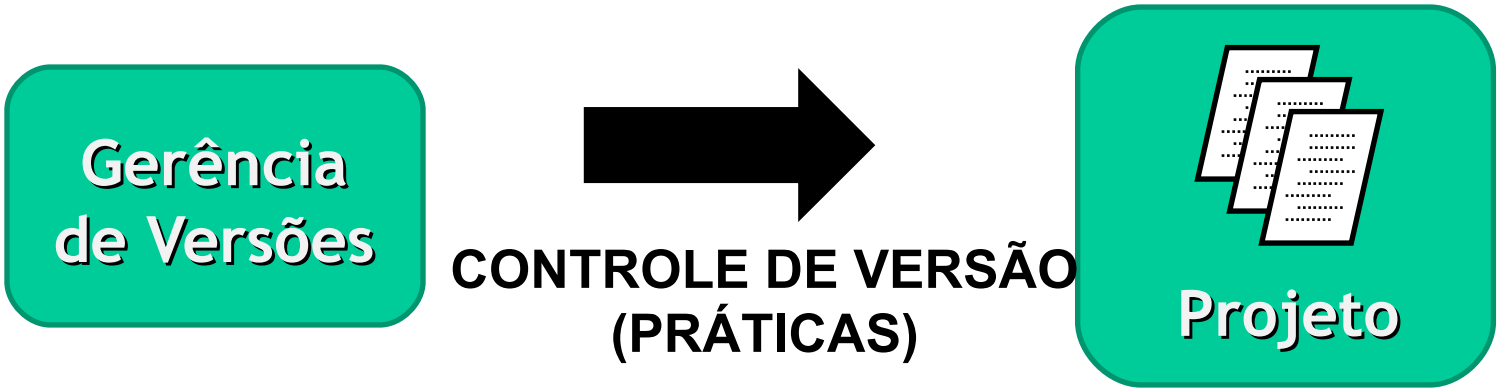
Sistemas de Controle de Versão

Controle de versão

- **Controle de versão**
  - *"conjunto de práticas cujo objetivo principal é manter controle sobre as modificações efetuadas em um determinado (conjunto) arquivo."*
  - Utilizadas no contexto de uma **Gerência de Versões**
  - Aplicados aos arquivos de um **Projeto**.



Sistemas de Controle de Versão (SCV)

- Ferramenta de apoio ao Gerenciamento de Versões
  - De documentos, códigos fontes, etc.

- Permitem
  - Automatizar o processo de controle de versão.
  - Gerenciar o histórico, modificações e diferentes versões;
  - Que várias pessoas trabalhem simultaneamente em um mesmo documento;
  - Controlar as versões através de tags;
  - Visualizar diferenças entre as versões;
  - *merging* entre versões conflituosas;

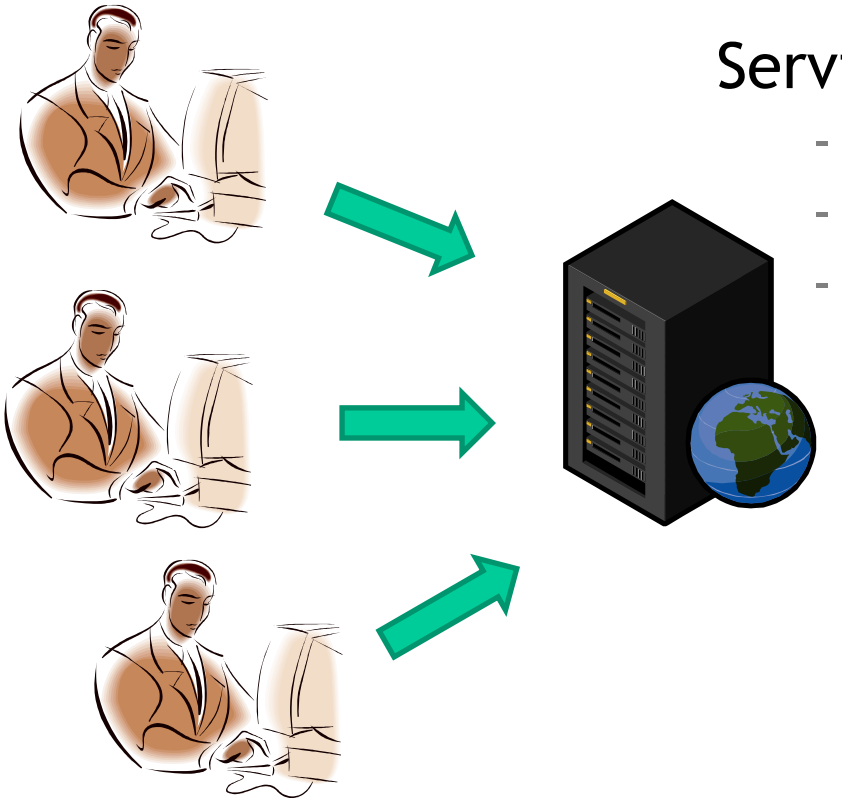
O que não um SCV é?

