SOFT
DISCIPLINA: Engenharia de Software **DATA: _____/_____/_____**
AULA NÚMERO: 20 **PROFESSOR: Andrey**

APRESENTAÇÃO

O objetivo desta aula é apresentar e discutir conceitos relacionados a Gestão de configuração e Engenharia Reversa.

DESENVOLVIMENTO

Gestão de Configuração de Software

É o conjunto de atividades desenvolvidas para administrar modificações ao longo do ciclo de vida do software. Pode ser vista como uma atividade de garantia da qualidade de software.

A Gestão de Configuração de Software tenta responder às seguintes perguntas:

- O que mudou e quando mudou?
- Por que mudou?
- Quem fez a mudança?
- Pode-se reproduzir essa mudança?

A Gestão de Configuração de Software tem três grandes atividades:

- O controle de versão;
- O controle de mudança;
- A auditoria de configuração.

Divididas em:

- Identificar os produtos de trabalho que podem ser modificados, estabelecendo relacionamento entre eles;
- Definir mecanismos para administrar as diferentes versões desses produtos de trabalho;
- Controlar as modificações impostas e fazendo auditoria e preparando relatórios sobre as modificações realizadas.
- Garantir que os procedimentos e políticas para criar, modificar e testar o código estão sendo seguidos;
- Tornar acessível as informações sobre o projeto.

Elementos de um sistema de Gestão de configuração:

- Elementos de componente. Ferramentas que possibilitam acesso e gestão de cada item de configuração de software
- Elementos de processo. Procedimentos e tarefas para a gestão da modificação para todas as partes envolvidas na gestão, engenharia e uso do software.
- Elementos de construção. Ferramentas que automatizam a construção garantindo que o conjunto adequado de componentes foi montado
- Elementos humanos.

Item de configuração de software ICS

Um item de configuração de software é um elemento unitário para efeito de controle de versão ou um conjunto de elementos tratados como uma unidade única.

Pode ser composto de:

- código fonte;
- documentação;
- diagramas;
- casos de teste;
- dados e
- outros artefatos.

É descrito por:

- nome;
- descrição;
- lista de recursos;
- realização e
- número de versão.

Relacionamentos entre itens de configuração:

- Dependência: um componente utiliza funções que são implementadas em outro;
- Agregação: um componente é formado por outros;
- Realização: um componente concreto é a implementação de outro mais abstrato;
- Especialização: um componente é uma variante mais específica de outro.

Rastreabilidade

- Rastreabilidade entre itens de código;
- Rastreabilidade entre artefatos de análise e design;
- Matriz de rastreabilidade

Versões de Itens de Configuração de Software

- Normalmente é identificada por um número;
- As versões se sucedem no tempo;
- dois tipos:
  - revisão: uma nova versão de um item que vai substituir a anterior;
  - variante: uma nova versão de um item que será adicionada á configuração sem a intenção de substituir a anterior;

Configuração de software

É o estado em que o software se encontra em um determinado momento. É definida pela lista de Itens de Configuração de Software que compõem o software.

Baseline

É uma configuração de software criada para uma finalidade específica que é modificada por um processo formal de mudança.

Release

É a distribuição de uma versão do software para fora do ambiente de desenvolvimento.

Controle de Versão

O controle de versão rastreia todos os itens de configuração e mantém o controle sobre o trabalho paralelo dos desenvolvedores. Funcionalidades:

- Manter e disponibilizar cada versão já produzida para cada item;
- Ter mecanismos para disponibilizar diferentes ramos de desenvolvimento de um mesmo item;
- Estabelecer políticas de sincronização;
- Histórico de mudanças;

Políticas de sincronização:

- trava-modifica-destrava (exclusive Lock)
- copia modifica resolve (optimistic merges)

Repositório para Gestão de Modificações

Conjunto de Mecanismos e estruturas de dados para gerir modificações de forma efetiva. Deve ter as seguintes funções:

- Integridade dos dados;
- Compartilhamento de informação;
- Integração de ferramenta;
- Integração de dados;
- Imposição de metodologia e
- Padronização de documentação.

Principais Ferramentas

CVS

Mercurial

GIT

SVN (subversion)

SourceSafe (Microsoft)

Serena PVCS

ClearCase (IBM)

CVS

O CVS (*Concurrent Version System*) permite que múltiplos usuários editem os mesmos arquivos independentemente, trabalhando nas suas próprias cópias. Existe um "repositório" onde os arquivos mestres são mantidos. Cada usuário realiza "checks out" para pegar uma cópia de trabalho. Eles então editam a cópia, e realizam um "check in" ou um "commit" para armazenar suas alterações de volta no repositório, e realizam um "update" para incorporar outras alterações em suas cópias de trabalho. O CVS também mantém as versões anteriores de seus arquivos, de maneira que você possa comparar versões e reverter para uma versão anterior caso uma alteração introduza um erro.

Os repositórios CVS contêm arquivos baseados em texto como os arquivos fonte (.Java), Makefiles, e algumas documentações. Os repositórios não devem conter arquivos compilados (.class) ou arquivos texto gerados, como a documentação gerada pelo Javadoc. O seu repositório CVS deve conter apenas arquivos que não possam ser gerados a partir de outros arquivos do repositório.

