Công Cụ & Phương Pháp Thiết Kế - Quản Lý (Phần Mềm)

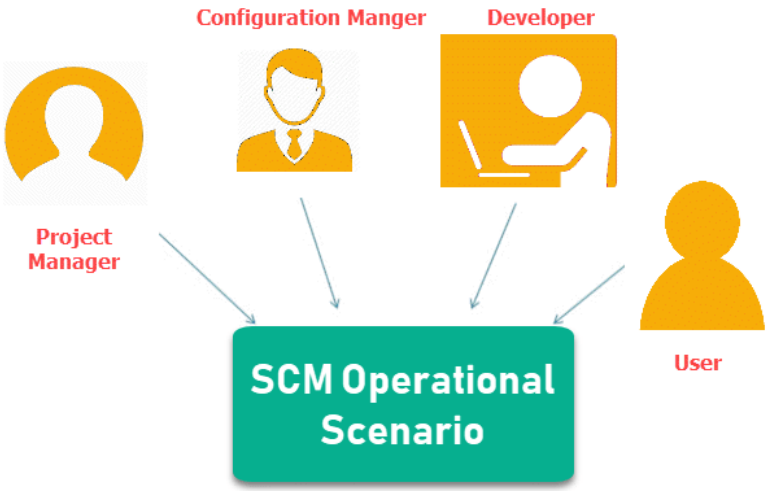
TRAN KIM SANH
Instructor of DTU

Email: trankimsanh@dtu.edu.vn
Tel: 0987 409 464

Defining a Configuration Management Process

Contents

- What is Configuration Management?
- Why use Configuration Management?
- Where does Configuration Management fit?
- How does Configuration Management work?
- Terminology



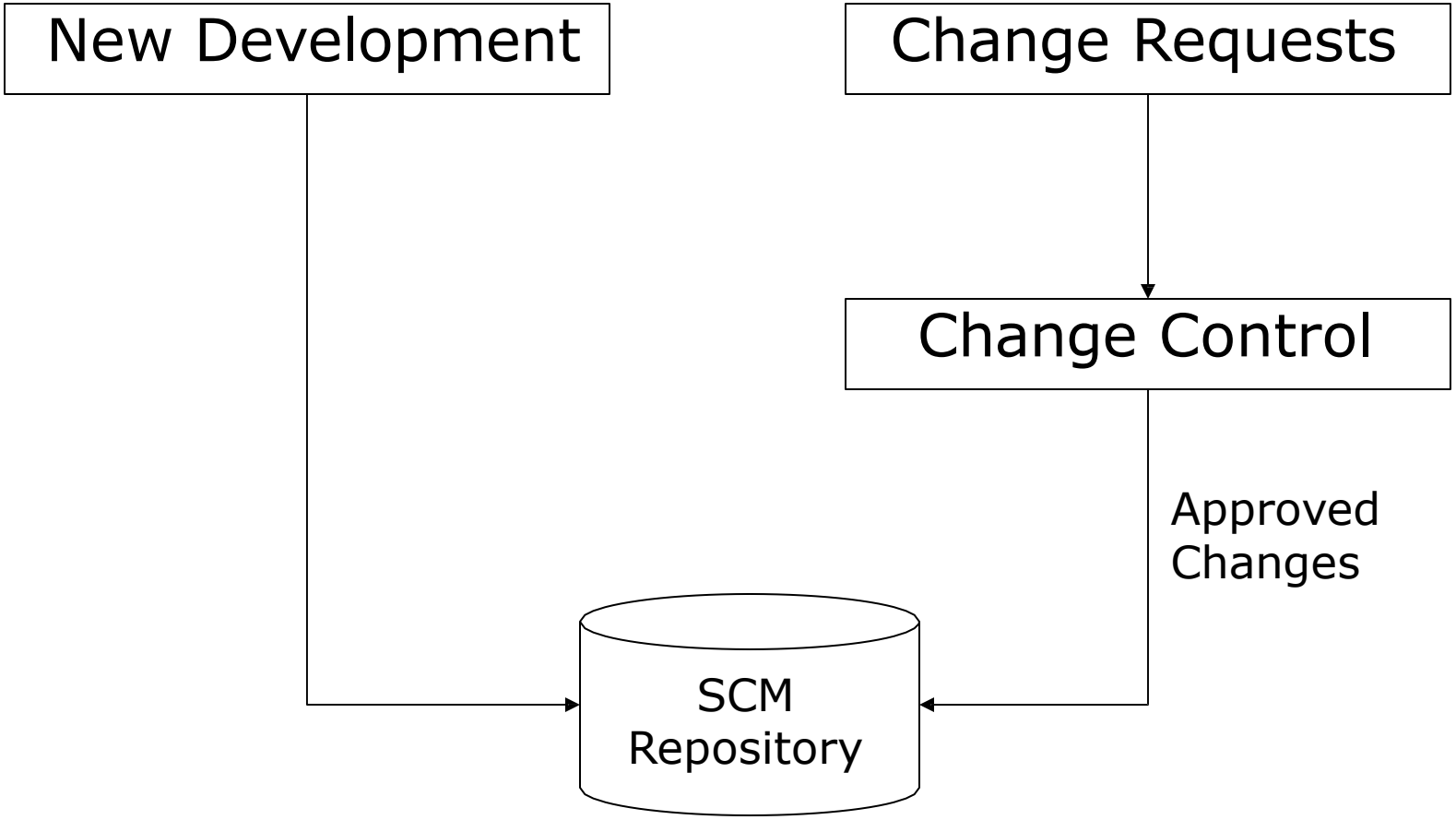
What is Configuration Management?

