

CPS847: Software Tools for Startups

Andriy Miranskyy

February 6, 2018

Outline

- Continuous Integration (CI)
- Practicum
 - Travis CI
 - Get your free access via <https://education.github.com/pack>
 - Code coverage

Configuration Management

Taken from Ch. 25
of Sommerville's Software Engineering

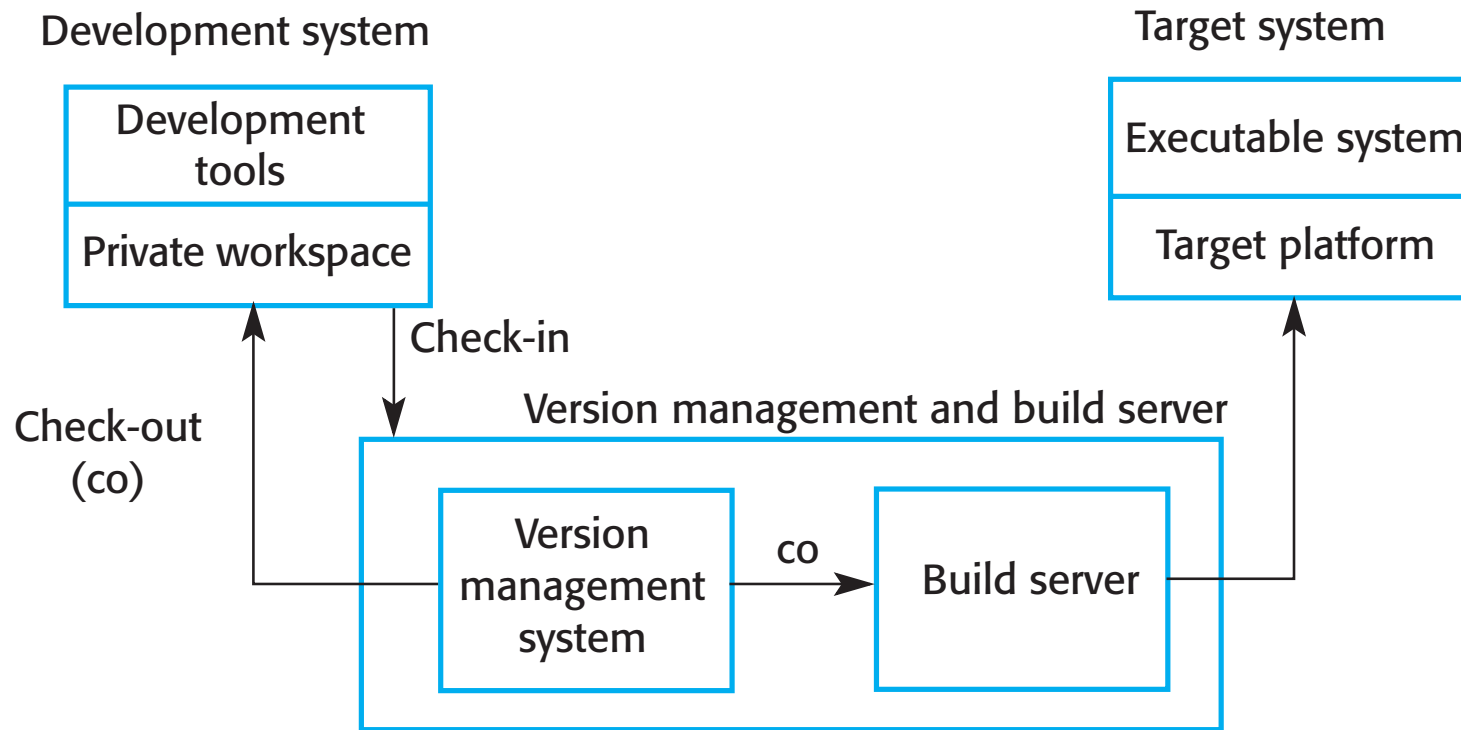
System building

- System building is the process of creating a complete, executable system by compiling and linking the system components, external libraries, configuration files, etc.
- System building tools and version management tools must communicate as the build process involves checking out component versions from the repository managed by the version management system.
- The configuration description used to identify a baseline is also used by the system building tool.

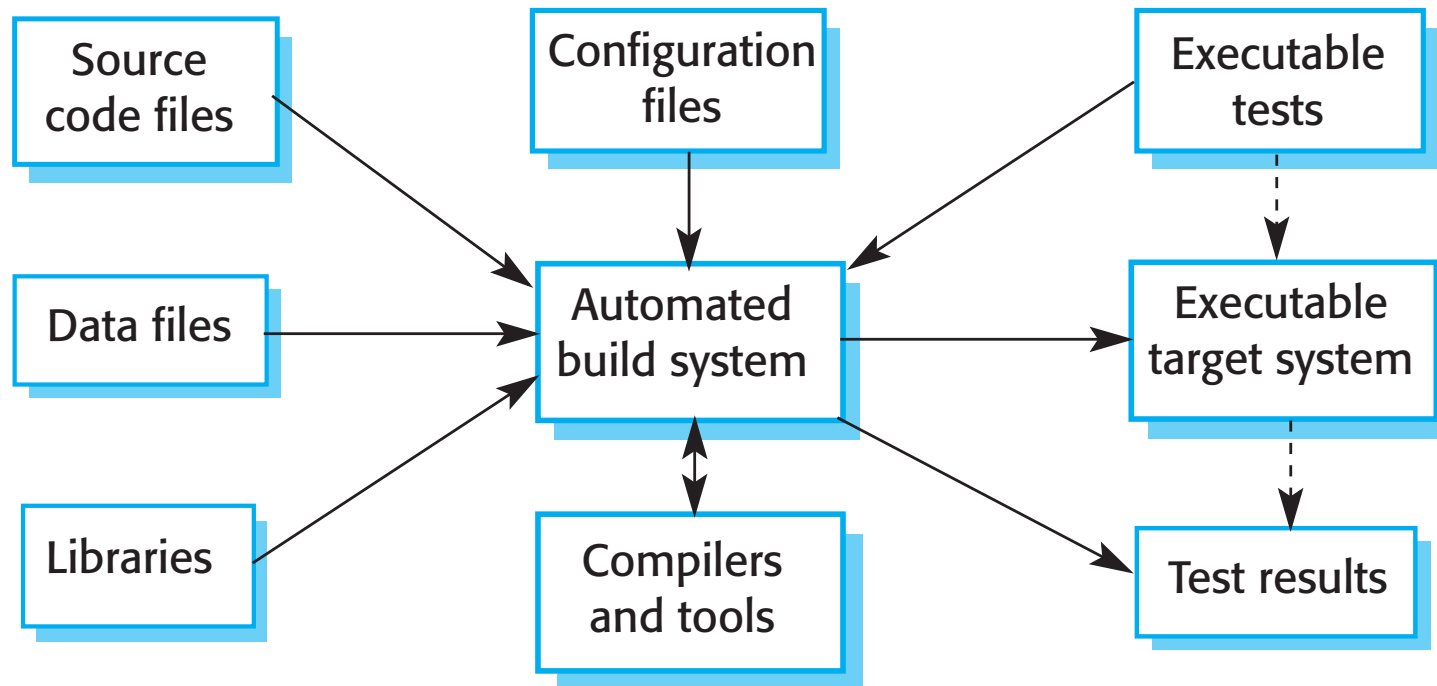
Build platforms

- The development system, which includes development tools such as compilers, source code editors, etc.
 - Developers check out code from the version management system into a private workspace before making changes to the system.
- The build server, which is used to build definitive, executable versions of the system.
 - Developers check-in code to the version management system before it is built. The system build may rely on external libraries that are not included in the version management system.
- The target environment, which is the platform on which the system executes.

