SISTEMAS DE CONTROLES DE VERSÃO

Ana Paula Corrêa
Diego Rodrigo GRein
Otavio Rodolfo Piske

Centro Universitário Positivo
Rua: Prof Pedro Viriato Parigot de Souza, 5300

RESUMO

Este artigo aborda os aspectos referentes a sistemas de controle de versão, bem como contém uma análise de diversas ferramentas para controle de versões, suas características, funcionalidades e qualidades. Além disso também aborda interfaces gráficas disponíveis para dois dos SCVs mais utilizados

INTRODUÇÃO

O crescimento do software livre se deu graças a Internet e ao seu irrestrito alcance. Analistas e desenvolvedores do mundo todo mantém milhares de projetos, o site www.sourceforge.org é uma grande prova disso, onde contém vários projetos sendo desenvolvidos e atualizados.

O objetivo deste estudo é demonstrar as diferenças, funcionalidades e características comuns aos SCVs em geral de modo a possibilitar aos profissionais da área de TI o entendimento sobre essas ferramentas de modo que no futuro estes sejam capaz de escolher entre as diversas opções disponíveis no mercado.

Muitos sistemas em uso atualmente, devem à utlização de um sistema de controle de versões (SCVs) o fato de serem estáveis, cheios de funcionalidades e seguros (embora a simples utilização de um SCV não seja fator primordial para obtenção destas qualidades). Os SCVs tornam banais inúmeras tarefas que seriam bastante complicadas sem o auxílio dos mesmos.

O objetivo deste artigo é dar uma visão geral sobre o funcionamento de SCVs em geral, bem como uma revisão sobre funcionalidades, qualidades e defeitos específicos ao SCVs mais comuns disponíveis atualmente. Este artigo terá como enfoque a utilização de SCVs em ambientes de desenvolvimento

CORPO DO ARTIGO

O que é Controle de versão?

Dentro da área de Engenharia de Software, controle de versão refere-se um conjunto de práticas cujo objetivo principal é manter controle sobre as modificações efetuadas em um determinado arquivo. Quando se fala em gerenciamento de versões, na área de desenvolvimento de softwares, de modo geral, geralmente está se fazendo uma referência ao gerenciamento de versões de um conjunto de arquivods (como no caso de um projeto).

Sistemas de Controle de Versão

Sistemas de controle de versão são sistemas utilizados para ajudar e automatizar o processo de controle de versão de documentos, códigos fontes de programa e quaisquer arquivos utilizados dentro de um determinado ambiente que sofra alterações. Com um sistema de controle de versões é possível gerenciar o histórico, modificações e versões simultâneas diferentes de um mesmo documento, resultando em uma maior organização e controle sobre os mesmos. Além disso, um sistema de controle de versões permite que duas ou mais pessoas trabalhem em um mesmo documento simultâneamente, aumentando a produtividade e auxiliando o gerenciamento de conflitos introduzidos pelas mudanças.

O funcionamento de qualquer SCV consiste basicamente em manter um repositório, centralizado ou descentralizado, com os arquivos de dados a serem controlados, bem como um histórico de suas mudanças, log, versionamento, patches, etc. De modo a otimizar o espaço em disco ocupado pelas diversas versões de um arquivo, uma boa parte dos SCVs utiliza um método de compressão conhecido como Delta, que consiste em armazenar apenas as diferenças entre versões sucessivas de um determinado arquivo. O princípio básico da compressão delta é

que as somas das diferenças entre versões sequênciais de um determinado arquivo, partindo da versão original (ou mais antiga), é capaz de produzir a versão mais nova

Tipos de SCV: Centralizado vs. Descentralizado.

Podemos separar os sistemas de controle de versão em 2 tipos básicos, de acordo com a maneira como eles gerenciam seu repositório: centralizado e descentralizado. Nos SCVs centralizados as mudanças são feitas para um servidor central único, ao passo que nos SCVs descentralizados, os desenvolvedores trabalham diretamente no seu repositório.

É possível afirmar, também, que SCVs descentralizados utilizam um enfoque "ponta-a-ponta", ao passo que SCVs centralizados utilizam um método "cliente/servidor".

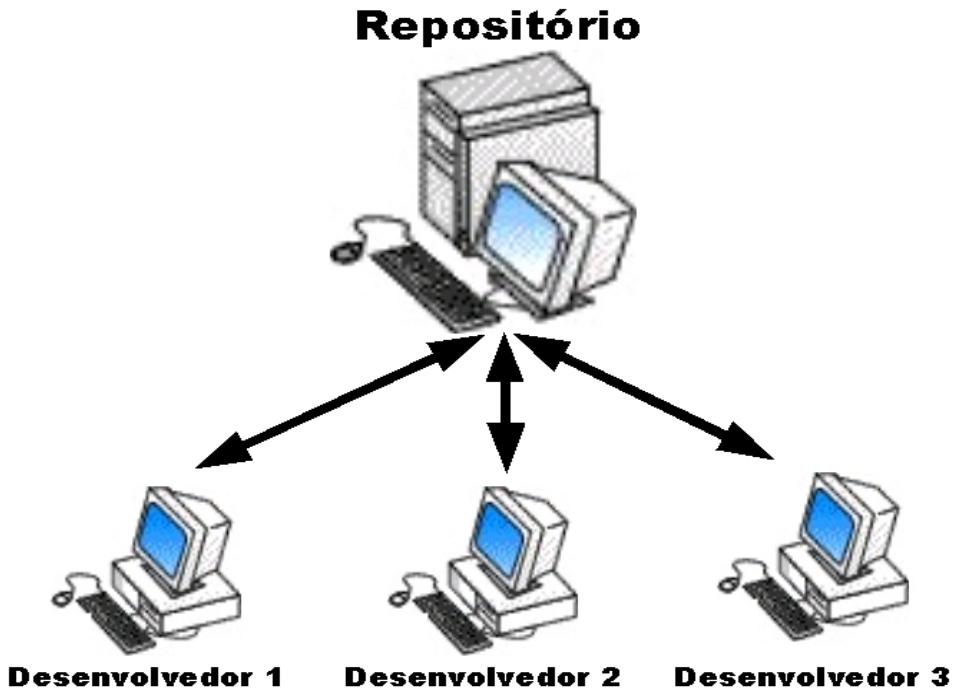
Centralizado: vantagens e desvantagens

Os SCVs centralizados, apesar de terem um funcionamento muito mais simples que os SCVs descentralizados, oferecem algumas características interessantes em relação aos seus concorrentes descentralizados, muitas delas dizem respeito a sua própria simplicidade e facilidade de centralizar os dados e controlar um único repositório.

Como o próprio nome diz, os SCVs centralizados mantém as informações referentes aos arquivos em um repositório único, acessível (dentro de um determinado contexto de segurança específico ao SCV em questão) a todos os desenvolvedores de um projeto. Pode-se dizer que os SCVs centralizados utilizam uma arquitetura Cliente/Servidor.

Em um SCV centralizado, cada desenvolvedor mantém uma cópia dos arquivos contidos no repositório, além de alguns metadados referentes a estes arquivos, que são utilizados para "cruzar" com as informações existentes no repositório. Ao efetuar uma mudança em um arquivo qualquer, o desenvolvedor atualiza o repositório, tornando publicas suas alterações, através de um processo conhecido como "commit".

Adicionalmente a isto é importante ressaltar a facilidade de controlar o acesso em um SCVs centralizado devido ao fato de as informações estão localizadas em apenas um repositório.



