Homologacao de Artefatos

Nesta pagina

Informacao passo-a-passo sobre o processo de homologacao de artefatos existentes dentro do repositorio versionado.

Usuarios DVCS

O procedimento mostrado aqui eh baseado no uso do software Subversion, um classico repositorio cliente-servidor.
Caso voce voce utilize um repositorio distribuido (DVCS) como Mercurial, Git ou outro, talvez este processo nao seja o mais adequado para voce.

Veja tambem

Versionamento de Artefatos para informacoes sobre o controle de numeracao de artefatos versionados.

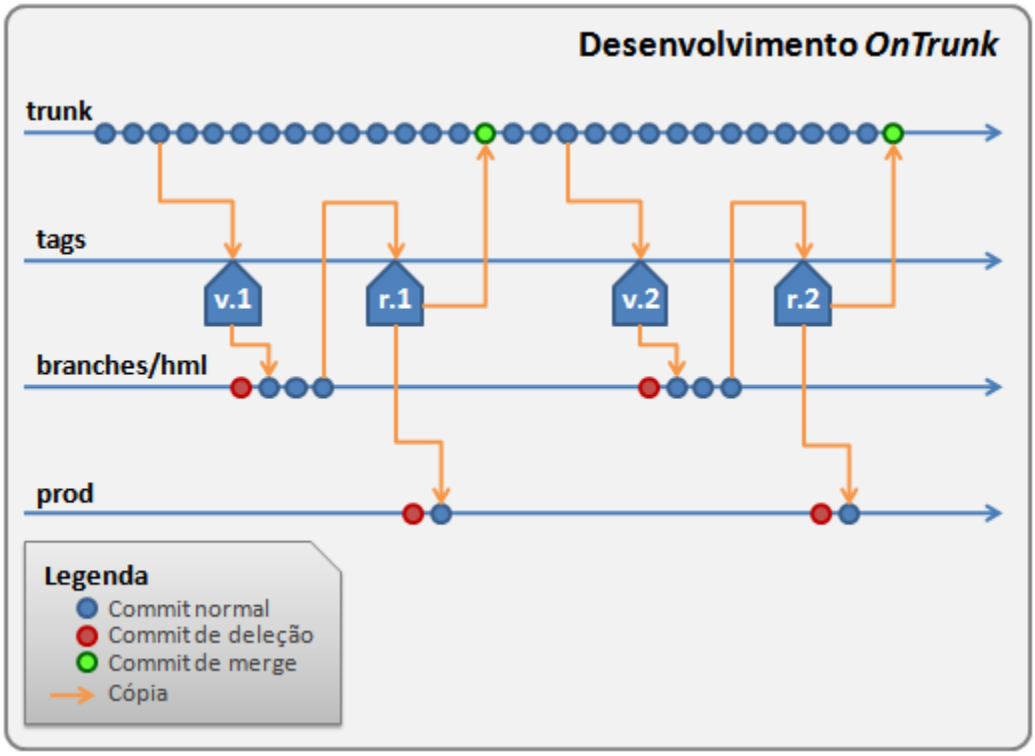
1. Estrutura de Diretorios

O controlde de versoes de sistemas e seus respectivos codigos-fonte é realizado pelo software Subversion, seguindo a estrutura classica de branches, tags e trunk, conforme:

- ***trunk/***: Contem, a qualquer momento, o codigo-fonte compilavel do projeto pronto para ser colocado em homologacao. Todos desenvolvedores podem escrever neste diretorio. Sobre este diretorio é executado o servico de Integracao Continua, gerando em intervalos regulares a ultima versao de desenvolvimento disponivel do projeto.
- ***tags/***: Sao snapshots do projeto, e uma vez criados nao devem ser alterados. Somente podem escrever neste diretorio usuarios do grupo da qualidade (QA). Por padrao, as tags seguem o formato de nome *aaaa.mm.dd - versao*, onde "aaaa.mm.dd" representa a data de criacao da tag e "versao" um valor representando a versao logica do projeto frente ao gerenciamento do mesmo; por exemplo "tags/2009.12.01 - versao 0.1" ou entao "2009.12.01 - sprint 44". Existem dois tipos de tag, as tags de entrada "V" (version - com codigo de desenvolvimento para analise e homologacao) e tags de saida "R" (release - com codigo homologado que tanto vai para a producao, quanto volta para o desenvolvimento fazendo merge).
- ***branches/***: Sao usados para trabalhos esporadicos para de correcao de codigo de producao ativo ou antigo. Todos podem escrever aqui. Para cada correcao, um branche novo eh criado a partir de uma tag "R" de producao (a ultima ou uma antiga). A correcao (commits) ocorrem aqui ate que o codigo possa ser homologado. Nesse caso, a equipe de QA copia esse codigo para uma nova tag "V" e o ciclo de homologacao acontece, ate que uma tag "R" seja produzida. Se o branche de correcao equivaler ao ultimo codigo de producao (ativo), entao a saida do codigo "R" deve voltar para o desenvolvimento (merge), caso contrario, eh facultativo esse retorno para o desenvolvimento (uma vez que era uma versao antiga do software, talvez esse problema ja tenha sido trabalhado no fluxo trunk/, ou mesmo esse problema nao ocorre nas versoes mais novas ou ativa do sistema).

