Dr. Michael Eichberg
Software Engineering
Department of Computer Science
Technische Universität Darmstadt

Introduction to 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

Historically

  - Running the tests
  - Running static analysis tools
  - Deployment to the test system/production system(s)
  - Creating and publishing documentation, release notes, web pages, ...

Software is Build using Build Automation Systems.

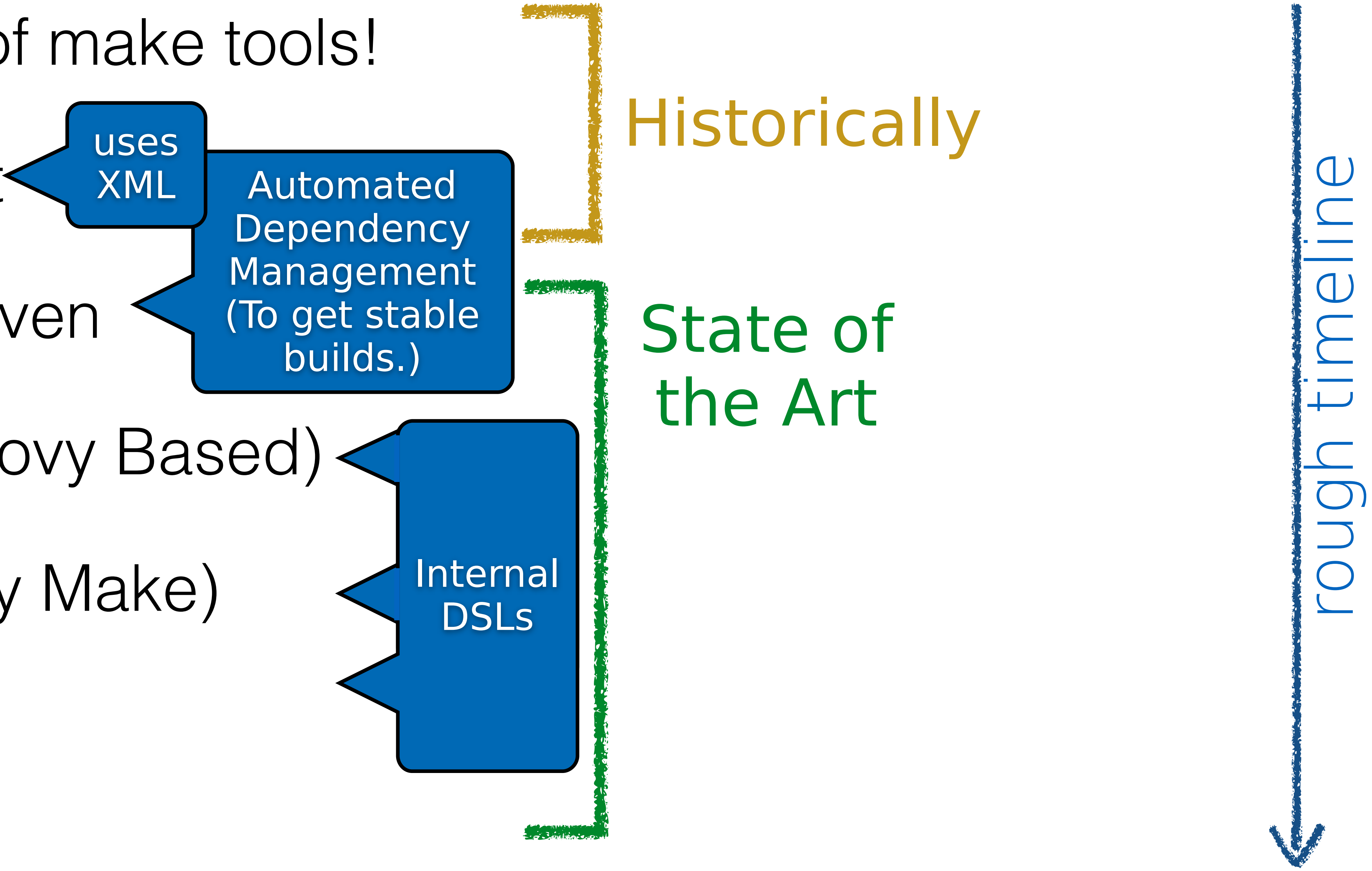
- Given a Build Automation System, the product can be built:
  - **On-Demand**
    (e.g., by a developer)
  - **Scheduled by a build server**
    (e.g., every night)
  - **Triggered**
    (e.g., on every commit to a version control system)