- Configuration Management (CM) is the process of controlling and documenting change to a continually evolving system
- It is part of change management
- Software Configuration Management (SCM) is Configuration Management specifically designed to meet the needs of software development and maintenance projects

Why Use SCM?

- Consider this scenario:
  - Software Engineer1 opens foo.cpp
  - Software Engineer2 opens foo.cpp
  - Software Engineer1 makes a change to foo.cpp
  - Software Engineer2 makes a change to foo.cpp
  - Software Engineer1 saves foo.cpp
  - Software Engineer2 saves foo.cpp, wiping out Software Engineer1's changes
- Or this one:
  - A serious defect is reported for a previous release of our product. How do we fix that defect only and generate a new release?

Where Does SCM Fit?



Why Use SCM?

- So we will always know:
  - Where all revisions of all files necessary to create any product version can be found
  - Which source files revisions were used in each build
  - What changed in each revision of our source files, and why
  - How to recreate previous product releases
  - How to work as a team to make changes (defects and/or new features) in future builds and releases

Why Use SCM?

- So it is easy for developers to make changes, test their changes and integrate the results into the product
- So developers don’t waste a lot of time tracking down bugs in other people’s code that prevent them from testing
- So the QA team has a stable product to work with
- So the project team can easily release updated versions of their product in a repeatable manner
- If done right, significantly reduces Development & Test time

What Goes Under SCM Control?

- Everything!
  - Product source code
  - Scripts to build source code
  - Tools used to build product
  - 3rd party products used to create product
  - Data files

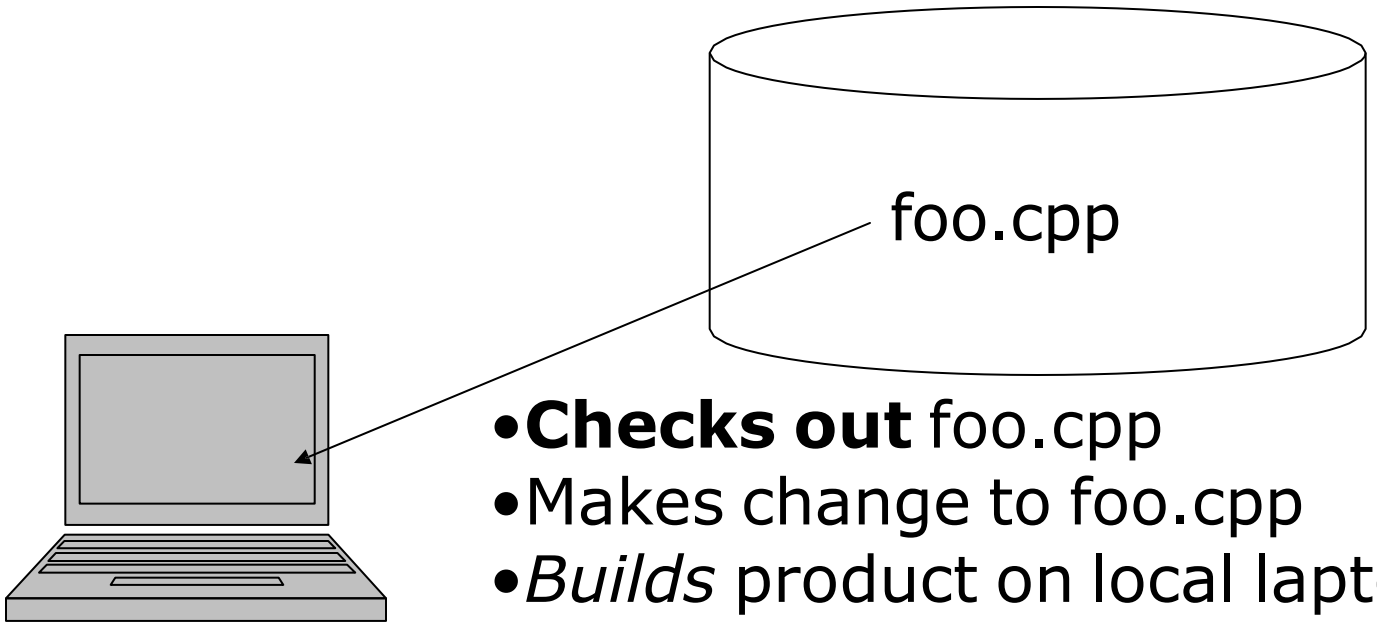
What Goes Under SCM Control?

