**Using Git to save history**

**Revision control**

* Process of managing multiple version of the file.
* 3 pros:
  + Revision control makes it easier to collaborate.
  + Recover from mistakes. – Revert back to earlier version
  + Work simultaneously.

Exercise.

**What does RCS stands for?**

Revision Control Software.

**What is a revision?**

Version of multiple pieces of information. When you take a file, mod it and save, you get a newer version of the file.

**Benefits of RCS**

Help a single user manage revisions of a single file

Help a developer recover from a incorrect modification to a code file

Makes it easier for a group of developers to collaborate on a project

  Manage the drift between multiple versions of your project

  Detect when multiple developers make incompatible changes to the same file

**Explain RCS in a team project context.**

Suppose You are doing a team project with Tom, Dick, and Harry but those three have not even heard the term RCS. How do you explain RCS to them as briefly as possible, using the project as an example?

RCS manages multiple version of a single piece of information, to assist in simultaneous working together.

**Repositories**

The database of the history of a directory being tracked by an RCS software (e.g. Git).

* Git helps to keep track of all the different versions of the file. (.git/)
  + Can revert the versions without creating many different copies of the same file.

**Saving History**

In a repo, we can specify which file to track / ignore. – Temporary files created during build/ testing should not be revision controlled.

**Staging and Commiting**

Committing **saves a snapshot of the current state of the tracked files in the revision control history**

**When ready to commit, we first** stage **the specific changes we want to commit**

**Each commit is a recorded point in the history of the project, uniquely identified by auto-generated hash.**

**You can tag a commit with a understandable name. eg. V.1.0.2**

**Commit**: Saving the current state of the working folder into the Git revision history **Stage**: Instructing Git to prepare a file for committing.

**IDE**

Integrated Design Environment

* Contains
  + source code editor
  + compiler and/or an interpreter
  + Debugger

Exercise

Features that IDE have

a. Compiling.

b. Syntax error highlighting.

c. Debugging.

d. Code navigation e.g., to navigate from a method call to the method implementation.

e. Simulation e.g., run a mobile app in a simulator.

f. Code analysis e.g. to find unreachable code.

g. Reverse engineering design/documentation e.g. generate diagrams from code

h. Visual programming e.g. Write programs using ‘drag and drop’ actions instead of typing code.

i. Syntax assistance e.g., show hints as you type.

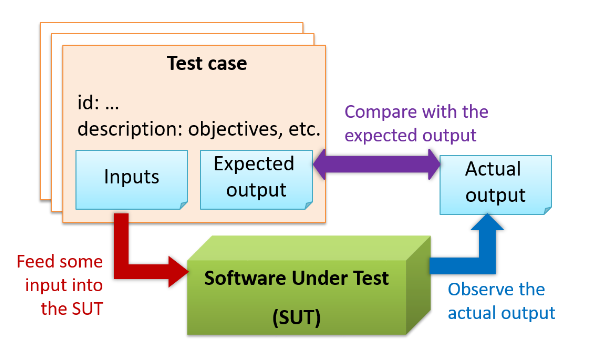
j. Code generation e.g., to generate the code required by simply specifying which component/structure you want to implement.

k. Extension. i.e. ability add more functionality to the IDE using plugins.

**Navigation**

Use Ctrl B to navigate to the method.

**Testing**



**Testing**: Testing is operating a system or component under specified conditions, observing or recording the results, and making an evaluation of some aspect of the system or component.

Test case: A test case specifies how to perform a test. At a minimum, it specifies the input to the software under test (SUT) and the expected behavior.

**A test case failure is a mismatch between the expected behavior and the actual behavior**

**A failure is caused by a defect (or a bug).**

**Regression Testing – Usually automated**

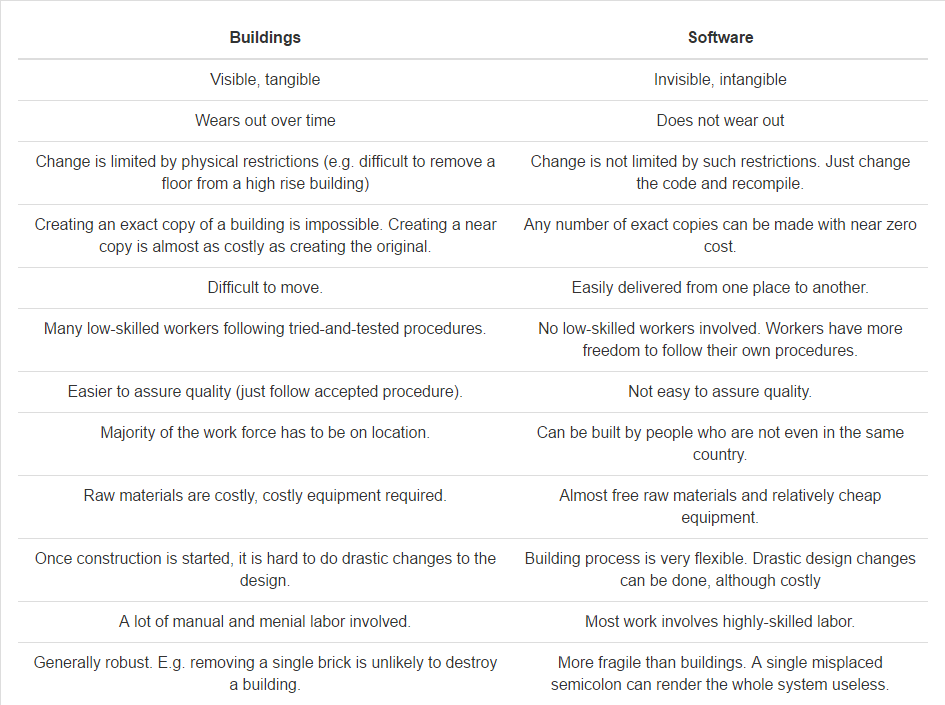
**When we modify a system, the modification may result in some unintended and undesirable effects on the system. Such an effect is called a regression.**

* **re-testing the SUT to detect regressions.** Noted that to detect regressions, we need to retest all related components, even if they were tested before.

**Test Automation**

**An automated test case can be run programmatically and the result of the test case (pass or fail) is determined programmatically.**

**Software Engineering vs CE**



**Building software vs building bridges**

Depends on the size of the software. Manpower required for software is very costly. On the other hand, we can create a very valuable software (e.g. an iPhone application that can make million dollars in a month) with a just a PC and a few days of work!