

Dr. Michael Eichberg

Software Engineering

Department of Computer Science

Technische Universität Darmstadt

Software Engineering

Building Software



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Non-trivial Software is generally Build using Build Automation Systems.

- The goal of a Build Automation System is to **fully automate all steps** required to build the product given the source artifacts of the project.

The result of the build should always be the same - independent of the developer's local configuration.

“We want stable builds.”

The Build Automation Systems is responsible for automatically carrying out all steps necessary to build the product.

- A Build Automation typically executes the following tasks:
 - Formatting the source code
 - Code Generation
 - Source Code Compilation
 - [if necessary] Linking Code/Packaging Code
 - Running the tests
 - Running static analysis tools
 - Deployment to the test system/production system(s)
 - Creating and publishing documentation, release notes, web pages, ...



Historically

Software is Build using Build Automation Systems.

- Given a Build Automation System, the product can be built:
 - On-Demand
(e.g., by a developer)
 - Scheduled by a build server
(e.g., every night)
 - Triggered
(e.g., on every commit to a version control system)

Historically

State of
the Art

Some Examples of (Open-Source) Tools to Automate Builds

- The family of make tools!

- Apache Ant

uses XML

- Apache Maven

Automated Dependency Management
(To get stable builds.)

Historically

- gradle (Groovy Based)

- RAKE (Ruby Make)

- sbt

- ...

Internal DSLs

State of the Art

rough timeline



```

import AssemblyKeys._

name := "BugPicker"

version := "1.1.0"
scalaVersion := "2.11.4"

scalacOptions in (Compile, doc) := Seq("--deprecation", "--feature", "--unchecked")
scalacOptions in (Compile, doc) += Opts.doc.title("OPAL - BugPicker")

libraryDependencies += "org.scalafx" %% "scalafx" % "1.0.0-R8"

jfxSettings
JFX.addJfxrtToClasspath := true
JFX.mainClass := Some("org.opalj.bugpicker.BugPicker")

assemblySettings
jarName in assembly := "bugpicker-" + version.value + ".jar"
test in assembly := {}
mainClass in assembly := Some("org.opalj.bugpicker.BugPicker")

resourceGenerators in Compile <+= Def.task {
  val versionFile = (baseDirectory in Compile).value / "target" / "scala-2.11" / "classes" / "org" /
"opalj" / "bugpicker" / "version.txt"
  versionFile.getParentFile.mkdirs()
  IO.write(versionFile, (version in Compile).value)
  Seq(versionFile)
}

```

Version Information

Compiler Settings

Project Dependencies

Project Settings

Deployment information

Generation of other Artifacts

Easily hundreds of lines for larger projects.

- Continuous integration basically just means that the developer's working copies are synchronized with a shared mainline several times a day.
It was first named and proposed by Grady Booch.
- The goal is to avoid integration issues.
- CI is in particular useful in combination with automated unit tests.
- In practice a special build server is used.
(e.g., Hudson/Jenkins)

- 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
One commit - one feature; no "Mega-commits"
- Keep the build fast
- 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

- A hosted continuous integration service for open source and private projects.

The screenshot displays the Travis CI web interface. On the left, a sidebar shows a search bar and a list of recent repositories. The main content area shows the details for the repository `angular/angular.js`, including a description, build history tabs, and a build log for a recent commit. The build log shows the commit message, a description of the fix, and a build matrix table.

Repository Overview:

Repository	Count
angular/angular.js	15292
SC5/sc5-styleguide	470
robmorgan/phinx	745
zhiyee/mdserver	17
AnyFetch/dropbox-provider.anyfetc...	285
yandex-shri-ekb-2014/team1	30