- Everything!
  - Scripts to build or populate database
  - Specifications
  - User documentation
  - Test scripts
  - Plans
  - Etc.

SPECIFICATIONS

How Does SCM Work?

- Generic Model

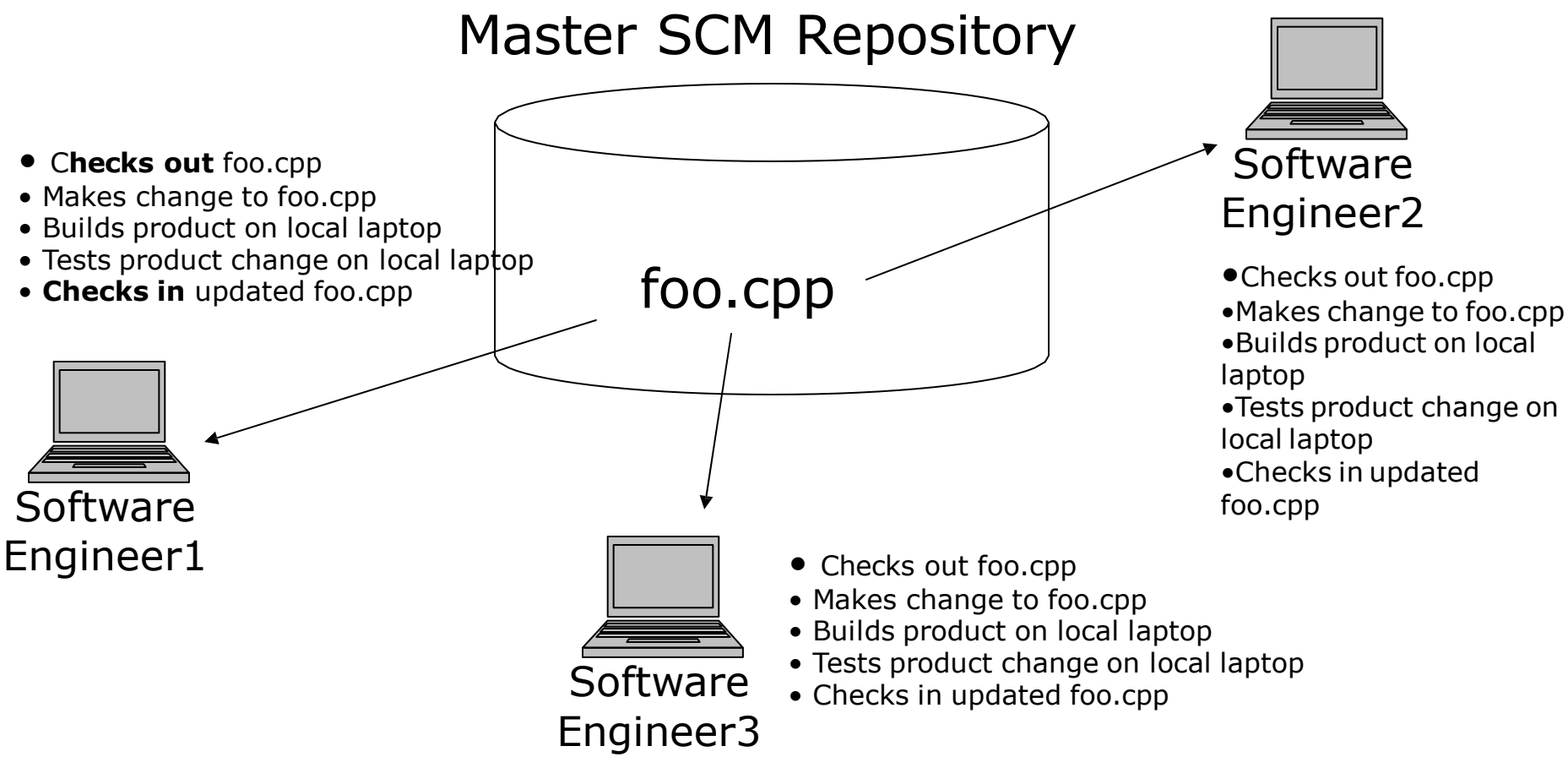


Software Engineer1

- **Checks out** foo.cpp
