Relatório sobre ferramentas para controle de controle de versão

Os sistemas de controle de versão são ferramentas de software que ajudam as equipes de software a gerenciar as alterações ao código-fonte ao longo do tempo. Como os ambientes de desenvolvimento aceleraram, os sistemas de controle de versão ajudam as equipes de software a trabalhar de forma mais rápida e inteligente

Os projetos de desenvolvimento de software, em geral, envolvem dezenas, às vezes, centenas de pessoas. O gerente de projeto tem o (árduo) desafio de gerenciar recursos, colaboradores, requisitos e prazos. Não é uma tarefa fácil, especialmente quando nos lembramos que os códigos-fontes são escritos e reescritos milhares de vezes.

Para piorar, todas as linhas são criadas simultaneamente por pessoas diferentes. Ou seja, em algum momento, será necessário integrar todo o trabalho da equipe, algo impossível de ser feito manualmente. É aqui que entra o controle de versão de software (*Version Control System*ou VCS).

O funcionamento do controle de versão varia em cada sistema (local, centralizado ou distribuído, os quais veremos adiante). Entretanto, em essência, um sistema de versionamento é capaz de gerenciar todos os comandos executados na edição de um arquivo, salvando a sequência de alterações, de forma independente, em um determinado repositório.

A CVS é uma das ferramentas de controle de software mais antigas no mercado. A primeira versão dela foi desenvolvida em 1968. Essa ferramenta possui como maior desvantagem o fato de ser considerada como uma tecnologia antiga. Porém, ainda é bastante utilizada por equipes de desenvolvedores, é muito simples de ser operada. Isso significa que a sua equipe pode aprender rapidamente como usar todas as funcionalidades da CVS com eficiência.

O RCS foi o 2º controle de versão de grande impacto, mas é mantido até hoje pelo GNU Project. Desenvolvido em 1982 por Walter Tichy, o RCS usava técnicas mais avançadas do que o pioneiro SCCS. Sua fragilidade estava em sua capacidade de trabalhar apenas com arquivos individuais e não com projetos inteiros.

Conhecido como SVN, o Apache Subversion é um controle de versão de código aberto desenvolvido em 2000 pela CollabNet. Atualmente, entretanto, é um software mantido como um projeto da Apache Software Foundation.

A proposta do SVN era oferecer uma estrutura semelhante ao CVS (arquitetura similar, com modelo cliente-servidor), corrigindo, entretanto, algumas deficiências do seu antecessor. O SVN, por exemplo, é multiplataforma, o que permite que clientes com diferentes SOs visualizem o mesmo servidor que mantém o repositório.

O SVN trabalha com operações atômicas e é extremamente flexível, sendo capaz de manter o histórico mesmo após a execução de comandos como mover, copiar ou renomear, uma novidade em relação aos controles de versão anteriores.

Conhecido como P4, o Perforce é uma ferramenta de controle de versão de software muito popular entre os desenvolvedores devido à sua robustez, extensa quantidade de avaliações e disponibilidade de suporte. Aliás, esse sistema tem a reputação de prover um nível de suporte fora do comum.

Diferentemente de algumas aplicações comerciais, o Perforce enfatiza a versatilidade: está disponível nas plataformas mais “obscuras” (como IBM OS/390 e Open WMS). Além disso, essa ferramenta provê algumas funcionalidades que não estão presentes no CVS. Lida muito bem como arquivos binários e suporta operações atômicas. Diferentemente do CVS, Perforce tem seu próprio modelo de segurança.

Desenvolvido desde 2005, o Git foi inicialmente concebido para o Kernel (núcleo) do Linux. Sua criação começou quando diversos desenvolvedores Kernel Linux resolveram deixar o BitKeeper, um controle de versões distribuído que passou a ser pago quando o detentor dos direitos autorais da ferramenta optou por torná-la proprietária.

É muito usado em projetos open source e possui algumas características marcantes, como concepção simples (mas poderosa), adequação a grandes projetos e velocidade. A maior parte das operações no Git necessita apenas de recursos locais para operar, o que explica sua velocidade maior do que os demais sistemas.

Não há mais servidores centrais como no SVN. Dentre alguns que usam o Git, podemos mencionar Linux Kernel, Wine e Fedora. Para assimilar bem o trabalho com essa ferramenta, o ideal é não pensar no funcionamento de outros VCSs, como Perforce ou Subversion, isso porque, apesar de ter interface similar, o Git pensa as informações de forma diferente dos outros tipos de controle de versão de software

No meio corporativo, o Subversion é uma ferramenta de controle de versão de software bastante utilizada. Ela é bastante rápida na execução das funcionalidades do sistema e ainda se mostra como uma das mais simples de ser empregada. Isso significa que com um conhecimento básico de conceitos relacionados ao controle de versão de software é possível executar comandos na ferramenta. A aprendizagem da equipe também é rápida nesse aspecto.

Um dos problemas do Subversion são as críticas relacionadas à eficácia do software. No passado, essa ferramenta apresentou problemas na hora de executar as principais funções de um controle de versão de software eficiente. Porém, as últimas versões lançadas parecem ter solucionado tudo que foi apontado como desvantagem do programa.

Mercurial é mais um controle de versão de software distribuído. Da mesma forma que o Git, o release 0.1 do Mercurial foi lançado em 2005, quando um grupo de desenvolvedores decidiu deixar de utilizar o BitKeeper. O nome faz referência ao deus mitológico cujas características eram a inteligência, a eloquência e a rapidez. De fato, o Mercurial VCS tem todos esses atributos.

O sistema foi criado para suportar o desenvolvimento de projetos de grande porte, muito além de pequenos trabalhos de designers/desenvolvedores independentes. Essa ênfase nos trabalhos de grande envergadura não significa, entretanto, que times enxutos não possam utilizar o Mercurial.

Trata-se de um modelo mais simples que o Git, com funções similares a outros sistemas distribuídos. Apesar possuir interface funcional e armazenamento eficiente, o Mercurial não foi escolhido para gerenciamento do código do Kernel do Linux. Adotou-se o Git como padrão.

Vantagens do controle de versão no desenvolvimento ágil

A força-motriz das metodologias de desenvolvimento ágil é a integração entre colaboradores, algo que um controle de versão faz com perfeição. Essa espécie de “máquina do tempo” dos arquivos oferece vários benefícios a quem atua com método ágil: