

Gerência da Configuração

Controle de Versões

Prof. Márcia Pantoja.

marcia.pantoja@unama.br

Introdução

- Organização e controle na gestão de projetos de desenvolvimento de softwares podem fazer toda a diferença entre o sucesso ou fracasso do resultado.
- Implementações de melhorias durante a fase de desenvolvimento e, principalmente, após o lançamento do produto, exigem um controle rigoroso, já que podem afetar diretamente no funcionamento do sistema. Esse controle pode ser feito por meio do versionamento de software.

- **Exemplo**

- Estou em um projeto para um mercadinho e agora estou desenvolvendo uma página web em HTML com o intuito de listar todos os produtos do mercado. A princípio cheguei no seguinte resultado:

Mercadinho		Todos os produtos	Considere-nos
Todos os produtos			
Refrigerante - R\$ 5,99			
Sabão em pó - R\$ 9,99			
Arroz - R\$ 10,99			

- **Exemplo**

- Mostrando para o cliente essa primeira versão, ele falou o seguinte:
 - *"Está bem legal, porém, ficaria melhor se, no momento que clicar em algum produto, aparecer uma descrição e quantidade do mesmo."*

- **Exemplo**

- Partindo da necessidade do cliente, fiz alguns ajustes na página que lista todos os produtos, veja como ficou:

Todos os produtos

Refrigerante - R\$ 5,99
Descrição: Dolly Guaranará o melhor!
Quantidade: 100
Sabão em pó - R\$ 9,99
Arroz - R\$ 10,99


- **Exemplo**

- Apresentando essa segunda versão da lista de produtos para o cliente, ele falou o seguinte:
 - *"Está atendendo muito bem o que eu preciso! Porém, ainda fica difícil saber qual é o produto apenas lendo... Em outras palavras, quero que também adicione uma imagem do produto dentro dessa caixa da descrição e quantidade".*

- **Exemplo**

- Considerando essa nova necessidade do cliente, fiz os ajustes e cheguei a esse resultado:

Todos os produtos

Refrigerante - R\$ 5,99	
	<p>Descrição: Dolly Guarana o melhor!</p> <p>Quantidade: 100</p>
Sabão em pó - R\$ 9,99	
Arroz - R\$ 10,99	

- Exemplo

- Mostrei novamente para o cliente, e ele respondeu o seguinte:

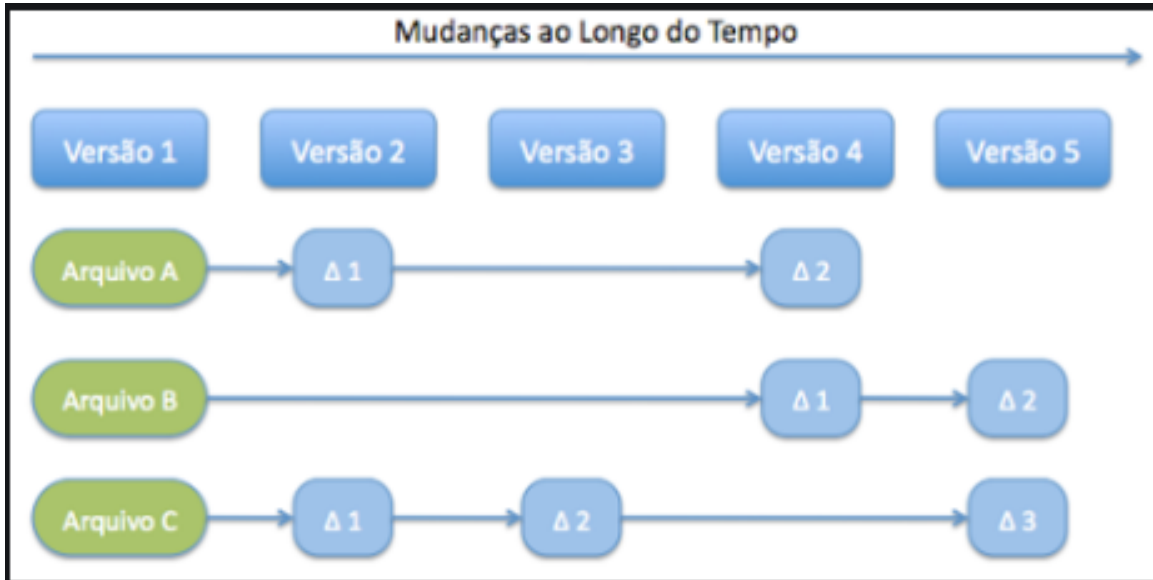
- *"Ficou muito legal a lista de produtos com a descrição, imagem e quantidade. Mas eu comecei a perceber que tende a ficar poluído, ou seja, é melhor deixar a lista como estava desde a primeira vez que você me mostrou, então você cria uma nova página que contenha as informações do produto: imagem, descrição e quantidade..."*