- Mecanismo para backup
- Ferramenta para a construção de builds
- Substituto para comunicação entre desenvolvedores
  - Conflitos não são resolvidos automaticamente
- Ferramenta de solicitação de mudanças
  - Não controla requisições de mudança

SCVs não são a solução para todos os problemas!

SCVs são utilizados dentro de um contexto maior, e possuem um papel específico.
Veremos outras ferramentas, com outros papeis, ainda neste curso!

Arquitetura Cliente/Servidor



Servidor mantêm

- **Repositório centralizado**
- arquivos a serem controlados,
- histórico de suas mudanças, log, etc.

Obs.: A maioria dos servidores armazenam apenas as diferenças (deltas) entre versões sucessivas de um determinado arquivo (as somas das diferenças a produzem versão mais nova).

Clientes mantêm

- **Área de trabalho** (Workspace)
- cópia dos arquivos do repositório

Principais SCV OpenSource

- Concurrent Versions System (CVS)
  - *Referência para todas as outras ferramentas subseqüentes.*
  - **Porém...**
    - tem perdido espaço para outras ferramentas
- Subversion (SVN): *Substituto melhorado do CVS.*
    - Versionamento de diretórios;
    - Commits atômicos;
    - Uso eficiente de rede.
- Git: SCV **distribuído**
    - Muito rápido
    - Repositórios nunca ficam inconsistentes
    - Sem a necessidade de servidores

Tendências

- **Grande procura pelo SVN e pelo Git**
- CVS está caindo em desuso
- Ferramentas comerciais
  - ClearCase, Perforce, Bitkeeper

continuam a ser adotadas apenas em empresas de Grande Porte.

String de busca: CVS version, SVN version, Git version

Comparação entre alguns SCVs

	CVS	SVN	ClearCase	Git
Licença	Open Source	Open Source	Comercial	
Formato Repositório	Arquivos RCS [1]	BD relacional		BD de objetos
Atomic Commit	Não	Sim	Sim	Sim
Copiar e Renomear Arquivos e Diretórios	Não	Sim	Sim	Sim
Merge Tracking	Não	Sim	Sim	Sim
Tags	Sim	Sim[2]		Sim
Conjunto de Comandos	Simples	Excelente	Excelente	Excelente
Deployment	Bom	Bom	Fraco [3]	Bom
Velocidade	Médio[4]	Muito Bom	Média [5]	Excelente
Portabilidade	Bom	Excelente	Médio	Bom

[1] Arquivos RCS podem ser alterados manualmente quando corrompidos, porém não suportam transações.
[2] Suportado através de copias.
[3] O ClearCase tem uma instalação difícil..
[4] Para suportar segurança, o CVS precisa ser tunelado dentro de outros protocolos.
[5] Servidor e clientes precisam estar na mesma rede para se obter uma performace aceitável.
*Principais Fontes: Wikipedia Comparison e Better SCM Comparison
Comparação detalhada: http://better-scm.shlomifish.org/comparison/comparison.html

Fluxo de Trabalho com SVC

Modelos de “Versionamento”

