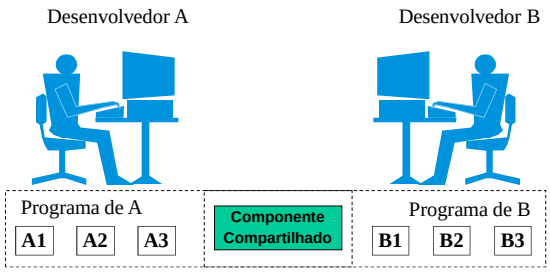
Gerenciamento de Configuração

Motivação: Problema dos Dados Compartilhados - Cenário



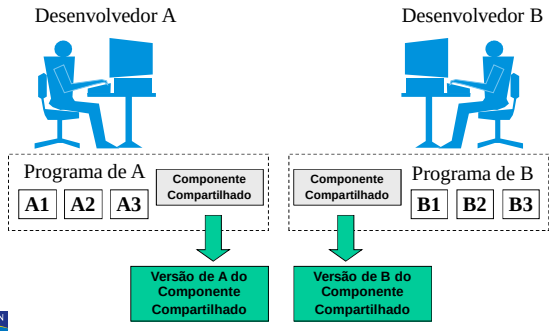
Motivação: Problema dos Dados Compartilhados - Cenário

- O desenvolvedor A modifica o componente compartilhado
- Mais tarde, o desenvolvedor B realiza algumas alterações no mesmo componente
- 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

Motivação: 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 (Manutenção Múltipla)

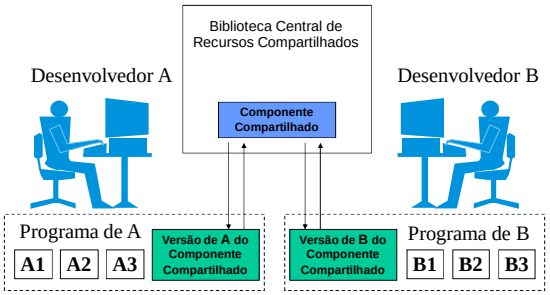
Motivação: Problema da Manutenção Múltipla



Motivação: Problema da Manutenção Múltipla (continuação)

- 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...

Motivação: Problema da Atualização Simultânea



Motivação: Problema da Atualização Simultânea – Cenário 1

- 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

Motivação: Problema da Atualização Simultânea – Cenário 2

- 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 um outro defeito em sua versão do componente, sem saber do defeito corrigido por A
- O desenvolvedor B copia sua versão do componente para a biblioteca central
- Além de o trabalho de A ser desperdiçado, a versão do componente que se encontra na biblioteca central continua apresentando um defeito
- O desenvolvedor A julga o problema como resolvido

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

O que é Gerência de Configuração?

- O gerenciamento de configuração exerce controle sobre os artefatos produzidos pelo desenvolvimento de software:
  - Mudança em um sistema é uma atividade realizada em equipe;
  - O CM tem por objetivo controlar os custos e o esforço envolvidos na realização das mudanças em um sistema.
- É o processo de identificar, organizar e controlar modificações ao software sendo construído
- A ideia é maximizar a produtividade minimizando os enganos

Gerenciamento de configuração

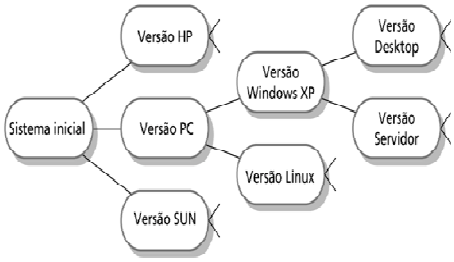
- Envolve o desenvolvimento e a aplicação de procedimentos e padrões para gerenciar um produto de software em evolução.
- O Controle de Mudanças (CM) pode ser visto como parte de um processo mais geral de gerenciamento do projeto.
- Artefatos que estão sob gerenciamento de configuração são chamados de itens de configuração

Gerenciamento de configuração

- Novas versões de sistemas de software são criadas quando eles:
  - Mudam para máquinas/OS diferentes;
  - Oferecem novas funcionalidades;
  - São configurados para requisitos de usuários particulares.

Gerenciamento de configuração: Famílias de sistemas

Figura 29.1
Famílias de sistemas.



