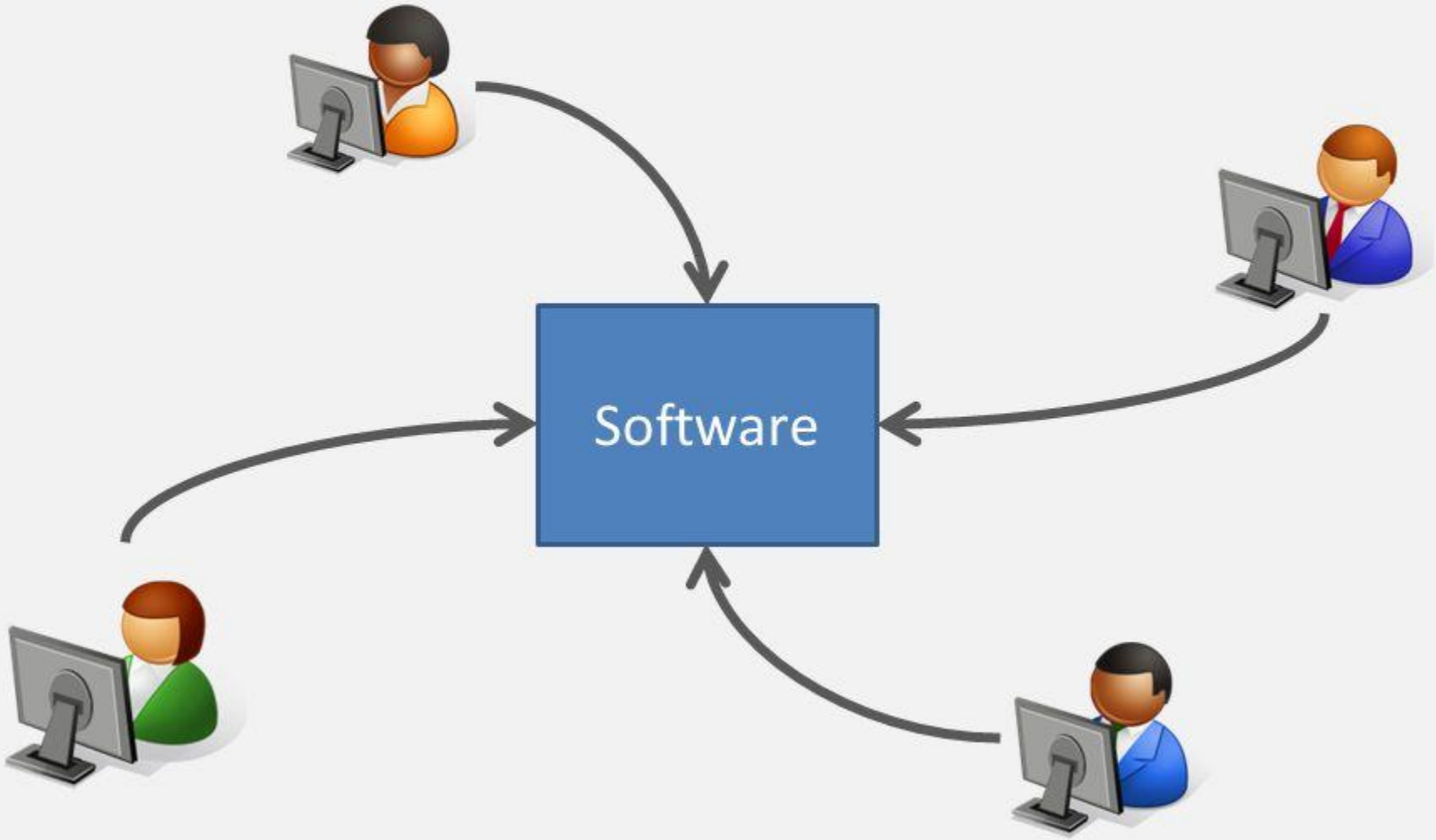
CONTROLE DE VERSÃO

Profª. Ana Flávia

Controle de versão: o que é?



*O **controle de versão** é um sistema que registra as mudanças feitas em um arquivo ou um conjunto de arquivos ao longo do tempo de forma que você possa recuperar versões específicas.*

*Um sistema de **controle de versões** (ou versionamento), VCS (version control system) ou SCM (source code management), é um software que tem a finalidade de gerenciar diferentes versões no desenvolvimento de um documento qualquer.*

Esses sistemas são comumente utilizados no desenvolvimento de software para controlar as diferentes versões — histórico e desenvolvimento — dos códigos-fontes e também da documentação.

