Instituto de computação

Gerência de Configuração: Subversion

Leonardo Gresta Paulino Murta

leomurta@ic.uff.br

Agenda

- Introdução
- Controle de concorrência
- Repositório
- Espaço de trabalho
- Junção
- Propriedades

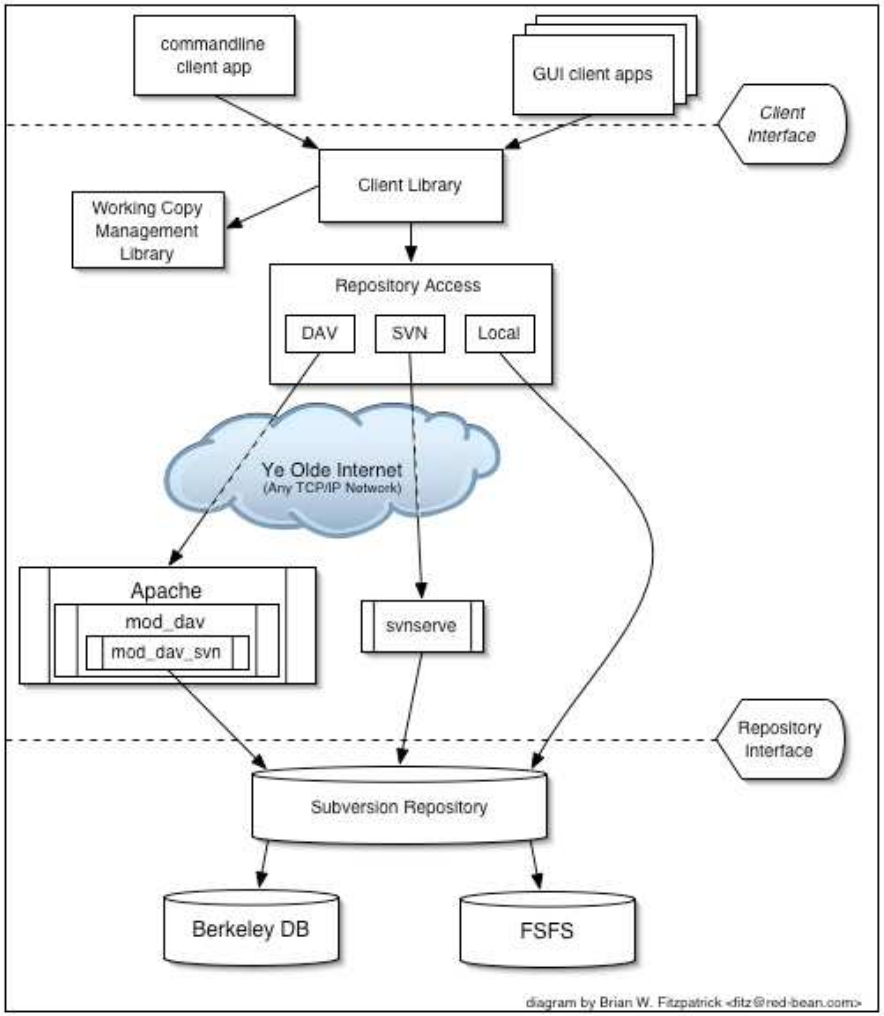
Histórico

- Projeto iniciado em 2000
  - Iniciativa da CollabNet
  - Open-Source
- Intuito de substituir o CVS
  - Similar no uso
  - Melhor na implementação
- Auto controlado desde agosto de 2001

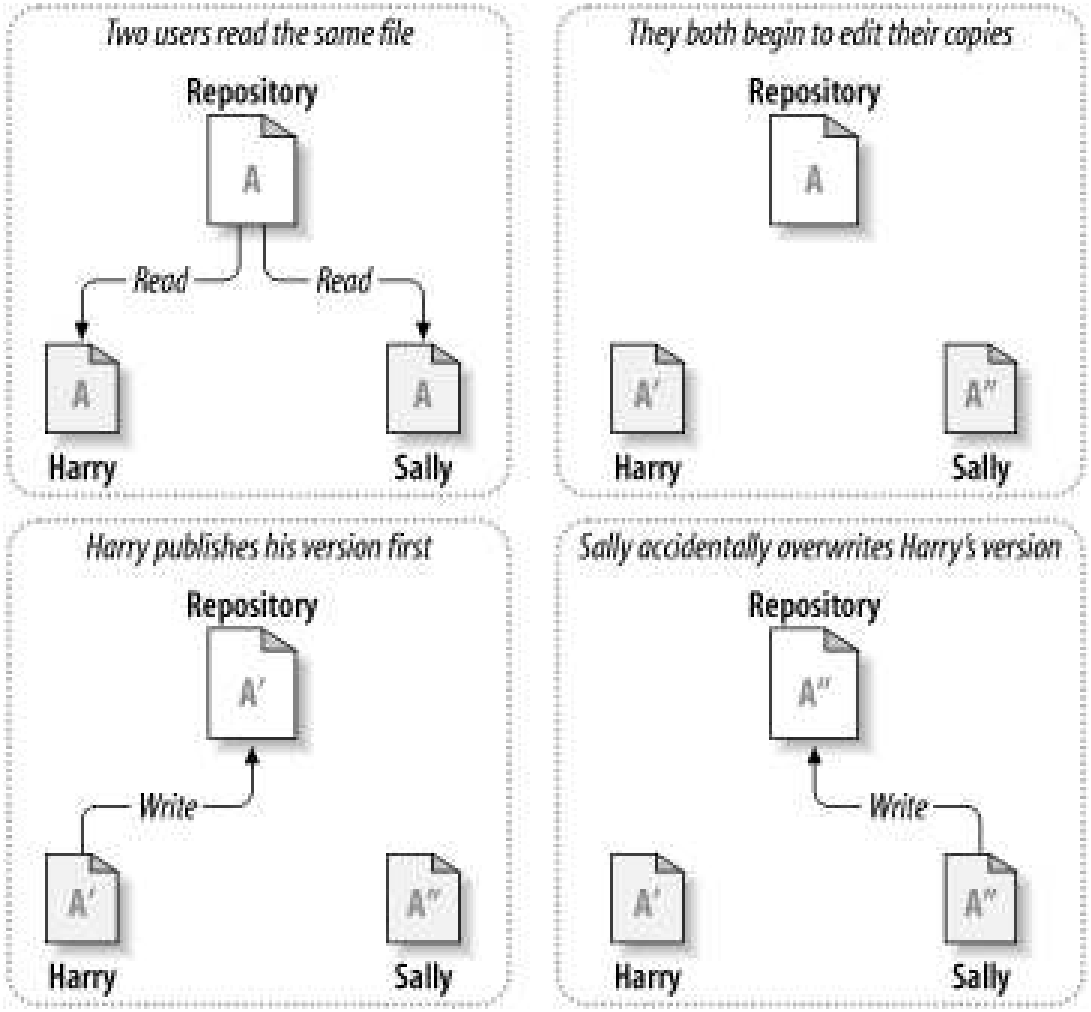
Características

- Versionamento de diretórios
- Copia, renomeação e movimentação com histórico
- Check-ins atômicos
- Versionamento de meta-dados
- Acesso via http/https
- Uso extensivo de deltas
  - Delta de binários
  - Delta bidirecional na comunicação cliente/servidor