2. Fluxo de Promocao de Codigo-fonte de Desenvolvimento ate Producao

A figura abaixo mostra o fluxo de trabalho tradicional para promocao de artefatos em linha de desenvolvimento, passando por homologacao ate entrada em producao e, eventual ajuste corretivo pos-producao.



2.1.1. Caso de Uso

Objetivo	Realizar todo o fluxo de criacao, alteracao e remocao de codigo-fonte ate entrada em producao, habilitando pontos de checagem para posterior volta de versao.
Ator principal	QA
Atores envolvidos	Desenvolvimento, Usuario, Administador
Resumo	Seguir o fluxo de operacoes: trunk/ (commits de desenvolvimento) > tag "v" (copia do trunk) > hml/ (commits de qualidade) > tag "r" (copia da homologacao) > (merge para desenvolvimento) trunk/ & (copia para producao) prod/ > (branche para correcao) > tag "r".

2.1.2. Premissas

1. Todo o desenvolvimento ocorre na pasta trunk/ do projeto.
2. Existe integração contínua diária automática na pasta trunk/.
3. Commits na pasta trunk/ não podem quebrar a compilação do projeto.
4. Toda equipe de desenvolvimento pode realizar commits na pasta trunk/.
5. Existe um branche de homologação, chamado hml/.
6. Existe integração contínua no branche de homologação, que é iniciada manualmente pelo homologador do sistema no momento de iniciar um novo processo de homologação.
7. O branche de homologação visa conter código de estabilização e validações finais antes da entrada de produção.
8. Todo codigo de homologação deve ser marcado (tageado) na sua entrada (com a tag "v") e na sua saída (com a tag "r").
9. O código de homologação é proveniente do código de desenvolvimento, devidamente tageado.
10. Toda homologação de sucesso deve gerar uma tag "r", que visa marcar código estavel para entrada em produção.
11. Existe um branche de produção, chamado prod/.
12. Não existe integração contínua para código de produção.
13. O branche de produção visa conter código de produção, efetivamente validado pela homologação.
14. Os unicos commits validos no branche de produção deve ser os relativos ao processo de merge, proveniente da tag "r".
15. Na eventualidade de existir correcoes em codigo de producao, um branche especifico deve ser criado.
16. A finalizacao de uma correcao de producao no branche relacionado deve ser encerrada com uma tag "r" e, a partir disso, um processo de "build" manual deve ser realizado.

2.1.3. Fluxo Principal

1. Desenvolvimento diario
    a. Programador realizar checkout da pasta trunk/.
    b. Programador realizar alteracoes e commits na pasta trunk/.

2. Promocao para homologacao (equipe e usuarios acordar momentos adequados, exemplo, a cada 15 dias)
    a. QA criar tag "v", marcando o incio de revisao de homologacao.
    b. QA copiar ultima versao da pasta trunk/ para dentro da tag "v".
    c. QA apagar conteudo da pasta hml/.
    d. QA copiar conteudo da tag "v" para dentro da pasta hml/
    e. QA realizar checkout da pasta hml/.
    f. QA realizar testes unitarios e integrados, alteracoes corretivas e commits na pasta hml/, se necessario.
    g. QA invocar integracao continua da homologacao.
    h. Usuario testar funcionalmente aplicacao em ambiente de homologacao.
    i. Repetir 08 até encerrar o ciclo de correções.
    j. Usuario protocolar aceite de homologacao.
    k. QA atualizar arquivo "release-notes" do projeto na pasta hml/.
    l. QA realizar commit final de revisao de homologacao na pasta hml/.
3. Promocao para producao
    a. QA criar tag "r", marcando o fim de revisao de homologacao.
    b. QA apagar conteudo da pasta prod/.
    c. QA copiar conteudo da pasta "r" para dentro da pasta prod/.
    d. QA comunicar Administrador de nova versao de producao disponivel.
    e. Administrador realizar deployment de producao.
4. Reintegracao de desenvolvimento
    a. QA realizar checkout da pasta trunk/.
    b. QA realizar merging da pasta hml/ para pasta trunk/ local.
    c. QA resolver conflitos na pasta trunk/ local.
    d. QA realizar commits de resolucao de conflitos na pasta trunk/.
    e. Programador realizar update na pasta trunk.

3. REFERENCIAS

http://svnbook.red-bean.com/en/1.0/ch04s04.html#svn-ch-4-sect-4.1

http://daptivate.com/archive/2008/08/28/subversion-best-practices-for-web-applications.aspx

http://svn.collab.net/repos/svn/trunk/doc/user/svn-best-practices.html (leitura obrigatoria)

http://stackoverflow.com/questions/570071/subversion-branch-trunk-best-practice-keeping-branch-up-to-date

https://collaborate.txcorp.com/collaborate/support/best-practices-branching-tagging-and-releasing-using-subversion/ (interessante abordagem para integracao continua)

http://svnbook.red-bean.com/en/1.0/ch04s04.html#svn-ch-4-sect-4.1 (dicas de uso do SVN, com enfase para ressurreicao de artefatos)