Project Management

Overview

- How to manage a project?
- What is software configuration management?
- Version control systems
- Issue tracking systems
- Continuous integration

What is Project Management?

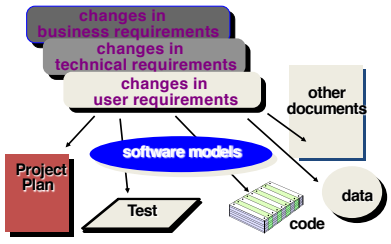
- Effective project management focuses on the 4 P's:
  - People: the most important element
    - recruiting, training, performance management
  - Product: the software to build
    - Project objectives, scope, alternative solutions
  - Process: define activities and tasks involved
    - Milestones, work products, QA points
  - Project: progress control
    - Planning, monitoring, controlling

The "First Law"

- **No matter where you are in the system life cycle, the system will change, and the desire to change it will persist throughout the life cycle.**

Bersoff, et al, 1980

What Are These Changes?



Software Configuration Management (SCM)

- Definition
  - The task of tracking and controlling changes in software
- SCM repository
  - tools that allow developers to effectively manage changes
    - Version control system
    - Issue tracking system
- CI servers

Version Control System

What Is Version Control System?

- VCS, also known as Revision Control System
- To manage changes to documents, programs, large websites, and other collections of information
  - CVS, SVN, Mercurial, GIT

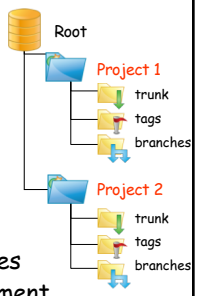
What Do We Mean by "Manage Changes" ?

- What changes have been made?
- Why are the changes made?
- Who makes the changes?
- Can we redo/undo some changes?
- Can we branch the project?

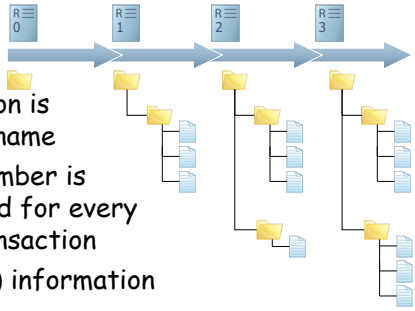
Subversion Version Control System (SVN)

Subversion Repository Layout

- One SVN server can hold many repositories
- One repository can hold many projects
- One project contains
  - Trunk: Main line of development
  - Tags: Markers to highlight notable revisions—major releases
  - Branches: Side lines of development



Each project has multiple revisions

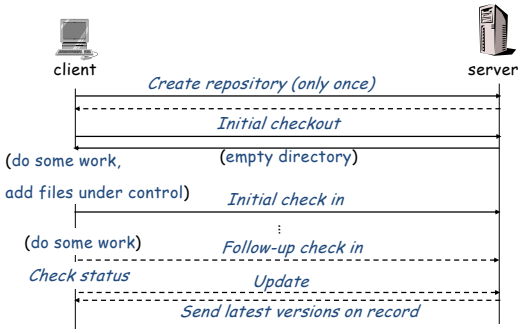


- Each revision is assigned a name
- Revision number is incremented for every commit transaction
- Delta (diff) information is recorded

Basic Features of a Repository

- Keep the history of all changes to files and directories
  - You can check in new versions
  - You can recover any previous version
- Access control
  - Read/write permission for users
- Logging
  - Author, date, and reason for a change

Typical Workflow



Additional Features

- Diff
- Branch
- Merge

Diff

- To display the differences between two revisions
  - What has been changed?
  - Add or delete a line of text
  - No update, or move

```
Version 1:          Version 2:
   x = 0;              x = 1;
   y = 1;              y = 1;
          Diff:
             - x = 0;
             + x = 1;
```


Key Points about Diff

- A key operation of version control systems
- A lot of features are based on diff
  - Save new versions
  - Recover a prior version
  - Patch
- We use Diff(v1, v2) to represent changes on v1 for v2
  - Diff(v1, v2) != Diff(v2, v1)

Diff: a Real Example

```
Index: trunk/compiler/org/eclipse/jdt/internal/compiler/ast/Expression.java
===================================================================
--- trunk/compiler/org/eclipse/jdt/internal/compiler/ast/Expression.java        (revision 9042)
+++ trunk/compiler/org/eclipse/jdt/internal/compiler/ast/Expression.java        (revision 9043)
@@ -223,7 +223,7 @@
                                this.implicitConversion = (runtimeTimeType.id << 4) + compileTime
                                break;
                        default : // regular object ref
-//                             if (compileTimeType.isRawType() && runtimeTimeType.isParameterize
+//                             if (compileTimeType.isRawType() && runtimeTimeType.isBoundParamet
//                                 scope.problemReporter().unsafeRawExpression(this, compileTime
//                             }
                }
```