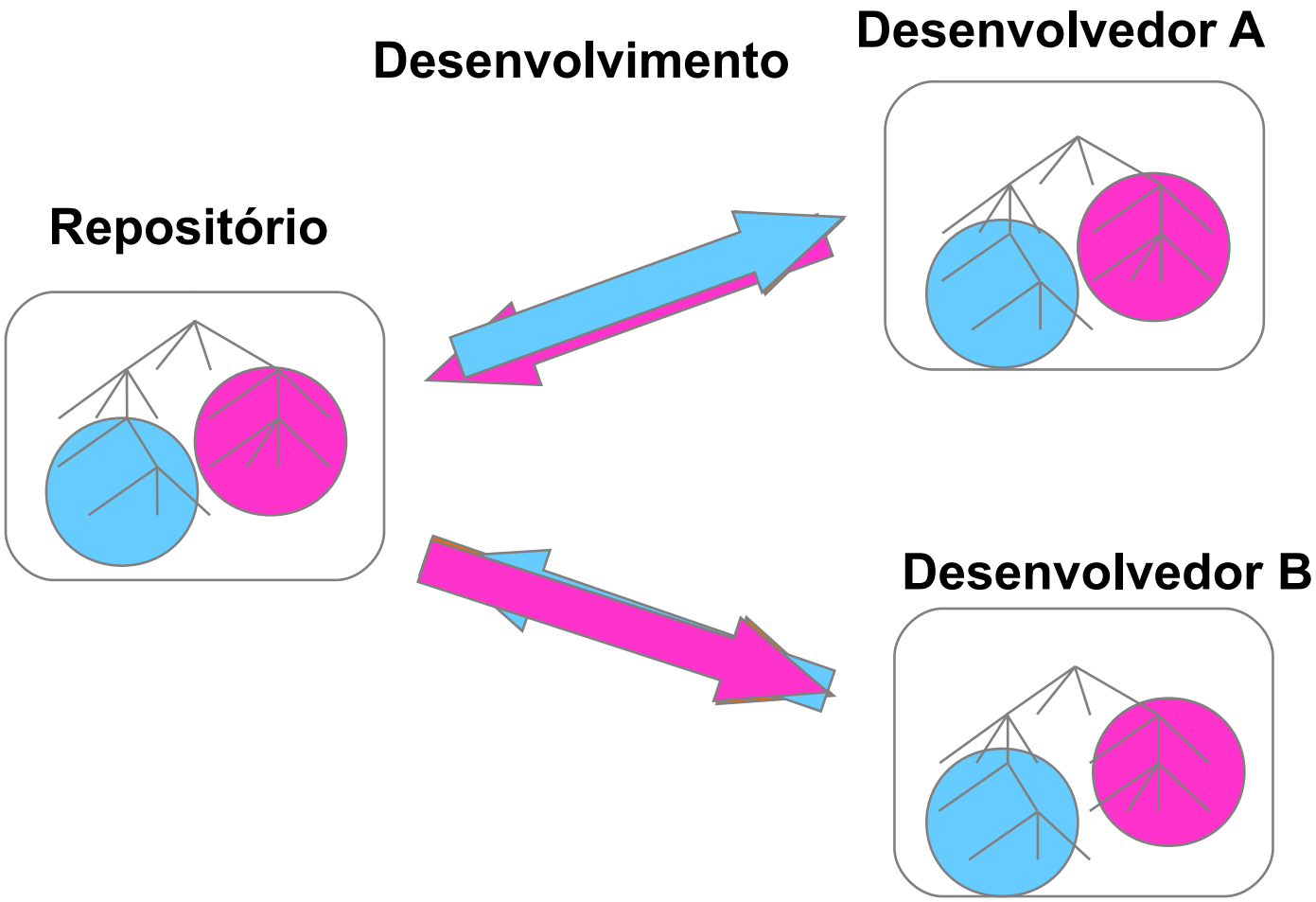
Lock-Modify-Unlock

- Falsa noção de segurança.
  - Mais problemas do que parece.
- Você só consegue alterar um arquivo se conseguir destravá-lo.
  - Desenvolvedores esquecem arquivos travados freqüentemente!
- Dificulta uso off-line.