Development, build, and target platforms



System building



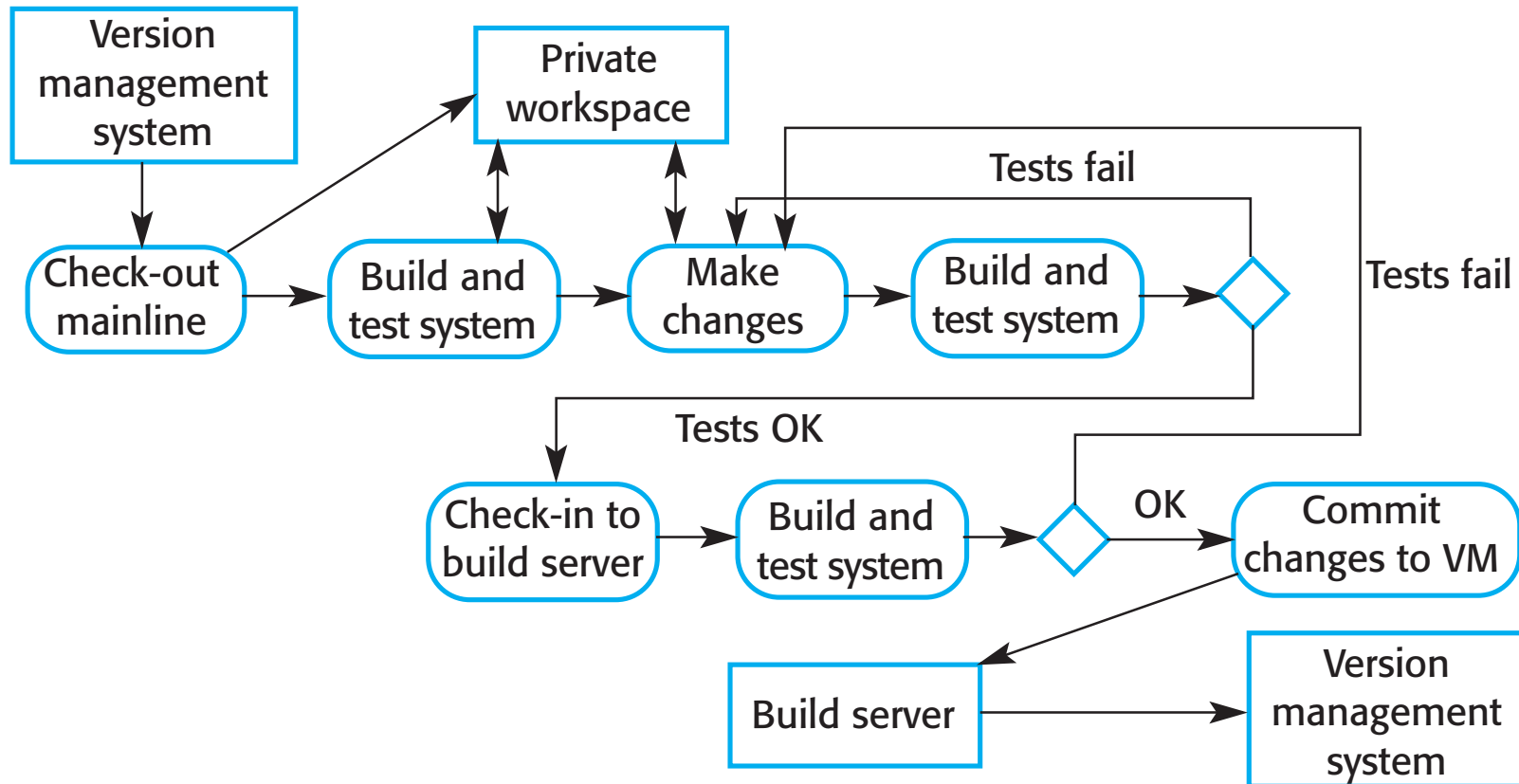
Build system functionality

- Build script generation
- Version management system integration
- Minimal re-compilation
- Executable system creation
- Test automation
- Reporting
- Documentation generation

Agile building

- Once the system has passed its tests, check it into the build system but do not commit it as a new system baseline.
- Build the system on the build server and run the tests. You need to do this in case others have modified components since you checked out the system. If this is the case, check out the components that have failed and edit these so that tests pass on your private workspace.
- If the system passes its tests on the build system, then commit the changes you have made as a new baseline in the system mainline.

Continuous integration



This scheme can be made more complex if needed

Daily building

- The development organization sets a delivery time (say 2 p.m.) for system components.
 - If developers have new versions of the components that they are writing, they must deliver them by that time.
 - A new version of the system is built from these components by compiling and linking them to form a complete system.
 - This system is then delivered to the testing team, which carries out a set of predefined system tests
 - Faults that are discovered during system testing are documented and returned to the system developers. They repair these faults in a subsequent version of the component.
- *These days this scheme is often pushed to the limit by rebuilding after every commit (if resources permit)*

