Gestão de configuração de software

Gerência de configuração de software como uma área da engenharia responsável por apoiar o desenvolvimento de software. Gerenciamento de configuração como a operação realizada para manter os sistemas conforme o desejado e de forma consistente.

Prof. Brunelli Gabriel Cupello

1. Itens iniciais

Propósito

Entender como deve ser a gestão da configuração de software é essencial aos profissionais de Tecnologia da Informação (TI), a fim de que façam as melhores escolhas para o seu produto.

Objetivos

- Reconhecer a necessidade de gerenciamento de diferentes versões de um produto.
- Identificar os conceitos básicos de gerência de software.
- Reconhecer a gerência de software na prática.
- Identificar as principais ferramentas automatizadas de gerência de software.

Introdução

O gerenciamento de configuração de software é usado para gerir de forma constante um software. Tal ferramenta possibilita ter certeza de que o sistema funciona da forma que deveria e de que as mudanças que precisarão ser feitas ao longo do tempo serão realizadas no intervalo de tempo esperado.

Para aplicar o gerenciamento das configurações de um sistema de Tecnologia da Informação (TI), é fundamental identificar o que se pretende controlar, definir o controle a ser estabelecido, verificando se as modificações estão de acordo com o especificado e assegurando um registro que reconheça a obtenção de informações do estado em que se encontram os itens sob controle.

O gerenciamento de configuração de software apresenta as seguintes funções básicas: identificação de configuração, controle de configuração, auditoria de configuração e administração de estados.

Na **identificação de configuração**, os elementos são os itens de configuração, ou seja, todos os entes capazes de serem manipulados, identificados de forma única e, então, gerenciados. São considerados itens de configuração: documentos, código, tabelas de parâmetros, estruturas de arquivos etc. Eles formam um conjunto de objetos hierarquizados e inter-relacionados.

Por **controle de configuração**, entende-se que um ou mais itens de configuração estão em conformidade com os requisitos do projeto e devem ser protegidos de alterações desautorizadas. Após ter sido estabelecida uma configuração básica, quaisquer alterações nos itens devem ser controladas fielmente, lançando mão de procedimentos humanos e ferramentas automatizadas.

Na **auditoria de configuração**, um grupo investiga determinado produto, mas não é necessário que esse próprio grupo o tenha desenvolvido. Essa auditoria deve ser realizada antes que uma configuração básica seja definida e a cada alteração feita no produto a ser entregue, a fim de viabilizar o gerenciamento do software ao longo de seu ciclo de vida. Com a auditoria, afirma-se ao usuário que o produto está de acordo com as especificações contratuais e os documentos técnicos.

A **administração de estados** é um processo para a obtenção de informações sobre o estado das configurações-base. A obtenção dessas informações deve ser iniciada assim que houver a identificação dos itens de configuração. Esse processo permite localizar as modificações ocorridas durante o desenvolvimento do software, identificando o autor da modificação, bem como o tipo de modificação e o momento em que foi efetuada.

É de suma importância inserir o processo de gerenciamento de configuração nos softwares, visto que ele possibilita visualizar toda e qualquer alteração que seja feita no sistema, desde as menores até as maiores. E, caso alguma alteração não esteja documentada, o processo não pode ser concluído. Dessa forma, evita-se: incidentes de segurança nos ambientes em containers ou orquestrados pelo kubernetes (plataforma de código aberto, portável e extensiva para o gerenciamento de cargas de trabalho e serviços distribuídos em contêineres), baixo desempenho, instabilidade, downtime e inconsistências ou não conformidades, capazes de tornar inseguro um software que lida com operações de negócios.

As avaliações de funcionalidade do produto são necessárias, para que as configurações não saiam do padrão esperado de atendimento das necessidades. Além disso, um gerenciamento eficiente utiliza os feedbacks

para programar melhor suas ações, quer sejam de atualização, identificação de problemas, reconfiguração ou aplicação de patches, de uma forma bem simples, possibilitando qualquer coisa que o sistema precise.

Além disso, o gerenciamento permite que os desenvolvedores de software estabeleçam as configurações, criem e mantenham esses sistemas de acordo com as configurações básicas, ajudando usuários e administradores no que tange a determinados serviços, em especial os estados das aplicações no momento desejado.

Cabe ressaltar ainda que há uma diferença entre manutenção de software e gerência de configuração de software. Enquanto a primeira é desenvolvida após a entrega do software e envolve a engenharia de software, a última envolve atividades de controle e de acompanhamento, que são realizadas durante o ciclo de produção do software.

No vídeo a seguir, destacamos a importância do processo de gestão e configuração de software e seu papel no processo de desenvolvimento e operação, além de ressaltar a importância do conhecimento desses conceitos para o profissional de TI.

Introdução

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

O que é controle de versão?

Neste vídeo, falaremos sobre diferentes sistemas: baseados em regras, indicadores, algoritmos de otimização e aprendizado de máquina.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Controle de versão são procedimentos e ferramentas cuja principal finalidade é coordenar versões dos arquivos criados durante o desenvolvimento do software. Ele possibilita de maneira muito eficiente e inteligente a organização do projeto, tornando possível o acompanhamento histórico do que é produzido e desenvolvido enquanto o registro acontece.

O controle de versão ainda fornece algumas vantagens, como a personalização de uma versão, incluindo detalhes para corrigir erros ou gerar melhorias, layout, entre outros detalhes. Tudo isso sem mexer no projeto principal ou voltar para um ponto do projeto que não tinha bugs ou problemas gerais, ou seja, mais estável no que tange ao funcionamento. Parece ser a solução padrão-ouro para o desenvolvedor: usar um sistema de controle de versão que acabe com quaisquer tipos de dúvida e inconsistência no sistema que poderiam ser geradas por falta de organização.

Atenção

Também chamado “controle de fonte”, o controle de versão é imprescindível na produção just in time, quando são necessárias mudanças constantes e gerenciamento do código-fonte no decorrer do tempo. Além disso, por manter um registro de todas as alterações realizadas, a correção é mais certeira e precisa.

Com o controle de versão, é possível identificar, armazenar e gerenciar tanto os itens de configuração como suas versões ao longo do ciclo de vida do software, avaliar o histórico de alterações de configurações já realizadas e recuperar configurações perdidas ou deixadas de lado por algum motivo. Os objetivos do controle de versão incluem automatização de rastreio de arquivos, recuperação de versões anteriores, desenvolvimento de configurações em paralelo, redução do espaço de armazenamento gasto, entre outros.

No controle de versão, os arquivos ficam guardados no servidor formando um histórico das versões, que podem ser usadas para consultas ou para voltar a uma versão mais estável em caso de problemas, o que possibilita realizar o trabalho em uma versão mais segura. Recomenda-se que a cada mudança realizada no código se atualize a versão no servidor. Observe:

Esquema de funcionamento dos controles de versão

Às vezes os desenvolvedores editam o arquivo que fica no servidor. Se isso acontecer, o que pode ser feito? Para evitar problemas como esse, o sistema de controle de versão oferece ferramentas úteis para misturar o código e evitar problemas.

Exemplo

Se um desenvolvedor atualizou o projeto (usando a função chamada de check-out ou update) fazendo as alterações que julgou necessárias e, ao mesmo tempo, outro desenvolvedor fez outras modificações e atualizou a versão do servidor. Quando a versão do primeiro desenvolvedor (usando uma função chamada de check-in ou commit) for enviada, o sistema de controle de versão avisará que o arquivo está desatualizado.

