# System Building and Change Management

Idaho State University | Computer Science

## Isaac Griffith

CS 3321
Department of Computer Science
Idaho State University

ROAR

# Topics Covered

- System building
- Change management

# 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.
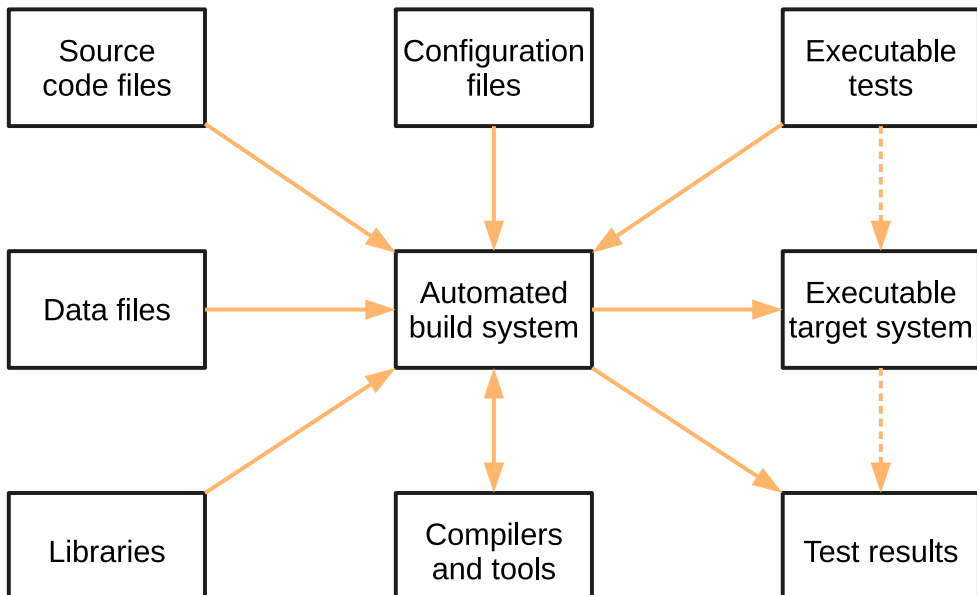
ROAR

# 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.

ROAR

# System building

# Build system functionality

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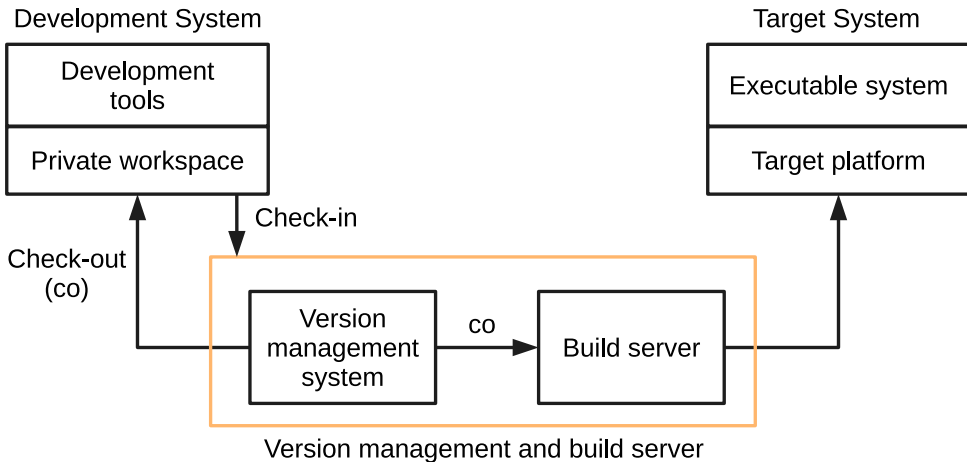- Reporting
- Documentation generation

# System platforms

- The development system, which includes development tools such as compilers, source code editors, etc.

- The build server, which is used to build definitive, executable versions of the system. This server maintains the definitive versions of a system.

- The target environment, which is the platform on which the system executes.
  - For real-time and embedded systems, the target environment is often smaller and simpler than the development environment (e.g. a cell phone)

# Development, build, and target platforms

Development System

| Development tools |
| --- |
| Private workspace |

Target System

| Executable system |
| --- |
| Target platform |

Check-in

Check-out (co)

co

| Version management system | → | Build server |

Version management and build server

# Agile building

- Check out the mainline system from the version management system into the developer's private workspace.

