Controle de Configuração

Engenharia de Software 2021
Moisés Gomes de Carvalho

Sumário

- O problema
- Histórico
- Conceitos Básicos
- Estratégias Conhecidas
- Ferramentas
- Resumo e Conclusões

Controle de Versão

O problema

- Quando muitos desenvolvedores precisam atuar em paralelo no mesmo projeto, eventualmente eles podem reter várias cópias das diferentes versões do programa e rotulá-las apropriadamente.
- Essa abordagem simples foi usada em muitos grandes projetos de software. Embora esse método possa funcionar, é ineficiente, pois muitas cópias quase idênticas do programa precisam ser mantidas.
    - **Isso requer muita autodisciplina por parte dos desenvolvedores** e geralmente leva a erros.

Controle de Versão

O problema

- Quando muitos desenvolvedores precisam atuar em paralelo no mesmo projeto, eventualmente eles podem reter várias cópias das diferentes versões do programa e rotulá-las apropriadamente.
- Essa abordagem simples foi usada em muitos grandes projetos de software. Embora esse método possa funcionar, é ineficiente, pois muitas cópias quase idênticas do programa precisam ser mantidas.
    - **Isso requer muita autodisciplina por parte dos desenvolvedores** e geralmente leva a erros.
- Como a base de código é a mesma, também é necessário conceder permissão de leitura-gravação-execução a um conjunto de desenvolvedores, e isso aumenta a pressão de alguém que gerencia as permissões para que a base de código não seja comprometida, o que adiciona mais complexidade.
- Consequentemente, sistemas para automatizar parte ou todo o processo de controle de revisão foram desenvolvidos. Isso garante que a maioria das etapas de gerenciamento de controle de versão fiquem escondidas nos bastidores.

Controle de Versão

O problema

- Conforme as equipes projetam, desenvolvem e implantam software, é comum que várias versões do mesmo software sejam implantadas em locais diferentes e que os desenvolvedores do software trabalhem simultaneamente nas atualizações.
- Bugs ou recursos do software geralmente estão presentes apenas em certas versões (devido à correção de alguns problemas e à introdução de outros à medida que o programa se desenvolve).

Controle de Versão

O problema

- Conforme as equipes projetam, desenvolvem e implantam software, é comum que várias versões do mesmo software sejam implantadas em locais diferentes e que os desenvolvedores do software trabalhem simultaneamente nas atualizações.
- Bugs ou recursos do software geralmente estão presentes apenas em certas versões (devido à correção de alguns problemas e à introdução de outros à medida que o programa se desenvolve).
- Portanto, para fins de localização e correção de bugs, é de vital importância ser capaz de recuperar e executar diferentes versões do software para determinar em qual (is) versão (ões) o problema ocorre.
- Também pode ser necessário desenvolver duas versões do software simultaneamente: por exemplo, onde uma versão tem bugs corrigidos, mas sem novos recursos (branch), enquanto a outra versão é onde os novos recursos são trabalhados (tronco).

Controle de Versão

Histórico

A necessidade de uma maneira lógica de organizar e controlar as revisões existe há quase tanto tempo quanto a escrita existe, mas o controle de revisões se tornou muito mais importante e complicado quando a era da computação começou.

A numeração de edições de livros e de revisões de especificações são exemplos que datam da era somente para impressão.

Além do desenvolvimento de software, na prática jurídica e de negócios e em outros ambientes, é cada vez mais comum que um único documento ou fragmento de código seja editado por uma equipe, cujos membros podem estar geograficamente dispersos e perseguir interesses diferentes e até contrários .

Um mecanismo de controle de revisão sofisticado que rastreia e contabiliza a propriedade de alterações em documentos e código pode ser extremamente útil ou mesmo indispensável em várias situações situações.

Controle de Versão

Histórico

A ferramenta de atualização de software OS / 360 IEBUPDTE da IBM data de 1962, foi possivelmente um precursor das ferramentas VCS.

Um sistema completo projetado para controle de código-fonte foi iniciado em 1972, chmado de SCCS para o mesmo sistema (OS / 360).

A introdução do SCSS, publicada em 4 de dezembro de 1975, implicava historicamente que foi o primeiro sistema deste tipo - "The Source Code Control System" . IEEE Transactions on Software Engineering.

O RCS veio logo após [4], com sua versão CVS em rede. A próxima geração após o CVS foi dominada pelo Subversion [5], seguida pelo surgimento do controle de revisão distribuída (por exemplo, git).

Controle de Versão

Conceitos Básicos

- O controle de revisão gerencia as alterações em um conjunto de dados ao longo do tempo. Essas mudanças podem ser estruturadas de várias maneiras.

- Freqüentemente, os dados são considerados uma coleção de muitos itens individuais, como arquivos ou documentos e as alterações em arquivos individuais são rastreadas.

- Quando os dados que estão sob controle de revisão são modificados, após serem recuperados por check-out, isso não é em geral imediatamente refletido no sistema de controle de revisão (no repositório), mas deve ser verificado ou confirmado.

