Instituto de Matemática e Estatística

USP

MAC 5856 - Software Livre

Professor: Fabio Kon

REPOSITÓRIOS

1

ÁLVARO HENRY MAMANI ALIAGA
POLIANA MAGALHÃES REIS

Resumo da Apresentação

- Definições
- Vantagens
- Atividades Comuns
- Integração com outros Softwares
- Como funciona ?
- Repositórios Centralizados
- Repositórios Distribuídos

Resumo da Apresentação (Cont.)

- Sistemas de Controle de Versão
  - Centralizados
    - RCS – Revision Control System
    - CVS – Concurrent Version System
    - SVN – Subversion
    - VSS – Microsoft Visual SourceSafe
    - Rational ClearCase
  - Distribuidos
    - GIT
    - BAZZAR
    - Mercurial
    - GNU Arch
    - Monotone
    - Darcs

DEFINIÇÕES

- Um repositório pode ser um sistema de arquivos local ou um servidor remoto, e são extensamente usados em **Sistemas de Controle de Versão**.

- Um **sistema de controle de versão**, **VCS** (Version Control System) ou ainda **SCM** (Source Code Management) por sua vez, providencia árvore de versão para gerenciar versões diferentes de arquivos.

EXEMPLO : Várias ferramentas de controle de versão usam arquitetura de cliente/servidor e os clientes podem pegar qualquer informações sobre qualquer arquivo acessando o repositório.

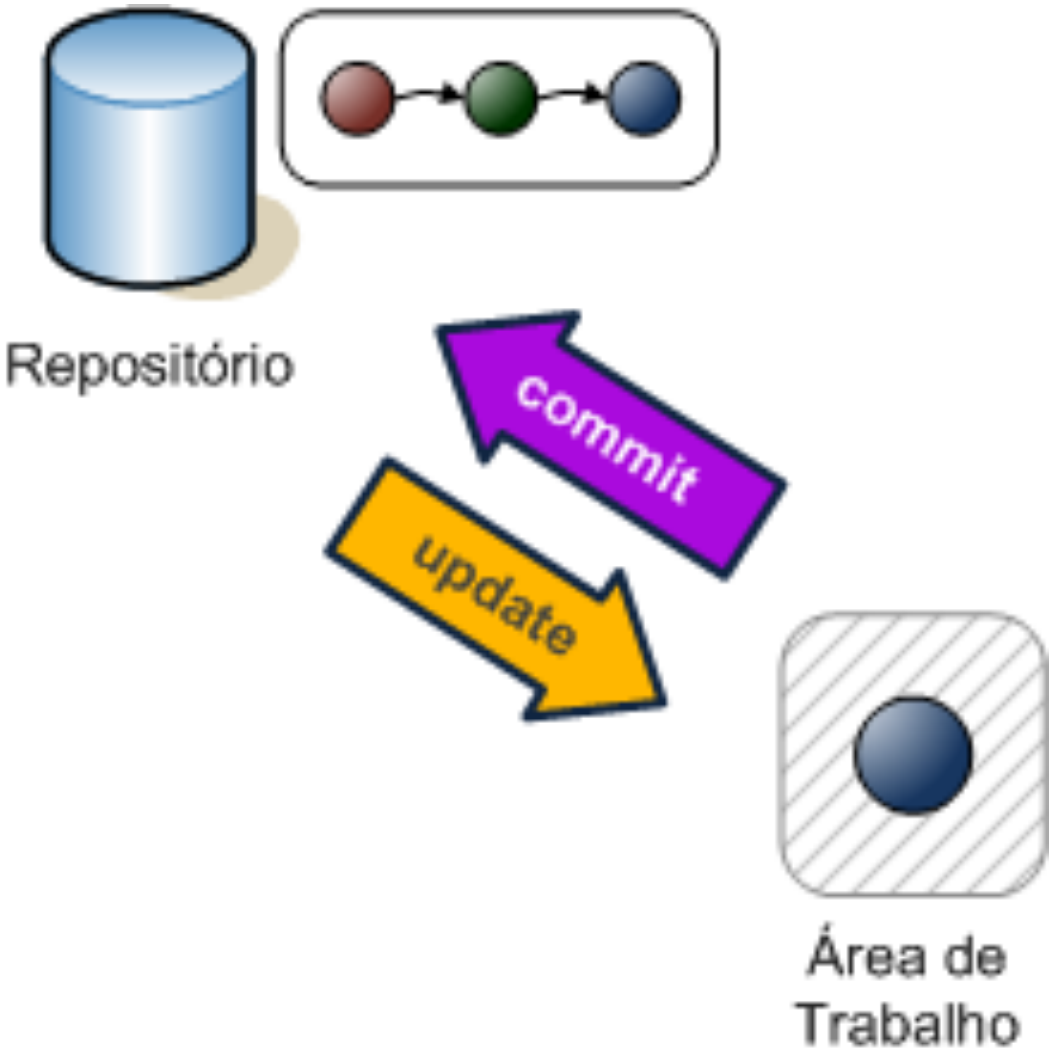
Principais Vantagens

- Controle do histórico
- Trabalho em equipe
- Marcação e resgate de versões estáveis
- Ramificação de projeto

