
Engenharia
de
Software

Aula 12

Ementa:
Gerenciamento de configuração e controle de versões.

Objetivo Aula

Gerenciamento de configuração e controle de versões.

Soluções para realizar o controle: Git e SubVersion (SVN).

Plano de Ensino e Planejamento da Disciplina

MÓDULO 4 – Gerenciamento de configuração e controle de versões

Aula	Data	CONTEÚDO / TEMA DA AULA
12	09/05	Gerenciamento de configuração e controle de versões; Planejamento de gerenciamento de configurações.
13	16/05	Ferramentas para gerenciamento de configurações.
14	23/05	Avaliação P2.
15	06/06	XIX Congresso FHO / Semana Científica do Curso de Sistemas de Informação.
16	13/06	Correção P2. Avaliação dos Resultados Obtidos e Lições Aprendidas. Aplicação da Prova Substitutiva.
17	20/06	Fechamento do Semestre. Atendimento aos Alunos.
18	27/06	Fechamento do Semestre.

Gerência de Configuração

- Gerência de configuração (GC) é o processo de **identificar, organizar e controlar** modificações ao software sendo construído

- A ideia é maximizar a produtividade minimizando os enganos no processo diário.

Objetivos Gestão de Configuração

- Compreender a importância do uso de mecanismos de gerência de configuração (GC) e de mudança, seus métodos, processos e ferramentas.
- Fornecer os principais conceitos relacionados a GC.
- Criar uma visão geral de como GC pode ser aplicada a um projeto de software.

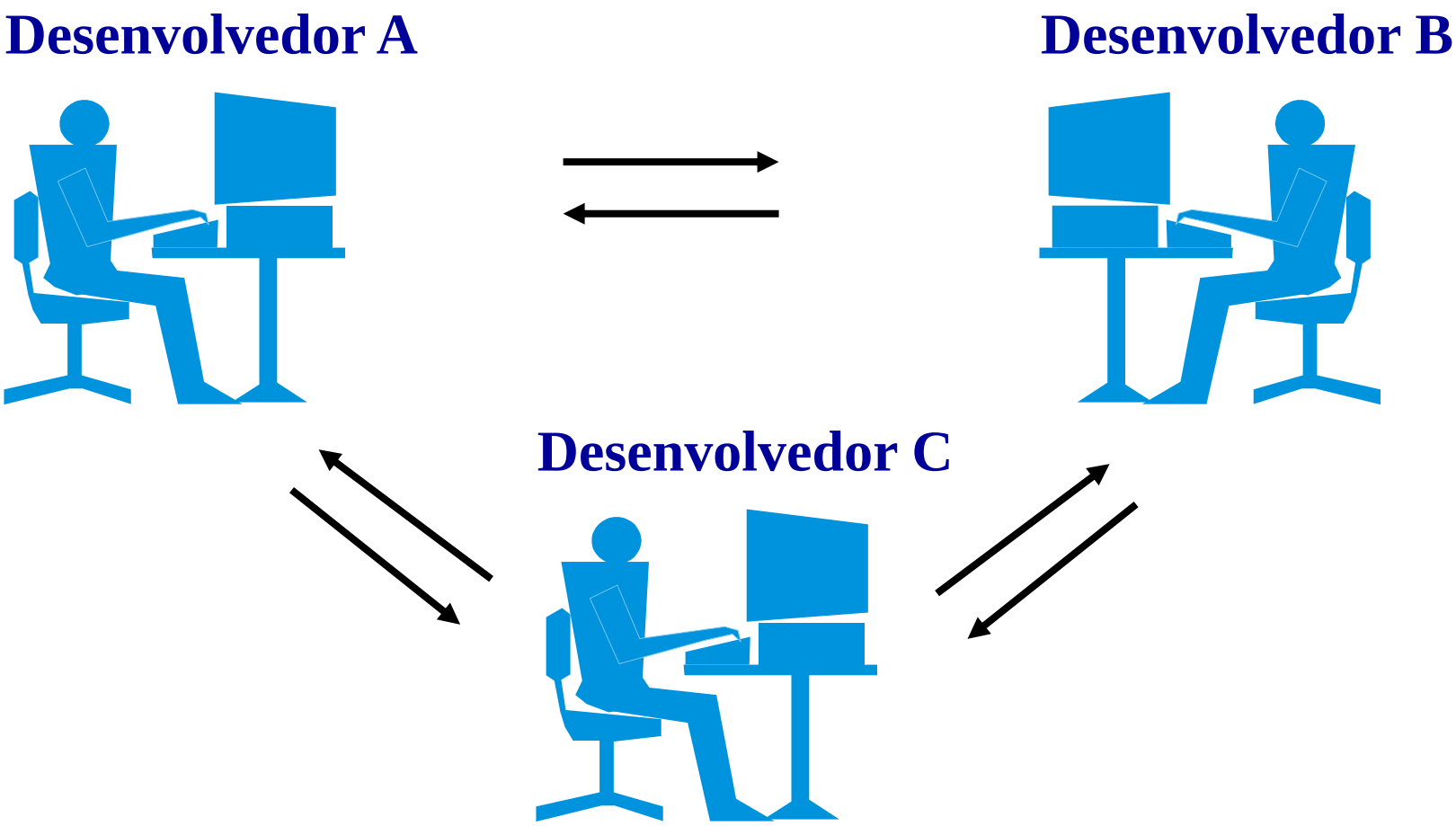
Objetivos Gestão de Configuração

- Definir o ambiente de desenvolvimento.
- Definir políticas para controle de versões, garantindo a consistência dos artefatos produzidos.
- Definir procedimentos para solicitações de mudanças.
- Administrar o ambiente e auditar mudanças.
- Facilitar a integração das partes do sistema.