Introdução

- Observe que desde o começo do projeto, estamos sujeitos a mudanças constantes. Portanto, mesmo que entreguemos a *feature* para o nosso cliente da maneira como ele pede, ainda sim existe o risco dele querer mais alguma coisa.
- Além disso, perceba que também corremos o risco do nosso cliente simplesmente chegar e falar que queria o projeto do jeito como era antes, algo que fizemos há muito tempo atrás e que, dependendo da quantidade de mudanças, fica bem difícil de reverter...

Introdução

Pensando em todos esses detalhes, o que podemos fazer para evitar esse tipo de situação?



Versionamento

- O versionamento de software é um **processo de controle de versões** estabelecido por meio de numerações diferentes. Isso permite que os programadores saibam quando e quais alterações foram realizadas, acompanhando as mudanças aplicadas no software. Além disso, permite que os usuários finais identifiquem as novidades e reconheçam as versões mais atualizadas.

Versionamento

- Versionamento é a disciplina através da qual são preservadas as versões de um artefato, de modo sistemático e seguro, não limitado em número de versões.
- Versão:
 - Estado definido de um objeto num dado momento.
 - “Fotografia” do objeto.

Versionamento

- Para o versionamento, pode-se atribuir um número único ou um conjunto deles para especificar a versão utilizada de um programa de software, arquivo, firmware, driver de dispositivo ou mesmo hardware. À medida que as edições e atualizações vão sendo implementadas no programa, o número da versão deve aumentar.

Versionamento

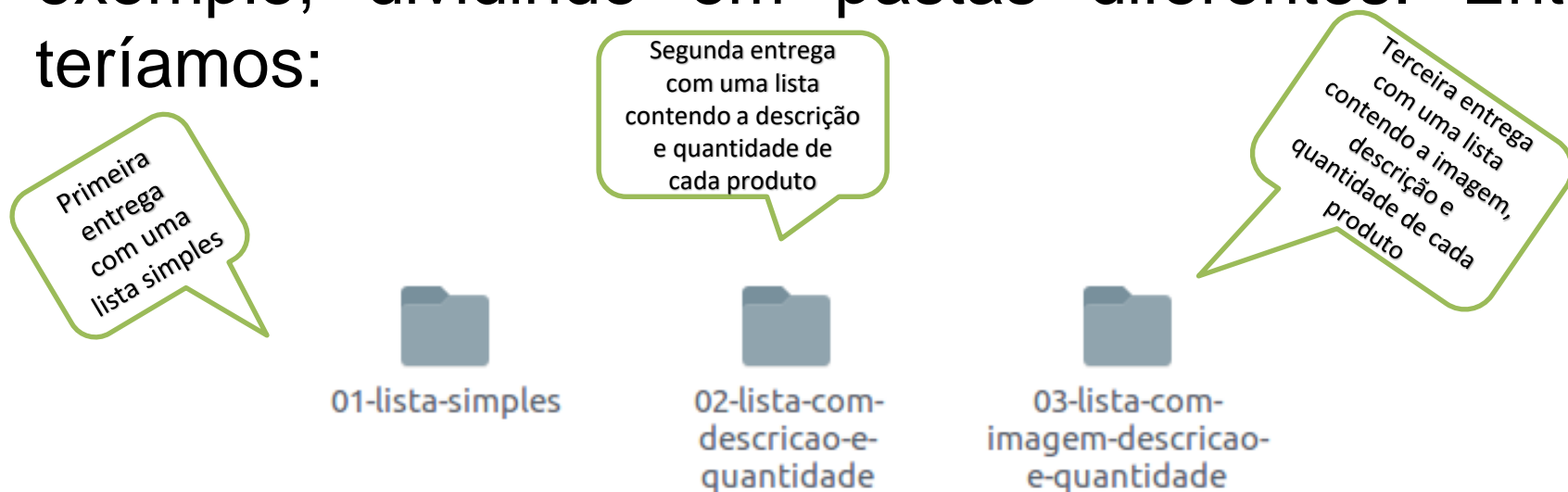
- Isso significa que você pode comparar o número da versão de qualquer software instalado no seu computador ou dispositivo móvel com o número da versão mais atualizada disponível atualmente e fazer um update para usar a versão mais nova.

Como funciona o Versionamento?

