Manage quality processes with **Bugzilla**

Bugzilla in a Nutshell

- An open-source bugtracker and testing tool initially developed by Mozilla.
  - Initially released by Netscape in 1998.
  - Similar to many in-house bug-tracking and testing tools developed in leading IT firms.
  - Have been used widely in open-source code and developments
- Developed on Perl. It is built on,
  - A database, usually MySQL, PostgreSQL, or Oracle;
  - Web server, i.e. Apache, or Microsoft IIS.
  - A suitable SMTP server.

Bugzilla in a Nutshell

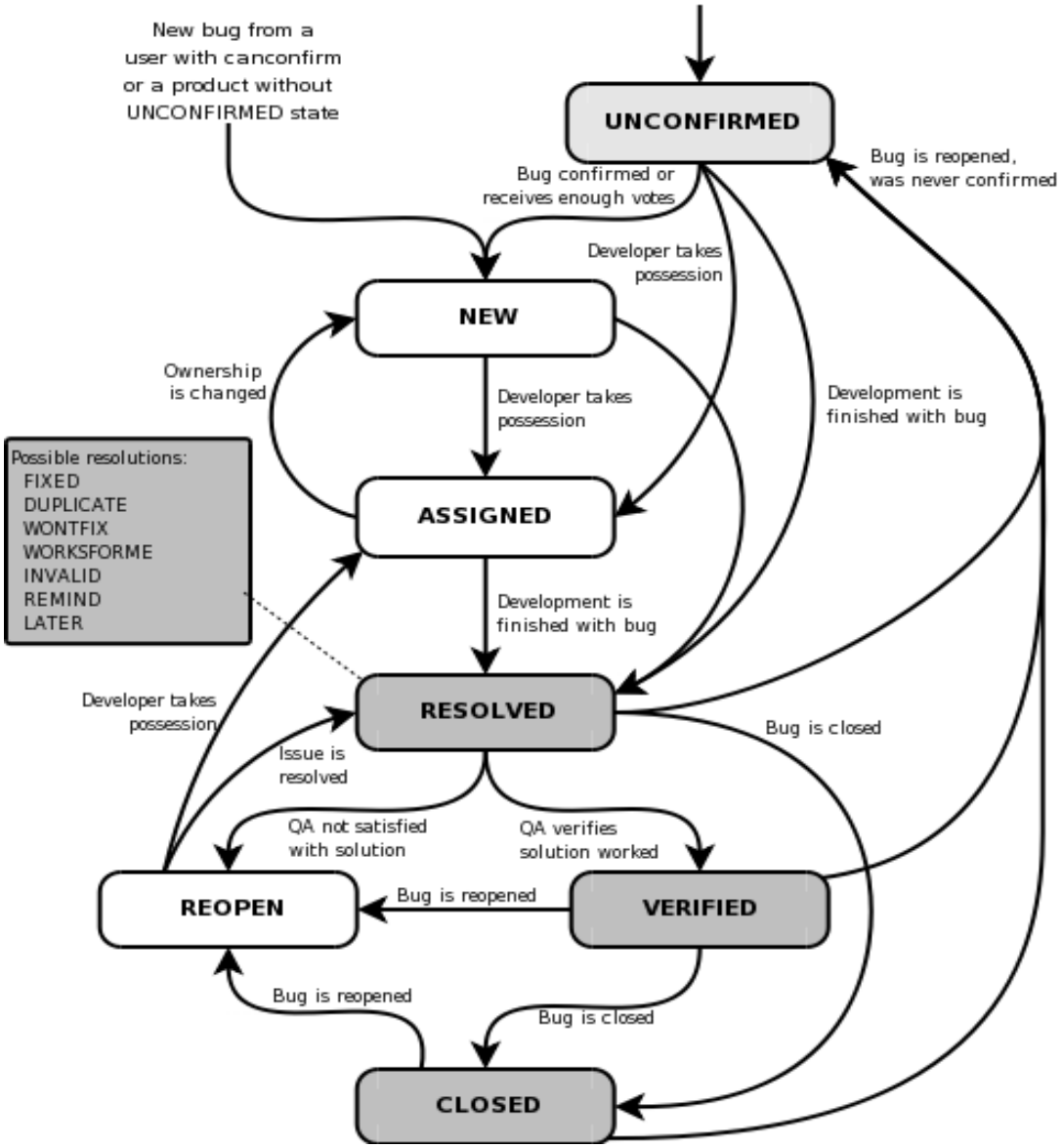
Bugzilla provides the following services,