VANTAGENS

As principais vantagens de se utilizar um *sistema de controle de versão* são:

- ✓ **Controle do histórico**: facilidade em *desfazer* e possibilidade de analisar o histórico do desenvolvimento, como também facilidade no resgate de versões mais antigas e estáveis. A maioria das implementações permitem analisar as alterações com detalhes, desde a primeira versão até a última.

- ✓ **Trabalho em equipe**: um *sistema de controle de versão* permite que diversas pessoas trabalhem sobre o mesmo conjunto de documentos ao mesmo tempo e minimiza o desgaste provocado por problemas com conflitos de edições. É possível que a implementação também tenha um controle sofisticado de acesso para cada usuário ou grupo de usuários.

VANTAGENS

As principais vantagens de se utilizar um *sistema de controle de versão* são:

- ✓ **Marcação e resgate de versões estáveis**: a maioria dos sistemas permite marcar onde é que o documento estava com uma versão estável, podendo ser facilmente resgatado no futuro.

- ✓ **Ramificação de projeto**: a maioria das implementações possibilita a divisão do projeto em várias linhas de desenvolvimento, que podem ser trabalhadas paralelamente, sem que uma interfira na outra.

- ✓ **Segurança**: Cada software de controle de versão usa mecanismo para evitar qualquer tipo de invasão de agentes infecciosos nos arquivos. Somente usuários com permissão poderão mexer no código.

VANTAGENS

As principais vantagens de se utilizar um *sistema de controle de versão* são:

- ✓ **Rastreabilidade**: com a necessidade de sabermos o local, o estado e a qualidade de um arquivo; o controle de versão trás todos esses requisitos de forma que o usuário possa se embasar do arquivo que deseja utilizar.

- ✓ **Organização**: Com o software é disponibilizado interface visual que pode ser visto todo arquivos controlados, desde a origem até o projeto por completo.

- ✓ **Confiança**: O uso de repositórios remotos ajuda a não perder arquivos por eventos imponderáveis. Além disso e disponível fazer novos projetos sem danificar o desenvolvimento.

FUNCIONAMENTO

- A maior parte das informações - com todo o histórico - ficam guardadas num repositório, num servidor qualquer.

- Geralmente um cliente pode aceder ao repositório pela rede (via *socket*) ou localmente quando o cliente está na mesma máquina do servidor.

FUNCIONAMENTO

- O repositório armazena a informação - um conjunto de documentos - de modo persistente num sistema de arquivos ou num banco de dados qualquer - onde ocasiona um tipo de hierarquia entre arquivos e diretórios.

- Inúmeros clientes podem se conectar em um repositório, e assim leem e escrevem nesses arquivos. Incrementando dados, um usuário disponibiliza a informação para outros; fazendo a leitura dos elementos, um usuário recebe a informação de outros.

FUNCIONAMENTO

- É viável armazenar o conteúdo em outros dispositivos capazes de "eternizar" e resgatar facilmente a informação.

- Cada servidor pode ter vários sistemas de controle de versão e cada sistema pode ter diversos repositórios, limitando-se na capacidade de gerenciamento do software e também no limite físico do hardware. Geralmente um repositório possui um endereço lógico que permite a conexão do cliente. Esse endereço pode ser um conjunto IP/porta, uma URL, um caminho do sistema de arquivos, etc.

FUNCIONAMENTO

- Cada desenvolvedor possui em sua máquina uma cópia local (*working copy*) somente da última versão de cada documento.
- A cópia local geralmente é feita num sistema de arquivos simples.
- A cada alteração relevante do desenvolvedor é necessário "atualizar" as informações do servidor submetendo (*commit*) as alterações.

FUNCIONAMENTO

- O servidor então guarda a nova alteração junto de todo o histórico mais antigo.

- Se o desenvolvedor quer atualizar sua cópia local é necessário atualizar as informações locais, e para isso é necessário baixar novidades do servidor (*update*).

ENVIO E RESGATE DE VERSÕES

- **A principal função do sistema de controle de versão é:**

Armazenar todo o histórico de desenvolvimento do documento, desde o primeiro envio até sua última versão. Isso permite que seja possível resgatar uma determinada versão de qualquer data mais antiga, evitando desperdício de tempo no desenvolvimento para desfazer alterações quando se toma algum rumo equivocado.

ENVIO E RESGATE DE VERSÕES

- O envio das alterações é feito pelo desenvolvedor (do lado do cliente).
- Recomenda-se que uma alteração seja enviada a cada vez que o software estiver minimamente estável, a cada nova parte (uma função) ou a cada alteração relevante que esteja funcionando corretamente.

