UNIVERSIDADE FEDERAL DO AMAZONAS

GERÊNCIA DE PROJETOS

GT3 – CONTROLE DE VERSÃO

MARCOS FELIPE PAES PESSOA
CRISTINA ARAÚJO

iComp
Instituto de Computação

O que é?

- Um sistema de controle de versão tem a finalidade de gerenciar diferentes versões de um artefato.

- Com isso ele oferece uma maneira muito mais inteligente e eficaz de organizar projetos, pois é possível acompanhar o histórico de desenvolvimento

Como funciona?

- Basicamente, os arquivos do projeto ficam armazenados em um repositório (um servidor) e o histórico de suas versões é salvo nele.

- Os desenvolvedores podem acessar e resgatar a ultima versão disponível e fazer uma cópia local, na qual poderão trabalhar em cima dela e continuar o processo de desenvolvimento.

- A cada alteração feita, é possível enviar novamente ao servidor e atualizar a sua versão a partir outras feitas pelos demais desenvolvedores

Como funciona?

Classificação

- Atualmente, os sistemas de controle de versão são classificados em dois tipos:

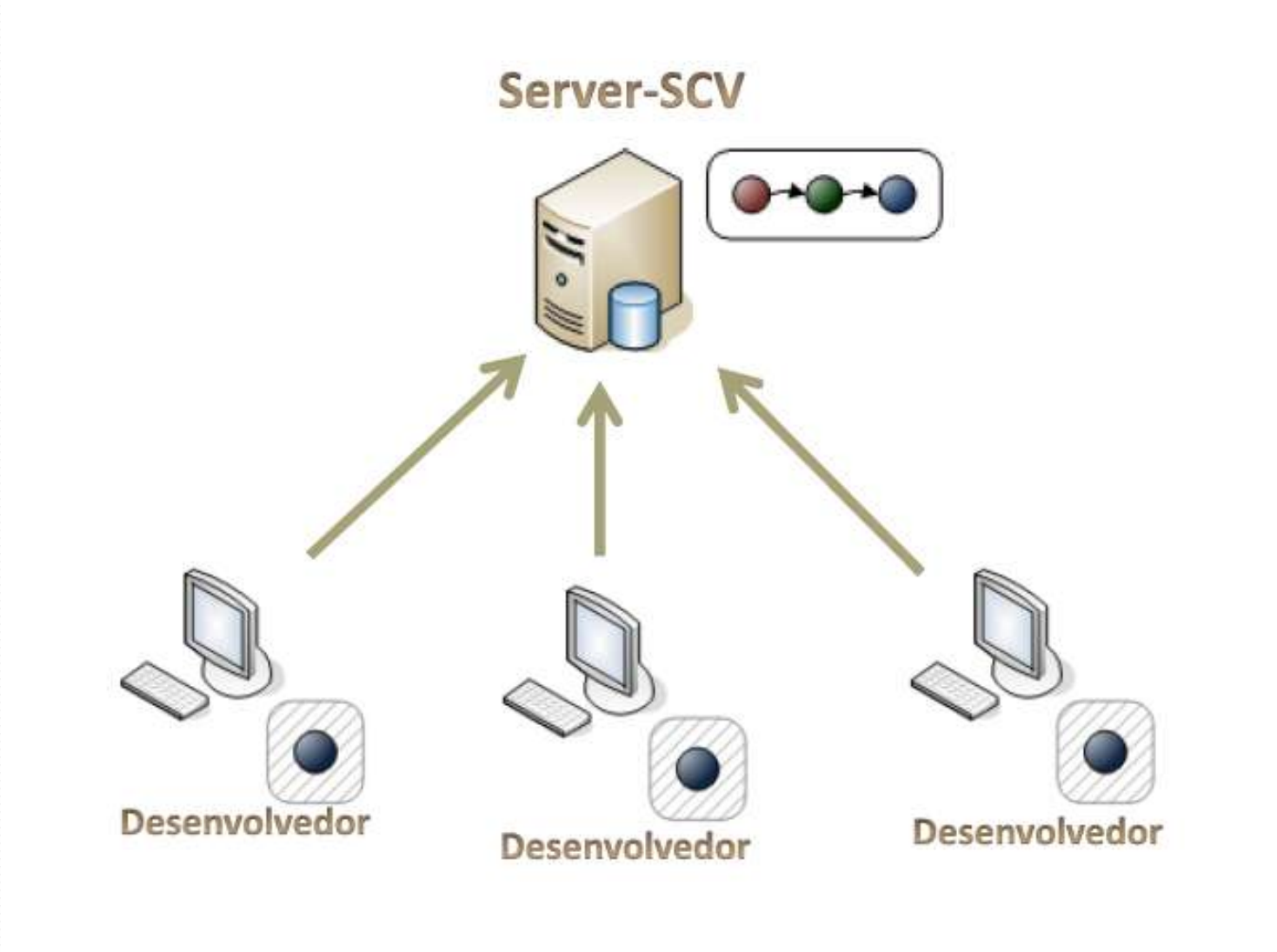
- Centralizados

- Distribuídos.

- O **centralizado** trabalha apenas com um servidor central e diversas áreas de trabalho, baseados na arquitetura cliente-servidor.

- Essa versão atende muito bem a maioria das equipes de desenvolvimento de pequeno e médio porte, e que trabalhem em uma rede local.

