

DESCRIÇÃO

Gerenciamento de configurações no processo de desenvolvimento de software.

PROpósito

Compreender o gerenciamento de configurações como parte do processo de desenvolvimento de software, incluindo o controle de alterações, a construção de sistemas e o papel das ferramentas CASE, é importante para a formação do engenheiro de software.

OBJETIVOS

MÓDULO 1

Descrever o planejamento de gerenciamento de configurações

MÓDULO 2

Descrever o gerenciamento de mudanças, versões e releases

MÓDULO 3

Descrever a construção de sistemas

MÓDULO 4

Descrever as ferramentas CASE para gerenciamento de configurações

INTRODUÇÃO

Este tema trata do processo de gerenciamento de configurações de software que inclui o gerenciamento de mudanças, de versões e de releases de um sistema de software.

O gerenciamento de mudanças permite que as alterações sejam aplicadas ao sistema de uma maneira controlada, ou seja, garante que a evolução do sistema seja controlada e que as alterações mais urgentes e com bom custo-benefício sejam priorizadas, cabendo ao gerenciamento de versões controlar as diferentes versões de componentes de software e dos sistemas, envolvendo também a garantia de que as alterações feitas por diferentes desenvolvedores não interfiram umas com as outras.

O processo de construção de sistema executável, que ocorre por meio da compilação e da ligação dos componentes do sistema, das bibliotecas externas, dos arquivos de configuração e de outras informações necessárias ao referido processo é fundamental no contexto do gerenciamento de configurações.

Trataremos também das ferramentas CASE, pois estas permitem que as atividades principais de gerenciamento de configuração resultem em ganhos de produtividade no gerenciamento e controle de itens de configuração gerados ao longo do processo de desenvolvimento de software.

MÓDULO 1

-
- Descrever o planejamento de gerenciamento de configurações

CONCEITO DE GERENCIAMENTO DE CONFIGURAÇÕES

Consideremos que uma equipe de engenheiros de software desenvolve uma aplicação para gestão de vendas cujo fluxo de processo é iterativo e incremental, aplicando-se a cada subconjunto de requisitos as tarefas típicas de um processo de desenvolvimento de software, isto é, **levantamento de requisitos, análise, projeto, implementação, testes e implantação**.

Ao final, após o aval da equipe de qualidade, é disponibilizada aos usuários uma versão em produção. Imediatamente, essa equipe aplica as referidas tarefas em um segundo subconjunto de requisitos, sendo que, durante a implementação da versão 2, soa o alarme do ambiente de produção informando que foi detectada uma falha na versão 1.

Temos dois cenários possíveis:

CENÁRIO 1

1

Finalizar a versão 2 com as alterações necessárias à eliminação do defeito.

OU

CENÁRIO 2

2

Aplicar as alterações na versão 1, colocá-la em produção e refletir as alterações na versão 2.

O que você faria? Seria tecnicamente viável um software com defeito em produção até que seja substituído por uma nova versão?

RESPOSTA

Este cenário, sem dúvida, seria um risco. Neste caso, a melhor solução técnica é o cenário 2. Temos basicamente duas tarefas a executar: Primeira, obter a versão 1 e realizar as alterações devidas e, segunda, replicar as alterações realizadas na versão 1 na versão em desenvolvimento.

A solução para o nosso pequeno estudo de caso exige a implantação de um processo de gerenciamento de configurações de software (GCS).

ATENÇÃO

Importante, o software está em constante processo de mudança, principalmente pela volatilidade dos requisitos, pela evolução tecnológica ou pelos defeitos detectados, ou seja, surgem necessidades de alterações em função de atualização de plataforma, de novos recursos de hardwares, de adaptações a novos requisitos do negócio, de restrições orçamentárias ou de cronograma que causam a redefinição do sistema ou produto e outros.

O GCS inclui um conjunto de atividades destinadas a gerenciar as alterações identificando os artefatos que precisam ser alterados, estabelecendo relações entre eles, definindo mecanismos para gerenciar diferentes versões desses artefatos, controlando as alterações impostas e auditando e relatando as alterações feitas. O GCS gerencia alterações através de todo o ciclo de vida de um software, podendo ser visto como uma atividade de garantia de qualidade do software aplicada através de todo o processo de desenvolvimento do software.

Comumente, tais atividades cabem à equipe de qualidade, podendo ser designada uma equipe específica para controlar o processo GCS. Cabe destacar que alterações não controladas podem impactar negativamente a qualidade do software, sendo a gestão de alterações parte essencial da gestão da qualidade.

Pressman (2016), identifica quatro importantes elementos que devem compor um sistema de GCS:

ELEMENTOS DE COMPONENTE

Conjunto de ferramentas acopladas em um sistema de gestão de arquivos que possibilita acesso à gestão de cada item de configuração de software.

ELEMENTOS DE PROCESSO

Coleção de ações e tarefas que definem uma abordagem eficaz da gestão de alterações para todas as partes envolvidas na gestão, engenharia e uso do software.

ELEMENTOS DE CONSTRUÇÃO

Conjunto de ferramentas que automatizam a construção do software, assegurando que tenha sido montado o conjunto apropriado de componentes validados.

ELEMENTOS HUMANOS

Conjunto de ferramentas e características de processo usados pela equipe de software para implementar um GCS eficaz.

PLANEJAMENTO DE GERENCIAMENTO DE CONFIGURAÇÕES

O plano de gerenciamento de configurações descreve os padrões e procedimentos que devem ser adaptados para atender aos requisitos e às restrições de cada projeto específico. O plano de GCS deve incluir:

1. Definição dos itens de configuração de Software (ICS) que serão gerenciados e o esquema que se deve usar para identificar essas entidades.

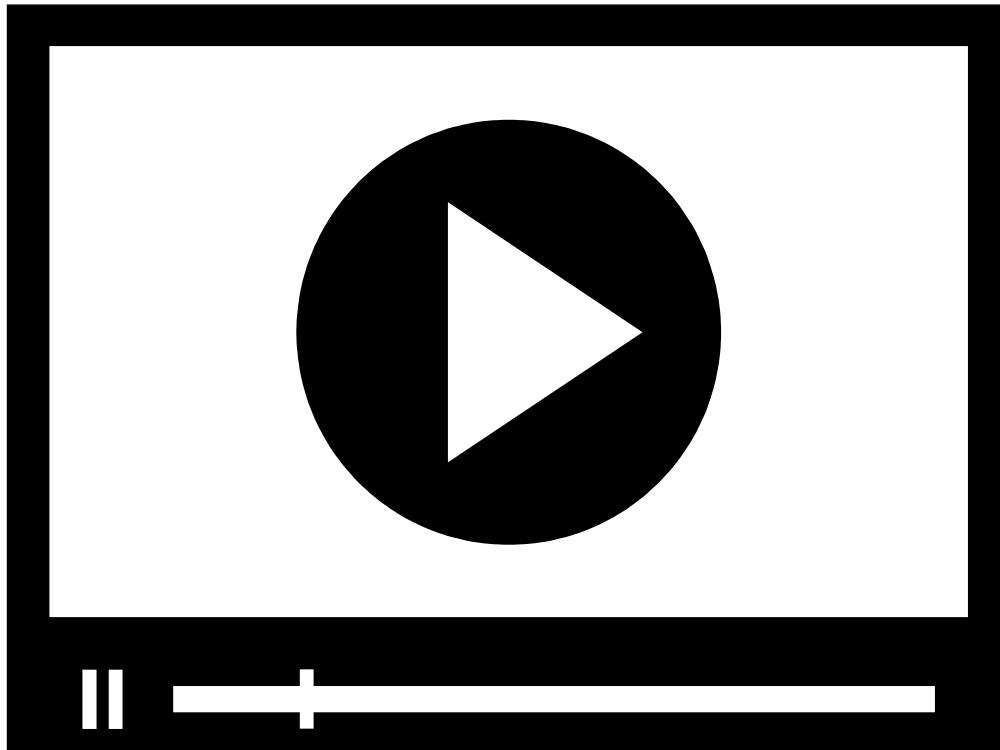
2. Estabelecimento do responsável pelos procedimentos de gerenciamento de configuração e pela submissão de itens controlados para a equipe de gerenciamento de configurações.

3. Definição das políticas de gerenciamento de configurações que todos os membros da equipe devem adotar para o controle de mudanças e gerenciamento de versões.

4. Especificação das ferramentas aplicadas no gerenciamento de configurações e os processos para uso dessas ferramentas.

5. Descrição da estrutura do banco de dados de configuração usada para registrar as informações de configuração e as informações que devem ser mantidas neste banco de dados, ou seja, os registros de configurações.

A inclusão de outros tópicos no plano GCS é possível, tal como o gerenciamento de software oriundo de fornecedores externos e procedimentos de auditoria dos processos de GCS. A definição de responsabilidades inclui quem é o responsável pela entrega de cada documento ou de componente de software para a garantia da qualidade e para o gerenciamento de configuração, bem como os revisores de cada documento.



Assista, a seguir, um vídeo sobre Itens de Configuração de Software:



GESTÃO DE ITEM DE CONFIGURAÇÃO DE SOFTWARE (ICS)

Um fato comum no desenvolvimento de software é a alteração, pois clientes querem modificar requisitos, engenheiros de software e gerentes querem modificar seus métodos.

COMENTÁRIO

Por que tanta necessidade de mudanças? O entendimento do problema evolui durante o processo de desenvolvimento de software, gerando a necessidade de alterações.

Vamos ao entendimento do conceito item de configuração de software (ICS), cuja identificação está prevista no plano de GCS. Um ICS é uma informação criada como parte do processo de engenharia de software, tal como única seção de uma grande especificação ou um **caso de teste** em um grande conjunto de testes, ou seja:

CASO DE TESTE

Em Engenharia de Software, caso de teste é um conjunto de condições usadas para teste de software.