Benefícios

- Aumento de produtividade no desenvolvimento
- Menores Custos de Manutenção
- Redução de defeitos
- Maior rapidez na identificação e correção de problemas

Problema a ser resolvido

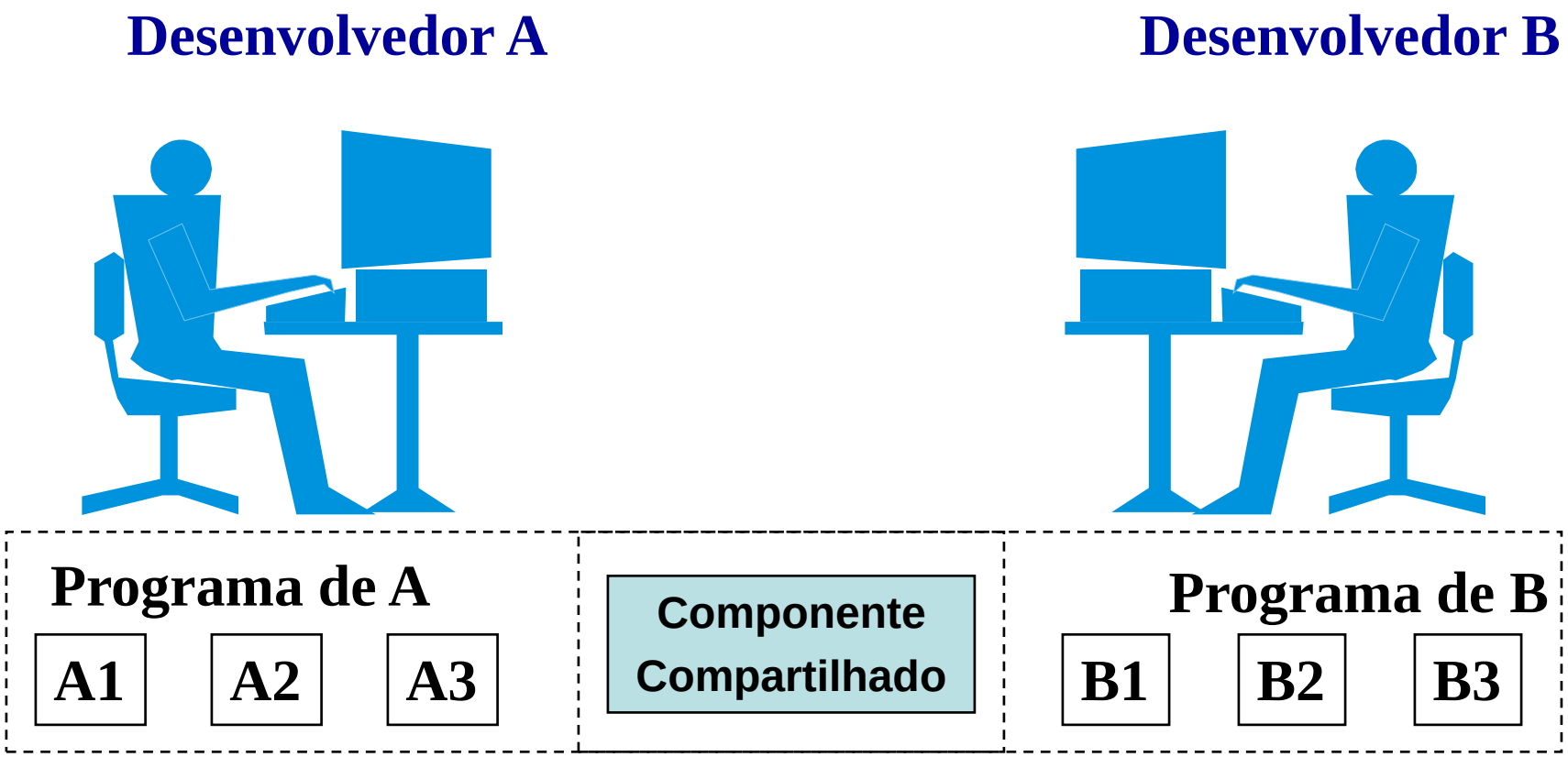


Problema da Quebra de Comunicação (continuação)

- Falhas de comunicação em equipes
- Ocorre pelas mais diversas razões:
  - Vocabulários incompatíveis
  - Culturas de desenvolvimento diferentes
  - Distância geográfica
  - Dificuldade de expressão

- Quando este problema acontece:
  - Os sistemas produzidos não atendem aos requisitos
  - Força de trabalho é desperdiçada

