Project Management

Overview

• How to manage a project?
• What is software configuration management?
• Version control systems
• Issue tracking systems
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.

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
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 add 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

Create repository (only once)

Initial checkout

Initial check in

Update

Send latest versions on record

N. Meng, L. Zhang

13
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

<table>
<thead>
<tr>
<th>Version 1:</th>
<th>Version 2:</th>
</tr>
</thead>
<tbody>
<tr>
<td>x = 0;</td>
<td>x = 1;</td>
</tr>
<tr>
<td>y = 1;</td>
<td>y = 1;</td>
</tr>
</tbody>
</table>

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 $\text{Diff}(v_1, v_2)$ to represent changes on $v_1$ for $v_2$
  - $\text{Diff}(v_1, v_2) \neq \text{Diff}(v_2, v_1)$

Diff: a Real Example

- 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

Branch

• 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
- Where to put the major development, branch, trunk, both?
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

Merge

• 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
    
```c
void f(int i) {
<<<<<<<<<<< .mine
int j = 3;
<<<<<<<
int j = 4;
>>>>>>> .rl3

```  – 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[1]
Git Initialization [1]

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 [1]

> git commit -m ‘my first commit’
Branches Illustrated [1]

> git commit (x2)

Branches Illustrated [1]

> git checkout -b bug123
Branches Illustrated

> git commit (x2)

Branches Illustrated

> git checkout master
Branches Illustrated

> git merge bug123
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?

[Image of an issue tracking board with details and descriptions]
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

1. Requester: Opens an issue
2. Triager: Assigns a developer
3. Developer: Investigates the issue and tries to solve it
4. Developer: Closes the issue
5. Somebody: Reopens the issue again because it was not solved successfully
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

- 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

Reference

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