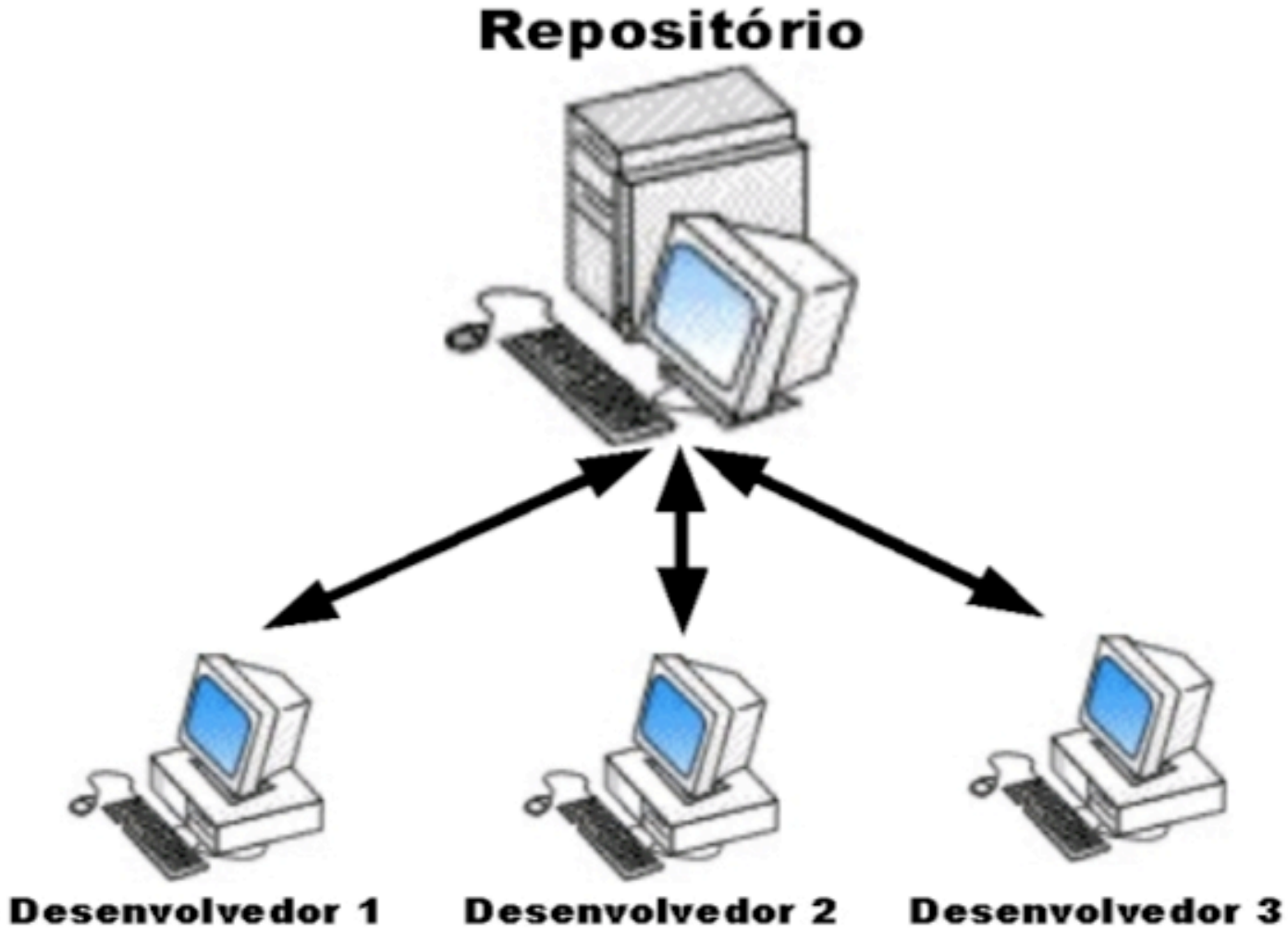
Integração com outros softwares

- Os sistemas de controle de versões mais flexíveis permitem que seja possível integrá-los a outros *softwares*.
- A integração mais comum é em ambientes de desenvolvimento através de plugins .
- Alguns ambientes que suportam a integração de alguns sistemas são:
  - IntelliJ IDEA, Eclipse, NetBeans e Visual Studio.
- O TortoiseSVN, o TortoiseCVS e o TortoiseHg, clientes do SVN, do CVS e do Mercurial, respectivamente, funcionam sobre o Windows Explorer.

Como Funciona o Controle de Versão?



