Gestão de configurações

Metodologias e Desenvolvimento de Software

Pedro Salgueiro
pds@uevora.pt
CLV-256

Gestão de configurações

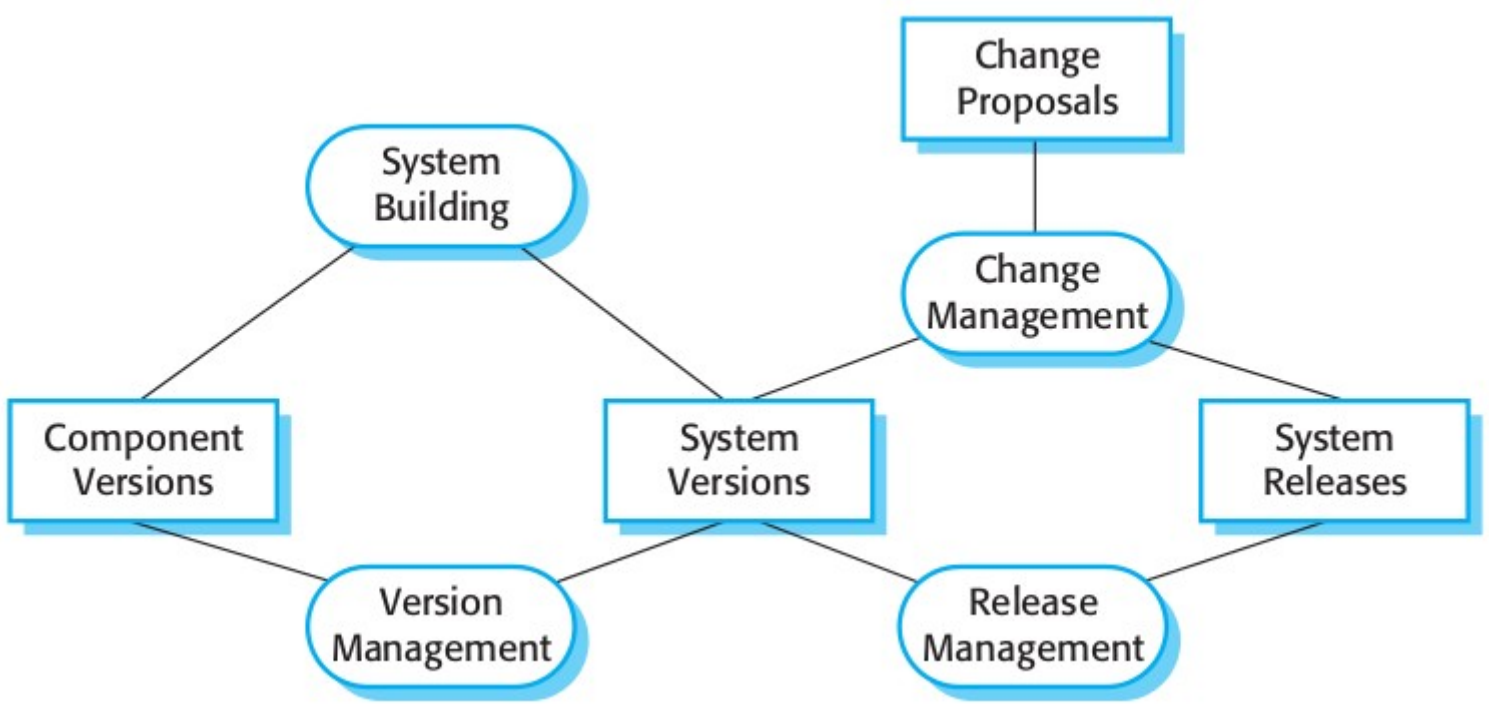
- Software muda frequentemente.
  - Sistemas podem ser vistos como uma série de versões
  - Cada versão tem de ser mantida e gerida
- Cada versão
  - Implementa pedidos de alterações
  - Correções de falhas
  - Adaptações diversas
- Gestão de configurações
  - “Políticas”, ferramentas e processos para gerir alterações/mudanças
  - Essencial
    - Muito fácil de perder o rasto das alterações e de quais as alterações que devem fazer parte de uma versão do sistema

Gestão de configurações

Atividades

- Gestão de alterações
  - Manter o “rasto” dos pedidos de alterações feitas pelos clientes e programadores, quais os custos dos impactos destas alterações e decidir quais as alterações que devem ser implementadas.
- Gestão de versões
  - Manter o rasto das várias versões dos componentes do sistema, garantindo que as alterações dos diferentes programadores não interfere com outras.
- Criação do sistema (system building)
  - Processo de construir componentes do sistema, dados e bibliotecas, para depois compilar tudo num sistema executável
- Gestão de *releases*
  - Preparar o software para ser entregue e manter registo de todas as versões entregues (ao cliente).

Atividades


Change Proposals
System Building
Change Management
Component Versions
System Versions
System Releases
Version Management
Release Management

Processos Ágeis e Gestão de Configurações

- Essencial nos processos ágeis
  - Sistemas são alterados várias vezes por dia
  - Várias versões por dia
- Versões dos componentes
  - Repositório/projeto partilhado com toda a equipa de desenvolvimento
- Processo de desenvolvimento
  - Código é copiado do repositório partilhado para a área de trabalho
  - Modificação do código para criar nova versão/sistema
  - Nova versão é testada localmente
  - Código com da versão é enviada para o repositório partilhado

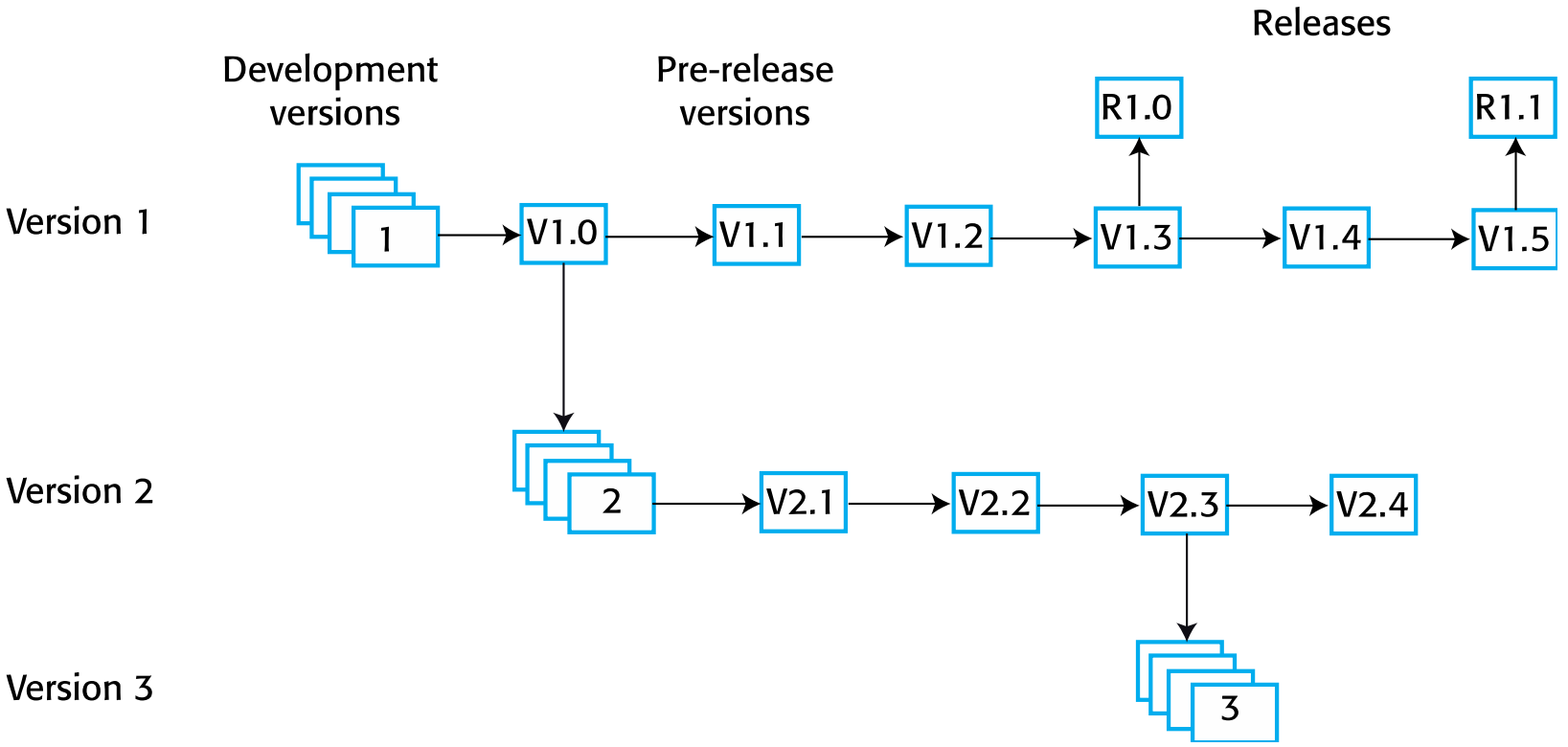
Etapas de desenvolvimento

- Desenvolvimento
  - Novas funcionalidades são adicionadas
  - Gestão de configurações
- Testes
  - Versão do sistema é entregue internamente para testes
  - Não são adicionadas novas funcionalidades
  - Correção de bugs, fixes, melhorias de desempenho, correção de vulnerabilidades de segurança
- Release
  - Nova versão é entregue ao cliente para utilização
  - Podem surgir novas versões da Release para corrigir erros, vulnerabilidades ou adicionar novas funcionalidades sugeridas pelos clientes

Sistemas multi-versão

- Sistemas grandes
- Não existe apenas uma versão funcional
- Existem várias versões do sistema
  - Em diferentes etapas de desenvolvimento
- Podem existir varias equipas de desenvolvimento
  - Diferentes versões do sistema

Sistemas multi-versão – processo de desenvolvimento



Terminologia

Term	Explanation
Baseline	A baseline is a collection of component versions that make up a system. Baselines are controlled, which means that the versions of the components making up the system cannot be changed. This means that it is always possible to recreate a baseline from its constituent components.
Branching	The creation of a new codeline from a version in an existing codeline. The new codeline and the existing codeline may then develop independently.
Codeline	A codeline is a set of versions of a software component and other configuration items on which that component depends.
Configuration (version) control	The process of ensuring that versions of systems and components are recorded and maintained so that changes are managed and all versions of components are identified and stored for the lifetime of the system.
Configuration item or software configuration item (SCI)	Anything associated with a software project (design, code, test data, document, etc.) that has been placed under configuration control. There are often different versions of a configuration item. Configuration items have a unique name.
Mainline	A sequence of baselines representing different versions of a system.

Terminologia

Term	Explanation
Merging	The creation of a new version of a software component by merging separate versions in different codelines. These codelines may have been created by a previous branch of one of the codelines involved.
Release	A version of a system that has been released to customers (or other users in an organization) for use.
Repository	A shared database of versions of software components and meta-information about changes to these components.
System building	The creation of an executable system version by compiling and linking the appropriate versions of the components and libraries making up the system.
Version	An instance of a configuration item that differs, in some way, from other instances of that item. Versions always have a unique identifier.
Workspace	A private work area where software can be modified without affecting other developers who may be using or modifying that software.

Gestão de versões

- Gestão de versões é o processo de manter o “rasto” das diferentes versões dos componentes de software, dos itens de configuração de e do sistema.
- Deve também garantir que alterações feitas por diferentes programadores não interferem com o trabalho de outros programadores.
- Pode ser visto como um processo usado para gerir *codelines* e *baselines*

