UNICESUMAR – UNIVERSIDADE CESUMAR
CENTRO DE CIÊNCIAS EXATAS TECNOLÓGICAS E AGRÁRIAS
CURSO DE GRADUAÇÃO EM ENGENHARIA DE SOFTWARE

ECLIPSE

Danielle Sayuri Kitagawa

MARINGÁ – PR
2020

ECLIPSE

No ambiente de programação, os membros da equipe podem utilizar seu workbench, para fazer o desenvolvimento/alteração do código, e para fazer a publicação, é feito em um repositório CVS. Este modelo de branch, permite que os projetos sejam um pouco isolados uns dos outros(workbench), porém são interdependentes.

De forma geral, para fazer as alterações no workbench, deve ser feito as atualizações, para caso tenha alterações mais recentes. Posteriormente, deve ser feito um merge no local workbench, e assim pode ser feito um commit para a branch. Quando os membros da equipe fizerem a atualização, receberão as alterações feitas.

1.1 Pull/Merge request

Para fazer as alterações de um documento, é preciso fazer o commit das alterações escolhidas, e fazer um push, para enviar para um repositório remoto. Porém, para fazer um pull request/merge request, é necessário a utilização de outra ferramenta (por exemplo o github), e partir disso pode ser feita a solicitação de pull.

1.2 Merge e conflitos

O merge é a "fusão" entre as alterações de uma branch/tag, ou seja, ocorre a união das ramificações. Porém, podem ocorrer alguns problemas, e para isso há algumas estratégias que podem ser utilizadas, por exemplo, pode ocorrer um conflito, quando for feita uma alteração local, para uma branch mais recente, ou seja, antes de commitar na branch não foi feito um pull.

O que o usuário pode fazer:

- Descartar as alterações locais, e fazer a atualização mais recente(abortar)

- `Podem ser feitas as alterações locais para a branch, de forma que o usuário pode sobrescrever as alterações de outra pessoa.

- Normalmente, é mais utilizado esta opção, pois é basicamente mesclar a alterações locais com as branch. Sendo o Synchronize view quem indica os recursos que estão em conflitos. Assim, você poderá escolher quais mudanças serão aceitas, sendo que geralmente ocorre uma fusão entre as alterações.

Os versionamentos tem como objetivo salvar um snapshot do estado atual do projeto em um determinado período. Sendo que quando ele é versionado é criado uma cópia que terá as alterações. O controle de versão cria um histórico com todas as versões do projeto, sendo que os projetos podem ser controlados a partir do workspace ou da branch. Há dois métodos de versionamento, o primeiro em que o projeto é criado como uma versão de uma branch ou é criado um projeto como uma versão do workbench(mais indicado).

1.3 Teste

A ferramenta JUnit é muito útil para automatização de testes unitários/regressão para projetos de Java, podendo ser integrado ao eclipse, a partir da configuração da biblioteca JUnit.

1.4 Code review

O Eclipse possui uma opção de History View que é uma ótima forma, para quando se precisa ter um controle sobre o histórico de alterações de uma determinado projeto, para fazer um checkout ou pesquisa de um determinado commit, criar um branch/tag/patch baseado em um commit. Também é possível utilizar o plug-in de revisão, o EGerrit é uma integração do Gerrit no eclipse. Sendo que esta ferramenta busca com essa integração, permitir melhores recursos de revisão, também votação/envio/abandono do que foi revisado, e também permite que o usuário possa comentar no arquivo.

1.5 Estratégia de branching

Para estratégia de branching, o eclipse sugere a utilização do recurso "Git integration on Eclipes - GitFlow support", sendo a integração do Gitflow no Egit

do eclipse, que permite um controle sobre as branchs, features, hotfix e as features. Sendo esta estratégia de Git Flow uma ótima forma de coordenar as branches facilitando assim, a resolução de conflitos e evitando também problemas quando a repetição de bugs.

1.6 Rastreabilidade

O git também fornece um repositório que armazena os objetos desenvolvidos durante o projeto, sendo que esses objetos possuem uma identificação a partir de hash SHA-1. Sendo muito vantajoso, por questões de segurança, mas também para identificação do objeto do commit com seu conteúdo e histórico. Esse repositório, é uma ótima forma de rastrear o commit, pois garante o histórico deste. O repositório é quem identifica as mudanças/estágios do arquivo, sendo que podem ser tracked, quando foram adicionados recentemente ao índice(arquivo em binário, com todas as informações do objeto) ou o último snapshot(status), podendo ser unmodified, modified, staged. Ou podem ser untracked, se não foram adicionados aos índices ou relacionados ao último snapshot.

Um plug-in encontrado no eclipse marketplace é o Mylyn que é uma integração da IDE Eclipse com Redmine, que é uma ótima forma para manter a rastreabilidade do requisito e o código, sendo possível rastrear o commit, a visualização do link para commits principal, como o autor, mensagens, lista de branches do commit e entre outras.

REFERÊNCIA BIBLIOGRÁFICA

ECLIPSE FOUNDATION. **EGit/User Guide**. Disponível em < https://wiki.eclipse.org/EGit/User_Guide#Show_Revision_Comment>. Acesso dia 15 de setembro de 2020.

ECLIPSE MARKETPLACE. **MDT: Mylyn contex Connector**. Disponível em < https://marketplace.eclipse.org/content/mdt-mylyn-contex-connector>. Acesso dia 15 de setembro de 2020.

ECLIPSE. **Eclipse documentation – Current Release**. Disponível em < https://help.eclipse.org/2020-06/index.jsp>. Acesso dia 15 de setembro de 2020.