Planejamento de gerenciamento de configuração

- Todos os produtos do processo de software podem ser gerenciados:
  - Especificações;
  - Projetos;
  - Programas;
  - Dados de teste;
  - Manuais de usuário.
- Milhares de artefatos separados podem ser gerados para um sistema grande e complexo de software.
- É necessário definir quais estão sujeitos ao gerenciamento de configuração

Principais Atividades do Gerenciamento de Configuração

- Controle de Versões e Releases
- Gerenciamento e Registro de Mudanças
- Organização e Geração dos *Builds do Sistema*

Controle de versões e releases

- Elaborar um esquema de identificação para versões de sistema.
- Planejar quando uma nova versão de sistema será produzida.
- Assegurar que procedimentos e ferramentas de gerenciamento das versões sejam adequadamente aplicados.
- Planejar e distribuir *releases* da nova versão do sistema.

Versões/variantes/releases

- Versão é uma instância de um sistema que é funcionalmente distinta, de alguma maneira, de outras instâncias de um sistema.
- Variante é uma versão de um sistema que tem apenas pequenas diferenças com relação a outras instâncias (normalmente devido a diferenças no hardware/software alvo)
  - Ex.: O Office para MacOS é uma variante do Office para Windows
- Release é uma instância de um sistema distribuída para os usuários fora da equipe de desenvolvimento.
  - Ex. Office 2007

Identificação de versões

- Os procedimentos para identificação de versões devem definir uma maneira não-ambígua de identificar versões
- Existem duas técnicas básicas para identificação de componentes:
  - Numeração de versões;
  - Identificação baseada em atributos;

Numeração de versões

- É um esquema simples de numeração que usa uma derivação linear
  V1, V1.1, V1.2, V2.1, V2.2 etc.
- A estrutura de derivação real é uma árvore ou uma rede, e não uma seqüência.

Um Exemplo: Números de Versões no Linux

A.B.C[.D]

A – versão do *kernel* (apenas duas mudanças: em 1994 e em 1996)

B – revisão importante do *kernel*

C – mudanças menores: novos *drivers* e novas funcionalidades individuais

D – atualizações de segurança e correções de *bugs*

Exemplo de versão: 2.6.27.1

Identificação baseada em atributos

- Os atributos podem ser associados a uma versão com a combinação de atributos que a identificam.
  - Exemplos de atributos são Data, Criador, Linguagem de Programação, Cliente, Status, etc.
- É mais flexível do que um esquema explícito de atribuição de nomes para recuperação de versões;
- Na prática, uma versão também necessita de um nome que facilite a referência

Consultas baseadas em atributos

- Identificação baseada em atributos pode apoiar consultas
- A consulta seleciona uma versão dependendo dos valores de atributos
  - **Ex.: (linguagem = Java, plataforma = XP, data = Jan 2003).**

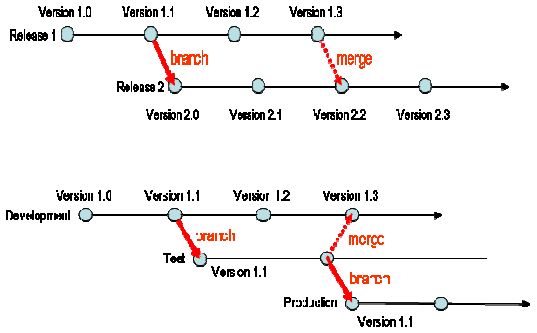
Branching e Merging

- Um elemento fundamental do gerenciamento de configuração
- *Branching:* Consiste em usar diferentes "ramos" de desenvolvimento para aumentar o paralelismo
  - Cada ramo é chamada de *branch*
  - Código não é compartilhado entre *branches*
- Merging: a combinação de uma desses ramos com o ramo principal
  - Diferenças entre os *branches* combinados precisam ser resolvidas
- Compromisso entre produtividade e risco

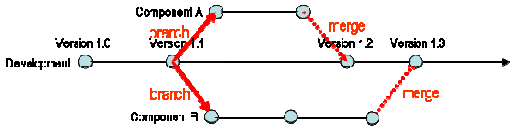
Algumas Razões para se Criar um *Branch*

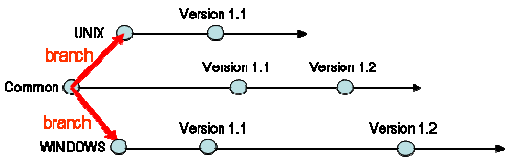
- Implementar uma solicitação de mudança
- Implementar uma funcionalidade pontual
- Paralelizar o desenvolvimento dos componentes do sistema
  - Também aplicável ao desenvolvimento paralelo de diferentes versões do sistema
  - Atribuição de tarefas a diferentes partes da equipe de desenvolvimento