- Makes change to foo.cpp
- *Builds* product on local laptop
- *Tests* product change on local laptop
- *Rebuilds & retests* product using latest source code
- **Checks in** updated foo.cpp

How Does SCM Work?

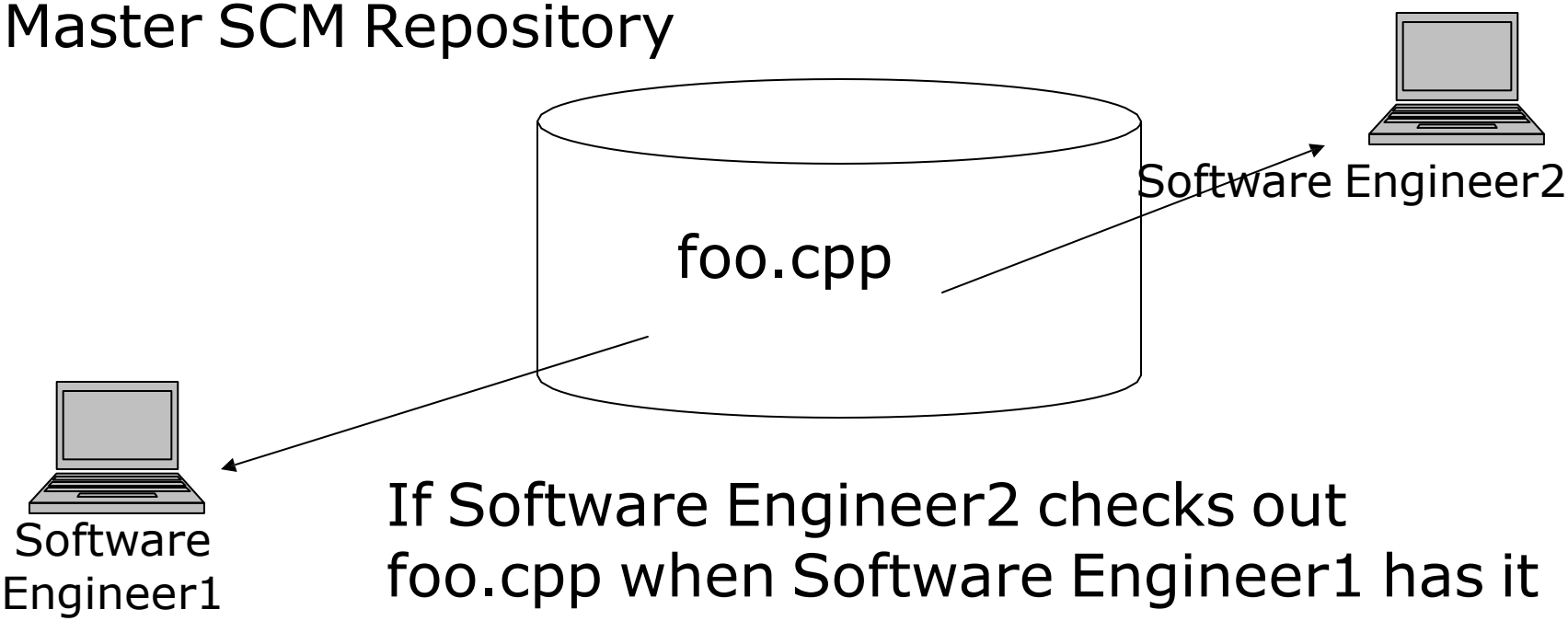
- Generic Model



How Does SCM Work?

