Versionamento de Código

Núcleo de Desenvolvimento de Software

Por quê?

Facilidades de utilizar um sistema de versionamento de código.

Várias versões

- Quando se salva uma nova versão de um arquivo, a versão anterior é sobrescrita.
  - É possível criar um novo arquivo para cada versão mas isso ocupa muito espaço.

- Um sistema de versionamento possibilita gravar todas as alterações feitas a um arquivo.
  - Não ocupa o mesmo espaço que criar uma nova cópia do arquivo.
  - Quanto maior o histórico, maior o tamanho do repositório.
  - O tamanho da cópia local não muda com o histórico.

Várias versões

- A possibilidade de se acessar diversas versões permite:
  - Voltar a uma versão anterior caso uma mudança se mostre errada.
  - Descartar mudanças.
  - Verificar mudanças feitas a um arquivo.
  - Descobrir quem escreveu o código de cada linha de um arquivo.

Junção de versões

- Quando se realiza o desenvolvimento em grupo, é comum que mais de um desenvolvedor faça modificações a um mesmo arquivo.
- É preciso juntar as modificações de ambos em uma versão única.
- Os sistemas de versionamento fazem a junção de diferentes versões automaticamente.

Acesso

- É possível baixar uma cópia do código de qualquer local com acesso ao repositório.
- É possível controlar o nível de acesso às pastas.
- Cada usuário pode usar uma senha e nome de usuário para registrar suas mudanças.

Subversion

Sobre o Subversion

Breve histórico

- O Subversion (também conhecido como SVN) foi desenvolvido como uma alternativa ao CVS, buscando melhorar alguns aspectos do mesmo.
- O projeto é open source, foi iniciado em 2000 e está em desenvolvimento.
- Roda em diversos sistemas operacionais (Windows, Linux, OS X).
- Possui diversos clientes com interface gráfica, entre eles o TortoiseSVN.
- Atualmente a versão estável é a 1.6.1 de 10/04/2009.

Funcionalidades

- Commits são atômicos
- Rastreamento de junções
- Consegue lidar com arquivos binários
- Servidor *standalone* próprio (svnserve)
- Design pensado para uso humano ou automatizado
  - Diversos plug-ins para várias linguagens
  - Integração com Apache
  - Diversos clientes
- Faz quase tudo que o CVS fazia.

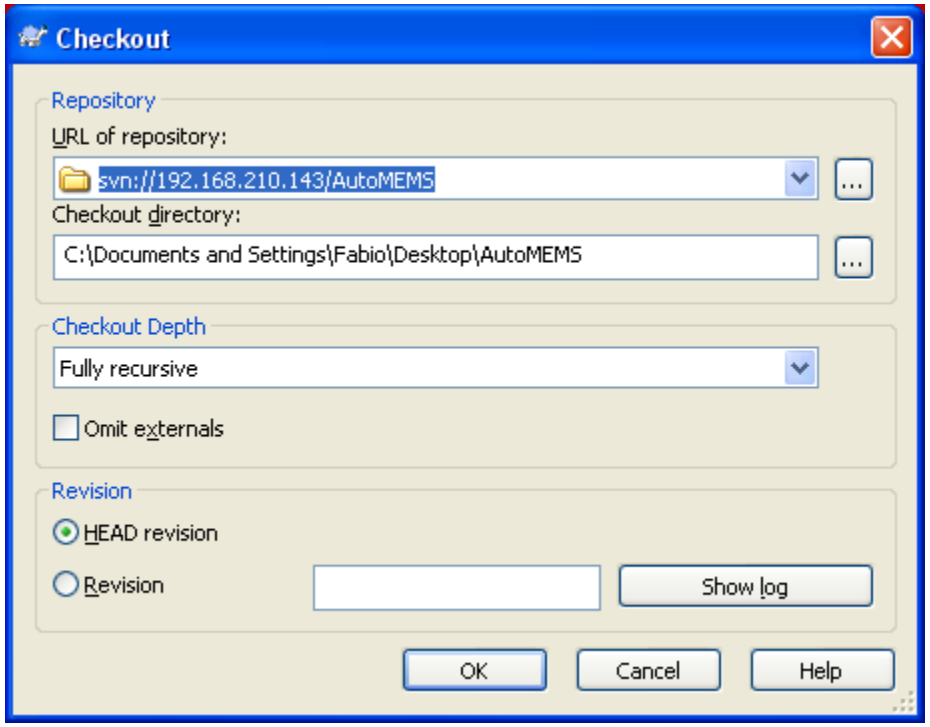
Operações Básicas

Funcionalidades básicas necessárias para se utilizar o SVN.