Repositórios Centralizados



REPOSITÓRIOS CENTRALIZADOS

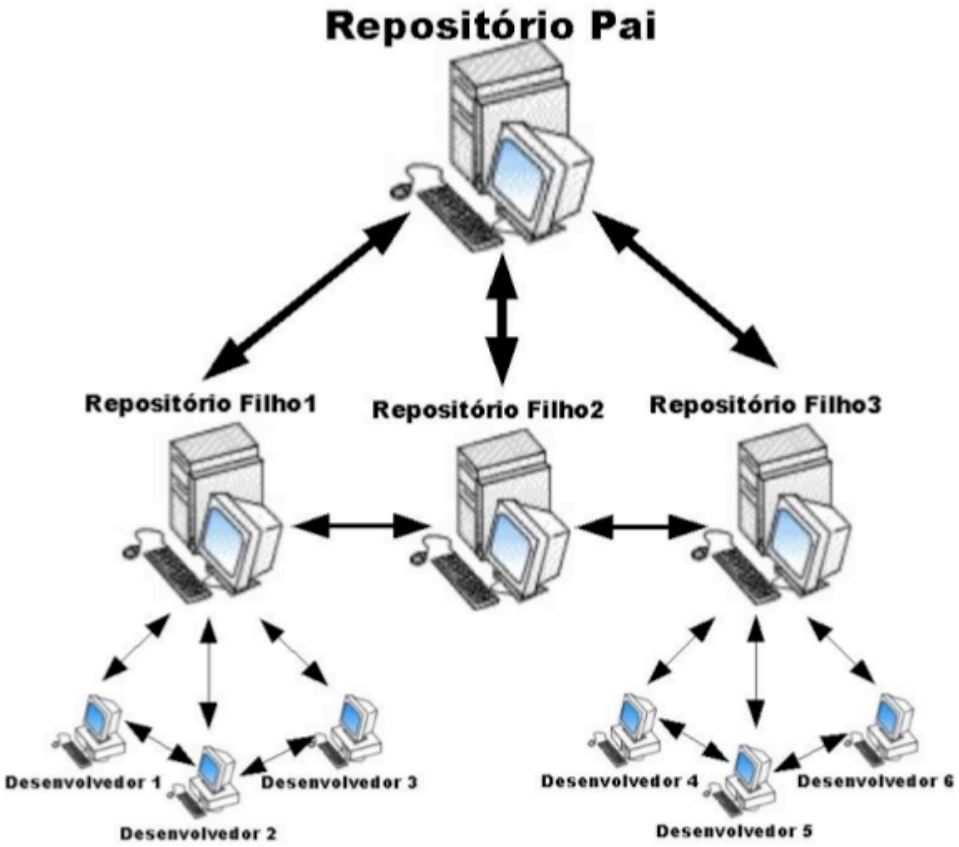
VANTAGENS

- Mantém as informações referentes aos arquivos em um repositório único acessível a todos os desenvolvedores de um projeto.
- Simplicidade
- Facilidade de controlar o acesso

DESVANTAGENS

- São extremamente dependentes do repositório que contém os dados
- Necessidade de uma boa estrutura de backup

Repositórios Distribuídos



- Design muito mais complexo
- Uma conexão permanente com o servidor não é requisito fundamental para o funcionamento do sistema

Repositórios Centralizados

- RCS – Revision Control System
- CVS – Concurrent Version System
- SVN – Subversion
- VSS – Microsoft Visual SourceSafe
- Rational ClearCase

RCS - Revision Control System

- RCS foi inicialmente desenvolvido em 1980 por Walter F. Tichy [1985]
- Consiste em um conjunto de comandos UNIX
- Modelo checkout/checkin [Feiler 1991]
- Os arquivos são armazenados no repositório na forma de árvore.
- Armazena somente os arquivos correntes no repositório, que conhece somente as diferenças entre as revisões armazenadas

RCS – "DEFEITOS"

- O método de bloqueio (locking) dos arquivos impede o desenvolvimento paralelo . OBS: A programação paralela pode ser simulada com a utilização de branches, seguidos da instrução merge.
- O sistema não permite que exista uma hierarquia de diretórios dentro de um repositório. (O **RCS** gerencia os artefatos apenas dentro de um determinado diretório.)

Estas limitações incentivaram o desenvolvimento do chamado CVS (Concurrent Versions System).

REPOSITÓRIOS CENTRALIZADOS

CVS - CONCURRENT VERSION SYSTEM

- Primeira versão lançada em 1984 por Dick Grune
- O código que evoluiu para a versão atual do CVS foi iniciada por Brian Berliner em abril de 1989.
- Usa o formato para armazenamento de arquivo do RCS
- Pode ser usado tanto localmente como remotamente
- Licença GPL
- Suporte ao rastreamento de mudanças linha-a-linha (annotate),