- Generic Model

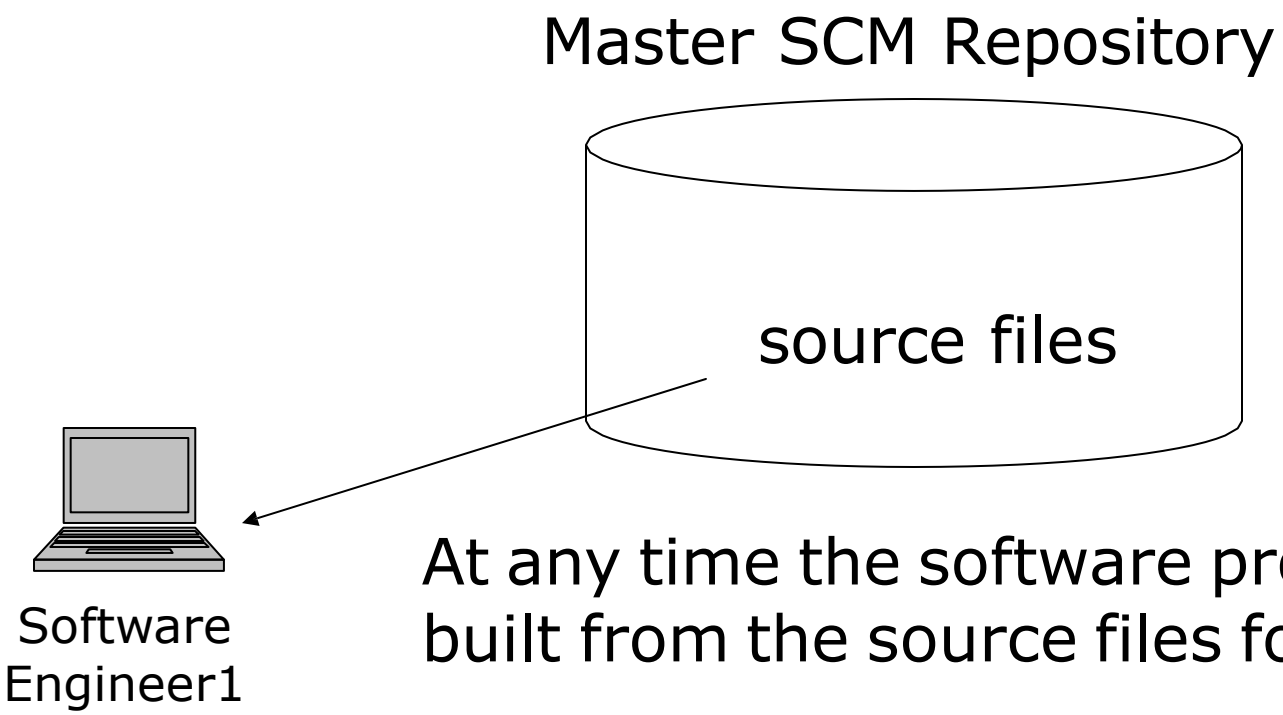
Master SCM Repository



If Software Engineer2 checks out foo.cpp when Software Engineer1 has it checked out, then the SCM system will automatically **merge** the files when they are checked in

How Does SCM Work?

- Generic Model



At any time the software product can be built from the source files for testing

Terminology

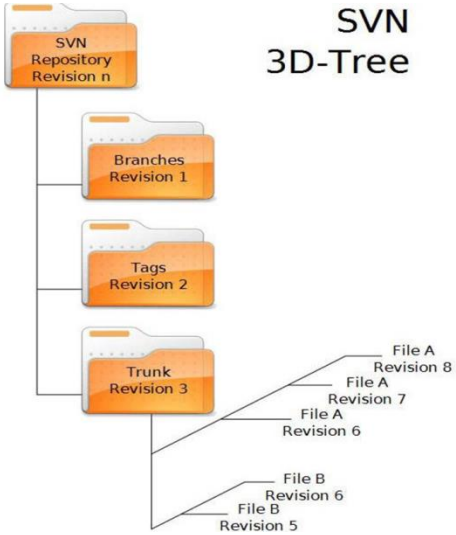
- Source File
  - Any file required to build the product on a clean machine. Includes build scripts, DDLs, install scripts, tests, etc.
- Product Build
  - Classic definition: Source files compiled and linked into an executable file
  - Modern definition: Putting everything associated with a product together in a usable format