Checkout

- Os arquivos são salvos em um servidor, no repositório.
- Para que se possa trabalhar nos arquivos é preciso criar uma cópia dos arquivos na máquina.
- Essa operação é chamada de checkout.

Checkout


Checkout
Repository
URL of repository:
svn://192.168.210.143/AutoMEMS
Checkout directory:
C:\Documents and Settings\Fabio\Desktop\AutoMEMS
Checkout Depth
Fully recursive
Omit externals
Revision
HEAD revision
Revision
Show log
OK
Cancel
Help

Checkout

Schema	Access Method
file:///	direct repository access (on local disk)
http://	access via WebDAV protocol to Subversion-aware Apache server
https://	same as http://, but with SSL encryption.
svn://	access via custom protocol to an svnserve server
svn+ssh://	same as svn://, but through an SSH tunnel.

Update

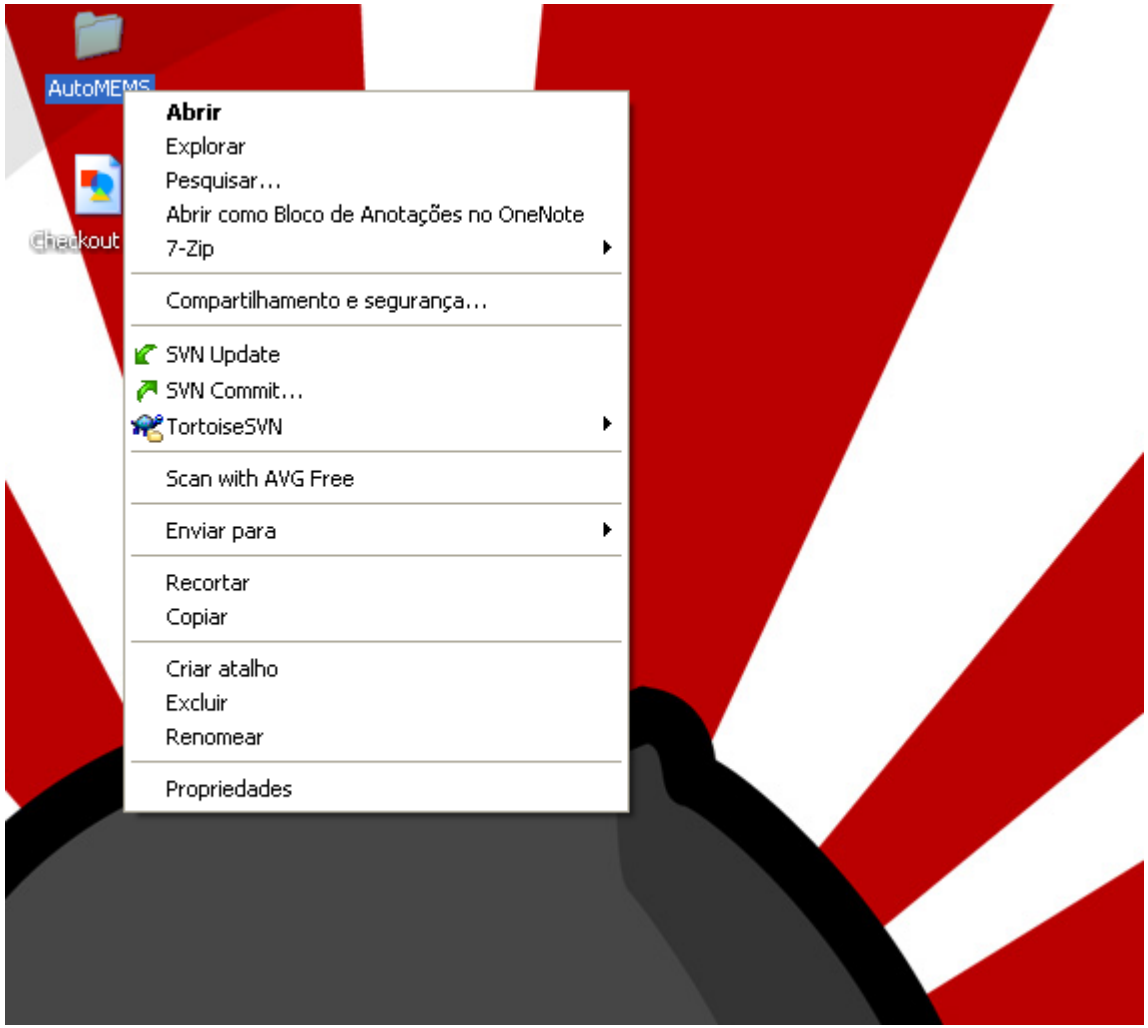
- Para trazer do repositório as mudanças realizadas é preciso fazer uma atualização ou *update.*
- Ao realizar um update, o SVN fará uma junção dos arquivos da cópia local com os arquivos no repositório.
- Conflicts:
  - Se houver uma modificação feita na cópia local em uma linha que seria modificada ocorre um conflito ou *conflict*.
  - Um conflito deve ser resolvido manualmente. O TortoiseSVN possui uma ferramenta gráfica para realizar esta tarefa.
  - Não é possível dar commit com arquivos em conflito.