- Os números de versão geralmente são atribuídos em ordem crescente e correspondem a atualizações, inclusões de ferramentas e recursos ou exclusões para melhorar o seu funcionamento. Alguns softwares possuem números de versão internos (para maior controle dos desenvolvedores) que diferem dos números de versão do produto (para conhecimento do usuário).

Aplicando o Versionamento

- Utilizando o exemplo anterior, basicamente, precisamos, de alguma forma, salvar os estados do nosso projeto, ou seja, fazer com que cada alteração que entregarmos seja uma versão, por exemplo, dividindo em pastas diferentes. Então teríamos:



Aplicando o Versionamento

- De fato esse tipo de abordagem funciona, porém, note que para cada uma das versões que forem surgindo teremos que realizar esse processo manual diversas vezes. Se observarmos bem, essa abordagem tende a ser muito trabalhosa e cheia de riscos, tais, como:
 - **Controle:** para cada mera mudança no projeto teremos que lembrar de criar uma pasta do projeto atual.
 - **Histórico:** fica muito difícil de ter uma visão ampla e objetiva do que foi feito e quando foi feito, como também o que mudou de uma versão para outra.
 - **Reversão:** se precisarmos **voltar apenas uma parte** do que já foi feito há muito tempo, mantendo as outras coisas novas que foram implementadas, provavelmente teremos um trabalho tremendo.

Sistemas de controle de versão

- Para lidarmos com tais situações de uma maneira mais eficiente, podemos usar sistemas de controle da versão, também conhecidos como VCS do inglês Version Control System, ou na tradução, Sistema de Controle de Versão.
- Um VCS muito utilizado é o Git. Uma ferramenta muito comum na comunidade para versionar projetos que mantenham arquivos fontes como o nosso

VCS: O que é?

- Um software para gerenciar diferentes versões de um documento qualquer.
- O processo de guardar o histórico de alterações de diferentes arquivos.

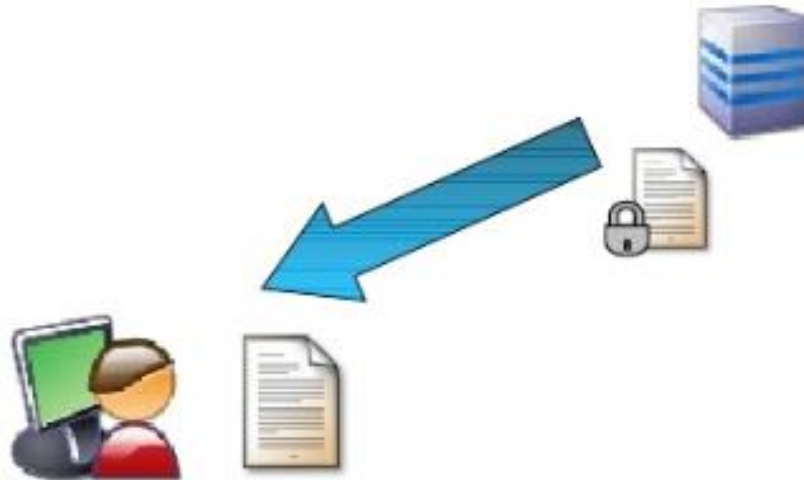
VCS: Vantagens

- Backup automático de todos arquivos
- Controle do histórico
- Trabalho em equipe
- Marcação e resgate de versões estáveis
- Ramificação do projeto

- 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.

- **Checkout**

- Obter um elemento do sistema de controle de versões.



- **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.

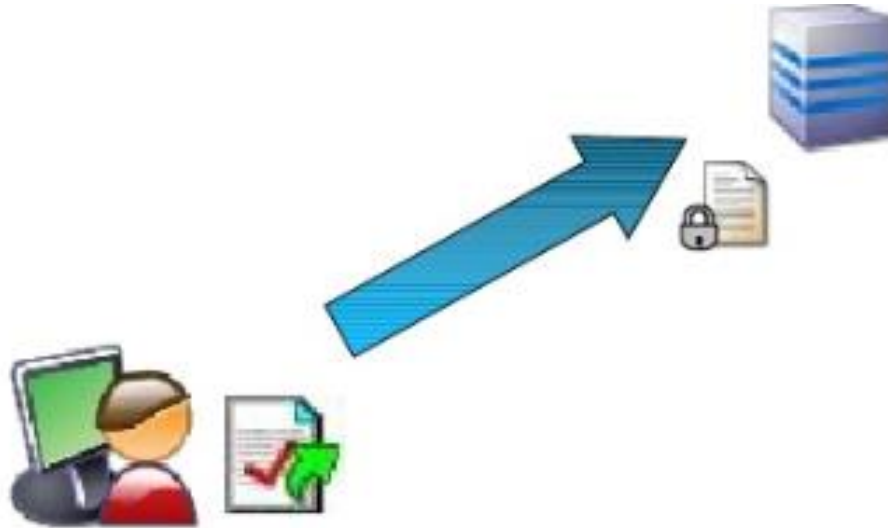
- **Não reservado (*copy-modify-merge*)**

- Vários usuários podem ter a posse do arquivo, permitindo o trabalho simultâneo.