Problema 2: Dados Compartilhados



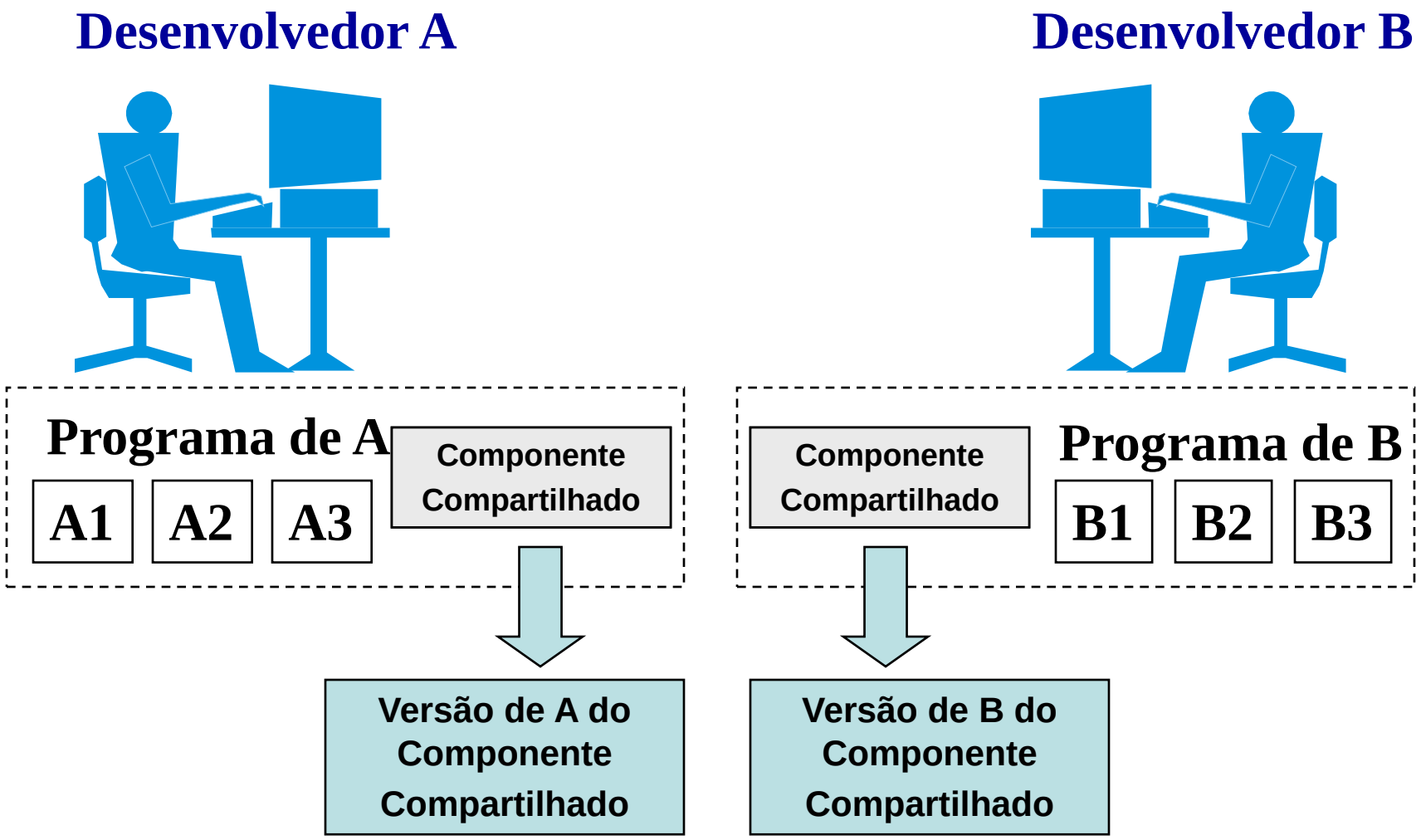
Problema 2

- O desenvolvedor A modifica o componente compartilhado
- Mais tarde, o desenvolvedor B realiza algumas alterações no mesmo
- Ao tentar compilar o componente, erros são apontados pelo compilador, mas nenhum deles ocorre na parte que B alterou
- O desenvolvedor B não tem a menor ideia sobre a causa do problema

Problema dos Dados Compartilhados - Solução simplista

- **Solução simplista:**
  - **cada desenvolvedor trabalha em uma cópia "local" do componente**
  - **resolve o Problema dos Dados Compartilhados, mas cria um novo problema...**