Build Log for #15292:

```
master - fix($filter): add int support for negated strict comparison
- negation is achieved by adding '!' which fails if key is a integer and strict
  comparison is set to true.
- This commit fixes it by checking typeof and parsing accordingly

Closes #10141

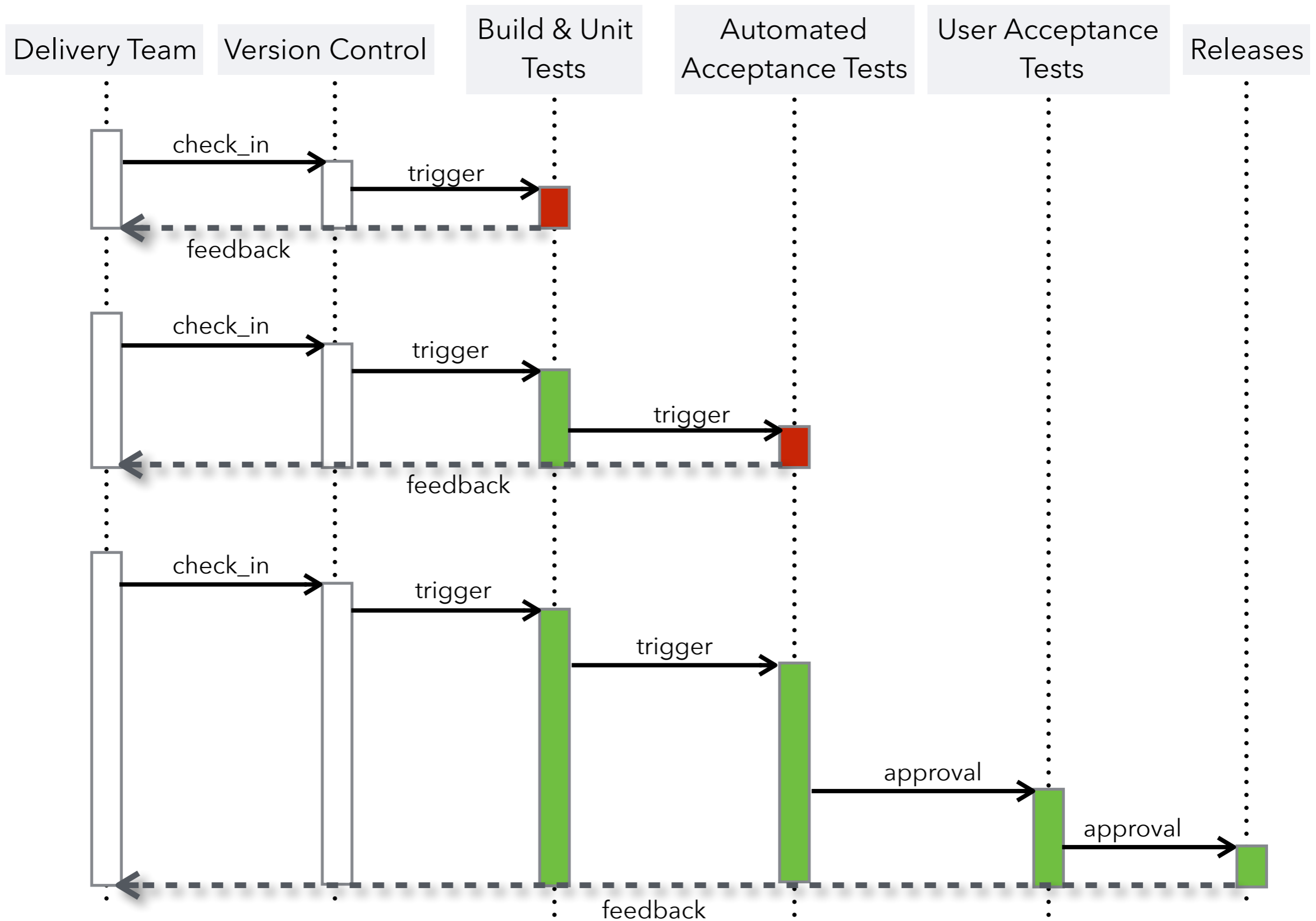
Adi Chikara authored and committed Commit 650ddda #10145: fix($filter): add int support for negated strict comparison
```

Build Matrix:

Job	Duration	Finished	Node.js	ENV	OS
15292.1	8 sec	-	0.10	JOB=unit	linux
15292.2	8 sec	-	0.10	JOB=e2e TEST_TARGET=jqlite	linux
15292.3	-	-	0.10	JOB=e2e TEST_TARGET=jquery	linux

- *Always be able to put a product into production (The evolution of continuous integration.)*
- Practices
 - Unit/Acceptance-tests
 - Code coverage and static analysis
 - Deployment to integration environment
 - Integration tests
 - Deployments to performance test environment
 - Performance tests
 - Alerts, reports and release notes sent out
 - Deployment to release repository