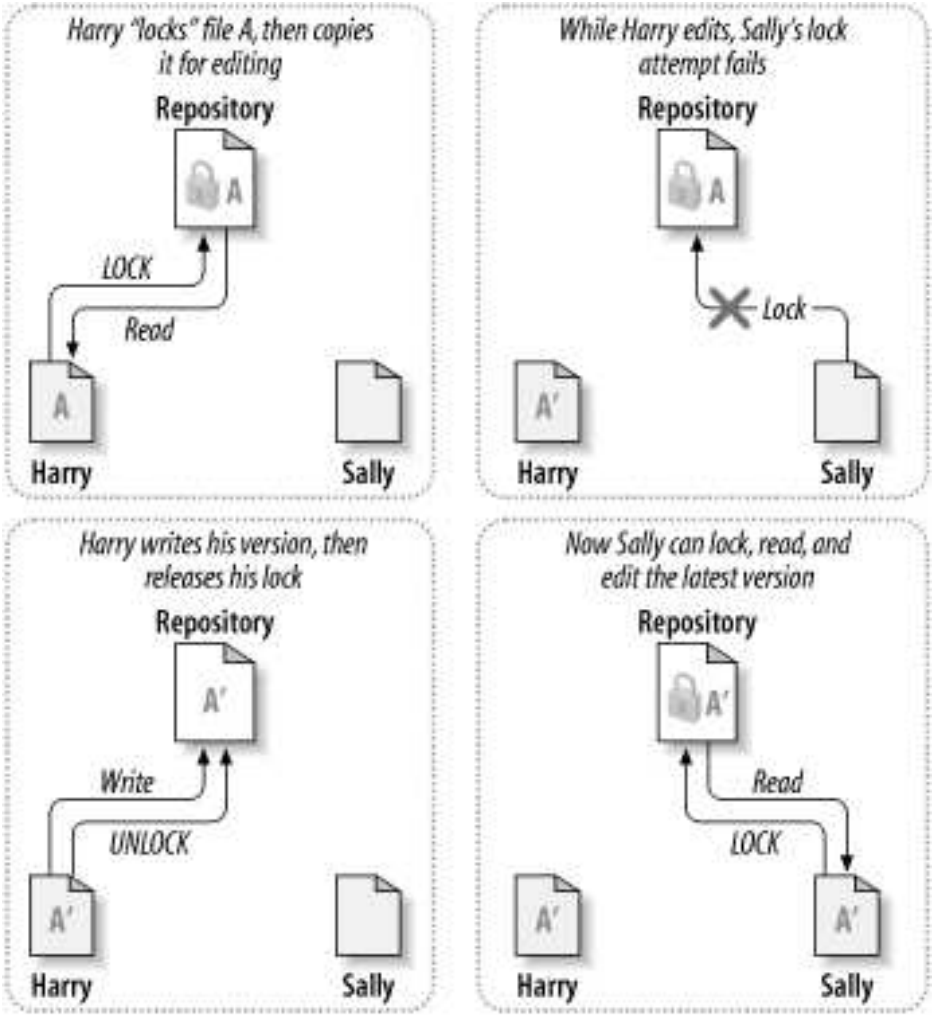
Arquitetura



Problema da concorrência



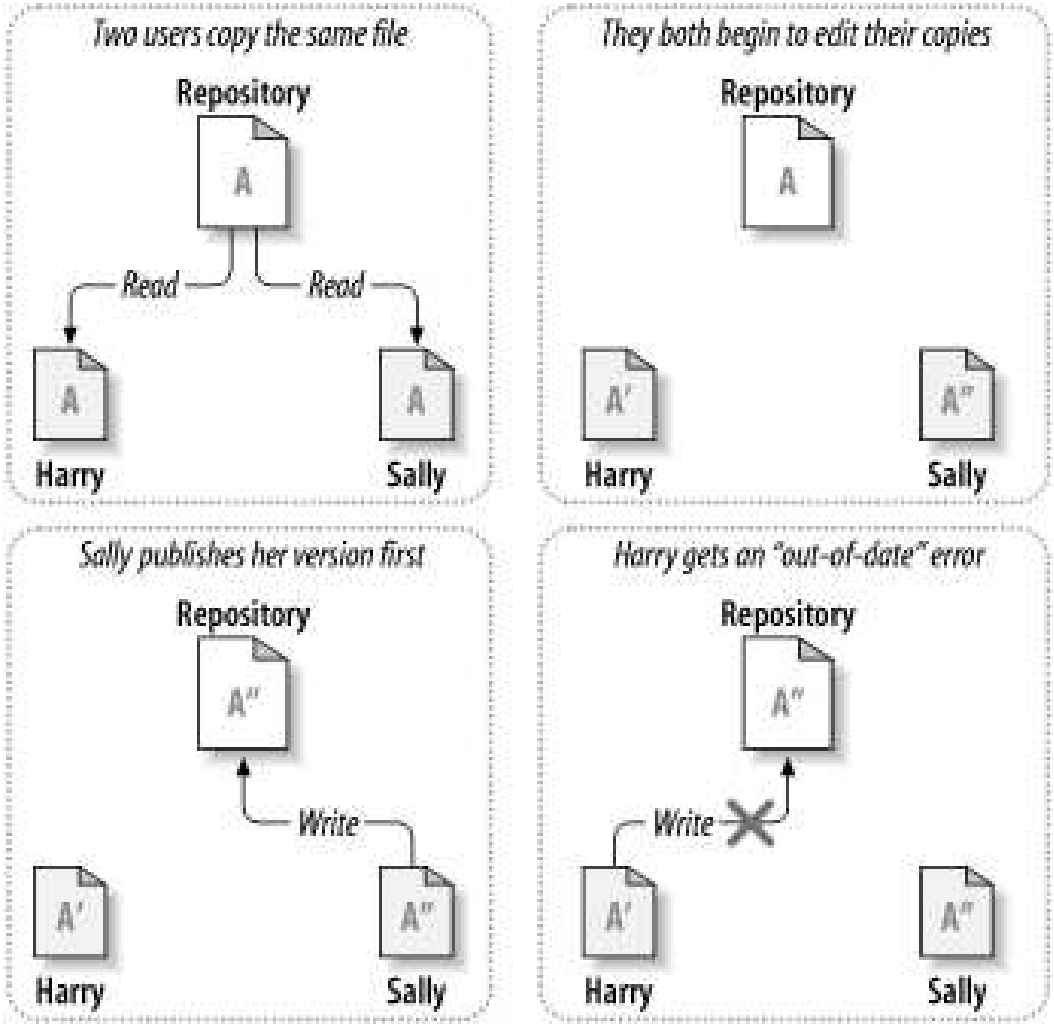
Política pessimista



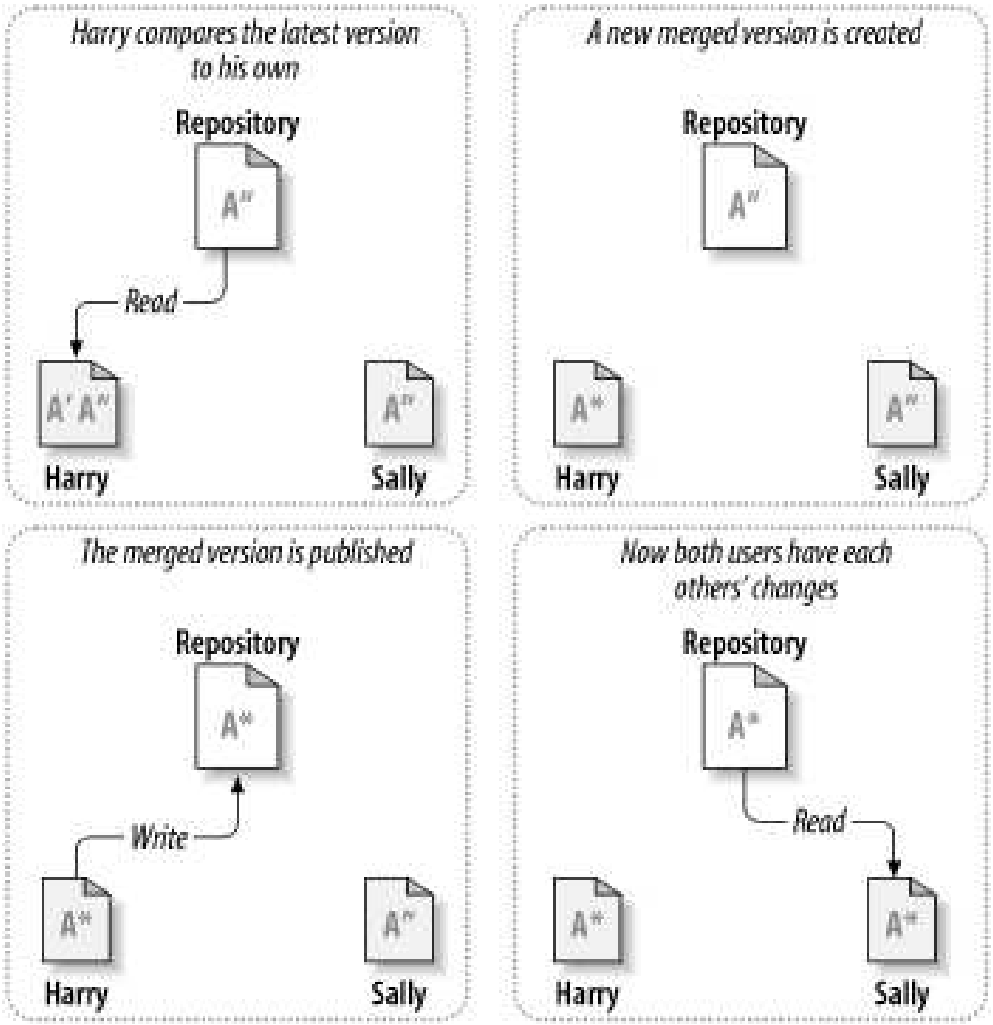
Política pessimista

- Problemas administrativos
  - Bloqueios esquecidos podem atrasar o andamento do projeto
- Problemas de serialização desnecessária
  - Em alguns casos, as modificações atuam sobre partes independentes dos arquivos bloqueados
- Falsa sensação de segurança
  - Dependências semânticas podem cruzar a fronteira de arquivos
- Bloqueios são necessários
  - Quando se trata de arquivos que não podem ser combinados

Política otimista (1/2)



Política otimista (2/2)



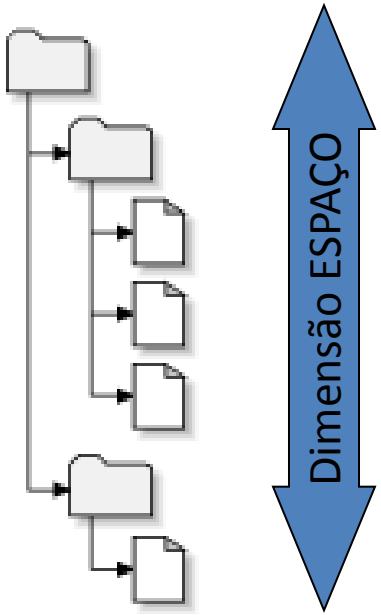
Controle de concorrência no Subversion

- Política pessimista e otimista em conjunto
- Qualquer artefato pode ser sujeito a bloqueio
- Artefatos podem ser demarcados com “necessita de bloqueio”
- Artefatos bloqueados por um desenvolvedor podem ser editados por outros
  - O *commit* dos outros somente ocorrerá depois do término do bloqueio
  - Os outros deverão fazer *merge*
- Artefatos bloqueados podem ser “roubados”
  - Bloqueio apóia à comunicação
  - Bloqueio não engessa o processo

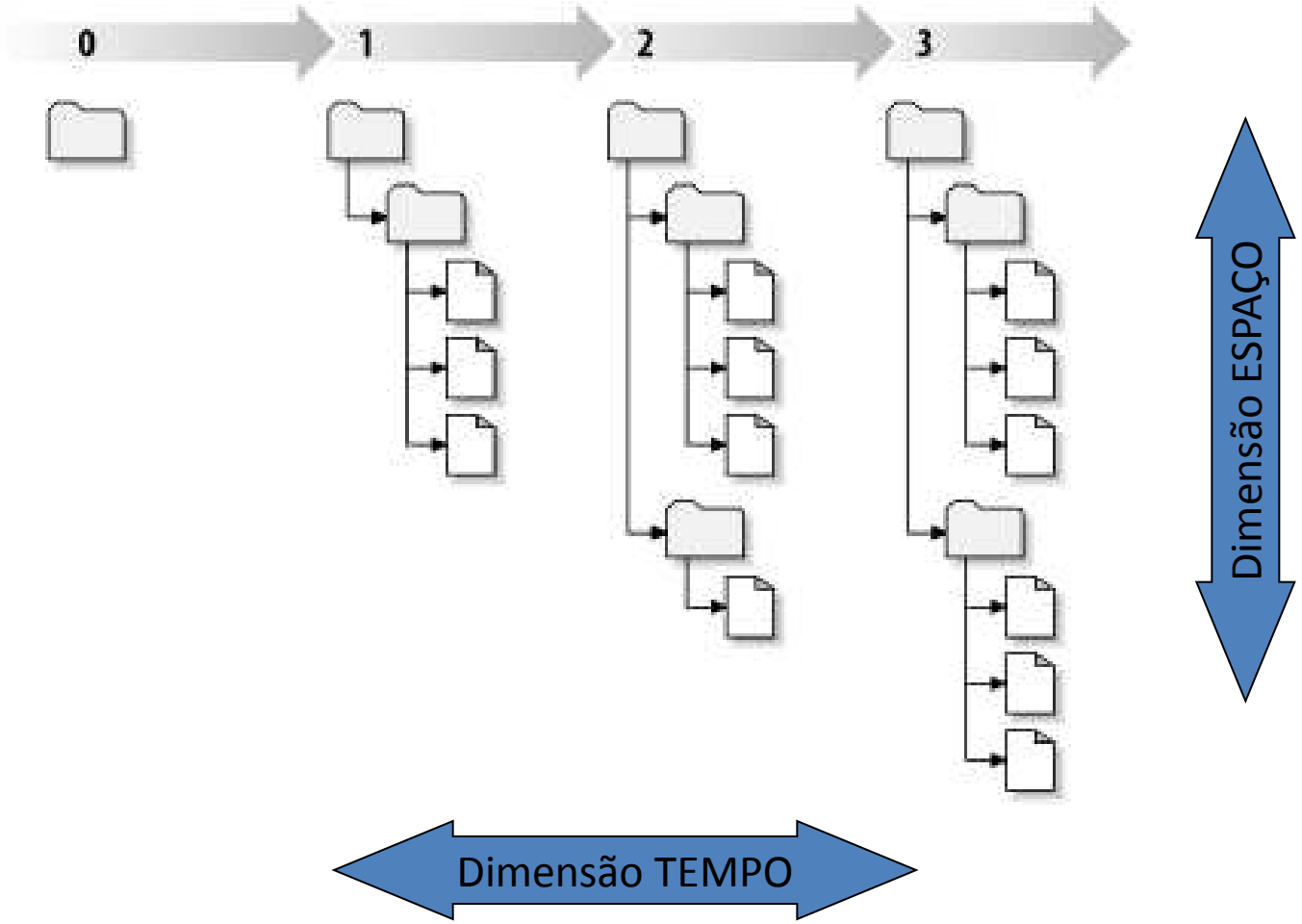
Repositório

- Sistema de arquivos versionado
  - Check-in equivale a uma foto do sistema de arquivos num dado momento
- Versionamento global
  - Número de versão dado por commit
  - Identificador implícito de conjunto de modificações
- Algoritmo “Bubble up”
  - Economia de espaço
  - Velocidade na atualização
  - Controle do histórico