Update

- É possível dar um update até uma determinada versão.

- O update mantém a profundidade do checkout.

Update


AutoMEMS
Checkout
Abrir
Explorar
Pesquisar...
Abrir como Bloco de Anotações no OneNote
7-Zip
Compartilhamento e segurança...
SVN Update
SVN Commit...
TortoiseSVN
Scan with AVG Free
Enviar para
Recortar
Copiar
Criar atalho
Excluir
Renomear
Propriedades

Commit

- Após fazer modificações a um arquivo, é preciso enviar estas modificações para o repositório.
- A operação que realiza esta tarefa é o *commit*.
- É recomendável que o *commit* seja feito na pasta principal, e não arquivo a arquivo.
  - Evita esquecer de dar *commit*.
  - Evita esquecer de adicionar arquivos criados recentemente.
- O *commit* falha se houver uma versão mais nova do arquivo no repositório. Nesse caso deve-se realizar um *update* antes do *commit*.

Commit

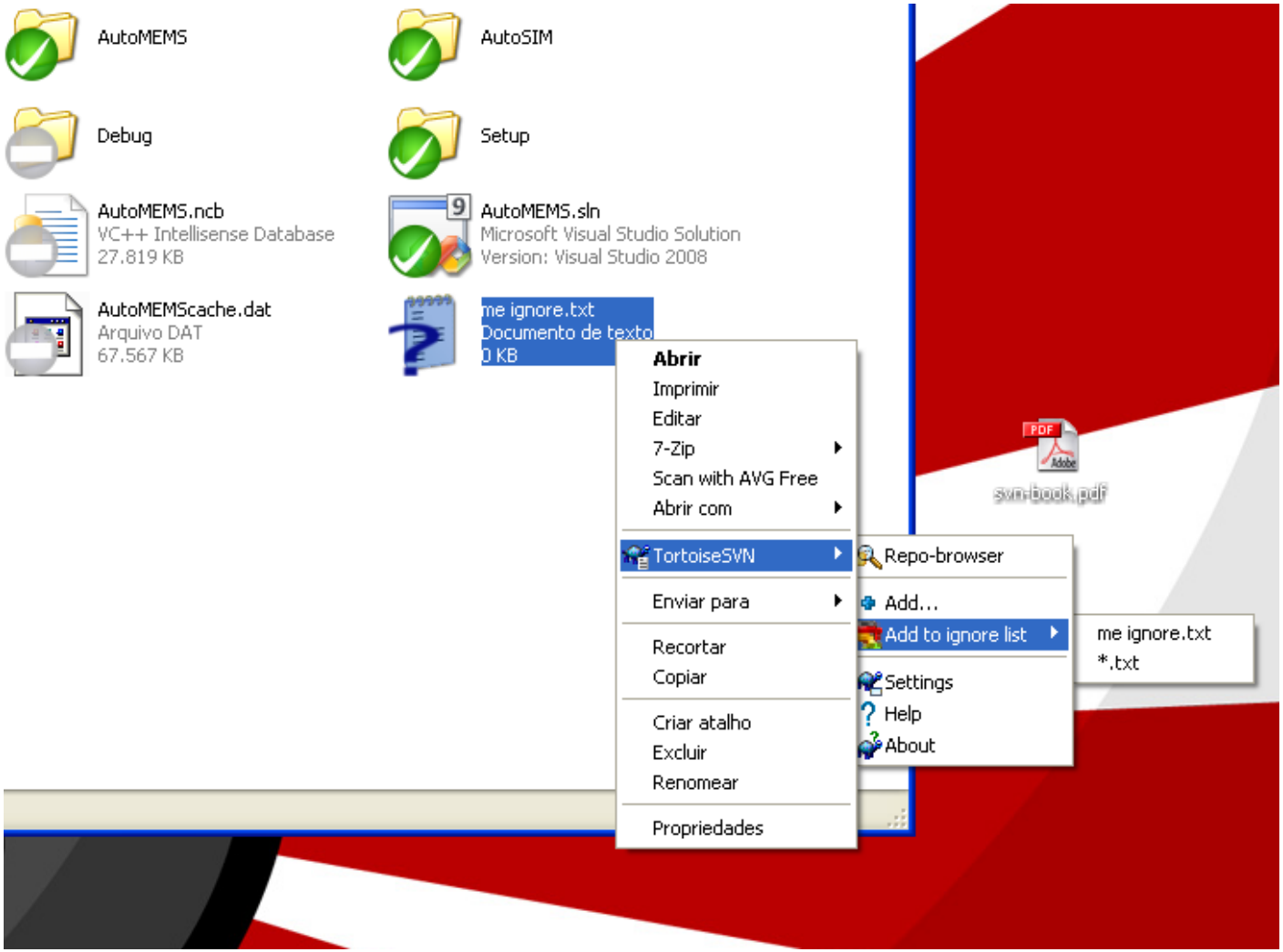
- Deve-se colocar um comentário quando se dá o *commit* explicando o que foi feito.
- Os comentários devem ser suficientemente claros e descritivos para que se possa voltar a uma versão anterior se necessário.
- O tamanho do repositório não muda com o número de *commits* mas sim com o número de mudanças, então dêem bastante *commits*.
- O repositório evita perda de dados. O que está na cópia local não pode ser recuperado se houver um problema com o computador.