Terminology (cont)

- Product Release
  - Providing software files to others for use. Typically this is a build that passed tests and is considered 'good'
- Revision
  - A instance of a source file that contains specific changes
- Version
  - A unique identifier given to a Release, typically associated with all source file revisions that a build is comprised of

Revision Control

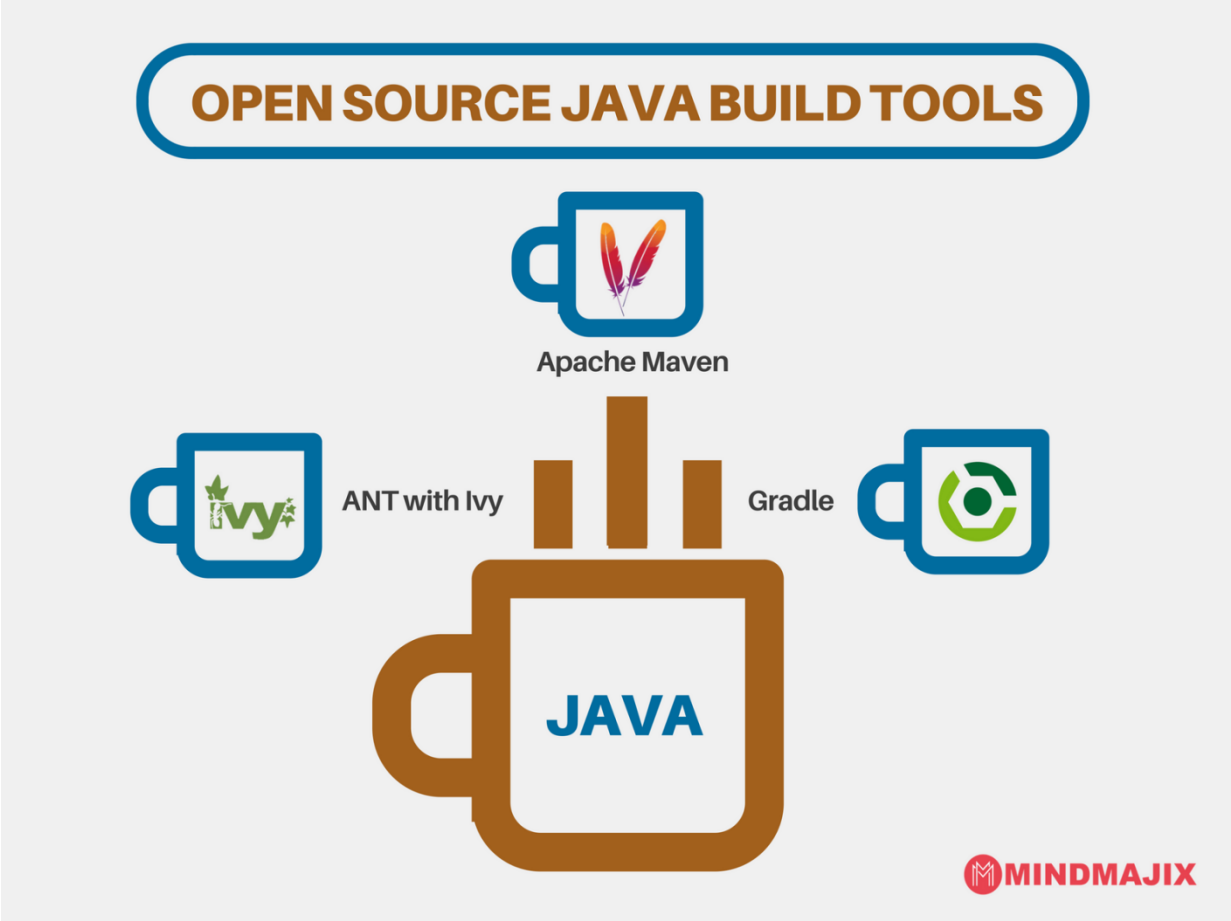
- Changes to source files must be captured using a revision control system so the team always knows what changed, and can revert to previous source file revisions if necessary
- Example:
  - Source File foo.cpp
    - ✓ Revision 1.0 (First time checked
    - ✓ Revision 1.1 (Bug fixed)
    - ✓ Revision 1.2 (Another bug fixed)
    - ✓ Revision 1.3 (Yet another bug fix