- Não é garantido que a sua versão será consolidada

- **Checkin**

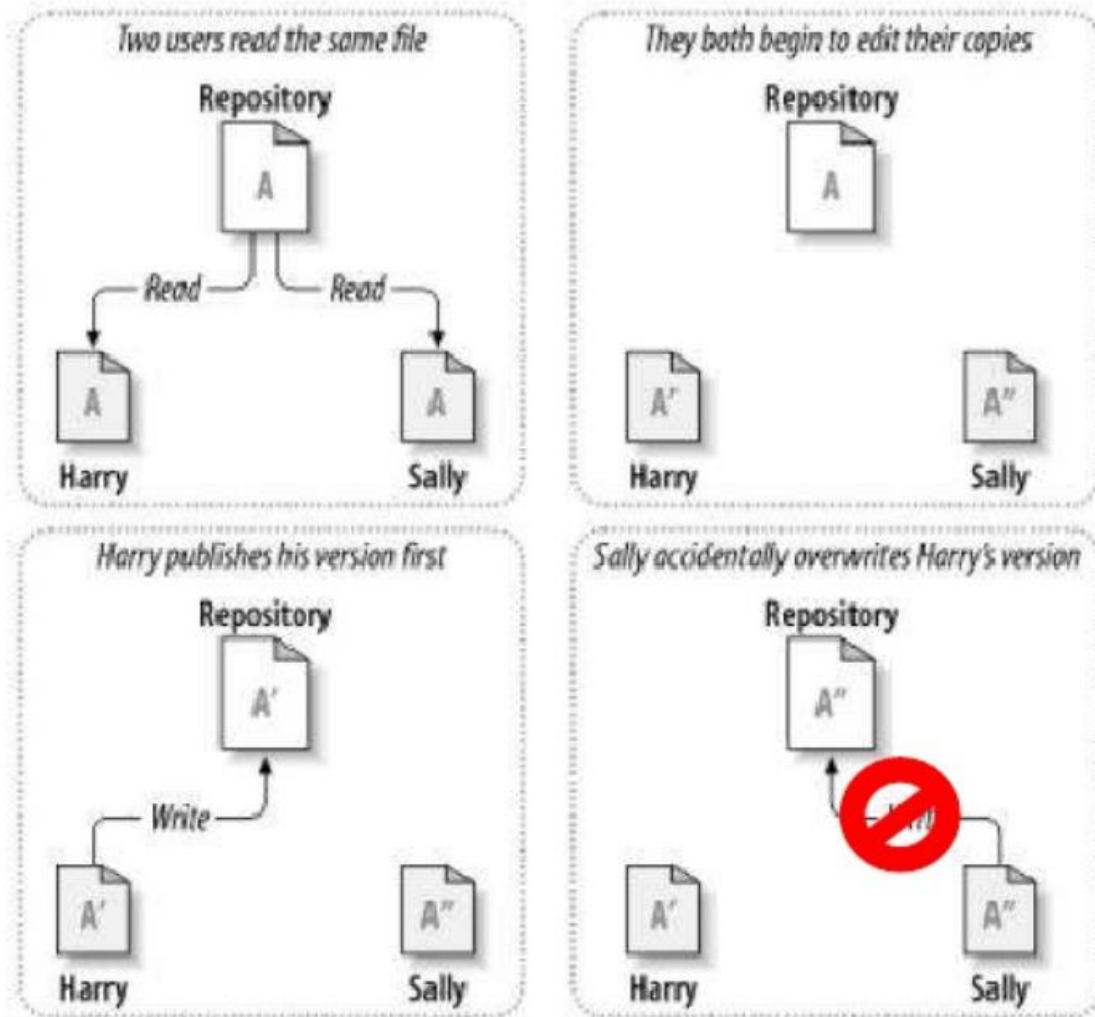
- Gravar as alterações no repositório gerando uma nova versão do elemento.