Sistema de arquivo convencional



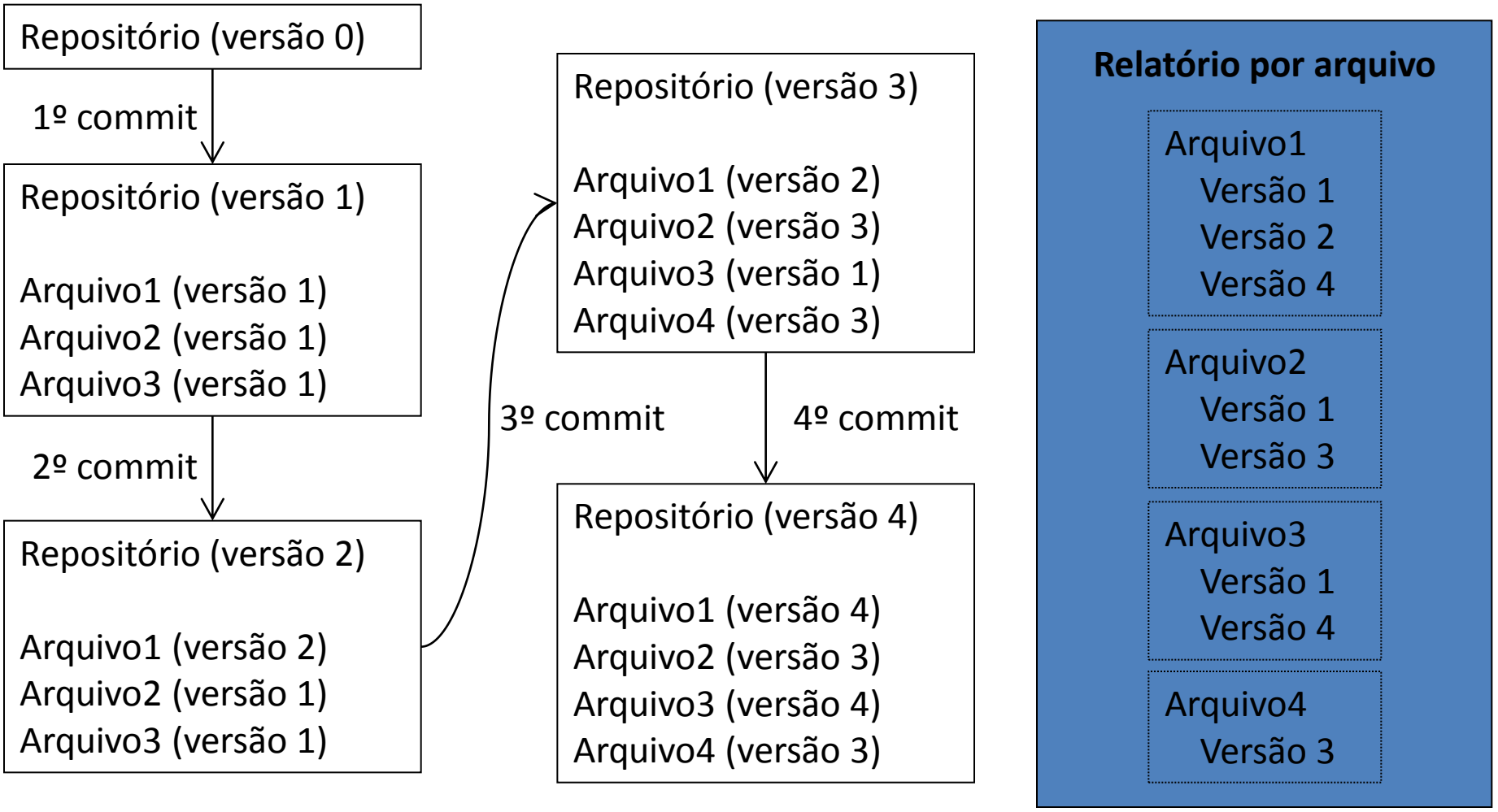
Sistema de arquivo versionado



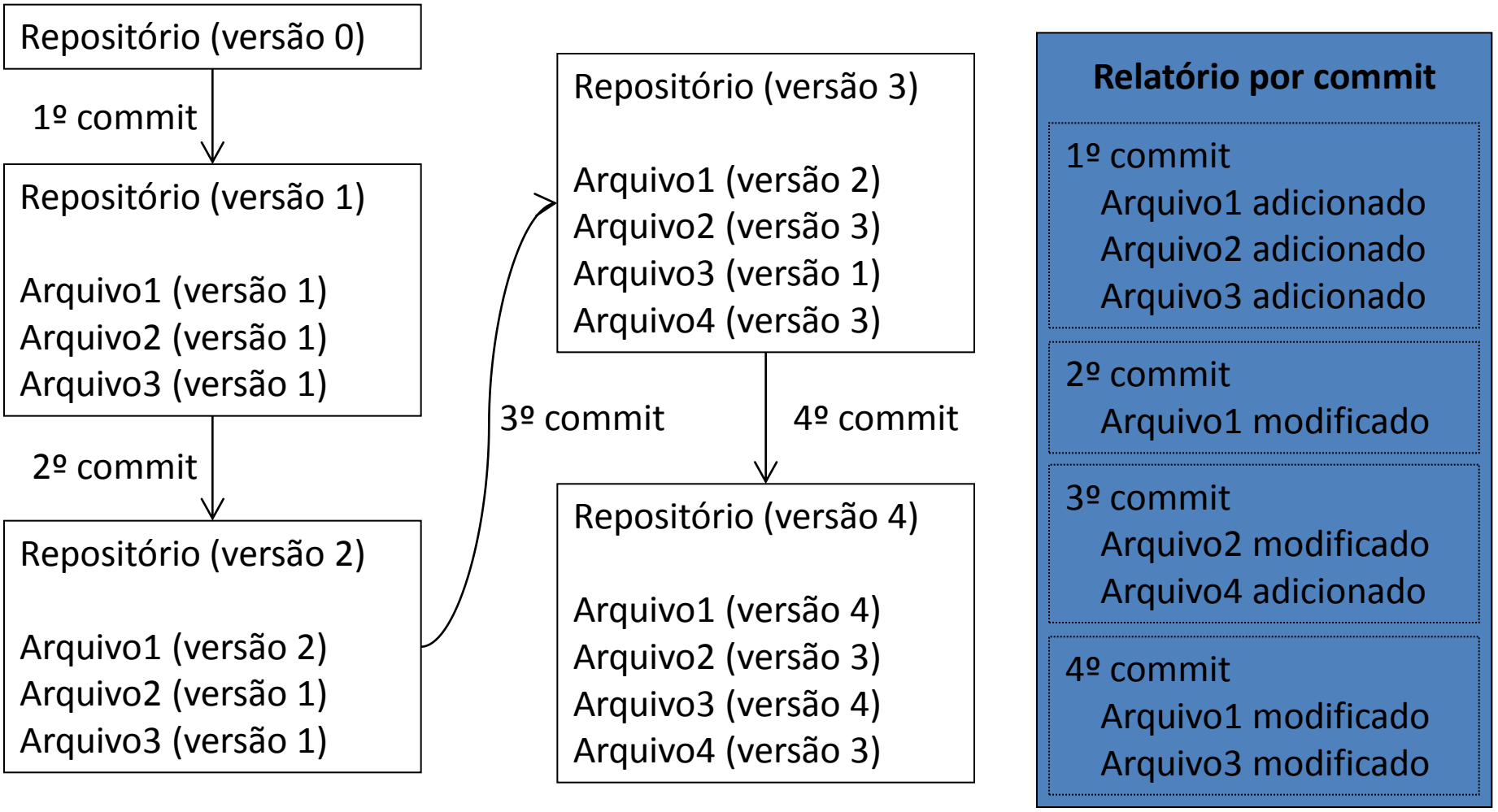
Versionamento global

- O repositório é mais que um conjunto de arquivos independentes
  - Mudança de versionamento de arquivos para versionamento do repositório
  - Versão N é o estado do repositório depois do commit N
  - “Versão 5 do arquivo X” passa a ser lido como “arquivo X na versão 5 do repositório”

Versionamento global



Versionamento global



Comando log

- Fornece um relatório sobre as versões do repositório
  - Por arquivo
  - Por commit
- Exibe, para cada versão
  - Identificação (número da modificação)
  - Quem (autor)
  - Quando (data)
  - Onde (caminhos)
  - Como (ação nos caminhos)
  - O que (mensagem)
  - Por que (número da solicitação de modificação)

Comando log

Sintaxe

```
svn log [-q] [-v] [-r VERSÃO] [URL]
```

Exemplo

```
svn log -q https://reuse.cos.ufrj.br/svn/brecho/trunk/build.xml

r92 | ronaldo | 2007-04-01 17:28:55 -0300 (dom, 01 abr 2007)
r91 | paulacibele | 2007-03-19 12:53:47 -0300 (seg, 19 mar 2007)
r90 | paulacibele | 2007-03-19 12:44:20 -0300 (seg, 19 mar 2007)
r51 | marinho | 2006-01-18 19:03:39 -0200 (qua, 18 jan 2006)
r47 | alexrd | 2006-01-07 10:44:46 -0200 (sáb, 07 jan 2006)
r37 | mlopes | 2005-09-27 00:46:04 -0300 (ter, 27 set 2005)
r31 | alexrd | 2005-09-12 11:15:33 -0300 (seg, 12 set 2005)
...
```

Comando log

Sintaxe

```
svn log [-q] [-v] [-r VERSÃO] [URL]
```

Exemplo

```
svn log -v -r 92 https://reuse.cos.ufrj.br/svn/brecho

r92 | ronaldo | 2007-04-01 17:28:55 -0300 (dom, 01 abr 2007) | 1 line
Caminhos mudados:
   M /trunk/build.xml
   M /trunk/src/br/ufrj/cos/reuse/biblioteca/category/CategoryUtil.java
   A /trunk/src/br/ufrj/cos/reuse/biblioteca/category/Suggestions.java
   M /trunk/src/br/ufrj/cos/reuse/biblioteca/category/dao/HibernateCategoryDAO.java

Issue #234: Troca do algoritmo de sugestão de categorias
```

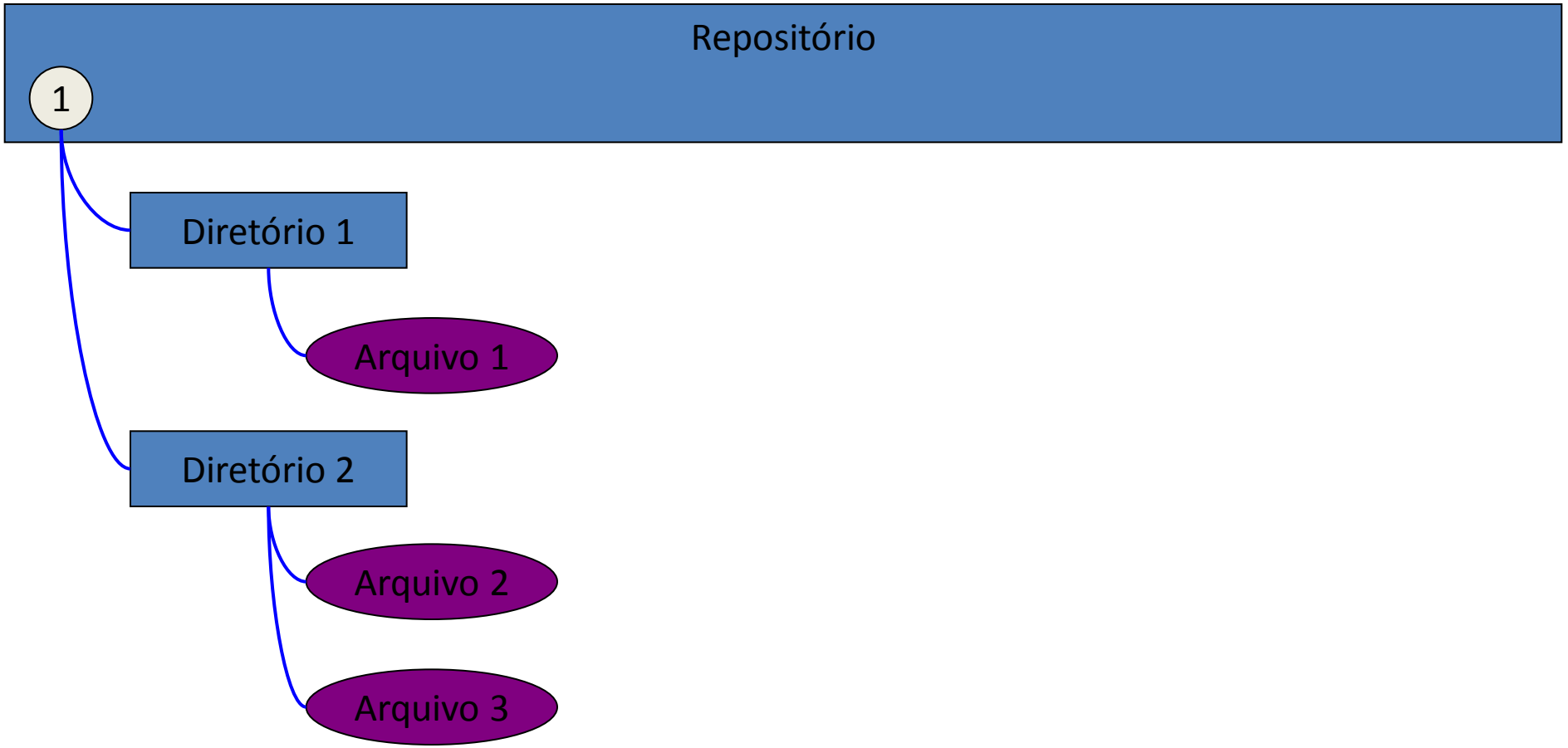
Nomes padrões para versões

- HEAD
  - Versão mais recente do recurso no repositório
- BASE
  - Versão do recurso no espaço de trabalho
- COMMITTED
  - Versão mais recente, anterior ou igual a BASE, em que o recurso foi modificado
- PREV
  - COMMITTED - 1

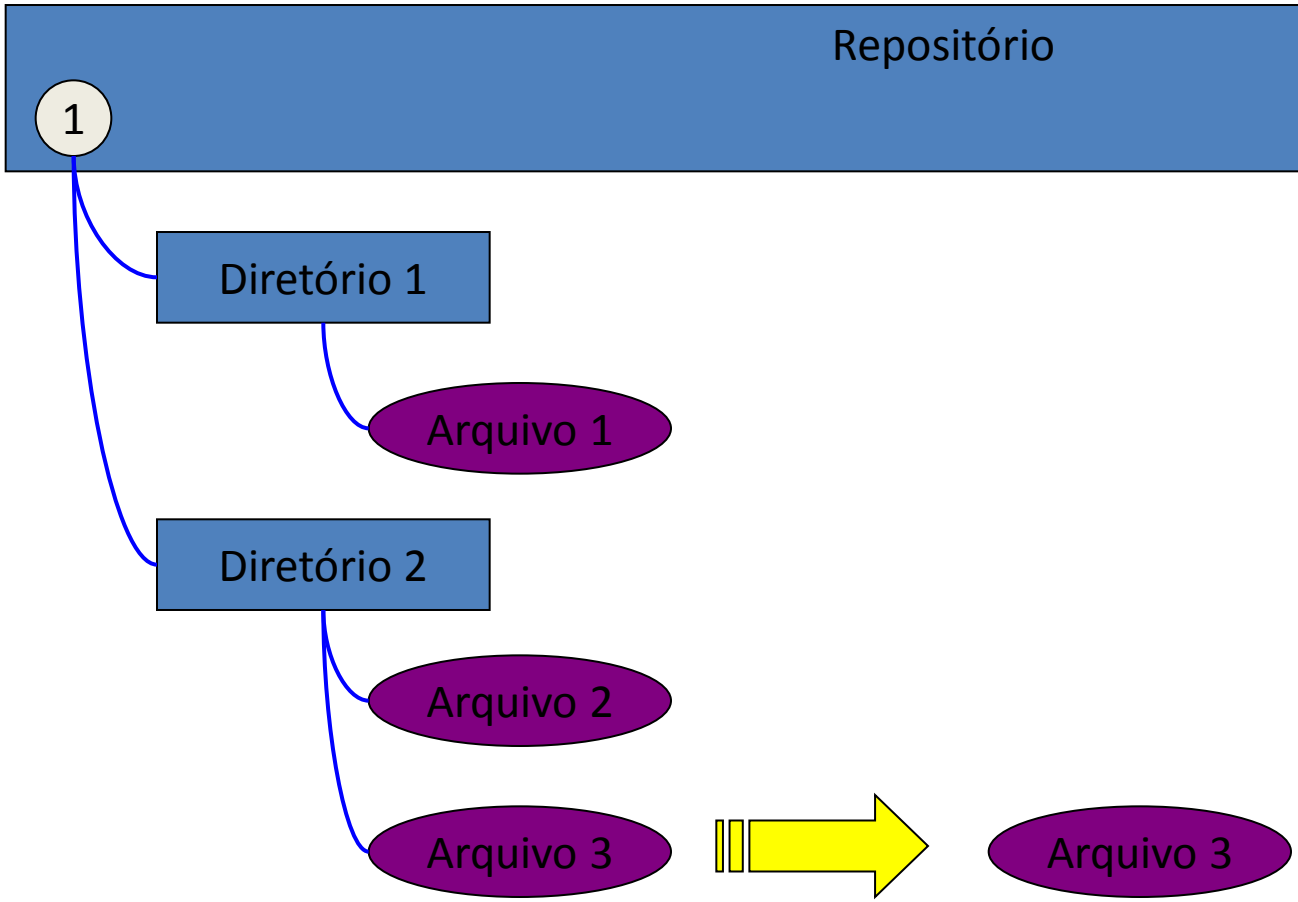
Algoritmo “Bubble up”

- Mecanismo interno do Subversion para controle do repositório
  - Entender esse mecanismo ajuda a entender o funcionamento do Subversion
- Para um dado commit as ações são
  - Aplicadas nas folhas
  - Propagadas para os diretórios pais
- Arquivos e diretórios não afetados pelas modificações são preservados
- Algoritmo utilizado para armazenamento
  - *Reverse delta*

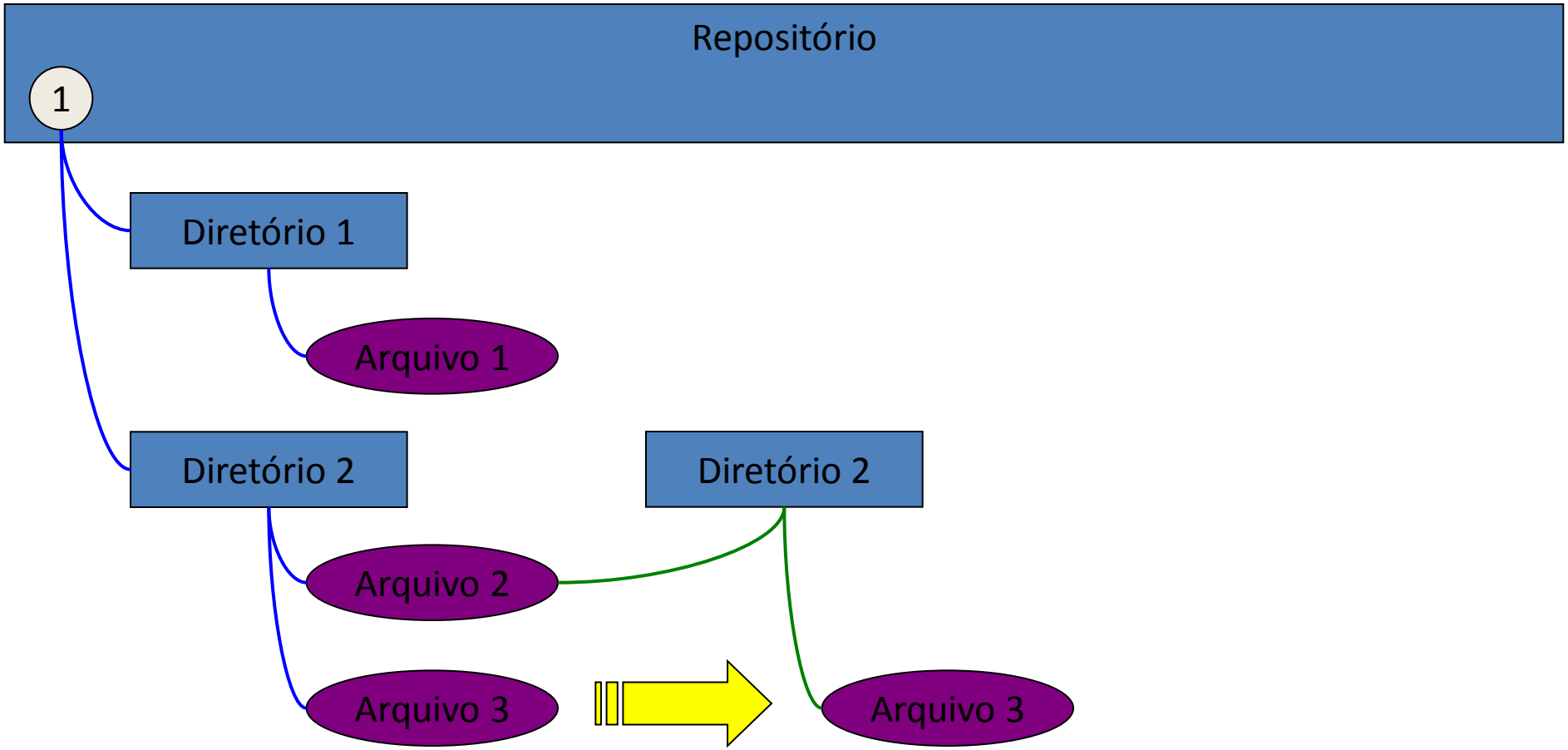
Repositório inicial