O mesmo sistema enviará as novas informações que foram adicionadas e dará a opção de mesclar as diferentes versões. Não só isso, o sistema também mostrará onde as atualizações foram realizadas e os trechos dos códigos incluídos ou removidos. Em caso de conflitos, onde as linhas escritas se sobrepuseram às outras e dá a opção de mesclar manualmente, assim escolhendo a melhor solução.

Tipos de controle de versão

Neste vídeo, apresentaremos o controle centralizado e distribuído de versão de software, seus principais passos e métodos.

Conteúdo interativo

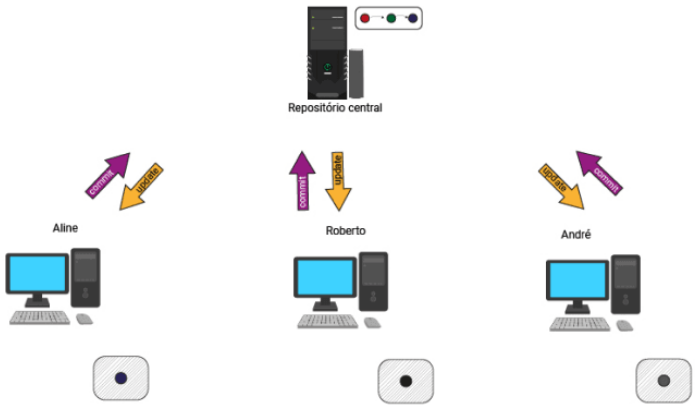
Acesse a versão digital para assistir ao vídeo.

Classificação do sistema de controle de versão

Atualmente, o sistema de controle de versão é classificado em dois tipos: centralizado e distribuído. O **sistema de controle centralizado** realiza sua função com um servidor central e diversas áreas de trabalho baseando-se na arquitetura cliente-servidor. Por ser centralizado, as áreas de trabalho necessitam passar primeiramente pelo servidor para que haja comunicação.

Esse tipo de sistema atende muito bem equipes de desenvolvedores que não sejam muito grandes ou que trabalhem em uma rede local. Além disso, não apresenta problemas de velocidade para mandar e receber os

dados e tem um tempo rápido de resposta do servidor. Um dos principais sistemas de controle de versão do tipo centralizado é o Subversion. Veja na imagem a seguir:



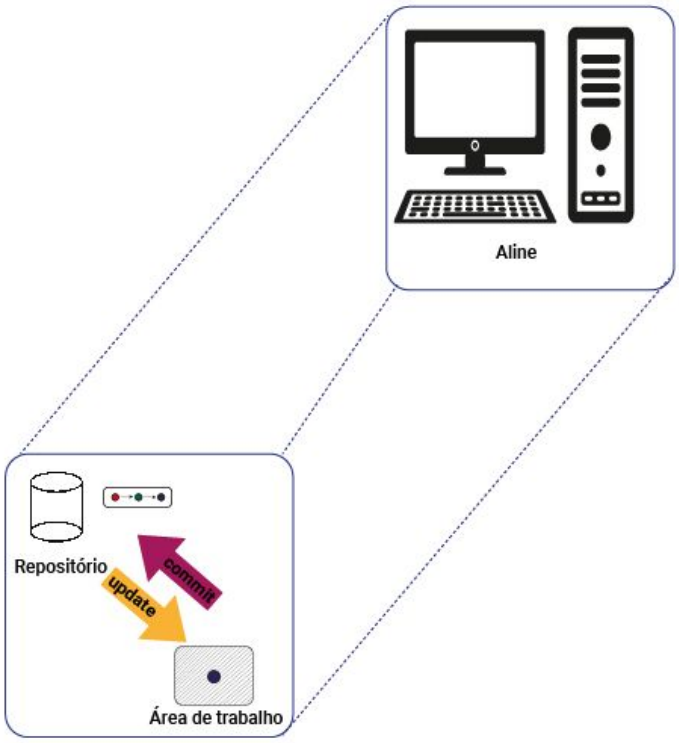
Modelo de controle de versão centralizado

O sistema de controle distribuído vai mais longe, ele é ideal para times com um grande número de desenvolvedores que estejam em diferentes localidades.

> No sistema de controle distribuído, cada parte do trabalho tem um servidor específico, assim, as operações são feitas automaticamente pela própria máquina. A comunicação entre o servidor principal e as áreas de trabalho ocorre por meio das operações de push e pull.

O pull traz a versão de outra área de trabalho e mescla com a sua, já o push faz a operação inversa. Com essas operações é possível atualizar e mesclar o projeto. No entanto, para que as áreas de trabalho se comuniquem entre si, precisam usar um único servidor que centralize o envio dos arquivos. Assim, evita-se perda de controle sobre o projeto.

O sistema de controle distribuído ganha do centralizado em velocidade, uma vez que o processo ocorre em uma mesma máquina. No entanto, exige que o desenvolvedor possua maior conhecimento da ferramenta. Observe:



Modelo de controle de versão distribuído

Sincronização de mudanças concorrentes

O controle de versão facilita o trabalho paralelo e concorrente de alguns desenvolvedores sobre os mesmos arquivos, de forma que evita que um sobrescreva o código de outro, fugindo do aparecimento de defeitos e da perda de funcionalidades.

No entanto, só a área de trabalho não é capaz de dar conta de todo o problema. É indispensável uma forma de sincronizar os esforços de todos os membros da equipe. Um modo de resolver isso é a sincronização pela combinação de revisões concorrentes em uma única resultante. Essa operação é chamada de merge (mesclagem) e pode ser aplicada tanto no controle de versão centralizado como no controle de versão distribuído.

Na sincronização do controle de versão centralizado, o repositório cria duas cópias de trabalho, que iniciam a partir do mesmo estado para dois desenvolvedores pelo comando check-out. Cada um desses desenvolvedores pode fazer suas alterações em suas cópias de trabalho. No entanto, se um deles envia mais rapidamente ao repositório, o desenvolvedor mais lento, ao tentar enviar sua versão com as atualizações, recebe uma recusa do repositório, visto que as alterações foram baseadas em arquivos desatualizados. Então, o repositório envia ao desenvolvedor mais lento, por meio do check-out, a versão que o primeiro desenvolvedor atualizou e o controle de versão já mescla automaticamente as revisões.

Na sequência, esse último desenvolvedor observa se as atualizações e a mesclagem surtiram os efeitos esperados e envia as mudanças ao repositório. Enquanto isso acontecia, o primeiro desenvolvedor já trabalhava em outras tarefas. O commit do primeiro desenvolvedor será aceito se nenhuma das revisões que vieram depois da atualização da cópia de trabalho tiver alterado os mesmos arquivos. É uma situação possível de acontecer, mas não é comum.

Outro ponto importante é a função lock, que bloqueia o arquivo para que não seja modificado por outros enquanto estiver com um dos desenvolvedores. Os sistemas distribuídos mais conhecidos são o Git e o Mercurial.

Na sincronização do **controle de versão distribuído**, um desenvolvedor clona o repositório do outro e ambos partem do mesmo ponto. Então, cada um publica suas alterações em seu próprio repositório, sem interferir no do outro. Um desses desenvolvedores sincroniza seu repositório com as revisões publicadas do outro, sem afetar a área de trabalho alheia.

