Gestão de Configuração e Mudanças de Software

Allan Lima

Arquitetura, Design e Implementação de Sistemas para Internet
Pós Graduação
Faculdade 7 de Setembro

arglbr@gmail.com

nov/2015

Controle de Versões

Conceitos

Problemas comuns

- Quem alterou isto? Quando foi?
- Pensando bem, é melhor como estava semana passada
- Hein? Onde está o que eu fiz ontem?
- Não deu certo meu teste. Preciso deixar o projeto como estava.

Por que controlar versões?

- Manutenção do histórico do projeto
- Reproducibilidade
- Trabalho paralelo

Tipos de Controle de Versão

- Local
- Centralizado
- Distribuído

Controle de Versão Local

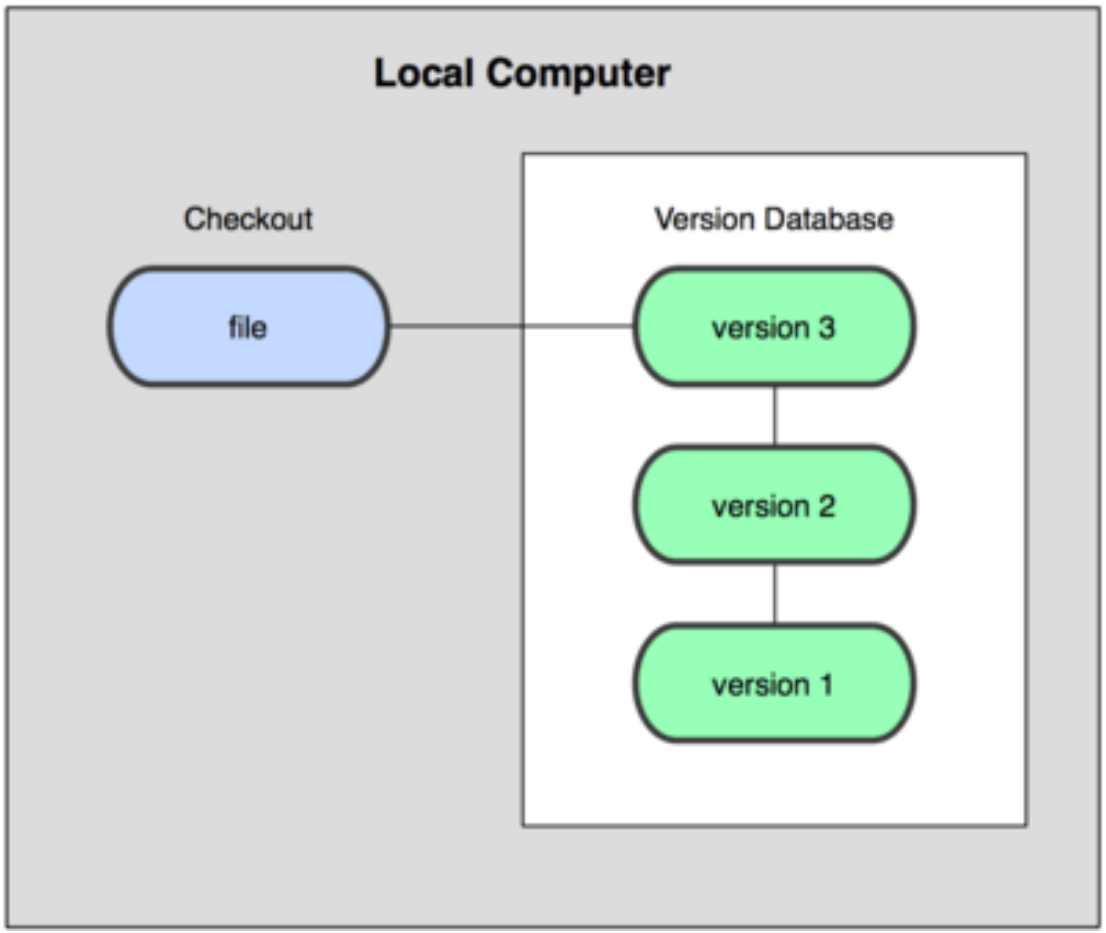


Figure: RCS, SCCS

Prós e Contras

- **Prós:** $\emptyset$, $\nexists$, $\{\}$, $lim \rightarrow 0$
- **Contras:** ultrapassado e limitado

Controle de Versão Centralizado

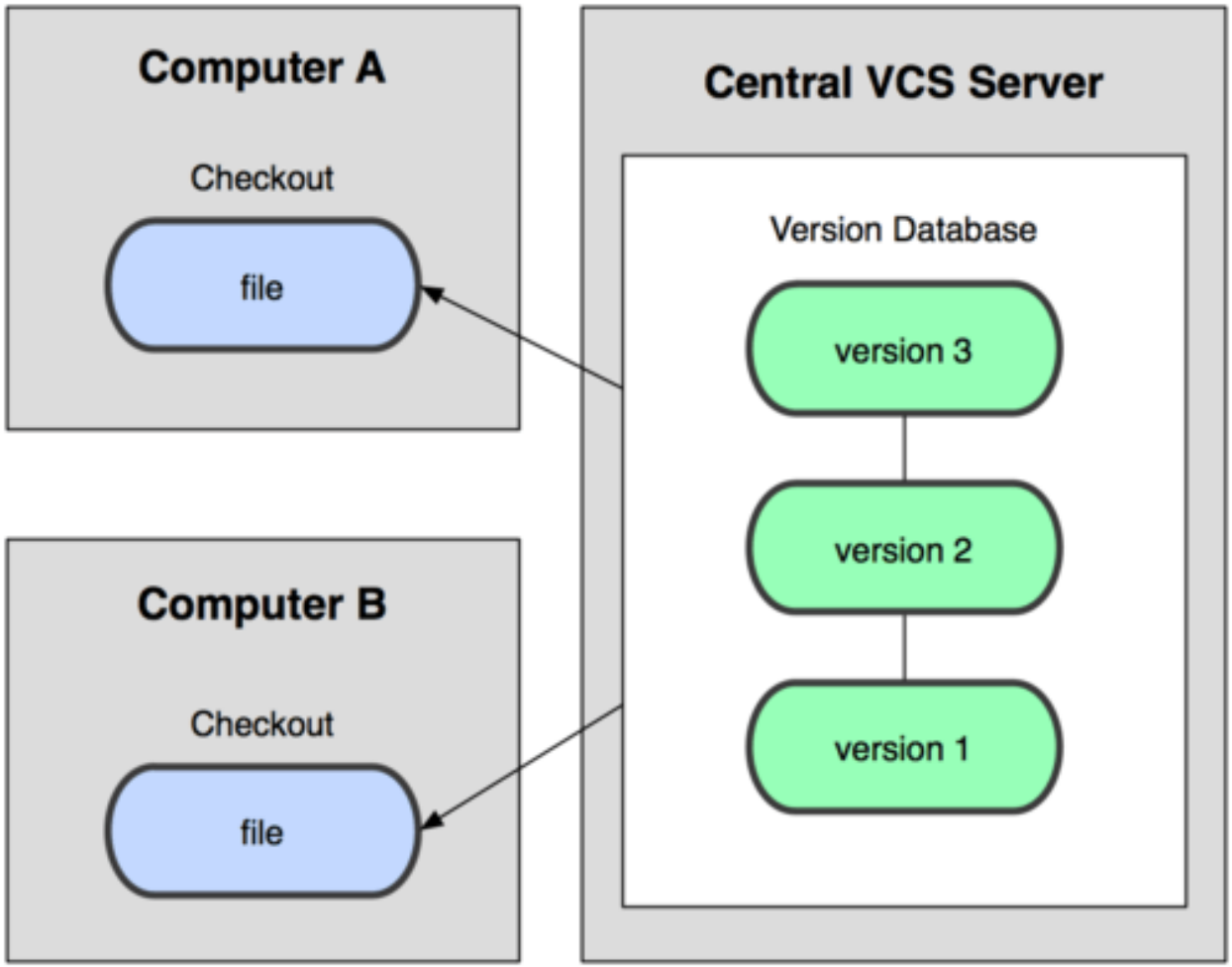


Figure: CVS, Subversion, Clearcase, Perforce, TFS - Team Foundation Service

Prós e Contras

- **Prós:** fácil compreensão do funcionamento; segurança
- **Contras:** ponto único de falha; escalabilidade; dependência de conectividade

Controle de Versão Distribuído

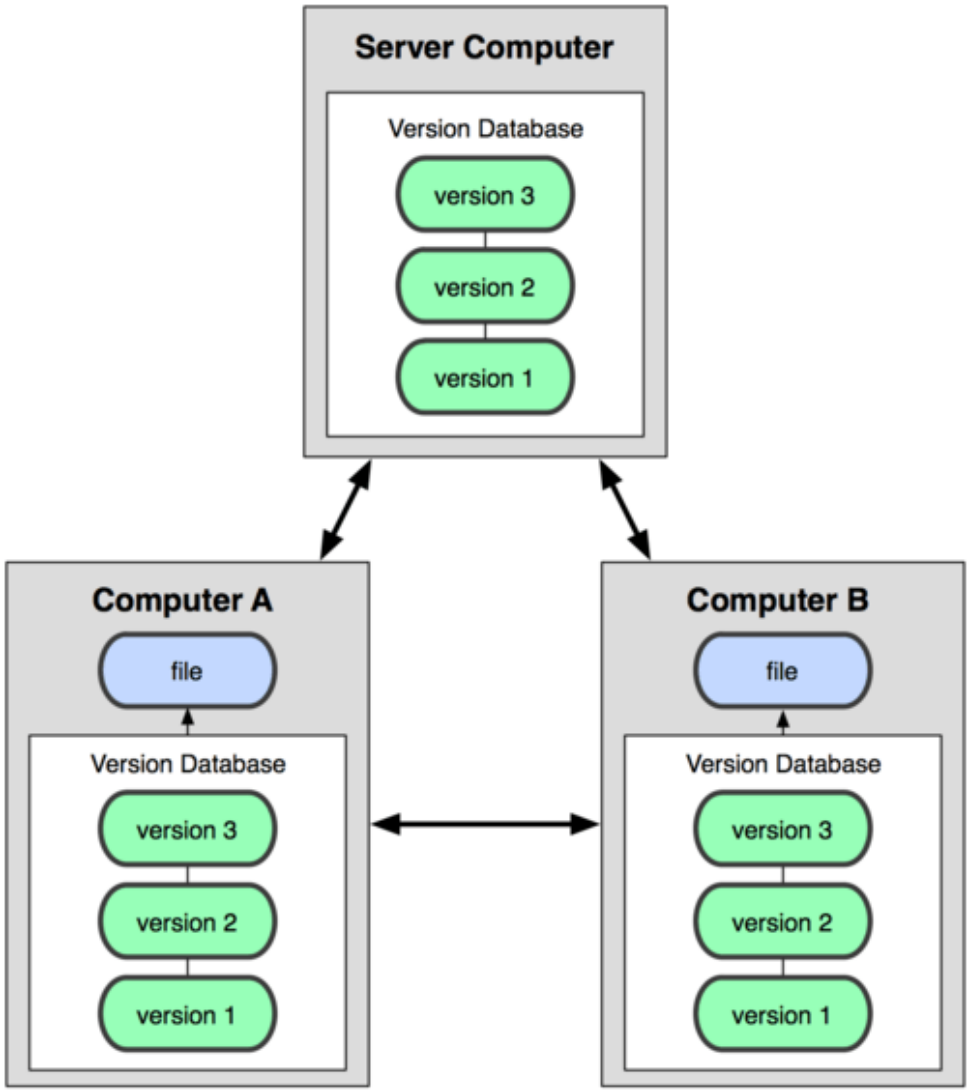
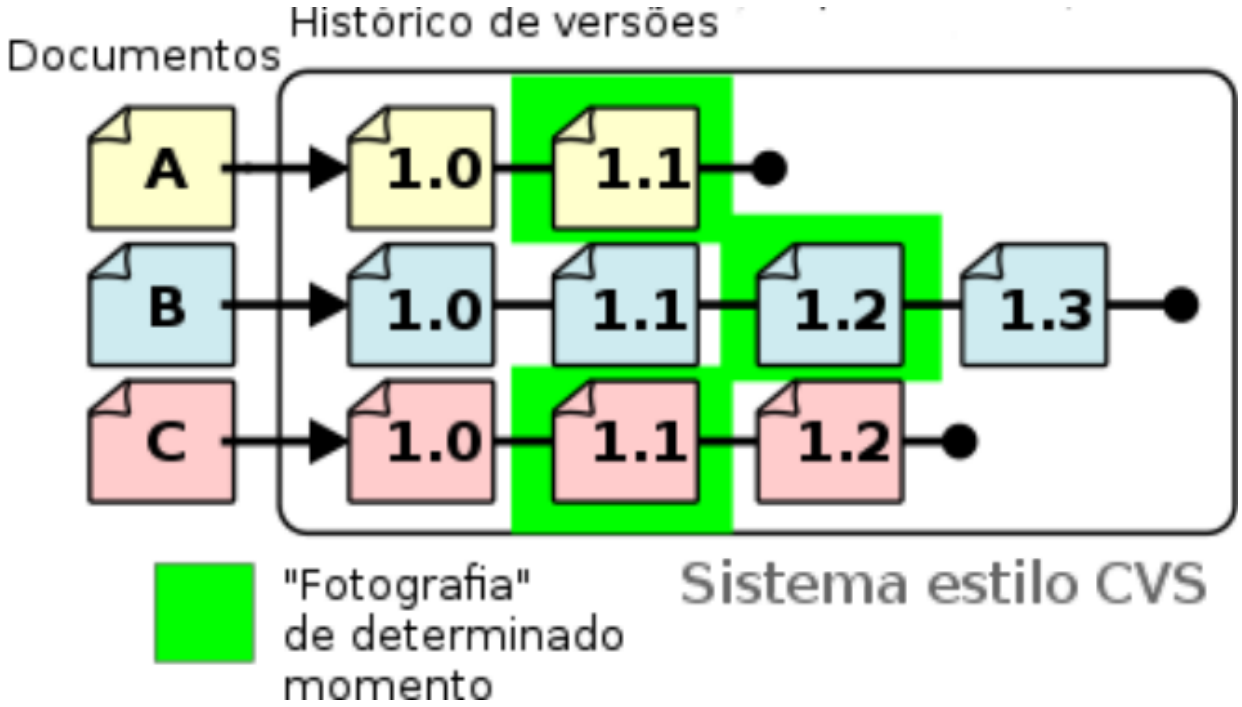