- **Checkin**

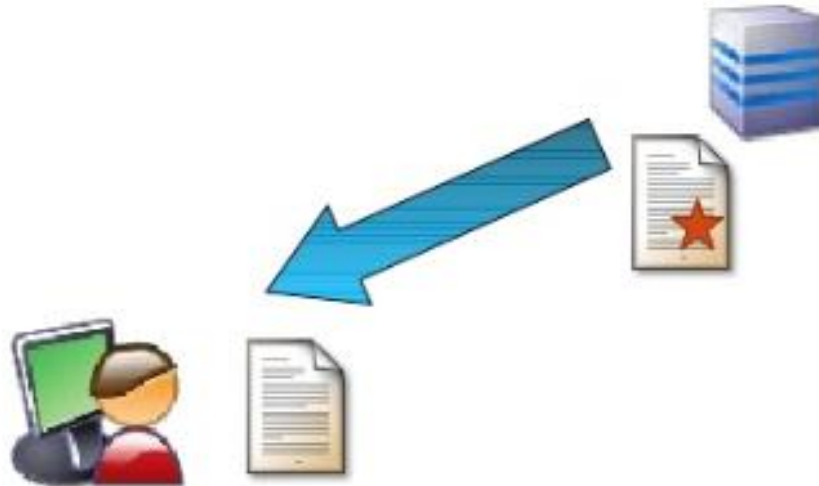
- Em algumas ferramentas chamado de Commit.
- Em casos de Checkout Não reservado pode gerar conflito de versões.

- **Checkin x Conflitos**



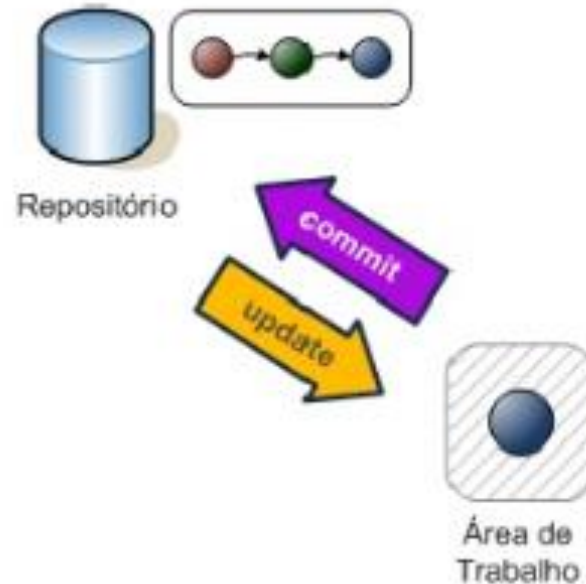
- **Update**

- Obter uma versão mais recente do elemento do sistema de controle de versões.



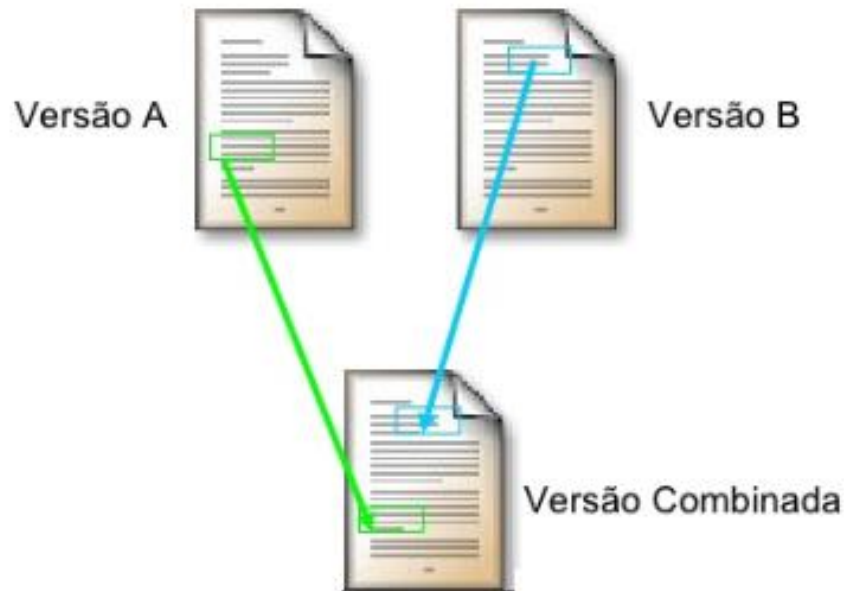
- **Update**

- Não confundir “atualizar” com “versionar”



- **Merge (Combinação)**

- Consiste em combinar alterações feitas por usuário distintos sobre o mesmo arquivo.