Arquitetura de um SCV centralizado

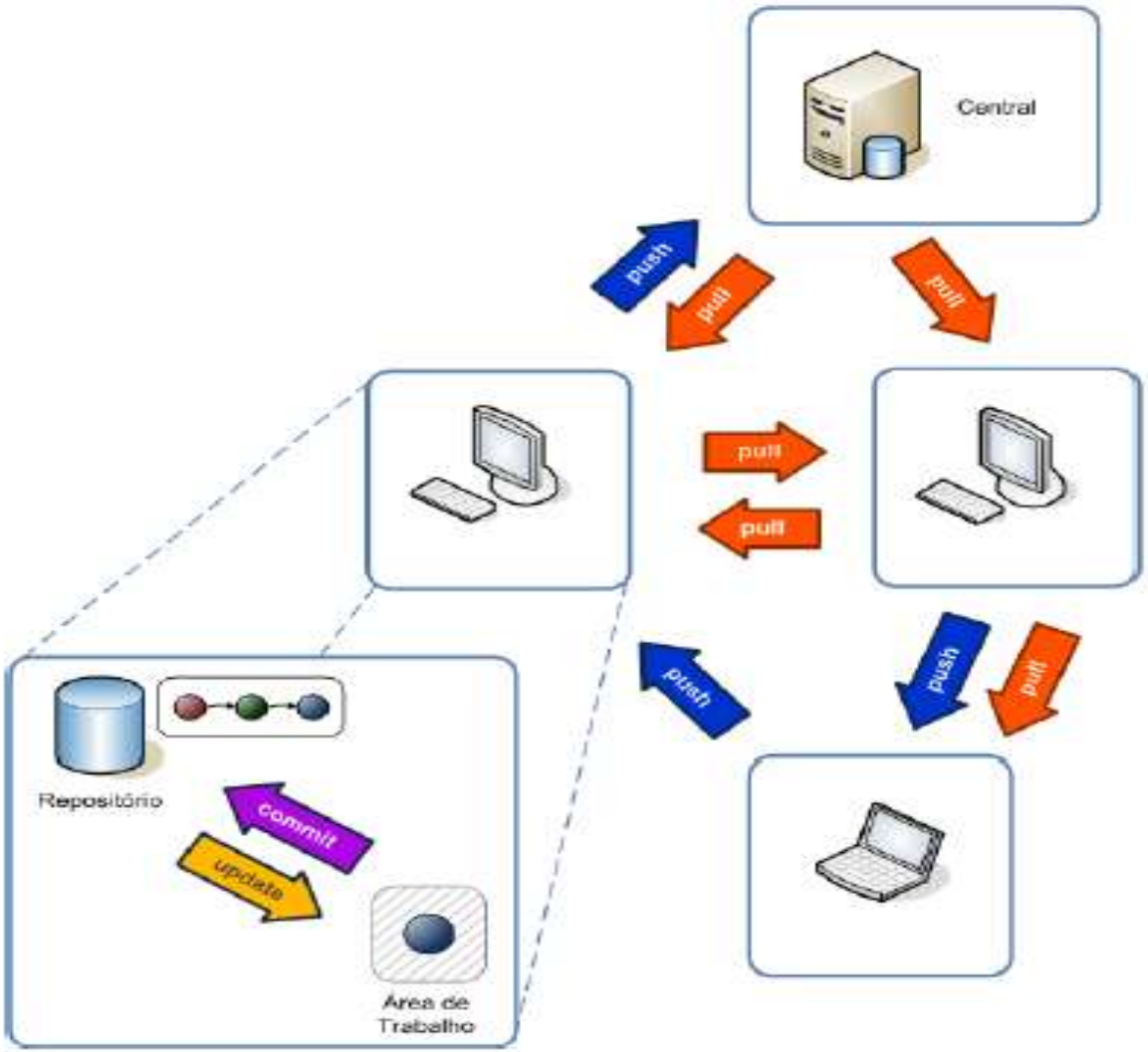


- O **distribuído** vai mais além. Ele é recomendado para equipes com muitos desenvolvedores e que se encontram em diferentes locais.

- Esta versão funciona da seguinte maneira: cada área de trabalho tem seu próprio “servidor”, ou seja, as operações de check-in e check-out são feitas na própria máquina.

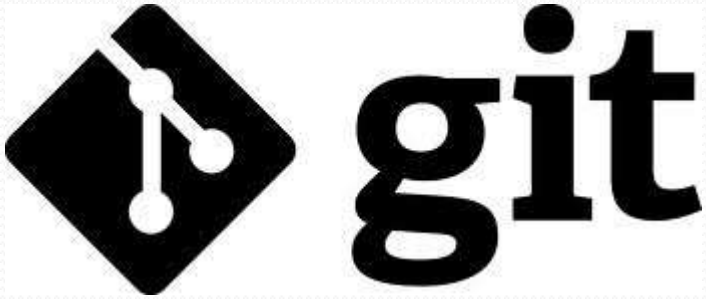
- Porém diferentemente do centralizado, as áreas de trabalho podem comunicar-se entre si, recomenda-se usar um servidor como centro do envio dos arquivos para centralizar o fluxo e evitar ramificações do projeto e a perda do controle sobre o mesmo.

Arquitetura de um SCV distribuído



- Por ser na própria máquina, o sistema de controle distribuído acaba sendo mais rápido, porém exige maior conhecimento da ferramenta e de inicio podem atrapalhar o desenvolvedor.

- Portanto, por esse tratamento de mesclagem ser diferente, podem ocorrer situações onde o trabalho de alguém possa ser sobreposto e gerar tormento para os desenvolvedores.



Porque usar?

- Qual a necessidade de utilização de ferramentas para controle de versão?

- Quais as vantagens de sua utilização?

- **Histórico**. Registra toda a evolução do projeto, cada alteração sobre cada arquivo. Com essas informações sabe-se quem fez o que, quando e onde. Além disso, permite reconstruir uma revisão específica do arquivo sempre que desejado;

- **Colaboração**. O controle de versão possibilita que vários desenvolvedores trabalhem em paralelo sobre os mesmo arquivos sem que um sobrescreva o código de outro, o que traria reaparecimento de defeitos e perda de funcionalidades;

- **Variações no Projeto**. Mantém linhas diferentes de evolução do mesmo projeto. Por exemplo, mantendo uma versão 1.0 enquanto a equipe prepara uma versão 2.0.