Devido à própria simplicidade inerente à sua implementação, os SCVs centralizados tem algumas desvantagens em relação aos descentralizados, desta maneira tornando-os pouco indicados para ambientes mais complexos, ou aonde os desenvolvedores encontram-se "espalhados" geograficamente e não exista uma conexão estável e permanente com o repositório de dados. Desta maneira, pode-se dizer que os SCVs centralizados são extremamente dependentes do repositório que contém os dados, de tal maneira que um desenvolvedor não pode sequer requisitar informações de um determinado arquivo se não existir uma conexão com o repositório, embora isto não o impeça de continuar editando os arquivos..

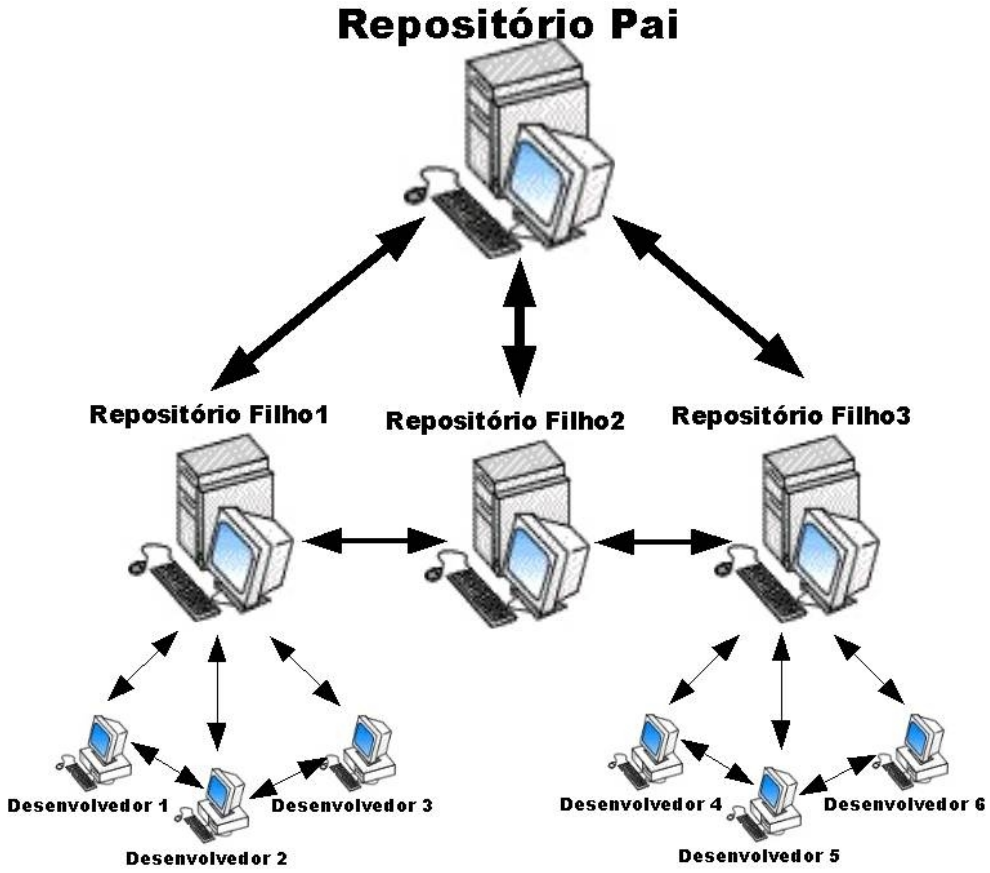
Apesar de o fato de existir apenas um repositório seja vantajoso quando relacionado ao controle de acesso, isto pode ser um problema se não existir uma política de backup séria e confiável. É importante ressaltar, porém, que a identificação da melhor forma de backup (incremental, completo, etc) para um repositório de dados de um SCV é variável de acordo com a maneira como o SCV armeza os arquivos e seus metadados, pois alguns SCVs se utilizam de bancos de dados para manter suas informações ao passo que outros se utilizam de uma árvore de arquivos logicamente ordenada para este fim.