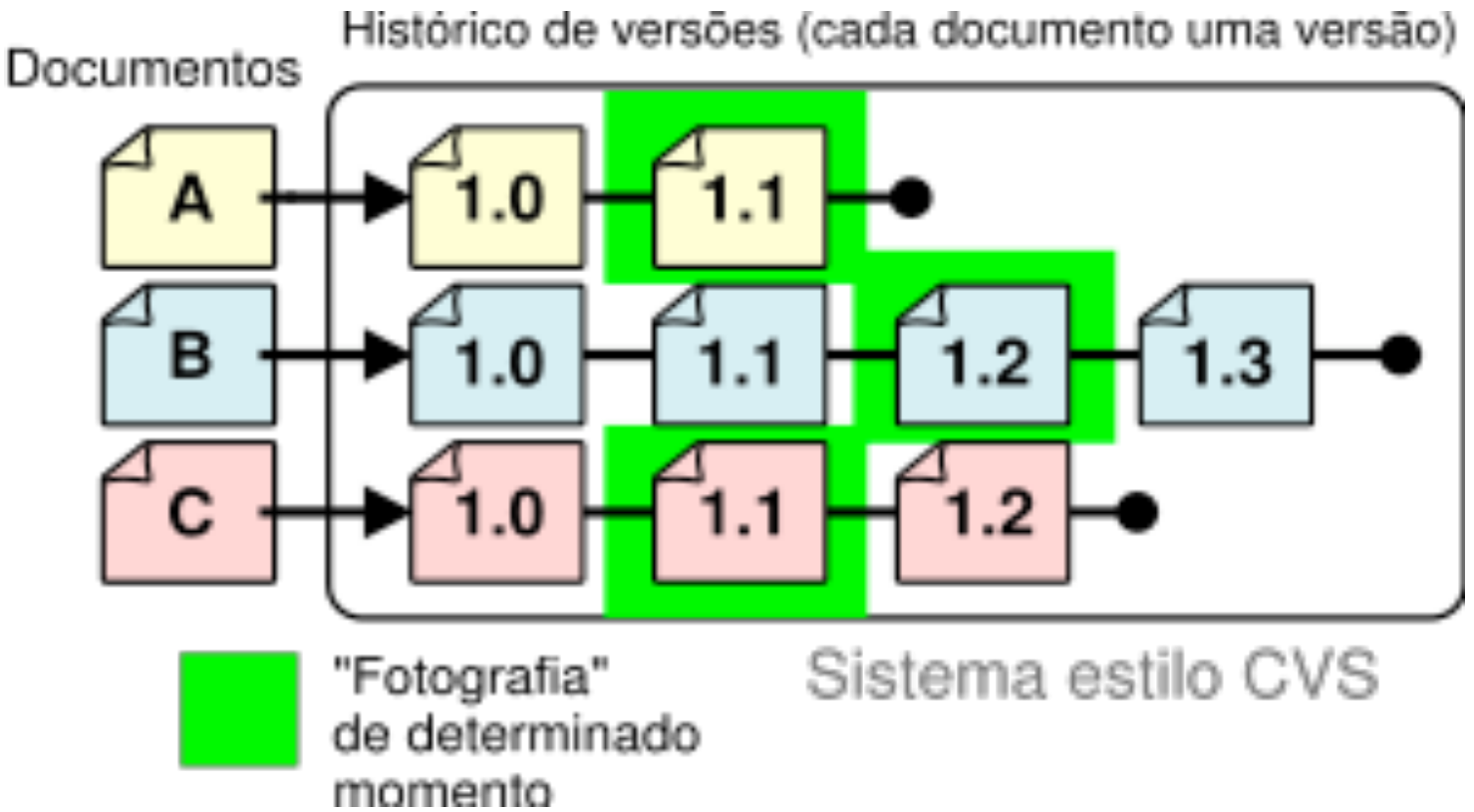
REPOSITÓRIOS CENTRALIZADOS

CVS - CONCURRENT VERSION SYSTEM

- Suporte ao desenvolvimento de um software originalmente desenvolvido por outra pessoa ou empresa . (Import)
- Criação de módulos dentro de um repositório.
- Permite total personalização sobre o armazenamento do registro de mudanças
- Desenvolvimento concorrente – Paradigma "copy-modify-merge" [Berliner 1990]
- Conceito de "bloqueio fraco"

Um levantamento bastante completo das funcionalidades do CVS pode ser encontrado em [Fogel 1999] e [Cederqvist 1993].

CVS - Envio e resgate de versões



CVS – "Defeitos"

- Os commits não são atômicos,
- Histórico de versões não muito bom
- Requer maior administração que o RCS
- Curva de aprendizagem íngreme para os novatos

SVN - SUBVERSION

- O projeto Subversion começou em 2000, financiado pela empresas CollabNet e Red Hat
- Em agosto de 2001 o sistema se tornou auto gerenciável (Os desenvolvedores do Subversion pararam de usar o CVS para gerir seu próprio código-fonte, e começaram a usar o próprio Subversion no lugar.)
- Primeira versão lançada em 2004
- Atualmente encontra-se na versão 1.6
- Licença CollabNet/Tigris.org
- Os custos são proporcionais ao tamanho das mudanças e não ao tamanho dos dados
- Enquanto o CVS usa versionamento por **arquivo**, o SVN usa versionamento por **módulo**.

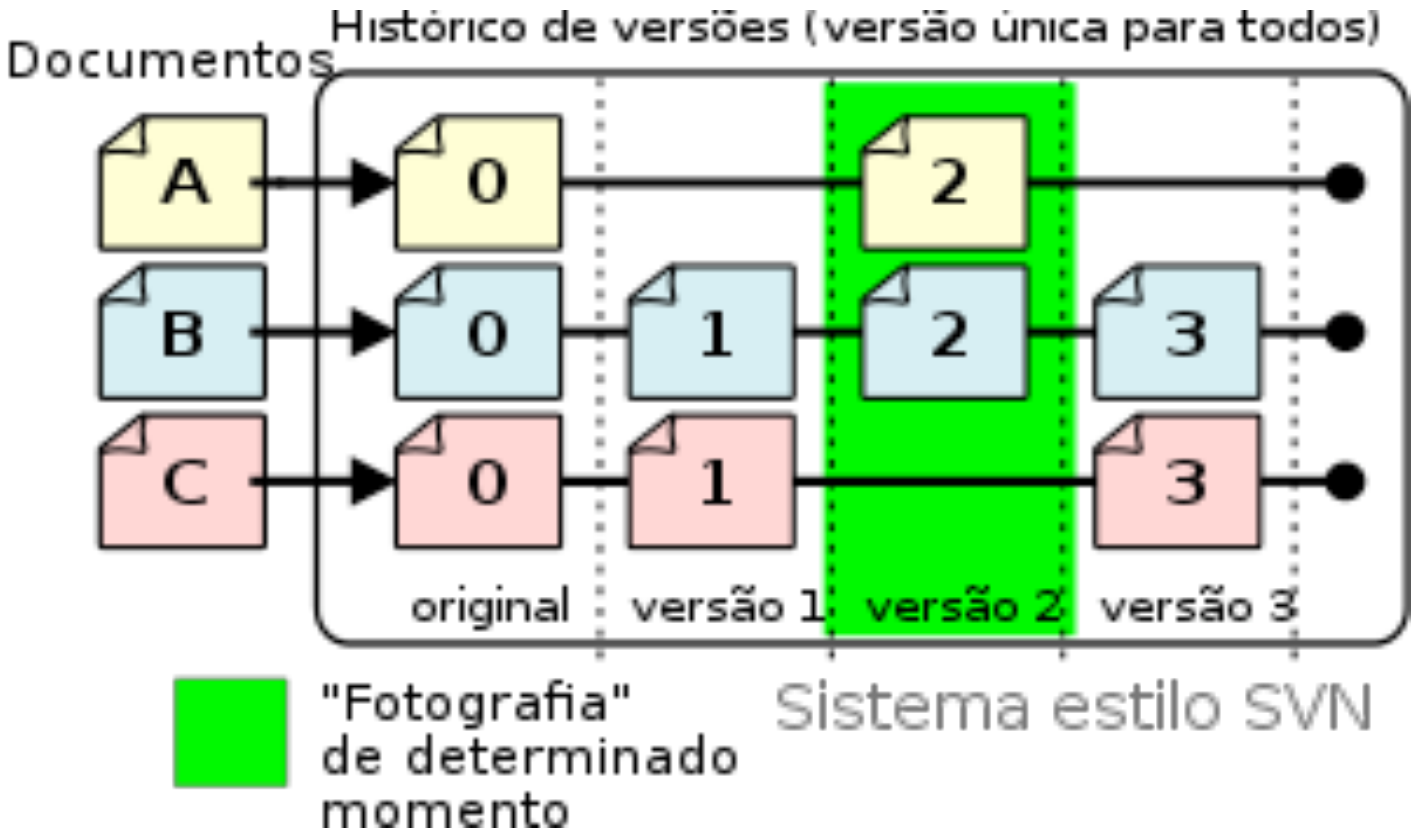
SVN - SUBVERSION

- O Subversion conta com várias características inexistentes no seu irmão mais velho, o CVS, e entre elas podemos citar:
  - Versionamento de diretórios
  - Histórico de versões bom
  - Commits atômicos
  - Escolha das camadas de rede (associa-se ao servidor Apache HTTP pelo protocolo WebDAV/DeltaV)

http://svnbook.red-bean.com/en/1.4/svn-book.html

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.