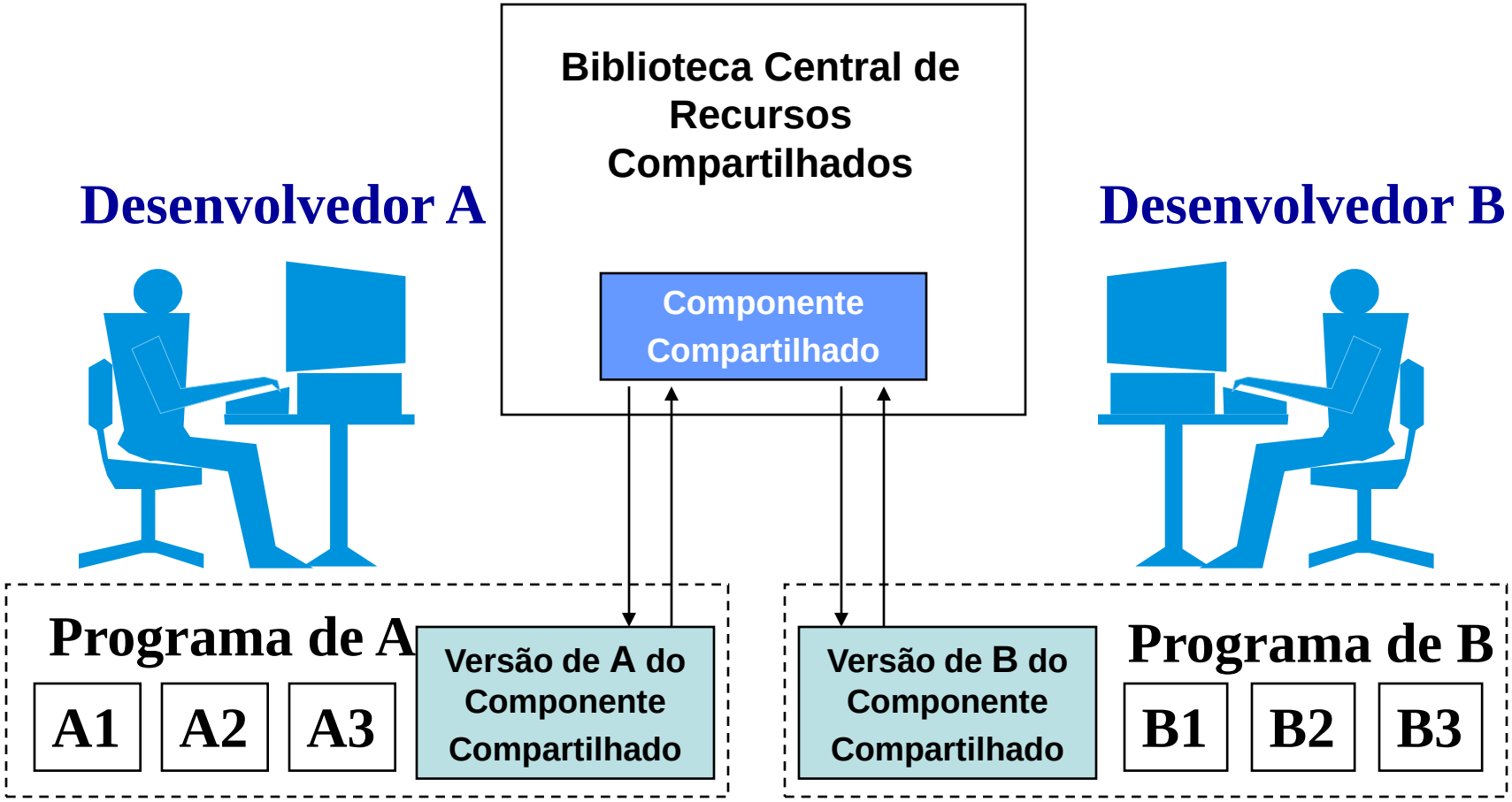
Problema 3: Manutenção Múltipla



Manutenção Múltipla

- Ocorre quando cada desenvolvedor trabalha com uma cópia "local" do que seria o mesmo componente
- Dificuldade para saber:
  - Que funcionalidades foram implementadas em quais versões do componente
  - Que defeitos foram corrigidos
- Evitado através de uma biblioteca central de componentes compartilhados
  - Nesse esquema, cada componente é copiado para a biblioteca sempre que alterado
  - Resolve o Problema da Manutenção Múltipla, mas...

Problema 4: Atualização Simultânea



Problema 4: Atualização Simultânea

- O desenvolvedor A encontra e corrige um defeito em sua versão do componente compartilhado
- Uma vez corrigido, o componente modificado é copiado para a biblioteca central
- O desenvolvedor B encontra e corrige o mesmo defeito em sua versão do componente por não saber que A já tinha feito isso
- O trabalho de A é desperdiçado

Como Resolver?

- **O problema da atualização simultânea não pode ser resolvido simplesmente copiando componentes compartilhados para uma biblioteca central.**

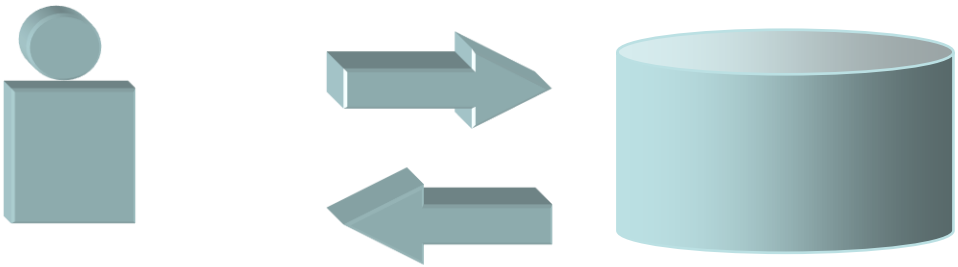
- **Algum mecanismo de controle é necessário para gerenciar a entrada e saída dos componentes.**

Softwares

- Apresenta diversas soluções para realizar o controle:
  - CVS - Livre
  - GIT - Livre
  - SVN - Livre
  - SourceSafe – Microsoft
  - ClearCase – IBM

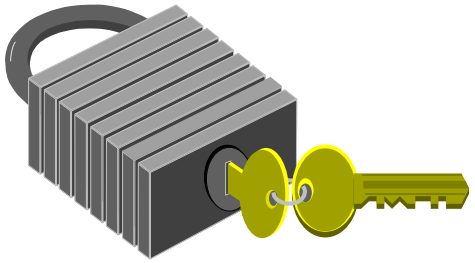
Repositório

- Local (físico e lógico) onde os itens de um sistema são guardados
- Pode conter diversas versões do sistema
- Utiliza mecanismos de controle de acesso