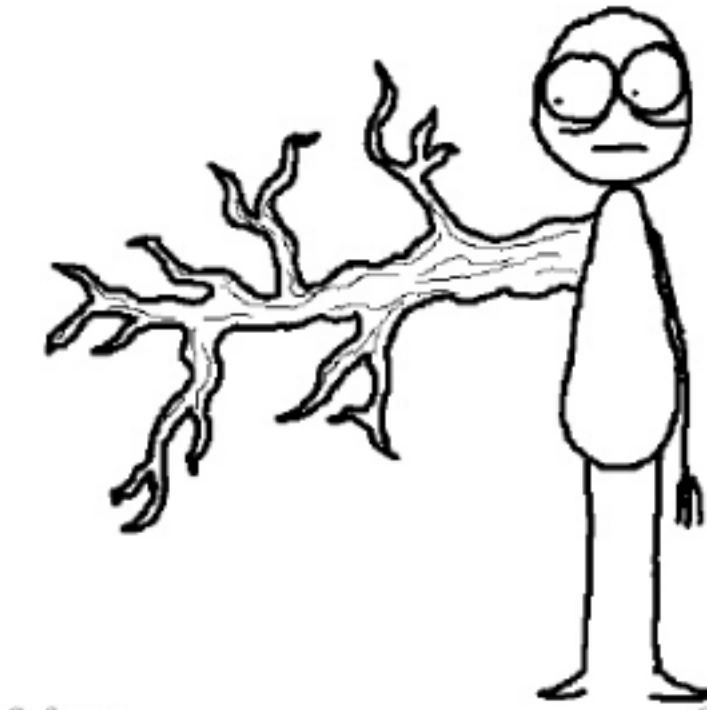


- **Merge (Combinação)**

- 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).

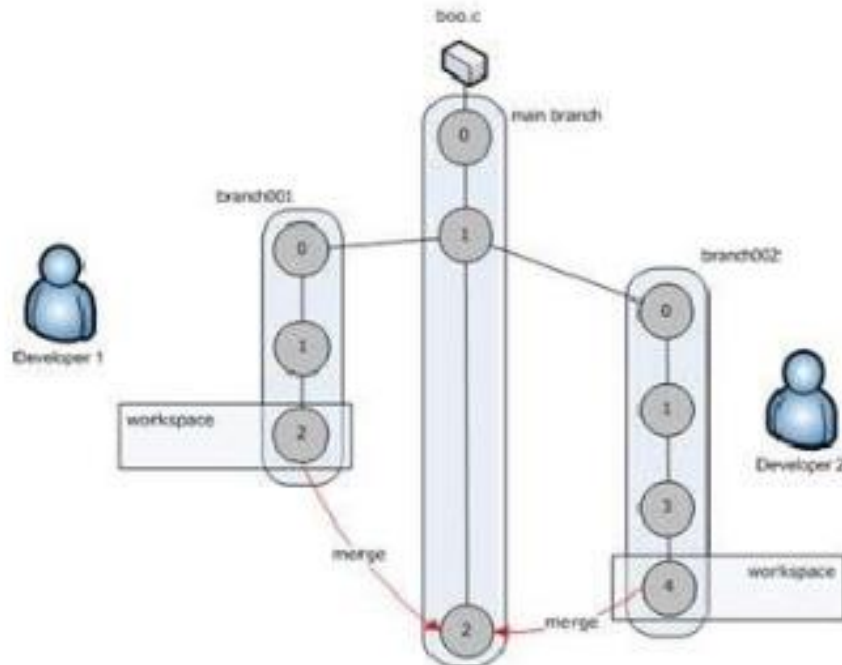
- **Branching (Ramificação)**

- Consiste em criar ramos paralelos de desenvolvimento (*branches*).



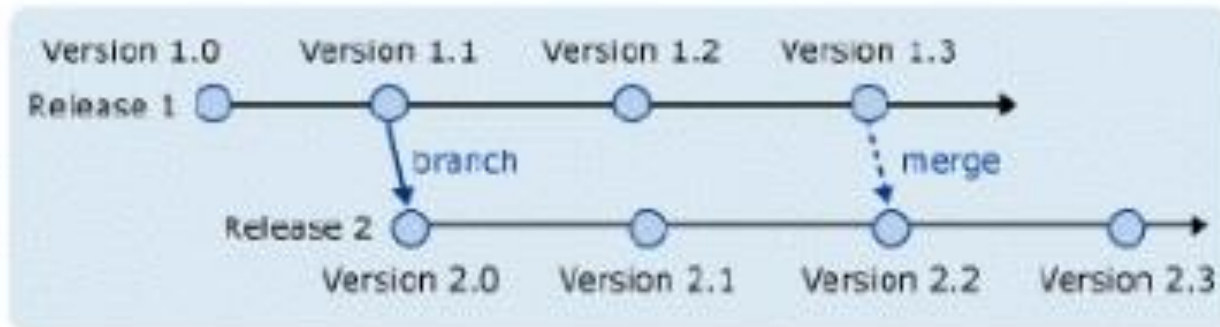
- **Branching (Ramificação)**

- Por default, existe pelo menos um *branch* em um projeto, o principal, também conhecido como “*mainline*” ou “*trunk*” ou “*Head*”