EXEMPLO

Um ICS é todo ou parte de um artefato, por exemplo, um documento, um conjunto inteiro de casos de teste ou um programa ou componente com nome.

Um ICS pode incluir ferramentas de software sob o controle de configuração, tais como versões específicas de editores, compiladores, browsers e outras ferramentas automáticas.

Como essas ferramentas foram usadas para produzir documentação, código-fonte e dados, elas devem estar disponíveis quando alterações forem feitas na configuração do software, pois uma nova versão de uma ferramenta (por exemplo, uma **IDE**) pode produzir resultados diferentes daqueles da versão original.

IDE

Do inglês Integrated Development Environment ou Ambiente de Desenvolvimento Integrado, é um programa de computador que reúne características e ferramentas de apoio ao

desenvolvimento de software com o objetivo de agilizar este processo.

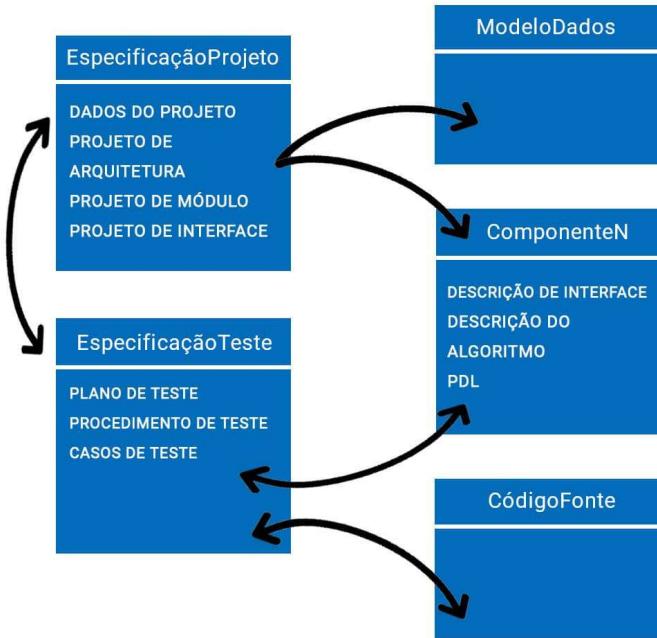


Figura 1 – Objetos de configuração. Fonte: O Autor.

Os ICS são organizados como objetos de configuração que podem ser persistidos em um banco de dados, sendo que cada objeto de configuração tem um nome, atributos e pode estar associado a outros objetos. De acordo com a Figura 1, os objetos de configuração *EspecificaçãoProjeto*, *ModeloDados*, *ComponenteN*, *CódigoFonte* e *EspecificaçãoTeste* são definidos individualmente, estando relacionados entre si, como mostram as setas. As setas curvas são indicativas de relacionamento de composição, de modo que *ModeloDados* e *ComponenteN* são parte do objeto *EspecificaçãoProjeto*.

?

VOCÊ SABIA

Um conceito importante na gestão de configuração de software é a referência, que é uma especificação ou produto que tenha sido formalmente revisado e acordado, que depois serve de base para mais desenvolvimento, e pode ser alterado somente por meio de procedimentos formais de controle de alteração.

Uma vez estabelecido um ICS como uma referência, podem ser feitas alterações, mas deve ser aplicado um processo específico e formal para avaliar e verificar cada alteração. No contexto de Engenharia de Software, uma referência é um marco no desenvolvimento de

software. Uma referência é marcada pelo fornecimento de um ou mais ICS que foram aprovados em consequência de uma revisão técnica.

Vamos exemplificar:

Considerando que os elementos de um modelo de projeto foram documentados e que todas as partes do modelo foram revisadas, corrigidas e então aprovadas, o modelo do projeto torna-se uma referência. Outras alterações na arquitetura do programa, documentadas no modelo de projeto, podem ser feitas apenas depois que cada uma tenha sido avaliada e aprovada.

Embora as referências possam ser definidas em qualquer nível de detalhe, as referências de software mais comuns estão na Figura 2.

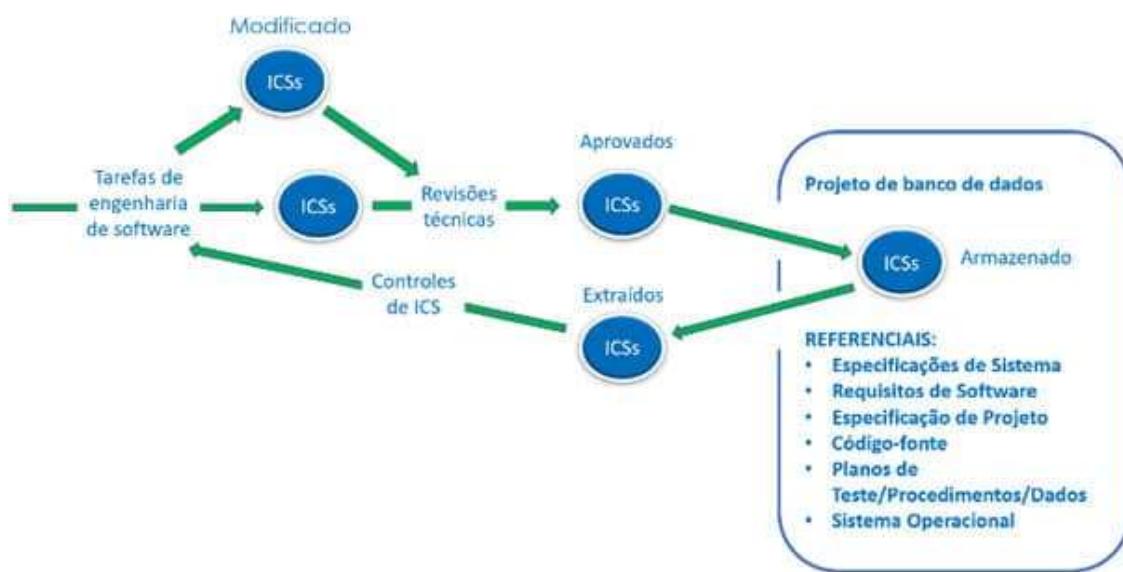


Figura 2 – ICS versus banco de dados de projeto. Fonte: O Autor.

A Figura 2 ilustra a sequência de eventos que levam a uma referência, pois a partir de tarefas de engenharia de software são gerados os ICSs, sendo posteriormente revisados, aprovados e registrados em um banco de dados de projeto ou repositório de software.

Quando um engenheiro de software quer fazer uma alteração em um ICS que se tornou referencial, o mesmo é copiado do banco de dados de projeto para o ambiente de desenvolvimento do engenheiro. Porém, esse ICS somente poderá ser modificado se os controles de GCS, tal como o controle de versão, forem seguidos.

RESUMINDO

Vimos neste módulo que o gerenciamento de configurações de software (GCS) inclui um conjunto de atividades destinadas a gerenciar alterações, identificando os artefatos que precisam ser alterados, estabelecendo relações entre eles, definindo mecanismos para gerenciar diferentes versões desses artefatos, controlando as alterações impostas e auditando e relatando as alterações feitas.

O plano de gerenciamento de configurações descreve os padrões e procedimentos que devem ser adaptados para atender aos requisitos e às restrições de cada projeto específico, cabendo destaque à identificação dos itens de configuração de software (ICS) e da base de dados onde serão armazenados os ICSs.

Um ICS é uma informação criada como parte do processo de engenharia de software, tal como um caso de teste em um grande conjunto de testes, podendo o mesmo tornar-se um referencial registrado no banco de dados. A partir deste registro, qualquer alteração será aplicada de acordo com controles de GCS.

Agora é com você! Vamos verificar se atingimos o objetivo deste módulo? Você está convidado a realizar as atividades a seguir.

VERIFICANDO O APRENDIZADO

MÓDULO 2

-
- Descrever o gerenciamento das mudanças, versões e releases

PROCESSO DE GERENCIAMENTO DE CONFIGURAÇÕES DE SOFTWARE

O processo de gerenciamento de configurações de software (GCS), proposto por Pressman (2016), inclui tarefas que têm quatro objetivos primários:

Identificar todos os itens que coletivamente definem a configuração do software

Gerenciar alterações de um ou mais desses itens

Facilitar a construção de diferentes versões de uma aplicação

Assegurar que a qualidade do software seja mantida à medida que a configuração evolui com o tempo

O referido processo define cinco tarefas de GCS: Identificação, controle de versão, controle de alteração, auditoria de configuração e relatos, tal como ilustra a Figura 3.