Imaginem o seguinte quadro:

Você chega na sua empresa de manhã cedinho para terminar aquele trabalho que você passou o dia anterior fazendo, só que chegando lá você descobre que aquele estagiário novo alterou acidentalmente os arquivos que você tanto demorou pra fazer. É um desespero não? Seu prazo está estourado e você vai ter que passar a madrugada toda trabalhando para poder entregar o trabalho em tempo.

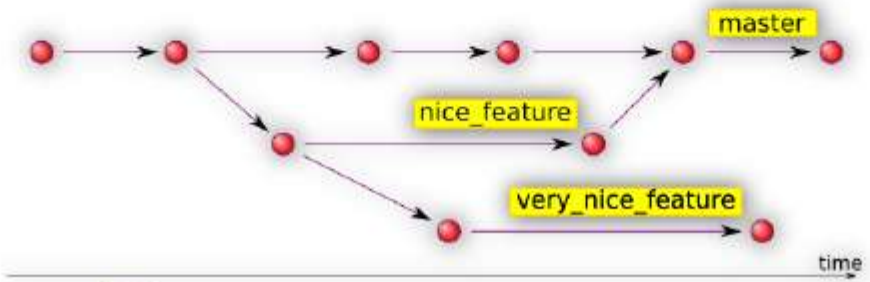
- Utilizando um sistema de controle de versão isso poderia ser facilmente resolvido com um simples comando.

- Isso porque todo o código(assim como seu histórico) fica guardado em um servidor.

Benefícios do Sistema Gerenciador de Controle de Versão (SCV)

1 Manter histórico

2 Colaboração



3 Variações do projeto

CVS

Dick Grune
Criador

SUBVERSION

Karl Fogel
Líder do projeto SVN

Outras fontes de informação

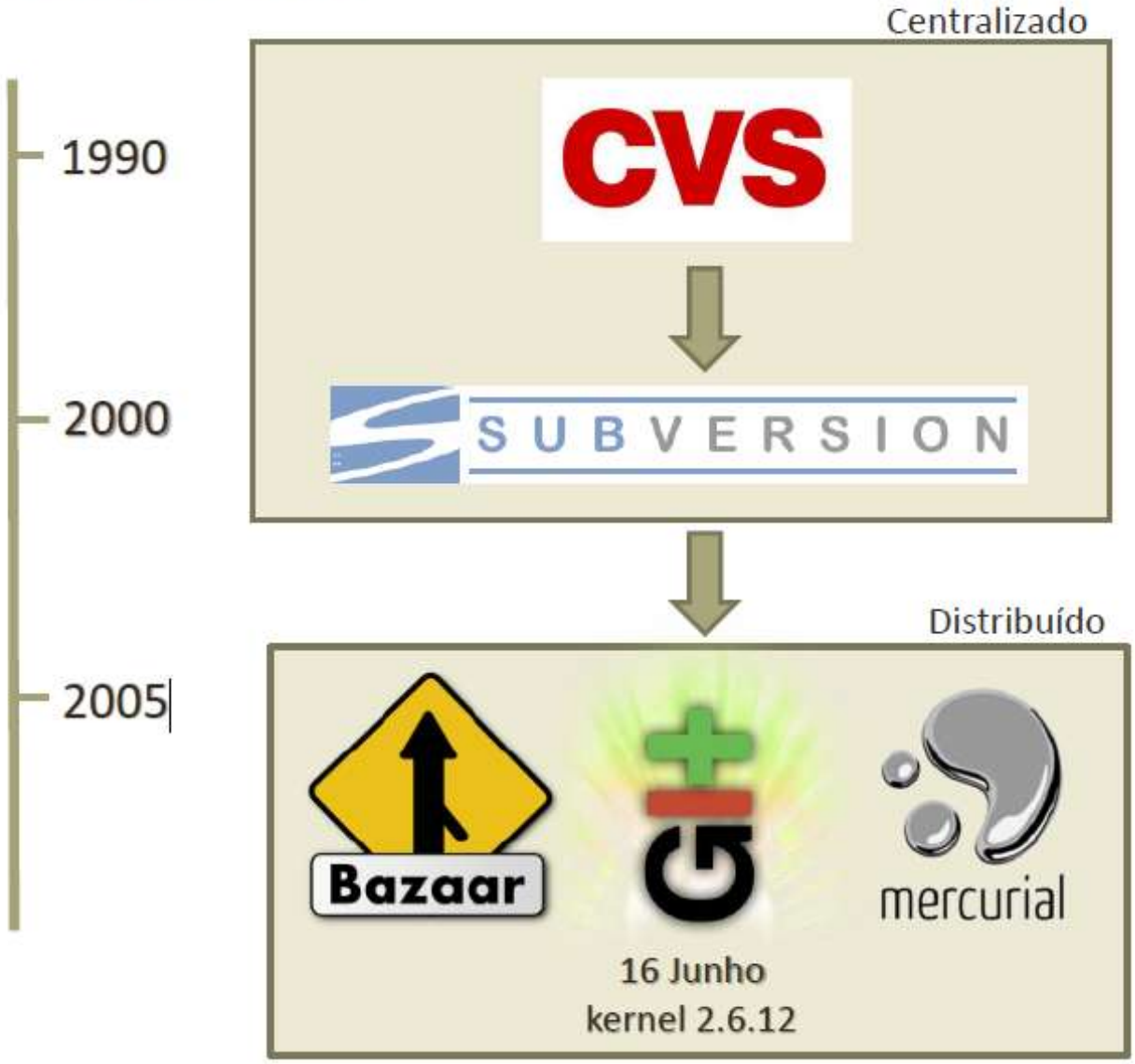
CVS

- http://cvs.nongnu.org/
- http://en.wikipedia.org/wiki/Concurrent_Versions_System