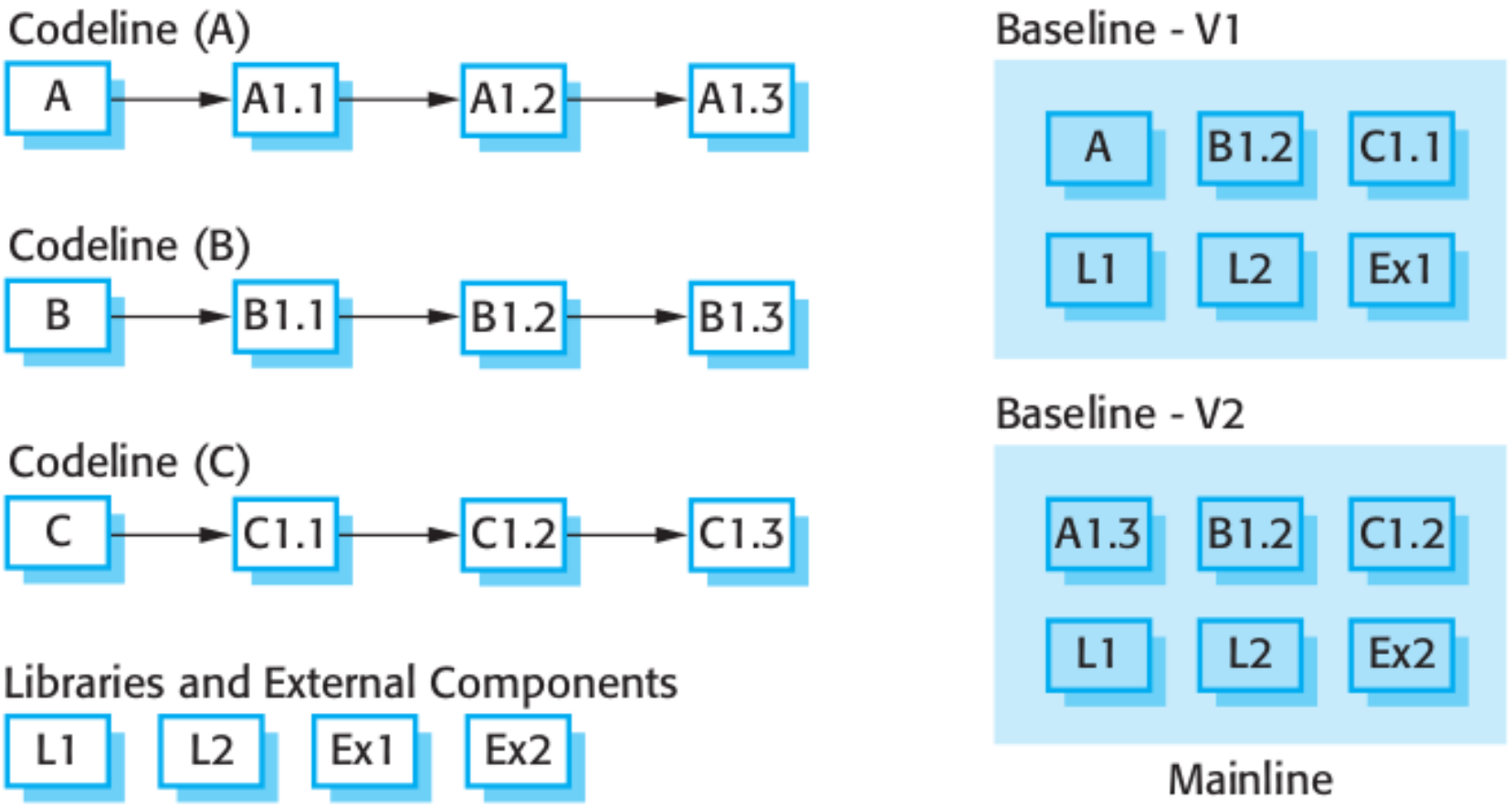
Codelines e baselines

- Um *codeline* é uma sequência de versões de código, onde as versões mais recentes são derivadas de versões mais antigas
- *Codelines* aplicam-se normalmente aos componentes do sistema
  - Dá origem a diferentes versões de cada componente
- Uma *baseline* é uma definição de um sistema especifico
- A *baseline* especifica as versões dos componentes que são incluídos no sistema, junto com a especificação das bibliotecas usadas, ficheiros de configurações, etc.

Baselines

- Especificação de *Baselines*
  - Recorrendo a linguagens/ferramentas específicas
  - Definir quais os componentes que fazem parte de uma versão do sistema
- Importância das *Baselines*
  - É necessário reconstruir uma versão específica do sistema
    - É necessário saber quais as versões exactas dos componente que compõem a versão do sistema

Codelines e baselines



Sistemas de gestão de versões *(VCS)*

- Identificação de versões e *releases*
  - Cada versão tem um identificador único

- Gestão de armazenamento
  - Por forma a reduzir o espaço em armazenamento necessário para gerir todas as versões dos componentes, que podem variar muito pouco entre si, os sistemas de gestão de versões normalmente usam métodos que *agilizam* este armazenamento

- Registo de todas as alterações
  - Todas alterações feitas ao código devem ser registadas.

Gestão de configurações

Sistemas de gestão de versões (VCS)

- Identificar, armazenar e controlar acessos
  - Diferentes versões do sistema
- Dois tipos de VCS
  - Centralizados:
    - repositório central onde são geridas todas as versões de todos os componentes
    - Subversion, a.k.a.: SVN
  - Distribuídos:
    - repositório distribuído com todos os componentes
    - várias “cópias” do repositório
    - Git

Gestão de configurações

Sistemas de gestão de versões – características essenciais

- Identificação de versões e *releases*
- Gestão do histórico de alterações
- Permitir o desenvolvimento individual
- Permitir o desenvolvimento de vários projetos ao mesmo tempo
- Mecanismos eficientes do armazenamento das versões