SVN - ENVIO E RESGATE DE VERSÕES



VSS- VISUAL SOURCE SAFE

- É um sistema de controle de versões e administração de código fonte vendido pela Microsoft junto com o pacote Microsoft Visual Studio.
- A grande vantagem é a integração com outras ferramentas Microsoft como o Visual Studio .NET.
- Boa interface com usuário
- Podemos definir o VSS em 3 funções básicas de um SCV:
  - Centralização os arquivos
  - Gerênciamento de acesso
  - Histórico das modificações

VSS - DESVANTAGENS

- Ruim gerenciamento em redes
- Dificiuldade de gerenciar módulos externos
- Lentidão na busca por versões anteriores no seu repositório
- Não existe versão nativa para Linux ou Unix.

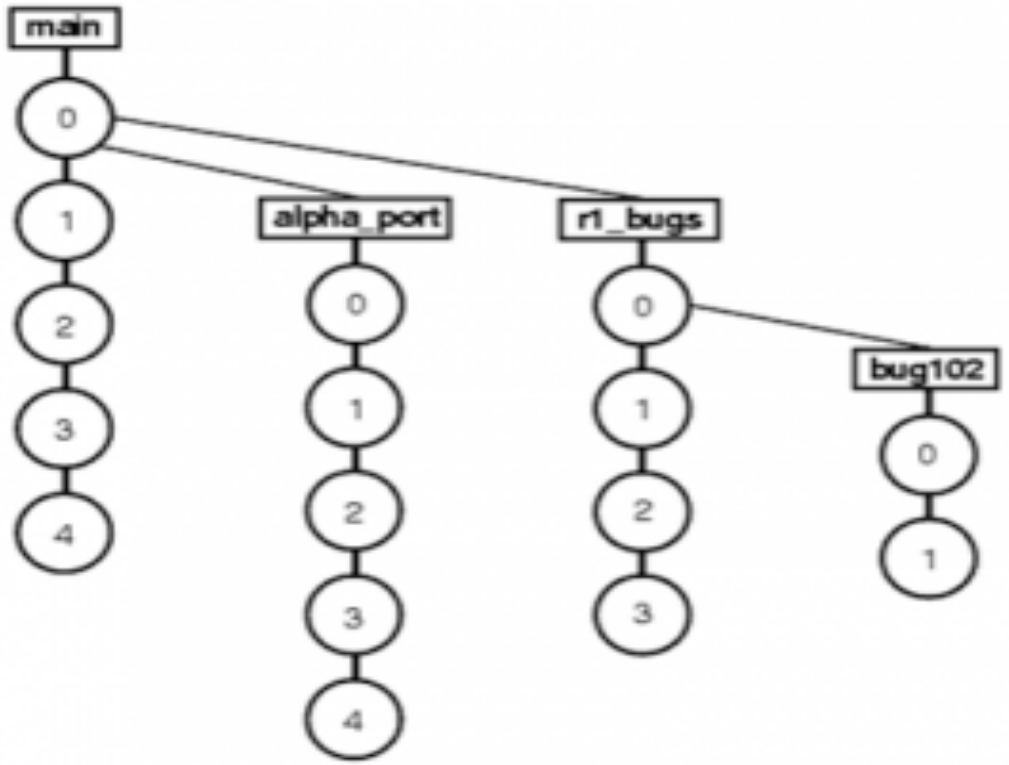
OBS : Hoje em dia o VSS está em desuso e sua utilização não é muito indicada, a Microsoft apostou em um novo produto, o Visual Studio 2005 Team System (VSTS). O VSTS não herdeu nada do VSS e foi construído a partir do zero, tendo toda estrutura de gerenciamento refeita. Além disso, trabalha sobre o banco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

Clear Case

- Permite criar dois tipos de projetos: Base e UCM.
- Projeto proprietário da Rational (IBM)
- Um projeto do **ClearCase Base** possui recursos similares ao SVN e CVS. (check-in, check-out, branch, labels, etc…)
- No **ClearCase UCM**
  - **Stream:** Seria como um branch integrado a atividades. Geralmente são separadas por desenvolvedores e existe uma stream de integração.
  - **Delivery:** É a entrega do código de uma atividade para a stream de integração;
  - **Rebase:** Representa a atualização da sua stream de desenvolvimento a partir de uma baseline;

BRANCH X STREAM

- Um branch é uma ramificação no controlador de versão do fonte do nosso projeto. Essa ramificação é muito utilizada para realizar manutenções evolutivas e correções no software.



BRANCH X STREAM

- Branch pode evitar que o código parcial de uma manutenção evolutiva vá por engano para homologação/ produção junto com a correção de um bug simples aberto e corrigido nas últimas horas.

- Se uma pessoa fizer algo errado e realizar o check-in, essa pessoa pode impactar o trabalho de outras pessoas, mesmo em projetos com testes unitários

- Para evitar isso o ClearCase UCM existe a opção de utilizar o que é chamado de desenvolvimento em paralelo que é um pouco diferente do conceito do CVS / SVN ou mesmo ClearCase Base.