Aqui está uma lista típica de passos de utilização do CVS:

1. Realize um "Update" nos seus arquivos da cópia de trabalho.
2. Adicione/edite um arquivo.
3. Realize um "Update" novamente para verificar se os arquivos foram alterados no repositório desde que você começou a editá-los. Se eles foram, as alterações irão ser intercaladas na sua cópia de trabalho automaticamente.
4. Se existir um conflito de "merge", onde duas alterações entram em conflito uma com a outra, decida manualmente qual alteração deve ser mantida.
5. Realize um "Commit", também conhecido como "Check in", no arquivo.

Cada usuário terá sua própria cópia de trabalho em algum lugar. Nós sugerimos que você crie um conjunto de diretórios ~/projeto/groups/se*NNM*/*username*.

Utilização Básica

Atualizando seus arquivos

Para atualizar sua cópia de trabalho com a última versão revisada do repositório (mantendo qualquer alteração que você tenha feito em sua cópia local), use:

```
cvs update
```

O CVS tentará intercalar quaisquer alterações feitas, por você e pelos outros, desde o seu último cvs update. Se alguma de suas alterações entrar em conflito com as alterações dos outros, o cvs update irá lhe informar, e o arquivo fonte será modificado para incluir ambas versões de qualquer porção conflitante (as suas e as outras do repositório), no seguinte formato:

<<<<<<< *filename*

SUA VERSÃO

=======

VERSÃO DO REPOSITÓRIO

>>>>>>> *número de revisão do repositório de versão*

Você deverá solucionar o conflito pela edição do arquivo, removendo as marcações, e deixando a versão de código que você preferir (ou combinando as duas manualmente). (Procurar por "<<<" até que você solucione todos os conflitos é geralmente uma boa idéia.) Uma vez que você tenha resolvido todos os conflitos, você pode seguramente armazenar o arquivo de volta no repositório.

Note que o CVS trabalha no estilo linha-por-linha. Isto é, ele só sabe se uma linha inteira foi alterada, adicionada, ou apagada.

(o cvs update exibe o estado de quaisquer arquivos do seu diretório de trabalho que tenham sido alterados ou sejam diferentes daqueles do repositório com um flag de uma letra: "U" significa que ele foi substituído pela última cópia do repositório. "M" significa que sua cópia de trabalho é diferente da última versão do repositório, e que o merge foi executado com sucesso. "C" significa que existem conflitos de merge.)

Persistindo, adicionando, e removendo arquivos

Para persistir um arquivo do repositório que você tenha editado, use:

```
cvs commit -m "a log message" filename
```

Se você omitir o campo *filename*, o CVS irá armazenar todos os arquivos do diretório corrente de novo no CVS. O campo *message* é uma mensagem de log que permite que você se informe sobre as alterações sem precisar olhar no código. Elas são incrivelmente úteis, e você deverá sempre fornecer uma mensagem de log descritiva; nada do tipo, "Foram consertados alguns erros." Se você omitir o flag -m, o CVS irá fornecer um prompt para uma mensagem de log.

Identificando alterações

Para verificar o log de alterações, que é uma lista de mensagens utilizadas no momento da realização do "check in" das alterações,

cvs log *filename*

Para verificar diferenças entre a cópia de trabalho e a última cópia do repositório:

cvs diff *[filename]*

Omita o campo *filename* para verificar as diferenças entre todos os arquivos. Use o flag -r1.xx para comparar com uma revisão em particular, e use dois flags -r para comparar duas versões uma com a outra.

Engenharia reversa e Reengenharia.

Engenharia Reversa para software é o processo de analisar um programa num esforço para criar uma representação de programa em um nível de abstração maior do que o código fonte. A engenharia reversa é o processo de recuperação do projeto.

Elementos da Engenharia Reversa:

- Nível de abstração: conforme o nível de abstração aumenta, mais compreensíveis se tornam as informações.
- Completude do Processo: refere-se ao nível de detalhes que é fornecido em cada nível de abstração.
- Interatividade:
  - refere-se ao grau de participação do ser humano no processo de engenharia reversa.
  - conforme o nível de abstração aumenta, a interatividade deve aumentar ou a completitude será prejudicada
- direcionalidade:
  - se a direcionalidade tem sentido único, toda informação extraída a partir do código fonte é usada durante as atividades de manutenção
  - se a direcionalidade tem sentido duplo, a informação é usada para "alimentar" uma ferramenta de reengenharia.

Reengenharia é o processo de recuperar as informações de projeto de um software existente e usar essas informações para alterar ou reconstituir o sistema, num esforço para melhorar sua qualidade global. Na maioria dos casos o sistema que sofreu reengenharia reinplementa a função do software existente, mas ao mesmo tempo adiciona novas funcionalidades e/ou melhora o desempenho global.

BIBLIOGRAFIA BÁSICA

WAZLAWICK, R. S.. Engenharia de Software: Conceitos e Práticas. Elsevier. Rio de Janeiro, 2013.

PRESSMAN, R. S.. Engenharia de Software. 6a. ed. McGraw-Hill. 2006

CARVALHO, A. M. B. R.; CHIOSSI, T. C. S.. Introdução à engenharia de software. Editora da Unicamp. 2001.

OLIVEIRA, M. Técnicas de Testes Caixa-Preta. Apresentação no II Encontro Brasileiro de Teste de Software, outubro 2007.