Gestão de configurações

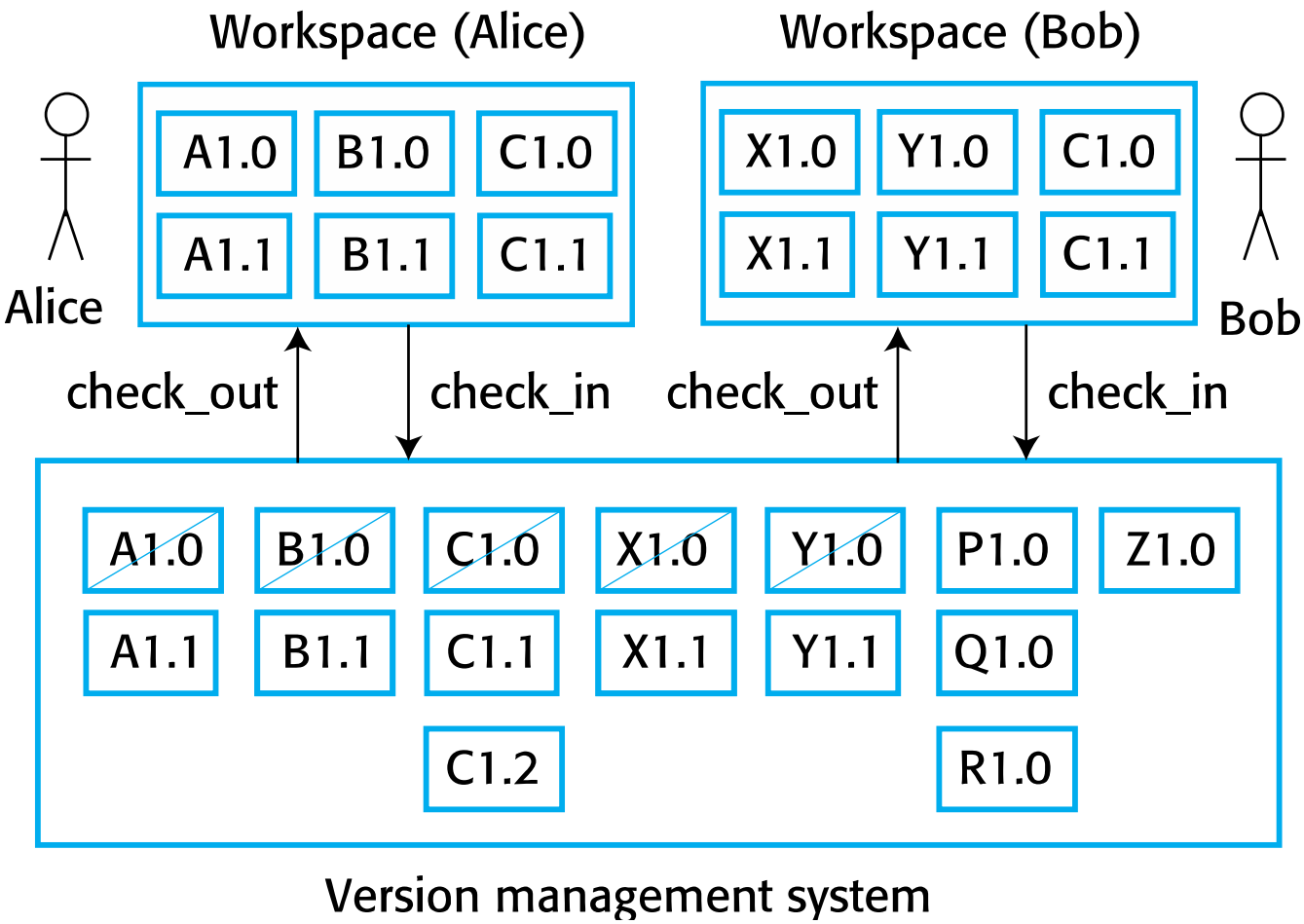
Repositórios publicos e privados

- Desenvolvimento independente e sem interferência
  - Repositório do projeto
  - Repositório privado
- Repositório do projeto
  - Versão “master” de todos os componentes
  - Usado para criar as *baselines* durante o *system building*
- Modificação de componentes
  - *Developers* copiam componentes do repositório para a sua área de trabalho *(check-out)*
  - Trabalham sobre essa cópia
- Depois das alterações estarem concluídas
  - *Developers* copiam componentes da sua área de trabalho para o do repositório *(check-in)*

Controle de versões centralizado

- Programadores copiam componentes ou pastas (check-out)
  - Do repositório do projeto
  - Para o seu ambiente de trabalho
  - Trabalham sobre essas cópias
- Depois de terminarem as alterações
  - Copiam os componentes para o repositório do projeto (check-in)
- Vários programadores a trabalhar no mesmo componente
  - Cada pessoa copia o componente do repositório (check-out)
  - Se o componente foi previamente copiado
    - O VCS alerta que o componente foi copiado (check-out) por outro programador

Check-in e Check-out

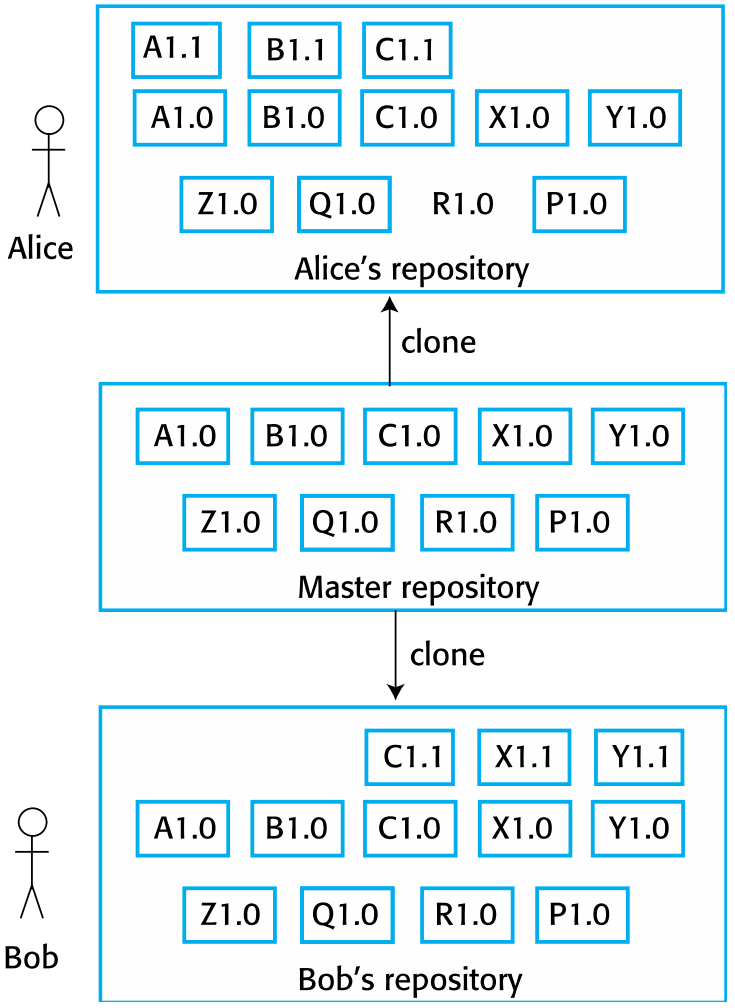


Gestão de configurações

Controle de versões distribuído

- Repositório “master”
  - Criado num servidor
  - Mantém o código criado pela equipa de desenvolvimento
- Em vez de copiar apenas os ficheiros necessários (check-out)
  - Criado um clone do repositório do projeto
  - Copiado e instalado na sua área de trabalho
- Programadores sobre o clone do repositório do projeto
  - Repositório privado
- Quando as alterações estiverem prontas
  - Fazem “commit” das alterações
  - Update do seu repositório privado
  - Enviam as alterações para o repositório do projeto (push)