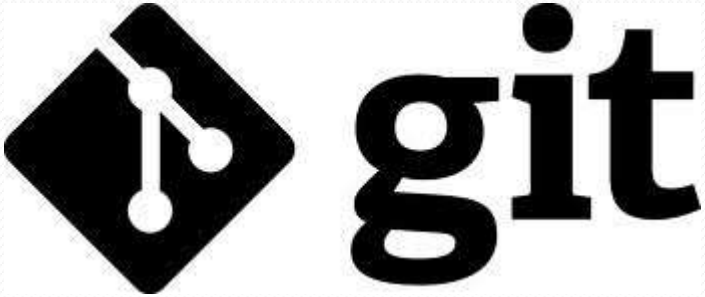
Subversion

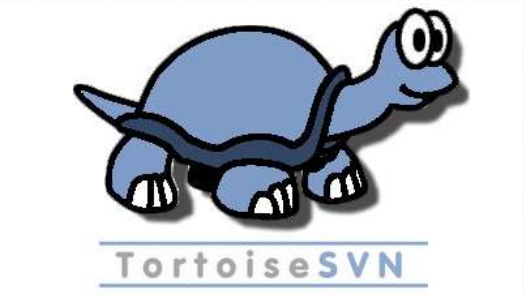
- http://subversion.apache.org/
- http://wiki.apache.org/subversion/
- http://en.wikipedia.org/wiki/SVN

Histórico


Centralizado
1990
CVS
2000
SUBVERSION
Distribuído
2005
Bazaar
mercurial
16 Junho
kernel 2.6.12

Exemplos de soluções





- **Soluções comerciais**

- Microsoft Visual SourceSafe (VSS) – produto da Microsoft para controle de versão, integrado a muitas IDEs da Microsoft.

- Rational ClearCase – produto da IBM para controle de versão.

- Borland StarTeam – produto da Borland para controle de versão e de equipe.

- **Soluções livres**

- Concurrent Version System (CVS) – software livre clássico e bem testado.
- **Subversion (SVN)**
- **Git – Software para controle de versão distribuído com foco na velocidade.**
- MediaWiki – software livre que possui um sistema integrado de controle de versões. Sites com os projetos da Wikimedia, tal como a Wikipédia mantém o sistema MediaWiki para o controle das versões dos documentos. Esse sistema permite o trabalho simultâneo de milhares de voluntários.
- GNU CSSC
- Revision Control System (RCS)
- **Bazaar**
- Darcs
- **Mercurial**
- Monotone
- SVK

Subversion

- Subversion é um sistema de controle de versão livre/open-source. Isto é, o Subversion gerencia arquivos e diretórios, e as modificações feitas neles ao longo do tempo.

- Isto permite que você recupere versões antigas de seus dados, ou que examine o histórico de suas alterações. Devido a isso, muitas pessoas tratam um sistema de controle de versão como uma espécie de “máquina do tempo”.

Características do Subversion

- Versionamento de diretórios
- Histórico de versões efetivo
- Commits atômicos
- Versionamento de metadados
- “Adaptabilidade”

Versionamento de diretórios

- O Subversion implementa um sistema de arquivos "virtual"

- Esse diretório fica sob controle de versão que rastreia modificações a toda a árvore de diretório ao longo do tempo.

Histórico de versões efetivo

- Como o CVS é limitado apenas ao versionamento de arquivos, operações como cópia e renomeação—que podem ocorrer com arquivos também, mas que são realmente alterações no conteúdo de algum diretório.

- Com o Subversion, você pode adicionar, excluir, copiar, e renomear ambos os arquivos ou diretórios. E cada novo arquivo adicionado começa com um histórico próprio e completamente novo.

Commits atômicos

- Um conjunto de modificações ou é inteiramente registrado no repositório, ou não é registrado de forma nenhuma.

- Isto possibilita aos desenvolvedores criarem e registrarem alterações como blocos lógicos, e também evita problemas que possam ocorrer quando apenas uma parte de um conjunto de alterações seja enviada com sucesso ao repositório.

