Engenharia de Software

Anhanguera

AVALIE SUA PROFISSÃO

QUANDO APARECER EM SEU PORTAL UMA AVALIAÇÃO SOBRE SEU CURSO, RESPONDA:

NOTAS **9 OU 10** SIGNIFICA QUE VOCÊ INDICA

NOTAS **7 OU 8** SIGNIFICA QUE VOCÊ NÃO INDICA

Anhanguera

Anhanguera

CONTROLE DE VERSÕES

GERENCIAMENTO DA CONFIGURAÇÃO DE SOFTWARE (GCS)

A GCS é um conjunto de práticas que proporciona o controle de todo o processo de desenvovimento de software, necessário pela complexidade envolvida no processo de desenvolvimento e pela grande quantidade de componentes de um software.

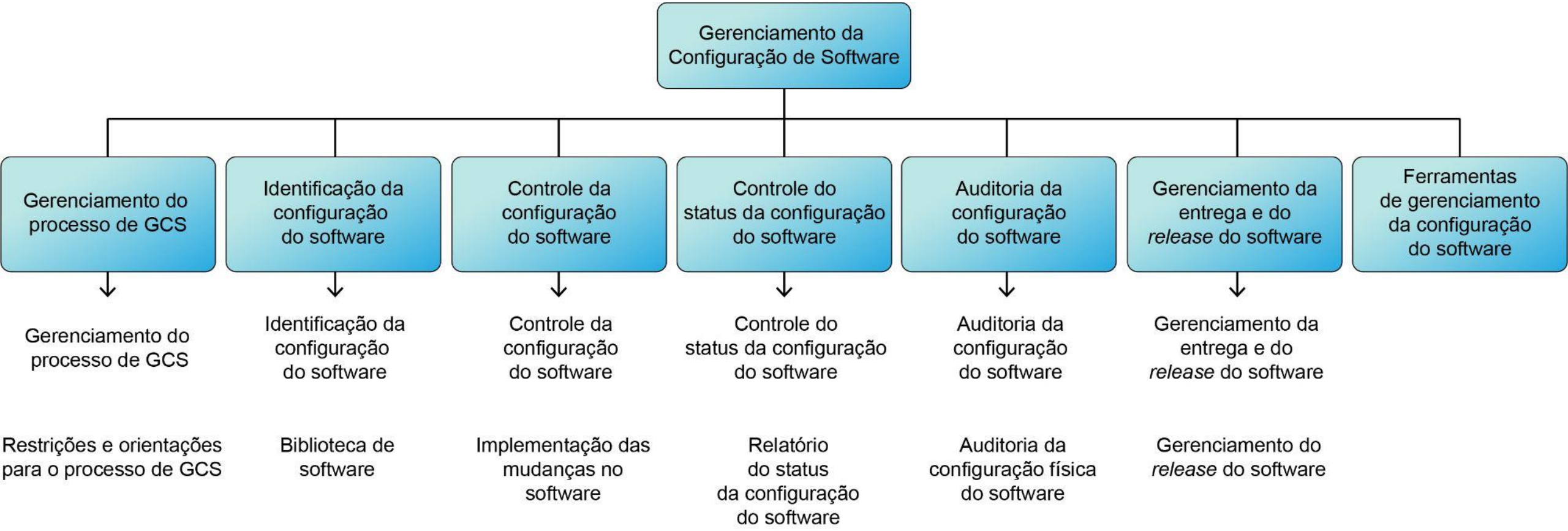
Um software é um produto em constante evolução. Da sua concepção até a entrega (e além dela), ele nasce, cresce e carece de manutenção ao longo de um ciclo de vida bem definido e conhecido. Na condição de um produto dinâmico, seu processo de construção e sua manutenção requerem gerenciamento contínuo emtodos os aspectos de sua construção e, mesmo em um estado considerado apto para entrega, um software passa por diversas modificações, e cada uma delas temcomo resultado uma nova versão dele próprio. Por isso, um controle eficiente das revisões é necessário, o que é atingido por meio da utilização de uma ferramentade controle de versões.