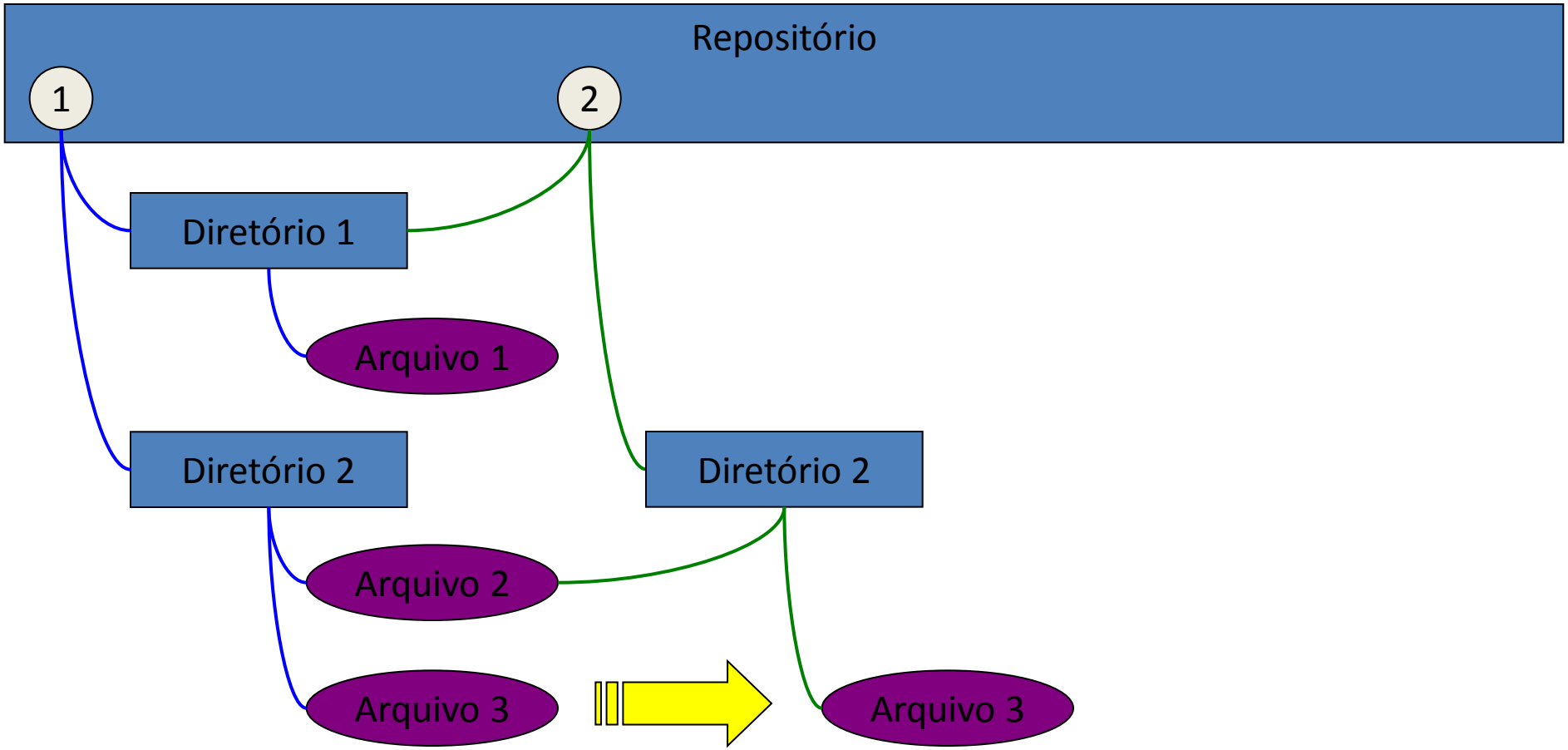
Commit: modificação em um único arquivo



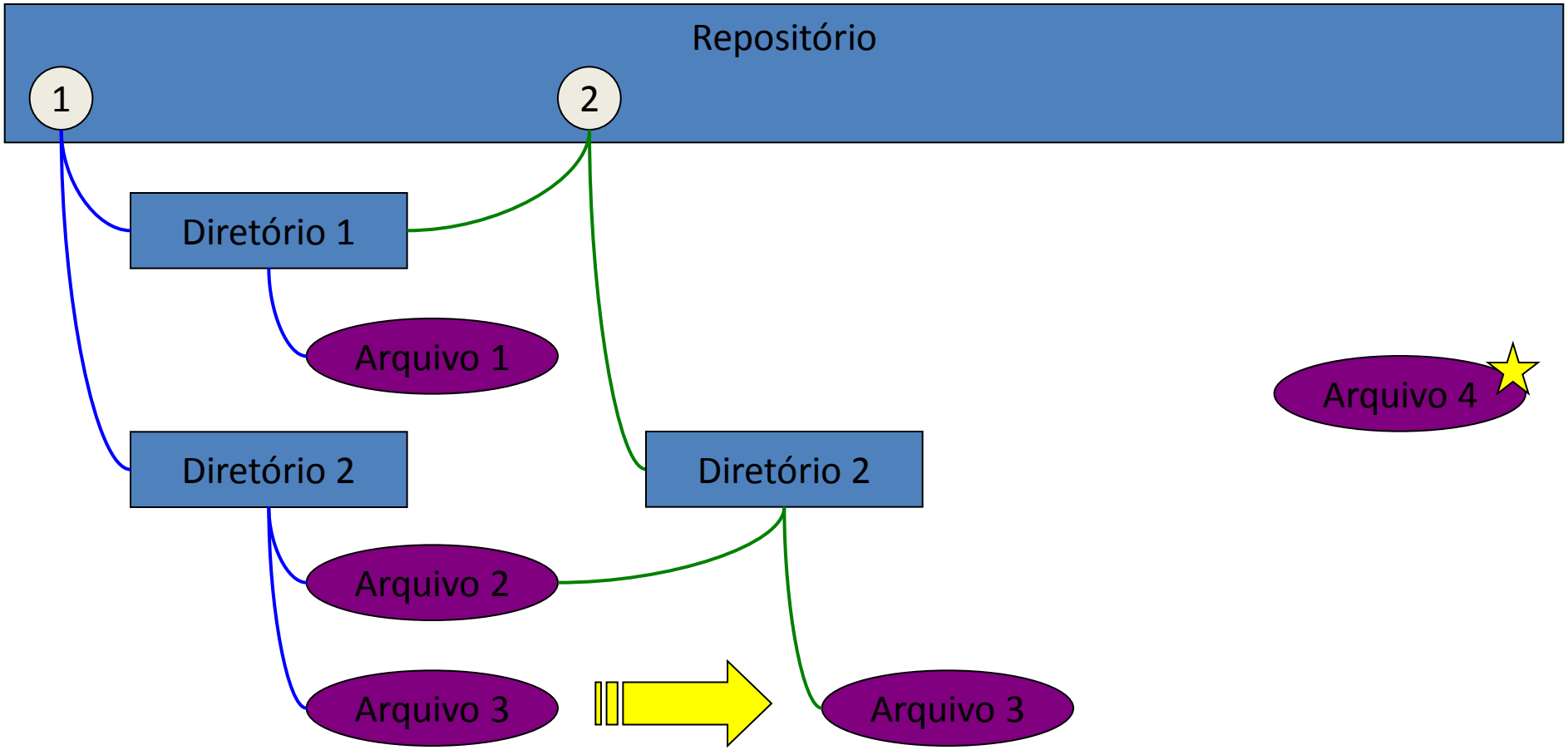
Propagação para o diretório pai



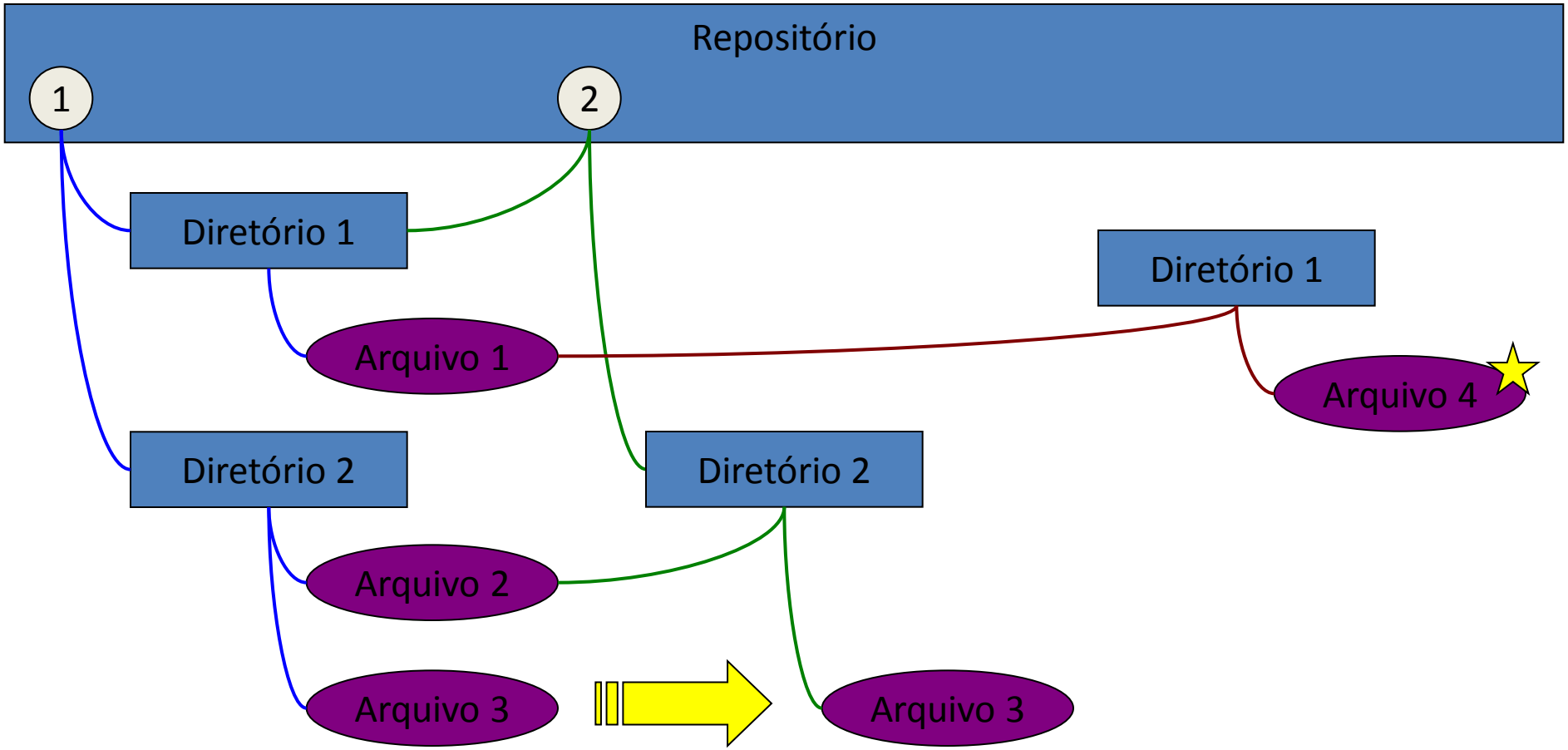
Criação da nova versão



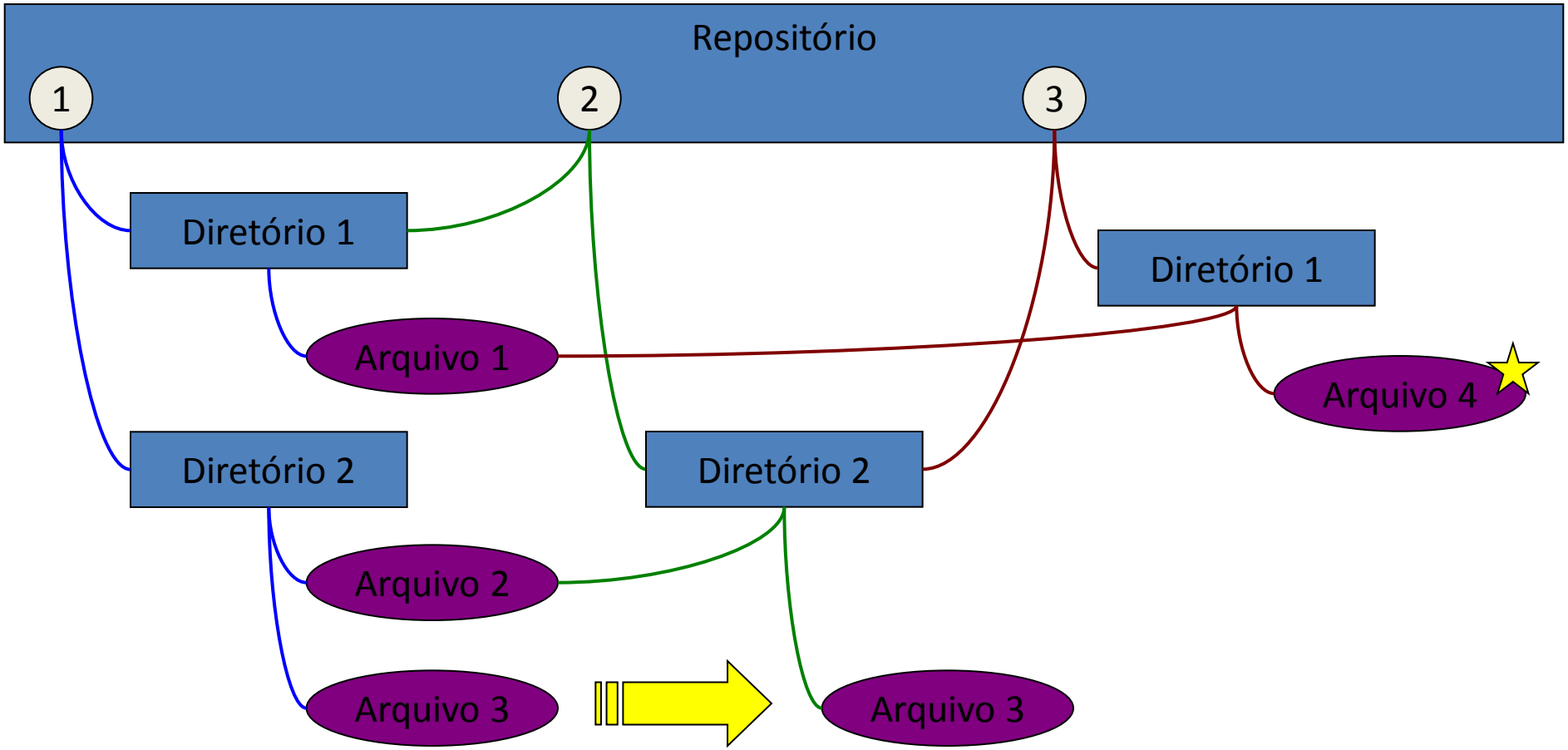
Commit: adição de um novo arquivo



Propagação para o diretório pai



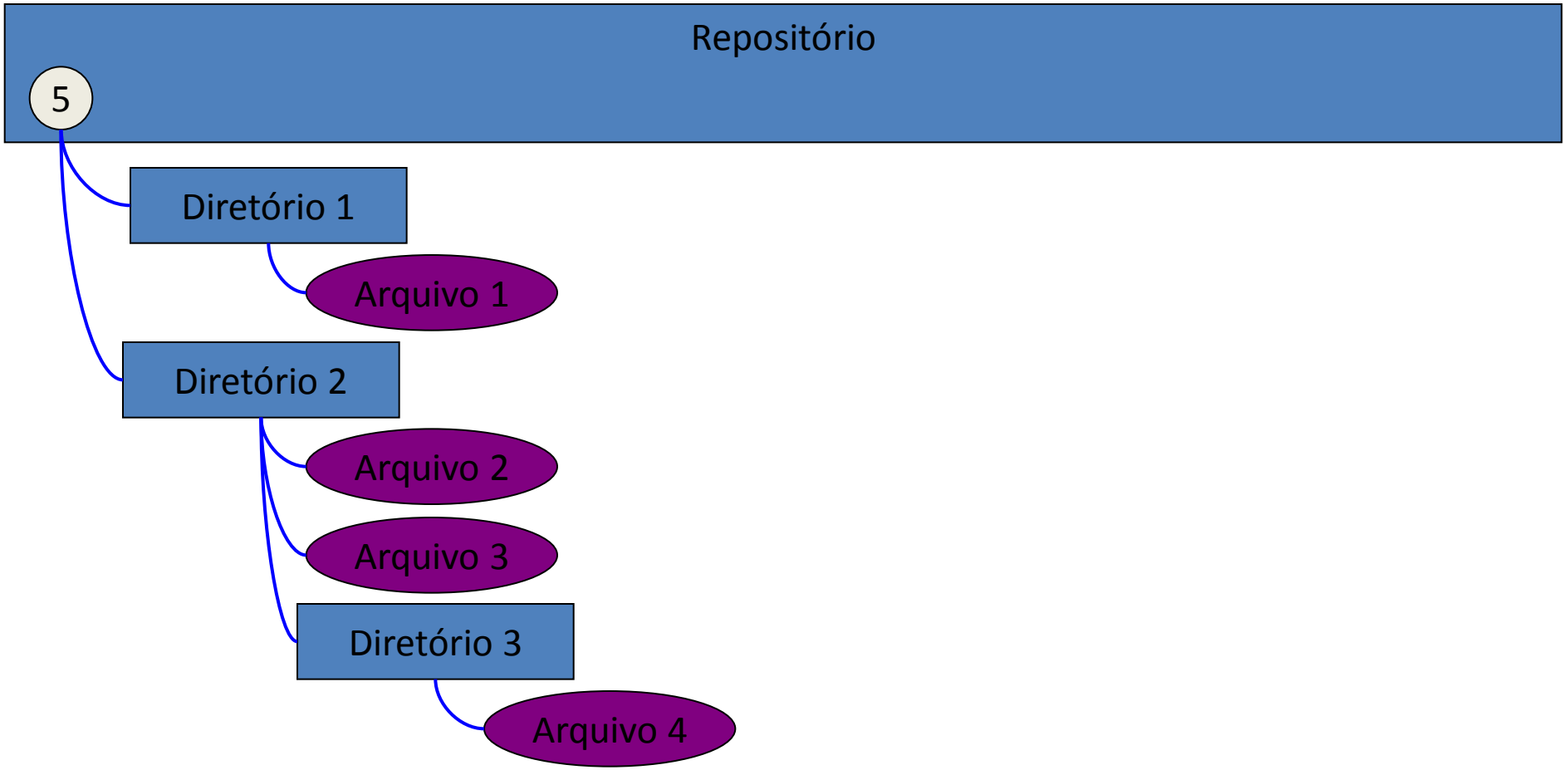
Criação da nova versão



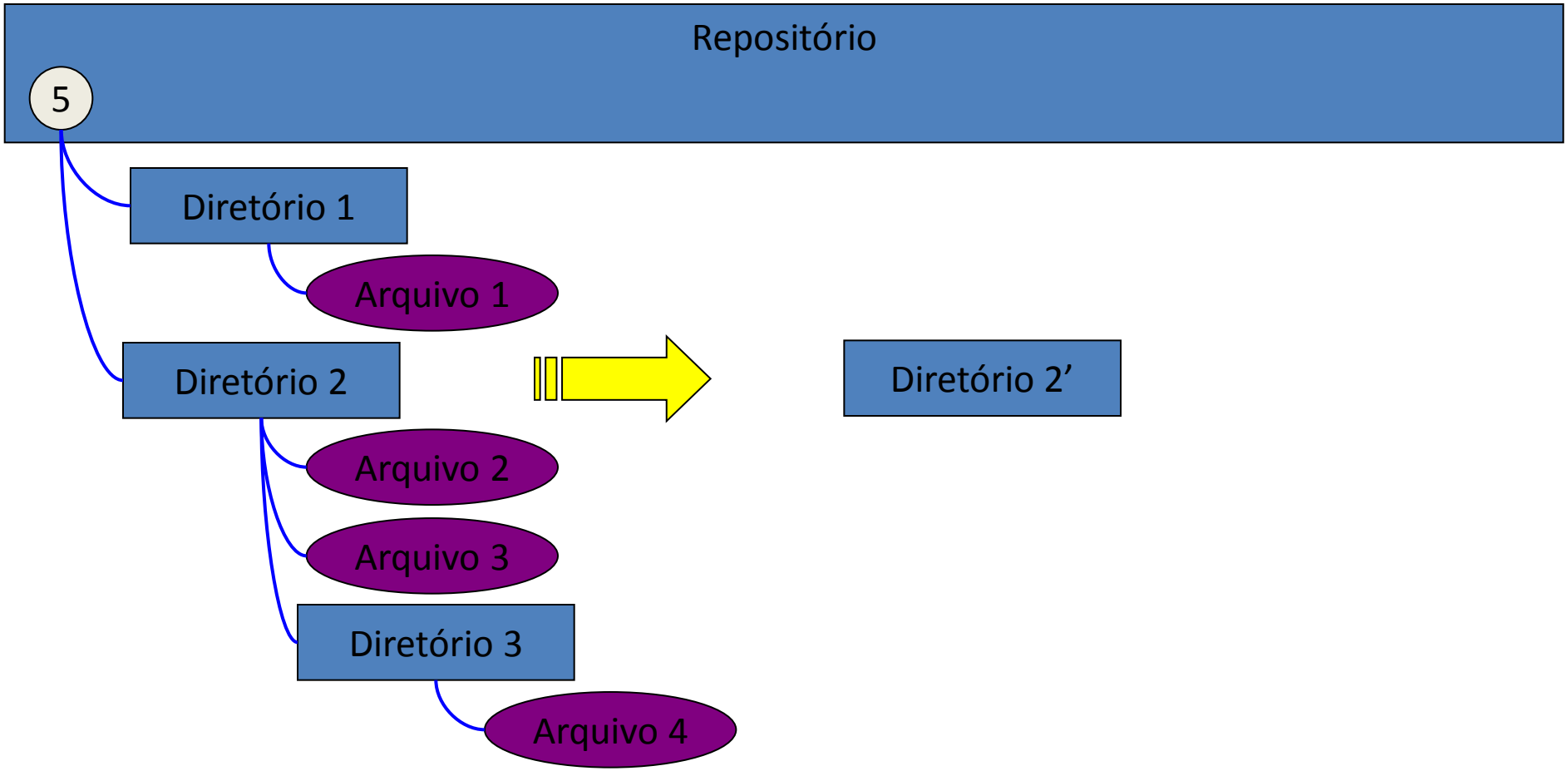
Cópia “barata”

