cespe

Cebraspe
Centro Brasileiro de Pesquisa em Avaliação e Seleção e de Promoção de Eventos

CONCURSO PÚBLICO – TRE/RS

CARGO 2: ANALISTA JUDICIÁRIO
ÁREA: APOIO ESPECIALIZADO
ESPECIALIDADE: ANÁLISE DE SISTEMAS

Prova Escrita – Questão 1

Aplicação: 20/12/2015

PADRÃO DE RESPOSTA DEFINITIVO

Solução para o caso: sistema mais adequado e seu funcionamento básico

Nessa situação hipotética, o uso de um sistema de controle de versão distribuído seria o mais adequado para o processo de produção do *software* em equipe. Nesse sistema de controle de versão, os clientes fazem cópias completas das últimas versões dos arquivos. Dessa forma, cada área de trabalho tem seu próprio servidor e todas as operações são salvas localmente. Na prática, um servidor é designado para que seja o "repositório oficial" de modo que todos possam contribuir em um mesmo código. Somente quando há a sincronização dos arquivos, é realizada a comunicação com o servidor.

Três vantagens e três desvantagens do sistema de controle de versão escolhido

Vantagens principais:

1) Velocidade: são mais rápidos que sistemas de controle de versão convencionais (centralizados). As operações são processadas localmente, não sendo necessário acessar o servidor central para se fazerem as operações de *commit*, *log* ou *diff*.
2) Redução de custos com servidor e infraestrutura externa de rede: o servidor tem a única função de ser o repositório central do projeto, não atuando como processador centralizado de operações.
3) Aumento da produtividade e confiabilidade: uma pane no servidor não interrompe o desenvolvimento, e os repositórios dos desenvolvedores funcionam como cópias de *backup* do projeto.
4) Autonomia: o desenvolvedor pode trabalhar sem comunicação com o servidor. A conexão só é necessária para a troca de revisões com outros repositórios.
5) Distribuído: permite que o desenvolvedor trabalhe de forma colaborativa com diferentes grupos de pessoas, de diversas maneiras, simultaneamente, no mesmo projeto.

Desvantagens principais:

1) Complexidade: exige um maior conhecimento da ferramenta.
2) O controle de concorrência em arquivos binários (como figuras) é problemático, por possuírem esses arquivos um formato interno que não é fundamentado em linhas de texto.
3) Controle de mudanças centralizado.

Dois possíveis modos de sincronização de mudanças em projetos colaborativos

~~Existem duas maneiras principais de se integrarem mudanças de uma ramificação (*branching*) em outra: união e reaplicação. A maneira mais simples de integração de *branches* é a operação união. Ela executa uma junção de três vias entre os dois últimos *snapshots* (cópias em um determinado ponto no tempo) e, resumidamente, integra o conteúdo divergente entre dois *branches* com o ancestral comum deles mais recente, criando um novo *commit*.~~

~~Por outro lado, uma operação de reaplicação serve para reaplicar todas as modificações comitadas de um *branch* para outro. Ela verifica o ponto em comum entre o *branch topic* e o *master*, busca as modificações de cada *commit* do *branch topic* que não existem no *master* e faz que o *branch master* aponte para o mesmo *commit do*~~

~~*branch topic* e só depois ela reaplica cada uma das mudanças. Desse modo, a *reaplicação* garante um histórico ordenado de todos os *commits*, facilitando a organização de um projeto colaborativo.~~

Sincronização em projetos colaborativos refere-se ao conceito de transferência de mudanças entre um repositório remoto e um repositório local, sendo feita através de operações de “trazer” e “empurrar”, ou em inglês, “*fetch”* e “*push”*. A operação *fetch* basicamente atualiza as referências locais com relação às referências remotas, mas não necessariamente faz a união (merge) com o *branch* local. Por outro lado, a operação *push* incorpora as mudanças de um repositório remoto para o *branch* local.