Clonagem do repositório



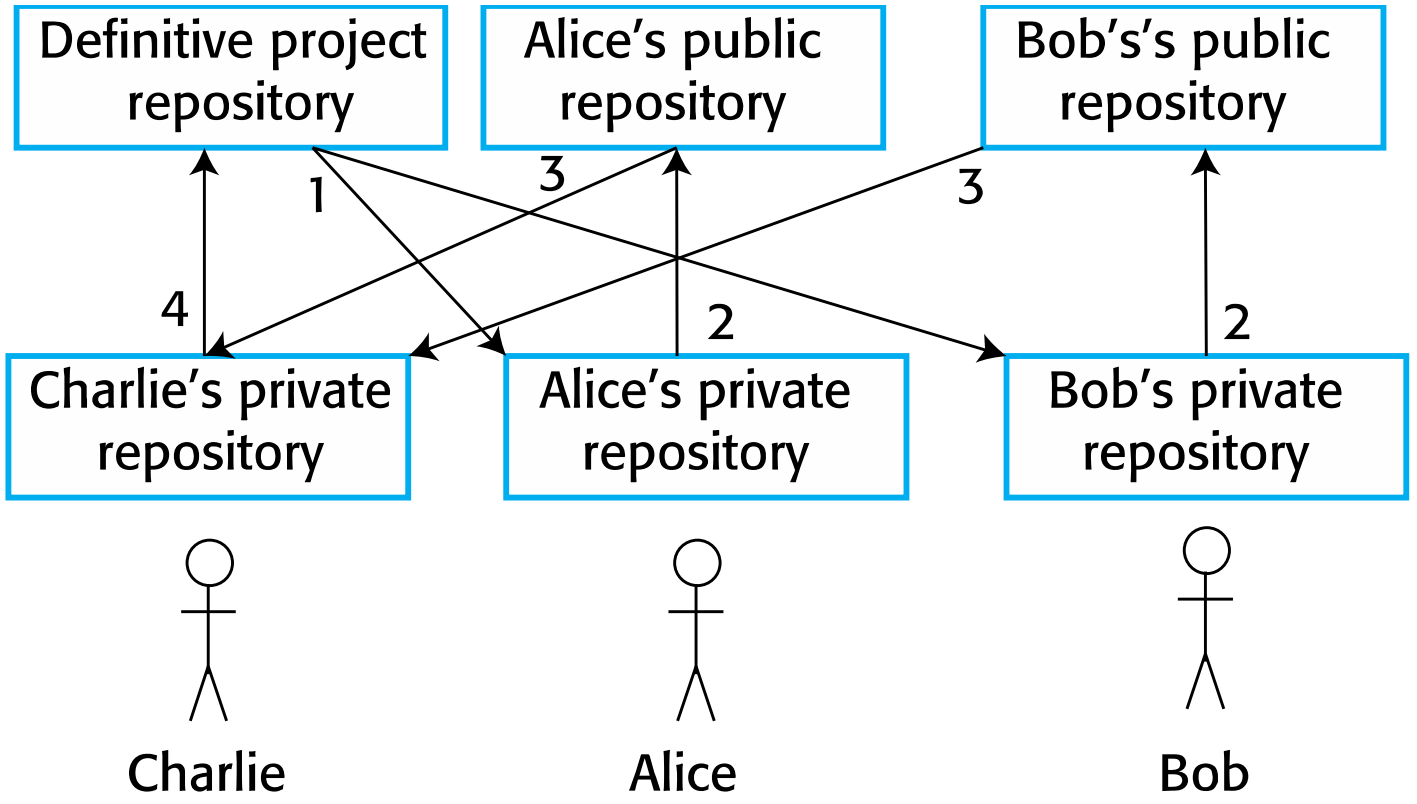
Gestão de versões distribuída – vantagens

- Mecanismo de backup do repositório
  - Se o repositório ficar corrompido, o trabalho pode continuar e reposto através das cópias locais
- Permite trabalhar em modo “off-line”
  - Programadores podem fazer commit de alterações em qualquer altura, mesmo quando “off-line”
- Programadores podem compilar e testar o sistema todo localmente
  - Testar alterações feitas

Desenvolvimento open source

- Gestão de versões distribuída
  - Essencial para desenvolvimento open source
  - Várias pessoas podem estar a trabalhar ao mesmo tempo no mesmo componente sem coordenação central
- Além do repositório privado
  - Programadores também fazem a gestão de um repositório público
  - Novas versões são enviadas para o seu repositório publico (push)
  - Gestor do sistema decide se/quando inclui as alterações no repositório “master” do sistema