Resumindo

A mesclagem entre as revisões dos desenvolvedores é feita explicitamente na área de trabalho do primeiro por meio de um comando merge. Enquanto isso, o outro desenvolvedor gera nova revisão no seu repositório. Então, ambos fazem a conferência, vendo se a mesclagem gerou o resultado esperado, e um deles já pode publicar mais uma vez no seu repositório. Por último, um envia suas revisões ao repositório do outro, que as combina com o histórico de revisões já existentes.

Verificando o aprendizado

Questão 1

No desenvolvimento de sistemas, o controle de versão do código-fonte é uma atividade essencial para o sucesso. O que é controle de versão?

A

São procedimentos e ferramentas cuja principal finalidade é coordenar versões dos arquivos.

B

São controles desenvolvidos pelos desenvolvedores durante o ciclo de vida do projeto para apagá-lo em caso de erros irreversíveis.

C

São ferramentas para unir versões de softwares anteriores e mantê-los no histórico de criações separadamente.

D

São versões capazes de transformar o software e usadas por apenas um desenvolvedor.

E

São ferramentas antiquadas que já não são usadas em DevOps.

A alternativa A está correta.

O controle de versão são procedimentos e ferramentas cuja principal finalidade é coordenar versões dos arquivos criados durante o desenvolvimento do software. Ele possibilita de maneira muito eficiente e inteligente a organização do projeto, porque torna possível o acompanhamento de um histórico do que é produzido e do que é desenvolvido enquanto o registro acontece.

Questão 2

O controle de versão é essencial em qualquer processo de desenvolvimento de software e, para tal, são utilizados sistemas automatizados. Como funcionam os sistemas de controle de versão?

A

Os arquivos se deslocam entre os servidores e em cada um há um histórico de versões inalteráveis.

B

Os arquivos ficam guardados no servidor e nele fica um histórico das versões, que pode ser usado para consultas ou para voltar para uma versão mais estável em caso de problemas.

C

Os arquivos voltam para versões mais estáveis de forma aleatória.

D

Os arquivos deslocam-se e tornam-se mais estáveis de forma aleatória e, após esse processo, são apagados.

E

Os arquivos não podem ser alterados, mantendo-se estáveis para que sejam analisados pelo controle de versões.

A alternativa B está correta.

Os arquivos ficam guardados no servidor e nele fica um histórico das versões, que pode ser usado para consultas ou para voltar a uma versão mais estável em caso de problemas. Assim, o desenvolvedor pode realizar seu trabalho em uma versão mais segura. Recomenda-se que, a cada mudança realizada no código, seja atualizada a versão no servidor.

Dependências de software

Neste vídeo, apresentaremos o conceito de dependência de software e como agir para resolver eventuais problemas.

> **Conteúdo interativo**
>
> Acesse a versão digital para assistir ao vídeo.

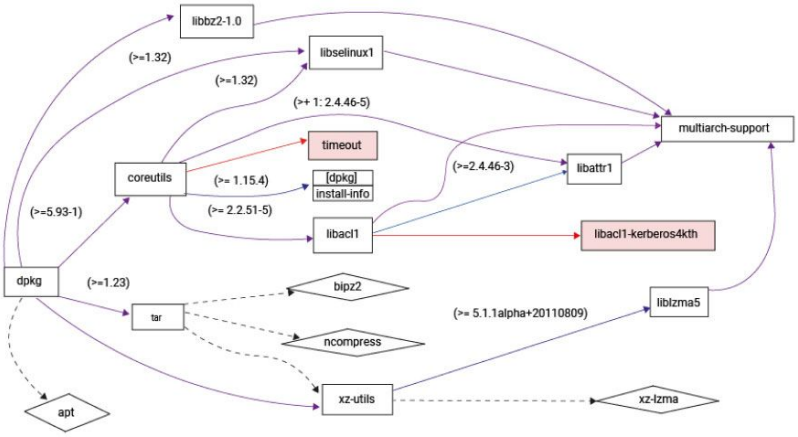
Cerca de duas décadas atrás, durante o processo de desenvolvimento de aplicações, era necessário acessar um banco de dados para que, então, o desenvolvedor fizesse o “copy-paste” (basicamente, o “copia e cola”) de uma parte do desenvolvimento de outra aplicação que se parecia em parte com a que desejava realizar.

No entanto, com a evolução do desenvolvimento das aplicações, o “copy-paste” de um código passou a ser o “CTRL C + CTRL V” de um pacote de códigos. Este era introduzido manualmente ao projeto em desenvolvimento, causando muitas vezes problemas como a não alteração do pacote copiado ou até seu mau funcionamento na aplicação em desenvolvimento.

O gerenciamento manual se tornou impraticável e surgiram os **gerenciadores de pacotes**, ferramentas que realizam automaticamente os procedimentos antes feitos manualmente. Dessa forma, otimiza-se o tempo do desenvolvedor, que pode focar no que é mais importante: o desenvolvimento do projeto que está criando com menos chances de erros.

Ainda existe a possibilidade de se fazer o “copia e cola” de pacotes de códigos, no entanto, na maioria das vezes eles são dependentes uns dos outros, gerando um gráfico de dependências – o que não é ruim, só causa problemas ou se torna impossível quando manuseados manualmente. E caso o desenvolvedor adicione apenas um pacote manualmente, a falta dessa outra parte do pacote pode invalidá-lo ou fazê-lo funcionar de forma limitada, causando reprocesso.

Por essa razão se diz que, independentemente do tamanho do projeto, é necessária a utilização do gerenciador de dependências. Ele administra todas as ações do projeto no que tange à listagem, adição, remoção, atualização, análise do grafo de dependências e garantia de obtenção das dependências dos pacotes. Observe:



Grafo de dependências

Principais gerenciadores de dependência

Gerenciadores de dependência

Neste vídeo, apresentaremos os principais gerenciadores de dependência de software, suas características e usos mais comuns.

> Conteúdo interativo
>
> Acesse a versão digital para assistir ao vídeo.

Gerenciador de dependências é uma plataforma automatizada (software) que ajuda o desenvolvedor no uso de bibliotecas. É comum que bibliotecas dependam de outras para funcionar. É aí que entra o gerenciador. Segue a lista das plataformas de desenvolvimento com as principais ferramentas de criação, compartilhamento e consumo de códigos para o intercâmbio entre os desenvolvedores.

1) NPM (Node Package Manager) e Yarn

Funcionam em aplicações Web e JavaScript.
A seguir, enumeramos o que o NPM é capaz de fazer:

- Cria ferramentas executadas do terminal com JavaScript.
- Abre o site e o repositório do projeto e procura por pacotes não enunciados no package.json.
- Permite ao desenvolvedor criar um pacote de forma rápida por meio de perguntas lançadas e das respostas dadas pelo desenvolvedor.
- Analisa as dependências que estão desatualizadas e trava as versões de suas dependências, se necessário.
- Salva as dependências, instala a produção (mesmo que o desenvolvedor não vá mais fazer alterações no projeto, ignorando as dependências de desenvolvimento) e lista os pacotes instalados.

No entanto, o NPM é considerado lento e os desenvolvedores têm reclamado disso, principalmente os que trabalham com aplicações SPA, utilizando: Ember.js com Ember-CLI ou AngularJS, React etc. Por esse problema, surgiu o Yarn. No entanto, ele não veio para tornar o NPM obsoleto. Muito ao contrário: os registros do NPM são compatíveis com os do Yarn, tanto que, caso o desenvolvedor esteja usando o package.json, não será necessário alterá-lo.

Além disso, o Yarn é capaz de instalar um pacote usado anteriormente sem necessidade de conexão com a Internet e as dependências serão instaladas na mesma ordem que foram colocadas anteriormente. Além disso, se uma das requisições falhar, ela será refeita, pois as requisições são enfileiradas.

2) NuGet

Pode ser usado em aplicações.NET, ele é um gerenciador voltado para aplicações da plataforma.NET, que mantém seu repositório público e é passível de publicações, pesquisas e consumo de seus pacotes. Com essa plataforma, é possível desenvolver uma biblioteca com arquivos no formato DLL (código compilado) ou projetos de uma biblioteca (arquivo compactado), e eles serão alocados em repositórios públicos e/ou privados e então usados em demais projetos.

O NuGet está disponível em diversas versões. Portanto, é necessário que o desenvolvedor estabeleça qual versão da plataforma será utilizada em razão da compatibilidade com o pacote que está criando. Esse gerenciador analisa se o pacote do desenvolvedor é compatível com a versão definida do projeto-destino e faz o mesmo com as dependências do pacote. Dessa forma, evita-se que o pacote seja obtido mais de uma vez.

Resumindo

NuGet é uma plataforma que cria códigos para serem usados em outros projetos, por não ser um projeto executável. No entanto, permite a criação de repositórios privados ou locais e pode ser comparado com um elo entre os criadores dos pacotes e os desenvolvedores que os consomem.

3) Pip

Pode ser usado em aplicações Python, ele possui uma página de busca de pacotes disponíveis para atualização e é usado para:

- instalar;
- remover;
- atualizar pacotes no projeto em desenvolvimento.

Para usá-lo, é necessário baixar, salvar e executar no repositório o arquivo get-pip.py. A execução desse arquivo precisa ser feita pelo terminal ou cmd: python get-pip.py. O Pip está disponível para instalação, remoção, listagem e atualização de pacotes.

4) Maven e Gradle

O Maven funciona em aplicações Java e Kotlin, é um gerenciador de build e dependências. Para ser usado, precisa ser baixado em arquivo compactado e extraído no diretório de preferência do desenvolvedor. Durante as configurações das dependências, permite ao desenvolvedor configurá-las transmitindo um arquivo chamado pom.xml. Esse arquivo é o coração do MAven, pois é ele que mostra tudo o que será executado durante o build.

> O Maven é responsável por caçar os arquivos jars e colocá-los no classpath do projeto, além de gerenciar procedimentos que serão efetuados durante o build da aplicação.

Para executar o build, o desenvolvedor precisa dizer ao Maven as fases ou uma ordem predefinida do que será executado. No entanto, se necessário, é possível adicionar fases por meio de plugins.

Já o Gradle funciona em Java, Kotlin, além de Groovy, Android, Scala e JavaScript, uma vez que:

- Assegura ferramenta completa de gerenciamento e automatização de builds, aborda linguagem Groovy (linguagem de programação) nos scripts de build e elimina o uso do XMLs.
- Reconhece que tarefas sejam programadas com a criação de funções, loops e ifs, tendo modo de declaração parecida com a do Maven.
- Possibilita a invocação de scripts Ant.

Além disso, o Gradle é fundamental para a criação de scripts de build mais complexos e já é usado por projetos como o SDK do Android e o Hibernate, o que deixa claro sua confiabilidade e qualidade.

5) Composer e PEAR Installer

Ambos funcionam em aplicações PHP. No entanto, o Composer é o gerenciador de dependências mais usado do PHP. Foi lançado em 2012 e inspirado no NPM. Segundo o Treinaweb (s.d.): "... ele possui recursos de carregamento automático para bibliotecas para facilitar o uso de código de terceiros, tornando o desenvolvimento do projeto mais simples e permitindo que o desenvolvedor se concentre apenas em seu código, não no uso de código de terceiros".

O Composer permite que o desenvolvedor estabeleça que dependências farão seu projeto funcionar: o gerenciador faz a instalação, a atualização e a remoção, tirando essas responsabilidades do desenvolvedor. Além disso, de acordo como o Treinaweb (s.d.), o Composer possui outras funcionalidades, como:

> Instalação de novas dependências no projeto, atualização das dependências já instaladas no projeto, remoção de dependências instaladas no projeto, autoload para os arquivos do projeto, além dos pacotes de terceiro, execução de scripts, plugins para estender o comportamento padrão, permite determinar dependências que serão instaladas no modo de desenvolvimento ou em modo de produção.
>
> —
>
> (ANDRADE, 2022.)

Por outro lado, o PEAR Installer (PHP Extension and Application Repository) oferece a solução para alguns problemas, como geração de PDFs, imagens, manipulação de ID3 de MP3 etc. Também possui o repositório Pecl (PHP Extension Community Library), que instala extensões compiladas do PHP. Além disso, o PEAR – que usa o PHP – em sua 5ª versão escrito em linguagem C , capaz de escrever e/ou utilizar código de alta performance para scripts em PHP. O PHP não tem recursos como ponteiros de C, por isso, algumas operações podem ter velocidade reduzida.

6) RubyGems e Mixplorer

O RubyGems funciona em aplicações Ruby, é um gerenciador de pacotes avançado e flexível do Ruby. E Gem é a sua biblioteca autossuficiente de código reutilizável, projetada para gerenciar a instalação de gems e um servidor para distribuí-los. É usado pelo Facebook, Microsoft, Amazon, Instagram, Google, Slack, Mozilla, Discord, Elastic, Intuit, Reddit e muitas outras empresas conhecidas.

De acordo com Edvaldo Brito (2022), devido a problemas ocorridos na RubyGems, no que tange à aquisição de pacotes não autorizados, hoje se pesquisa a vulnerabilidade dentro da plataforma e, para isso, algumas condições precisam ser atendidas: "A Gem em direcionamento deve ter um ou mais traços no nome; a palavra antes do primeiro traço representa Gem controlada pelo atacante em RubyGems.org.; a Gem sendo retirada ou alterada deve ter sido criada há 30 dias ou não foi atualizada há mais de 100 dias".

O **MiXplorer** funciona em aplicações Elixir, é uma ferramenta de criação, compilação e testagem de projetos Elixir. Sua base é um projeto colocado em arquivo específico. Após a definição do projeto, tarefas-padrão do Mix podem ser executadas diretamente na linha de comando. Essas tarefas têm suas funções e configurações específicas. Para iniciar o primeiro projeto, é interessante escrever o seguinte comando: mix new my_project.

É por meio dos projetos que o MiX se torna extensível. Mas para usar dependências, é necessário adicionar um comando na configuração do projeto.

Ademais, o Mix suporta diferentes ambientes, que permitem aos desenvolvedores a preparação e a organização de seu projeto em diferentes cenários. No Mix, os ambientes são:

“:dev-

O ambiente padrão.

“:test-”

o ambiente “mix test”.

“:prod-”

O ambiente de execução das dependências.

7) Hex

Pode ser usado em aplicações Erlang. Como para cada linguagem temos um gerenciador de dependências, para o ecossistema Erlang não é diferente: temos o gerenciador Hex. Por meio do site hex.pm é possível pesquisar pacotes, entrando em “Packages”, onde há uma lista de pacotes disponíveis para download.