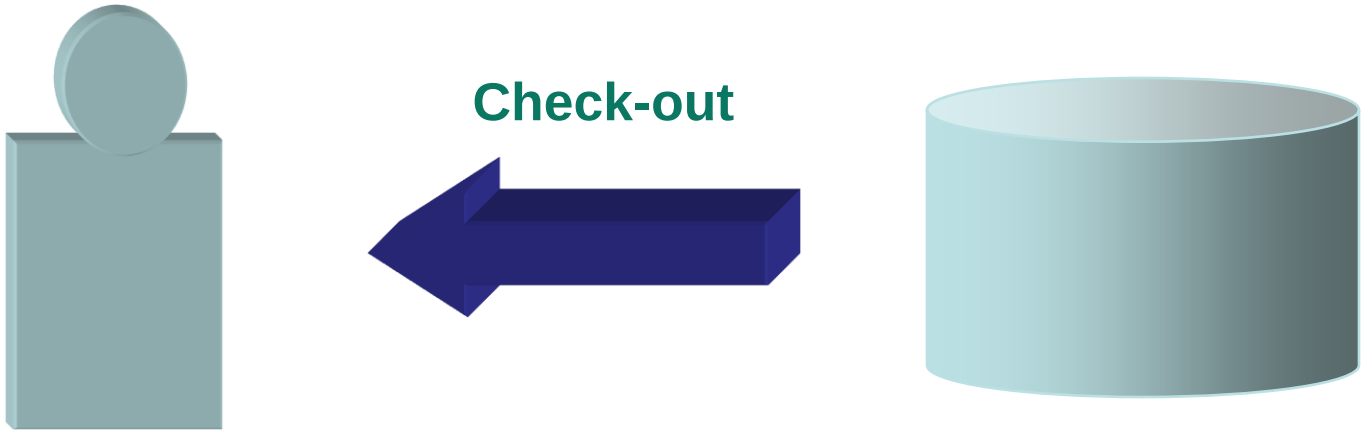


Lock

- Resolve a Atualização Simultânea
- Garante que apenas o usuário que detém o lock pode alterar o arquivo
- Problema: “serializa” o trabalho dos desenvolvedores



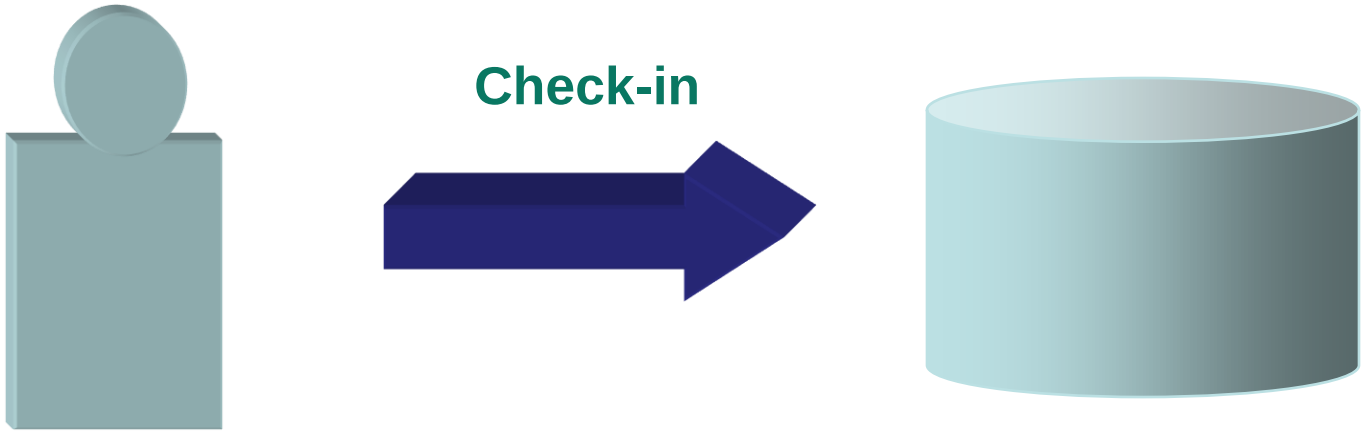
Check-Out



Check-Out (continuação)

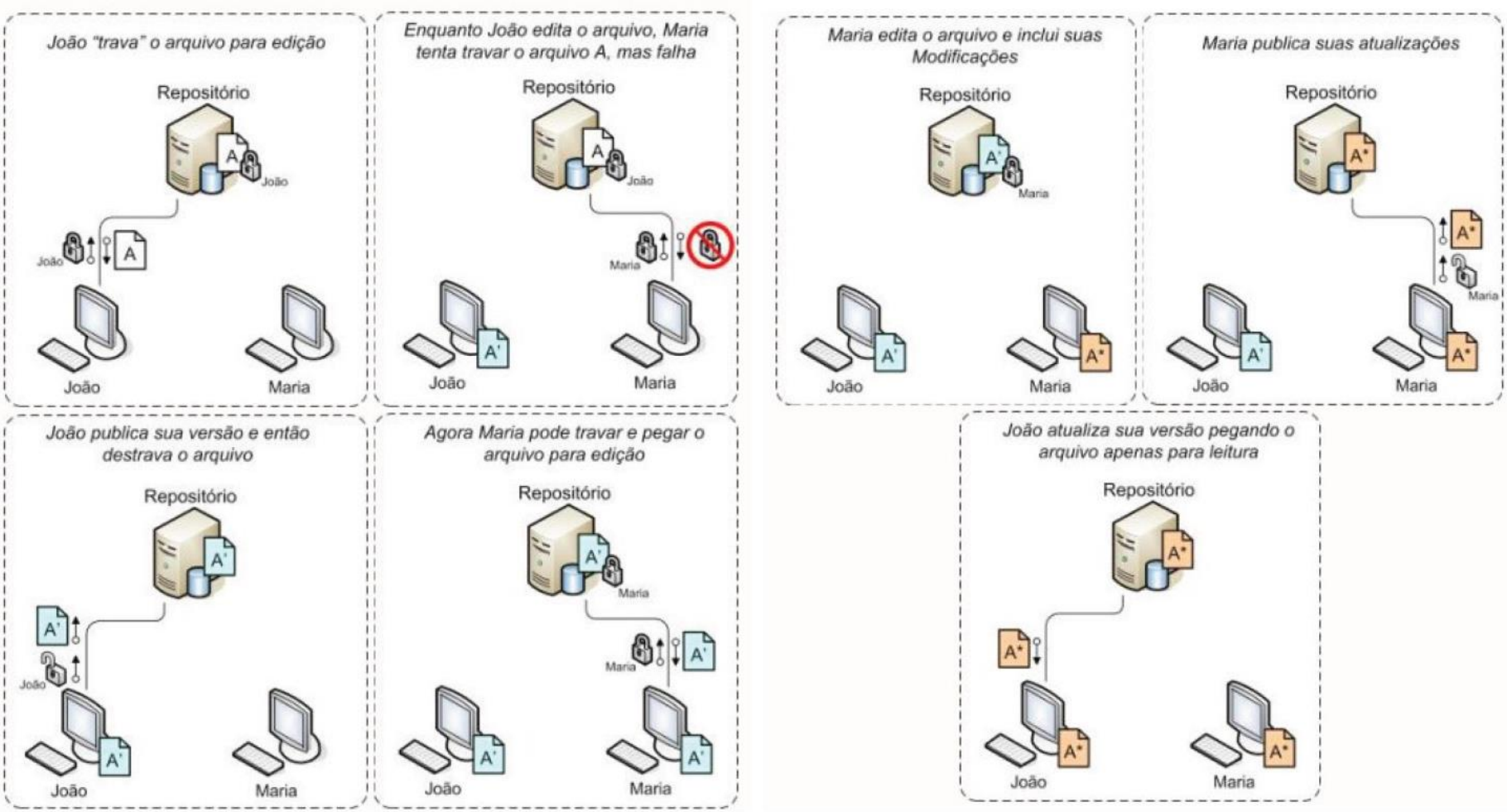
- Recupera a (última) versão de um item de configuração guardada no repositório
  - Escrita
    - Verifica que ninguém detém o lock do item de configuração
    - Obtém o lock do item
    - Cria uma cópia, para edição, no cliente
  - Leitura
    - Verifica que alguém já detém o lock
    - Cria uma cópia, apenas para leitura, no cliente