ENVIO E RESGATE DE VERSÕES

- Não é recomendável o envio quando o documento como um todo possa causar alguma dificuldade no desenvolvimento de outro colaborador, como por exemplo um código não compilado ou com algum defeito que comprometa a execução geral.

ENVIO E RESGATE DE VERSÕES

- Cada "envio" é na maioria dos sistemas chamado de "*commit*" (às vezes "submit"), ou seja, efetivar as alterações no (ou "submeter" ao) repositório.
- Cada envio produz uma nova versão no repositório e é armazenado como "uma fotografia" do momento.

ENVIO E RESGATE DE VERSÕES

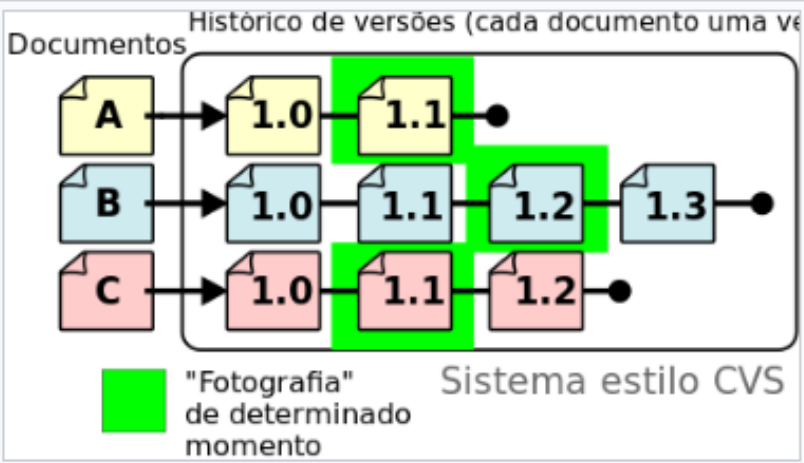


fig. c *- Em sistemas no estilo do CVS, cada documento tem sua versão controlada individualmente. Assim, é necessário adicionar marcas ('tags' - em inglês) para que se tenha uma "fotografia" de determinado momento. Ou seja, o versionamento do conjunto fica a cargo do usuário pois a numeração gerada pelo sistema de controle de versão não fornece "alinhamento" aos documentos.*

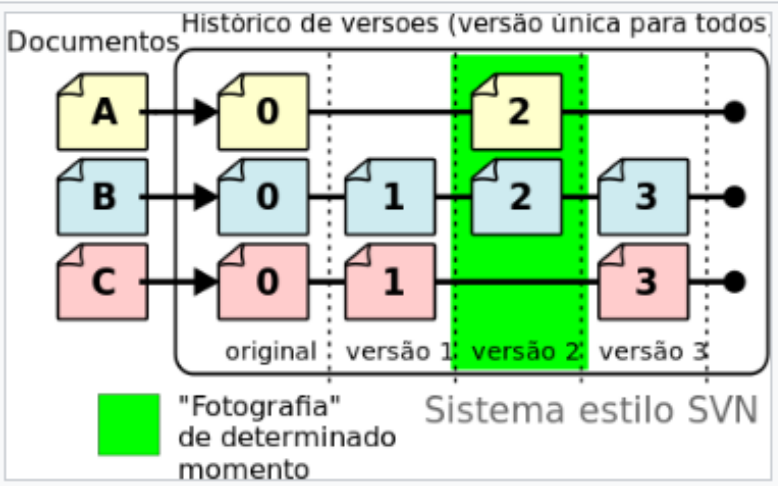


fig. d *- Já em sistemas no estilo do SVN, o controle de versão é feito por cada envio ao servidor. Ou seja, há uma versão global para todos os documentos. Se você enviar 4 versões do arquivo B enquanto você enviou apenas 2 versões do A, ambos estão na versão final que é a 4. Perceba que toda "fotografia" de qualquer momento será sempre uma coluna alinhada como mostra a figura.*

ENVIO E RESGATE DE VERSÕES

- Se o arquivo fosse armazenado diretamente num **sistema de arquivos simples**, não haveria histórico, a menos que isso fosse feito manualmente através de cópias completas dos documentos numa organização (que pode ou não envolver pastas) que dependerá do desenvolvedor.