Start line in the old version — Start line in the new version

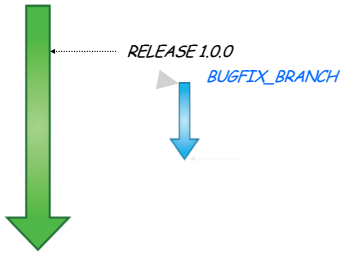
- svn diff -r v1:v2 filename
- "+": added lines, "-": deleted lines
- Some unchanged lines are shown to indicate program context

Changes Detected by Diff

- Addition/Deletion of directories
- Addition/Deletion of files
- A renamed file is reported as a separate addition and a separate deletion
- Addition/Deletion of lines

- Scenario
  - You deliver a great product to your customers: REL-1.0.0
  - Your development team continue adding new features on the trunk
  - Customers report a major bug in the product and ask for a fix
  - What do you do?

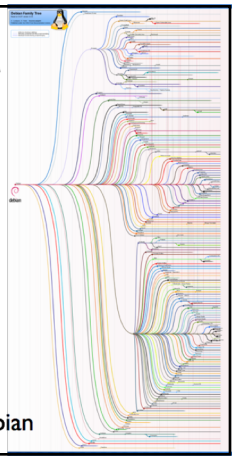
Branch and patch separately!



- svn copy path/to/trunk path/to/branch

Other reasons to branch

- Separate branches for
  - Tentative new features
  - Different products
  - Different teams
  - Different releases



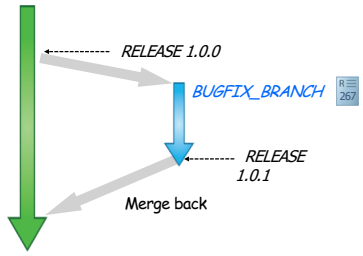
Debian

Pros and Cons of Branch

- Pros
  - Separation of concerns among teams and developers
  - Parallel version history without interference between branches
- Cons
  - Branches may diverge a lot
  - Hard to propagate changes across branches

- Scenario
  - After fixing the major bug on a branch, you have to apply the same/similar changes to the trunk
  - What do you do?

Merge back the patch!



- svn merge –reintegrate path/to/branch

What can happen when merging?

- Conflict
  - Two people edit the same file

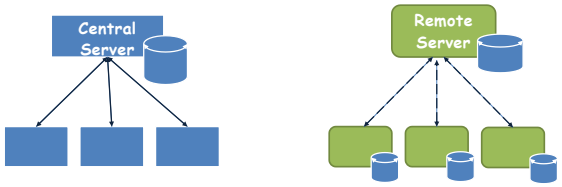
```
void f(int i) {
<<<<<<<< .mine
int j = 3;
========
int j = 4;
>>>>>>>> .r13
```

  - Resolve the conflict manually and checked in again

Distributed Version Control: GIT

- Everyone has their own local version control repository
  - Like a local branch of the project
  - Remote updates and commits are like branch merge
  - Local commits used to backup projects
  - Github allows developers to contribute by working on branches

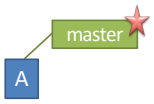
Centralized VC vs. Distributed VC[3]



Git Initialization

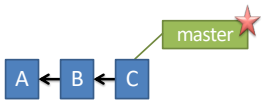
```
C:\> mkdir CoolProject
C:\> cd CoolProject
C:\CoolProject > git init
Initialized empty Git repository in
C:/CoolProject/.git
C:\CoolProject > notepad README.txt
C:\CoolProject > git add .
C:\CoolProject > git commit -m 'my first commit'
[master (root-commit) 7106a52] my first commit
 1 file changed, 1 insertion(+)
 create mode 100644 README.txt
C:\CoolProject > git remote add origin remote
repository URL
# Sets the new remote
C:\CoolProject > git push origin master
# Pushes the changes in your local repository to
the remote repository
```


Git Branch & Merge



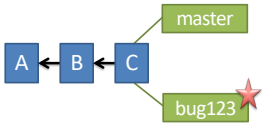

```
> git commit –m ‘my first commit’
```