Controle de Versão

Conceitos Básicos

- Uma cópia fora do controle de revisão é conhecida como "cópia de trabalho". Como um exemplo simples, ao editar um arquivo de computador, os dados armazenados na memória pelo programa de edição são a cópia de trabalho, que é confirmada ao salvar.

- Concretamente, pode-se imprimir um documento, editá-lo manualmente e só depois inserir manualmente as alterações em um computador e salvá-lo.

- Para controle do código-fonte, a cópia de trabalho é em vez uma cópia de todos os arquivos em uma revisão particular, geralmente armazenada localmente no computador do desenvolvedor; neste caso, salvar o arquivo apenas altera a cópia de trabalho, e fazer o check-in no repositório é uma etapa separada.

Controle de Versão

Conceitos Básicos

- Se várias pessoas estão trabalhando em um único conjunto de dados ou documento, elas estão implicitamente criando ramificações dos dados (em suas cópias de trabalho) e, portanto, surgem problemas de mesclagem.

- Para a edição colaborativa simples de documentos, isso pode ser evitado usando o bloqueio de arquivos ou simplesmente evitando trabalhar no mesmo documento em que outra pessoa está trabalhando.

- Os sistemas de controle de revisão são frequentemente centralizados, com um único armazenamento de dados autorizado, o repositório e check-outs e check-ins feitos com referência a este repositório central.

- Alternativamente, no controle de revisão distribuída, nenhum repositório único é considerado o principal e os dados podem ser retirados e registrados em qualquer repositório. Ao fazer check-in em um repositório diferente, isso é interpretado como uma fusão ou patch.

Controle de Versão

Conceitos Básicos

- As revisões ocorrem em sequência ao longo do tempo e, portanto, podem ser organizadas em ordem, por número de revisão ou carimbo de data / hora.

- As revisões são baseadas em revisões anteriores, embora seja possível substituir em grande parte ou completamente uma revisão anterior, como "excluir todo o texto existente, insira o novo texto ".

- No caso mais simples, sem ramificação ou desfazer, cada revisão é baseada apenas em seu predecessor imediato, e elas formam uma linha simples, com uma única versão mais recente, a revisão ou dica "HEAD".
    - Um grafo linear.

Controle de Versão

Conceitos Básicos

- Se houver ramificação, então múltiplas revisões futuras são baseadas em uma revisão passada ou até o caso de uma revisão depender de uma revisão mais antiga que sua predecessora imediata, então o grafo resultante **é uma árvore** (cada nó pode ter mais de um filho), e tem várias revisões sem filhos ("última revisão em cada ramo").
- Por outro lado, quando uma revisão pode ser baseada em mais de uma revisão anterior (quando um nó pode ter mais de um pai), o processo resultante é chamado de merge e é um dos aspectos mais complexos do controle de revisão.
- Isso ocorre com mais frequência quando ocorrem alterações em vários ramos (na maioria das vezes dois, mas mais são possíveis), que são então mesclados (merged) em um único ramo que incorpora as duas mudanças.
- Se essas alterações se sobreporem, pode ser difícil ou impossível mesclar e exigir intervenção manual ou reescrita.

Controle de Versão

Estratégias Conhecidas - Centralizadas

Os sistemas de controle de revisão tradicionais usam um modelo centralizado onde todas as funções de controle de revisão ocorrem em um servidor compartilhado.

Se dois desenvolvedores tentarem alterar o mesmo arquivo ao mesmo tempo, sem algum método de gerenciamento de acesso, os desenvolvedores podem acabar substituindo o trabalho um do outro.

Os sistemas de controle de revisão centralizados resolvem esse problema em um dos dois "modelos de gerenciamento de origem" diferentes: bloqueio de arquivo e fusão de versão.

Controle de Versão

Estratégias Conhecidas - Centralizadas

- Operações atômicas - operação central: commit atômico

  Uma operação é atômica se o sistema for deixado em um estado consistente, mesmo se a operação for interrompida. A operação de confirmação é geralmente a mais crítica neste sentido

- Bloqueio de arquivo

  O método mais simples de evitar problemas de "acesso simultâneo" envolve o bloqueio de arquivos para que apenas um desenvolvedor de cada vez tenha acesso de gravação às cópias do "repositório" central desses arquivos

Controle de Versão

Estratégias Conhecidas - Centralizadas

- Mesclagem de versão - operação principal - Merge

  O primeiro desenvolvedor a "registrar" as alterações no repositório central sempre é bem-sucedido. O sistema pode fornecer recursos para mesclar outras alterações no repositório central e preservar as alterações do primeiro desenvolvedor quando outros desenvolvedores fizerem check-in.

  O segundo desenvolvedor que verifica o código precisará tomar cuidado com o merge, para ter certeza de que as alterações são compatíveis e que a operação não introduza seus próprios erros lógicos nos arquivos.

