Version Control

Steven J Zeil

February 19, 2013

Contents

1 Version Control

Version Control

Version control (a.k.a. *version management* is concerned with the management of change in the software artifacts being developed.

- Sometimes considered a sub-area of the problem of configuration management
  - a.k.a., Software Configuration Management (SCM)
    - Oddly enough, many tools labeled as SCM tools only address version control

. . . . . .

2 Issues in Version Control

Issues

The issues addressed by version control are:

- History
  - How has the software changed since *date-or-version-number*? Who made those changes? Why were they made? Can we go back?

- Exploration
  - Can we try out a set of plausible changes without affecting the “main” software build? Even if exploration of the effects of those changes may take a long time?
- Collaboration
  - Can we have multiple developers working on the code without interfering with one another’s work?

. . . . . .

3 Background - Changing the Code Base

ed

One of the earliest Unix text editors, **ed** applies a series of editing commands like ’a’ to append to th end of a file, ’i’ to insert a line at the current location, ’d’ to delete the current line, etc.

- Few people use **ed** now
  - though its line-oriented “child”, **sed** is still a popular scripting tool.

. . . . . .

diff

diff compares two files line by line, listing the differences between them.

- Differences are listed as a series of line replacements, insertions, and/or deletions
  - Reduces each line to a hash code
  - Uses the dynamic programming algorithm for computing the Levenshtein distance between the two sequences of hash codes to obtain an approximately shortest sequence of commands
- Can emit differences as **ed** commands:

```
diff --ed file1 file2 > file12.diff
echo w file2 >> file12.diff
ed file1 < file12.diff
```

would "rebuild" `file2` from `file2` and the diff.

.....

patch

patch takes a slightly more sophisticated approach to the idea of applying a **diff** output to a file

```
diff file1 file2 > file12.diff
  ⋮
patch file1 file12.diff
```


- Allows a variety of different **diff** variants
- Can detect if `file1` has already been changed so that the line numbers and other info in the patch file `file12.diff` are no longer accurate.
  - Attempts to compensate

. . . . . .

Integrating Changes

Suppose that we have two patch files created from the same base file `file1`

```
patch -o file2a file1 patchA
patch -o file2b file1 patchB
```

Change integration is the problem of combining both sets of changes to form a desired file `file2`.

. . . . . .

Two-way Change Integration

```
patch -o file2a file1 patchA
patch -o file2b file1 patchB
```

1. Compare `file2a` and `file2b`

2. Wherever the two are different, prompt the human to select the desired change.

.....

Three-way Change Integration

Takes the base file into account as well as the two changed files.

```
patch -o file2a file1 patchA
patch -o file2b file1 patchB
```

1. Compare `file1` and `file2a`, then `file1` and `file2b`
2. Any lines that differ from `file1` in only one of the two other files can be applied automatically.
3. Wherever both `file2a` and `file2b` are different from `file1`, prompt the human to select the desired change.
   - This is called a *conflict*.

.....

4 Approaches and Tools

Version Control Systems

If we could extend `patch` multiple files at once, we could, in theory, patch an entire software system to move it from version 1 to version 2, then patch it again to move to version 3, etc.

- This would be the heart of a version control system,

. . . . . .

Approaches and Tools

- *Local* version control systems manage history by setting aside directories on the same file system where the software under control is housed.
  - **sccs**, **rcs**
- *Centralized* version control systems keep the system history at a centralized location accessible via the network.

  Developers check out a copy of the current (or a desired older) version of the software onto their own machines.
  - CVS, Subversion
- *Distributed* version control systems allow developers to keep the full system history on their own machines.

  A central location may hold a base copy for management/distribution purposes, but this is not required.
  - **git**

. . . . . .