Branches Illustrated



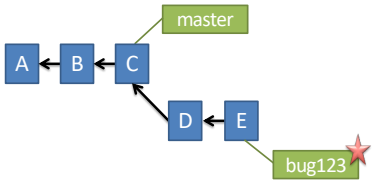

```
> git commit (x2)
```

Branches Illustrated



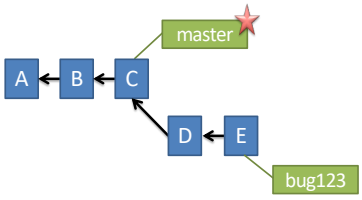

```
> git checkout –b bug123
```

Branches Illustrated



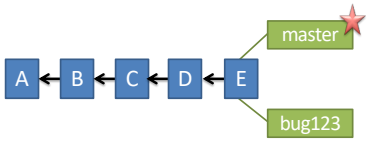

```
> git commit (x2)
```

Branches Illustrated



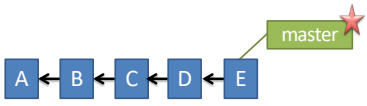

```
> git checkout master
```

Branches Illustrated




```
> git merge bug123
```

Branches Illustrated



Tips for Version Control

- Small commits
  - Check in logically relevant changes as a commit
- Write meaningful commit messages
  - Facilitate change understanding, applying, and reverting
- Avoid commit noise
  - Commit compliable or even deliverable code

Issue Tracking System

What Is Issue Tracking System?

- ITS, also known as trouble ticket system, support ticket, request management, or incident ticket system
- Manages and maintains lists of issues, as needed by an organization
  - To create, update, and resolve reported issues by customers or developers
  - Bugzilla, JIRA

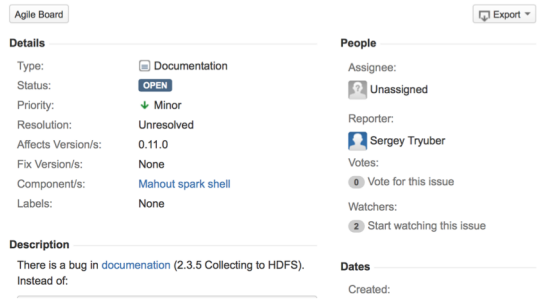
What Do We Mean by "Issues"?

- A unit of work to accomplish an improvement in a system
- It could be
  - a bug
  - a requested feature
  - a patch
  - missing documentation, …

Why Do We Need Issue Tracking?

- Developers need communication while making changes
  - Mailing List
    - Hard to manage, come with all other mails
    - Not well organized
  - Forum
    - Categorized by topic
    - Notify people when a reply is posted
    - No track to code and issue status

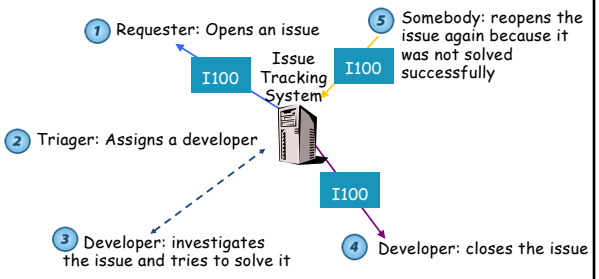
What Is Included in An Issue?



Basic Features

- Structurally describe issues
  - Solving status, severity levels
- Track status of the issue
- Assign a unique ID to each issue
  - Some system automates connection between commit and issue via issue ID

Typical Workflow



Resolution of An Issue

- Fixed
  - A bug is fixed, a feature is added, a patch is applied
- Invalid
  - Bug cannot be reproduced, features do not make sense, patch is not correct
- Duplicate
  - It is a duplicate of an existing issue
  - Get merged with the other issue

Resolution of An Issue

- Won't fix
  - The developers decide not to fix the bug or accommodate the new feature
  - Limited human resource, lack of essential information to reproduce a bug, lack of expertise

Issue Tracking & Version Control

- Many project hosting websites include issue tracking systems
  - Google Code
  - Github
  - BitBucket
  - Sourceforge

Continuous Integration

Martin Kropp[1]
Modified by Na Meng

Overview

- Why integration?
- What is Continuous Integration (CI)?
- Continuous Integration process
- CI infrastructure
- CI tools

49

Integration

- Integration occurs when changes are merged with the source code repository

50

Broken Integration

- You have a broken integration when:
  - Source code server does not build successfully
  - Shared component works in one system, but breaks others
  - Unit tests fail
  - Code quality fails (coding conventions, quality metrics)
  - Deployment fails

51

Manual Integration

- Integration becomes expensive
  - if made manual (build, test, deployment, …)
  - if too few checkin's (months or years…)
  - if integration problems and bugs are detected too late
- Reduces desire to refactor
  - long time between integration increases risk of merge

52

What is Continuous Integration?