Controle de Versão

Estratégias Conhecidas - Centralizadas

- Baselines, rótulos e tags

  A maioria das ferramentas de controle de revisão usará apenas um desses termos semelhantes (baselines, rótulo, tag) para se referir à ação de identificar um snapshot ("rotular o projeto") ou o registro do snapshot ("tentar com o baseline X") . Normalmente, apenas um dos termos baseline, rótulo ou etiqueta é usado na documentação ou discussão; eles podem ser considerados sinônimos em alguns sistemas.

Controle de Versão

Estratégias Conhecidas - Descentralizadas

Os sistemas de controle de revisão distribuída (DRCS) adotam uma abordagem ponto a ponto, em oposição à abordagem cliente-servidor de sistemas centralizados.

Em vez de um único repositório central no qual os clientes sincronizam, a cópia de trabalho de cada par da base de código é um repositório de confiança.

O controle de revisão distribuída conduz a sincronização trocando patches (conjuntos de alterações) de ponto a ponto. Isso resulta em algumas diferenças importantes de um sistema centralizado:

Controle de Versão

Estratégias Conhecidas - Descentralizadas

Nenhuma cópia canônica de referência da base de código existe por padrão; apenas cópias de trabalho.

Operações comuns (como commits, visualização de histórico e reversão de alterações) são rápidas, porque não há necessidade de se comunicar com um servidor central.

Em vez disso, a comunicação só é necessária ao empurrar ou puxar mudanças para ou de outros pares.

Cada cópia de trabalho funciona efetivamente como um backup remoto da base de código e de seu histórico de alterações, fornecendo proteção inerente contra perda de dados.

Controle de Versão

Ferramentas

Comerciais

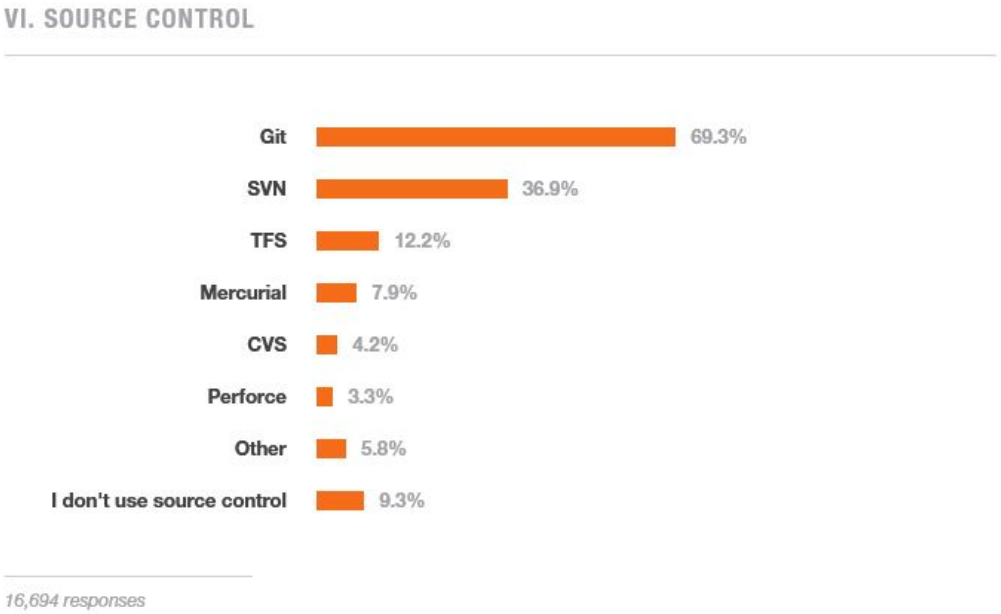
- Microsoft Visual SourceSafe (VSS)
- Rational ClearCase
- Borland StarTeam
- PVCS - Serena do Brasil

Livres

- Concurrent Version System (CVS)
- Subversion (SVN)
- Git -
- MediaWiki
- GNU CSSC
- Revision Control System (RCS)
- Bazaar
- Darcs
- Mercurial
- Monotone
- SVK

Controle de Versão

Ferramentas



Controle de Versão

Resumo e Conclusões

- Controle de Versão não é um problema novo!
- O trabalho em equipe é provavelmente a principal razão do desenvolvimento de sistemas de controle de versão de documentos
- Em teoria, deveria ser algo transparente ao usuário, mas na prática exige bastante cuidado no uso das ferramentas
- Centralizado vs Distribuído
- Diferenças entre Ferramentas?

Controle de Versão

Primeira Atividade Prática

- Executar todo o curso de GIT encontrado em
  - https://www.youtube.com/watch?v=WVLhm1AMeYE&list=PLInBAd9OZCzzHBJjLFZzRl6DgUmOeG3H0
- Fazer uma apresentação em google slides (prova de execução) e ao final adicionar slides respondendo
  - O que vc aprendeu no curso e que não conhecia?
  - O que você usa mais frequentemente no seu trabalho/projetos/disciplinas?