Além disso, nesse site é possível, com o uso de palavras-chave, achar frameworks (que servem para o desenvolvimento de softwares) e dependências (é possível visualizar como usar essa dependência). Dessa forma, é possível conferir o código, ver tudo o que está disponível no repositório dele próprio, além de documentar diretamente no código tudo o que é escrito – tudo pode ser usado novamente no futuro em outras aplicações.

8) Cargo

Cargo é o gerenciador de pacotes-padrão do Hust e possui documentação dentro do Ferrugem. É disponibilizado nos idiomas espanhol e inglês. O Cargo é usado para download de dependências dos pacotes Rust e compilação, dessa forma eles se tornam distribuíveis e facilitam o upload para o Craters, que é o registro de pacotes da comunidade Rust.

9) CPAN

Acrônimo para Comprehensive Perl Archive Network (rede de repositórios em linguagem de programação Perl), a CPAN é usada para aplicações PEAR e possui mais de 130 mil módulos de softwares. A sigla CPAN é usada tanto para designar a própria rede de armazenamento como o programa Perl, pois este pode ser utilizado para instalar softwares automaticamente – um gerenciador de pacotes – ou como interface para a rede. Além disso, o Perl possui mecanismos para que o programador consiga usar bibliotecas externas ao código.

Verificando o aprendizado

Questão 1

No desenvolvimento de aplicações, é muito comum o uso de bibliotecas e suas dependências. Nesse contexto, o que é o gerenciamento de dependências?

A

Um copia e cola de banco de dados.

B

Parte do desenvolvimento de outra aplicação.

C

A revolução do gerenciamento manual, ferramentas que realizam automaticamente os procedimentos antes feitos manualmente.

D

Retorno ao copy-paste de arquivos.

E

Desenvolve arquivos dependentes entre eles.

A alternativa C está correta.

Gerenciamento de dependências são ferramentas que realizam automaticamente os procedimentos antes feitos manualmente. Dessa forma, otimiza-se o tempo do desenvolvedor, que pode se concentrar no que é mais importante: o desenvolvimento do projeto com menores chances de erros.

Questão 2

NPM é um gerenciador de dependências. Em quais aplicações o NPM (Node Package Manager) funciona ?

A

Yarn

B

Net

C

Python

D

Web e JavaScript

E

Java

A alternativa D está correta.

O NPM é capaz de criar ferramentas executadas do terminal com JavaScript, além de abrir o site e o repositório do projeto e procurar por pacotes não enunciados no package.json. Além disso, permite ao

desenvolvedor criar um pacote de forma rápida por meio de perguntas lançadas e das respostas dadas pelo desenvolvedor.

Tarefas que antecedem o início do gerenciamento de softwares

Neste vídeo, apresentaremos as tarefas necessárias para iniciar o gerenciamento de software.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Para que seja possível realizar o gerenciamento de configuração de softwares, é necessário lançar mão de uma sequência de atividades. A primeira delas é fazer a **seleção dos itens passíveis de serem gerenciados e separá-los por categoria**, isso evita gastos acima do esperado.

Essas categorias devem incluir os itens genéricos, os que foram programados para serem reutilizados, os de importância para a segurança, os que são mais usados no ciclo de vida do aplicativo em desenvolvimento e os que podem ser modificados por um grupo de desenvolvedores simultaneamente. Cabe ressaltar que os itens não selecionados podem ser alterados livremente.

Atenção

Item é um produto de software ou um produto de desenvolvimento de software. Já um produto de software são programas de computador, procedimentos ou documentação relacionada, juntamente às informações que deverão ou não ser entregues ao usuário final.

Outro ponto levado em consideração nessa seleção de itens é a interação deles com outros, para que seja possível fazer a manutenção do software. A partir da **interação dos itens, ou seja, de sua classe de relacionamento**, o processo para achá-los no software é feito de forma mais rápida, sendo executável achá-los devido às alterações sofridas. As classes de relacionamento são divididas em cinco:

Equivalência
O mesmo item está presente em dois locais.

Dependência
A descrição de um projeto é dependente de outro.

Derivação
Um código-objeto, por exemplo, é derivado de um código-fonte.

Sucessão
Uma versão é a sucessora direta de uma anterior.

Variante
Uma versão para DOS ou para UNIX.

Ao final de cada fase do ciclo de vida de um projeto, deve-se planejar suas linhas-base, após cada manutenção e de forma periódica, especificando-se que itens serão ou não armazenados em cada linha-base preestabelecida. E, por fim, deve-se descrever de que forma esses itens podem ser recuperados e arquivados no repositório.

Cada item precisa ser identificado. Assim, cada item deve possuir identificação/nome único, para que por meio do nome seja possível fazer o reconhecimento da hierarquia entre os itens e de sua evolução nas versões do projeto.

Controle de mudanças

Controle de mudanças

Neste vídeo, apresentaremos o conceito de controle de mudanças de software e as diferenças para o controle de versão.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

O **controle de mudanças** é uma combinação de ações humanas e ferramentas automatizadas que, juntas, têm o papel de tornar todas as alterações realizadas no software organizadas, a fim de evitar alterações desorganizadas que culminem na perda ou na falha do projeto, e ainda permitir a previsão de seu efeito em todo o sistema. O momento ideal para a implementação do controle de mudanças é após a fixação de uma baseline.

Exemplo

Quando implementado adequadamente, o controle de mudanças permite que pedidos similares se agrupem, que solicitações de alterações sejam consideradas em conjunto com outras (evitando-se mais do que uma mudança desnecessariamente) e que solicitações incompatíveis com o sistema sejam negadas. Ainda é possível criar graus de prioridades de solicitações e, a partir daí, montar um cronograma de alterações do software.

O controle de versões, como já dito, é responsável por melhorar os itens desde a criação até sua versão final para utilização. De forma que, após a versão final ser atingida, os itens devem atender aos requisitos preestabelecidos e necessários para o bom funcionamento do software, ao qual cada item será direcionado, evitando-se um processo caótico.

Geralmente as empresas que trabalham com DevOps possuem diversos desenvolvedores e estes trabalham em conjunto, com vistas a acelerar o processo de criação do software. Por isso, para que sejam possíveis as alterações dos itens sem a perda ou a sobreposição de alterações, é usado o controle de versões com a

identificação e o armazenamento das versões em repositórios. A identificação deve ser feita, idealmente, no esquema de árvore, para a visualização do histórico das versões de um item, bem como suas ramificações a partir de qualquer versão.

Auditoria e controle de interface de software

Neste vídeo, apresentaremos o conceito de auditoria de configuração e seu papel no desenvolvimento e configuração de software.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Auditoria de configuração de software

A auditoria de configuração complementa o controle de versões, ela garante que as alterações feitas foram realizadas e implementadas corretamente e que, por isso, são obrigatórias no ciclo de desenvolvimento e manutenção de software. Existem dois tipos de auditoria de configuração:

Auditoria funcional

É uma auditoria de revisão técnica, consiste em avaliar aspectos internos dos arquivos e é considerada uma verificação formal. Faz parte do controle de qualidade, visto que seu objetivo é esclarecer quaisquer erros e esquecimentos capazes de interferir nos padrões de construção do software.

Auditoria física

É a auditoria de configuração propriamente dita. São feitos questionamentos aos desenvolvedores acerca de suas ações no que tange ao desenvolvimento do software levando em consideração: se as alterações foram realizadas na ordem especificada e se foram especificadas também a data e o desenvolvedor responsável pelas alterações; se houve modificações adicionais incorporadas; se foi realizada a revisão técnica formal e se os padrões exigidos foram seguidos; se os atributos do item de configuração fazem sentido na alteração realizada e se foram seguidas as etapas de gerenciamento de configuração, bem como outras interrogações pertinentes ao software em desenvolvimento.