- "*Continuous Integration is a software development practice where members of a team integrate their work frequently, usually each person integrates at least daily - leading to multiple integrations per day. Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible.*"

http://martinfowler.com/articles/continuousIntegration.html

53

Why continuous integration ?

- Maintain a code repository
- Automate the build
- Make the build self-testing
- Everyone commits to the baseline every day
- Every commit (to baseline) should be built
- Keep the build fast

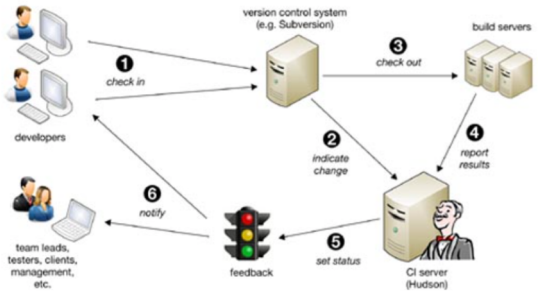
54

Why continuous integration ? (cont'd)

- Test in a clone of the production environment
- Make it easy to get the latest deliverables
- Everyone can see the results of the latest build
- Automate deployment

55

CI Workflow



56

Realizing Continuous Integration

- Monitor a VCS repository for changes
  - If changes are found, then start the build
- Build your application
  - through your existing Ant or Maven scripts
- Run your xUnit Test suite
- Run code audit tools
  - Checkstyle, code coverage, …

57

Realizing Continuous Integration (cont'd)

- Report on the build results
  - Send formatted email notifications
  - Publish results to a website
- (Optionally) Publish the application
- Configuration is through a central XML file

58

The Agile Process

- Continuous Integration is only one aspect of an overall process. For it to work best, you need to
  - Plan iteratively
    - Schedule regular releases with evolving levels of functionality
  - Implement incrementally
    - Identify and implement small work tasks
    - refactor if necessary

59

The Agile Process (cont'd)

- Report proactively
  - Identify exactly the contents (CIs) of any build, in both file and content
  - Automate reports!

60

CI Benefits

- Reduced Risks
  - Always aware of current status of the project
  - Less time spent investigating integration bugs
    - Integrating testing performed early
    - Integration bugs caught early
  - Less time wasted because of broken code in VCS
    - Broken builds caught early

61

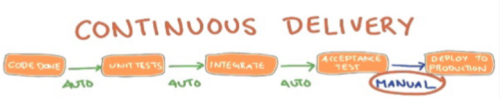
CI Benefits (cont'd)

- Prove your system can build!
- Increase code quality with additional tasks

62

Continuous Delivery

- Continuous Delivery is the ability to get changes of all types—including new features, configuration changes, bug fixes and experiments—into production, or into the hands of users, *safely* and *quickly* in a *sustainable* way.



63

Continuous Deployment

- The next step of continuous delivery: Every change that passes the automated tests is deployed to production automatically. Continuous deployment should be the goal of most companies that are not constrained by regulatory or other requirements.

64

CI Obstacles [2]

- 1. Individuals may see CI counterproductive
  - Product managers want to launch new features
  - Project managers want the team to meet the deadline
  - Programmers want to fix the bug they are working on
  - It seems that keeping the build is an extra burden on people
  - Solution: team leaders help employees understand the costs and benefits of CI

65

- 2. Tough to integrate CI into an existing development flow
  - Solution
    - Give enough time for people to develop their new workflow
    - Ensure them that the company has their backs even if things might not go very smoothly at the beginning

66

- 3. Requiring developers of writing more test cases
  - Solution
    - Emphasize that writing test cases from early on could save a lot of time for your team and ensure high test coverage of your product
    - Embed the idea in the company culture that test cases are as valuable assets as the codebase itself.

67

- 4. Developers ignoring error messages
  - Developers ignore or mute the overwhelming amount of CI notifications, and may miss the updates that are relevant to them
  - Solution
    - Only send critical updates
    - Only send the notification to developers who are in charge of fixing it

68

- 5. Creating Fear-Driven Development
  - Being afraid of breaking the build or not passing tests could build up pressure and fear in individuals.
  - Solution
    - Make team members consider failed test as positive results, the earlier their tests fail the earlier they can resolve problems.

69

Reference

[1] Martin Kropp, Continuous Integration, https://web.fhnw.ch/plattformen/swc/Literatur e/10-continuousintegration.pdf. Last visited: 1/18

[2] 5 Challenges That Can Break Your Continuous Integration Efforts, https://medium.com/flow-ci/5-challenges-that-can-break-your-continuous-integration-efforts-94aded600b3d. Last visited: 1/18

[3] Mark Groves, Introducing Git version control into your system, PPT

70