- User management, a user can be a bug submitters, a developer, and a voter.
- Complete bug life cycle management, including creation, handling, tracking and closing of a bug.
- Support Orthogonal Defect Classification.
  - A user can record orthogonal factors such as trigger, severity, components, ...
- Improve the visibility of quality processes, for example, one can query bugs by,
  - Component,
  - Developers assigned to.

Bugzilla around world

- Free Software Projects
  - Mozilla: https://bugzilla.mozilla.org/
  - Linux Kernel: http://bugzilla.kernel.org/
  - Gnome: http://bugzilla.gnome.org/
  - KDE: http://bugs.kde.org/
  - Apache Project: http://issues.apache.org/bugzilla/
  - Open Office: http://www.openoffice.org/issues/query.cgi
  - Eclipse: http://bugs.eclipse.org/bugs/
- Linux Distributions
  - Red Hat: https://bugzilla.redhat.com/bugzilla/
  - Mandriva: http://qa.mandriva.com/
  - Gentoo: http://bugs.gentoo.org/
  - TurboLinux: https://bts.turbolinux.co.jp/bugtraq/
  - Novell: https://bugzilla.novell.com/
- Companies
  - NASA: http://itos.gsfc.nasa.gov/~bugzilla/
  - Facebook: http://bugs.developers.facebook.com/
  - Plus Akamai, Nokia, The New York Times, Yahoo! and many more..Mozilla firefox

Bug Life Cycle



- This diagram implies a generic quality process.
- You need to define your bug life cycle in your test strategy and test plan documents.

Create a Bugzilla Record

- First choose product,
- Necessary fields:
  - Component
  - Assign To: gets filled in automatically with default assignee for the component
  - Summary: pitchy, one-line description
  - Description: complete bug description. All technical details needed to reproduce the bug, including,
    - Trigger, reproduction steps, error outputs, etc.

Bugzilla and SVN

- It is important to record your changes using SVN revision number.
  - Other can know when and how a bug is fixed.
- Cross-reference bugs with svn log entries. A two-step process:
  - When submitting a change to SVN, you SVN log message comment shall include a line “bug# xxxx” to record which bug this change is related to.
  - In the bug: What changes were made to the source code to fix this bug?
    - Cut and paste your submission log, and paste it in the description field of a bug.

Bugzilla @ senior design

- A bugzilla server has been set up,
  - design.tricity.wsu.edu/bugzilla.
  - Each of you will receive your bugzilla acount.
- Use bugzilla to track bugs and tasks.
  - Your project manager will create a task for each milestone objective, and assign it to the proper team member.
- Integrate bugzilla into your team process.
  - You shall document the usage of bugzilla in test strategy and/or test plan.
  - What is discussed here is a generic process, your team needs to modify it to fit your own process.