Descentralizados: Vantagens e Desvantagens

Ao contrário dos SCVs centralizados, os SCVs descentralizados tem um design muito mais complexo e até certo ponto, complicado de se entender, mas que traz uma série de vantagens inexistentes nos seus semelhantes centralizados.

Do ponto de vista operacional, um SCVs descentralizado não é muito diferente de um SCV centralizado: usuários fazem "commit" das suas modificações, geram patches, atualizam seus arquivos, verificam seu histórico, etc, da mesma maneira que fariam com um SCV centralizado.

A grande diferença reside no fato de que não existe um repositório, mas sim vários repositórios e um repositório para cada usuário, dependendo da ferramenta utilizada. Deste modo, é possível fazer, por exemplo, com que repositórios filhos de um mesmo repositório pai compartilhem suas mudanças entre si.



Em um SCV descentralizado, uma conexão permanente com o servidor não é requisito fundamental para o funcionamento do sistema. Uma vez que cada usuário mantém o seu próprio repositório, as mudanças podem sofrer um "commit" local e somente depois atingirem o repositório pai ou irmãos, se este existirem. Adicionalmente a isto, os país deste repositório filho pode fazer um "commit" destas mudanças, se aplicável neste contexto, ao seu repositório pai, desta maneira tornando as suas modificações publicas para outros membros do conjunto de repositórios.

Esta maneira de organizar e acessar os repositórios torna os SCVs descentralizados idéiais para grupos de trabalho espalhados ao redor do mundo, aonde uma conexão entre os membros nem sempre existe ou é estável. O que não quer dizer que uma conexão de rede, ou pelo menos uma forma de propagar estas modificações não seja importante para um SCV descentralizado. Ocorre que, um desenvolvedor utilizando um SCV descentralizado poderá ter acesso as informações do repositório pai mesmo que, no momento, não exista uma conexão com o mesmo. Estas informações, porém, serão referentes à última sincronização do repositório local, com o repositório pai ou irmão. Além disto, o fato de ser descentralizado não quer dizer que este repositório filho não necessite replicar as suas modificações para o seu repositório pai, caso contrário as suas modificações não seriam replicadas para os outros membros do conjunto.

Assim como nas redes de trocas de arquivos p2p, em um SCV descentralizado é um pouco difícil ter controle sobre o acesso aos dados existentes em um determinado repositório, embora muitas das ferramentas disponíveis tenham funcionalidades específicas para resolver este problema.

SCVs e o Mundo Real

No universo do software livre o uso de sistemas de controle de versão é uma realidade em muitos casos. Projetos dos mais variados tamanhos, tipos e políticas fazem uso dessa ferramenta para auxiliar o seu desenvolvimento.

Entre os projetos usuários de SCVs centralizados podemos citar: SourceForge.net (cvs), CodigoLivre (cvs), Projeto Gentoo (cvs/subversion), Projeto Debian (cvs), Berlios.de (cvs/subversion), entre muitos outros, e entre os projetos usuários de SCVs descentralizados o exemplo mais conhecido é o do desenvolvimento do kernel do Linux, que utiliza a ferramenta Bitkeeper. Aqui no Brasil, pode-se citar o caso de sucesso da Conectiva (atualmente Mandriva), que há alguns anos vem utilizando o subversion para gerenciar as versões de código-fonte dos programas distribuídos em sua distribuição Linux.

Sistemas de Controle de Versão - Visão Geral

CVS

O CVS, ou Concurrent Versions Systems, é um dos sistemas de controle de versão mais antigos, lançado em 1984 por Dick Grune com a intenção de auxiliar o desenvolvimento do ACK (Amsterdam Compiler Kit), desenvolvido em conjunto com seus alunos Erik Baalbergen e Maarten Waage. Inicialmente se chamava cmt. O CVS é um SCV centralizado, podendo ser utlizado tanto remotamente quanto localmente. O CVS é, provavelmente, uma das ferramentas de controle de versão mais conhecidas e utilizadas, provavelmente devido a sua maturidade, licença de uso não restritiva (GPL), simplicidade e facilidade de uso. O CVS foi, também, inicialmente baseado em um sistema de controle de versões bastante simples, chamado RCS (Revision Control System), capaz de trabalhar apenas com arquivos individuais, e não projetos inteiros, como o CVS.