GERENCIAMENTO DE CONFIGURAÇÃO DO SOFTWARE

Embora o controle de versões seja o foco desta seção, será necessário contextualizá-lo em um tema mais abrangente, do qual ele faz parte como um elemento muito importante. Esse tema mais abrangente se chama Gerenciamentoda Configuração do Software (GCS) que, de modo objetivo, é o meio pelo qual se proporciona controle a todo o processo de desenvolvimento de software (LEON,2004). Esse controle se torna necessário, entre outros motivos, pela complexidade envolvida no processo de desenvolvimento e pela grande quantidade decomponentes de um software.

A motivação maior para a criação e para a estruturação de uma disciplina que cuida da Gerência de Configuração de Software foi a necessidade de controlar as modificações pelas quais inevitavelmente passa um programa, o que é feito com o uso de métodos e de ferramentas e com o intuito de maximizar a produtividade e minimizar os erros cometidos durante a evolução.

Como o GCS não se resume a uma ação apenas, ele é composto por uma série de atividades bem definidas e estruturadas. De acordo com IEEE (2004), essas atividades incluem: gerenciamento e planejamento do processo de GCS, identificação de configuração do software, controle de configuração de software, controle de status de configuração de software, auditoria de configuração de software, gerenciamento da entrega e do lançamento de software e configuração das ferramentas de gerenciamento. A Figura 1.10 ilustra essas atividades em uma estrutura hierárquica.


Gerenciamento da Configuração de Software
Gerenciamento do processo de GCS
Identificação da configuração do software
Controle da configuração do software
Controle do status da configuração do software
Auditoria da configuração do software
Gerenciamento da entrega e do *release* do software
Ferramentas de gerenciamento da configuração do software
Gerenciamento do processo de GCS
Identificação da configuração do software
Controle da configuração do software
Controle do status da configuração do software
Auditoria da configuração do software
Gerenciamento da entrega e do *release* do software
Restrições e orientações para o processo de GCS
Biblioteca de software
Implementação das mudanças no software
Relatório do status da configuração do software
Auditoria da configuração física do software
Gerenciamento do *release* do software

Na sequência serão discutidas as três primeiras grandes atividades do SCM, colocadas nos retângulos da Figura 1.10, de acordo com IEEE (2004).

Gerenciamento do Processo de CGS: conforme temos mencionado, o GCScontrola a evolução e a integridade de um produto e o faz por meio daidentificação dos seus elementos, pelo gerenciamento e controle das mudanças aplicadas e pela verificação, registro e relato das informações de configuração. Da perspectiva do engenheiro de software, o SCM facilita o desenvolvimento e as atividades de implementação de mudanças. Umaimplementação de SCM bem-sucedida requer planejamento e gerenciamento cuidadosos, o que, por sua vez, requer compreensão do contexto organizacional e das restrições impostas ao projeto e à implementação do processo de SCM.

Identificação da configuração do software: essa atividade identifica os itensa serem controlados, estabelece esquemas de identificação para os itens e suas versões e, por fim, estabelece as ferramentas e técnicas a serem utilizadas na aquisição e no gerenciamento de itens controlados.

Controle de configuração de software: essa atividade trata do gerenciamentode mudanças durante o ciclo de vida do software. Ela abrange o procedimento para determinar quais mudanças fazer, a autoridade para aprovar certas mudanças, o suporte para a implementação dessas mudanças e o conceito de desvios formais dos requisitos do projeto.

Mesmo com a evidente necessidade de se gerenciar a configuração de um software, essa prática sofre com alguns mitos, muitos deles com potencial para desencorajar sua adoção nas organizações. Nas próximas linhas trataremos de três desses mitos e daremos bons motivos para que eles sejam desconstruídos, sempre com base na visão de Leon (2004).

Mito 1 – Promover o gerenciamento da configuração de um software significa ter mais trabalho e adotar novos procedimentos: de fato aimplantação e o gerenciamento adequados de uma sistemática de GCS não é uma tarefa simples. O período de transição entre um ambiente sem gerenciamento e um cenário de efetiva utilização de uma ferramenta para esse fim costuma ser difícil, já que novas habilidades devem ser desenvolvidas, novos procedimentos devem ser seguidos, entre outros desafios. No entanto, a transição será tão mais simples quanto mais eficiente for o trabalho das equipes de gerenciamento e de implementação do GCS. Ao utilizarem osistema, os desenvolvedores e outros atores do processo de criação do software deverão compreender seus potenciais benefícios e o esforço que conseguirão evitar por meio da automação de tarefas e da eliminação de erros. Atualmente, as ferramentas de gerenciamento de configuração de um software automatizam todas as tarefas repetitivas, facilitando assim a atuação do pessoal envolvido com o produto.