Version Control with Subversion

Subversion System

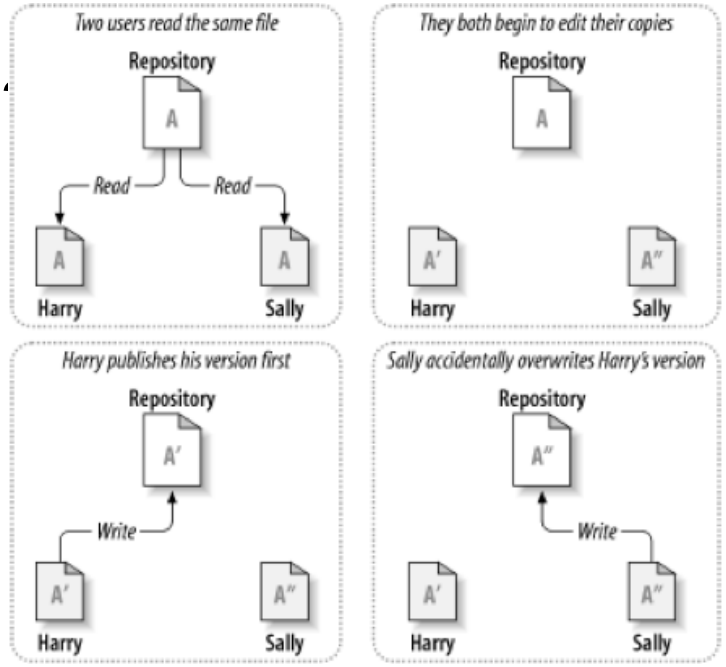
- For managing large projects with multiple people
  - widely used, open source. initiated in 2000 by CollabNet Inc.
    - In 2007, No.1 market share in Software Configuration Management (SCM) software and a strong performer in Software Configuration and Change Management (SCCM).
    - Use in many open-source projects such as Apache, Google code, etc.
  - works across network as client-server
- Fixes many of shortcomings of CVS, for example, it can preserve file/directory history when a user,
  - Delete a file
  - Rename/move a file.

Subversion System

- store and retrieve all versions of all directories and files in a project
  - usually source code
  - also documentation, tests, binaries, ...
- support multiple concurrent users
  - independent editing of files
  - merged into single version

SVN Basics

- Files are stored in a centralized *repository*
  - Contains a database of files and directories, and internal version information
  - Can be local or remote.
- Why version control system
  - Things we want to avoid :



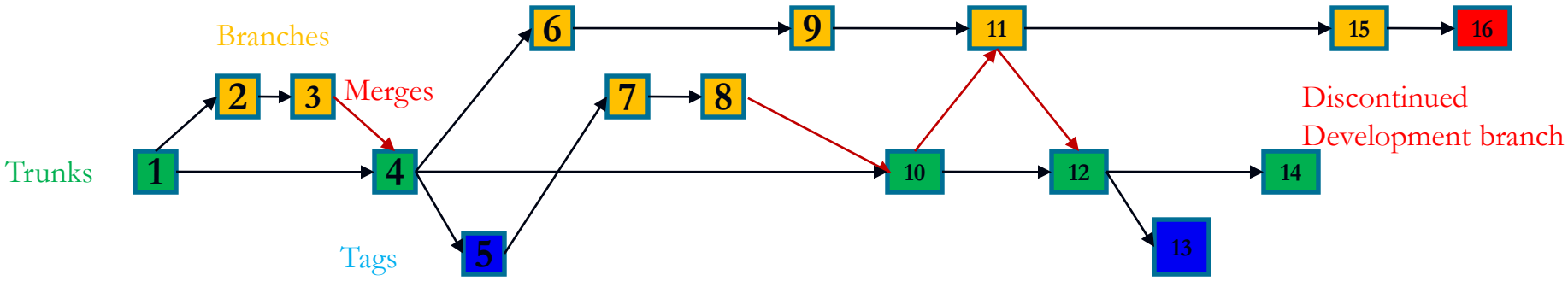
SVN Basics

- Developers *checkout a private working-copy of the* project
  - Modifications are made locally
  - When a developer is satisfied with the changes, a *commit propagates them to the repository*
- Revision numbers
  - Each commit of the project gets a *revision number*
    - Unlike CVS, a SVN revision number is an integer.
  - The initial revision number is 0

Tags, Branches, and Directories

SVN support tags and branches using directories.

- Unlike CVS, svn doesn't make difference among tags, branches, and directories.
- A typical file structure of SVN, usually you create 3 directories:
  - Trunk: contain head revision.
  - Tags: tag your files.
  - Branches: different branches of same file.
- A typical scenario of tags and branches, and merging difference revision.



- A new tag will be created by creating a new directory under "tags".
- A new branch will be created by creating a new directory under "branches".
- New branch/tag is created in repository using "copy" command.

Get Started

- SVN uses client-server architecture.
  - Client side commands start with “svn”
  - Server side commands start with “svnadmin”
- Create a new repository (serve-side command *create, executed on* the server, e.g. elec.tricity.wsu.edu):
  - svnadmin create /home/svn/cptsXXX/username, where cptsXXX is this class.