Figure: Git, IBM-RTC, Mercurial, Bazaar, Monotone, BitKeeper

Prós e Contras

- **Prós:** menor dependência de conectividade; elimina ponto central de falha; branches locais;
- **Contras:** segurança; “quem tem a versão mais atual?”; gerenciamento mais complexo;

Versionamento

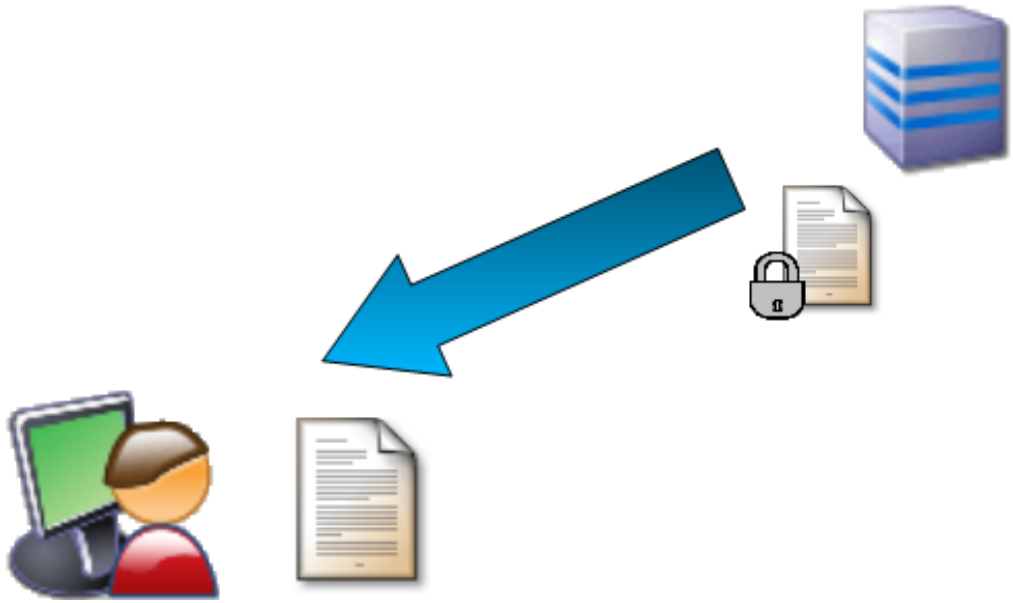


Ação: Adicionar ao Controle de Versões

- Adicionar um elemento ao sistema de controle de versões gerando a sua primeira versão
- Torna o elemento acessível aos usuários do repositório