Comparativo Geral

SCV	RCS	CVS	SVN	VSS	Clear Case
Desenvolvimento Concorrente	Não permite	Bom	Bom	Não permite	Muito Bom
Comits Atômicos	Não	Não	Sim	Não	Sim
Versiona-mento	Versiona-mento de arquivos	Versiona-mento de arquivos	Versiona mento de módulos	Versiona mento de arquivos	Versiona mento de arquivos
Histórico de versões	Regular	Regular	Muito bom	bom	Muito bom

Comparativo Geral

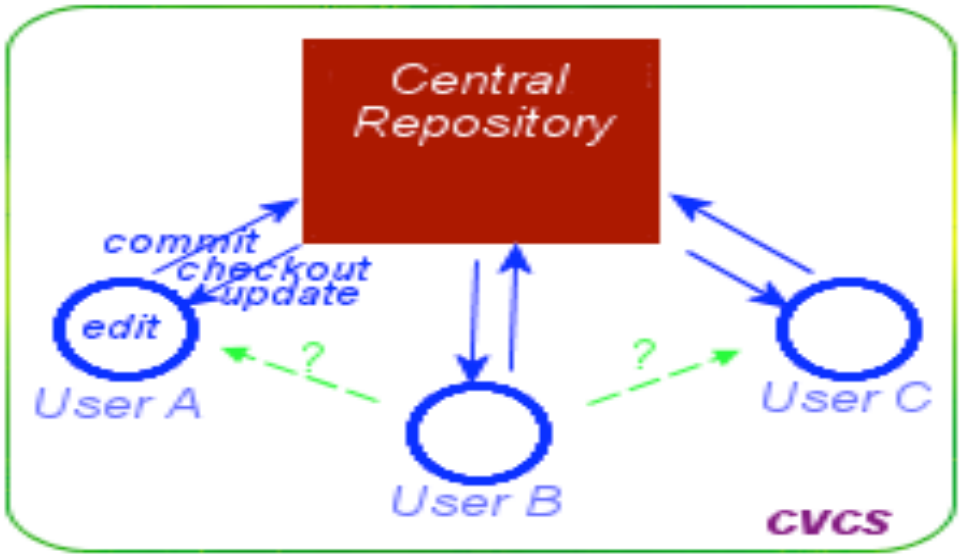
SCV	RCS	CVS	SVN	VSS	Clear Case
Lançado em	1980	1986	2004	1995	1992
Desenvolvido por	Walter F. Tichy	Dick Grune	CollabNet e Red Hat	One Tree Software	Atria Software
Licença	GPL	GPL	CollabNet/ Tigris.org	MSDN Universal License	Rational Licence
Método de Bloqueio	Sim	Não (Pode ser simulado)	Não	Sim	Não
Bloqueio Fraco	Não	Sim	Sim	Não	Sim

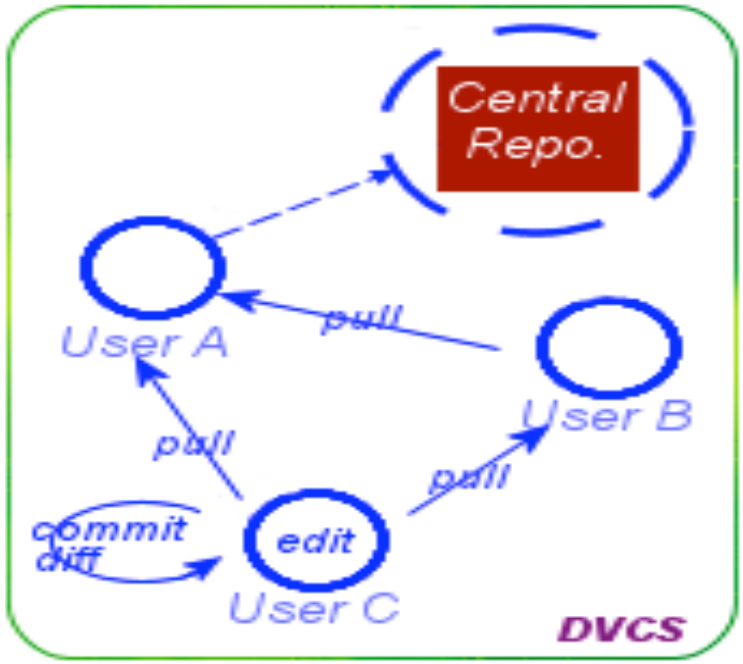
CONCLUSÕES

- Entre os mais comuns encontram-se *as soluções livres:* CVS e SVN; e *as comerciais:* SourceSafe e ClearCase.

- Optar por uma solução comercial geralmente está relacionada à garantia, pois as soluções livres não se responsabilizam por erros no software e perdas de informações, apesar das soluções livres poderem ter melhor desempenho e segurança que as comerciais