Get Started

- Setting up the files (client-side command *import, performed on* your working machine):
  - Build the following directory structure on your local machine:
    - ../homework1/branches/
    - ../homework1/tags/
    - ../homework1/trunk/
  - Import into repository:
    - cd to your local homework1 directory

    svn import ../homework1
    svn+ssh://username@elec.tricity.wsu.edu/home/svn/cptsXXX/username -m "initial import"
    - -m stands for "--message", the log information.
- SVN supports a set of access methods.
  - Possible access methods: file: svn: svn+ssh: http: https:

Client-Side Commands

- Getting the source (client-side command *checkout):*
  - svn checkout svn+ssh://username@elec.tricity.wsu.edu/home/svn/cptsXXX/username
- After you edit your files, commit changes (client-side command *commit):*
  - svn commit -m “Added comments” foo.c
- “svn update” updates the current work version of file.
  - Sample output of update
    - U Map.java - local copy was updated with changes from repository
    - A Map.java - file was added
    - Map.java - file was deleted
    - Map.java - file was replaced
    - Map.java - file was successfully merged

Client-Side Commands

- Add files to repository:
  - svn add main.java
- Important: you need to use commit to actually add the file to the repository
  - It is always a good practice to check out files in a clean directory to make sure that the file is committed.
- Remove files from the repository:
  - svn delete extrafoo.java
  - Commit to actually remove the file from the repository.

Client-Side Commands

- Copy files: svn preserves version history via copy
  - Copy is the way to tag revisisions and/or create new branches

svn copy
svn+ssh://username@elec.tricity.wsu.edu/home/svn/cptsXXX/username/homework1/trunk
svn+ssh://username@elec.tricity.wsu.edu/home/svn/cptsXXX/username/homework1/branches/mybranch

- Move files:
  - svn move file_to_move new_path
- Create directories:
  - svn mkdir directory_name

Client-Side Commands

- “svn diff”: check the different between two revisions.
  - “svn diff -r N” compares the current working version against the version N.
  - “svn diff -r N:M” compares the revision M against the version N. For example:
    - “svn diff -r HEAD” compares the current working version against HEAD revision.
    - “svn diff -r HEAD:COMMITTED” compares the committed revision against the HEAD revision.
- Reverting to version in repository: svn revert

Client-Side Commands

- “svn merge”: merge two revisions.
  - As an improvement to CVS, SVN can merge the difference between two different revisions to another revision.
  - For example, suppose you want to merge mybranch@revision 3 to trunk, you can issue the following command on your working trunk copy:
    - svn merge –r N:M svn+ssh://home/svn/user_name/myproj/mybranch

Client-Side Commands

- If a developer submitted a change before you do, and then both of you change the same line of code, then there is a conflict when you update your working copy.
  - In case of conflict, 3 additional files are created:
    - Foo.java.mine
    - Foo.java.rOLDREV
    - Foo.java.rNEWREV
  - Manually resolve the conflict, delete all other files, then commit
  - You can also check the potential conflictions without updating: svn status

Client-Side Commands

- Looking at log messages: svn log
  - Displays a list of all commit messages
  - Tracking changes
- svn annotate
  - Displays the file line by line, with the name of the person that last
  - modified that line

Use of SVN in class

- svn is installed on ELEC system.
  - Each of you has a personal repository, accessible only by yourself and me.
  - For your project, a repository will be created for your team.
    - An important note: please don’t delete/rename repository directly on ELEC system using native file system operations. Doing so will destroy the integrity of your repository.
  - We will make extensive use of SVN as part of tools we used to standardize our processes. It will be used in,
    - Homework submission;
    - Concurrent code development in course project;
    - Tracking website changes, etc.

Use of SVN in class

- You can access SVN from your home using svn+ssh protocol.
  - You may install SVN on your linux machine, or,
  - Use tortoisesvn on Windows.
  - Many modern IDEs, such as netbean for java, include the support for SVN so you can check out the code directly through these IDEs.