Podemos citar alguns tipos de conexões:

- Conexão direta com autenticação por usuário e senha: usuários de um repositório precisam fornecer um login e uma senha para obter informações, fazer commits, atualizações etc. Embora este controle de acesso seja relativamente comum, é um tanto inseguro, pois as informações (incluindo login e senha) trafegam sem criptografia pela rede, portanto passíveis de serem obtidas e analisadas por alguém mal-intencionado.

- Conexão direta com autenticação através de shell remoto: os usuários podem utilizar um shell remoto, como o rsh ou o ssh, para fazer autenticação e transferências de dados, estando sujeitos as permissões de acesso especificas aos repositórios nos quais têm acesso. Uma solução bastante utilizada, através deste esquema de autenticação, consiste em utilizar o ssh em conjunto com chaves público/privadas, provendo um método de autenticação e criptografia consideravelmente seguro, se comparado com as outras opções. Uma das desvantagens deste esquema, é que usuários do repositório que não estejam utilizando a mesma solução de shell remota ou, pelo menos uma shell remota cujo formato de chaves público/privadas seja compatível, não conseguirão conectar.

- Conexão direta com GSSAPI: usuários de sistemas de segurança compatíveis com o GSSAPI (Generic Security Services Application Programming Interface - Inteface de Programação de Aplicações Genérica de Serviços de Segurança) também podem contar com este suporte por parte do CVS. Uma das desvantagens é que o CVS precisa ser compilado com suporte a esta tecnologia.

Entre outras características do CVS, pode-se citar o suporte ao rastreamento de mudanças linha-a-linha (annotate), verificação de mudanças pré e pós commit (bem como a geração de patches com as diferenças de Duas versões), entre outras.

Apesar das suas inúmeras funcionalidades, o CVS tem alguns defeitos, alguns dos quais resultado da sua idade e design. Alguns dos seus principais defeitos são: os commits não são atômicos, o design do cvs não é puramente cliente/servidor, inexistência de changesets, etc.

Mesmo assim, o CVS é, provavelmente, uma das ferramentas de controle de versão mais conhecidas e utilizadas, provavelmente devido a sua maturidade, licença de uso não restritiva (GPL), simplicidade e facilidade de uso.

Subversion

O subversion, lançado em 2004, é uma ferramenta de controle de versões cuja intenção é ser um substituto melhorado para o CVS. Ele é desenvolvido por muitos dos ex-desenvolvedores do CVS e conta com a experiência e o "know-how" de pessoas que desenvolveram uma das ferramentas de controle de versão mais

conhecidas e utilizadas no mundo. O projeto Subversion começou em 2001, financiado pela empresas CollabNet e Red Hat, a qual disponibilizou um de seus funcionários por tempo indefinido para o projeto. A partir do início de seu desenvolvimento, o Subversion levou em torno de 14 meses para ser capaz de se gerenciar os próprios arquivos do seu projeto.

O Subversion conta com várias características inexistentes no seu irmão mais velho, o CVS, e entre elas podemos citar:

- Versionamento de diretórios: o subversion é capaz de manter informações sobre operações em diretórios, deste modo é possível reorganizaruma árvore de dados sem perder as informações.

- Commits atômicos: atomicidade nos commits significa que, se por algum motivo o servidor sofrer uma parada, ou a comunicação com o cliente for interrompida, o repositório continuará em um estado confiável.

- Suporte a diversas camadas de rede: ao contrário do cvs, que não foi inicialmente desenvolvido levando em conta as diversas arquiteturas de rede existentes atualmente, o subversion conta não apenas com suporte a repositórios locais mas também repositórios remotos através do seu servidor próprio "svnserve". Adicionamente ao svnserve, uma outra camada oferece a possibilidade de fornecer acesso aos repositórios do CVS através do servidor HTTP Apache, através do protocolo WebDAV/DetaV (uma extensão do protocolo http 1.1).