Ação: *Checkout*

- Obter um elemento do sistema de controle de versões

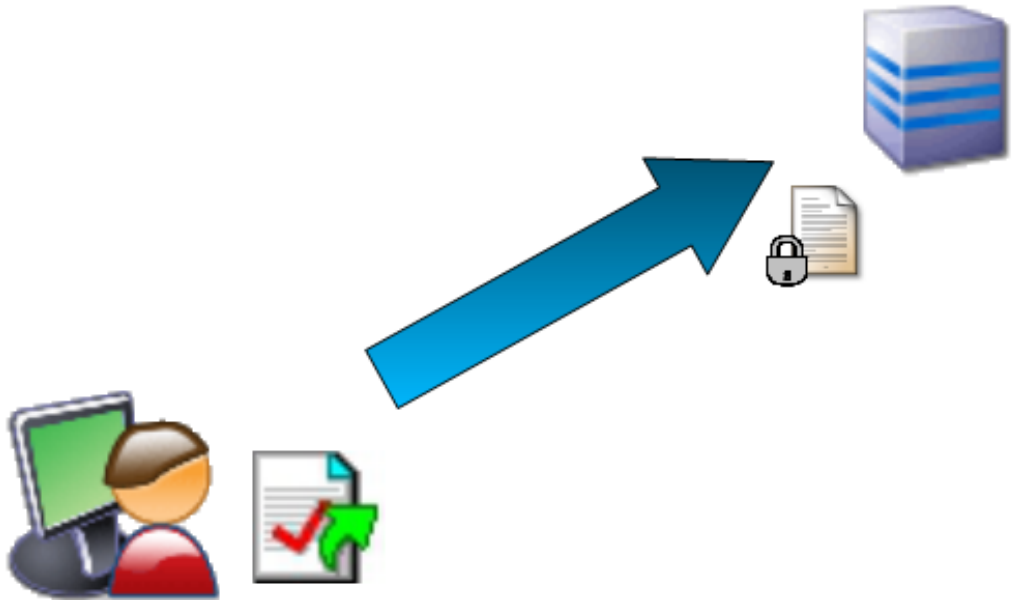


Ação: *Checkout*

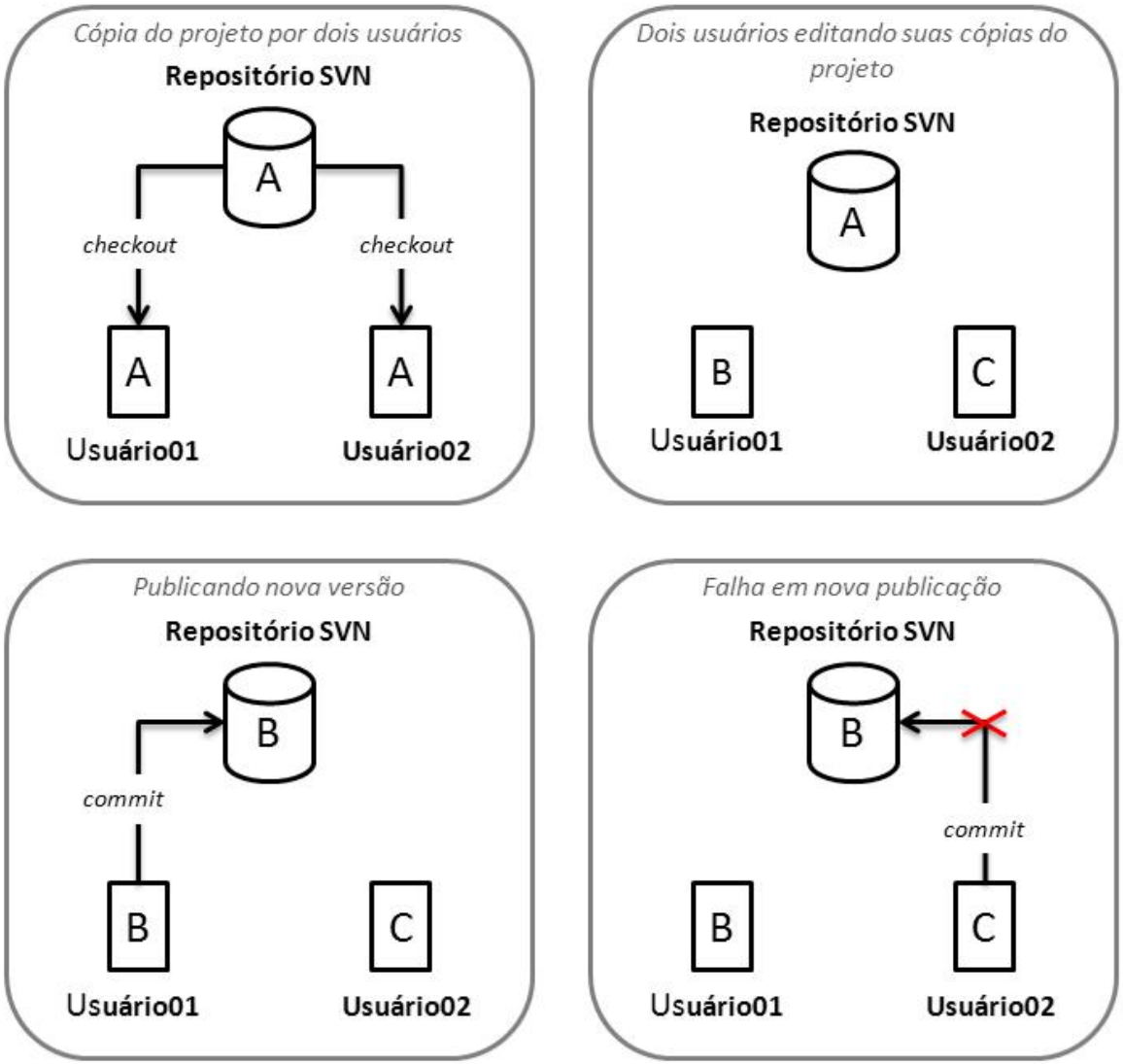
- Algumas ferramentas permitem dois tipos de checkout:
  - **Reservado** (*lock-modify-unlock*)
    - ↓ Somente um usuário terá a posse do arquivo, **não permitindo** o trabalho simultâneo.
    - ↑ Suas alterações irão constituir uma nova versão
  - **Não Reservado** (*copy-modify-merge*)
    - ↑ Vários usuários podem ter a posse do arquivo , **permitindo** o trabalho simultâneo.
    - ↓ Não é garantido que a sua versão será consolidada.