Historically

State of the Art

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

- The family of make tools!
- Apache Ant
- Apache Maven
- gradle (Groovy Based)
- RAKE (Ruby Make)
- sbt
- …



uses XML

Automated Dependency Management (To get stable builds.)

Internal DSLs

Historically

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", "-unc

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).v                              " / "classes" / "org" /
"opalj" / "bugpicker" / "version.txt"
 versionFile.getParentFile.mkdirs()
 IO.write(versionFile, (version in Compile).value)
```

Version Information

Compiler Settings

Project Dependencies

Project Settings

Deployment information

Generation of other Artifacts

Easily hundreds of lines for larger projects.

Continuous Integration

- 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)

Continuous Integration - Best Practices

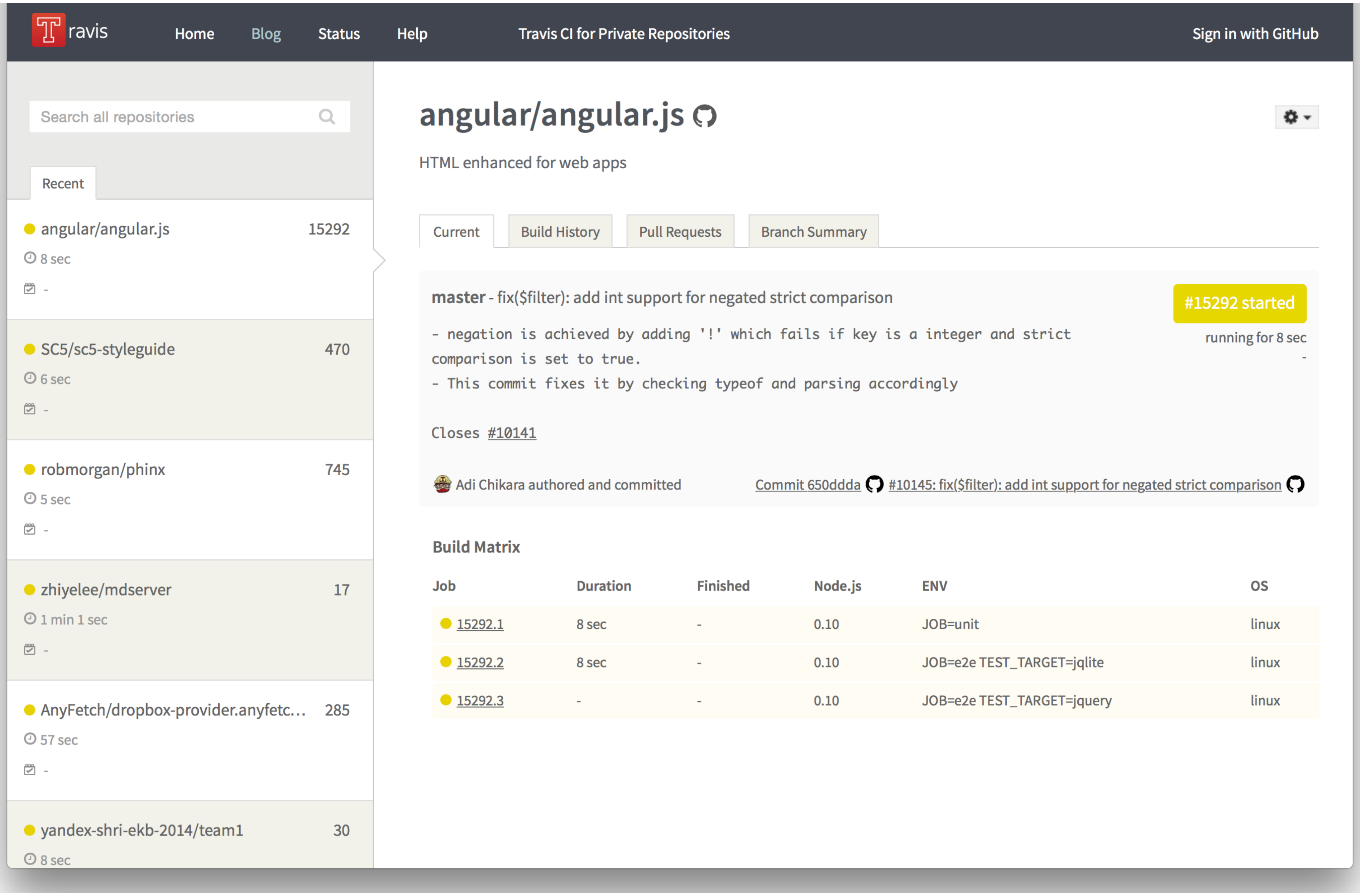
- Maintain a code repository
- Automate the build
- Make the build self-testing
- Everyone commits to the baseline every day
- Every commit (to baseline) should be built
  One commit - one feature; no "Mega-commits"
- Keep the build fast
- Test in a clone of the production environment
- Make it easy to get the latest deliverables
- Everyone can see the results of the latest build
- Automate deployment

must haves

standard

Travis CI

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



● angular/angular.js	15292	8 sec
● SC5/sc5-styleguide	470	6 sec
● robmorgan/phinx	745	5 sec
● zhiyelee/mdserver	17	1 min 1 sec
● AnyFetch/dropbox-provider.anyfetc...	285	57 sec
● yandex-shri-ekb-2014/team1	30	8 sec

angular/angular.js

HTML enhanced for web apps

master - fix($filter): add int support for negated strict comparison

#15292 started

running for 8 sec