Ignorando arquivos

- É possível ignorar arquivos para que os mesmos não sejam comitados nem apareçam na lista de arquivos passíveis de commit.

- A exclusão pode ser feita por pasta, por arquivo ou por tipo de arquivo (extensão).

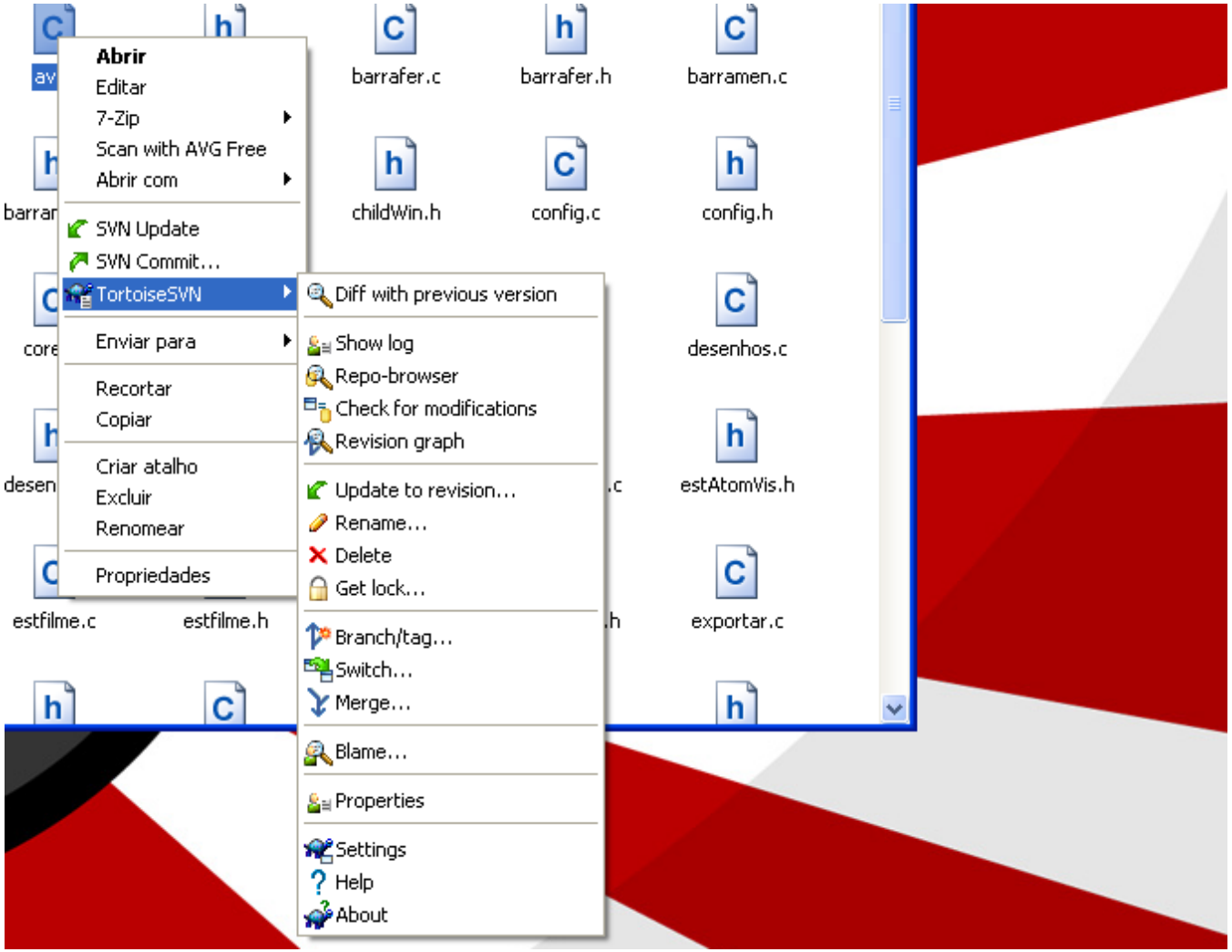
Ignorando arquivos



Revert, Diff, Delete e Lock

- *Revert* permite descartar mudanças realizadas na cópia local, recuperando a mesma versão do repositório.
- *Diff* permite comparar o arquivo da cópia local com qualquer outra versão.
- *Delete* apaga um arquivo do repositório. Quando os demais fizerem um *update* o arquivo será removido. Simplesmente apagar o arquivo não faz isso.
- *Lock* permite travar o arquivo para que nenhum outro usuário modifique o arquivo (útil quando o arquivo não é passível de junção).

Menu Extendido



Branches

O que são e como usar

Alice e Bob

- Alice e Bob trabalham em um mesmo projeto e seu código é versionado no SVN.
- Bob precisa fazer uma grande modificação no programa, que vai alterar todos os arquivos. Enquanto ele faz essa modificação, Alice precisa continuar trabalhando em outras partes do projeto.