Builds

- When your product is created from all of the source code in your repository the product is called a “build”
- Builds are usually given to a QA team for system testing
  - If QA finds defects that must be fixed, then those defects are fixed and a new build created for system testing
  - If the build passes QA, then that build becomes a release and made available to users/customers

Build tools


OPEN SOURCE JAVA BUILD TOOLS
Apache Maven
ANT with Ivy
Gradle
JAVA
MINDMAJIX

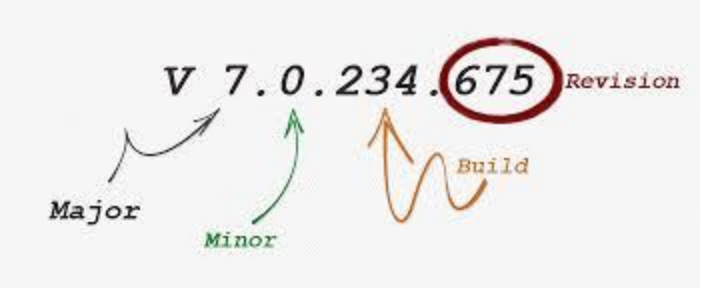
Builds and Automation

- Automate the creation of your builds
  - Use tools like Make or Ant, along with scripts
  - Embed the build number in your product
  - Ensure your builds are reproducible. That is, if you provide a build number, the system knows which source file revisions to include in the build
- Automate the packaging of your product, For example:
  - Creating releases for different platforms
  - Populating a master CD

Unique Release Numbering

- Each Build/Release must have a unique ID
- Required for developers to determine which build (& source files) a problem is occurring in
- Should be meaningful. I.e., XX.YY.ZZZZ, where:
  - XX is the Major Release Number
  - YY is the Minor (Maintenance) Release Number
  - ZZZZ is the Build Number
- Example: 1.01.0012
  - Would be the 12th build of release 1.01

Version Number



Database Version Numbering

- If your product has a database you'll need to establish a mechanism that ties database versions to product versions

Product Version	Runs with Database Version
1.01.0312	1.0, 1.1, 1.2
1.03.0229	1.3, 1.4
2.00.0332	1.5

- When your product starts up it checks to make sure that it is running against a supported database version

Notification and Logs

- Build process logs every activity of every step while it is executing to a log file
  - Anyone must be able to determine the state of any build by inspecting the build's log file
- Email notifications
  - Automatically sent to all developers that had new code checked in with that build
  - Summarizes the status of the build (good/bad)

Continuous Integration

- Build everything from scratch regularly
  - Constantly (if anything has changed)
  - Based on timer (developers have to get code in by a certain time)
  - At least nightly
- If successful, execute test suite
  - Automated tests are kept in SCM system too
- If all steps complete ok you have a successful build
  - Label the source files with the build number
  - Update logs and send notifications

Group discussion

- Discuss in groups of 4 students, comparing advantage/disadvantage of the following tools(10 minutes):

SoapUI POSTMAN Katalon Assertible

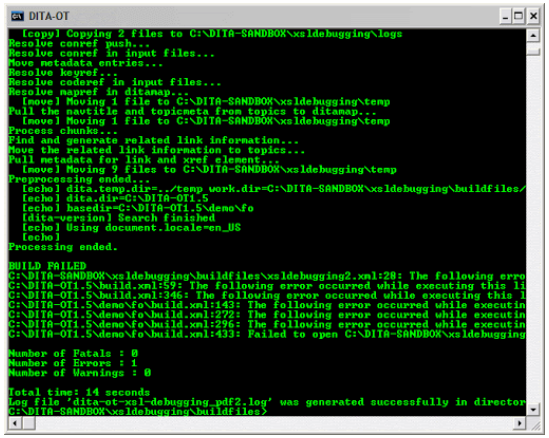
apigee jMeter REST-assured TRICENTIS Karate

Test Automation

- Ability for anyone to be able to run a suite of unit and/or system tests against the product at any time using simple commands
- Automated system tests are not intended to be exhaustive or replace system testing. The intent is to have a fast, repeatable process for ensuring that:
- Time spent adding new unit and/or system tests results in significantly shorter test cycles when trying to release the product