Mito 2 – O gerenciamento da configuração é de responsabilidade única dopessoal da administração do sistema: ao contrário do que sugere o mito, a implantação e a condução do GCS é responsabilidade de todas as pessoas envolvidas no processo de desenvolvimento de software embora a principal tarefa do pessoal da administração seja criar um ambiente organizacional no qual o GCS possa prosperar. Uma equipe de GCS precisa de todo o apoio dopessoal da administração para ter condições de implantar o sistema de gerenciamento da configuração aos poucos, pois é esse pessoal que deve monitorar a implementação e a operação do GCS, revisar periodicamente seu progresso e tomar as ações corretivas quando necessário.

Mito 3 – O gerenciamento da configuração é apenas para osdesenvolvedores: certamente as equipes de desenvolvedores constituem umdos mais frequentes usuários do GCS. Além disso, são os desenvolvedores que mais se beneficiam de um sistema de gerenciamento corretamente implementado. Problemas como perda de código-fonte, incapacidade deencontrar a última versão de um arquivo e reaparecimento de erros já corrigidos podem ser evitados com um sistema desses. Entretanto, há desenvolvedores que enxergam o GCS como perda de tempo e dele participam apenas por serem obrigados a isso. A potencial hostilidade voltada ao GCS pode ser eliminada se os desenvolvedores forem educados nos princípios daprática e esclarecidos sobre seus benefícios. As ferramentas atuais de GCS proporcionam um nível fenomenal de automação, o que acaba contribuindo para que outros atores do processo de desenvolvimento acabem se beneficiando delas, incluindo testadores e pessoal de qualidade, manutenção esuporte.

CONTROLE DE VERSÕES

Feita a contextualização do Gerenciamento de Configuração do Software com o ambiente em que o controle de versões está inserido, avançamos para a abordagem desse tema. De acordo com Caetano (2004), o controle de versões é o meio pelo qual o GCS controla, de forma consistente, as modificações realizadasem um sistema. Isso ocorre por meio das seguintes funções:

Recuperar versões anteriores do sistema.

Auditar as modificações realizadas, levantando quem alterou, quando alterou e o que alterou.

Automatizar o rastreamento de arquivos.

Estabelecer meios para obter a situação de um projeto em determinado ponto do tempo.

Prevenir conflitos entre desenvolvedores.

Permitir o desenvolvimento paralelo do um ou mais sistemas.

REPOSITÓRIO

Todas essas funções parecem convergir para um elemento bem definido: acentralização dos dados. E se não tivéssemos essa centralização? Bem, o uso descentralizado de um arquivo de código-fonte, por exemplo, permitiria que vários programadores acessassem vários arquivos e cada alteração feita nele não seria refletida nos demais. Assim, o desenvolvimento paralelo seria inviável e o conflito entre desenvolvedores inevitável. O recurso que as ferramentas de controle deversão (abordadas a seguir) usam é o repositório, local em que programas em desenvolvimento, fotos, vídeos e demais arquivos ficam armazenados e podem ser acessados de forma controlada por todos os envolvidos no desenvolvimento do produto.

BASELINES(LINHAS DE BASE)

Um conceito importante no escopo do controle de versão é a baseline de software. As baselines representam conjuntos de itens de configuração formalmente aprovados, os quais servem de base para as etapas seguintes de desenvolvimento. Quando, no entanto, uma entrega formal é feita ao cliente no final de uma iteração, denominamos tal entrega de release. Baselines e releases são identificadas no repositório de programas, na grande maioria das vezes, pelo uso de etiquetas (tags) (DANTAS, 2009).

O termo também é usado para se referir a uma versão específica de um item deconfiguração de software que foi acordado e, em qualquer caso, a baseline sópode ser alterada por meio de procedimentos formais de controle de mudanças. Uma baseline, junto com todas as alterações aprovadas na linha de base,representa a configuração aprovada atual.