Continuous Delivery

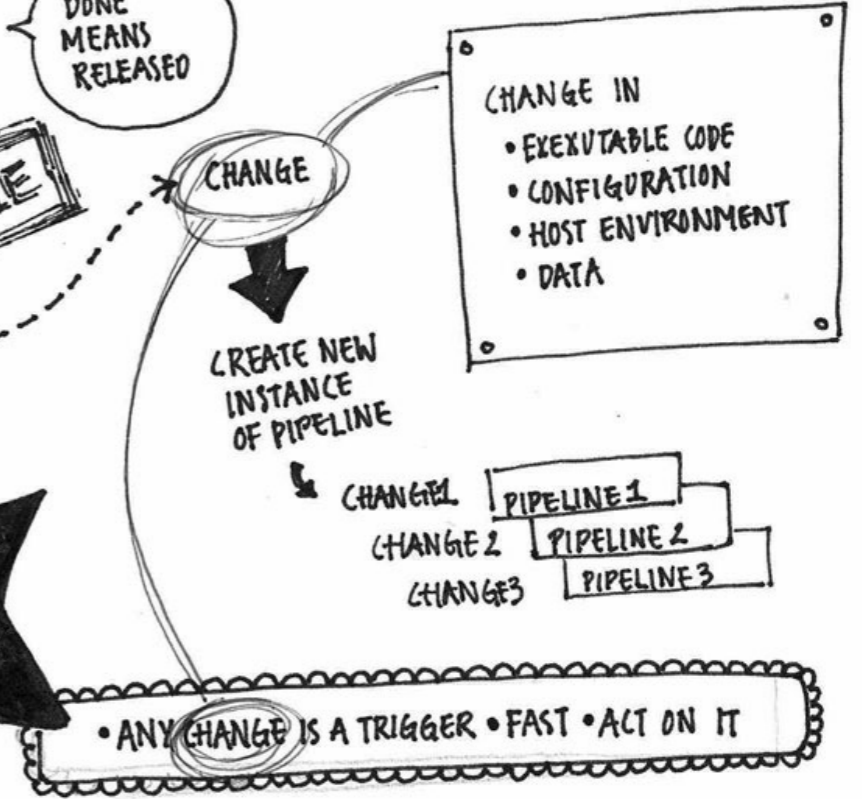
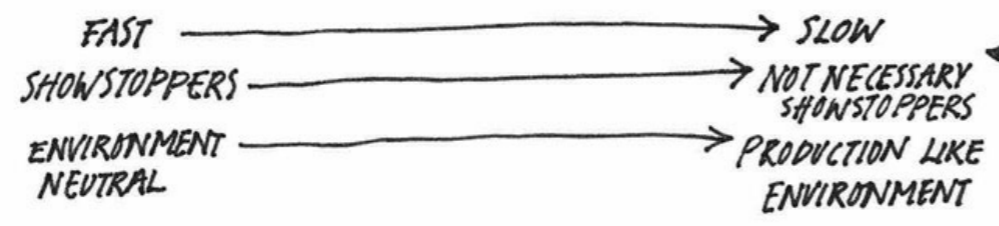
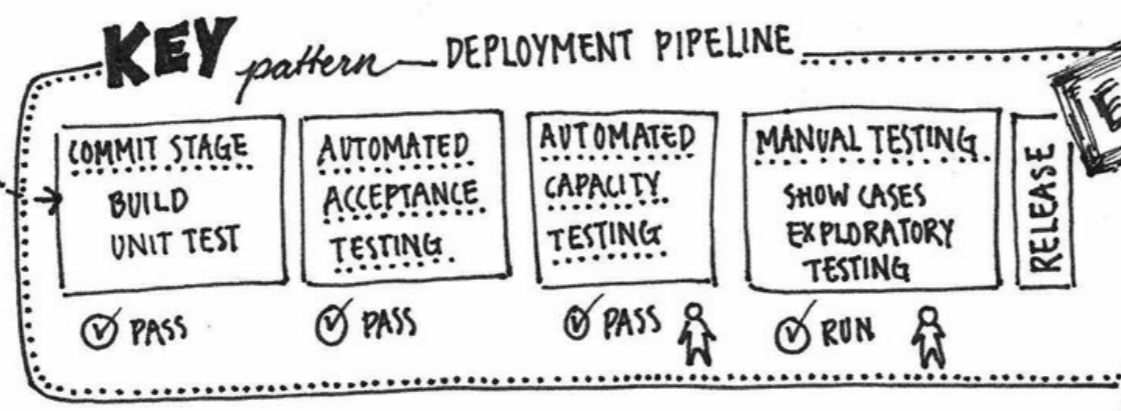