Branch-per-Release e *Code-Promotion-Branches*



Branch-per-Component e *Branch-per-Technology*

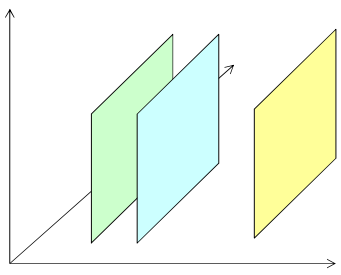




Baseline

- Uma especificação ou produto que foi formalmente revisado e aceito
  - Serve como base para os passos posteriores do desenvolvimento
- A configuração do software em um ponto discreto no tempo
- Só pode ser modificado através de procedimentos formais (solicitações de mudança)

Baseline



Razões para Criar um Baseline

- **Reproducibilidade** – a habilidade de reproduzir uma versão anterior do sistema
- **Rastreabilidade** – Estabelece uma relação predecessor-sucessor entre artefatos do projeto (projeto satisfaz requisitos, código implementa projeto, etc.)
- **Geração de Relatórios** – A comparação dos conteúdos de dois *baselines* ajuda na depuração e criação de documentação
- **Controle de Mudanças** – referencial para comparações, discussões e negociações

Baselines importantes

- Baselines são considerados marcos no processo de desenvolvimento:
  - Funcional: requisitos
  - De Produto: releases, iterações

Funcionalidades de um Sistema de Controle de Versões

- Manutenção de um repositório de itens de configuração
  - Com suporte ao *checkin* e ao *checkout* distribuídos
- Criação e manutenção de múltiplas versões
  - *Armazenamento de informações sobre cada versão*
- Criação e *merging* de *branches*
- Capacidade de realizar consultas sobre versões dos sistemas, com base em seus atributos

Gerenciamento e Registro de Mudanças

- Sistemas de software estão sujeitos a solicitações contínuas de mudanças:
  - De usuários;
  - De desenvolvedores;
  - De forças de mercado.
- O gerenciamento de mudanças está relacionado à manutenção da rastreabilidade dessas mudanças, de modo que:
  - Reparos realmente corrijam falhas
  - Novas falhas introduzidas por reparos possam ser identificadas rapidamente
  - Seja fácil descobrir quais mudanças foram implementadas e quando

igura 29.4

ormulário de solicitação de mudança arcialmente preenchido.

Formulário de Solicitação de Mudança

Projeto: Proteus/Ferramenta PCL **Número:** 23/02
Solicitante da mudança: I. Sommerville **Data:** 1/12/02
Mudança solicitada: Quando um componente é selecionado da estrutura, apresentar o nome do arquivo onde ele está armazenado.

Analista da mudança: G. Dean **Data da análise:** 10/12/02
Componentes afetados: Display-Icon.Select, Display-Icon.Display

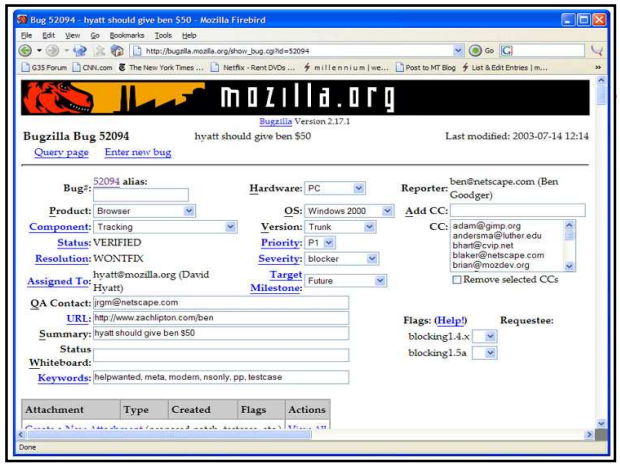
Componentes associados: FileTable

Avaliação da mudança: Relativamente simples de implementar se uma tabela de nome de arquivo estiver disponível. Requer o projeto e a implementação de um campo na tela. Não é requerida nenhuma mudança dos componentes associados.

