Lecture 11: Version Control

Engineering Design with Embedded Systems

Patrick Lam
University of Waterloo

January 28, 2013

Last Time

Assignment 2 review.

Intents: chaining Activities.

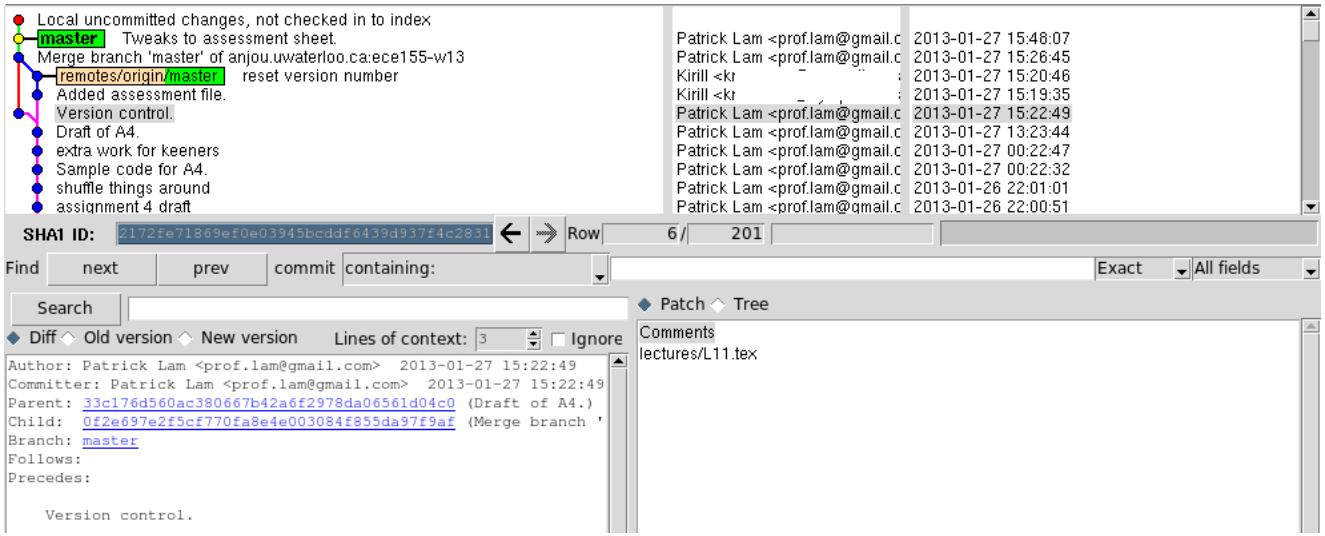
Saving and Restoring State.

Version Control

Ever wanted to undo your changes to software?

Ever needed to collaborate with others to develop software?

Conceptual Idea



- Store a **repository** of **revisions**.

  Each revision is a snapshot of a set of files.
- Can search by author, date, commit comment.
- Can revert to previous revisions.

Version Control Workflow

- **Copy**: to start, check out or clone a copy of the project, usually HEAD.
- **Modify**: do what you have to; test your changes; commit the result.
- **Merge**: others merge your changes into their working copies.

About Merging

Usually works smoothly.

Sometimes there's a **merge conflict**; must inspect:

- common ancestor;
- your change;
- conflicting change

and figure out what to do.

Obsolete Alternative

Lock-modify-unlock.

I don’t think anyone uses this anymore.

Distributed versus centralized

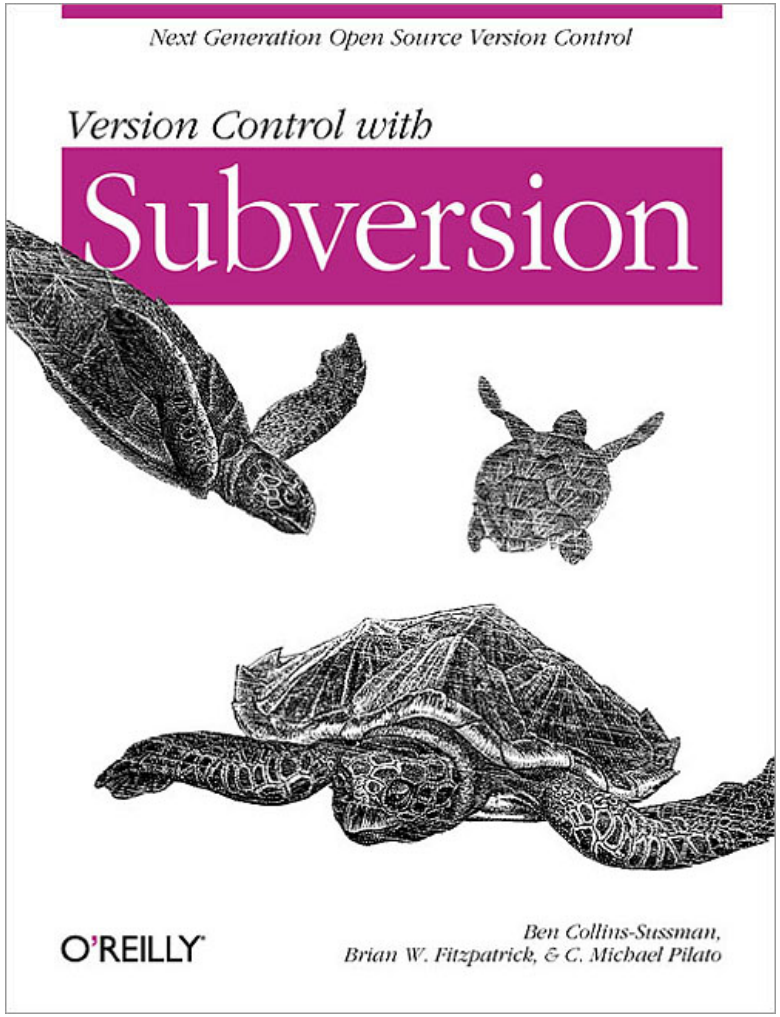
Traditionally:

- one central repository which all developers work against.
  Check out from master, commit back to master.

New-school:

- everyone's repository could serve as host, has full history.
  Can exchange code with others, no master needed.

Case Study: Subversion



(`http://svnbook.red-bean.com/`)

You are using Subclipse; I'll talk about command-line usage.

Creating a new repository

Create one from scratch:

```
svnadmin create c:\svn\repos
```

More commonly, check out a repository:

```
svn checkout http://k9mail.googlecode.com/svn/k9mail/trunk/
    k9mail-read-only
```

- creates a **working copy**. (You've done this.)

Adding and Ignoring Files

You've seen how to add files to the repository
(Team > Add to Version Control).
command-line: `svn add filename`

- failure to add files: leading cause of build breakage

You've seen ignored files like `R.java`.

- Generally, do not commit generated files!

Instead, tell Subversion to ignore them,
e.g. using wildcards.

Committing Files

On SVN, a commit makes your changes available to the world.
In decentralized version control, a commit records current version.

When to commit?

- What you commit must compile!
- Generally, one feature at a time, after testing. (Varies by source control system.)

Commit Messages

- An important form of project documentation[1].

Start with a one-line summary.

Establish the specific context of the change:

- Why is it necessary?
- How does it work?
- What are the effects?

Meta-commit message[2]:

```
Summarize clearly in one line what the commit is about

Describe the problem the commit solves or the use
case for a new feature. Justify why you chose
the particular solution.
```

[1] http://who-t.blogspot.ca/2009/12/on-commit-messages.html, accessed 27Jan13

[2] https://github.com/erlang/otp/wiki/Writing-good-commit-messages, accessed 27Jan13

Updating your repository

Pull changes from the repository to your working copy.

Use `svn update` to do that. If all goes well, you'll get output like this:

```
plam@noether:~/production/11.aosd.modularity$ svn up
D    example.tex
A    studies.tex
U    introduction.tex
A    sketch.tex
U    main.tex
```

Conflicts: the bane of your existence

This is a pain:

```
plam@noether:~/production/11.aosd.modularity$ svn up
C    example.tex
```

Why?

- Both the latest version and your version differ from the common ancestor.

Example conflict

How?

1. I wrote: “Here’s a line of text”.
2. Programmer X changes it to “Here’s a line of text that I modified.”
3. I change it again to “Here’s a modified line of text.”

The result:

```
<<<<<<< HEAD
Here’s a line of text that I modified.
=======
Here’s a modified line of text.
>>>>>>> zzz
```

You need to fix it and tell SVN that you’ve fixed it.

Stepping Back in Time

Major win of version control:

- can undo sketchy changes.

Can update to a past revision number or a date/time.

How to know which version to revert to?

- your detailed log messages!

Note: you can't commit an update, but you can merge it to your working copy.

Diffs

Basic unit of version control is the diff:

- describes what's **different** between two versions.

Inspect your diffs before committing.
Commit minimal diffs.

```
===================================================================
--- Text/abstract.tex (revision 17379)
+++ Text/abstract.tex (working copy)
@@ -1,10 +1,10 @@
 Runtime monitoring enables developers to (1) specify important program
 properties and (2) dynamically validate that these properties hold.
 In recent research, we have found that static analysis techniques can,
-in many cases, verify that runtime monitors never trigger.  In
-this paper, we describe a system which enables developers to visualize
-the remaining cases---potential
+in many cases, verify that runtime monitors never trigger.  In this
+paper, we describe a tool which enables developers to visualize the
+remaining cases---potential points
```

Basic SVN Workflow

Repeat until done:

- Update your working copy.
  (`svn update`)
- Edit files. Manipulate tracked files.
  (`svn add, svn rm, svn copy, svn move`)
- Examine changes.
  (`svn status, svn diff`)
- Undo changes, if necessary.
  (`svn revert`)
- Commit changes to the server.
  (`svn commit`)