- O projeto interno do Subversion visa prover cópias “baratas” de diretório
  - Os dados não são duplicados
  - Conceito semelhante a *hard-link* do unix
  - Somente quando há mudanças, o link é quebrado
  - Tempo constante gasto para cópias
- Mecanismo utilizado para
  - Etiquetas (*tags*)
  - Ramos (*branches*)

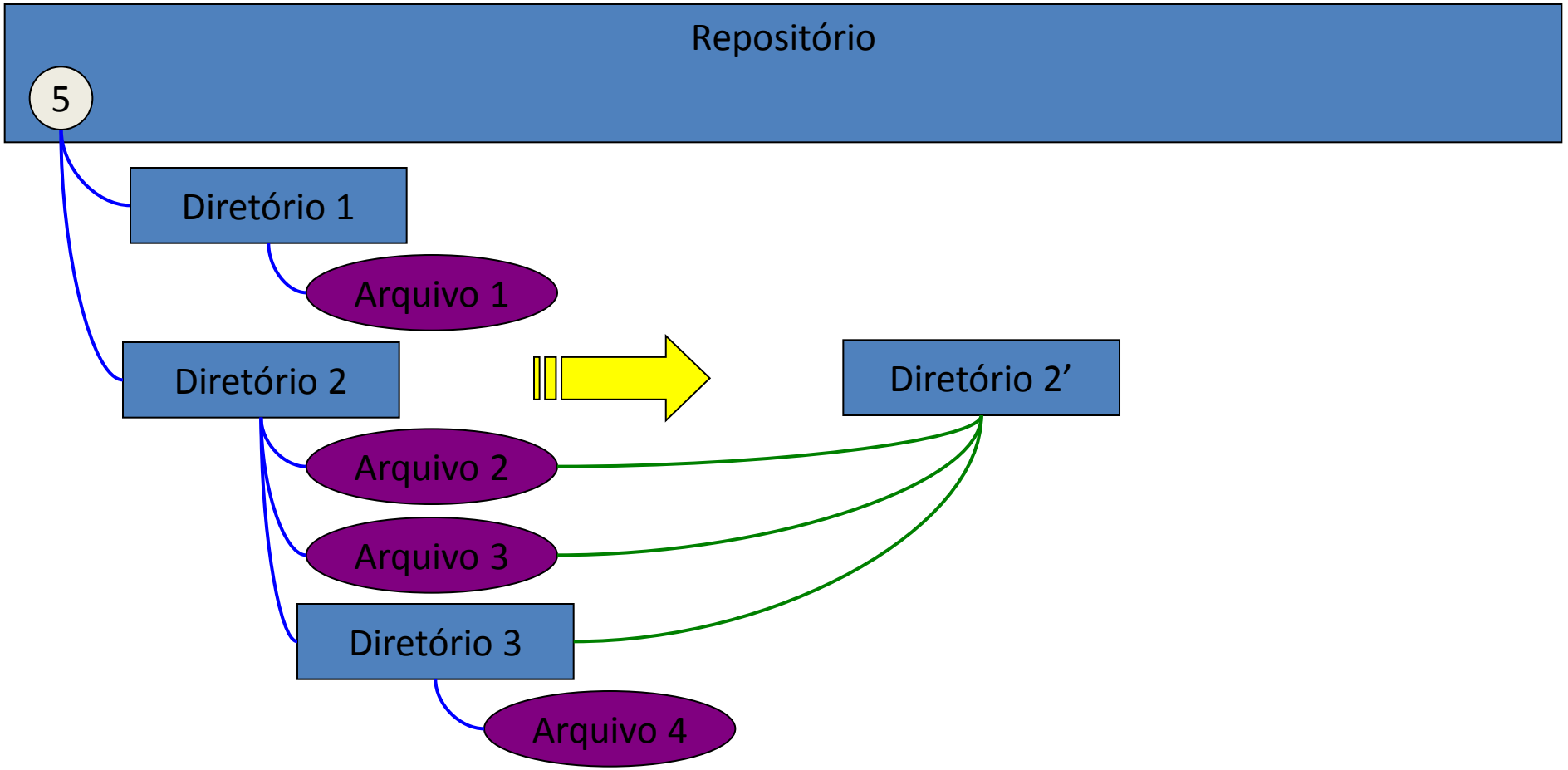
Versão atual do repositório



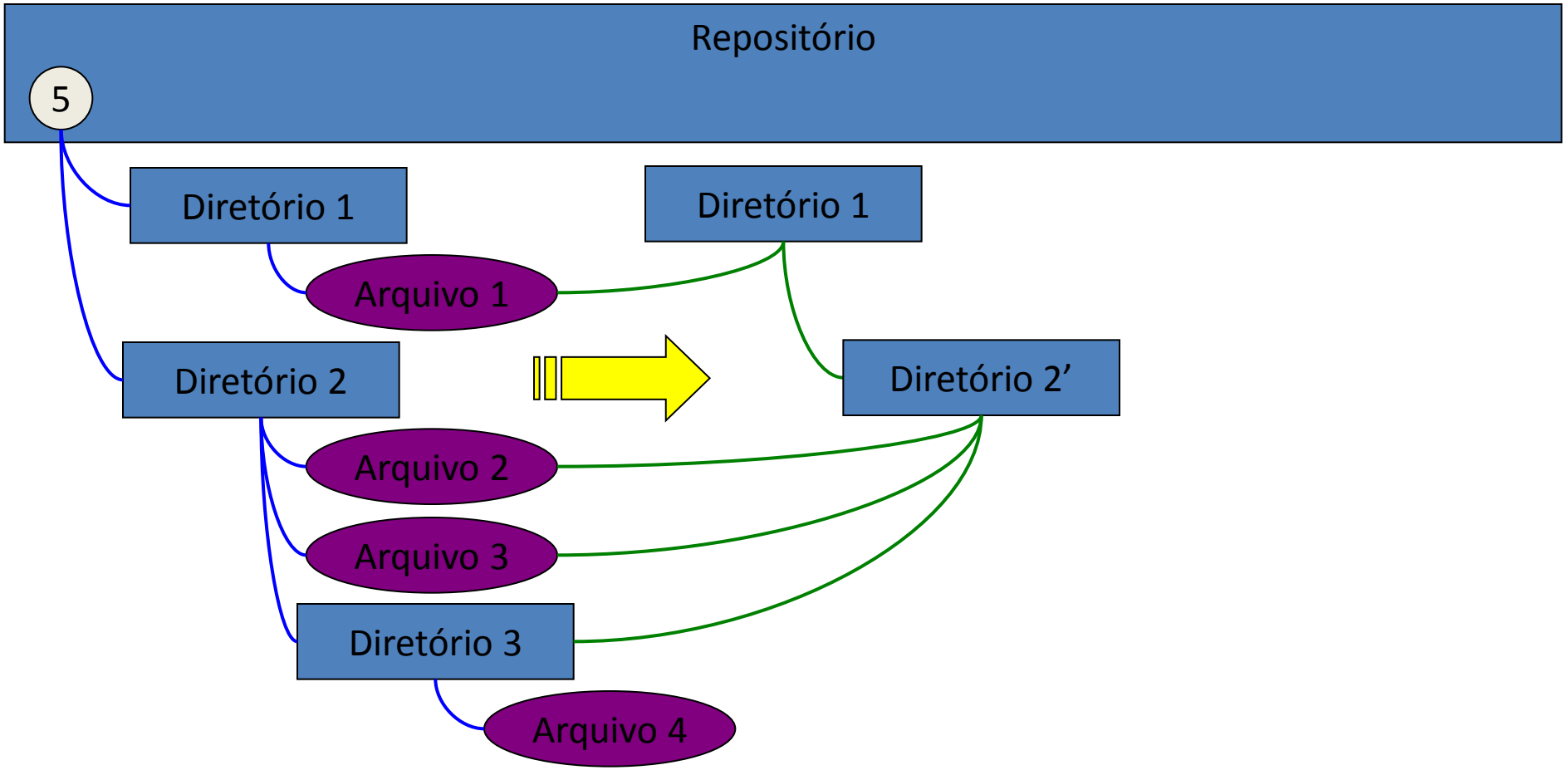
Diretório 2 copiado para dentro do Diretório 1 com outro nome



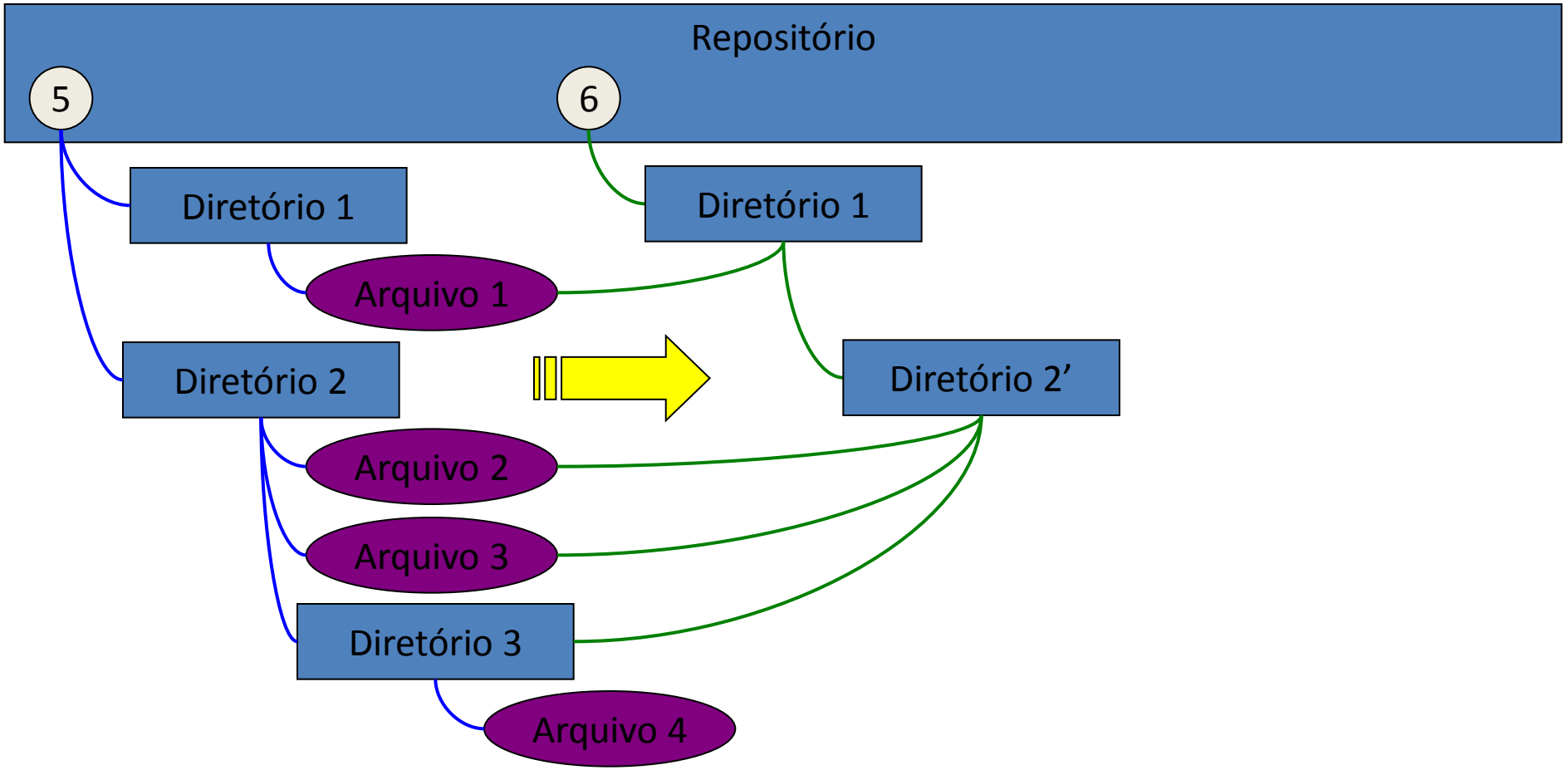
Conteúdo idêntico ao original



Propagação para o diretório pai



Criação da nova versão



Layout do repositório

- Layout é uma convenção
  - Permite mais de um projeto por repositório
  - Ajuda a organizar cada projeto
- Layout sugerido
  - Um diretório por projeto na raiz do repositório
- Layout para cada projeto
  - Um diretório para o ramo principal: *trunk*
  - Um diretório para os ramos: *branches*
  - Um diretório para as etiquetas: *tags*