CONTINUOUS DELIVERY

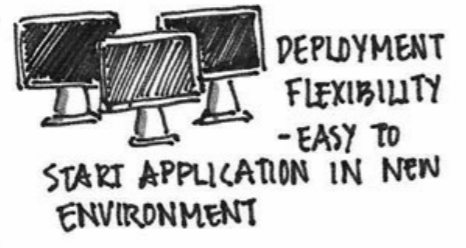
BY JEZ HUMBLE & DAVID FARLEY

- A CLOSER LOOK - COMMIT STAGE**
- ✓ CREATING EXECUTABLE CODE MUST WORK. VERIFIES THAT THE SYNTAX OF YOUR SOURCE CODE IS VALID
 - ✓ UNIT TEST PASS
 - ✓ FULFILL CERTAIN QUALITY CRITERIA SUCH AS TEST COVERAGE AND OTHER TECHNOLOGY-SPECIFIC METRICS



BENEFITS

- EMPOWERED - IN CONTROL
- LOW STRESS - SMALL RELEASES
- REDUCING ERRORS - CONFIG MGT. - VERSION CONTROL



SEEMS LIKE THE AUTHORS CAN'T STRESS IT ENOUGH. IT'S EVERYWHERE THROUGHOUT THIS BOOK.



“ ENCOURAGING GREATER COLLABORATION BETWEEN EVERYONE INVOLVED IN SOFTWARE DELIVERY IN ORDER TO RELEASE VALUABLE SOFTWARE FASTER AND MORE RELIABLY. ”

If it hurts, do it more frequently

Cloud Services for Continuous Delivery

The screenshot displays the Shippable web interface. At the top, the Shippable logo and navigation menu (FEATURES, PRICING, DOCS, ABOUT US, BLOG, LOGIN) are visible. The main content area shows a build summary for project 'delors/opal' on the 'master' branch. The build started 3 hours ago and lasted 12 minutes. The build status is '118.1' (likely a version or build number). The build details include:

- Project: delors/opal
- Branch: master
- Image: shippable/minv2
- Started at: 3 hours ago
- Commit SHA: 7622f5e
- Committer: delors
- Duration: 12 minutes
- Matrix Values: runtime=2.11.2 jdk=oraclejdk8
- Allow Failure: false
- Pull Request: false
- Commit Message: the bugpicker now shows all lines associated with an issue report Signed-off-by: Michael Eichberg <mail@michael-eichberg.de>

Below the build details, there are four summary cards:

- 1759 Passing
- 4 Failures
- 0 Errors
- 34 Skipped

A console log snippet shows a test failure: `class org.scalatest.exceptions.TestFailedException: expected: MetaInformationUpdate; actual: NoUpdate`.

The right side of the interface shows a 'Build History' table with columns: Status, Triggered, Duration, Changeset, Branch, Committer, and Actions. The table lists several successful builds:

Status	Triggered	Duration	Changeset	Branch	Committer	Actions
success	Today at 12:50 PM	12 minutes	7622f5e	master	Michael Eichberg	Refresh, Delete
success	Today at 10:23 AM	8 minutes	5c48f82	master	Michael Eichberg	Refresh, Delete
success	Yesterday at 3:21 PM	7 minutes	0e8616f	master	Michael Eichberg	Refresh, Delete
success	Yesterday at 2:33 PM	6 minutes	15230dd	master	Michael Eichberg	Refresh, Delete

Additional UI elements include a 'Badge' for 'build shippable', 'Queued/Running' status (No Queued/Running Builds), and 'Permissions' for the user Michael Eichberg.

Continuous Deployment

- Automatically **deploy the product into production** whenever it passes QA.
(The logical next step after Continuous Delivery)
- The release schedule is in the hands of the IT department
(With Continuous Delivery the release schedule is in the hands of the business.)

Attention: Sometimes the term "Continuous Deployment" is also used if you are able to continuously deploy to the test system.

Summary



TECHNISCHE
UNIVERSITÄT
DARMSTADT

The goal of this lecture is to enable you to systematically carry out small(er) software projects that produce quality software.

-
- Projects are build using build tools
 - A build script takes care of all steps necessary to build the project
(In case of an application, building means creating a runnable application.)

- The goal of this lecture is to enable you to systematically carry out small(er) commercial or open-source projects.