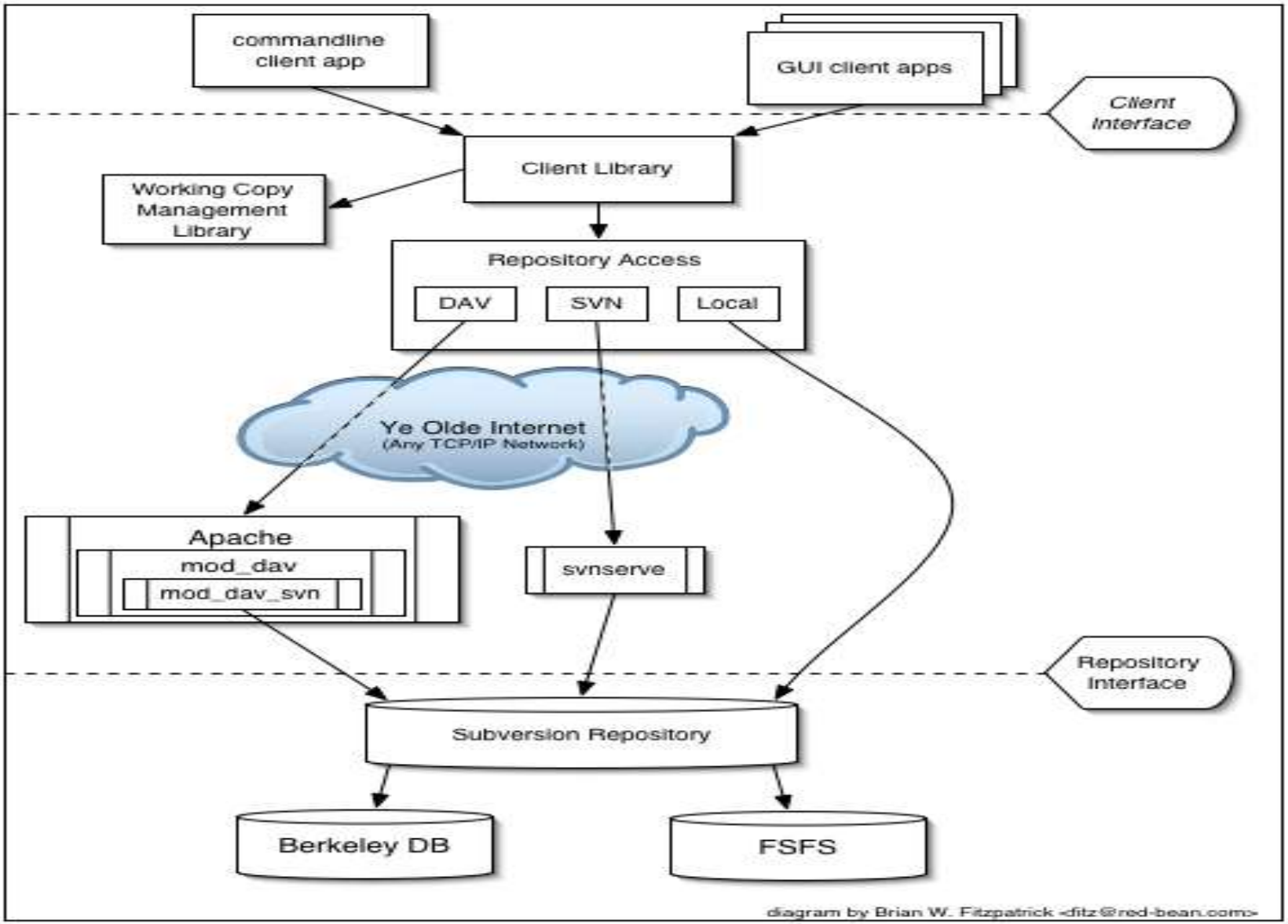
- **Versionamento de metadados**

Cada arquivo e diretório tem um conjunto de propriedades—chaves e seus valores—associados consigo. Você pode criar e armazenar quaisquer pares chave/valor que quiser. As propriedades são versionadas ao longo do tempo, tal como os conteúdos de arquivo.

- **Adaptabilidade**

O Subversion não tem qualquer bagagem histórica; ele é implementado como um conjunto de bibliotecas C compartilhadas com APIs bem definidas. Isto torna o Subversion extremamente manutenível e usável por outras aplicações e linguagens.

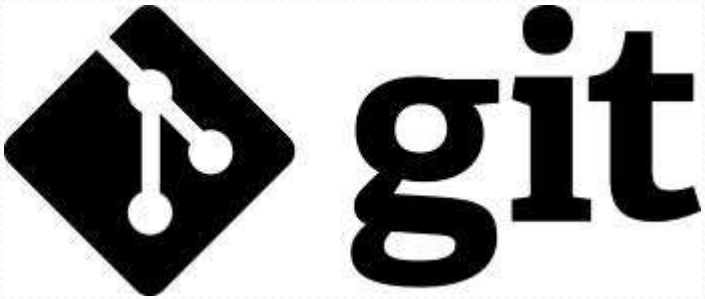
Arquitetura do subversion



GIT

- O Git é um sistema de controle de versão, projetado basicamente para facilitar a vida de quem quer executar projetos em equipe, permitindo que duas ou mais pessoas trabalhem juntas.

- Também é utilizado por quem trabalha sozinho, devido a possibilidade de 'controlar' as versões do projeto.



Linus Torvalds
Criador

Junio Hamano
Mantenedor

Outras fontes de informação:

- http://git-scm.com/
- http://book.git-scm.com/
- https://git.wiki.kernel.org/
- http://en.wikipedia.org/wiki/Git_%28software%29

GPL V2

Projetos que utilizam o Git

PostgreSQL

Hospedagem

Gratuitos	Pagos	Híbrido
GITORIOUS	unfuddle	Atlassian bitbucket
repo.or.cz	codebase	github SOCIAL CODING
sourceforge		
Google code		
Savannah		

Mercurial

- **Mercurial** é uma ferramenta multi-plataforma de controle de versão distribuído para desenvolvedores de software. O sistema é implementado principalmente em Python e C.

- Mercurial foi inicialmente escrito para rodar sobre Linux, mas foi portado para Windows, Mac OS X, e a maioria dos outros sistemas UNIX.

Matt Mackall
Criador e Mantenedor

mercurial

Download now
Mercurial **2.0.2**
TortoiseHg 2.2.2 with

Another OS?
Get Mercurial for:
- » Mac OS X
- » Windows
- » other

Release 1 janeiro

Outras fontes de informação:

- http://mercurial.selenic.com/
- http://mercurial.selenic.com/wiki/
- http://en.wikipedia.org/wiki/Mercurial
- http://hgbook.red-bean.com/read/

GPL V2

Projetos que utilizam o Mercurial

mozilla
FOUNDATION

OpenJDK

symbian
OS

OpenOffice.org™

Hospedagem

Gratuitos	Pagos	Híbrido
sourceforge	codebase	Atlassian bitbucket
Google code	Kiln	CodePlex Open Source Project Community
Savannah		

CANONICAL

Martin Pool
Criador e Mantenedor

Bazaar

Stable version: 2.4.2 27/10/2011
Test version: 2.5b5 20/01/2012

Downloads

Bazaar runs on **Windows**, **Ubuntu**, **Debian**, **Red Hat**, **SUSE**, **OS X**, **FreeBSD**, **Solaris**, **Gentoo** and more.