O **relato de situação** atua conjuntamente com as auditorias que vimos, esclarece a todos os envolvidos no desenvolvimento e manutenção do software as informações a seguir:

1. O que aconteceu?
2. Quem o fez?
3. O que mais será afetado?

Além disso, o relato de situação tem a função de questionar alterações das configurações de software.

Controle de interface e controle de subcontratados e fornecedores

O **controle de interface** é capaz de coordenar as alterações feitas em itens de configuração que são afetados por outros itens que não estão sendo controlados. Dessa forma, dizemos que o principal papel do controle de interface é o de segurança, porque auxilia na implementação de políticas de controle de acesso em sistemas de gerenciamento.

O ideal é que sistemas de softwares ou softwares de suporte sejam examinados na busca de possíveis interfaces para manter o projeto sob controle. Em cada interface, recomenda-se que seja descrito o tipo de interface e as unidades organizacionais que foram afetadas. Somado a isso, também deve-se descrever de que modo será realizado o controle sobre a interface e como os documentos de controle da interface serão aprovados.

Já o **controle de subcontratados e fornecedores** é responsável por fazer a testagem de itens que já foram desenvolvidos por solicitação a outras empresas ou que já foram adquiridos prontos. Após essa testagem, caso esteja tudo conforme o planejado e requerido, o item é incorporado ao repositório do projeto em andamento.

Recomendação

É importante que os desenvolvedores responsáveis por essa fase do processo de produção do projeto esclareçam se os requisitos solicitados pelo pessoal do gerenciamento e configuração estão sendo contemplados pelo subcontratado, bem como, interrogar de que maneira será feito o monitoramento sobre o subcontratado, de que maneira o código, a documentação e os dados externos serão testados, aceitos e adicionados ao projeto e como serão abordadas as questões de propriedade do código produzido, como direitos autorais e de propriedade (também chamadas de licenças) e royalties.

No caso dos itens que foram adquiridos já ativos, ou seja, prontos, é necessário discorrer sobre a forma como serão recebidos, testados e colocados sob controle de gerenciamento de configuração, de que maneira as alterações no software do fornecedor serão tratadas e se o fornecedor desses itens participará ou não do processo de gerenciamento de mudança do projeto em andamento. No caso dessa participação, deve-se descrever também como ela vai ocorrer.

Verificando o aprendizado

Questão 1

Sobre as tarefas a serem realizadas antes mesmo do início do gerenciamento de softwares, no que tange à interação dos itens e sua classe de relacionamento, correlacione:

(1) Sucessão

(2) Equivalência

(3) Dependência

(4) Variante

(5) Derivação

( ) Ocorre quando o mesmo item está presente em dois locais.

( ) Ocorre quando a descrição de um projeto é dependente de outro.

( ) Ocorre quando um código-objeto, por exemplo, é derivado de um código-fonte.

( ) Ocorre quando uma versão é a sucessora direta de uma anterior.

( ) Ocorre quando existe uma versão para DOS ou para UNIX.

A

1, 2, 3, 4, 5

B

5, 4, 3, 2, 1

C

3, 1, 2, 4, 5

D

5, 1, 2, 3, 4

E

2, 3, 5, 1, 2

A alternativa E está correta.

A partir da interação dos itens (sua classe de relacionamento), eles são encontrados no software de forma mais rápida, devido às alterações sofridas. As classes de relacionamento são divididas em cinco: 1) equivalência: ocorre quando o mesmo item está presente em dois locais; 2) dependência: ocorre quando a descrição de um projeto é dependente de outro; 3) derivação: ocorre quando um código-objeto, por exemplo, é derivado de um código-fonte; 4) sucessão: ocorre quando uma versão é a sucessora direta de uma anterior; e 5) variante: ocorre quando existe uma versão para DOS ou para UNIX.

Questão 2

Cada item a ser gerenciado nas atividades de gerência de software precisa de

A

nome e identificação única.

B

ciclo de vida.

C

controle de mudanças.

D

alterações.

E

um sistema em separado.

A alternativa A está correta.

Cada item precisa possuir identificação/nome único, viabilizando-se que pelo nome seja possível fazer o reconhecimento da hierarquia entre os itens e de sua evolução nas versões desenvolvidas do projeto.

Ferramentas de gerenciamento de software

Ferramentas de gerenciamento de software

Neste vídeo, apresentaremos as principais ferramentas de gerenciamento de software: CVS, RCS e SCCS.

> **Conteúdo interativo**
>
> Acesse a versão digital para assistir ao vídeo.

O papel das ferramentas de gerenciamento de configuração de software é classificar e gerenciar sistemas, separando-os por grupos e subgrupos, além de modificar configurações básicas de maneira centralizada. Essas ferramentas são capazes de implementar novas configurações em todos os sistemas aplicáveis, bem como automatizar atualizações, aplicações de patches e identificação de sistemas. Além de identificar configurações de baixo desempenho, desatualizadas e fora de conformidade, também priorizam ações, acessando e aplicando correções prescritivas.

Sistema de versões concorrentes – CVS

O *Concurrent Versions System* (CVS), sistema de versões concorrentes, foi criado como vários scripts de shell escritos por Dick Grune, postados no grupo de notícias “comp.sources.unixno”, volume 6, de julho de 1986. Três anos depois, o CVS ganhou suporte à ramificação do fornecedor. Ele é um sistema decontrole de versão.

O Concurrent Versions System (CVS), sistema de versões concorrentes, foi criado como vários scripts de shell escritos por Dick Grune, postados no grupo de notícias “comp.sources.unixno”, volume 6, de julho de 1986. Três anos depois, o CVS ganhou suporte à ramificação do fornecedor. Ele é um sistema de controle de versão.

Dick Grune

Também podemos dizer que o CVS é uma ferramenta open source, ou seja, disponibilizada sem restrições para o público, capaz de estabelecer as principais funções do processo de controle de versões. O CVS possui armazenado em seu repositório todas as modificações realizadas em um arquivo ao longo do seu ciclo de vida, em que cada modificação é identificada pelo que chamamos de “revisão” e denotada por um número.

Toda e qualquer revisão armazena tanto as modificações realizadas como quem as realizou, em que momento foram realizadas, entre outras informações. Sempre há a possibilidade de rastreamento para fins de consulta, comparação ou união com outras revisões.

> O CVS possui um mecanismo que possibilita o controle dos acessos simultâneos e das modificações paralelas, sem alterar a integridade das modificações e a atomicidade das operações.

Com o intuito de facilitar a vida dos desenvolvedores, a CVS evita o armazenamento de todas as versões já criadas de um determinado projeto. Na verdade, ele armazena todas as versões em um único arquivo, onde ficam armazenadas apenas as diferenças entre essas versões disponíveis. Além disso, o CVS isola os desenvolvedores uns dos outros, mesmo que estejam usando diferentes editores. Dessa forma, evita que duas pessoas editem o mesmo arquivo ao mesmo tempo.