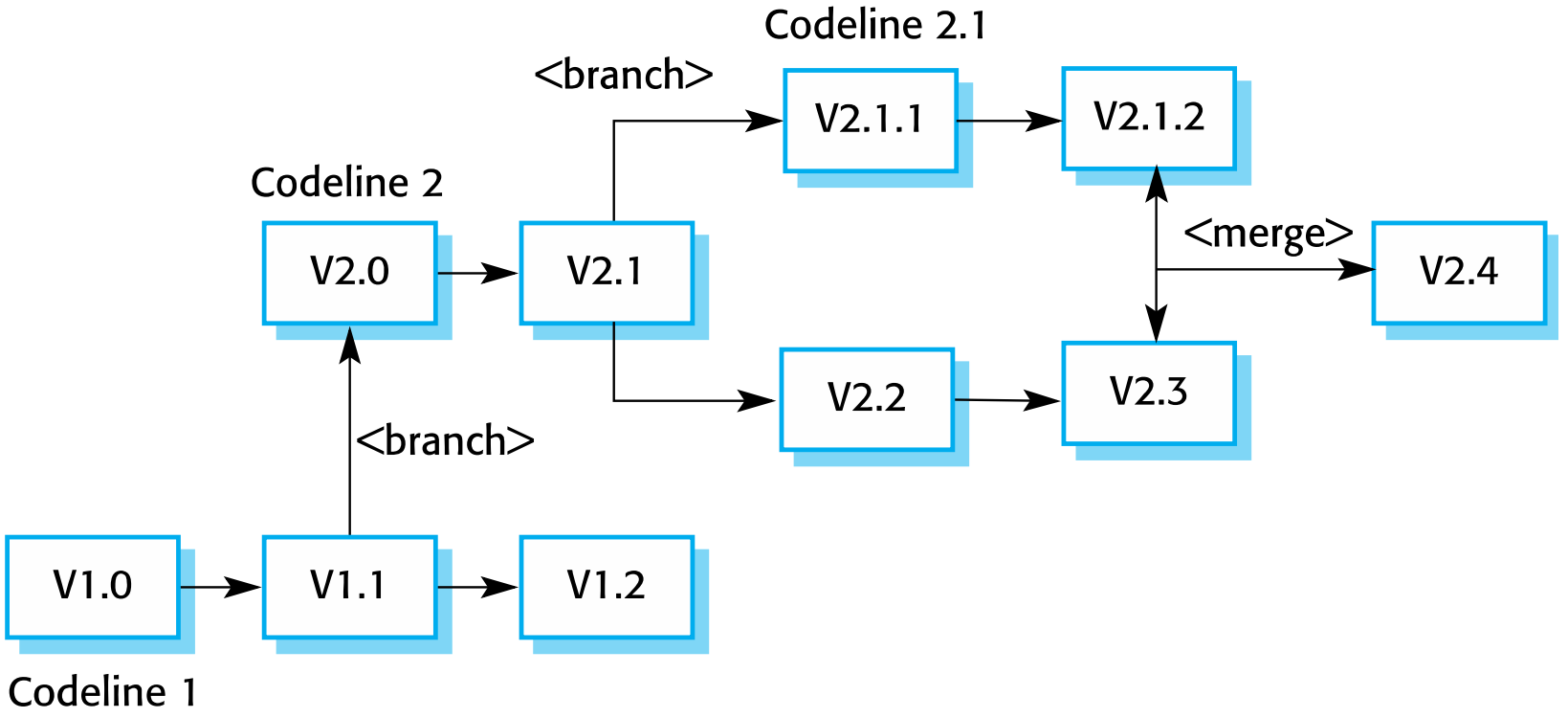
Desenvolvimento open source



Branching e Merging

- Em vez de uma sequência linear de versões que refletem as alterações a um componente ao longo do tempo
  - Podem existir várias sequências independentes
  - Muito comum
  - Vários programadores trabalham de forma independente em diferentes versões do sistema
  - Sistema é modificado de várias formas
- Obriga à junção de vários “codeline branches” (merging)
  - Por forma a criar uma nova versão do componente que inclua todas as alterações feitas
  - Se as alterações feitas envolverem diferentes partes de código, as diferenças podem ser “*merged*” de forma automática, combinado os vários *“deltas”*

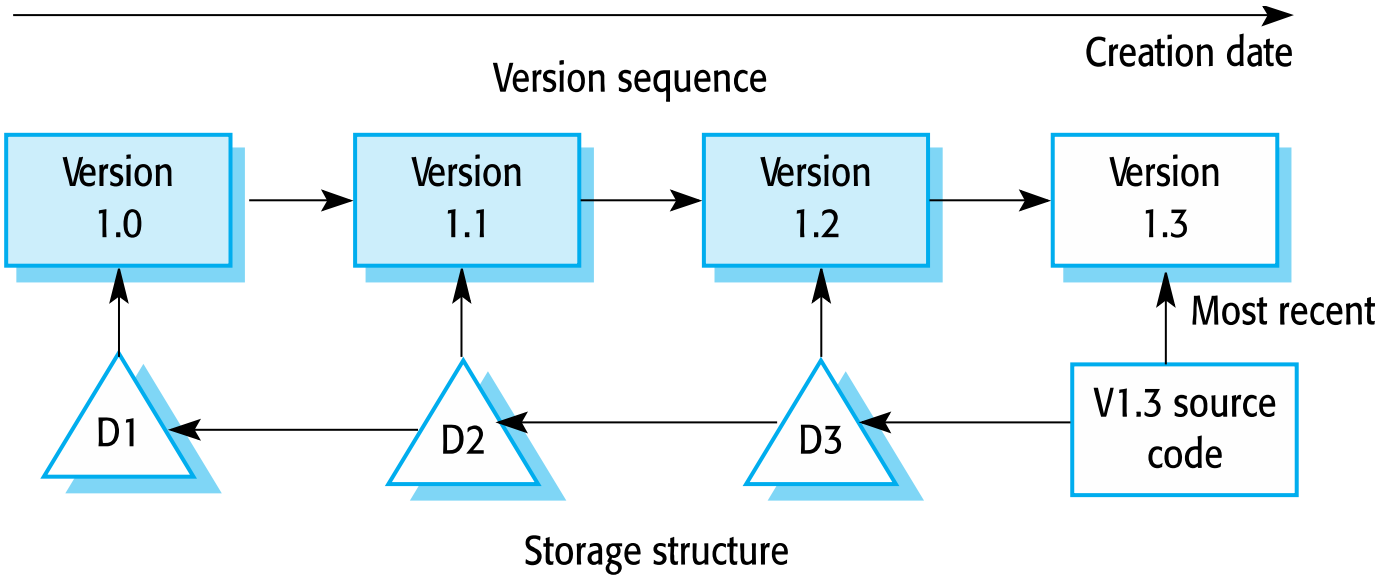
Branching e Merging



Gestão de armazenamento

- Primeiros sistemas de controle de versões
  - O armazenamento era uma das suas principais funções
- Espaço em disco era caro
  - Era importante minimizar o espaço em disco usado pelas várias cópias dos componentes
- Em vez de guardar uma cópia completa de cada versão
  - O sistema guarda a lista de diferenças (deltas) entre cada versão
  - Aplicando deltas correspondentes a uma versão, consegue-se recriar essa versão

Gestão de armazenamento usando deltas



Gestão de armazenamento no Git

- Hoje em dia o armazenamento em disco é barato
  - Git não usa deltas
  - Usa uma abordagem mais rápida
- Usa algoritmos de compressão
  - Ficheiros
  - Meta-dados
- Não armazena ficheiros duplicados
  - Aceder a um ficheiro implica apenas a sua descompressão
  - Não é necessário aplicar uma cadeia de operações (deltas)
- Conceito *packfiles*
  - Vários ficheiros pequenos são combinados num único ficheiro indexado
  - Reduz o problema associado a ter muitos ficheiros pequenos

Construção do sistema - System building

- Processo de criar uma versão completa e executável do sistema, compilando e *linkando* os vários componentes do sistema, bibliotecas externas, ficheiros de configuração, etc…