Get Bazaar

Outras fontes de informação:

- http://bazaar.canonical.com/en/
- http://wiki.bazaar.canonical.com/Documentation
- http://en.wikipedia.org/wiki/Bazaar_%28software%29

GPL V2

Projetos que utilizam o Bazaar

debian

ubuntu

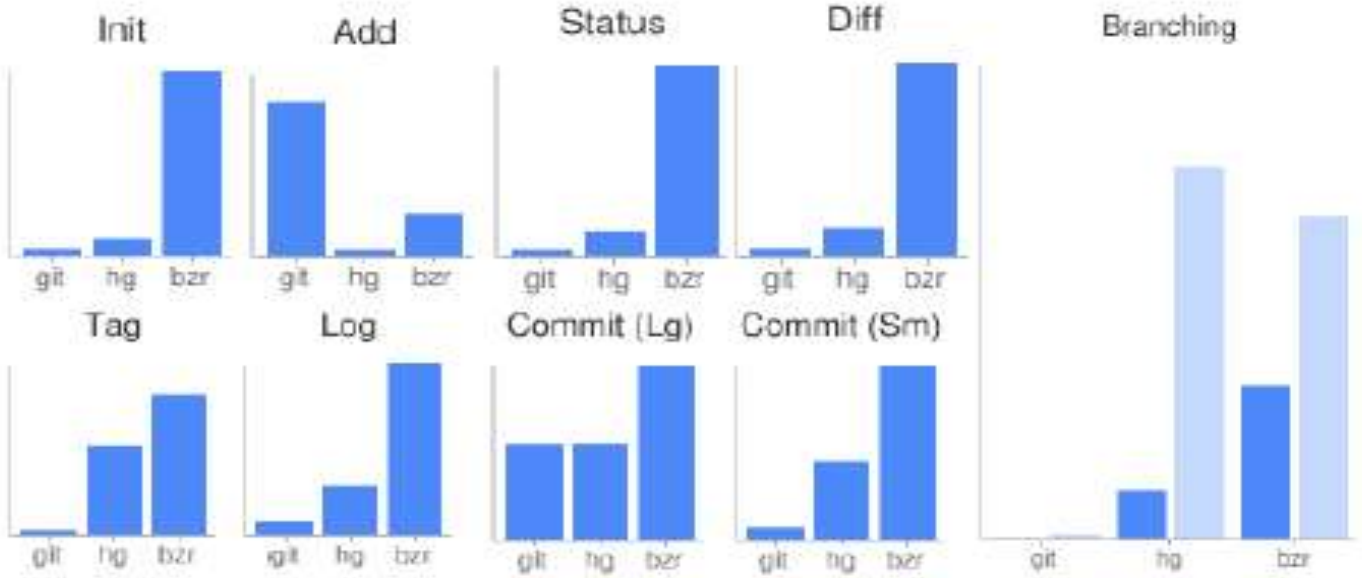
Hospedagem

Gratuitos	Pagos	Híbrido
launchpad	bzr.bz	
sourceforge		
 Savannah		

Estudo comparativo

VS

Git vs Hg vs Bzr: Velocidade



	Git	Hg	Bzr
Init	0.024s	0.059s	0.600s
Add	8.535s	0.368s	2.381s
Status	0.451s	1.946s	14.744s
Diff	0.543s	2.189s	14.248s
Tag	0.056s	1.201s	1.892s
Log	0.711s	2.650s	9.055s
Commit (Grande)	12.480s	12.500s	23.002s
Commit (Pequeno)	0.086s	0.517s	1.139s
Branch (Frio)	1.161s	94.681s	82.249s
Branch (Quente)	0.070s	12.300s	39.411s

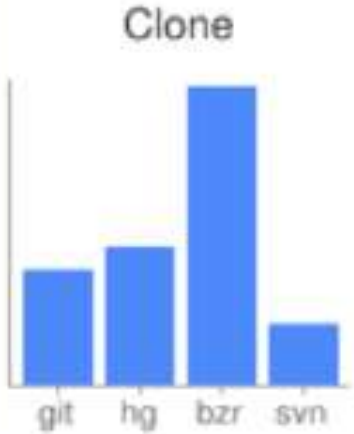
Git vs Hg vs Bzr: Tamanho do Repositório

	Git	Hg	Bzr	Bzr*	SVN
Só o Repo	24M	34M	45M	89M	
Diretório Inteiro	43M	53M	64M	108M	61M