Ação: *Checkin*

- Gravar as alterações no repositório gerando uma nova versão do elemento
- Em algumas ferramentas chamado de *Commit*
- Em casos de *Checkout* Não-reservado pode gerar conflito de versões

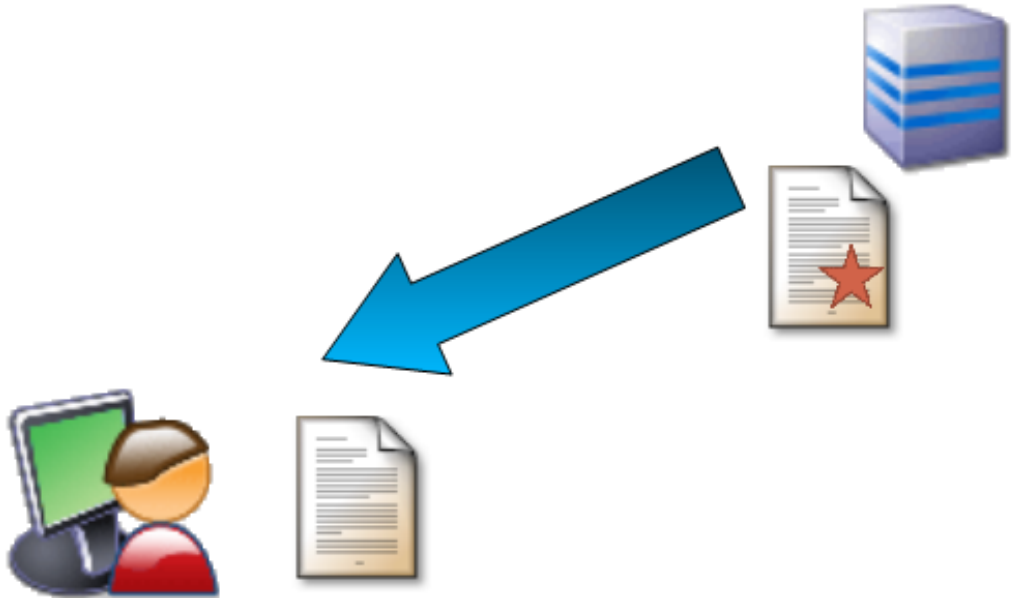


Ação: *Checkin* X Conflitos



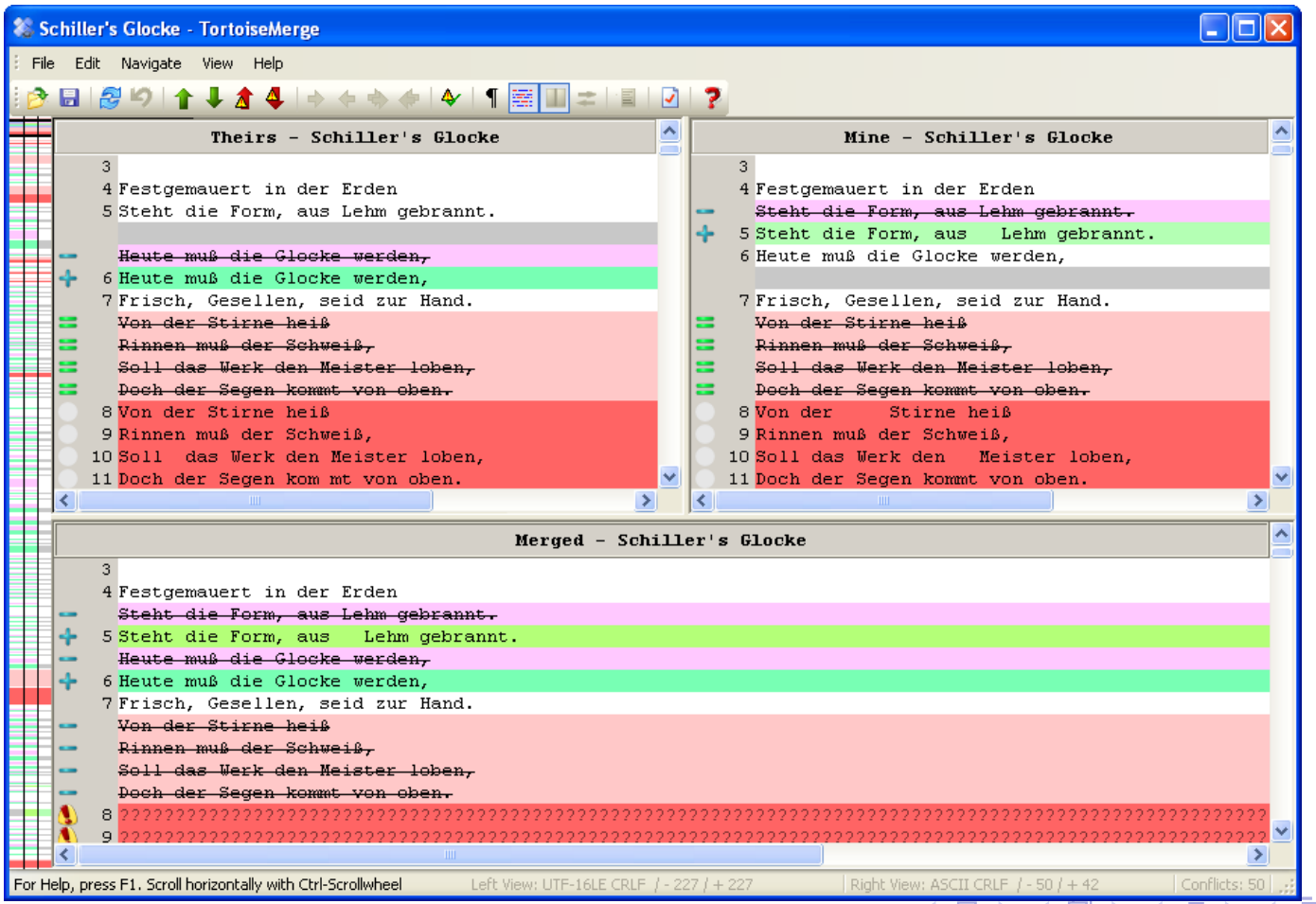
Ação: *Update*

- Obter uma versão mais recente do elemento do sistema de controle de versões
- Não confundir "atualizar" com "versionar"



Ação: *Merge* (Fusão)

- Consiste em verificar as diferencças entre as alterações feitas por usuário distintos sobre o mesmo arquivo a aplicar a fusão de conteúdo gerando uma nova versão



Ação: *Merge* (Fusão)