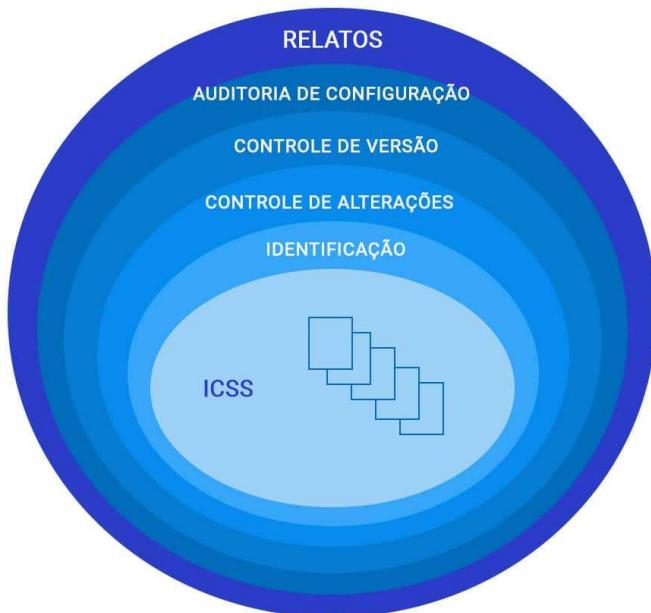


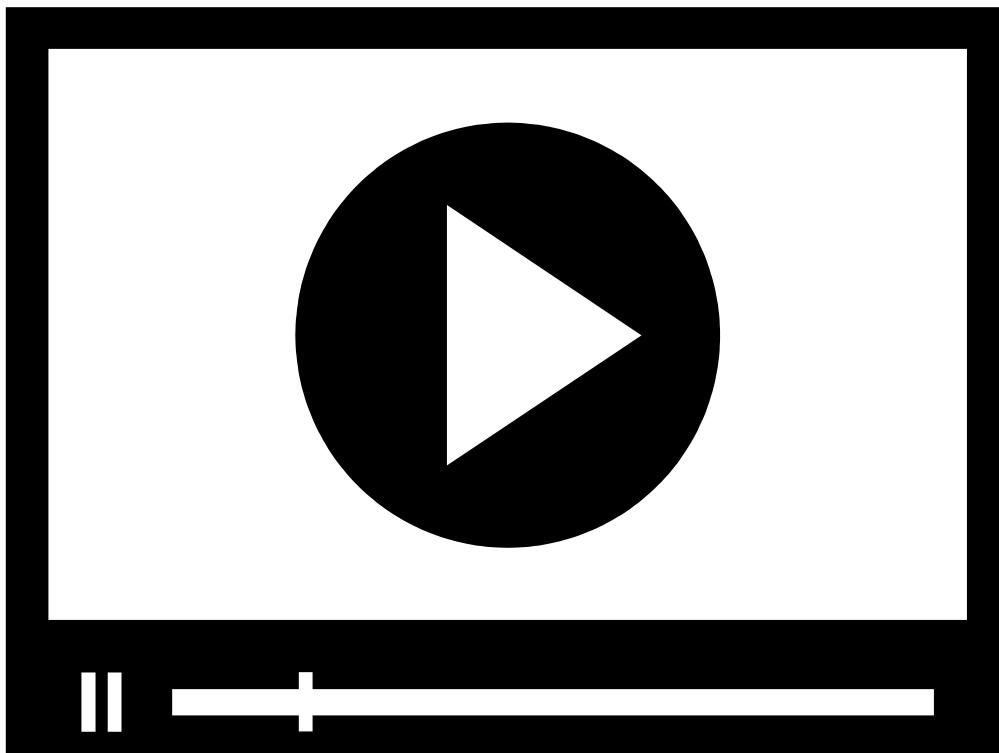
Figura 3 - Camadas do processo GCS. Fonte: O Autor.

► ATENÇÃO

As camadas são concêntricas, de modo que os itens de configuração de software (ICS) transitam da camada mais interna para as camadas externas, tornando-se parte da configuração do software de uma ou mais versões da aplicação ou sistema.

À medida que um ICS se move através de uma camada, as ações deduzidas por cada tarefa GCS podem ser ou não aplicáveis, pois caso não sejam solicitadas alterações para o ICS, a

camada de controle de alteração não se aplica. Um ICS possui uma versão específica do software, sendo mantido um registro do ICS, tal como seu nome, data de criação, designação da versão etc, para fins de auditoria da configuração e relato para aqueles que precisam ter conhecimento.



No vídeo a seguir, você assistirá explicações sobre Controle de alterações de software.



GERENCIAMENTO DE MUDANÇAS

A camada controle de alterações, definida na Figura 3, é denominada de gerenciamento de mudanças por Sommerville (2019). As mudanças podem ser necessárias a partir de alterações nas necessidades organizacionais e nos requisitos, ou em função de defeitos que necessitam ser corrigidos.

O gerenciamento de mudanças permite que as alterações sejam aplicadas ao sistema de uma maneira controlada, ou seja, garante que a evolução do sistema seja controlada e que as alterações mais urgentes e com bom custo-benefício sejam priorizadas.

A Figura 4 ilustra um processo de gerenciamento de mudanças que deve entrar em vigor quando o software for entregue para lançamento aos clientes ou para implantação dentro de uma organização, cabendo destacar que o formalismo depende da complexidade do software.

► ATENÇÃO

Importante, também, são as ferramentas utilizadas no gerenciamento de mudanças, tais como um sistema de rastreamento de defeitos ou um software integrado a um pacote de gerenciamento de configuração de sistemas de larga escala, como o **Rational Clear Case**.

RATIONAL CLEAR CASE

É uma família de ferramentas de software de computador que suporta gerenciamento de configuração de software (GCS) de código-fonte e outros ativos de desenvolvimento de software, sendo desenvolvido pela IBM.

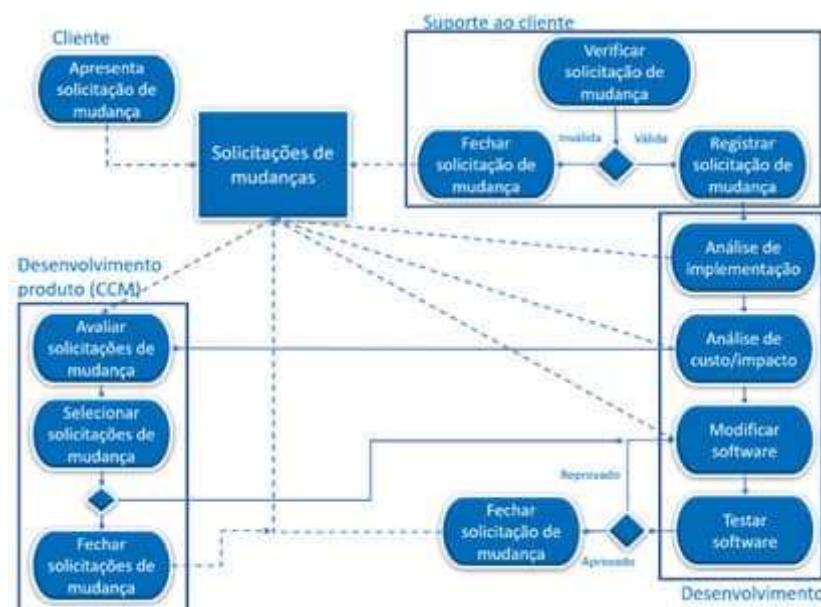


Figura 4 – Gerenciamento de mudança. Fonte: O Autor.

2
3
4
5

1

O referido processo começa quando uma parte interessada, clientes, usuários, testadores, desenvolvedores e outros, envia um formulário de solicitação de mudança (FSM), descrevendo a mudança necessária no sistema, podendo essa solicitação ser um relatório de defeitos, no qual os sintomas dos defeitos são descritos, ou um pedido para acrescentar funcionalidade ao sistema.

2

À medida que a solicitação de mudança é processada, informações são adicionadas ao FSM em cada estágio do processo representando uma fotografia do estado da solicitação de mudança, podendo ser registradas as recomendações pertinentes à alteração, os seus custos estimados, as datas em que a mudança foi solicitada, aprovada, implementada e validada, ou uma descrição como a alteração pode ser implementada.

3

Em seguida, a solicitação de mudança é conferida para garantir que seja válida, podendo ser rejeitada, mas caso seja válida, será registrada como uma solicitação pendente para a próxima etapa do processo, que é a avaliação da mudança e o cálculo dos custos. Esta função verifica o impacto da alteração, sendo preciso identificar todos os componentes afetados pela mudança, e, na sequência, estima-se o custo de proceder com essas mudanças.

4

Um grupo, designado de comitê de controle de mudança (CCM) ou grupo de desenvolvimento do produto, analisa e aprova todas as solicitações de mudança, a menos que as alterações envolvam simples correções de erros menores em telas, páginas da web ou documentos. Estes pequenos pedidos devem ser passados para implementação imediata pelo time de desenvolvimento.

5

O CCM, ou o grupo de desenvolvimento do produto, decide se a mudança em questão é economicamente justificável e prioriza as mudanças aceitas para serem implementadas, sendo as mudanças aceitas devolvidas ao grupo de desenvolvimento.

② VOCÊ SABIA

Os fatores que influenciam a decisão sobre a implementação ou não de uma mudança incluem as consequências de não fazer a mudança em caso de uma falha de software muito grave que compromete o uso operacional do sistema; usuários beneficiados pela mudança; os custos envolvidos na mudança; caso uma nova versão do software tenha sido disponibilizada recentemente aos clientes, pode fazer sentido postergar a mudança para a próxima mudança.

Nos produtos de software, tal como o **CAD**, as solicitações de mudança vêm da equipe de suporte ao cliente, da equipe de marketing da empresa ou dos próprios desenvolvedores, sendo que esses pedidos podem refletir sugestões e feedbacks de clientes ou análises do que é oferecido pelos produtos concorrentes. Neste caso, o processo de solicitação de mudança, mostrado na Figura 4, é iniciado depois da liberação do sistema aos clientes.

CAD

Desenho assistido por computador (DAC; em inglês: computer aided design - CAD) é o nome genérico de sistemas computacionais (software) utilizados pela Engenharia, Geologia, Geografia, Arquitetura e Design para facilitar o projeto e desenho técnicos.

Durante o desenvolvimento, quando novas versões do sistema são criadas por meio de construções sistemáticas, não há necessidade de um processo formal de gerenciamento de mudanças, pois as mudanças que afetam os componentes individuais são passadas diretamente para o desenvolvedor do sistema, que pode aceitá-las ou indicar por que não são necessárias. No entanto, uma autoridade independente, como um arquiteto do sistema, deve avaliar e priorizar as mudanças que afetam os módulos do sistema produzidos por diferentes times de desenvolvimento.