- Uso eficiente de rede: o subversion tenta utilizar a rede de maneira mais eficiente, através do envio de diffs em ambos as pontas da comunicação.

Estas são apenas algumas das melhoras do subversion em relação ao cvs.Muitas outras melhoras de menor porte, tornam todo o conjuto de funcionalidades do cvs uma alternativa a ser considerada seriamente por usuários que procuram uma versão melhorada do cvs. Apesar dessas inúmeras qualidades, o subversion ainda é um projeto bastante novo, e embora já esteja em uso em diversos projetos e empresas aoredor do mundo ainda não tem a mesma maturidade encontrada no CVS. É importante citar, porém, que segundo informações encontradas na internet, todo o repositório de códigos-fonte da Conectiva (criadora do Conectiva Linux), é armazenado em um repositório Subversion consistindo em mais de 20 Gb de dados.

Bitkeeper

O Bitkeeper é um SCV descentralizado. Apesar de ser uma ferramenta comercial, era o SCV utilizado para gerenciar o controle de versões do kernel do Linux.

Visual SourceSafe

Visual SorceSafe (VSS), é um sistema de controle de versões (SCV) e administração de código fonte vendido pela Microsoft junto com o pacote Microsoft Visual Studio. A grande vantagem do VSS é a integração com outras ferramentas Microsoft como o Visual Studio .NET. Podemos definir o VSS em 3 funções básicos de um SCV: centralização os arquivos, gerênciamento de acesso e histórico das modificações.

O VSS funciona de maneira centralizada, ou seja, guardando todos seus arquivos em apenas um repositório. O VSS armazena todos os arquivos do repositório em uma pasta chamada DATA, com todos os arquivos criptografados e descaracterizados. O VSS deixa apenas uma pessoa alterar o arquivo por vez, ou seja, quando um usuário solicitar um arquivo para edição, este arquivo estará disponível para outros usuários apenas no modo somente leitura. O arquivo é liberado para escrita somente quando nenhum usuário estiver editando-o. O VSS possui grandes desvantagens que devem ser observados antes de optar por esta solução para um sistema de controle de versões, vejamos algumas:

- o gerenciamento do VSS em redes não é muito bom, ocasiona lentidões e atrasos nas operações ao repositório. O acesso remoto ao repositório através da intenet é bastante inseguro por parte do VSS, e deve-se tomar cuidado ao utilizar este recurso.

- é muito dificil gerenciar módulos externos com o VSS. É muito comum que programadores utilizem módulos externos em seus sistemas, e isto traz grande grande desvatagem na utilização do VSS.

- o VSS é extremamento lento na busca pro versões anteriores no seu repositório

- não é recomendado para base de dados grandes. A própria Microsoft recomenda não ultrapassar 5 GB de dados.

- não existe versão nativa para Linux ou Unix.

Hoje em dia o VSS está em desuso e sua utilização não é muito indicada, a Microsoft apostou em um novo produto, o Visual Studio 2005 Team System (VSTS). O VSTS não herdeu nada do VSS e foi construído a partir do zero, tendo toda estrutura de gerenciamento refeita. Além disso, trabalha sobre o banco de dados SQL Server 2005 e tem acesso remoto nativo através da web, ao contrario do VSS, que contava com uma extensão externa, através da API, para isso.