Bzr* depois de rodar o Bzr pack

http://pt.whygitisbetterthanx.com

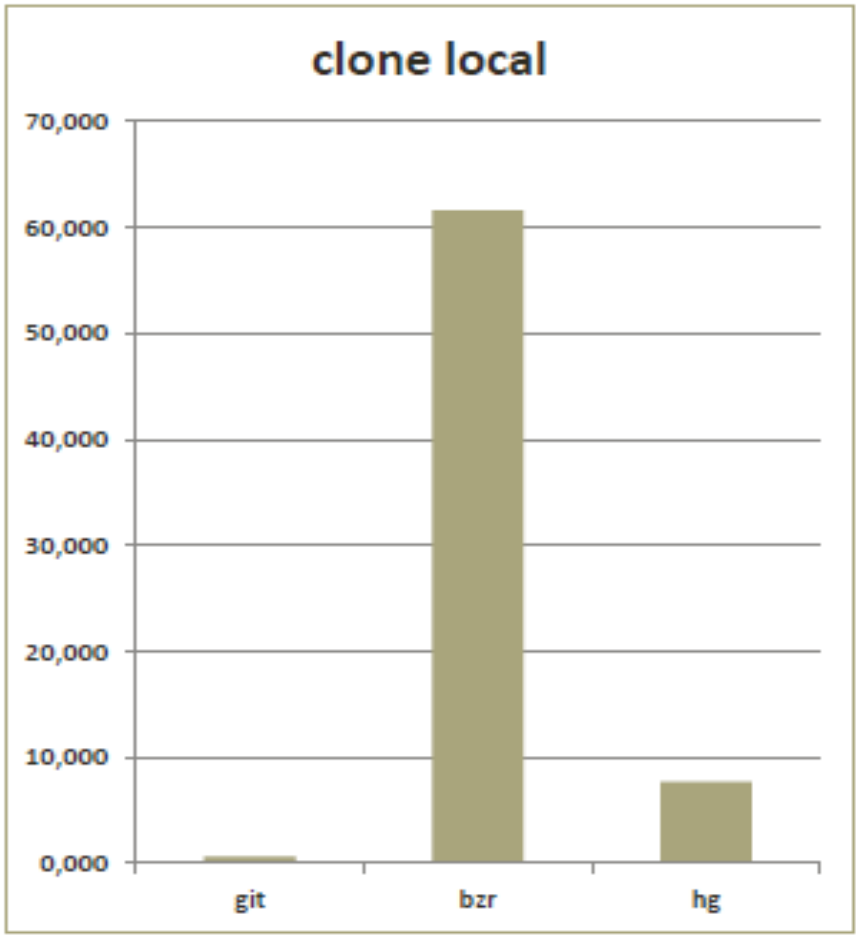
Git vs Hg vs Bzr: Tempo de Clone

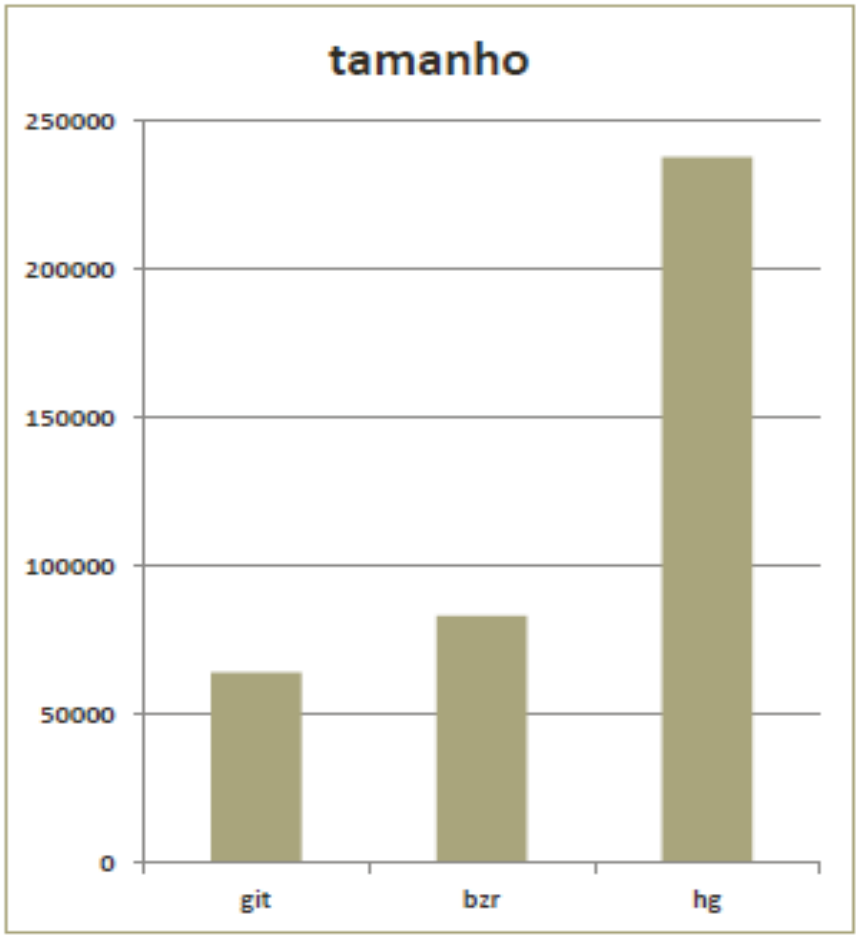


Git	1m 59s
Hg	2m 24s
Bzr	5m 11s
SVN	1m 4s

SVN compardo com o checkout

Git vs Hg vs Bzr: Tamanho do Repositório e Tempo de Clone





Testes realizados por Igor F. Steimacher, Igor S. Wiese e Ivanilton Polato em 01/02/2012

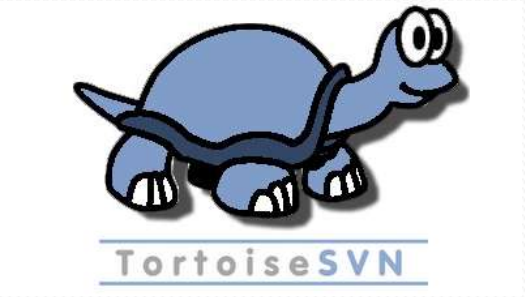
ESTUDO DE CASO: Jabil do Brasil

- **Empresa:** Jabil do Brasil
- **Problema:** Falta de sincronia no desenvolvimento de projetos
- **Impacto:** Quantidade excessiva de retrabalho e falha no cumprimento de prazos

ESTUDO DE CASO: Jabil do Brasil

- **Solução Inicial:** Utilização de ferramenta SVN (Tortoise)

- **Problemas:**
- Falta de conhecimento técnico no uso da ferramenta
- Resistência à mudanças por parte dos desenvolvedores



