[Handout for L2P3]
Your Own Private Time Machine: Introduction to Revision Control

Revision control (individual)

Revision control system (RCS) is an indispensable tool in the software engineer's tool box.

The following introduction to revision control was adapted (with minor modifications) from the excellent online book Mercurial: The Definitive Guide by Bryan O'Sullivan

Why revision control?

Revision control is the process of managing multiple versions of a piece of information. In its simplest form, this is something that many people do by hand: every time you modify a file, save it under a new name that contains a number, each one higher than the number of the preceding version.

Manually managing multiple versions of even a single file is an error-prone task, though, so software tools to help automate this process have long been available. The earliest automated revision control tools were intended to help a single user to manage revisions of a single file. Over the past few decades, the scope of revision control tools has expanded greatly; they now manage multiple files, and help multiple people to work together. The best modern revision control tools have no problem coping with thousands of people working together on projects that consist of hundreds of thousands of files.

Why use revision control?

There are a number of reasons why you or your team might want to use an automated revision control tool for a project. It will track the history and evolution of your project, so you don't have to. For every change, you'll have a log of who made it; why they made it; when they made it; and what the change was.

When you're working with other people, revision control software makes it easier for you to collaborate. For example, when people more or less simultaneously make potentially incompatible changes, the software will help you to identify and resolve those conflicts.

It can help you to recover from mistakes. If you make a change that later turns out to be in error, you can revert to an earlier version of one or more files. In fact, a really good revision control tool will even help you to efficiently figure out exactly when a problem was introduced.

It will help you to work simultaneously on, and manage the drift between, multiple versions of your project. Most of these reasons are equally valid—at least in theory—whether you're working on a project by yourself, or with a hundred other people.

The many names of revision control

Revision control is a diverse field, so much so that it is referred to by many names and acronyms. Here are a few of the more common variations you'll encounter:

- Revision control system (RCS)
- Software configuration management (SCM), or configuration management
- Source code management
- Source code control, or source control
- Version control system (VCS)

Some people claim that these terms actually have different meanings, but in practice they overlap so much that there's no agreed or even useful way to tease them apart.

It follows from the explanation above that using an RCS frees us from having to keep copies of all the past versions on our hard disk. It reduces problems such as 'updating the wrong version', 'overwriting latest changes with an older copy', 'multiple developers overwriting each others' changes' etc. If the RCS spans multiple machines, it reduces the risk of 'losing everything' in case of a hard disk crash.

Two approaches to RCS

There are two types of RCSs in use:

- Centralized: the RCS runs on a server; developers connect to the server to access the RCS. E.g., Subversion, CVS, ClearCase , Microsoft Team Foundation Server.
- Distributed: The RCS is spread over developers' machines. E.g., Mercurial , Git , Bazzar

Basic concepts of RCS (individual use)

- **Repository (Repo)**: The database where the files and historical data are stored, including the author of the changes and the summary of each change. Commonly called repo for short.
- **Working Copy**: The local directory of your files.
- **Check in (Commit)**: Uploads a changed file or a set of changed files to the repository. Commonly known as commit.
- **Revision**: A revision is the set of changes whenever a check in is performed. Each revision is given a number.
- **Check out**: Downloads a file or a set of files from the repository (for the first time).
- **Add**: Tells the RCS to track a file, a set of files or a directory. These tracked files do not go into the repository until the next check in.
- **Update:** Synchronize the files in your working copy with the latest files from the repository. This is normally performed after a check out.
- **Revert**: Discards all changes in the working copy and use a specified revision from the repository.
- **Tag**: Label a revision for easy reference.

---End of Document---