Em alguns métodos ágeis, os clientes propõem mudanças nos requisitos do sistema e trabalham com a equipe de desenvolvimento na avaliação dos impactos das mudanças e nas decisões relativas à implementação das mesmas.

GERENCIAMENTO DE VERSÕES

O gerenciamento de versões tem como objetivo controlar as diferentes versões de componentes de software e dos sistemas, envolvendo também a garantia de que as alterações feitas por diferentes desenvolvedores não interfiram umas com as outras. De acordo com Sommerville (2019), o gerenciamento de versões é o processo de gerenciamento de **codelines** e **baselines**.

CODELINES

Uma codeline é uma sequência de versões do código-fonte, sendo aplicadas a componentes de sistemas para que haja versões diferentes de cada componente.

BASELINES

Uma baseline é uma definição de um determinado sistema, especificando as versões dos componentes incluídas no sistema e a identificação das bibliotecas utilizadas, dos arquivos de configuração e de outras informações do sistema.

A Figura 5 ilustra que diferentes baselines usam versões diferentes dos componentes de cada codeline, onde podemos destacar a mainline que incorpora uma sequência de baselines representando diferentes versões do sistema.

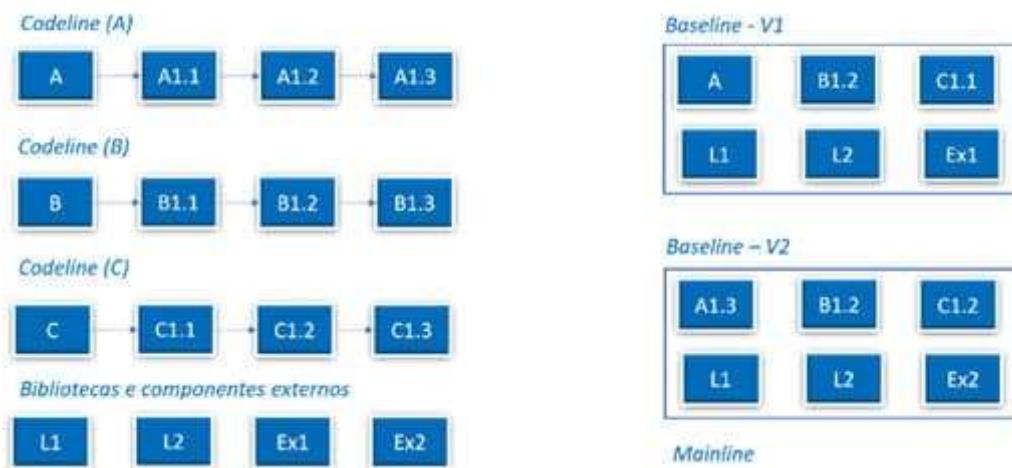


Figura 5 – Codelines e baselines. Fonte: O Autor.

As baselines são importantes porque, muitas vezes, é preciso recriar uma versão individual de um sistema, tal como a instanciação de um software em versões específicas para cada cliente, ou mesmo recriar a versão entregue a um cliente, caso relate defeitos em seu sistema que necessitem ser alterados.

Existem dois tipos de sistemas de controle de versão modernos:

1

Sistemas centralizados compostos por um único repositório mestre que mantém todas as versões dos componentes de software.

2

Sistemas distribuídos compostos por várias versões do repositório de componentes ao mesmo tempo.

Ambos os sistemas possuem funcionalidades semelhantes, mas as implementam de maneiras diferentes.

As principais características destes sistemas são:

IDENTIFICAÇÃO DE VERSÃO E LANÇAMENTO

As versões gerenciadas de um componente recebem identificadores únicos quando são submetidas a um sistema, permitindo que diferentes versões do mesmo componente sejam gerenciadas, sem alterar o nome dele.

REGISTRO DO HISTÓRICO DE MUDANÇAS

Manutenção dos registros das mudanças feitas para criar uma versão nova de um componente, com base em uma versão anterior.

DESENVOLVIMENTO INDEPENDENTE

Permite que diferentes desenvolvedores possam estar trabalhando simultaneamente no mesmo componente, controlando os componentes em que se fez check-out para edição, e assegura que as mudanças feitas em um componente por desenvolvedores diferentes não interfiram umas nas outras.

APOIO DE PROJETO

Apoia o desenvolvimento de vários projetos que compartilham componentes.

GERENCIAMENTO DE ARMAZENAMENTO

Pode utilizar mecanismos eficientes que garantam que não sejam mantidas cópias duplicadas de arquivos idênticos, tal como casos em que há apenas pequenas diferenças entre os arquivos. O sistema de controle de versão pode armazenar estas diferenças em vez de manter várias cópias dos arquivos.

A maior parte do desenvolvimento de software é uma atividade para ser realizada por um time, então, frequentemente, diversos membros trabalham no mesmo componente de modo simultâneo.

Vamos exemplificar um cenário baseado na Figura 6, a fim de que possamos entender como funcionam os sistemas com repositório centralizado, tal como o SVN (Subversion).

Um engenheiro de software utiliza uma área de trabalho “ALFA”, um segundo engenheiro, a área de trabalho “BETA”, ambos requerem mudanças em componentes comuns. Para apoiar o desenvolvimento independente sem interferências, todos os sistemas de controle de versão usam o conceito de um repositório de projeto, mantendo uma versão 'mestre' de todos os componentes, que é utilizada para criar as baselines para a construção do sistema.

Ao modificar componentes, os desenvolvedores copiam estes componentes do repositório para sua área de trabalho, ou seja, realizam a operação de check-out, e executam implementações

nestas cópias. Após implementadas as alterações, realizam a operação de check-in, ou seja, os componentes modificados são devolvidos para o repositório, criando uma versão nova do componente, que pode ser compartilhada.



Figura 6 – Check-in e check-out de repositório centralizado. Fonte: O Autor.

Em um sistema de controle de versão distribuído, tal como o **Git**, existe um repositório 'mestre' no servidor que mantém o código produzido pelo time de desenvolvimento. Em vez de simplesmente fazer o check-out dos arquivos de que precisa, um desenvolvedor cria um clone do repositório do projeto, que é baixado e instalado em seu computador.

Os desenvolvedores trabalham nos arquivos necessários e mantêm as novas versões em seu repositório particular, em seu próprio computador. Quando terminam de fazer as alterações, realizam um **commit** nas mesmas, atualizando seu repositório. A disponibilização dessas alterações para o repositório do projeto ocorre por meio de uma operação push, cabendo ao gerente de configuração 'puxar' (pull) esses arquivos para o repositório do projeto, ou seja, um repositório pode se comunicar com qualquer outro através das operações básicas **pull** e **push**.

COMMIT

Agrupamento de um conjunto de alterações candidatas a serem atualizadas.

PULL

Atualiza o repositório local com todas as alterações feitas em outro repositório.

PUSH

Envia as alterações do repositório local para outro repositório.

GIT

Git é um sistema de controle de versões distribuído, usado principalmente no desenvolvimento de software, mas pode ser usado para registrar o histórico de edições de qualquer tipo de arquivo (Exemplo: alguns livros digitais são disponibilizados no GitHub e escrito aos poucos publicamente). O Git foi inicialmente projetado e desenvolvido por Linus Torvalds para o desenvolvimento do kernel Linux, mas foi adotado por muitos outros projetos. (Wikipedia).

GERENCIAMENTO DE RELEASES

Uma release, ou lançamento de sistema, é uma versão do software distribuída para os clientes, podendo esses lançamentos serem categorizados, para software de mercado, em lançamentos principais (major releases), por oferecerem uma nova funcionalidade significativa, e lançamentos secundários (minor releases), que consertam defeitos e solucionam problemas relatados por clientes, tal como IDE 4.2.

● COMENTÁRIO

Isto significa um lançamento secundário 2 do lançamento principal 4 da IDE, sendo esses lançamentos secundários comumente gratuitos. Entretanto, provavelmente o cliente terá que pagar pela versão 5.0.

Um lançamento de produto de software não é apenas o código executável do sistema, podendo incluir:

Arquivos de configuração, que definem como o lançamento deve ser configurado para instalações específicas.

Arquivos de dados, como arquivos de mensagens de erro em diferentes idiomas, que são necessários para a operação bem-sucedida do sistema.

Um programa de instalação, a fim de ajudar na instalação do sistema no hardware de destino.

Documentação eletrônica ou em papel que descreva o sistema.

Embalagens e publicidade associada que foram concebidas para esse lançamento.

A complexidade do processo de gerenciamento de lançamentos do sistema depende do número de clientes, pois pode ser preciso criar lançamentos do sistema para cada cliente, em função, por exemplo, de diferentes hardwares. Neste caso, os sistemas e processos de gerenciamento de configurações têm de ser projetados para fornecer informações sobre quais clientes têm quais versões do sistema e a relação entre os lançamentos e as versões do sistema. No caso de um problema com um sistema entregue, o GCS deve ser capaz de recuperar todas as versões dos componentes utilizadas nesse sistema específico.

RESUMINDO

Vimos neste módulo que um processo de gestão de configuração de software (GCS) define cinco tarefas: Identificação dos itens de configuração de software (ICS), controle de versão, controle de alteração, auditoria de configuração e relatos.

O gerenciamento de mudanças permite que as alterações sejam aplicadas ao sistema de uma maneira controlada, ou seja, garante que a evolução do sistema seja controlada e que as alterações mais urgentes e com bom custo-benefício sejam priorizadas.

O gerenciamento de versões tem como objetivo controlar as diferentes versões de componentes de software e dos sistemas, envolvendo também a garantia de que as alterações feitas por diferentes desenvolvedores não interfiram umas com as outras.