Exemplo

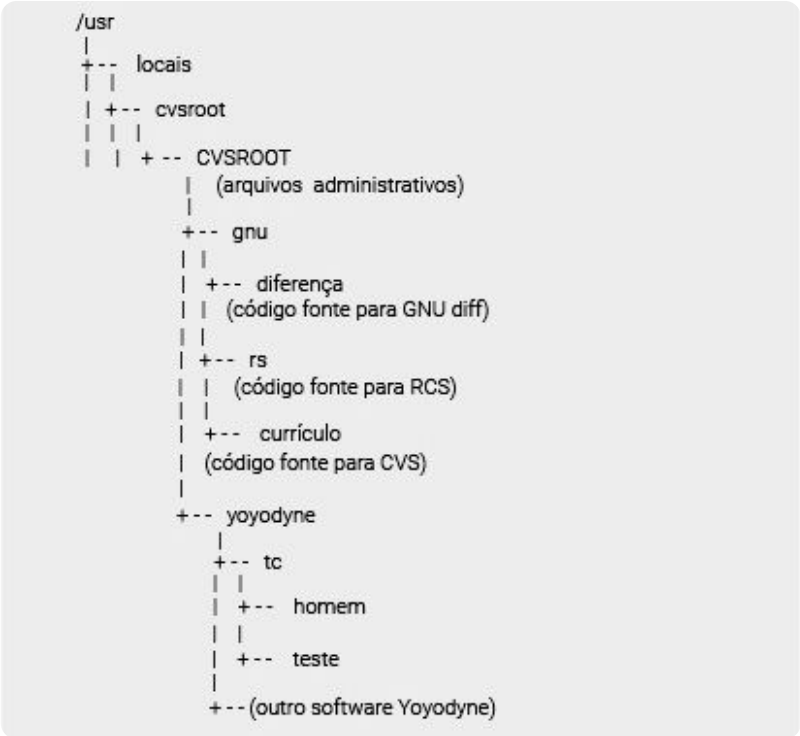
Esse isolamento não é possível se os desenvolvedores estiverem usando o editor GNU Emacs.

Após o trabalho dos desenvolvedores/editores, o CVS mescla as versões. A versão atual do CVS não possui nenhum código real dos scripts de shell, mas a resolução de conflitos do CVS advém deles.

Como e onde os dados são armazenados no repositório

Saber como os dados são armazenados no CVS pode ser necessário quando o desenvolvedor precisa rastrear bloqueios ou para lidar com as permissões de arquivo apropriadas para o repositório. Por isso (e para isso) deve-se avaliar: que dados estão armazenados no repositório, quais são as permissões do arquivo, os problemas específicos do Windows e os arquivos armazenados no sótão, quais são as informações adicionais no diretório CVS, que bloqueios CVS controlam acessos simultâneos, além de avaliar o armazenamento CVS ROOT.

No que tange ao local onde os dados são armazenados, deve-se lembrar que a estrutura geral do repositório é uma árvore de diretórios correspondente aos diretórios do diretório de trabalho (veja um exemplo na figura a seguir). Juntos a estes estão arquivos de histórico para cada arquivo sob controle de versão. O nome do arquivo de histórico é o nome do arquivo correspondente e possui “, v” anexado ao final do seu comando.



Árvore de diretórios

RCS e SCCS

Revision Control System – RCS

Este também é um software de controle de versão capaz de automatizar o armazenamento, a recuperação, o registro, a identificação e a fusão de revisões. Foi criado em 1982 por Walter F. Tichy. É o antecessor do CVS. Era a alternativa, à época, de open source mais evoluída do Source Code Control System (SCCS) – Sistema de Controle de Código-Fonte.

O design do RCS é uma melhoria do SCCS, inclui uma interface de usuário mais fácil e um armazenamento aprimorado de versões para a recuperação mais rápida.

Walter F. Tichy.

O RCS é funcional para uso em textos que serão revisados com certa frequência (por exemplo, programas, documentação, gráficos processuais, documentos e cartas) e melhora o desempenho com o armazenamento de uma cópia inteira da versão mais recente e, em seguida, armazena também as reversas (deltas). Além disso, o RCS pode lidar com arquivos binários e usa o GNU Diffutils para encontrar diferenças entre as versões, embora com uma eficiência reduzida quando comparado ao CVS.

O modo de operação do RCS se dá em arquivos individuais e não possibilita trabalhar com um projeto inteiro. No entanto, ele fornece ramificação para arquivos individuais, e sua sintaxe da versão é complicada, pois, ao invés de usar ramos, usa apenas o mecanismo interno de bloqueio e trabalha em um único ramo head.

Source Code Control System – SCCS

O Source Code Control System (SCCS) também é um sistema de controle de versão. Ele foi projetado para rastrear alterações no código-fonte e outros arquivos de texto durante o desenvolvimento de um software. Isso permite que o usuário recupere qualquer uma das versões anteriores do código-fonte original e as alterações que são armazenadas.

Marc Rochkind

Foi originalmente desenvolvido no final de 1972 por Marc Rochkind para um computador IBM System/370. O SCCS possui um recurso muito característico, a string sccsid, que é incorporada ao código-fonte e atualizada automaticamente pelo SCCS para cada revisão.

Esse sistema controla as alterações de arquivo e o histórico e, normalmente, é atualizado para uma nova versão. Assim, consegue corrigir bugs de forma a otimizar algoritmos e acaba por adicionar funções que antes não existiam ou não eram exploradas. Ao alterar o software, podem ocorrer problemas e estes exigirão o manejo do controle de versão para serem resolvidos.

Saiba mais

O código-fonte ocupa muito espaço, porque é repetido em todas as versões, por isso é difícil obter informações sobre “quando” e “onde” ocorreram as mudanças. Então, torna-se difícil encontrar a versão exata com a qual o cliente tem problemas. E foi exatamente por isso que o SCCS foi construído.

O SCCS da empresa AT&T possuía cinco versões principais para o IBM OS e cinco versões principais para o UNIX. Cabe ressaltar que existem ainda duas implementações específicas usando SCCS: o PDP 11 em Unix e o IBM 370 em OS.

Além disso, o SCCS consiste em duas partes que podemos considerar: os comandos SCCS e os arquivos SCCS. Todas as operações básicas como criar, excluir e editar podem ser realizadas por comandos acionados dentro do SCCS, onde os arquivos SCCS possuem um prefixo identificador de formato exclusivo 's.', o qual é controlado também por comandos SCCS.

> Os arquivos SCCS possuem três partes: a tabela delta, os sinalizadores de acesso e rastreamento e o corpo do texto propriamente dito.

A tabela delta consiste em armazenar os deltas, que são, cada um, uma única revisão em um arquivo SCCS. Dessa forma, cada arquivo SCCS possui seu próprio registro de alterações.

Quanto ao controle e rastreamento de sinalizadores em arquivos SCCS, pode-se dizer que a operação de cada um dos arquivos do SCCS é rastreada por sinalizadores e suas funções são as seguintes: configurar permissões para edição de cada arquivo SCCS, controlar a versão de cada arquivo SCCS e, dessa forma, permitir a edição colaborativa dos arquivos SCCS, além de realizar alterações de referência mútua dos arquivos SCCS.

Pages Versions Management – Version Web

Por último, mas não menos importante, existe o Web Pages Versions Management (Version Web), que foi desenvolvido em razão da constante evolução do ambiente de informações. A ideia por trás dessa ferramenta é disponibilizar o controle de versão da página da Web durante a navegação para os usuários, fornecendo aos webmasters uma maneira fácil de controlar as versões das páginas da Web, através dela própria.