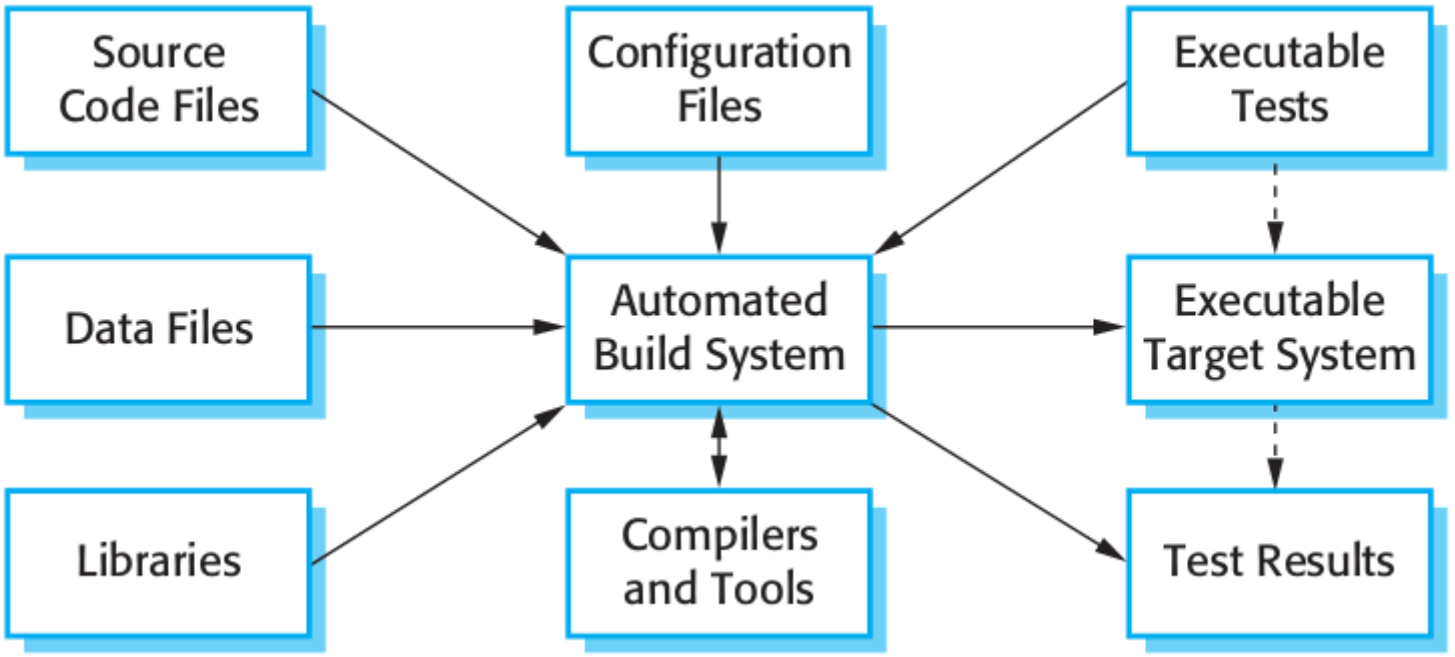
- Ferramentas de construção do sistema e ferramentas de gestão de versões devem comunicar entre si.
  - Fazer *check-out* de versões de componentes por fazer a construção do sistema

- Descrição da *baseline* deve ser feita na ferramenta de construção do sistema.

Plataformas de construção

- Plataformas de desenvolvimento
  - Incluem ferramentas de desenvolvimento
    - Compiladores, editores, etc
  - Programadores fazem checkout do código a partir do sistema de gestão de versões para um *workspace* privado antes de fazerem alterações ao sistema
- Build server
  - Usado para criar versões executáveis (releases) do sistema
    - Programadores fazem check-in do código para o sistema gestão de versões antes do *build* ser feito.
- Ambiente de produção
  - Plataforma onde o sistema é colocado em uso.

Construção do sistema (system building)



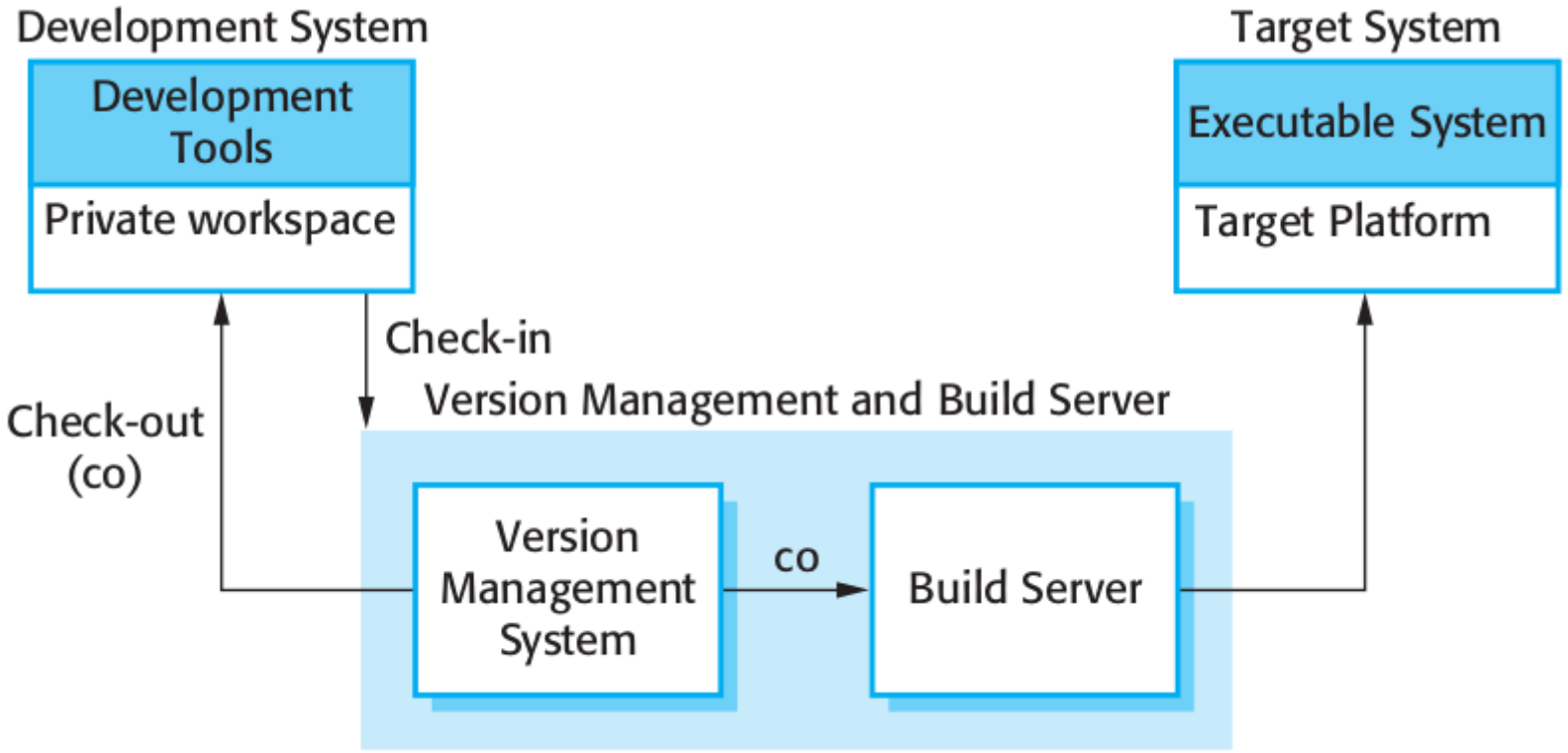
Build system

- Funcionalidades
  - Script para gerar o *build*
  - Integração com o sistema de gestão de versões
  - Recompilação mínima
  - Criação de um sistema executável
  - Automação dos testes
  - Geração de documentação e relatórios

