encontra a resolução do problema citado sobre a leitura múltipla de arquivos de lote.

Na construção, observou-se que para o desenvolvimento de qualquer pesquisa, não importando o seu tamanho, o mesmo requer grandes estudos para se levar à obtenção do êxito. Todo o estudo empregado, mesmo que às vezes pareça inútil, deve ser considerado uma grande descoberta, pois ao imaginar que outra pessoa estivesse fazendo o mesmo trabalho, quanto tempo à mesma levaria para se chegar ao mesmo ponto alcançado.

BIBLIOGRAFIA

ANSELMO**,** Fernando; **Tudo Sobre a JSP com o NetBeans em Aplicações Distribuídas.** 1. ed. Florianópolis: Visual Books, 2008.

APACHE – **TomCat e frameWorks** – Disponível em www.apache.org – Acesso em 03/11/2008.

CAETANO, Cristiano; **CVS Controle de Versões e Desenvolvimento Colaborativo de Software.** 1. ed. São Paulo: Editora Novatec, 2004.

GNULAMP - **Concurrent Versions System** – Disponível em <http://www.gnulamp.com/cvs.html> Acesso em: 19 out. 2008.

GONÇALVES, Edson; **Dominando o NetBeans.** 1. ed. Rio de Janeiro: Editora Ciência Moderna Ltda, 2006.

GUJ – **Noticias, tutorias, e o maior fórum brasileiro sobre Java** – Disponível em <www.guj.com.br> - Acesso em 19 out. 2008.

IBM – **Servidores Web** – Disponível em www.ibm.com - Acesso em 03/11/2008.

ISO - **ISO 8402:1994 –** Disponível em <http://www.iso.org/iso/iso_catalogue/catalogue_ics/catalogue_detail_ics.htm?csnumber=20115> - Acesso em 19 out. 2008.

ITAMAR VIRTUAL - **Principais Comandos do CVS** – Disponível em <http://itamar.oktiva.net/oktiva.net/1054/nota/1597 >Acesso em: 19 out. 2008.

NEUBERT, Marden; **CVS Guia de Consulta Rápida.** 1. ed. São Paulo: Editora Novatec, 2004.

MARCH HARE PTY LTD & CVSNT PROJECT – **CVSNT** – Disponível em <http://www.march-hare.com/cvspro/> - Acesso em 19 out. 2008.

SOARES, Walace; **AJAX(Asynchronous JavaScript And XML)**. 1. ed. São Paulo: Editora Érica, 2006.

SUN MICROSYSTEMS – **Especificação de comandos Java** – Disponível em www.sun.com – Acesso em 03/11/2008.

WIKIBOOKS – **Manual do CVS** – Disponível em <http://pt.wikibooks.org/wiki/Manual_de_CVS> - Acesso em 19 out. 2008.