O gerenciamento de releases, cuja complexidade depende do número de clientes, requer que os processos de gerenciamento de configuração sejam projetados para fornecer informações sobre quais clientes têm quais versões do sistema e a relação entre os lançamentos e as versões do sistema.

Agora é com você! Vamos verificar se atingimos o objetivo deste módulo? Você está convidado a realizar as atividades a seguir.

VERIFICANDO O APRENDIZADO

MÓDULO 3

- Descrever a construção de sistemas

CONSTRUÇÃO DE SISTEMAS



Fonte: Shutterstock

O processo de criar um sistema executável, também conhecido como build, compilando e ligando os componentes do sistema, as bibliotecas externas, os arquivos de configuração e outras informações, é denominado de construção de sistema.

DICA

As ferramentas de construção de sistema e de controle de versões devem ser integradas, pois o processo de construção usa versões dos componentes do repositório gerenciado pelo sistema de controle de versões. Considerando que a construção de um sistema envolve diferentes informações sobre o software e seu ambiente operacional, sugere-se o uso de uma ferramenta de construção automatizada para criar um build do sistema.

Podemos observar na Figura 7 que o referido build poderá incluir, além dos arquivos de código-fonte envolvidos na construção do sistema, as bibliotecas vinculadas externamente, os arquivos de dados, os arquivos de configuração, ou mesmo as especificações das versões do compilador e ferramentas de software utilizadas na construção.

Idealmente, deve ser possível construir um sistema completo com um único comando ou clique do mouse.



Figura 7 – Construção de sistemas. Fonte: O Autor.

Vejamos algumas funcionalidades usualmente disponíveis em ferramentas para integração e construção de sistemas:

GERAÇÃO DO SCRIPT DE CONSTRUÇÃO

Permite a análise do programa em construção, identificando componentes dependentes e gerando automaticamente um script de construção ou arquivo de configuração.

INTEGRAÇÃO COM O SISTEMA DE CONTROLE DE VERSÃO

Realiza o check-out das versões necessárias dos componentes que estão no sistema de controle de versão, de forma a garantir a integridade do build.

RECOMPILAÇÃO MÍNIMA

Analisa qual código-fonte precisa ser recompilado e realizar as compilações essenciais.

CRIAÇÃO DO SISTEMA EXECUTÁVEL

Vincula os arquivos objetos compilados uns aos outros e a outros arquivos necessários, como bibliotecas e arquivos de configuração, para criar um sistema executável.

AUTOMAÇÃO DOS TESTES

Executa automaticamente testes usando ferramentas de automação de teste, a fim de verificar a correta integração dos componentes do build em função das possíveis mudanças.

EMISSÃO DE RELATÓRIOS

Fornece relatórios sobre o sucesso ou a falha da construção, bem como dos testes executados.

GERAÇÃO DA DOCUMENTAÇÃO

Gera notas de lançamento sobre o build e páginas de ajuda do sistema.

O script de construção é uma definição do sistema a ser construído, incluindo informações sobre os componentes e suas dependências e versões das ferramentas usadas para compilar e ligar o sistema, sendo, então, necessário à linguagem de configuração adotada descrever os componentes do sistema a serem incluídos na construção e suas dependências.

Podemos observar na Figura 8 as plataformas de sistema envolvidas, o que confere um grau de complexidade no processo de construção.

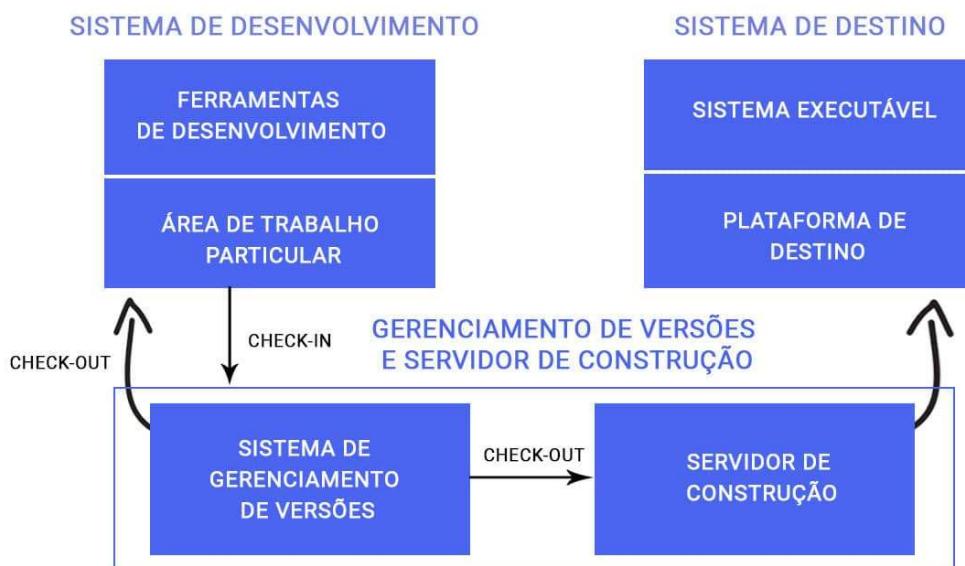


Figura 8 – Plataformas de desenvolvimento, construção e destino. Fonte: O Autor.

Vejamos as plataformas:

SISTEMA DE DESENVOLVIMENTO SERVIDOR DE CONSTRUÇÃO AMBIENTE DE DESTINO

SISTEMA DE DESENVOLVIMENTO

Inclui ferramentas de desenvolvimento, como compiladores e editores de código-fonte.

Os desenvolvedores fazem check-out do código do sistema de controle de versão, ou seja, transferem do repositório para uma área de trabalho particular antes de fazer alterações no sistema, sendo possível a criação de uma versão do sistema para teste em seu ambiente de desenvolvimento antes de efetivar as alterações feitas no sistema de controle de versão.

Ao final das alterações e/ou testes é realizado o check-in, ou seja, os componentes modificados são devolvidos ao repositório.

O processo que envolve o check-in e check-out requer o uso de ferramentas de construção locais que usam versões de componentes na área de trabalho particular.

SERVIDOR DE CONSTRUÇÃO

Utilizado na construção de versões definitivas e executáveis do sistema, mantendo as versões definitivas de um sistema.

Todos os desenvolvedores de sistema fazem check-in de código para o sistema de controle de versão no servidor de construção para construir o sistema.

O servidor de construção realiza o check-out de todos os componentes que irão compor a versão do sistema.

AMBIENTE DE DESTINO

Esse ambiente inclui a plataforma na qual o sistema é executado, podendo ser o hardware de destino semelhante ao do ambiente de desenvolvimento, ou, tal como em sistemas em tempo real e embarcados, o ambiente de destino pode ser mais simples.

Em determinados sistemas, o ambiente de destino inclui bancos de dados e outros sistemas de aplicações que não podem ser instalados em máquinas de desenvolvimento.

INTEGRAÇÃO CONTÍNUA

Os métodos ágeis recomendam que a construção do sistema seja realizada com bastante frequência, com testes automatizados utilizados para descobrir problemas de software. As construções frequentes fazem parte de um processo de integração contínua, como mostrado na Figura 9.

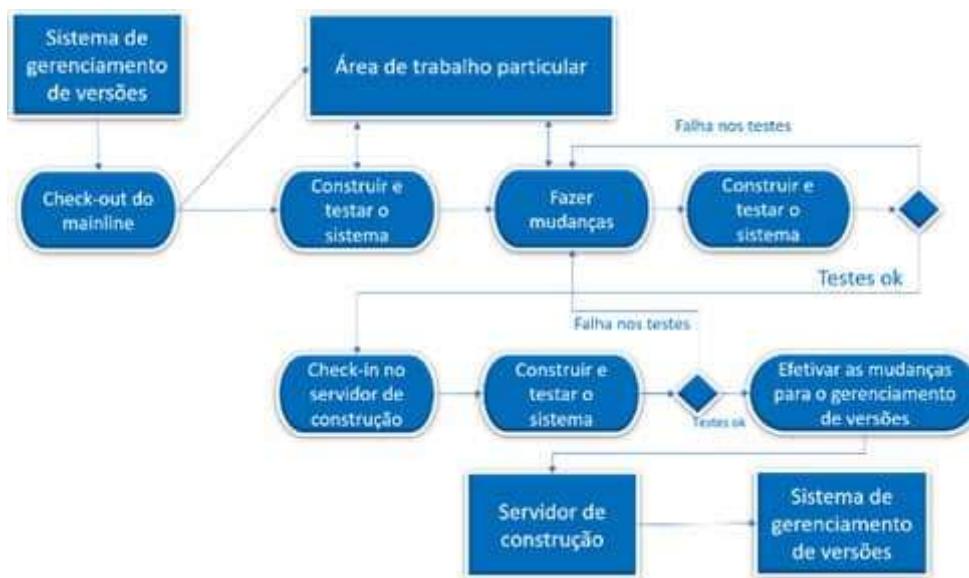
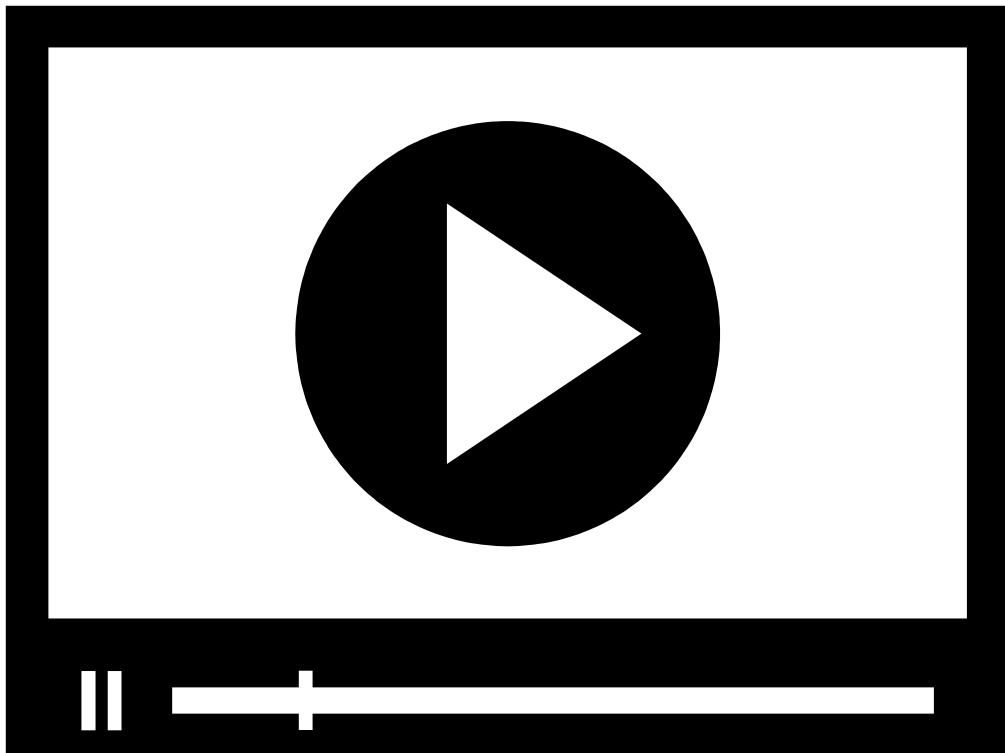
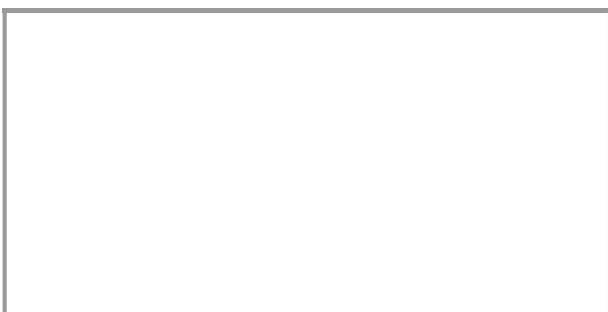