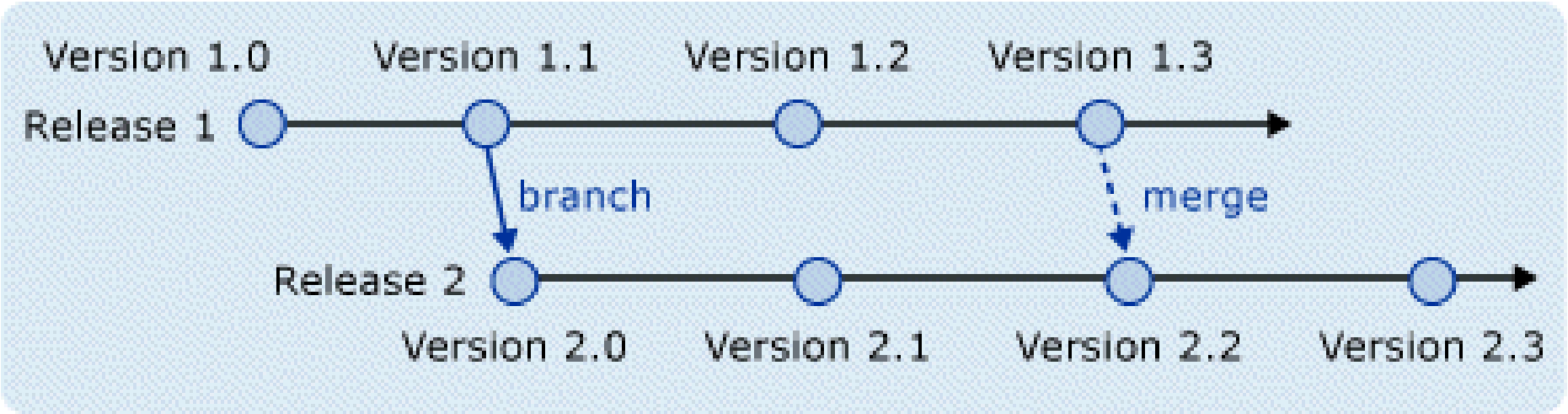
- Algumas ferramentas executam *Merge* Automático em casos onde não há conflitos (alterações no mesmo trecho de código)
- Em caso de conflito o usuário terá que executar o *merge* entre a versão dele e a versão remota (do repositório)

Ação: *Branching* (Ramificação)

- Consiste em criar ramos paralelos de desenvolvimento (*branches*)
- Por default, existe pelo menos um branch em um projeto, o Principal, também conhecido como *mainline* ou *trunk* ou *HEAD*.