COMPARAÇÃO: CENTRALIZADO E DISTRIBUÍDO



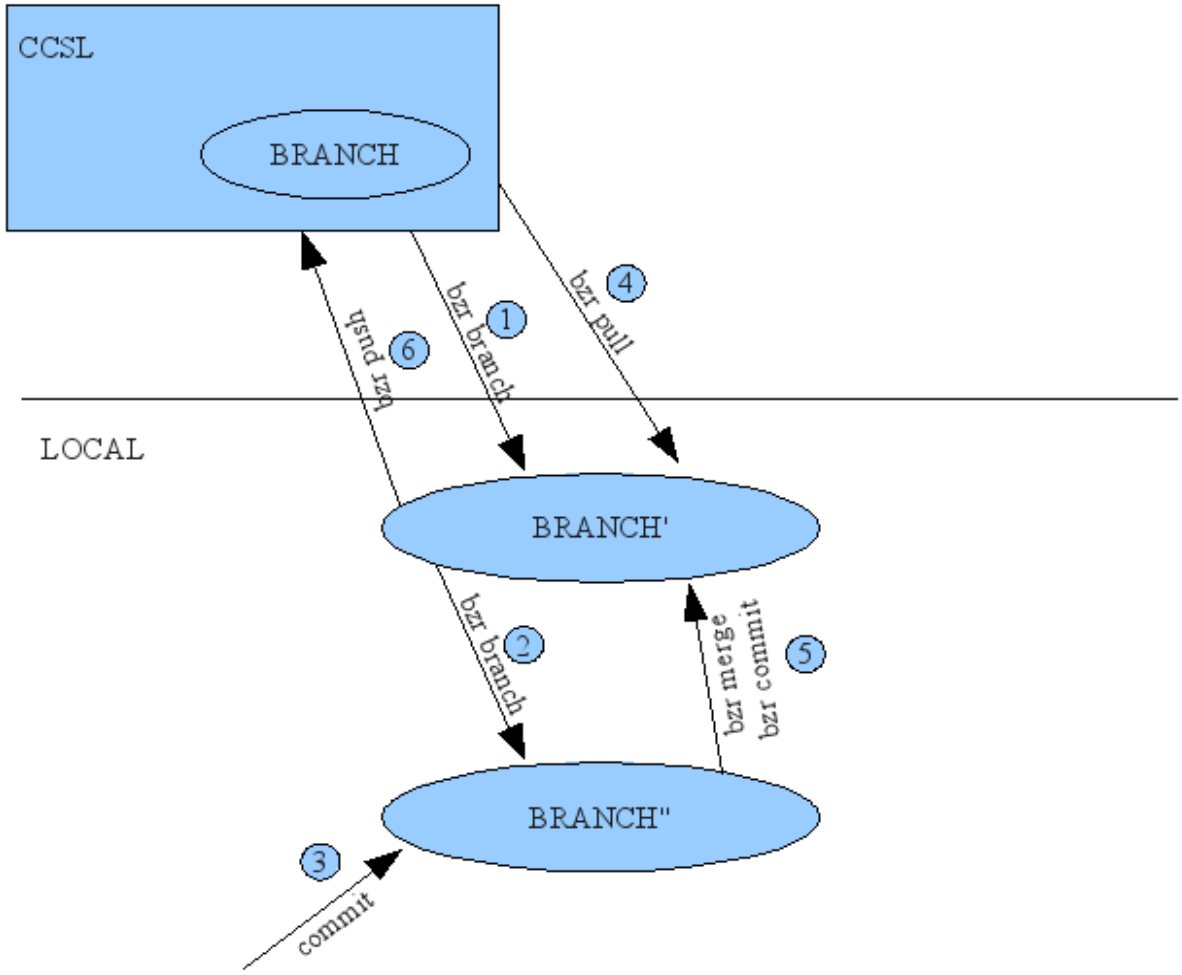


http://www.infoq.com/articles/dvcs-guide

Repositórios Distribuídos

- Cada projeto é um repositório completo, com histórico total.
- Independente de acesso à rede ou servidor central.
- Commits podem ser feitos offline.
- Commits podem ser transferidos (push e pull) de um repositório para outro facilmente.

Recomendável



Git

Git

- Criador: Linus Torvalds(2005).
- Responsável: Junio C Hamano
- Licença: GPL v2
- Última versão: v1.6.5.2
- Escrito em: C, Bourne Shell, Perl

GIT: CARACTERÍSTICAS

- Desenvolvido para manter o Kernel do Linux.
- Rapidez, eficiência e uso em grandes projetos.
- http://git.or.cz
- Commits offline. Commits parciais.
- Bisect
- Git – SVN: Permite usar Git em repositórios SVN

GIT: GUIS

- gitk
- GitX(Mac OS X)
- TortoiseGit (Windows)
- Git Extensions (Windows)
- Qgit (Qt)

REPOSITÓRIOS DISTRIBUÍDOS

GIT: HOSPEDAGEM

- Hospedagem pública
  - repo.or.cz - http://repo.or.cz
  - Gitorious - http://gitorious.org/
- Hospedagem privada
  - codebase - http://www.codebasehq.com/
  - Unfuddle - http://unfuddle.com/
- Hospedagem Privada e Pública
  - GitHub - http://github.com/
  - CodaSet - http://codaset.com/

GIT

- Projetos de software que usam Git:
  - Kernel do Linux,
  - Servidor X.org,
  - Qt(toolkit),
  - Android do Google
  - Ruby on Rails.
  - gcc

Bazaar

Bazaar

- Criador: Martin Pool
- Desenvolvido por: Canonical Ltd. and community
- Licença: GPL
- Última versão: October 14, 2009: Bazaar 2.0.1
- Escrito em: Python, Pyrex, C

BAZAAR: CARACTERISTICAS

- Comunidade grande e ativa(Canonical/Ubuntu)
- http://bazaar-vsc.org/
- O controle do repositório fica na raiz do projeto, em um diretório chamado .bzr
- Permite trabalhar nos repositórios SVN, Git, Mercurial.
- Workflows
- Focado na facilidade de uso.
- Suporta plugins.
- Criticado por mau desempenho

BAZAAR: GUIS

- GUIs para bazaar estão disponíveis para:
  - Windows
  - Linux
  - OS X

BAZAAR: HOSPEDAGEM

- Launchpad - https://launchpad.net/
- SourceForge - http://sf.net/
- GNU Savannah - http://savannah.gnu.org/

BAZAAR

- Projetos de software que usam Bazaar:

Mercurial

Mercurial

- Responsável: Matt Mackall
- Licença: GPL v2
- Última versão: 1.3.1 / 2009-07-23
- Escrito em: Python e C.