 Figura 9 – Integração contínua. Fonte: O Autor.



Assista, a seguir, um vídeo sobre O processo de integração contínua em metodologias ágeis.



📣 ATENÇÃO

As metodologias ágeis adotam, entre outras melhores práticas, a possibilidade de atendimento rápido às mudanças e a integração de forma sistemática, o que requer a reconstrução frequente da mainline que representa uma sequência de baselines representando diferentes versões do sistema.

As etapas da integração contínua ilustradas na Figura 9 são:

- 1**
- 2**
- 3**

4
5
6
7

1

Extrair o sistema mainline do sistema de gerenciamento de versões para a área de trabalho particular do desenvolvedor.

2

Construir o sistema e executar testes automatizados para garantir que o sistema construído passe em todos os testes; no caso de falha, a construção é considerada “quebrada”, sendo necessário informar quem fez o check-in no último sistema baseline, pois essa pessoa será responsável por reparar o problema.

3

Fazer as mudanças nos componentes do sistema.

4

Construir o sistema em uma área de trabalho particular e executar novamente os testes de sistema; caso ocorram falhas, a edição deve prosseguir.

5

Uma vez que o sistema tenha passado nos testes, é preciso devolvê-lo para o servidor do sistema de construção, mas não efetivá-lo como um novo sistema baseline no sistema de gerenciamento de versões.

6

Construir o sistema no servidor de construção e rodar os testes; caso ainda ocorra alguma falha, é preciso fazer o check-out dos componentes que falharam e editá-los em uma área de trabalho particular para que passem no teste.

7

Após a aprovação nos testes no sistema de construção, então devem-se confirmar as alterações feitas como um novo baseline no mainline do sistema.

A vantagem da integração contínua é que ela permite a descoberta e a alteração imediata dos problemas causados pelas interações entre diferentes desenvolvedores, sendo o sistema mais recente no mainline o sistema funcional definitivo. Entretanto, podem ocorrer restrições na aplicação da integração contínua, vejamos:

Se o sistema for muito grande, pode levar muito tempo para construí-lo e testá-lo, especialmente se a integração com outros sistemas de aplicações estiver envolvida, podendo ser inviável a construção do sistema várias vezes ao dia.

Caso a plataforma de desenvolvimento seja diferente da plataforma de destino, pode não ser possível executar os testes do sistema na área de trabalho particular do desenvolvedor, em função de possíveis diferenças no hardware, no sistema operacional ou no software instalado, gerando a necessidade de um tempo maior para os testes.

Nos sistemas grandes ou naqueles cuja plataforma de execução não é a mesma da plataforma de desenvolvimento, a integração contínua costuma ser impossível.

Vejamos uma alternativa considerando um sistema de construção diária:

ALTERNATIVA

A equipe de desenvolvimento estabelece um horário de entrega, tal como 14h, dos componentes de sistema, cabendo aos desenvolvedores que tiverem novas versões dos componentes realizar e entrega no prazo estipulado.

A construção do sistema ocorre por meio da compilação e ligação dos componentes, sendo gerado um sistema completo.

O novo sistema gerado passa por um conjunto de testes pela equipe de testes.

Os defeitos que foram descobertos durante o teste do sistema são documentados e devolvidos para os desenvolvedores do sistema que realizam a manutenção devida.

A vantagem de usar construções frequentes do software é a maior chance de encontrar falhas nas interações de componentes de forma precoce. Desta forma, a construção frequente incentiva testes unitários completos dos componentes com o objetivo de evitar que o check-in de uma determinada versão gere uma falha no sistema inteiro.

LIGAÇÃO DO CÓDIGO-FONTE AO CÓDIGO OBJETO

Como a compilação é um processo computacionalmente intensivo realizado por um determinado **compilador**, ferramentas de apoio à construção do sistema podem ser projetadas para minimizar a quantidade de compilação necessária, verificando se está disponível uma versão compilada de um componente, pois, em caso positivo, não há necessidade de recompilar esse componente.

COMPILADOR

É um programa de computador que, a partir de um código-fonte escrito em uma linguagem compilada, cria um programa semanticamente equivalente, porém escrito em outra linguagem,

código objeto. Classicamente, um compilador traduz um programa de uma linguagem textual facilmente entendida por um ser humano para uma linguagem de máquina, específica para um processador e sistema operacional.

Entendamos como ocorre a ligação do código-fonte de um componente ao seu código objeto equivalente.

Esta ligação é realizada por meio da designação de uma assinatura exclusiva a cada arquivo de componente de código-fonte armazenado, tendo o código objeto correspondente, que foi compilado a partir do código-fonte, uma assinatura relacionada; caso o código-fonte seja editado, esta assinatura é alterada.

Pela comparação das assinaturas nos arquivos de código-fonte e de código objeto, é possível decidir se o componente de código-fonte foi usado para gerar o componente de código objeto.



Figura 10 – Ligando código-fonte e código objeto. Fonte: O Autor. Figura 10 – Ligando código-fonte e código objeto. Fonte: O Autor.

Vejamos os dois tipos de assinaturas apresentados por Sommerville (2019), conforme exibido na Figura 10.

TIMESTAMPS DE MODIFICAÇÃO

A assinatura no arquivo de código-fonte é a data e a hora (timestamps) em que esse arquivo foi modificado. Sempre que ocorrer a modificação do arquivo de código-fonte de um componente, o sistema realiza a recompilação para criar um novo arquivo de código objeto correspondente.

Observemos a Figura 10, na qual os componentes *Comp.java* e *Comp.class* têm assinaturas de modificação de 17:03:05:15:02:2018 e 16:58:43:15:02:2018, respectivamente. Isto significa

que o código Java foi modificado às 17 horas, 3 minutos e 5 segundos do dia 15 de fevereiro de 2018 e a versão compilada foi modificada às 16 horas, 58 minutos e 43 segundos do dia 15 de fevereiro de 2018. Neste caso, o sistema recompilaria automaticamente *Comp.java*, porque a versão compilada tem uma data de modificação anterior à versão mais recente do componente.

SOMAS DE VERIFICAÇÃO (CHECKSUM) DE CÓDIGO-FONTE

A assinatura no arquivo de código-fonte é uma soma de verificação (checksum) calculada a partir dos dados no arquivo, gerando um número exclusivo a partir do texto de origem como entrada, de forma que qualquer alteração no texto, até mesmo um único caractere, gera uma soma de verificação diferente, sendo essa soma de verificação atribuída ao código-fonte somente antes da compilação e identifica exclusivamente o arquivo de origem.

O sistema de construção, em seguida, marca o arquivo de código objeto gerado com a assinatura da soma de verificação, portanto, se não houver nenhum arquivo de código objeto com a mesma assinatura que o arquivo de código-fonte a ser incluído em um sistema, a recompilação do código-fonte é necessária.

Como os arquivos de código objeto normalmente não são versionados, a abordagem timestamps significa que apenas o arquivo de código objeto compilado mais recentemente é mantido no sistema. A abordagem de soma de verificação tem a vantagem de permitir que sejam mantidas simultaneamente muitas versões diferentes do código objeto de um componente. A assinatura em vez do nome do arquivo é o vínculo entre código-fonte e objeto, pois o código-fonte e os arquivos de código objeto têm a mesma assinatura.

RESUMINDO

Neste módulo, pudemos compreender o processo de construção de um sistema executável por meio da compilação e da ligação dos componentes do sistema, das bibliotecas externas, dos arquivos de configuração e de outras informações necessárias ao referido processo.