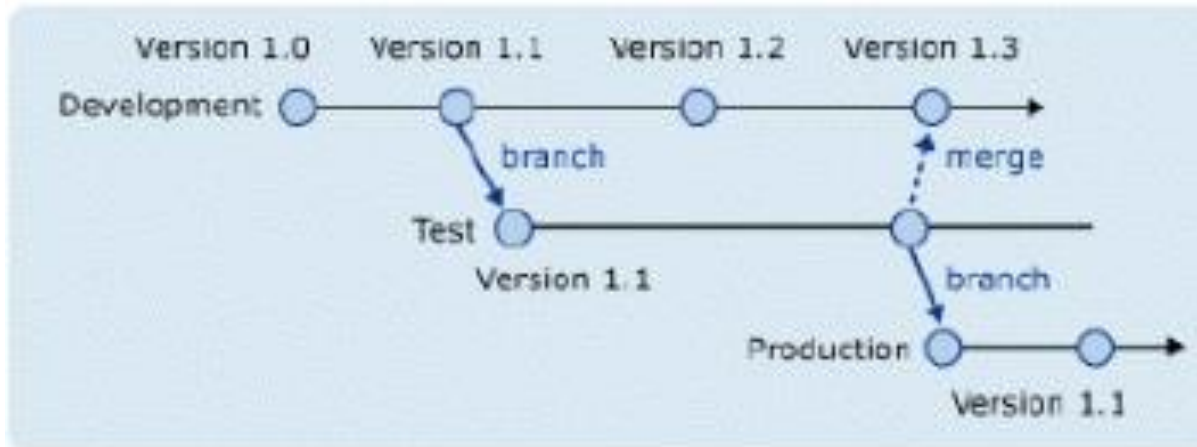
Branching

- Estratégias
 - *Branch Per Release*



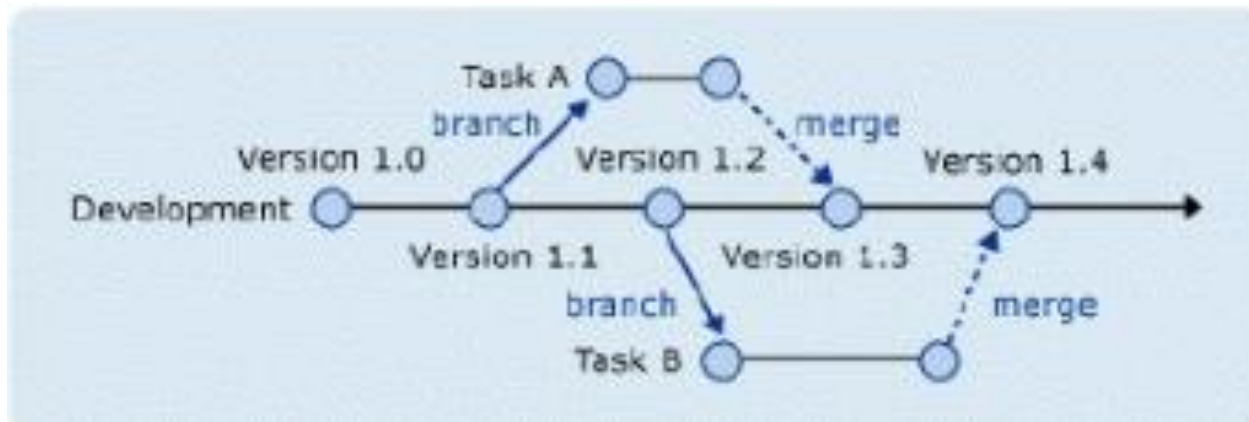
Branching

- Estratégias
 - *Code-Promotion Branches*



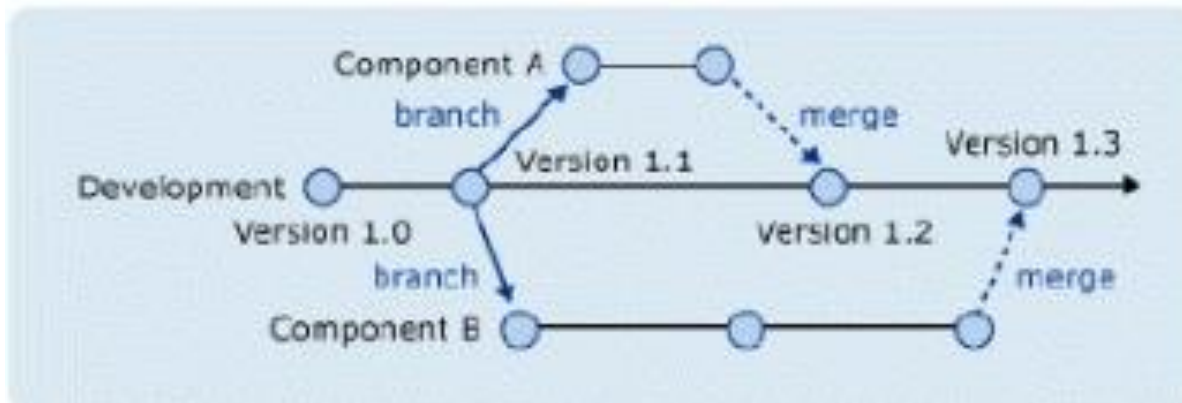
Branching

- Estratégias
 - *Branch Per Tasks*



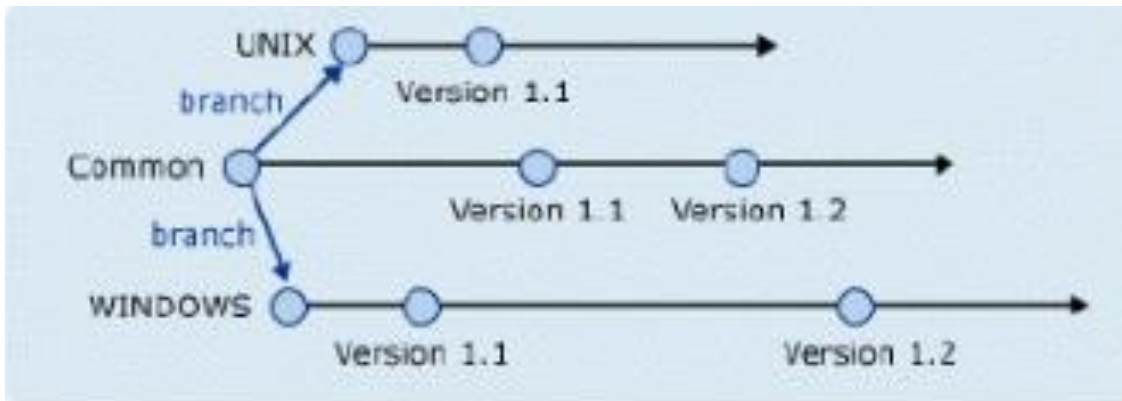
Branching

- Estratégias
 - *Branch Per Component*



Branching

- Estratégias
 - *Branch Per Technology*



- **Distributed Version Control Systems – DVCS**
 - **Equipe com centenas de desenvolvedores:** Mais processamento vai ser exigido do servidor central, piorando o tempo de resposta.
 - **Equipe espalhada em diferentes filiais da empresa:** Acesso remoto ao repositório com limitações de conexão e de permissão de escrita.

- **Vantagens**

- **Rapidez:** Ações executadas localmente.
- **Autonomia:** Possibilidade de trabalhar desconectado por um período.
- **Ramos privativos:** O desenvolvedor sempre trabalha em ramos privativos, decidindo o momento de combinar com os outros.
- **Facilidade de Merge:** Possuem mecanismos de rastreamento automático de merges.

- **Vantagens**

- **Confiabilidade:** No caso de uma pane os desenvolvedores conseguem trabalhar nos seus repositórios.
- **Redução de custos com servidor:** Sem a necessidade de possuir um servidor de grande poder de processamento.

- **Desvantagens**

- **Maior complexidade:** Exige maior conhecimento da ferramenta e do processo.
- **Tratamento de arquivos binários:** Controle de concorrência mais complicado.
- **As ferramentas de controle de mudanças ainda são centralizadas:** Ainda não existe integração entre ferramentas.

- **Ferramentas**

- Mercurial
- Git
- Bazaar

- **Projeto que usam o Git**
 - Android
 - Btrfs da Oracle
 - Gnome
 - Google
 - Kernel Linux
 - Perl
 - Samba...

Dúvidas??