Build & Release Data Collection

- Build Log Data
- Build & Release Process
  - # of builds
  - Changes (by CR#) in each build



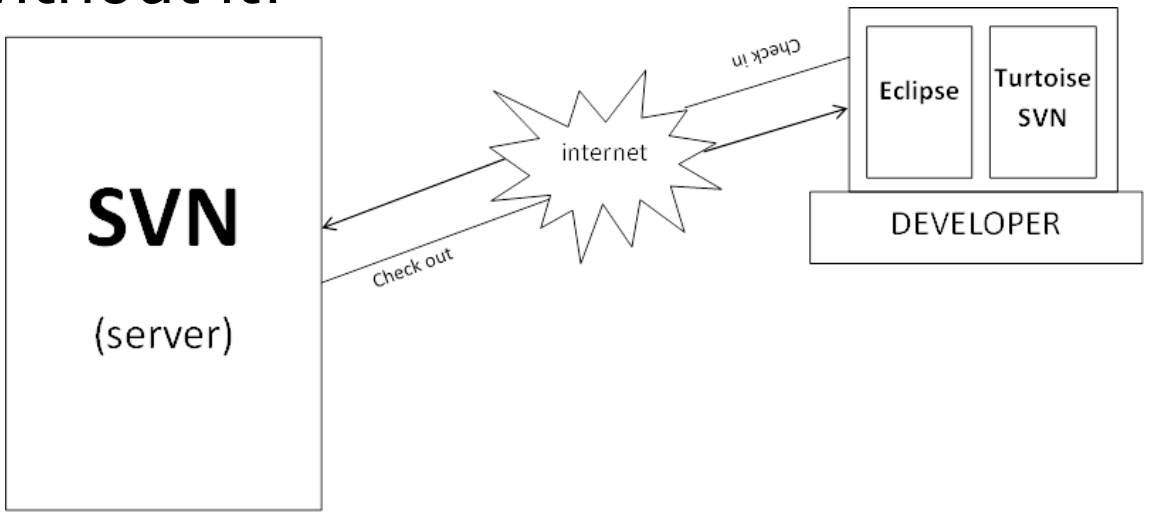
- System Test Process
  - # of product release cycles required to approve product is ready to ship
  - Elapsed time of each product release cycle
  - # of defects found during System Test per release cycle
  - Effort required to test each product release

Rules for Making Code Changes

- Developers are responsible for:
  - Keeping their machine in synch with the configuration management system
  - Creating/updating automated unit tests
  - Creating/updating automated unit and integration tests
  - Ensuring that all tests run successfully before checking code back into master repository
  - Resolving conflicts when checking code back in
  - Updating change documentation

Summary

- Software Development Teams MUST HAVE a Software Configuration Management system
- Implement Continuous Integration
  - Once you have you'll wonder how you ever got along without it!

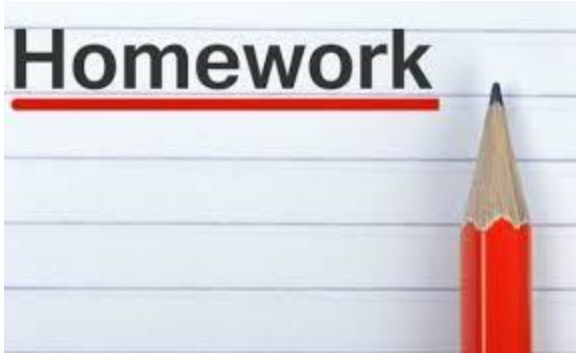


Video link

- https://www.youtube.com/watch?v=ce73O3pkgbU

Homework Assigment

- Software configuration management is a project function to increase the efficiency of technical and managerial activities by developing a configuration team for the whole organization or for each individual project.



Homework Assigment

- Required:
  - Install SVN Server
  - Grant Authorization
  - Create A project in Eclipse or Netbean
  - Checkin your project to SVN server
  - Using Tortoise SVN for checkin and checkout a document file to SVN

References

- Ian Sommerville. *Software engineering update 10th edition.* Wesley Computer Publishing 2018 Page: 730 - 749
- https://en.wikipedia.org/wiki/Software_configuration_management
- https://en.wikipedia.org/wiki/Apache_Subversion

© 2019, Tran Kim Sanh