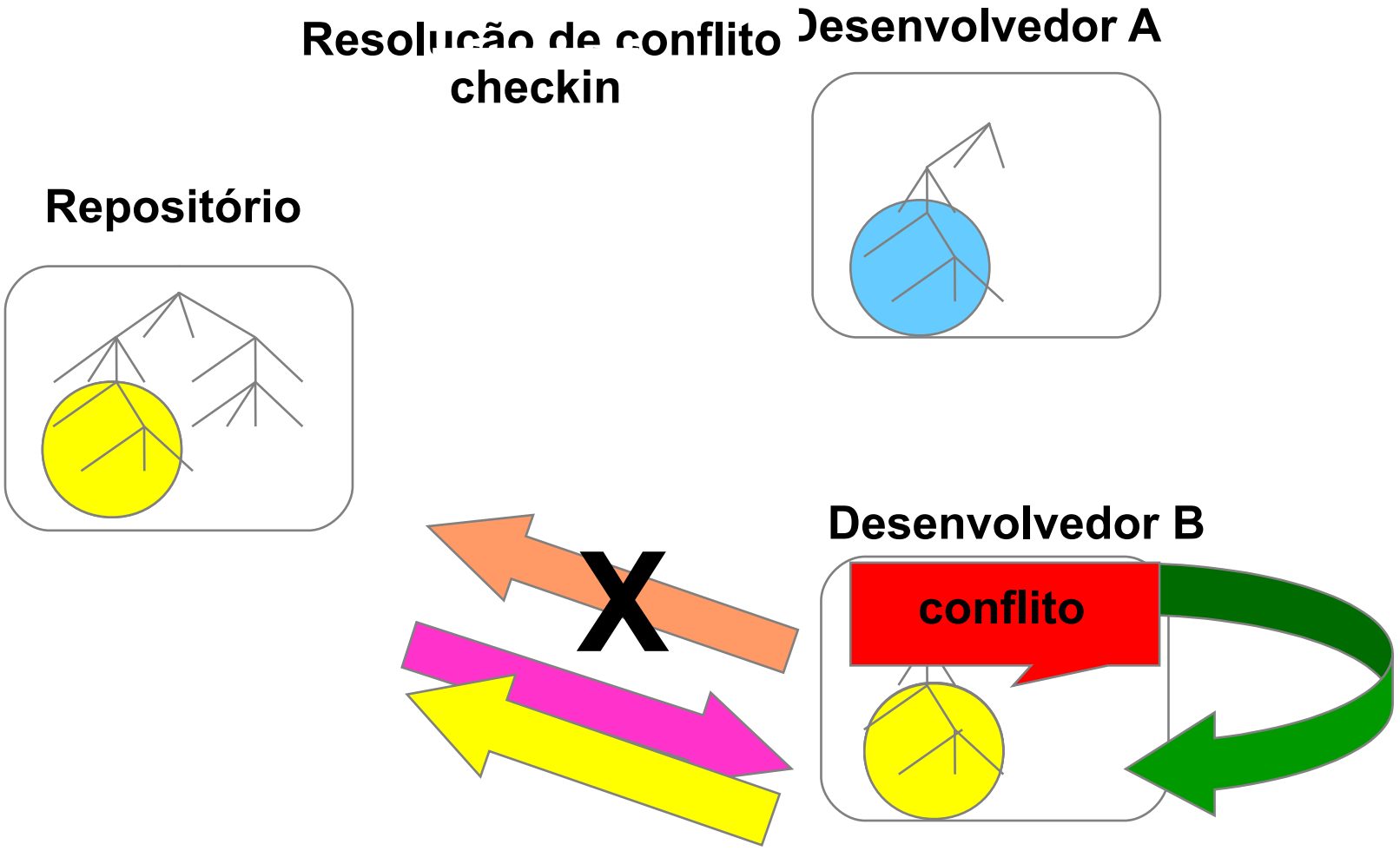
Copy-Modify-Merge

- Método recomendado
- Mais simples e prático.
  - menos problemático do que você imagina
- Desenvolvedores podem trabalhar
  - simultaneamente no mesmo arquivo.
  - Livremente em seu workspace
- Facilita o uso off-line.

Desenvolvimento ideal



Desenvolvimento real



Fluxo de Trabalho

- Comece sem nada.
  - Selecione um projeto em uma codeline (branch ou raiz) e selecione **Checkout;**
    - ou **update** com a versão mais atual do repositório se o projeto já existe localmente.
- Faça as mudanças.
  - Trabalhe localmente com o projeto, salvando as mudanças apenas na sua máquina;
- Sincronize, quando você estiver pronto
  - Update;
  - Examine as mudanças, faça as alterações necessárias;
  - Rode os Testes;
  - Commit;

Geração de releases

Baseline

- Identificação e empacotamento de artefatos entregues ao cliente (interno ou externo)
- Um release implica no estabelecimento de um novo baseline, de produto
- Garantia de que todos os itens de configuração foram devidamente testados, avaliados, aceitos
  - estão disponíveis no novo *baseline*

Release notes

- Relação de solicitações implementadas e testadas
- Automação parcial
- Comentários adicionais
  - Limitações atuais, problemas não resolvidos

Id	*Descrição*
1	Problema de performance na validação de pedido
2	Nova rotina de validação de crédito conforme normas de dezembro de 2002
…	…

Tipos de release

- Normalmente, releases estão associados aos *milestones* do plano de projeto
- Internos
  - Controle de qualidade, acompanhamento de projeto, controle de riscos, aceitação, aquisição de conhecimento através da coleta de feedbacks, desenho da estratégia de implantação
- Externos
  - Implantado e utilizado pelo cliente

Desafios

- Cultura organizacional
  - Agrupamento de solicitações em releases bem definidos e estabelecidos deve ser negociado com os *stakeholders* do projeto
  - Releases internos utilizados de forma efetiva como ferramenta de gestão de projeto
  - Cultura fundamentada no pragmatismo e automação de atividades repetitivas
- Integração entre sistemas de controle de versão e mudanças

Discussão

Alguns Fatos

- Mudança é inevitável
- Mudar é fácil – controlar diversas mudanças simultâneas é difícil
- A gerência de mudanças introduz controle sobre as mudanças de maior relevância
  - Todas as mudanças são analisadas
  - Apenas as aprovadas são realizadas
  - Sempre se sabe quem modificou o que, onde e quando

Tradeoffs

- Controle de mudanças
  - Nível de controle x overhead com o CCB
  - Defeitos de magnitude menor podem ser tratados pelo líder de projeto diretamente
  - Uso de versões draft de forma de minimizar controle de mudanças
- Automação do processo
  - Custo x benefício
  - Aquisição de ferramentas comerciais
  - Uso de ferramentas open source