Check-In

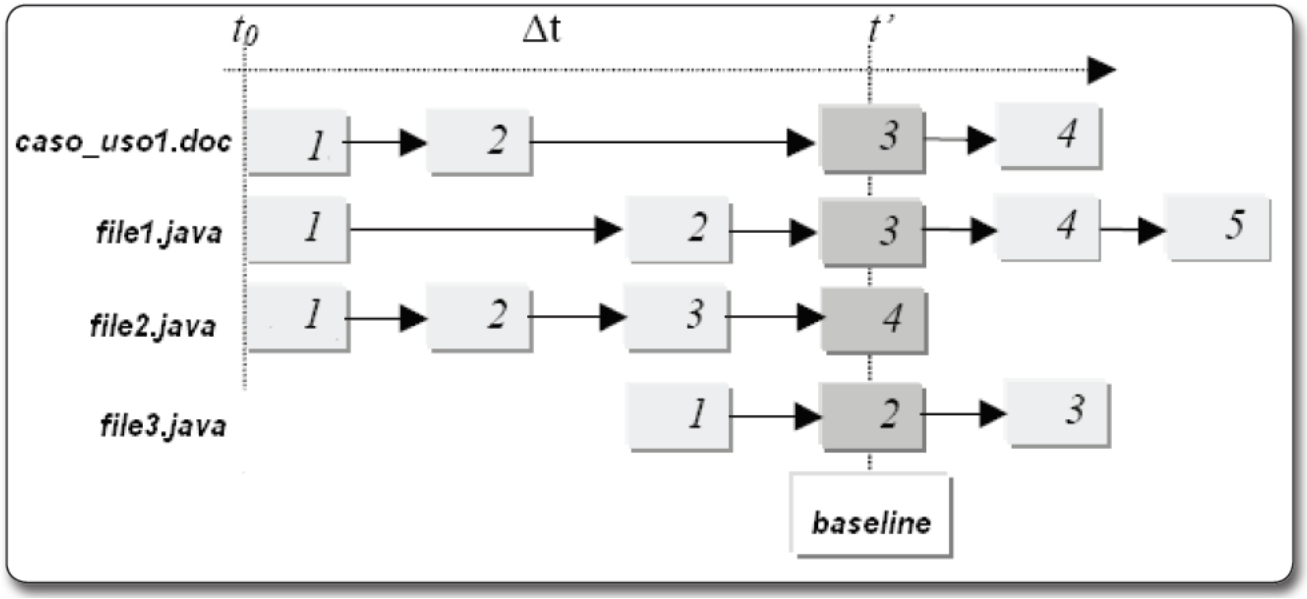


Check-In (continuação)

- Ação de inserir/atualizar um item de configuração no repositório
  - Verifica o lock do item de configuração, caso o mesmo já exista
  - Verifica e incrementa a versão do item
  - Registra informações das mudanças (autor, data, hora, comentários)
  - Inclui/atualiza o item


João "trava" o arquivo para edição
Repositório
A
João
A
João
Maria
Enquanto João edita o arquivo, Maria tenta travar o arquivo A, mas falha
Repositório
A
João
Maria
A'
João
Maria
Maria edita o arquivo e inclui suas Modificações
Repositório
A'
Maria
A'
João
A*
Maria
Maria publica suas atualizações
Repositório
A*
A*
Maria
A'
João
A*
Maria
João publica sua versão e então destrava o arquivo
Repositório
A'
A'
João
A'
João
Maria
Agora Maria pode travar e pegar o arquivo para edição
Repositório
A'
Maria
Maria
A'
A'
João
A'
Maria
João atualiza sua versão pegando o arquivo apenas para leitura
Repositório
A*
A*
A*
João
Maria