Layout do repositório

```
https://svn.ic.uff.br/
    proj1/
        trunk/
        branches/
        tags/
    dissertacoes/
        fulano/
            trunk/
            branches/
            tags/
        beltrano/
            trunk/
            branches/
            tags/
        ...
```

Etiquetas e Ramos

- Etiquetas (*tags*)
  - Cópias “baratas” do *trunk* para o diretório *tags*
  - Troca de nome pelo nome da etiqueta
  - Por convenção, nenhuma edição é feita sobre esse diretório copiado
- Ramos (*branches*)
  - Cópias “baratas” do *trunk* para o diretório *branches*
  - Troca de nome pelo nome do ramo
  - Edição no diretório copiado

Comando copy

- Copia um arquivo ou diretório com histórico entre caminhos no repositório ou espaço de trabalho
- Casos típicos
  - URL → URL: cópia executada atomicamente no servidor, utilizada usualmente para criar etiquetas e ramos
  - Diretório local → diretório local: cópia feita no cliente e adição (com histórico) agendada para o próximo commit

Comando copy

Sintaxe

```
svn copy ORIGEM DESTINO [-m MENSAGEM]
```

Exemplo

```
svn copy dir1 dir2/novo_dir_1

svn copy https://svn.ic.uff.br/proj1/trunk
         https://svn.ic.uff.br/proj1/branches/1.0.x
         -m “Criação do ramo 1.0.x”

svn copy https://svn.ic.uff.br/proj1/branches/1.0.x
         https://svn.ic.uff.br/proj1/tags/1.0.1
         -m “Criação da etiqueta 1.0.1”
```

Espaço de trabalho

- Diretório comum no sistema de arquivos local
- Guarda uma cópia limpa do checkout (.svn)
  - Permite algumas operações off-line
  - Permite transmissão de diffs para o servidor
- Área isolada das modificações de outros desenvolvedores
  - Suas modificações podem ser publicadas com *commit*
  - Modificações de outros podem ser incorporadas com *update*
- Um usuário pode ter vários espaços de trabalho para um mesmo projeto

Comando check-out

- Constrói um espaço de trabalho a partir de uma versão do repositório (ou parte dela)

Comando check-out

Sintaxe

```
svn checkout [-r VERSÃO] URL [CAMINHO]
```

Exemplo

```
svn checkout https://svn.ic.uff.br/proj1/trunk

svn checkout -r 15 https://svn.ic.uff.br/proj1/trunk

svn checkout https://svn.ic.uff.br/proj1/tags/1.0.3 rel1.0.3
```

Comando commit

- Envia modificações do espaço de trabalho para o repositório
  - Detecta automaticamente o que mudou
  - Libera todos os bloqueios
  - Aplica a modificação de forma atômica no repositório
- Pode não conseguir enviar caso algum outro usuário tenha dado *commit*
  - Necessário um *update*

Comando commit

Sintaxe

```
svn commit [-m MENSAGEM] [CAMINHO]
```

Exemplo

```
svn commit -m “Adição da versão 1.4.5 do modelo”

svn commit -m “Issue #34: Correção de erro de digitação” src
```

Comando update

- Atualiza o espaço de trabalho com as últimas modificações existentes no repositório
- Pode encontrar conflitos durante a atualização
  - É importante verificar cada um dos arquivos atualizados
- Ações sobre o espaço de trabalho
  - Adição de arquivos (A)
  - Remoção de arquivos (D)
  - Atualização de arquivos (U)
  - Arquivos com conflito (C)
  - Arquivos combinados (G)

Comando update

Sintaxe

```
svn update [-r VERSÃO] [CAMINHO]
```

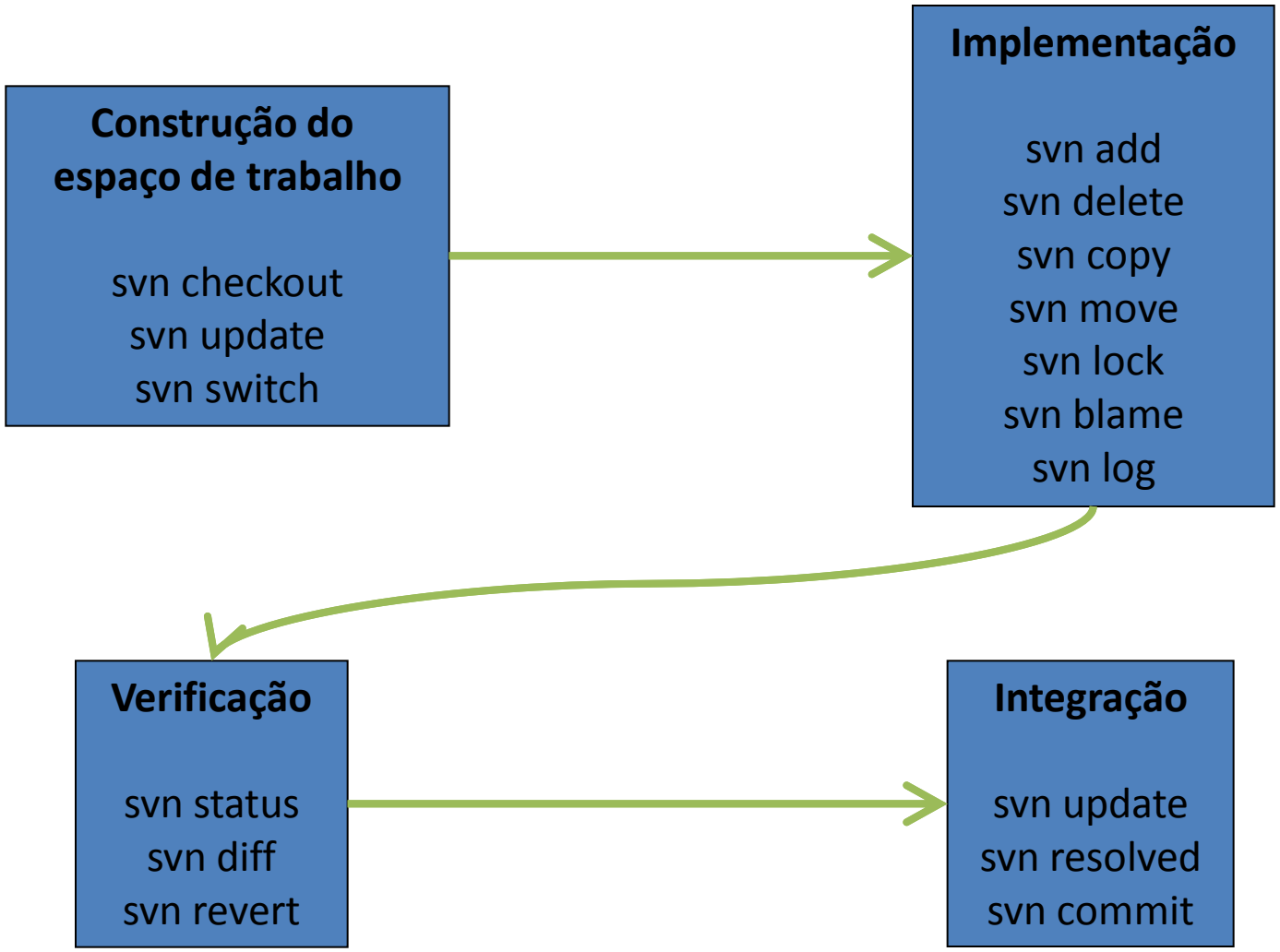
Exemplo

```
svn update

svn update -r 12

svn update dir1
```

Comportamento dos comandos

Arquivo modificado localmente	Arquivo modificado no repositório	Comando *commit*	Commando *update*
Não	Não	Nenhuma ação	Nenhuma ação
Não	Sim	Nenhuma ação	Versão local substituída pela versão do repositório
Sim	Não	Publica a versão local no repositório	Nenhuma ação
Sim	Sim	Falha, avisando que o arquivo está desatualizado	Versão local combinada com a versão do repositório

Ciclo básico de trabalho

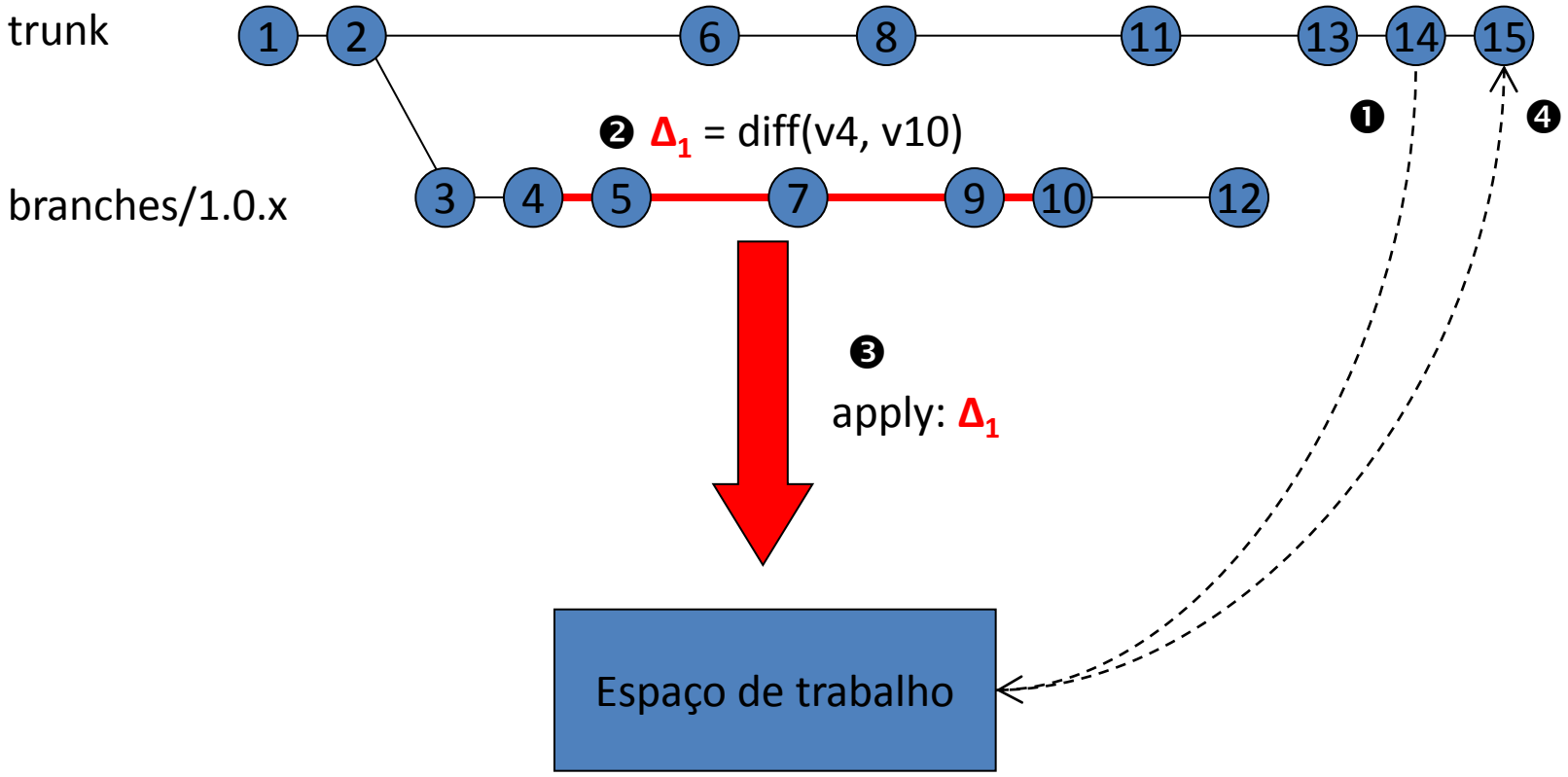


Junção de ramos

- Deve ser feita em um espaço de trabalho limpo
  - Check-out do destino da junção
- Deve ser entendida como *diff & apply*
- *Diff* é dependente de ordem
  - *Diff*(A, B): ações necessárias para transformar o caminho A no caminho B
- O *diff* pode atuar tanto no espaço quanto no tempo
  - Espaço: *diff* entre dois diretórios (copiados de um lugar comum)
  - Tempo: *diff* entre duas versões de um mesmo diretório