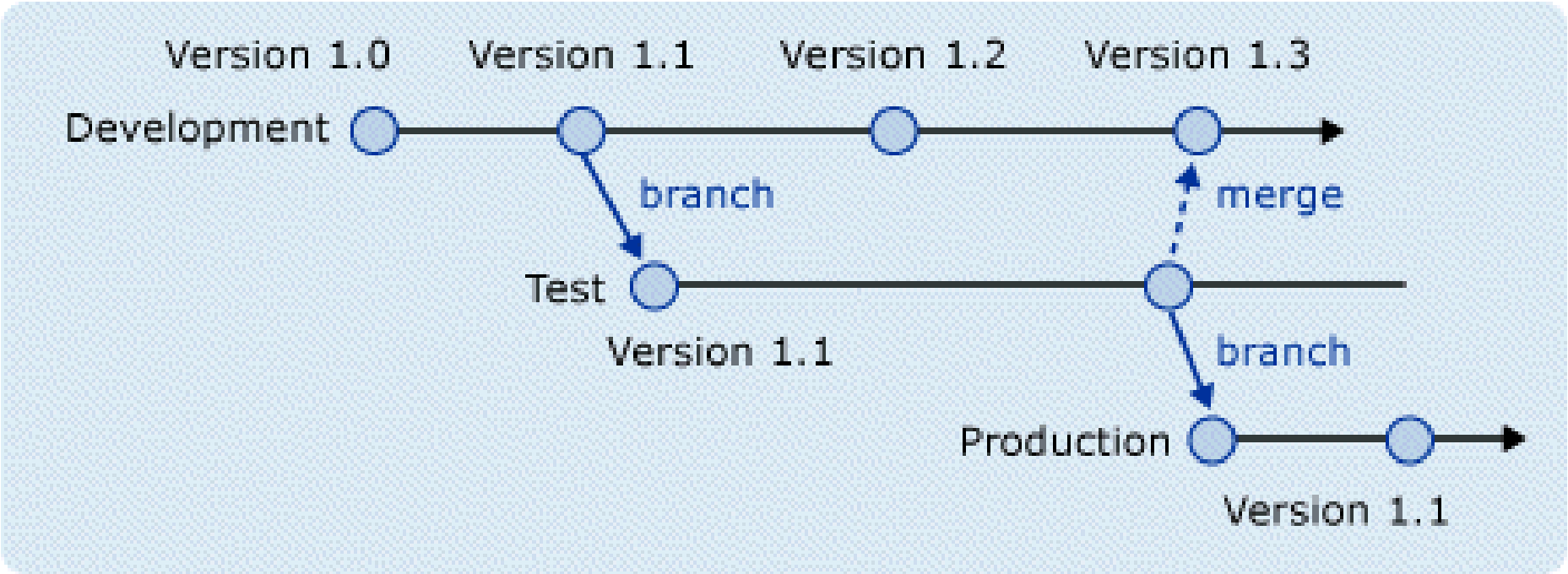
Branching: Estratégias

- *Release*



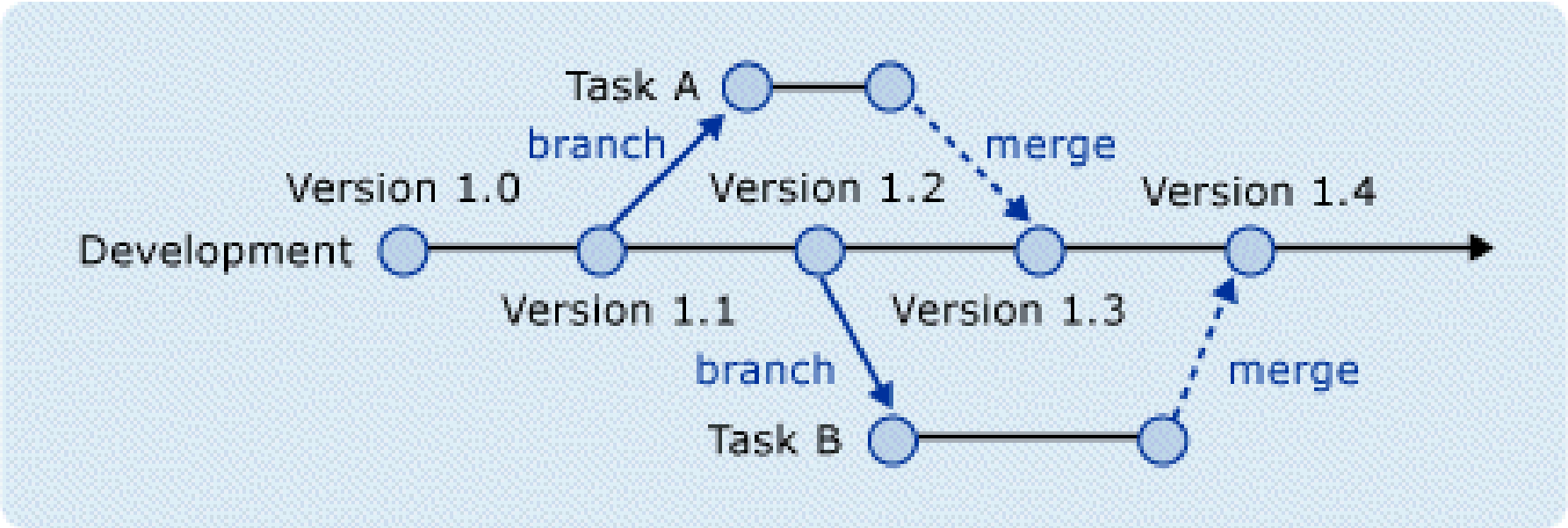
Branching: Estratégias

- Promoção de Código



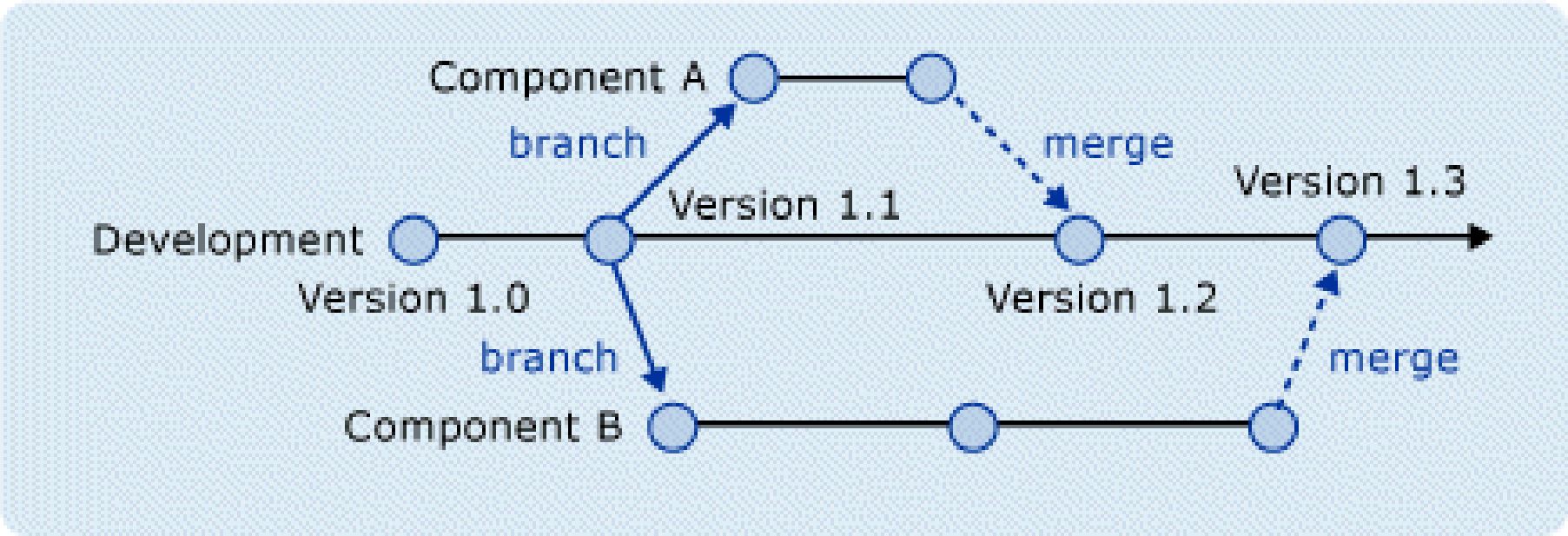
Branching: Estratégias

- Tarefas



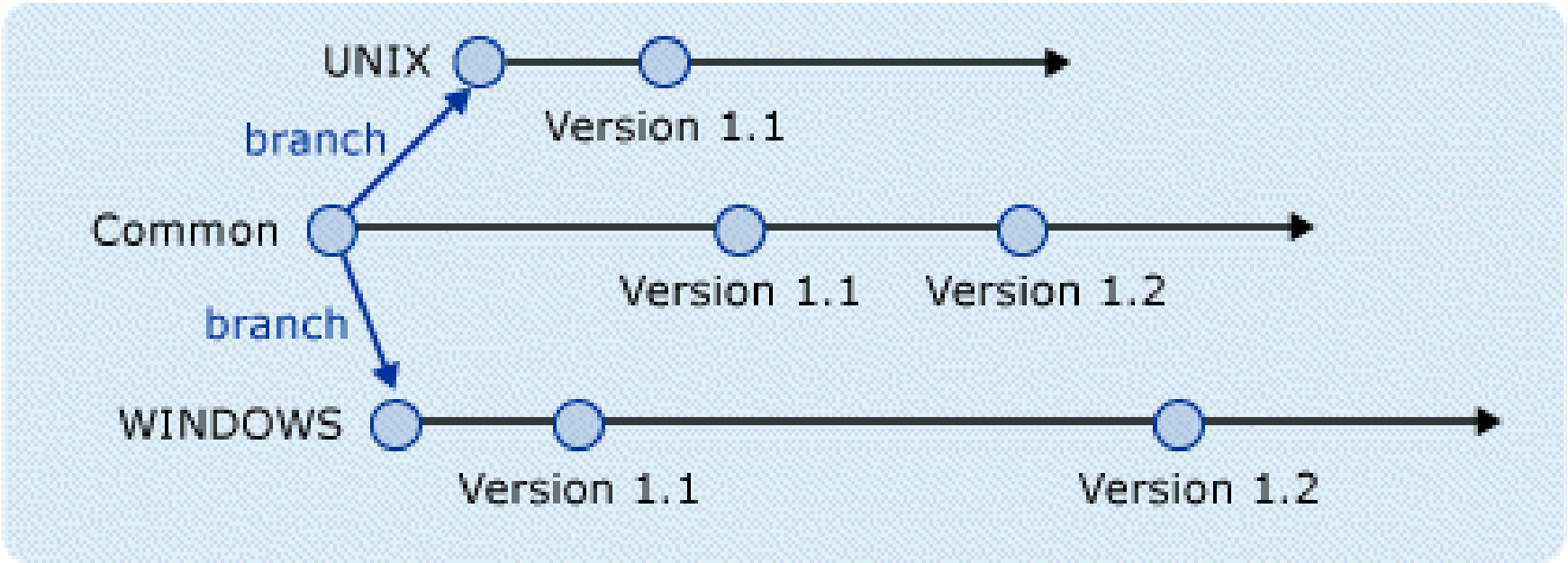
Branching: Estratégias

- Componente



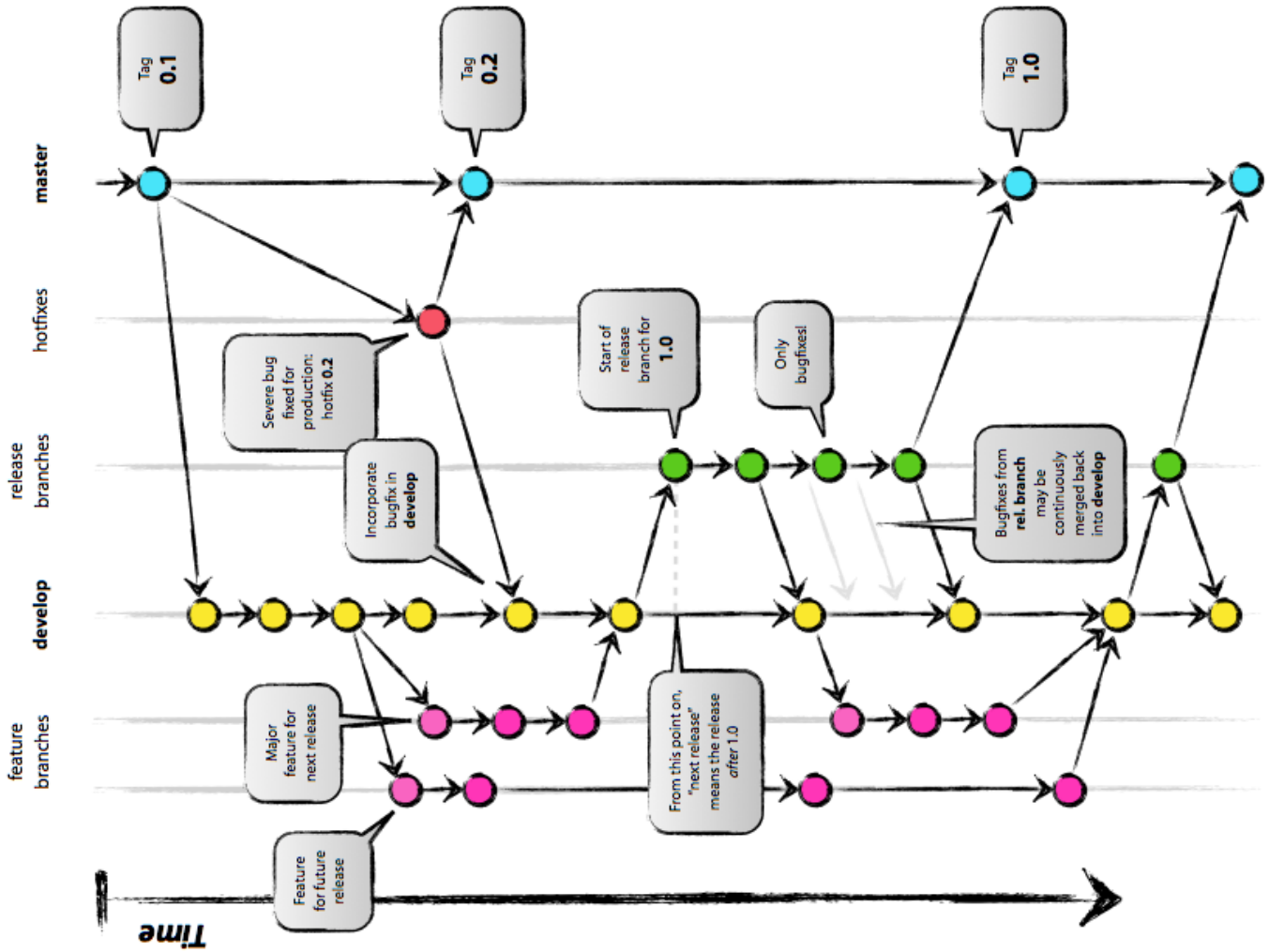
Branching: Estratégias

- Technologia



Branching: Estratégias

- Feature/Topic



Referências

- PRESSMAN, R. S., **Engenharia de Software**, 6ª. ed., 2006.
- HASS A. M. J., **Configuration Management Principles and Practice**, Addison Wesley, 432p, 2002.
- FREDERICKS T., **Software Configuration and Integration Management**, Marquette University, 2001.
- BIRMELE C., **Branching and Merging Primer**, Visual Studio 2005 Technical Articles, 2006.
- A successful Git branching model: `http://nvie.com/posts/a-successful-git-branching-model/`