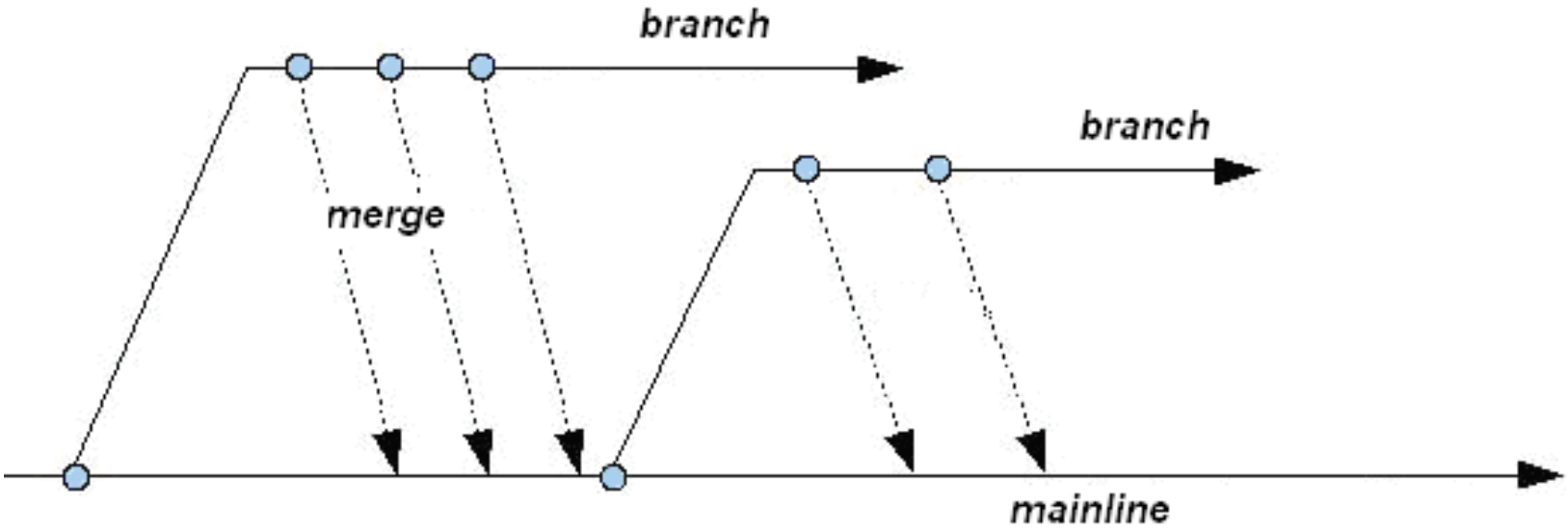
Baseline


t_0
Δt
t'
caso_uso1.doc
file1.java
file2.java
file3.java
1
2
3
4
5
baseline

Branch (ramos)

- Criação de um fluxo alternativo (ramificação) para atualização de versões de itens de configuração
- Recurso muito poderoso
- Devem existir regras bem definidas para criação de branches
  - Por que e quando devem ser criados?
  - Quais os passos?
  - Quando retornar ao fluxo principal?
- Branches normalmente se originam de correções em versões anteriores ou saídas de testes de regressão realizados visando novos produtos.