- Build the system and run automated tests to ensure that the built system passes all tests. If not, the build is broken and you should inform whoever checked in the last baseline system. They are responsible for repairing the problem.

- Make the changes to the system components.

- Build the system in the private workspace and rerun system tests. If the tests fail, continue editing.

# 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.

ROAR

# Continuous integration

<Slide 41>
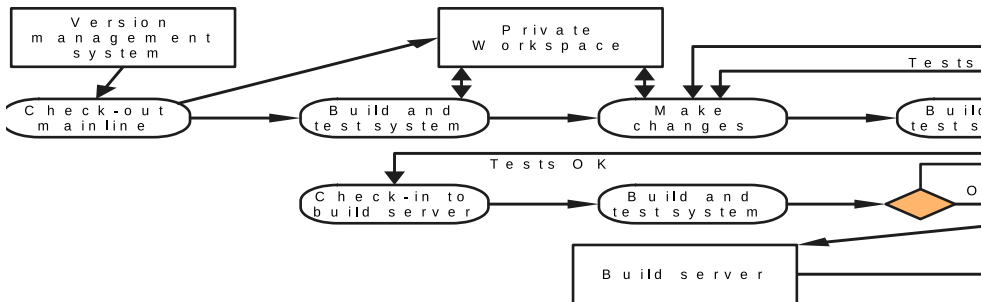
# Pros and cons of continuous integration

- Pros
  - The advantage of continuous integration is that it allows problems caused by the interactions between different developers to be discovered and repaired as soon as possible.
  - The most recent system in the mainline is the definitive working system.

- Cons
  - If the system is very large, it may take a long time to build and test, especially if integration with other application systems is involved.
  - If the development platform is different from the target platform, it may not be possible to run system tests in the developer's private workspace.

ROAR

# 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.

ROAR

# Minimizing recompilation

- Tools to support system building are usually designed to minimize the amount of compilation that is required.

- They do this by checking if a compiled version of a component is available. If so, there is no need to recompile that component.

- A unique signature identifies each source and object code version and is changed when the source code is edited.

- By comparing the signatures on the source and object code files, it is possible to decide if the source code was used to generate the object code component.

# File identification

- Modification timestamps
  - The signature on the source code file is the time and date when that file was modified. If the source code file of a component has been modified after the related object code file, then the system assumes that recompilation to create a new object code file is necessary.

- Source code checksums
  - The signature on the source code file is a checksum calculated from data in the file. A checksum function calculates a unique number using the source text as input. If you change the source code (even by 1 character), this will generate a different checksum. You can therefore be confident that source code files with different checksums are actually different.
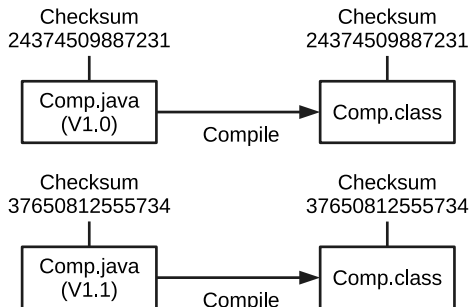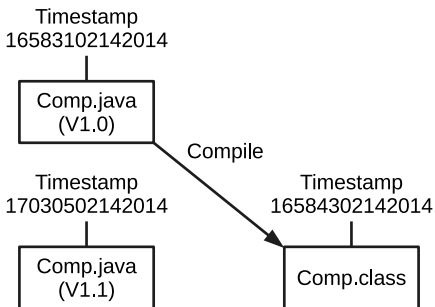
ROAR

# Timestamps vs checksums

- Timestamps
  - Because source and object files are linked by name rather than an explicit source file signature, it is not usually possible to build different versions of a source code component into the same directory at the same time, as these would generate object files with the same name.

- Checksums
  - When you recompile a component, it does not overwrite the object code, as would normally be the case when the timestamp is used. Rather, it generates a new object code file and tags it with the source code signature. Parallel compilation is possible and different versions of a component may be compiled at the same time.

# Linking source and object code

Timestamp
16583102142014

Comp.java
(V1.0)

Compile

Timestamp
16584302142014

Comp.class

Timestamp
17030502142014

Comp.java
(V1.1)

Checksum
24374509887231

Comp.java
(V1.0)

Compile

Checksum
24374509887231

Comp.class

Checksum
37650812555734

Comp.java
(V1.1)

Compile

Checksum
37650812555734

Comp.class

ROAR

# Change management

# Change management

- Organizational needs and requirements change during the lifetime of a system, bugs have to be repaired and systems have to adapt to changes in their environment.