```
- 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

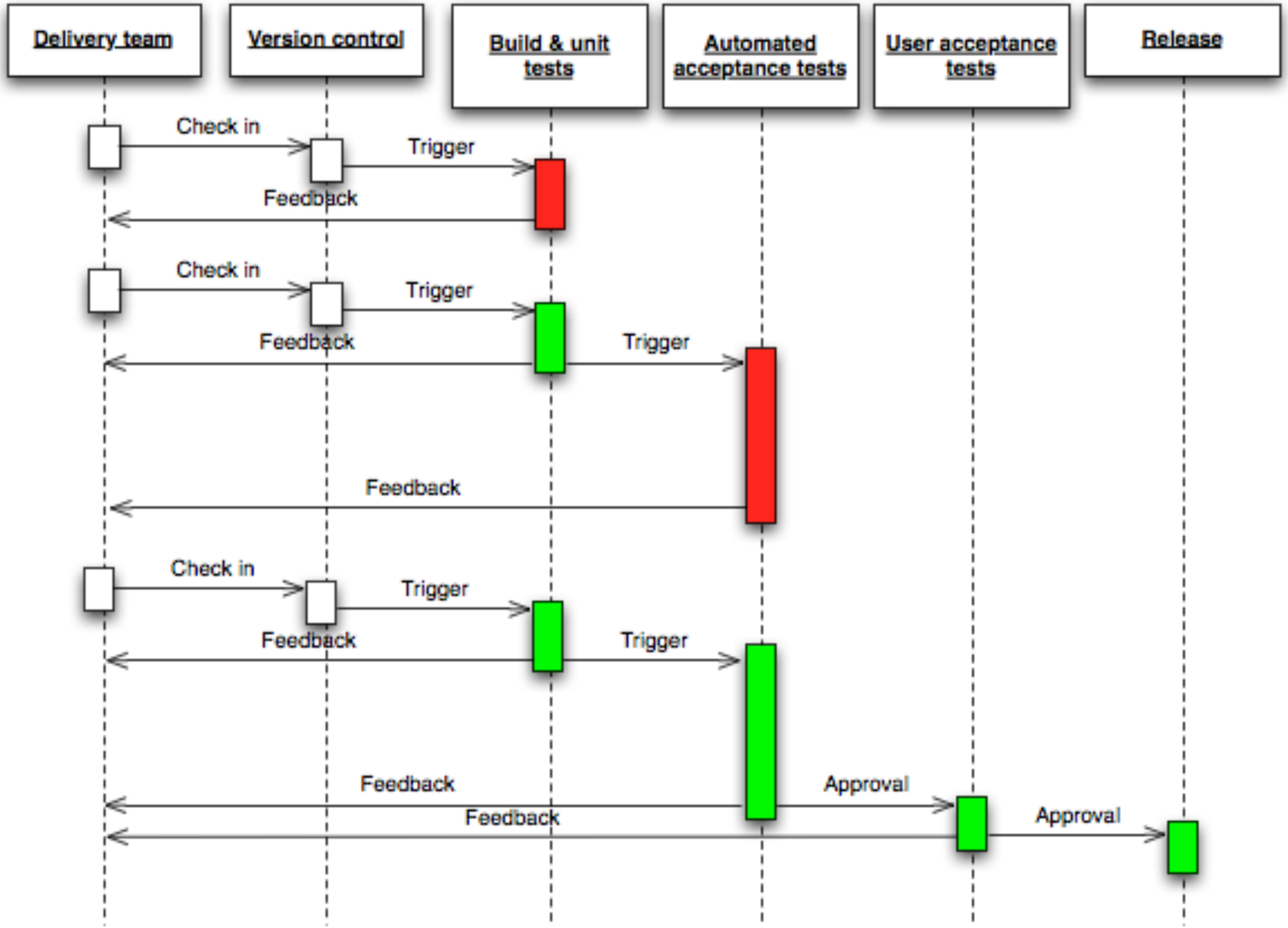
Job	Duration	Finished	Node.js	ENV	OS
● 15292.1	8 sec	-	0.10	JOB=unit	linux
● 15292.2	8 sec	-	0.10	JOB=e2e TEST_TARGET=jqlite	linux
● 15292.3	-	-	0.10	JOB=e2e TEST_TARGET=jquery	linux