Construção de sistemas - Plataformas

- Plataforma de desenvolvimento
  - Ferramentas de desenvolvimento (compiladores, IDEs, etc…)
  - Programadores fazem *check-out* do código do VCS para a sua área de trabalho antes de fazerem as alterações
  - Programadores podem querer compilar/construir e testar todo o código antes de fazerem *commit* das alterações
- Build server
  - Usado para criar versões “finais” do sistema
  - Mantém todas as versões do sistema
  - Programadores fazem *check-in* do código para o VCS
  - Build Server faz *check-out* do código para construir o sistema
- *Target environment*
  - Plataforma onde o sistema é executado

Plataformas de construção



System build – Métodos ágeis

- Check-out da *mainline* a partir do sistema de gestão de versões para o ambiente privado de desenvolvimento do programador.

- Construir o sistema e correr todos os testes de forma automática para ter a certeza de que todos os testes passam com sucesso. Se não, a *build* tem problemas e deve informar-se o responsável pela ultima baseline (responsável por corrigir o problema).

- Fazer as alterações aos componentes do sistema.

- Construir o sistema no ambiente de desenvolvimento e correr todos os testes. Se os testes falharem, continuar o trabalho até os testes correm com sucesso.

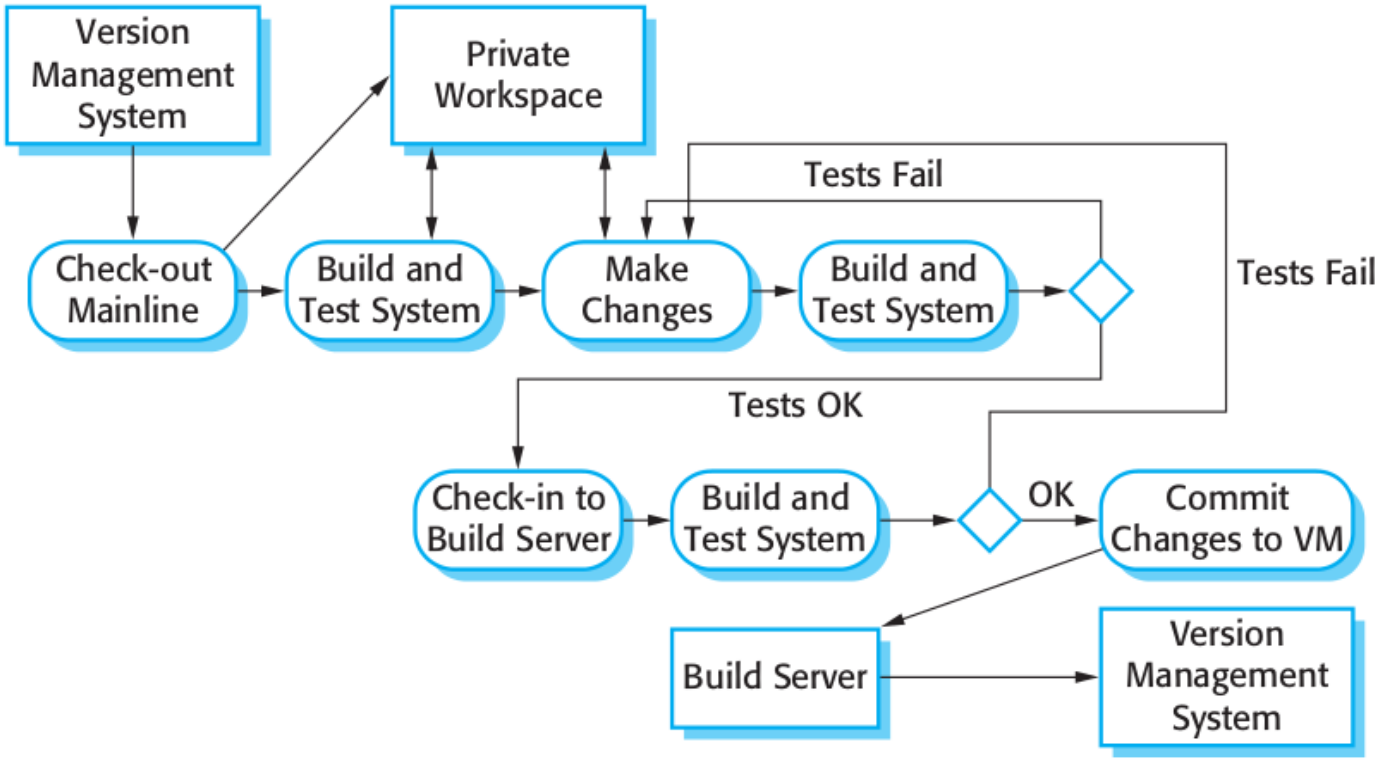
System build – Métodos ágeis

- Depois de todos os testes passarem com sucesso, fazer *check in* no sistema de gestão de versões.

- Construir o sistema no *build server* e correr todos os testes.
  - Necessário se houve outros programadores que fizeram alterações desde o *check out* inicial
  - Se houver testes que falham, é necessário corrigi-los no ambiente de desenvolvimento pessoal do programador.

- Se os testes passarem no *build server*, fazer *commit* das alterações como uma nova *baseline* na *mainline* do sistema.

Integração contínua



Integração contínua – vantagens e desvantagens

- Vantagens
  - Permite a descoberta e a correção de erros causados pela interação entre vários programadores, o mais cedo possível
  - O sistema mais recente na *mainline* é a versão actual do sistema
- Desvantagens
  - Se o sistema for muito grande, a construção e teste do sistema pode demorar muito tempo
  - Se a plataforma de desenvolvimento for diferente da plataforma de execução, pode não ser possível executar os testes na área de trabalho do programador

Bibliografia

- Software Engineering. Ian Sommerville. 10th Edition. Addison-Wesley. 2016. Capítulo 25.