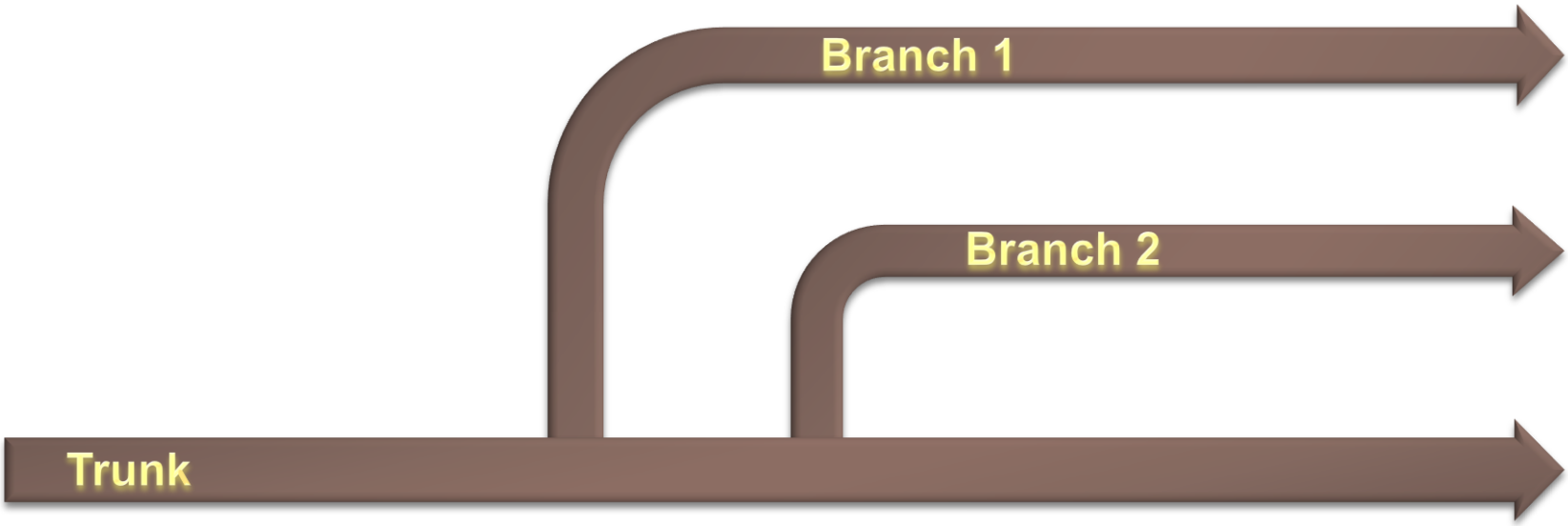
Opção 1

- Bob pode, em seu computador, desenvolver a mudança. Quando terminar a mudança, ele faz um grande *commit*.
  - Problemas:
    - O código do Bob não será versionado.
    - Quanto mais tempo bob fica sem dar *commit*, mais difícil ficará juntar o código depois.
    - Bob não poderá aproveitar as melhorias feitas por Alice.

Opção 2

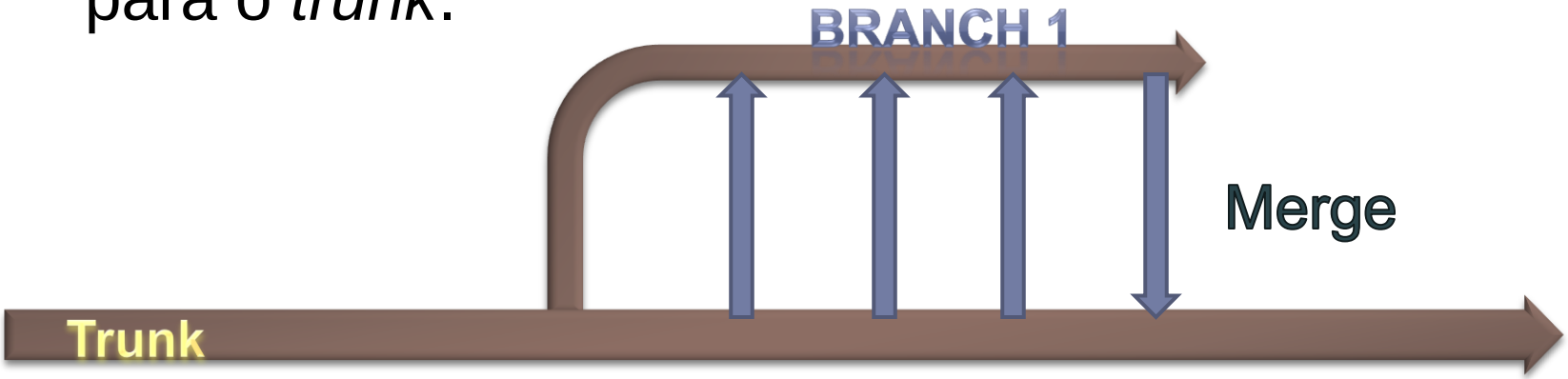
- Faça como uma árvore
- Crie um *branch*!

O que é um branch?



Desenvolvimento com branches

- Ao criar um *branch*, Bob pode desenvolver as novas funcionalidades.
- Enquanto Bob desenvolve, ele pode periodicamente fazer um merge do *trunk* para o seu *branch* e pegar as melhorias feitas por Alice.
- Quando Bob termina, ele faz um merge de seu *branch* com o *trunk*, passando as suas melhorias para o *trunk*.



Switch

- O comando *Switch* permite mudar a sua cópia local para que aponte para outro *branch*, *tag* ou o *trunk*.

- É idêntico a um *checkout* mas mais rápido.