As comumente usadas são as linhas de base funcionais, alocadas, de desenvolvimento e de produto. A linha de base funcional corresponde aos requisitos de sistema revisados. A linha de base alocada corresponde à especificação de requisitos de software revisada e à especificação de requisitos de interface de software. A linha de base de desenvolvimento representa a configuração de um software em evolução, em momentos selecionados, durante o ciclo de vida do software. A autoridade de mudança para essa linha de base normalmente é da organização de desenvolvimento, mas pode ser compartilhada com outras organizações. A linha de base do produto corresponde ao produto de software completo e entregue para integração de sistema (IEEE, 2004).

BRANCHES

A Gerência de Configuração de Software também permite que a implementação de novas funcionalidades por uma equipe seja realizada em paralelo, mas de forma isolada e independente das modificações de outros desenvolvedores. Oisolamento é obtido com uso de ramificações (branches). As linhas dedesenvolvimento (codelines) são designadas para cada projeto e são compartilhadas por vários desenvolvedores. A primeira linha de desenvolvimento definida no projeto é, por convenção, nomeada mainline. O ramo é criado no repositório e representa uma linha secundária de desenvolvimento, a qual podeser unida novamente à linha principal (mainline) por meio da operação de junção(merge). Atualmente, a necessidade de atender, ao mesmo tempo, as múltiplas demandas do projeto tornam o uso de ramos um diferencial (DANTAS, 2009).

FERRAMENTAS DE CONTROLE DE VERSÃO

Como era de se esperar, as funções próprias do controle de versões podem (e devem) ser executadas por uma ferramenta computacional, fato que naturalmente apresenta indiscutíveis vantagens em relação a uma eventual execução manual. Omercado disponibiliza ótimas ferramentas de controle de versão e três delas serão abordadas na sequência. Os critérios para adoção da ferramenta mais adequada deverão variar entre organizações, mas vale o alerta de que nenhuma decisão deveser tomada com base na convicção de que ela fará as vezes de um compilador, deque substituirá o gerente do projeto, ou de que ela pode se tornar um meio de automatizar testes. Passemos, então, à discussão de duas das mais importantes e utilizadas ferramentas de controle de versão.

GIT E GITHUB

Nossa primeira providência, ao tratarmos das ferramentas Git e GitHub, será a de diferenciá-las. Apesar de terem nomes bastante parecidos e de serem produtos do mesmo desenvolvedor, suas funções são distintas: o Git é uma ferramenta de controle de versão bastante popular entre desenvolvedores e apresenta recursosde colaboração bastante aprimorados, incluindo fóruns de discussão para osprojetos em desenvolvimento e para abordagem das alterações em curso.Presente também em outras ferramentas de controle de versão, os branches implementados no Git viabilizam o trabalho em paralelo entre membros da equipe e o controle de subprojetos que um desenvolvedor pode manter paraexperimentos próprios, incluindo correção de bugs e aprimoramento de funções, sem que os arquivos do repositório central sejam alterados e com a possibilidadede que seus experimentos sejam compartilhados (MAILUND, 2017).