ENVIO E RESGATE DE VERSÕES

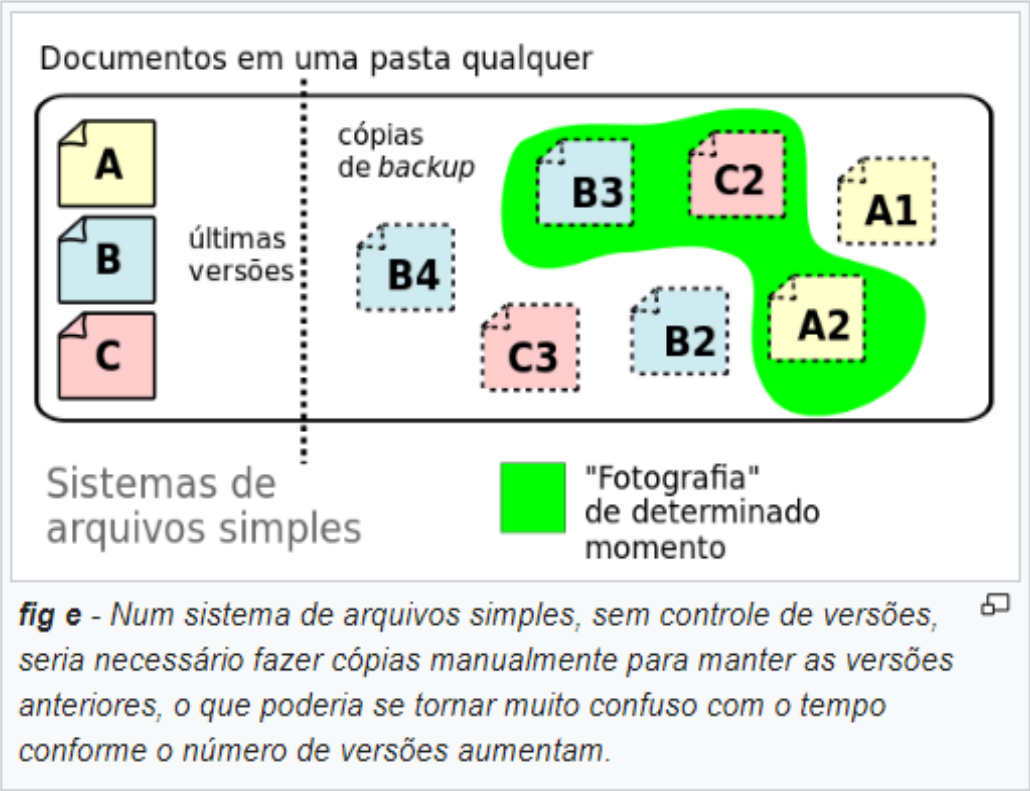


fig e *- Num sistema de arquivos simples, sem controle de versões, seria necessário fazer cópias manualmente para manter as versões anteriores, o que poderia se tornar muito confuso com o tempo conforme o número de versões aumentam.*

HISTÓRICO DE ENVIO

- Muitas vezes, é possível acrescentar comentários no envio das alterações, o que facilita também uma possível análise do histórico. Geralmente o relatório com as versões e os comentários de cada envio são chamados de "histórico" ou "log", e uma análise deste relatório pode facilitar em muitos aspectos no desenvolvimento do produto.

HISTÓRICO DE ENVIO

- A documentação do CVS, SVN e outros, recomendam que o comentário do histórico seja amplo, geral e abstrato, ou seja, que não se limite à explicação da mudança do código em si, mas sim sobre o que foi mudado ou acrescentado no conceito ou no funcionamento como um todo. A mudança do código pode ser analisada através de uma diferença (ou diff) entre duas versões, portanto o comentário seria útil apenas para explicar a mudança de forma lógica.

HISTÓRICO DE ENVIO

- Exemplo prático dessa recomendação: utilize "acréscimo de uma condição que verifica se existe saldo na conta para não permitir que seja sacado sem saldo" ao invés de um comentário contendo o próprio código modificado como:

- "+ if (!this.haSaldoNaConta()) { this.proibeSaque() }"

TRABALHO EM EQUIPE

- Sistemas de controle de versão também são convenientes quando diversos desenvolvedores trabalham sobre o mesmo projeto simultaneamente, resolvendo eventuais conflitos entre as alterações. A maioria dos sistemas possui diversos recursos como ramificação e mesclagem de histórico para auxiliar nessas tarefas.

TRABALHO EM EQUIPE

- Para que seja possível o trabalho em equipe, o sistema de controle de versão pode possuir um mini sistema de controle de usuários embutido ou pode utilizar algum outro sistema de autenticação separado. Assim, é possível identificar cada usuário, que geralmente fica protegido por uma senha pessoal, ou alguma senha criada pelo administrador de sistemas.

TRABALHO EM EQUIPE

- No CVS, por exemplo, é possível escolher o método de autenticação a ser usado, dentre várias opções. No caso do SVN, por exemplo, se ele estiver rodando via Apache, o controle de usuários poderá ser feito pela autenticação padrão do Apache. Embora menos comum, é possível também configurar o SVN para utilizar o usuário do sistema, como os usuários do Linux ou do Windows.