Tags

O que são e para que servem

Tags

- Tags são basicamente o mesmo que branches. A diferença e puramente conceitual. Um branch é criado para ser modificado para adicionar ou corrigir algo no código.
- Um tag é simplesmente uma representação mais simples de se lembrar, para humanos, de uma determinada versão do código.
- Por exemplo, se a revisão 709 do projeto SimMEMS é a versão 2.5.3 do programa que foi disponibilizada, pode-se criar um tag com o nome SimMEMS 2_5_3 a partir da revisão 709 para tornar mais fácil recuperar essa versão.

Tags

- Do ponto de vista do SVN, tags e branches são iguais.
- É possível modificar o código de uma tag e dar commit.
- Isso não deve ser feito, e em geral cabe aos programadores não o fazer.

Layout Padrão

Como branches, tags e o trunk são armazenadas

Padrão recomendado

- Para se armazenar branches tags e o trunk usa-se três pastas. É recomendável fazer o checkout do trunk ou do branch que se vai utilizar e não do projeto inteiro.

/trunk

/branches

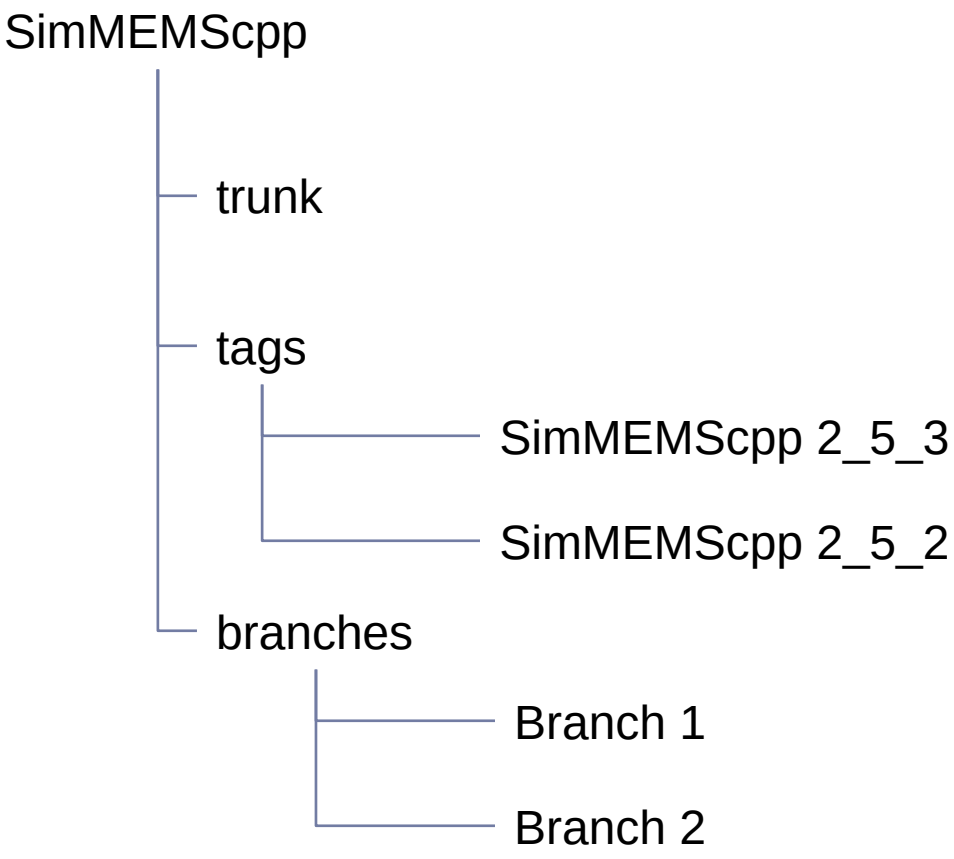
/tags

Padrão recomendado

- Se existe mais de um projeto, a estrutura é repetida para cada projeto.

/Projeto 1/trunk
/Projeto 1/tags
/Projeto 1/branches
/Projeto 2/trunk
/Projeto 2/tags
/Projeto 2/branches

Exemplo


SimMEMScpp
trunk
tags
SimMEMScpp 2_5_3
SimMEMScpp 2_5_2
branches
Branch 1
Branch 2

Referências

Sites úteis

Referências

- Usando o SVN
  - http://svnbook.red-bean.com/
- Código fonte e binários do SVN
  - http://subversion.tigris.
- Código fonte e binários do TortoiseSVN
  - http://tortoisesvn.tigris.org/
- Software para conversão de repositórios CVS2SVN
  - http://cvs2svn.tigris.org/