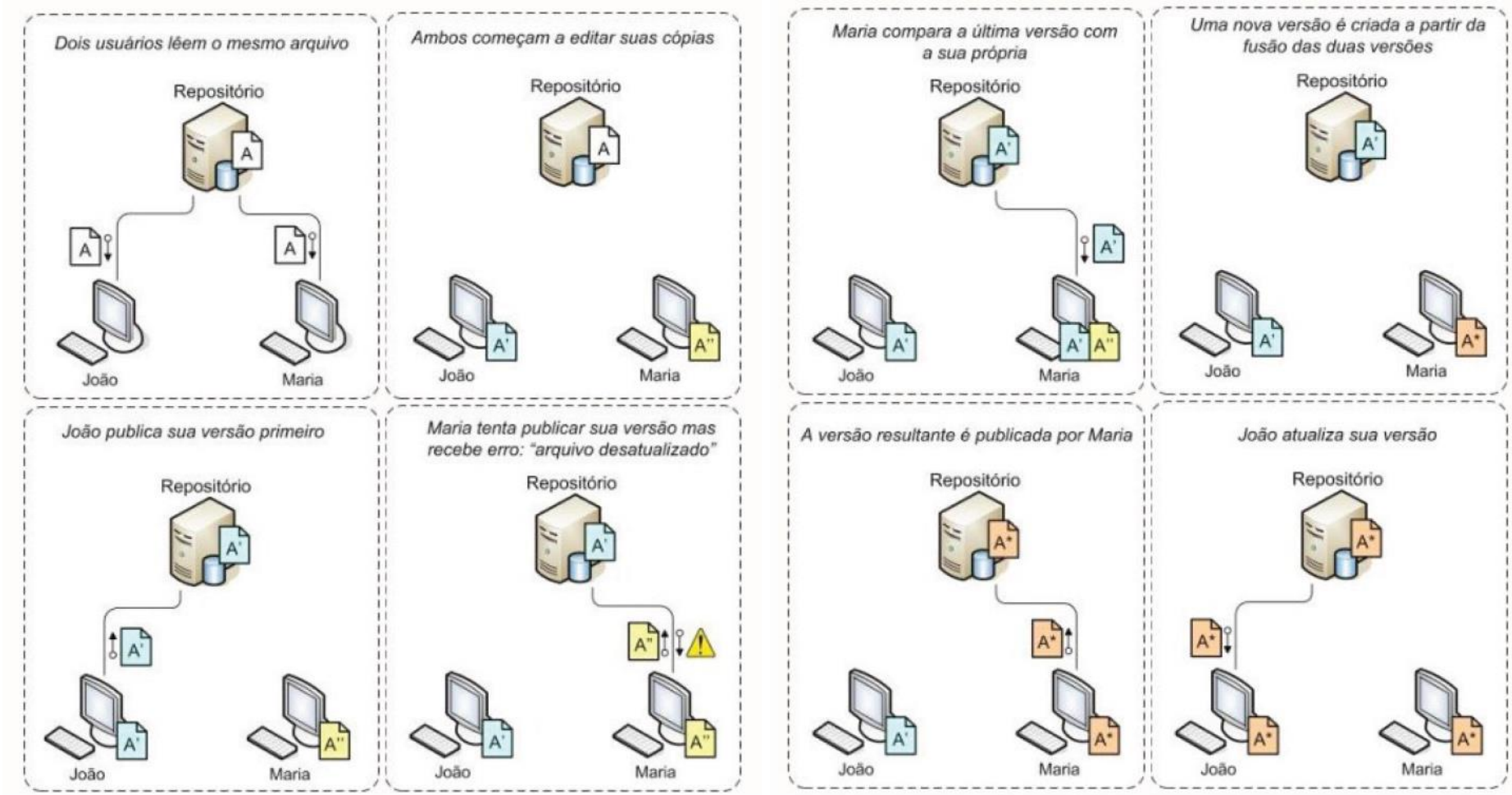
Branch



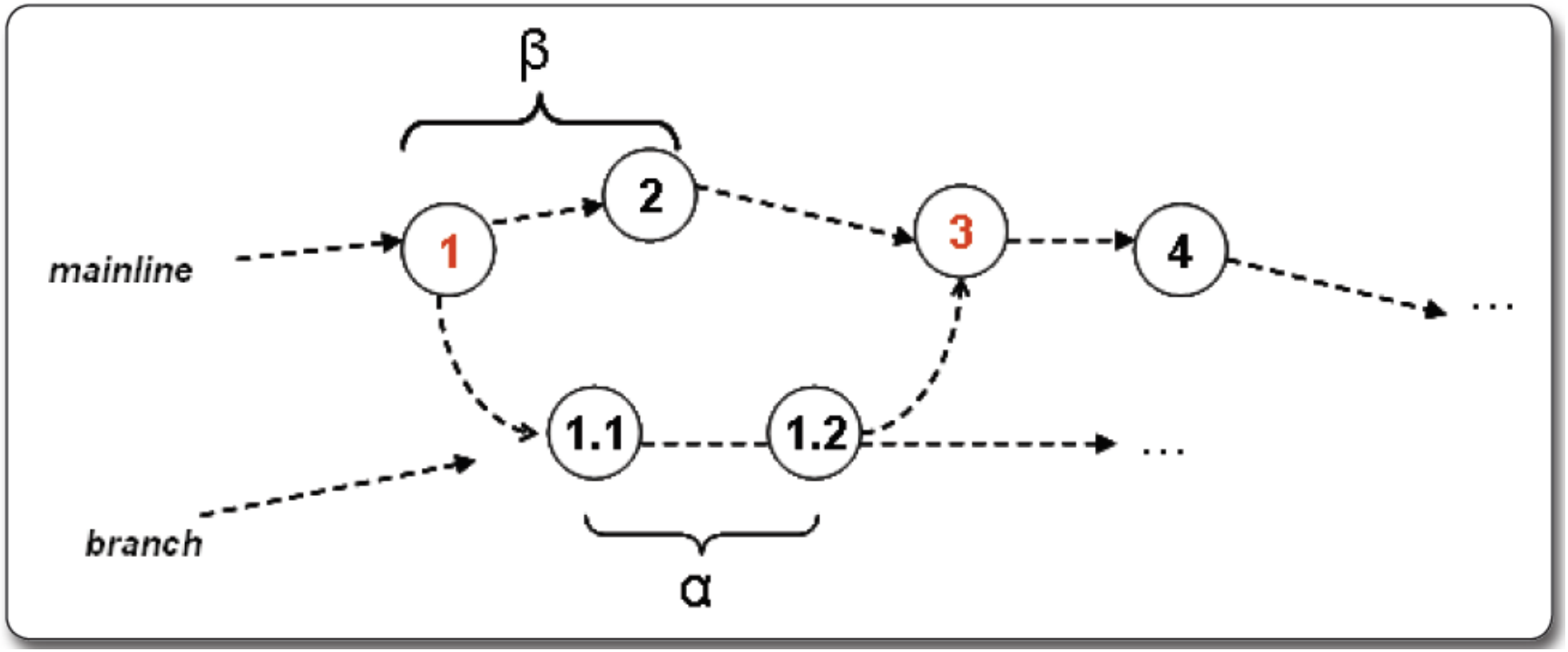
Merge (Junção)

- Unificação de diferentes versões de um mesmo item de configuração
- Integração dos itens de configuração de um **branch** com os itens de configuração do fluxo principal
- **Check-out** atualizando a área local
- Algumas ferramentas fornecem um mecanismo automático para realização de merges
  - Mesmo com o uso de ferramentas, em vários casos há necessidade de intervenção humana.

Merge


Dois usuários lêem o mesmo arquivo
Repositório
A
João
Maria
Ambos começam a editar suas cópias
A'
A''
Maria compara a última versão com a sua própria
Uma nova versão é criada a partir da fusão das duas versões
A*
João publica sua versão primeiro
Maria tenta publicar sua versão mas recebe erro: "arquivo desatualizado"
A versão resultante é publicada por Maria
João atualiza sua versão

Merge



Softwares

- Apresenta diversas soluções para realizar o controle:
  - CVS - Livre
  - GIT - Livre
  - SVN – Livre [https://tortoisesvn.net/downloads.html]
  - SourceSafe – Microsoft
  - ClearCase – IBM