Anhanguera

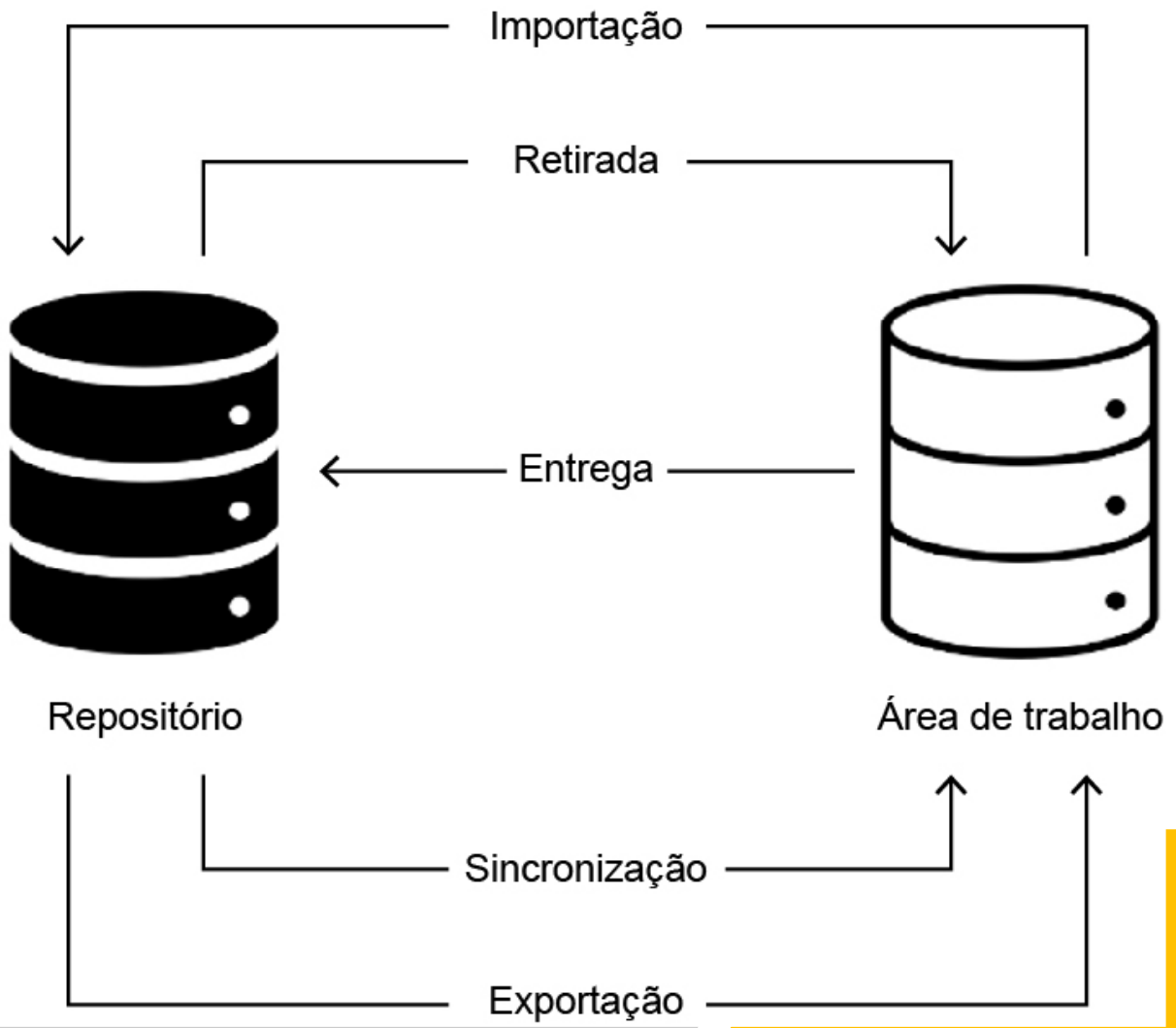
O Git é um sistema de controle de versão gratuito e de código aberto, concebido para lidar com todos os projetos, desde os pequenos até aqueles bem grandes eque movimentam praticamente toda as equipes. Ele é bem fácil de ser aprendido eocupará pouco espaço do servidor. Além disso, seu desempenho é bastantesatisfatório. Mas e a segurança? Bem, o modelo de dados que o Git usa garantirá a integridade criptográfica de cada bit dos projetos. Cada arquivo e cada operação de commit passa por verificação de checksum, operação também aplicada em suarecuperação. Será impossível extrair qualquer coisa do Git além dos bits exatos que foram adicionados.

O recurso do Git que realmente o diferencia de quase todos os outros sistemas de controle de versão existentes é seu modelo de branching (ou ramificação). Com o Git, é possível ter vários branches nas máquinas locais, os quais podem serindependentes uns dos outros. As operações de criação, merge e exclusão dos branches leva alguns poucos segundos apenas. Com esse diferencial, as equipespoderão trocar de contexto quase que imediatamente, além de criar branches para experimentar uma ideia e, mesmo aplicando a operação de commit, voltar aoponto de onde começou. O mesmo cabe para a aplicação de um patch, por exemplo.