Assim, o desenvolvimento de aplicações Web pode ser visto como um novo tipo de desenvolvimento de software. Desenvolvedores de páginas da Web (autores ou webmasters) enfrentam um trabalho árduo quando muitas pessoas estão envolvidas no desenvolvimento paralelo de conjuntos de páginas HTML relacionadas.

Como os autores podem trabalhar independentemente em suas cópias de páginas individuais, a integração dessas cópias em um hiperdocumento final (fusão de duas ou mais versões) torna-se um problema conhecido. Além disso, o volume dos documentos geralmente é alto, as modificações nos documentos acontecem rapidamente e são difíceis de rastrear. De fato, para fornecer um mecanismo eficiente, o controle de mudanças deve combinar procedimentos humanos com ferramentas automáticas.

Durante a navegação na Web, o usuário consegue recuperar informações anteriormente disponíveis da página da Web, visualizar as diferenças entre as versões da página e ser notificado quando novas versões estiverem disponíveis. Outro objetivo da ferramenta é facilitar a cooperação entre os autores, permitindo que eles tenham acesso simultâneo aos arquivos, gerenciem as diferentes versões de um arquivo e localizem as alterações realizadas. E para fornecer seu principal recurso funcional, o Version Web utiliza o Concurrent Versions System (CVS).

Verificando o aprendizado

Questão 1

Gerenciar a configuração de software é uma atividade importante para o desenvolvimento de sistemas. Qual é o papel das ferramentas de gerenciamento de configuração?

A

Classificar e gerenciar sistemas.

B

Consertar erros no sistema.

C

Juntar versões antigas e posteriores de softwares.

D

Apenas gerenciar sistemas.

E

Apenas classificar sistemas.

A alternativa A está correta.

As ferramentas de gerenciamento de configuração classificam e gerenciam sistemas, separando-os por grupos e subgrupos, além de modificar configurações básicas de maneira centralizada.

Questão 2

CVS (Control Version System ou sistema de controle de versão) é uma importante ferramenta no desenvolvimento de sistemas. Marque V ou F, no que tange ao CVS, nas sentenças a seguir:

( ) O CVS possui armazenado em seu repositório todas as modificações realizadas em um arquivo ao longo do seu ciclo de vida.

( ) Ele é um sistema de controle de versão.

( ) O CVS foi criado como scripts de shell escritos por Dick Grune.

A

F F F

B

V V V

C

V V F

D

F F V

E

F V F

A alternativa B está correta.

O CVS foi criado por scripts de shell escritos por Dick Grune. É uma ferramenta open source, um sistema de controle de versão. O CVS armazena em seu repositório todas as modificações realizadas em um arquivo ao longo do seu ciclo de vida, em que cada modificação é identificada pelo que chamamos de "Revisão". Esta é denotada por um número, e toda revisão armazena tanto as modificações realizadas como quem as realizou – e em que momento foram realizadas.

Considerações finais

O gerenciamento de configuração de software é um processo que mantém controladas todas as muitas mudanças realizadas ao longo do desenvolvimento do software, mantendo-o estável durante sua evolução até a fase final por meio da identificação da configuração de um sistema em diversos pontos em tempo.

Ressalte-se que o DevOps é capaz de acelerar e otimizar o gerenciamento de configuração, o qual mantém a integridade e a rastreabilidade da configuração durante o ciclo de vida do sistema. A estrutura é gerenciada como código e, então, diversas equipes de desenvolvedores colaboram utilizando práticas ágeis de desenvolvimento, como a revisão por pares, testes automatizados e entrega contínua. Dessa forma, busca-se evitar erros no desenvolvimento do projeto de software, diminuir o tempo de entrega aumentando sua segurança e possibilitando a fuga de futuros problemas. Além disso, controlar sistematicamente as mudanças realizadas, mantendo a integridade e a rastreabilidade da configuração durante o ciclo de vida do sistema.

Podcast

Neste podcast, apresentaremos o processo de gestão e configuração de software.

Conteúdo interativo

Acesse a versão digital para ouvir o áudio.

Explore +

Leia o artigo **O que é gerência de configuração de software**, no blog de André Felipe Dias, e correlacione o conteúdo com o que você aprendeu durante seus estudos.
https://blog.pronus.io/posts/controle-de-versao/o-que-eh-gerencia-de-configuracao-de-software

Referências

ALURA. **Elixir Parte 3**: conhecendo o Ecossistema Elixir. Consultado na Internet em: 17 nov. 2022.

ANDRADE, A. P. de. **O que é o Composer**. Treinaweb (s.d.). Consultado na Internet em: 17 nov. 2022.

BEZERRA, G. **Gerenciando dependências com o Maven em projetos Java**.Publicado em: 12 jun. 2018. Consultado na Internet em: 17 nov. 2022.

BRITO, E. **RubyGems corrigiu o bug de aquisição de pacotes não autorizados**. Blog do Edivaldo. Publicado em: 9 maio 2022. Consultado na Internet em: 17 nov. 2022.

COELHO, F. H. M. de S. **Processo de gerência de configuração de software**. Monografia (TCC do Curso de Engenharia de Computação) – Universidade São Francisco, Itatiba, São Paulo, dez. 2007. Consultado na Internet em: 17 nov. 2022. CPAN, 2021. Consultado na Internet em: 17 nov. 2022.

CVS (Concurrent Versions System v1.11.23). **Sistema de versões simultâneas v1.11.23**. Consultado na Internet em: 17 nov. 2022.

DIAS, A. F. **Conceitos básicos de controle de versão de software** — Centralizado e Distribuído. Publicado em: 11 maio 2016. Consultado na Internet em: 17 nov. 2022.

FREITAS, R. S.; PARREIRA JR., W. M. **Um comparativo entre os gerenciadores de dependências em Java**. Simpósio de Pós-Graduação. Instituto Federal Triângulo Mineiro, 2017. Consultado na Internet em: 17 nov. 2022.

GAEA. **Veja como funciona a gestão de configuração no mundo DevOps**. s.d. Consultado na Internet em: 17 nov. 2022.

HEXDOCS. **MiX v1.14.2**. Consultado na Internet em: 17 nov. 2022.

LINUX. **Cargo e Nix**: mais 2 sistemas de gerenciamento de pacotes para GNU/Linux. Consultado na Internet em: 17 nov. 2022.

NASCIMENTO, W. M. **O que é um gerenciador de dependências?** Treinaweb. Consultado na Internet em: 17 nov. 2022.

RAMOS, A. M. **Interface de controle de acesso para o modelo de gerenciamento Osi**. Dissertação (Mestrado em Ciência da Computação) – Universidade Federal de Santa Catarina, Florianópolis, ago. 1994. Consultado na Internet em: 17 nov. 2022.

RED HAT. **Gerenciamento de configuração**. Publicado em: 11 jul. 2019. Consultado na Internet em: 17 nov. 2022.

REVISION CONTROL SYSTEM. Consultado na Internet em: 17 nov. 2022.

RIBEIRO, R. T. **Instalação de Módulos Pear e Pecl**. Publicado em: 26 nov. 2010. Consultado na Internet em: 17 nov. 2022.

TREINAWEB. **Yarn**: um novo gerenciador de dependências para JavaScript. Consultado na Internet em: 17 nov. 2022.

UFPR. **Aula de gerenciamento de configuração**. 2018. Consultado na Internet em: 17 nov. 2022.