ESTUDO DE CASO: Jabil do Brasil

- **Solução:** Treinamento no uso da ferramenta

- **Impactos:**
- Melhorias no processo de integração de módulos dos sistemas
- Agilidade incrementada na entrega dos projetos

ESTUDO DE CASO: Jabil do Brasil

- **Novo desafio:** Desenvolvimento e implantação de sistemas em escala global

- **Problemas:** A solução atual não atendia as necessidades

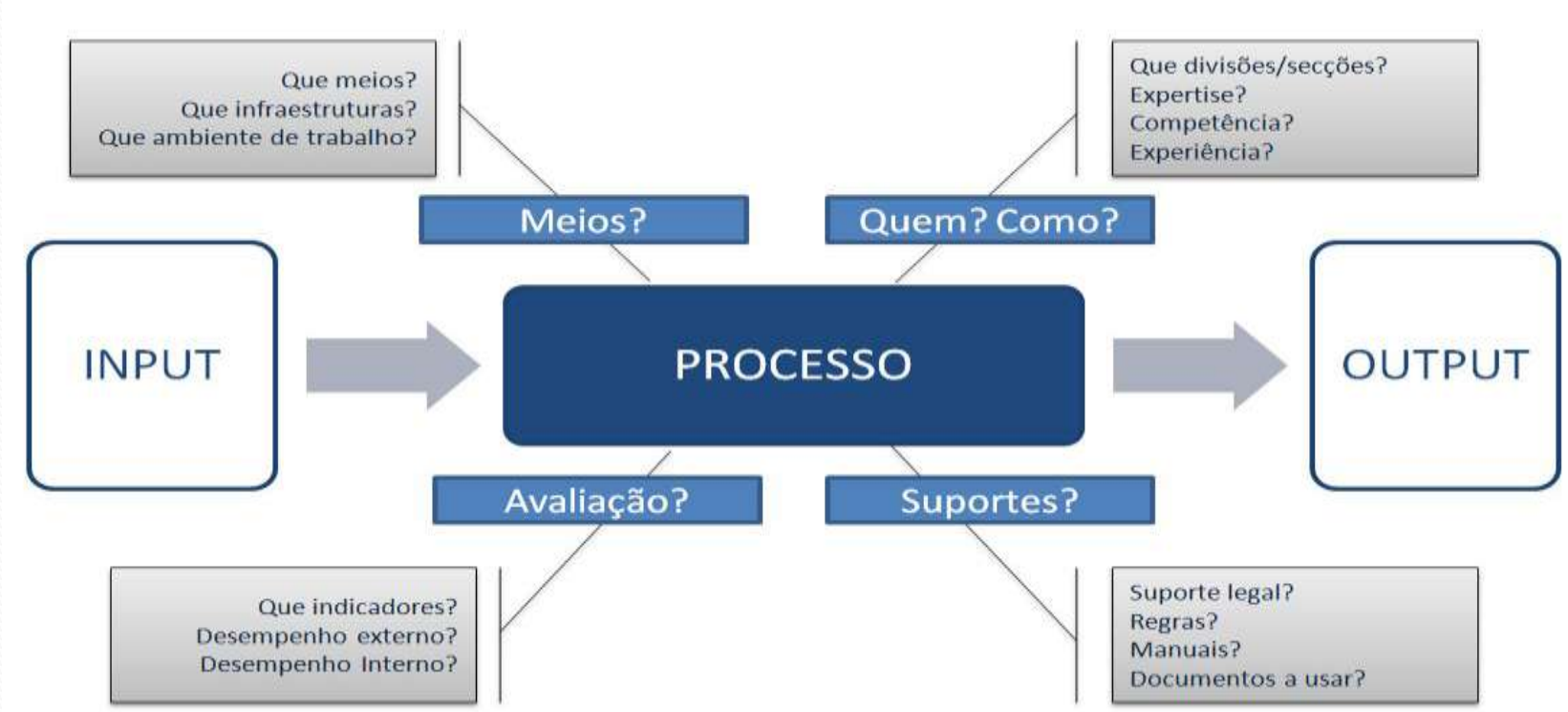
- **Impactos:**
- Dificuldade no controle dos módulos produzidos
- Impossibilidade da implantação do mesmo em outras filiais.

ESTUDO DE CASO: Jabil do Brasil

- **O problema estava na ferramenta?**

ESTUDO DE CASO: Jabil do Brasil

- **O problema estava nos processos e metodologias como um todo**



ESTUDO DE CASO: Jabil do Brasil

- **Solução:** Padronização dos processos como um todo (best practices), assim como ferramentas de auxílio para desenvolvimento distribuído.

- **Impactos:**
- Possibilidade de trabalho conjunta com outras filiais.
- Criação e implantação de sistema em escala global.

Sistema IT Spare Control

Considerações Finais

- Controle de versão resolve diversos problemas intrínsecos ao desenvolvimento de software. É uma prática de engenharia de software comprovadamente eficaz. Por isso, faz parte das exigências para melhorias do processo de desenvolvimento de certificações tais como CMMi, MPS-Br e SPICE.

- Existem várias ferramentas disponíveis para controle de versão. Para o controle de versão centralizado, é recomendado o Subversion. Para quem deseja usar o controle de versão distribuído, o Mercurial e o Git são as mais recomendadas.

Referências

- SVN

http://svnbook-pt-br.googlecode.com/svn/snapshots/1.4/svn.intro.whatis.html

- Comparação

http://pronus.eng.br/blog/http:/pronus.eng.br/blog/comparacao-entre-subversion-mercurial-e-git-parte-1

- Outras Informações

http://www.slideshare.net/IvaniltonPolato/gerenciadores-de-controle-de-verso-git-mercurial-e-bazaar

http://www.pronus.eng.br/artigos_tutoriais/gerencia_configuracao/conceitos_basicos_controle_versao_centralizado_e_distribuido.php