O GitHub é equivalente ao repositório do Git, mas disponível em uma plataforma mundial e com acesso gratuito aos desenvolvedores que lá desejarem armazenar seus programas e compartilhá-los com outros desenvolvedores. Por meio da página de cadastro (GITHUB, c2020) é possível criar sua conta no GitHub e começara fazer parte dessa grande comunidade. Bem, mas o que é possível obter com acriação de uma conta no GitHub além de um espaço para compartilhar código? Aresposta mais simples vem com apenas uma palavra: visibilidade. Empresas de TIjá consideram o repositório do candidato no GitHub como um dos critérios parasua contratação. Tamanha importância justifica nossa incursão por essa ferramenta.

CVS (CONCURRENT VERSION SYSTEM OU SISTEMA DE VERSÕES CONCORRENTES)

Trata-se de uma ferramenta open source que implementa as principais funções relacionadas ao processo de controle de versões. O CVS armazena em seurepositório as modificações realizadas num arquivo ao longo do tempo. Cada modificação é identificada por um número chamado revisão. Toda revisão armazena as modificações realizadas, quem realizou as modificações, quando foram realizadas, entre outras informações (CAETANO, 2004). A Figura 1.13 ilustraas principais funções realizadas pelo CVS.


Importação
Retirada
Entrega
Repositório
Área de trabalho
Sincronização
Exportação

O repositório CVS – assim como o de outras ferramentas – armazena uma cópia completa de todos os arquivos e diretórios que estão sob controle de versão. Normalmente, o desenvolvedor nunca acessa nenhum dos arquivos no repositóriodiretamente. Em vez disso, usa comandos CVS para obter sua própria cópia dos arquivos em uma área de trabalho e, em seguida, trabalha nessa cópia. Quando termina um conjunto de alterações, realiza uma operação chamada

Commit (ou confirmação) de volta para o repositório. Por isso, o repositório guarda as mudanças feitas pelo desenvolvedor, além de registrar exatamente o que equando foi alterado e outras informações semelhantes. Observe que o repositório não é um subdiretório da área de trabalho ou vice-versa e que eles devem estarem locais separados (GNU, [s.d.]).

A estrutura geral do repositório é uma árvore de diretórios correspondente aosexistentes no diretório de trabalho. Por exemplo, supondo que o repositório esteja em /usr/local/cvsroot, uma possível estrutura de diretório é mostrada na Figura1.14

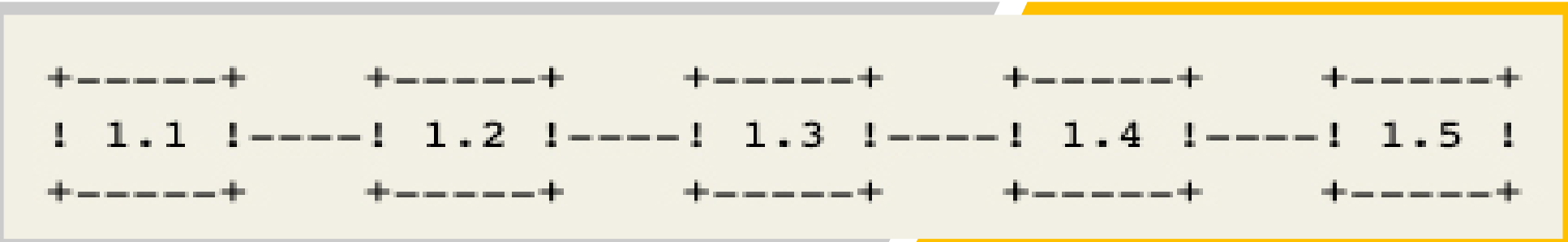
Anhanguera

```
/usr
 |
 +--local
 |     |
 |     +--cvsroot
 |     |      |
 |     |      +--CVSROOT
 |     |             |        (administrative files)
 |     |             |
 |     |             +--gnu
 |     |             |     |
 |     |             |     +--diff
 |     |             |     |     (source code to GNU diff)
 |     |             |     |
 |     |             |     +--rcs
 |     |             |     |     (source code to RCS)
 |     |             |     |
 |     |             |     +--cvs
 |     |             |           (source code to CVS)
 |     |             |
 |     |             +--yoyodyne
 |     |                   |
 |     |                   +--tc
 |     |                   |      |
 |     |                   |      +--man
 |     |                   |      |
 |     |                   |      +--testing
 |     |                   |
 |     |                   +--(other Yoyodyne software)
```