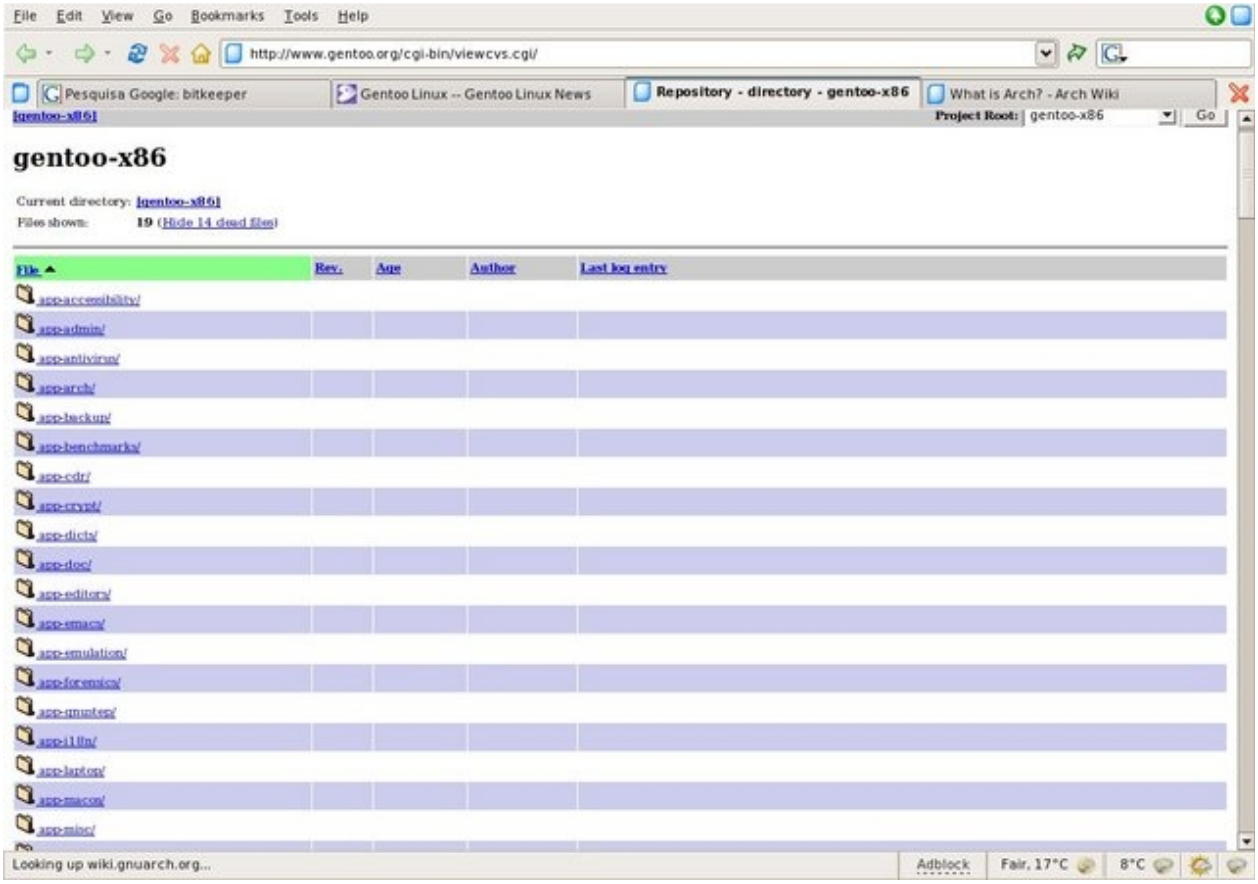
Interfaces Gráficas

Alguns sistemas de controle de versão tem a principal interface com o usuário em modo texto, ou seja, não contam com uma GUI (Graphical User Interface) para execução de comandos e gerenciamento do repositório. Este é o caso de ferramentas como o CVS e o Subversion.

Visando resolver este problema, muitos projetos desenvolveram interfaces gráficas para interagir com essas ferramentas. Estas interfaces gráficas podem ser dividas em duas partes: as interfaces através de aplicações desktop e as interfaces web.