Prioridade da mudança: Baixa
Implementação da mudança:
Esforço estimado: 0,5 dia
Data para o CCB: 15/12/02 **Data de decisão do CCB:** 1/2/03
Decisão do CCB: Aceita a mudança. Mudança a ser implementada no Release 2.1
Implementador da mudança: **Data da mudança:**
Data da submissão ao GQ: **Decisão do GQ:**
Data da submissão ao CM:
Comentários

Acompanhamento de mudanças

- O maior problema no gerenciamento de mudanças é o acompanhamento do status da mudança.
- Ferramentas de gerenciamento de mudanças fornecem meios para se acompanhar a situação de cada solicitação de mudança
  - Automaticamente enviam solicitações de mudança para as pessoas certas no tempo certo.
- São integrados a sistemas de e-mail, permitindo a distribuição eletrônica da solicitação de mudança.
  - Mesmo assim, é comum que solicitações de mudanças sejam ignoradas



Comitê de controle de mudanças

- As mudanças podem ser revisadas por um grupo que
  - decide se elas são ou não adequadas em termos de custo, tempo e risco
  - Sob um ponto de vista estratégico ou organizacional ao invés de um ponto de vista técnico.
- Esse grupo é, geralmente, chamado de comitê de controle de mudanças (CCB).
- O CCB pode conter representantes do cliente e do pessoal fornecedor.

Análise de impacto

- Mudanças de grande impacto devem ser comunicadas aos *stakeholders* envolvidos
- Análises de custo x benefício produzidas pelos *stakeholders*
- Mudança pode ser rejeitada se o CCB perceber que o custo será mais caro que o benefício percebido
- Mudanças devem ser priorizadas
- Por questões de eficiência, algumas solicitações de mudança podem ser agrupadas por tema, subsistema ou área de negócio

Procedência histórica

- É um registro das mudanças realizadas em um documento ou um componente de código.
- Deve registrar, em linhas gerais, a mudança feita, a lógica da mudança, quem fez a mudança e quando foi implementada.
- Pode ser incluída como um comentário no código.
  - Se um estilo de cabeçalho padrão é usado para a procedência histórica, as ferramentas podem processar isso automaticamente.

Informação de cabeçalho de componente

Figura 29.5
Informação de cabeçalho de componente.

```
// Projeto BANKSEC (IST 6087)
//
// BANKSEC-TOOLS/AUTH/RBAC/USER_ROLE
//
// Objeto: currentRole
// Autor: N. Perwaiz
// Data de criação: 10 de novembro de 2002
//
// (c) Lancaster University 2002
//
// Histórico de modificação
// Versão  Implementador  Data       Mudança          Razão
//1.0      J. Jones       1/12/2002  Adicionar cabeçalho Submetido ao CM
//1.1      N. Perwaiz     9/4/2003   Novo campo       Solicit. de mud. R07/02
```


Algumas Ferramentas de Gerenciamento de Mudanças

- Bugzilla 
- Eventum
- IBM Rational ClearCase
- Mantis 

- Também é possível usar um Wiki, planilhas e documentos (ex: Word) com esse fim

Construção (*build*) de sistemas

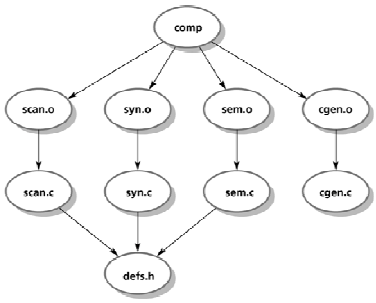
- É o processo de compilação e ligação de componentes de software em um sistema executável.
  - Pode incluir a execução de testes
- Sistemas diferentes são construídos a partir de combinações diferentes de componentes.
- Esse processo geralmente é apoiado por ferramentas automatizadas que são dirigidas por 'scripts de construção'.

Construção de sistemas

- A construção de um sistema grande é computacionalmente dispendiosa e pode levar várias horas.
- Centenas de arquivos podem estar envolvidos.
- As ferramentas de construção de sistemas podem fornecer:
  - Uma linguagem de especificação de dependência e um interpretador associado;
  - Seleção de ferramentas e apoio à instanciação;
  - Compilação distribuída.

Dependências entre componentes

Figura 29.9
Dependências entre componentes.



Construção frequente do sistema

- É mais fácil encontrar problemas que surgem das interações de componentes no início do processo.
  - Em especial quando usa-se incrementos pequenos e *builds* frequentes
- Isso encoraja o uso de testes automatizados – os desenvolvedores estão sob pressão para não 'quebrar a construção'.
- O processo de gerenciamento de mudanças precisa alcançar equilíbrio:
  - Burocracia vs. Rastreabilidade

Algumas Ferramentas de Controle de Versões e Geração de *Builds*

- CVS (+ WinCVS)
- SVN
- Git
- IBM Rational ClearCase
- Ant
- GNU Make