Na estrutura de diretórios estão os arquivos de histórico de cada arquivo sobcontrole de versão. O nome do arquivo de histórico é o nome do arquivocorrespondente com ‘, v’ anexado ao final. Os arquivos “index.php,v” e “frontend.c,v” são exemplos possíveis de arquivos de históricos. Eles guardam,entre outras coisas, informações suficientes para recriar qualquer revisão doarquivo, mais um log de todas as mensagens de commit e o nome do usuário queenviou a revisão. Os arquivos de histórico são conhecidos como arquivos RCS,porque o primeiro programa a armazenar arquivos nesse formato foi um sistemade controle de versão conhecido como RCS (GNU, [s.d.]).

Cada alteração feita no programa gera um novo número de versão que o identifica. O CVS atribui automaticamente números como 1.1, 1.2 e assim por diante para as versões geradas. Um arquivo pode ter várias versões e, da mesma forma, um produto de software pode ter várias versões. Esse produto geralmente recebe um número de versão como “4.1.1”. Cada versão de um arquivo possui um número de revisão exclusivo. Os números de revisão são semelhantes a “1.1”, “1.2”, “1.3.2.2”ou mesmo “1.3.2.2.4.5”.

Um número de revisão sempre tem um número par de inteiros decimaisseparados por ponto. Por padronização, a revisão 1.1 é a primeira de um arquivo.Cada uma delas recebe sucessivamente um novo número, aumentando o númeromais à direita em um. A Figura 1.16 exibe algumas revisões, com as mais recentes àdireita. Também é possível terminar com números contendo mais de um ponto,por exemplo “1.3.2.2”.



Antes de encerrarmos o conteúdo do CVS, vale tratarmos de mais algumas terminologias relacionadas ao assunto (GNU, [s.d.]).

Checkout: denominação dada à primeira recuperação (ou download) de um módulo do sistema vindo do repositório do CVS.

Commit: trata-se do envio do artefato modificado ao repositório do CVS.

Export (ou Exportação): trata-se da recuperação (ou download) de um módulo inteiro a partir de um repositório, sem os arquivos administrativos CVS. Módulos exportados não ficam sob controle do CVS.

Import (ou Importação): esse termo identifica a criação de um módulo completo no âmbito de um repositório CVS, feita por meio de um upload de uma estrutura de diretórios.

Module (ou módulo): é a representação de uma hierarquia de diretórios. Umprojeto de determinado software efetiva-se como um módulo dentro do repositório.

Release: este termo equivale a um “lançamento”. Um número de releaseidentifica a versão de um produto completo ou inteiro.

Merge: refere-se à fusão das diversas modificações feitas por diferentesusuários na cópia local de um mesmo arquivo. Sempre que alguém altera ocódigo, é necessário realizar uma atualização antes da aplicação da operação de commit, a fim de que seja feita a fusão das mudanças.

Uma das mais importantes funções de um sistema de controle de versões é a de_____ as alterações feitas no sistema, através do levantamento de quem as efetivou e quando as fez. Outra função bastante conhecida é aquela que permite o desenvolvimento _____ de um sistema ou de vários deles. Por fim, umaferramenta desse tipo permite também a criação de _____ do mesmo sistema.

Assinale a alternativa com os termos que preenchem corretamente as lacunas do texto.

a. auditar; paralelo; versões.
b. auditar; isolado; versões.
c. desconsiderar; paralelo; documentação.
d. auditar; isolado; versões.
e. afirmar; paralelo; cópias.

Como o GCS não se resume a uma ação apenas, ele é composto por uma série deatividades bem definidas e estruturadas. De acordo com IEEE (2004), essas atividades incluem: o gerenciamento e o planejamento do processo de GCS, a identificação de configuração do software, o controle de configuração de software, o controle de status de configuração de software, a auditoria de configuração desoftware, o gerenciamento da entrega e do lançamento de software e aconfiguração das ferramentas de gerenciamento.

Considerando as características e atividades próprias do Gerenciamento da Configuração do Software, assinale a alternativa que expressa a área ou a atividadecom que ele guarda relacionamento estreito.

a. Teste de software.
b. Projeto de software.
c. Segurança de dados.
d. Garantida da Qualidade do Software.
e. Desenvolvimento profissional.