CI best practices

- Maintain a code repository
- Automate the build
- Make the build self-testing
- Everyone commits to the baseline every day
- Every commit (to baseline) should be built
- Keep the build fast
 - I wish 😊 But for new /startup products it's typically easy.
- Test in a clone of the production environment
- Make it easy to get the latest deliverables
- Everyone can see the results of the latest build
- Automate deployment
 - We will look at this one later

Bullet points taken from https://en.wikipedia.org/wiki/Continuous_integration

Environments

- Do not push the code into production environment right away
- Dev → Stage → Production environments
- Stage environments vary depending on your product, e.g.:
 - Show changes to a small number of
 - internal users
 - external users

Key points

- System building is the process of assembling system components into an executable program to run on a target computer system.
- Software should be frequently rebuilt and tested immediately after a new version has been built. This makes it easier to detect bugs and problems that have been introduced since the last build.
 - Ideally after every commit, but for a large system it may be too expensive
- System releases include executable code, data files, configuration files and documentation. Release management involves making decisions on system release dates, preparing all information for distribution and documenting each system release.

Travis




Let's try to setup continuous integration

?

- Why automated test cases are important?
 - Many reasons, but one is regression testing
- Also, remember Test Driven Development from CPS406?

There is also integration of Travis and GitHub

Commits on Feb 1, 2017

	"Improving" functionality miranska committed 19 minutes ago	✗ Failure: The Travis CI build failed
	Intitial commit miranska committed 21 minutes ago	✓
	Initial commit miranska committed an hour ago	

Here is how configuration scripts are built

- <https://docs.travis-ci.com/user/languages/python/>
 - And see there examples for other languages
- The scripts are flexible and can be made more complex:
 - <https://github.com/tornadoweb/tornado/blob/master/.travis.yml>

Let's give it a try

- Register via GitHub Student pack
- Connect Travis to your GitHub repo
 - https://travis-ci.com/profile/your_user_id

It seems that public ones are shown here too anyway...

We're only showing your private repositories. You can find your public projects on travis-ci.org.



Flick the repository switch on



Add .travis.yml file to your repository



Trigger your first build with a git push

Let's give it a try


- Create .travis.yml in the **root** directory of your repository (check the one in <https://github.com/miranska/cps847-w18/commit/420b750b7730e2c7faf0ebcd44e4d3140e0da68a>)
 - What does . in front of the file name mean?
- Check sample code and unittest in https://github.com/miranska/cps847-w18/tree/master/ci_play
- It's alive!
 - Note that you do not have to run Travis on every branch; this may save you some resources: <https://docs.travis-ci.com/user/customizing-the-build#Building-Specific-Branches>
- Explore how the commits info has changed in GitHub
- Let's see what happens if we fail a unit test

Travis notifications

- You will get notifications by email

 **miranska / tst_travis** (master)


✓ **Build #1 passed.** 29 seconds

 **Andriy Miranskyy**
Initial commit

[04d932d Changeset](#) →

 **miranska / tst_travis** (master)

⚠ **Build #2 was broken.** 25 seconds

 **Andriy Miranskyy**
"Improving" functionality

[ea2d101 Changeset](#) →

Integrate Code Coverage Tools with Travis

Let's talk a bit more about continuous integration

?

- Why code coverage info is useful
 - We established that automatic tests are useful
 - But what are we covering with these tests?

Integration with Code Coverage Services

- <https://codecov.io/>
 - This one gives one private repo for free
 - So let's use it
- <https://coveralls.io/>
- There is also plenty of tools that you can run locally. In fact in our example codecov.io ingests the data from one of such tools.
 - But a report that the whole test has access too is more helpful

Codecov.io: Code Coverage

- Examples for integration with your language can be found at
 - <https://github.com/codecov>
 - <https://docs.codecov.io/docs/supported-languages>
- Let's tweak Travis YAML file to enable integration
 - <https://github.com/miranska/cps847-w18/commit/feab136d3f509ee2e8d4e83ff60e083c3672c252>
- Essentially, we will re-run test cases after *successful* execution of automatic test cases and then send coverage report to codecov.io for visualization
 - Note that this process is redundant, in practice we do need to run test cases twice (for most languages)

References

- <https://docs.travis-ci.com/user/getting-started>

Summary

- Continuous Integration (CI)
 - Need to move fast, automate and outsource as much as possible, CI is your friend
- Practicum
 - Travis CI
 - Get your free access via <https://education.github.com/pack>
 - Code coverage