- Change management is intended to ensure that system evolution is a managed process and that priority is given to the most urgent and cost-effective changes.

- The change management process is concerned with analyzing the costs and benefits of proposed changes, approving those changes that are worthwhile and tracking which components in the system have been changed.
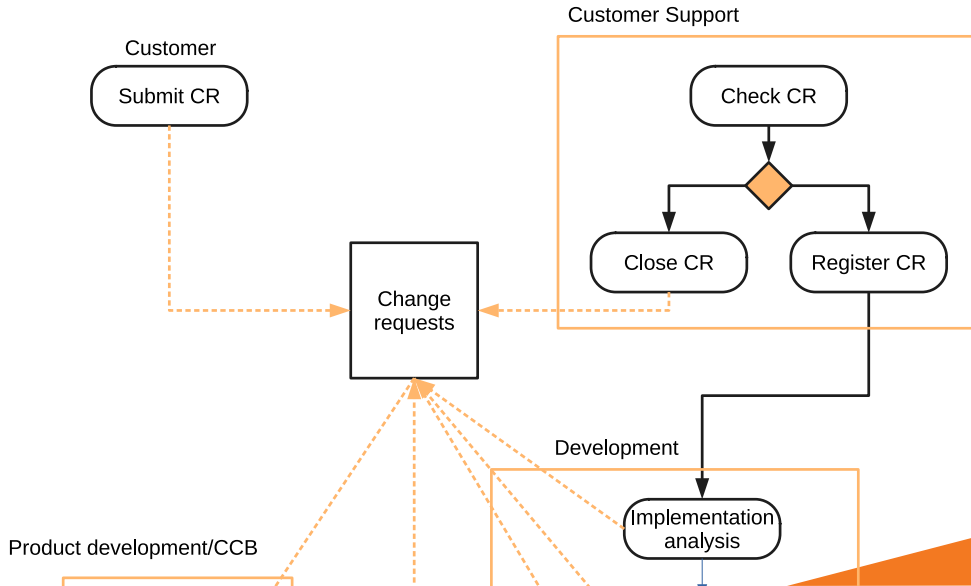
# A partially completed change request form

**Change Request Form**

**Project:** SICSA/AppProcessing          **Number:** 23/02
**Change requester:** I. Sommerville          **Date:** 20/07/12
**Requested change:** The status of applicants (rejected, accepted, etc.) should be shown visually in the displayed list of applicants.

**Change analyzer:** R. Looek          **Analysis date:** 25/07/12
**Components affected:** ApplicantListDisplay, StatusUpdater

**Associated components:** StudentDatabase

# A partially completed change request form

**Change Request Form**

**Change assessment:** Relatively simple to implement by changing the display color according to status. A table must be added to relate status to colors. No changes to associated components are required.

**Change priority:** Medium
**Change implementation:**
**Estimated effort:** 2 hours
**Date to SGA app. team:** 28/07/12   **CCB decision date:** 30/07/12
**Decision:** Accept change. Change to be implemented in Release 1.2
**Change implementor:**    **Date of change:**
**Date submitted to QA:**   **QA decision:**
**Date submitted to CM:**
**Comments:**

ROAR

# Factors in change analysis

- The consequences of not making the change
- The benefits of the change
- The number of users affected by the change
- The costs of making the change
- The product release cycle

```
// SICSA project (XEP 60807)
//
// APP-SYSTEM/AUTH/RBAC/USER_ROLE
//
// Object: currentRole
// Author: R. Looek
// Creation Date: 13/11/2012
//
// Copyright St. Andrews University 2012
//
// Modification history
// Version    Modifier    Date         Change       Reason
// 1.0        J. Jones    11/11/2009   Add header    Submitted to CM
// 1.1        R. Looek    13/11/2012   New field     Change req. R07/02
```

ROAR

# Change management and agile methods

- In some agile methods, customers are directly involved in change management.

- The propose a change to the requirements and work with the team to assess its impact and decide whether the change should take priority over the features planned for the next increment of the system.

- Changes to improve the software improvement are decided by the programmers working on the system.

- Refactoring, where the software is continually improved, is not seen as an overhead but as a necessary part of the development process.

ROAR

# 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.

- Change management involves assessing proposals for changes from system customers and other stakeholders and deciding if it is cost-effective to implement these in a new version of a system.

# Are there any questions?

ROAR