Interfaces Web

Intefaces Web são interfaces que utilizam o browser em conjunto com diversos protolocos comuns na web como meio de comunicação com o usuário. Não é raro encontrar entre projetos de software livre, links nas suas páginas principais para sub-paginas aonde os usuários podem navegar pelo repositório de dados do projeto utilizando interfaces web.



ViewCVS

ViewCVS é uma ferramente OpenSource que serve como interface de navegação para controlar repositório. Ele apresenta em formato HTML, os diretórios, revisões, branches e log de mudanças. Algumas de suas funcionalidades são:

- independência de repositório: o ViewCVS é capaz de acessar tanto repositórios CVS quanto repositórios Subversion;
- Suporte para configuração individual de host virtual;
- Capacidade de rodar como um script CGI ou como um servidor autonômo;

ViewSVN

O ViewSVN é uma interface simples e facil usar, para visualização e gerenciamento de repositórios. É utilizada em conjunto com o subversion.

Dentre as funcionalidades disponíveis no ViewSVN, pode-se destacar a navegação entre diretórios earquivos de diferentes versões, geração de diffs e visualização de logs.

Choro Repositoru Viewer

O Chora é outra ferramenta para visualizar e gerenciar repositorios utilizando os sistema de controle de versão CVS ou Subversion. O Choro é desenvolvido tendo em mente a facilidade de adaptação e customização, além de contar com uma interface inteligente para visualização de branches.

Interfaces Desktop

Intefaces desktop funcionam como um aplicativo desktop normal, fazendo a ligação entre o arquivos e o repositório através de uma interface gráfica de modo que o usuário não precisa saber todos os comandos do SCV escolhido.

As possibilidades de escolha de uma interface desktop para utilização variam de acordo com o Sistema Operacional atualmente em uso no desktop em questão, bem como o SCV escolhido. Assim sendo serão apenas citadas as ferramentas GUI mais conhecidas

CVS:

APLICAÇÃO	S.O.	SOFTWARE LIVRE
LinCVS	Windows/*nix	Sim
WinCVS	Windows	Sim
MacCVS	Mac OS	Sim

Subversion:

APLICAÇÃO	S.O.	SOFTWARE LIVRE
RapidSVN	Windows/*nix	Sim
TortoiseSVC	Windows	Sim
		Sim

CONCLUSÃO

Este artigo nos leva a acreditar que, uma vez que tenha sido escolhido um SVCs adequado a um determinado ambiente, este pode ser de grande valia para o aumento de produtividade e melhorias no controle dos arquivos. Porém, é importante ressaltar que para a correta escolha do SCV ideal a este determinado contexto, é preciso conhecer as características técnicas, funcionalidades, vantagens e desvantagens de cada ferramenta disponível, assim tornando possível identificar quais características são relevantes para esse determinado contexto.

REFERÊNCIAS

1 GNU Arch Tutorial: (acessado em 02/06/2005)
http://www.gnu.org/software/gnu-arch/tutorial/arch.html

2 Subversion Home Page: http://subversion.tigris.org/ (acessado em 20/05/2005)

3 Wikipedia (CVS): http://en.wikipedia.org/wiki/Cvs (acessado em 22/05/2005)

4 Wikipedia (RCS): http://en.wikipedia.org/wiki/Revision_Control_System (acessado em 10/06/2005)

5 Wikipedia (Version Control System): (acessado em 11/06/2005)
http://en.wikipedia.org/wiki/Version_control_system

6 Better SCM Iniciative: http://better-scm.berlios.de/ (acessado em 10/06/2005)

7 Comments on Open Source Software / Free Software (OSS/FS) Software (acessado em 12/06/2005)

8 Configuration Management (SCM) Systems: (acessado em 01/06/2005)
http://www.dwheeler.com/essays/scm.html

9 The New Breede Of Version Control Systems: (acessado em 12/06/2005)
http://www.onlamp.com/pub/a/onlamp/2004/01/29/scm_overview.html

10 Bitkeeper Documentation: http://www.bitkeeper.com/Documentation.html (acessado em 12/06/2005)