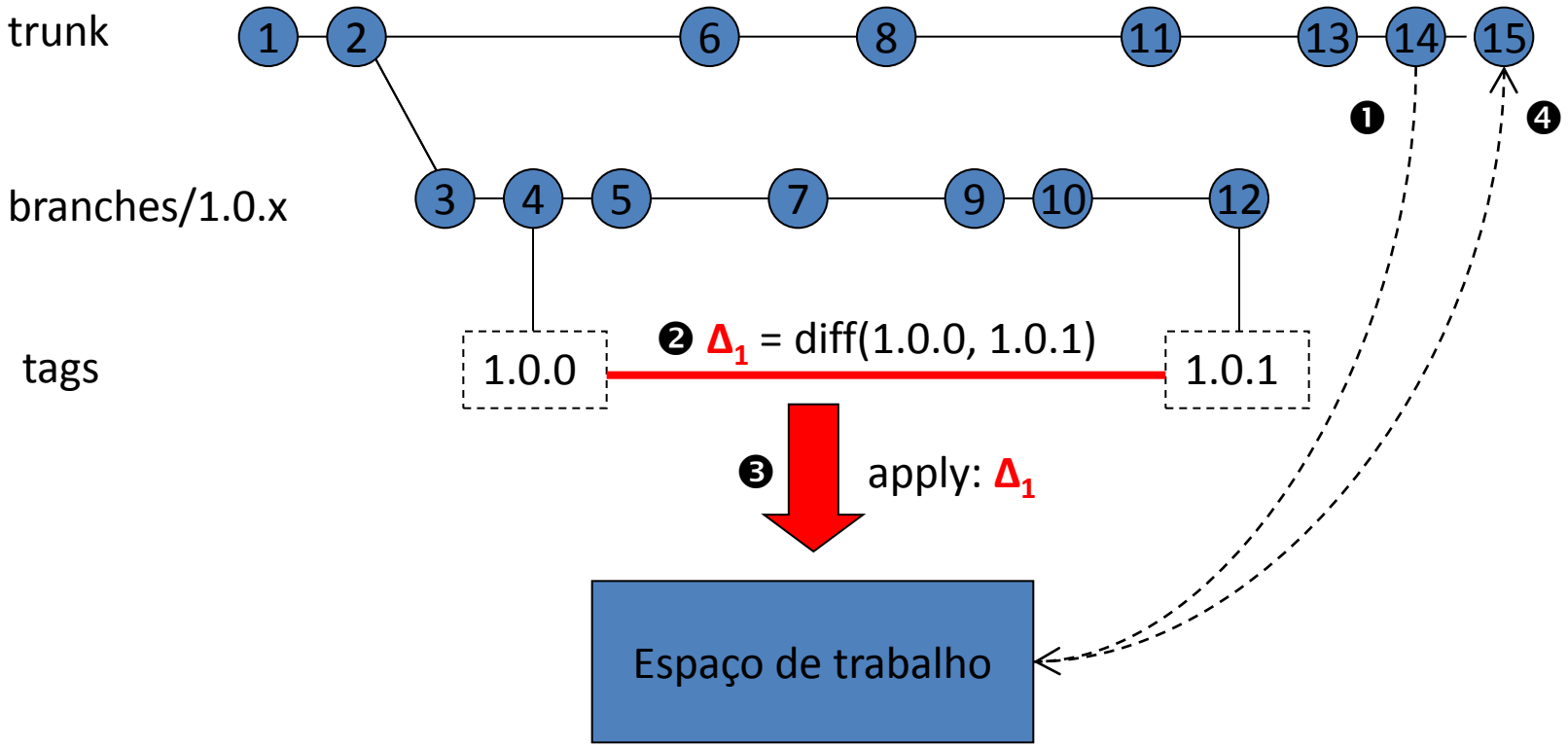
Merge no tempo

- Normalmente usado para ramos



Merge no espaço

- Normalmente usado para etiquetas



Comando merge

- Aplica a diferença entre dois caminhos no espaço de trabalho
- svn merge A B
  - Calcula as ações que precisam ser feitas para transformar o caminho A no B
  - Aplica essas ações no espaço de trabalho

Comando merge

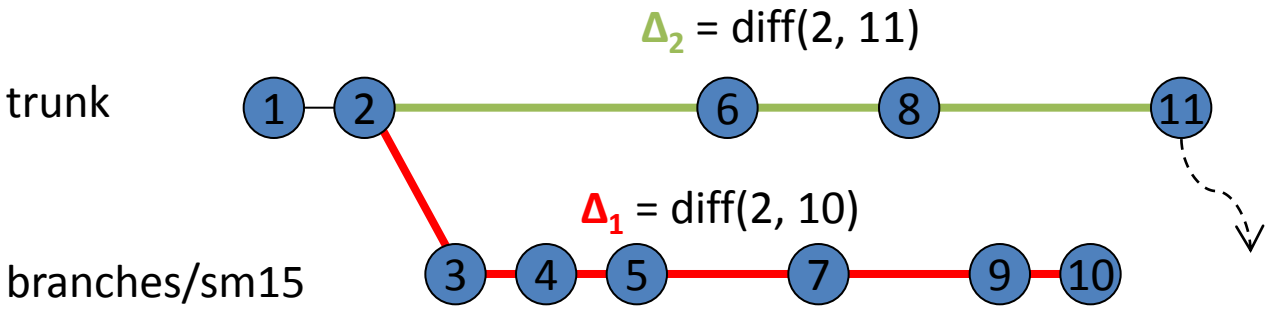
Sintaxe

```
svn merge [-c VERSÃO | -r VERSÃO1:VERSÃO2] CAMINHO
svn merge CAMINHO1 CAMINHO2
```

Exemplo

```
svn merge -c 23 https://svn.ic.uff.br/proj1/branches/1.0.x

svn merge -c -23 https://svn.ic.uff.br/proj1/branches/1.0.x

svn merge -r 23:29 https://svn.ic.uff.br/proj1/branches/1.0.x

svn merge https://svn.ic.uff.br/proj1/tags/1.0.1
          https://svn.ic.uff.br/proj1/tags/1.0.2
```

Sincronização



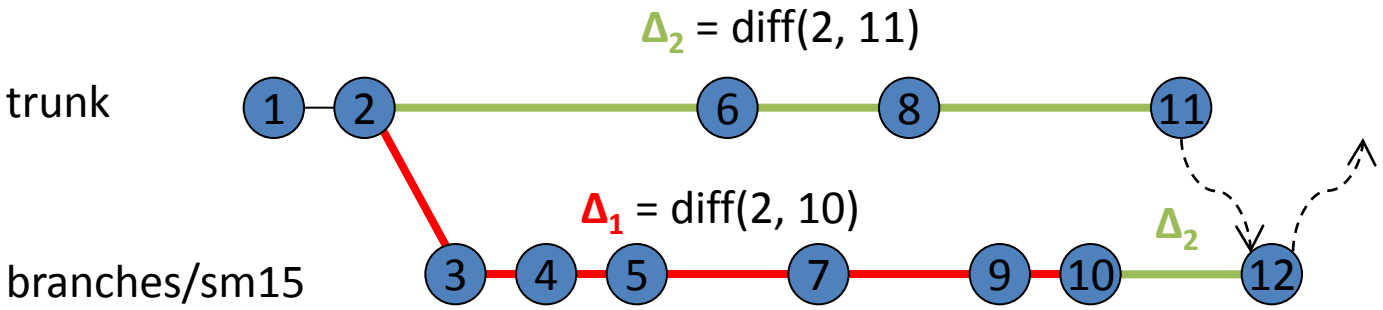
svn checkout https://svn.ic.uff.br/proj1/branches/sm15

Espaço de trabalho = branches/sm15

svn merge -r 2:11 https://svn.ic.uff.br/proj1/trunk

Espaço de trabalho = branches/sm15 + (v11 - v2)
= branches/sm15 + ((v2 + Δ2) - v2)
= branches/sm15 + Δ2

Sincronização



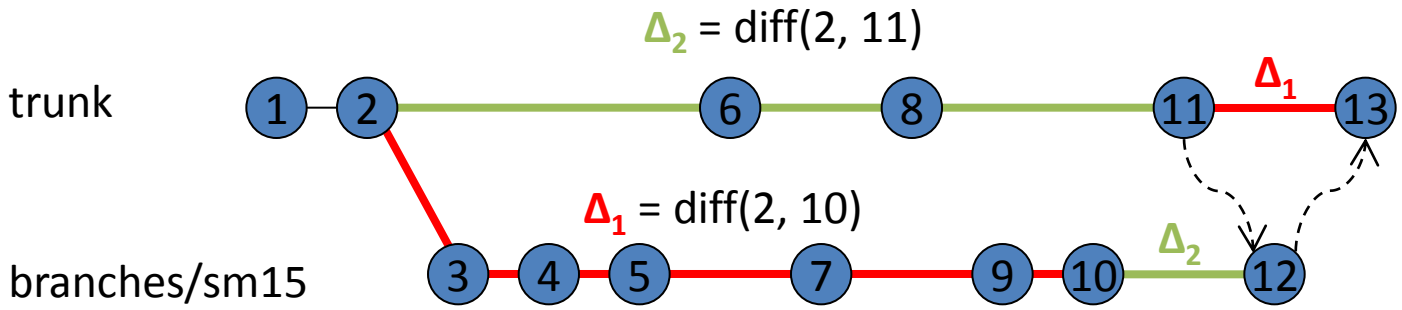
svn checkout https://svn.ic.uff.br/proj1/trunk

Espaço de trabalho = trunk

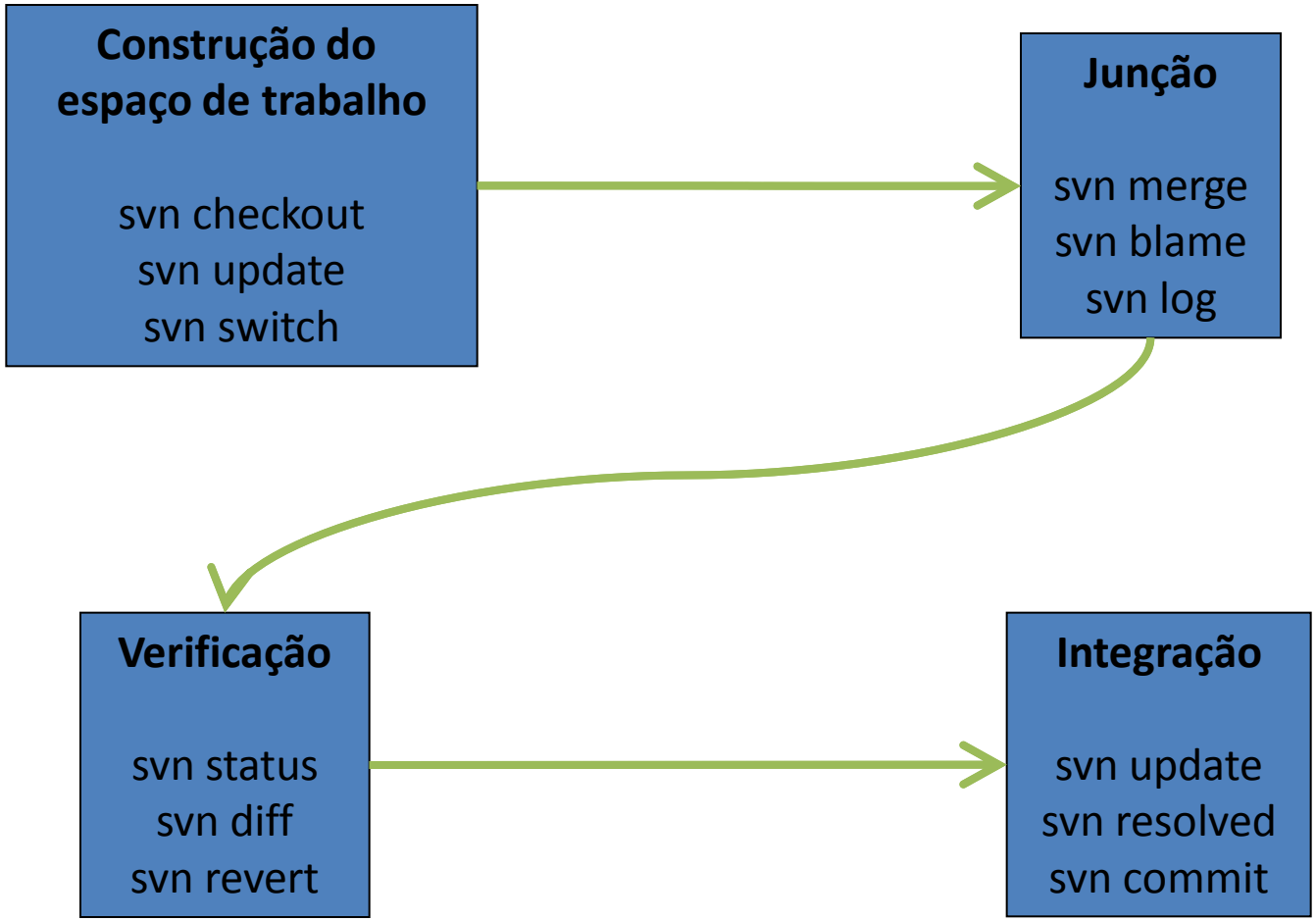
svn merge https://svn.ic.uff.br/proj1/trunk
https://svn.ic.uff.br/proj1/branches/sm15

Espaço de trabalho = trunk + (v12 - v11)
= trunk + ((v2 + **Δ1** + Δ2) - (v2 + Δ2))
= trunk + **Δ1**

Sincronização



Ciclo básico de junção



Propriedades

- Cada diretório ou arquivo pode ter metadados anexados a ele
  - Tuplas <nome, valor>
  - Versionados
- Algumas propriedades *built-in*
  - **svn:mime-type**: tipo do arquivo
  - **svn:ignore**: elementos que não devem ser versionados em um diretório
  - **svn:needs-lock**: indica que o arquivo precisa de política de controle de concorrência pessimista

Referências

- http://svnbook.red-bean.com/en/1.4
- http://subversion.tigris.org/design.html

Instituto de computação

Gerência de Configuração: Subversion

Leonardo Gresta Paulino Murta

leomurta@ic.uff.br