As ferramentas de construção de sistema e de controle de versões devem ser integradas, pois o processo de construção usa versões dos componentes do repositório gerenciado pelo sistema de controle de versões.

Entendemos que a integração contínua permite a descoberta e a alteração imediata dos problemas causados pelas interações entre diferentes desenvolvedores, sendo o sistema mais recente no mainline o sistema funcional definitivo.

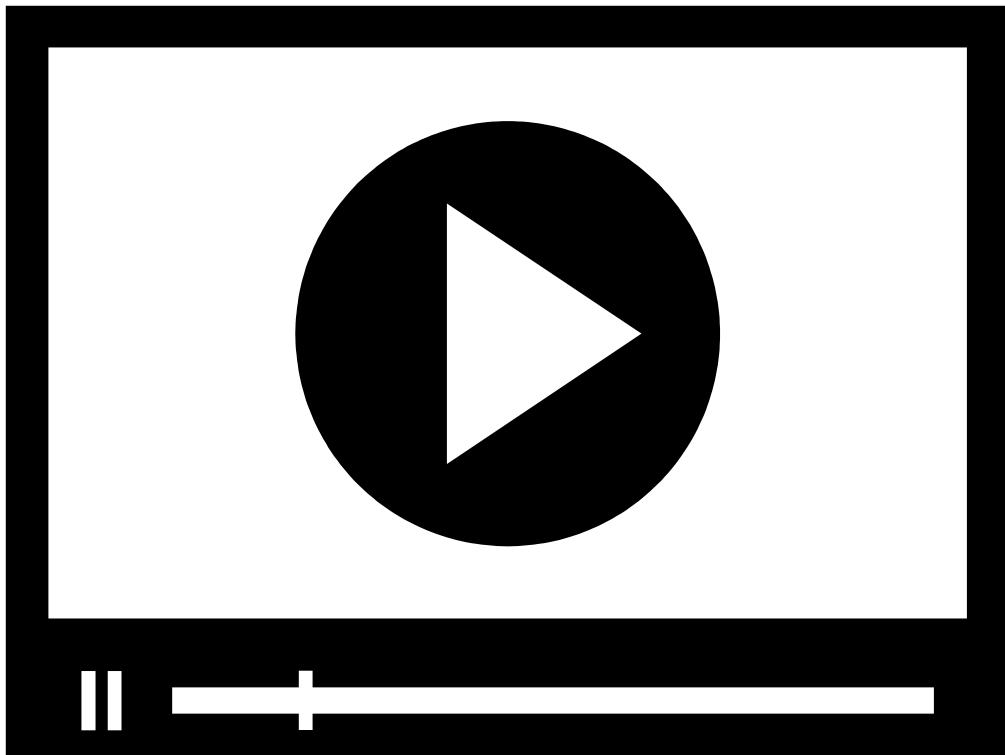
A compilação é um processo computacionalmente intensivo exigindo o gerenciamento da ligação do código-fonte de um componente ao seu código objeto equivalente, sendo esta ligação realizada por meio da designação de uma assinatura exclusiva a cada arquivo de componente de código-fonte armazenado, tendo o código objeto correspondente, que foi compilado a partir do código-fonte, uma assinatura relacionada.

Agora é com você! Vamos verificar se atingimos o objetivo deste módulo? Você está convidado a realizar as atividades a seguir.

VERIFICANDO O APRENDIZADO

MÓDULO 4

-
- **Descrever as ferramentas CASE para gerenciamento de configurações**



No vídeo a seguir, você compreenderá a importância do uso de ferramentas CASE para apoio ao gerenciamento de configurações.



AMBIENTES DE GERENCIAMENTO DE CONFIGURAÇÕES

Usualmente, os processos de gerenciamento de configurações são padronizados, ou seja, incluem aplicações de procedimentos predefinidos que requerem o gerenciamento de grande quantidade de dados.

Imaginemos um sistema que esteja sendo construído com base em versões de componentes e um determinado engenheiro de software realiza a publicação de uma versão com uma conexão de banco de dados X, sendo que ocorreu uma alteração de requisito não funcional para o

banco de dados Y, ou seja, a versão do componente da camada de persistência não estava corretamente gerenciada.

► ATENÇÃO

Com base nesse pequeno incidente, podemos concluir que o apoio de uma ferramenta **CASE** é essencial para o gerenciamento de configuração, podendo a mesma ser combinada para criar uma área de trabalho para apoiar todas as atividades de gerenciamento de configuração.

CASE

Ferramentas CASE (do inglês Computer-Aided Software Engineering) é uma classificação que abrange todas as ferramentas baseadas em computadores que auxiliam atividades de Engenharia de Software, desde análise de requisitos e modelagem até programação e testes.

Entendamos dois tipos de ambientes:

AMBIENTES ABERTOS

Incluem ferramentas de código aberto para cada estágio do processo que são integradas por meio de procedimentos organizacionais padronizados para uso dessas ferramentas, ou seja, o gerenciamento de mudança, o gerenciamento de versões e a construção de sistemas possuem ferramentas específicas que devem ser integradas.

AMBIENTES INTEGRADOS

Fornecem recursos integrados para controlar versões, a construção de sistemas e o rastreamento de mudanças.

Destacamos que sistemas complexos com equipes de desenvolvimento geograficamente distantes necessitam de ferramentas de gerenciamento de configuração que apoiem o trabalho em diferentes localidades com múltiplos repositórios de dados para itens de configuração.

APOIO PARA GERENCIAMENTO DE MUDANÇAS

Uma melhor prática sugere que um modelo de processo de mudanças deva ser projetado e integrado com um sistema de gerenciamento de versões, de modo que os documentos gerados nas atividades relacionadas com as mudanças sejam enviados para pessoas certas no tempo certo.

As ferramentas de gerenciamento de mudanças geralmente fornecem os seguintes recursos para dar suporte ao processo:

EDITOR DE FORMULÁRIOS

Permite às pessoas criar ou completar formulários com propostas de mudanças.

SISTEMA DE WORKFLOW

Permite à equipe de gerenciamento de configuração definir quem deve processar o formulário de solicitação de mudança e a ordem de processamento, sendo os formulários encaminhados automaticamente para as pessoas corretas no tempo certo, informando aos membros relevantes da equipe sobre o progresso da mudança.

BANCO DE DADOS DE MUDANÇA

Usado para gerenciar todas as propostas de mudança e que pode estar ligado a um sistema de gerenciamento de versões, no qual recursos de consulta permitem à equipe de gerenciamento de configuração encontrar uma proposta específica de mudança.

SISTEMA DE RELATO DE MUDANÇAS

Gera relatórios gerenciais sobre o status das solicitações de mudança enviadas.

APOIO PARA GERENCIAMENTO DE VERSÕES

As ferramentas de gerenciamento de versões controlam um repositório de itens de configuração no qual os conteúdos dos repositórios não podem ser alterados.

COMENTÁRIO

Para trabalhar num item de configuração, o engenheiro de software deve retirá-lo por meio de uma operação check-out do repositório e guardá-lo num diretório de trabalho, sendo devolvido ao repositório, após a realização das mudanças no componente, por meio de uma operação check-in, sendo uma nova versão automaticamente criada.

Os sistemas de gerenciamento de versões fornecem um conjunto básico de capacidades semelhantes, embora alguns recursos sejam mais sofisticados que outros.

Vejamos:

IDENTIFICAÇÃO DE VERSÕES E RELEASES

Versões gerenciadas recebem identificadores quando são enviadas ao sistema.

GERENCIAMENTO DE ARMAZENAMENTO

Um sistema de gerenciamento de versões fornece recursos de gerenciamento de armazenamento de maneira que as versões sejam descritas pela sua diferença em relação a uma versão mestre, sendo as diferenças entre versões representadas por um delta, que engloba as instruções necessárias para recriar a versão do sistema associado, tal como exemplifica a Figura 11, pois considerando que a última versão é a versão 1.3, a criação da versão 1.2 exige a aplicação do delta de mudança correspondente D3.

REGISTRO DO HISTÓRICO DE MUDANÇAS

Inclui o registro de todas as mudanças executadas no código de um sistema ou componente. Desenvolvimento independente – Múltiplas versões de um sistema podem ser desenvolvidas em paralelo, podendo cada versão ser modificada de forma independente, tal como um release 1 pode ser modificado depois de iniciado o desenvolvimento do release 2, pois o sistema de gerenciamento de versões mantém a rastreabilidade dos componentes retirados por meio do check-out para a edição.

SUPORTE A PROJETOS

Inclui o suporte a múltiplos projetos.

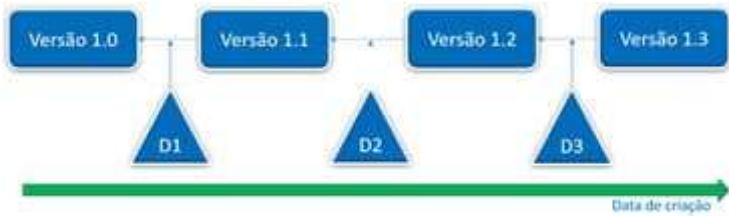


Figura 11 – Versões baseadas em delta. Fonte: O Autor.

SUporte para Construção de Sistemas

A construção de sistemas é um processo computacional intensivo, podendo incluir centenas de arquivos.

ATENÇÃO

As ferramentas de construção de sistemas automatizam o processo de construção para reduzir a possibilidade de ocorrência de erro humano e, quando possível, minimizar o tempo necessário para a construção de sistemas.

Vejamos alguns recursos que podem estar presentes em ferramentas CASE de construção de sistemas:

Uma linguagem de especificação de dependência e um interpretador associado, pois as dependências de componentes podem ser descritas e a recompilação minimizada.