MERCURIAL: CARACTERÍSTICAS

- Funciona basicamente linha de comando, sempre começa com “hg”.
- http://mercurial.selenic.com/
- Permitir fazer o clone via http
- Permitir fazer o pull (subir as alterações) para o servidor também via http

MERCURIAL: GUI

- Hgk (tcl/tk)
- GUIs para Mercurial estão disponíveis para:
  - Windows
  - Linux
  - OS X
- Netbeans, desde a versão 6
- TortoiseHg

MERCURIAL: HOSPEDAGEM

- FreeHg - http://freehg.org/
- SourceForce - http://sourceforge.net/
- Google Code- http://code.google.com/projecthosting/
- GNU Savannah - http://savannah.gnu.org/
- Bitbucket - http://www.bitbucket.org/

MERCURIAL

- Projetos de software que usam Mercurial:
  - Mozilla
  - Symbian OS
  - GNU Octave
  - Todos os projetos da Sun usam Mercurial, exceto MySQL.

LAUNCHPAD

LAUNCHPAD

- É uma aplicação web.
- Software livre.
- Criador: Canonical Ltd.
- Desenvolvido por: Canonical Ltd. and community
- Licença: GNU Affero General Public License
- Última versão: Launchpad 3.0 – 23 Sep 2009
- Escrito em: Python

LAUNCHPAD

- Usa Bazaar, como repositório
- Outros grandes projetos usando Launchpad são:
  - MySQL (code hosting)
  - Zope 3 (bug tracking)
  - Inkscape (bug tracking)
  - Bazaar
  - GNOME Do
  - Integrade

LAUNCHPAD

- 15,009 projetos
- 461,952 bugs
- 222,746 branches

GNU ARCH

- Parte do projeto GNU.
- Licença: GNU GPL
- Autor original: Thomas Lord
- Desenvolvedor: Andy Tai
- Versão estável: 1.3.5 / 2006-07-20.
- Criticado pela quantidade de comandos.

Monotone

- Desenvolvedores: Nathaniel Smith, Graydon Hoare
- Escrito em: C++
- Pode importar projetos CVS.
- Fácil de aprender.
- Menos popular do que Bazaar, Git e Mercurial.

Darcs

- Desenvolvido por: David Roundy
- Escrito em: Inicialmente C++, depois Haskell
- Desenvolvido para substituir os tradicionais CVS e SVN

COMPARAÇÃO DOS REPOSITÓRIOS

Software	Mantido por	Estado	Licença	Plataforma	Linguagem
Git	Junio Hamano	Ativo	GPLv2	Linux, Mac OS X – Windows 70%	C, Bourne Shell, Perl
Bazaar	Canonical Ltd.	Ativo	GPL	Linux, Mac OS X – Windows 100%	Python, Pyrex, C
Mercurial	Matt Mackall	Ativo	GPLv2	Linux, Mac OS X – Windows 50%	Python e C
GNU Arch	Andy Tai	Mantido	GPL	Linux, Windows, Mac OS X	Shell scripts
Monotone	Nathaniel Smith, Graydon Hoare	Ativo	GPL	Linux, Windows, Mac OS X	C++
Darcs	David Roundy	Ativo	GPL	Linux, Windows, Mac OS X	Haskell

Comparativo Geral – Parte 1

Característica	Arch	Bazaar	Git	Mercurial	Monotone
Commits Atômicos	Sim	Sim	Sim	Sim	Sim
Arquivos e Diretórios se podem mover e/ou renomear	Sim	Sim	Sim	Sim	Sim
Merging Inteligente, depois de mover ou renomear	Sim	Sim	Não	Sim	Sim
Copias de arquivos e diretórios	Não	Não	Não	Sim	Não
Replicação de Repositório	Sim	Sim	Sim	Sim	Sim

Comparativo Geral – Parte 2

Característica	Arch	Bazaar	Git	Mercurial	Monotone
Propagação de mudanças para o repositório pai	Sim	Sim	Sim	Sim	Sim
Mudanças tipo Traking Uncommited	Sim	Sim	Sim	Sim	Sim
Mensagens no commit por arquivo	Não	Sim, com plugin	Não	Não	Não
Documentação	Regular	Excelente	Boa	Boa	Boa
Replicação de Repositório	Sim	Sim	Sim	Sim	Sim

Comparativo Geral – Parte 3

Característica	Arch	Bazaar	Git	Mercurial	Monotone
Conjunto de comandos	Muitos, segue a convenção CVS e BitKeeper	Boa, segue a convenção CVS	Boa, segue a convenção CVS	Boa, segue a convenção CVS	Boa, segue a convenção CVS
Suporte a rede	Excelente, Ftp, Sftp, WebDAV.	Excelente, Http, ftp, Sftp, ssh	Excelente, Http e Https	Excelente, Http, ssh	Boa, usa protocolo "netsync"
Portabilidade	Boa sobre Unix, Linux. Camada de emulação sobre Windows	Excelente, Windows, linux, Mac OS X, FreeBSD	Boa sobre Linux, Unix Difícil com Windows	Excelente multiplataforma	Excelente multiplataforma

Dúvidas ?

BIBLIOGRAFIA

- MOLINARI, Leonardo. *Gerência de Configuração - Técnicas e Práticas no Desenvolvimento do Software*. Florianópolis: Visual Books, 2007. 85-7502-210-5

- MIKKELSEN, Tim, PHERIGO, Suzanne. *Parctical Software Configuration Management*: The Latenight Developer's Handbook. Upper Saddle River, NJ, EUA: Prentice Hall PTR, 1997. 0-13-240854-6

- Cristiano Caetano. *CVS*: Controle de Versões e Desenvolvimento Colaborativo de Software. ed. Novatec, 2004.

BIBLIOGRAFIA

- GNU Arch Tutorial: http://www.gnu.org/software/gnu-arch/tutorial/arch.html
- Subversion Home Page: http://subversion.tigris.org/
- Wikipedia (CVS): http://en.wikipedia.org/wiki/Cvs
- Wikipedia (RCS): http://en.wikipedia.org/wiki/Revision_Control_System
- Wikipedia (Version Control System): http://en.wikipedia.org/wiki/Version_control_system
- Configuration Management (SCM) Systems http://www.dwheeler.com/essays/scm.html