Seleção de ferramentas e apoio à instanciação, que inclui compiladores e outras ferramentas de processamento usadas para processar os arquivos de código-fonte.

Supporte à compilação distribuída em redes, de modo que antes que as compilações sejam executadas em uma única máquina, o construtor de sistema procura por um processador ocioso na rede e aloca um número de compilações paralelas, reduzindo significativamente o tempo de construção do sistema.

Gerenciamento de objetos derivados liga os diferentes códigos-fontes, derivando novamente um objeto somente quando ele for requerido pelas mudanças do código-fonte.

Vamos entender esta dependência entre objetos derivados a partir da Figura 12, bem como a minimização da recompilação.

Considere uma situação em que um programa de compilador chamado *comp* é criado por quatro módulos objetos chamados *scan.o*, *syn.o*, *sem.o* e *cgen.o*, sendo cada módulo objeto criado dos respectivos códigos-fontes, ou seja, *scan.c*, *syn.c*, *sem.c* e *cgen.c*. Um arquivo de declarações de variáveis e de constantes chamado *defs.h* é compartilhado por *scan.c*, *syn.c* e *sem.c*. Na Figura 12, as setas significam 'depende de', ou seja, a entidade na base da seta depende da entidade que está no topo, de modo que *comp* depende de *scan.o*, *syn.o*, *sem.o* e *cgen.o*; *scan.o* depende de *scan.c*, e assim, sucessivamente.

O que você acha que ocorreria se *scan.c* fosse alterado?

Inicialmente, a ferramenta de construção de sistemas detecta que o objeto derivado *scan.o* deva ser recriado pela comparação dos momentos da modificação do *scan.o* e do *scan.c*, pois *scan.c* foi modificado depois de *scan.o*. Em seguida, a referida ferramenta chama o compilador C para compilar o *scan.c* a fim de criar um novo objeto derivado, ou seja, o *scan.o*. A ferramenta de construção então usa a ligação de dependência entre *comp* e *scan.o* para detectar que *comp* deve ser recriado pela ligação de *scan.o*, *syn.o*, *sem.o* e *cgen.o*.

● COMENTÁRIO

A partir deste pequeno exemplo, podemos imaginar a complexidade de gerenciamento de diversos arquivos e respectivas dependências, ou seja, a automação do gerenciamento de versões é fortemente recomendada.

Muitas ferramentas de construção de sistemas usam a data de modificação de arquivo como um atributo-chave para decidir se a recompilação é necessária, de modo que, se um arquivo de código-fonte é modificado depois do seu correspondente arquivo de código objeto, o arquivo de código objeto é recriado.

Algumas ferramentas mantêm o registro da última versão do código objeto que corresponde ao mais recente componente de código-fonte modificado, outras ferramentas, entretanto, usam abordagens mais sofisticadas para o gerenciamento de objetos derivados, colocando uma marca nos objetos derivados com a identificação dos códigos-fontes usados para gerar os objetos, o que permite manter todos os objetos derivados.

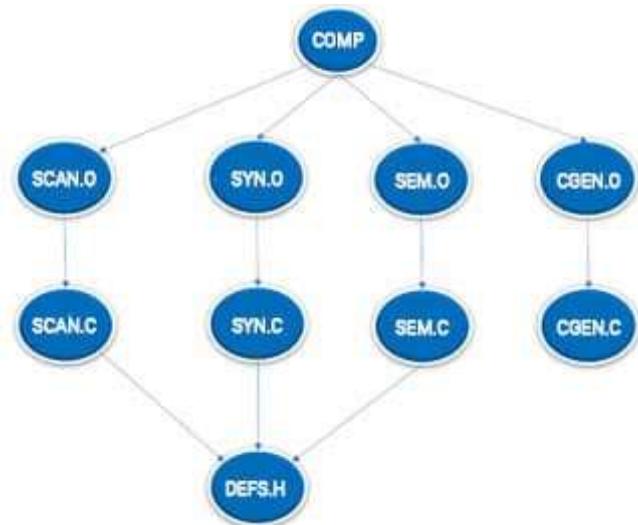


Figura 12 – Dependências entre componentes. Fonte: O Autor.

FERRAMENTAS CASE PARA GERENCIAMENTO DE CONFIGURAÇÃO

As ferramentas CASE permitem que as atividades fundamentais de gerenciamento de configuração resultem em ganhos de produtividade no gerenciamento e controle de itens de configuração gerados ao longo do processo de desenvolvimento de software.

De acordo com Andrade (2014), as ferramentas CASE podem ser divididas em três grupos principais ilustrados na Figura 13:

FERRAMENTAS DE SUPORTE INDIVIDUAL

Tratam o controle de versões, o controle de mudança e a integração contínua de itens.

FERRAMENTAS DE SUPORTE AO PROJETO

Suportam o projeto e apoiam a integração das atividades da gerência de configuração.

FERRAMENTAS DE SUPORTE COMPLETO À ORGANIZAÇÃO

Apoiam o processo da empresa e a gerência de configuração atrelada ao processo.



Figura 13 - Principais grupos de ferramentas. Fonte: O Autor.

Vejamos as ferramentas de suporte individual:

Controle de versões permite a identificação, armazenamento e gerenciamento dos itens de configuração.

Controle de mudança tem como foco os procedimentos pelos quais as mudanças de um ou mais itens de configuração são propostas, avaliadas, aprovadas e implantadas.

Integração contínua permite a identificação, empacotamento e preparação de uma baseline, garantindo que as mudanças sejam construídas, testadas e relatadas.

A tabela a seguir apresenta algumas ferramentas, divididas pelos grupos:

Tipo de ferramenta	Ferramentas open source	Ferramentas comerciais
Controle de versão	CVS Subversion Aegis	PVC ClearCase StarTeam

	Arch JEDI	Perforce BitKeeper Synergy Serena Microsoft Visual Source Safe
Controle de mudança	Bugzilla Trac Mantis Scarab	PVCS Jira FogBugz CaliberRM ClearQuest Perforce Synergy Serena
Integração contínua	Ant SCons Bitten Maven CruiseControl Gump TinderBox	Build Forge Anthill Pro FinalBuilder

Atenção! Para visualização completa da tabela utilize a rolagem horizontal

RESUMINDO

Neste módulo, destacamos a importância dos processos de gerenciamento de configurações, que incluem aplicações de procedimentos predefinidos que requerem o gerenciamento de grande quantidade de dados.

Um modelo de processo de mudanças deve ser projetado e integrado com um sistema de gerenciamento de versões, de modo que os documentos gerados nas atividades relacionadas com as mudanças sejam enviados para pessoas certas no tempo certo.

As ferramentas de gerenciamento de versões controlam um repositório de itens de configuração no qual os conteúdos dos repositórios não podem ser alterados, de modo que um item de configuração é retirado do repositório por meio de uma operação check-out, sendo devolvido ao repositório, após a realização das mudanças no referido item, por meio de uma operação check-in, sendo uma nova versão automaticamente criada.

As ferramentas de construção de sistemas automatizam o processo de construção para reduzir a possibilidade de ocorrência de erro humano e, quando possível, minimizar o tempo necessário para a construção de sistemas.

As ferramentas CASE permitem que as atividades fundamentais de gerenciamento de configurações resultem em ganhos de produtividade no gerenciamento e controle de itens de configuração gerados ao longo do processo de desenvolvimento de software.

Agora é com você! Vamos verificar se atingimos o objetivo deste módulo? Você está convidado a realizar as atividades a seguir.

VERIFICANDO O APRENDIZADO

CONCLUSÃO

CONSIDERAÇÕES FINAIS

Apresentamos neste tema o processo de gerenciamento de configurações de software, que inclui um conjunto de atividades destinadas a gerenciar alterações, identificando os artefatos que precisam ser alterados, estabelecendo relações entre eles, definindo mecanismos para gerenciar diferentes versões destes artefatos, controlando as alterações impostas e auditando e relatando as alterações feitas.

Ressaltamos que o gerenciamento de mudanças permite que as alterações sejam aplicadas ao sistema de uma maneira controlada, ou seja, garante que a evolução do sistema seja controlada e que as alterações mais urgentes e com bom custo-benefício sejam priorizadas, cabendo ao gerenciamento de versões controlar as diferentes versões de componentes de software e dos sistemas, envolvendo também a garantia de que as alterações feitas por diferentes desenvolvedores não interfiram umas com as outras.

O processo de construção de sistema executável ocorre por meio da compilação e da ligação dos componentes do sistema, das bibliotecas externas, dos arquivos de configuração e de outras informações necessárias ao referido processo.

Demos destaque às ferramentas CASE no contexto do gerenciamento de configurações, pois elas permitem que as atividades fundamentais desse gerenciamento resultem em ganhos de produtividade no gerenciamento e controle de itens de configuração gerados ao longo do processo de desenvolvimento de software.



PODCAST

⚠️ PODCAST

REFERÊNCIAS

ANDRADE, J. **Gerência de Configuração**. 1. ed. São Paulo: Pearson Education do Brasil, 2014.

PRESSMAN, R. S.; MAXIM, B. R. **Engenharia de Software**. 8. ed. Porto Alegre: AMGH, 2016.

SOMMERVILLE, I. **Engenharia de Software**. 10. ed. São Paulo: Pearson Prentice Hall, 2019.

EXPLORE+

Para saber mais sobre os assuntos tratados neste tema, leia:

A referência Pressman (2016): Capítulo 29.

A referência Sommerville (2019): Capítulo 25.

A referência Andrade (2014): Unidade 4.

CONTEUDISTA

Alberto Tavares da Silva

 CURRÍCULO LATTES