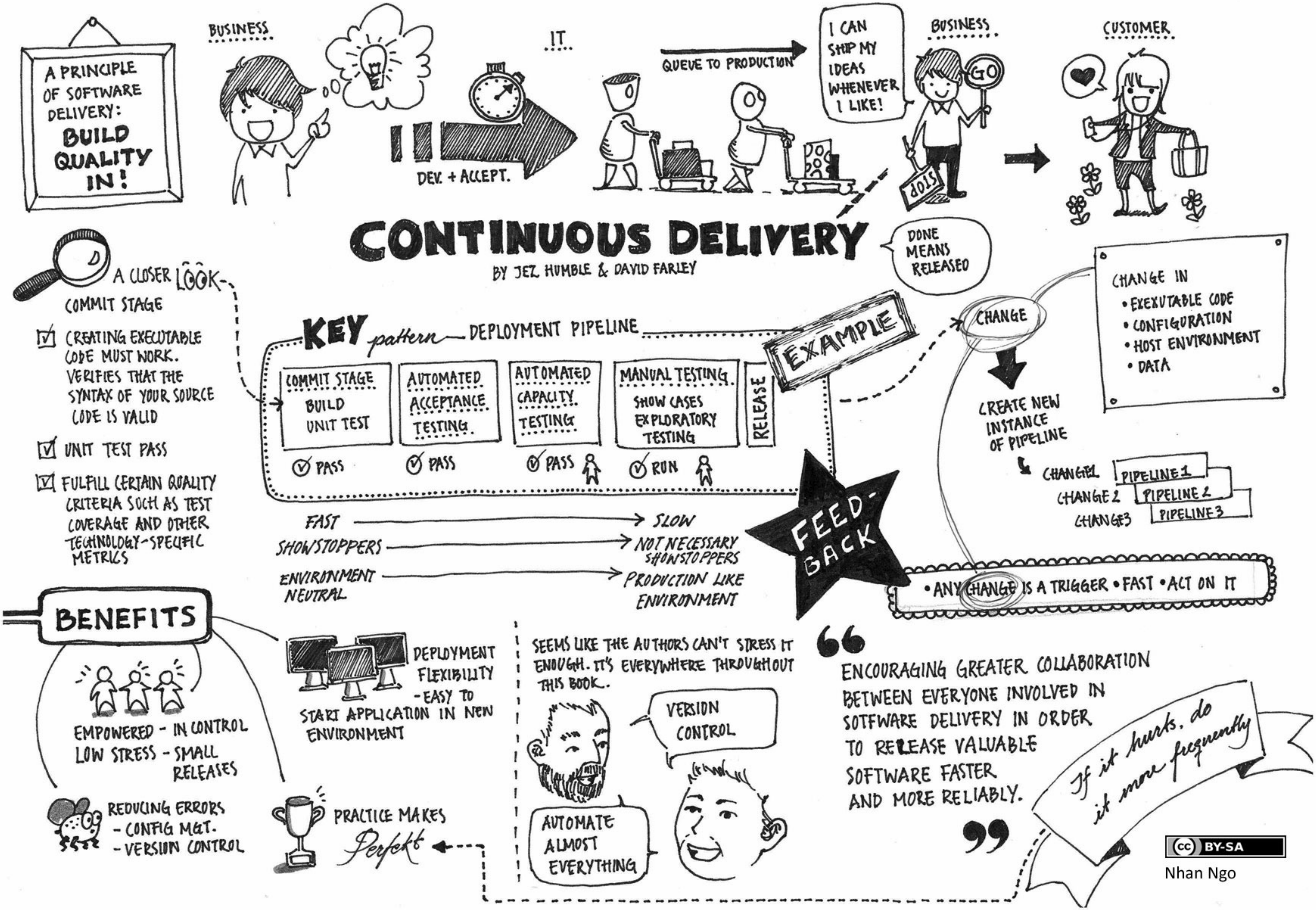
Continuous Delivery

- 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

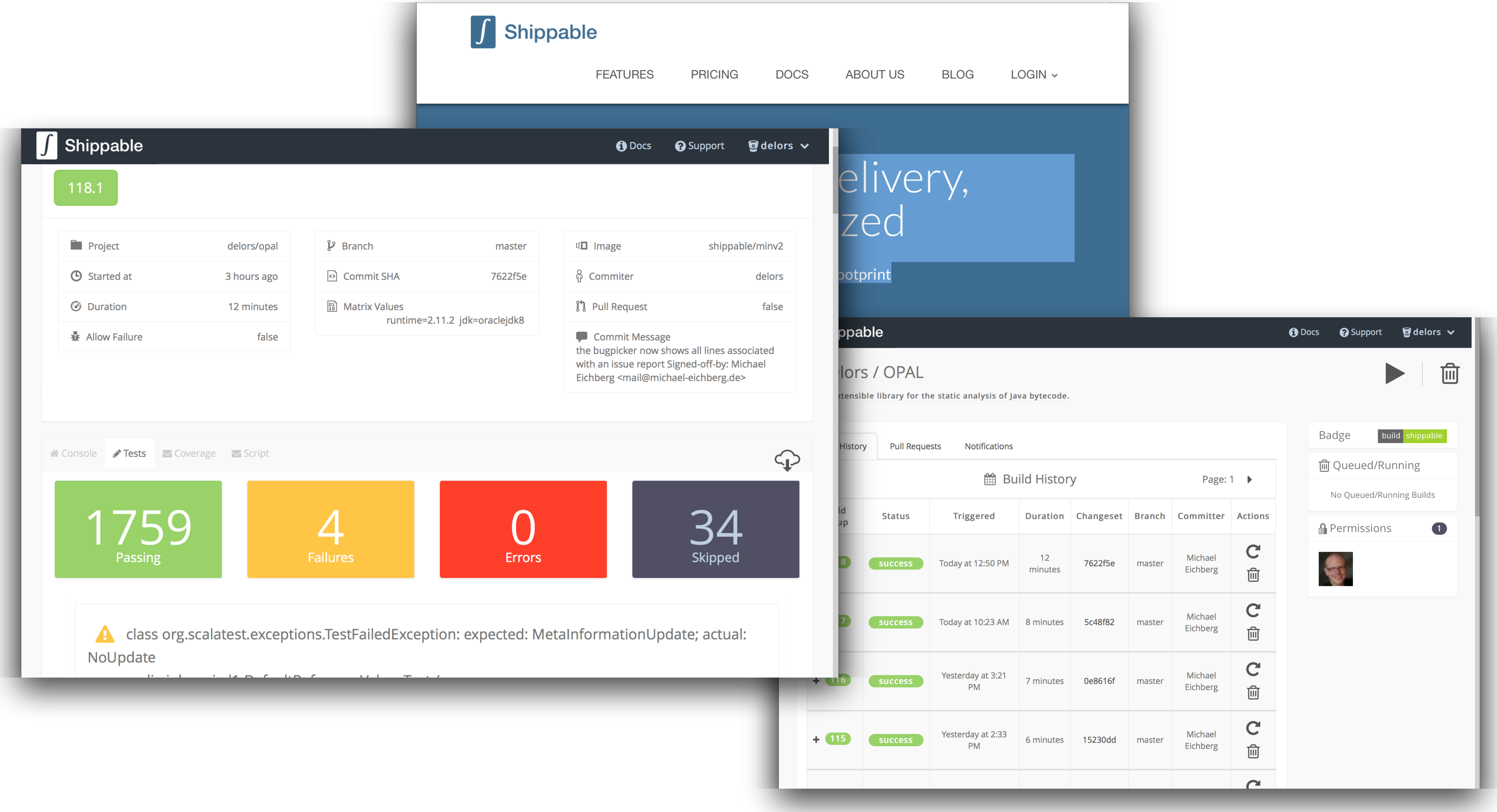
Continuous Delivery

© http://continuousdelivery.com/2010/02/continuous-delivery/ |




A PRINCIPLE OF SOFTWARE DELIVERY: BUILD QUALITY IN!
BUSINESS
IT
DEV. + ACCEPT.
QUEUE TO PRODUCTION
I CAN SHIP MY IDEAS WHENEVER I LIKE!
BUSINESS
GO
STOP
CUSTOMER
CONTINUOUS DELIVERY
BY JEZ HUMBLE & DAVID FARLEY
DONE MEANS RELEASED
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
KEY pattern — DEPLOYMENT PIPELINE
EXAMPLE
COMMIT STAGE
BUILD
UNIT TEST
✓ PASS
AUTOMATED ACCEPTANCE TESTING
✓ PASS
AUTOMATED CAPACITY TESTING
✓ PASS
MANUAL TESTING
SHOW CASES
EXPLORATORY TESTING
✓ RUN
RELEASE
FAST → SLOW
SHOWSTOPPERS → NOT NECESSARY SHOWSTOPPERS
ENVIRONMENT NEUTRAL → PRODUCTION LIKE ENVIRONMENT
FEED-BACK
CHANGE
CHANGE IN
• EXECUTABLE CODE
• CONFIGURATION
• HOST ENVIRONMENT
• DATA
CREATE NEW INSTANCE OF PIPELINE
CHANGE1 PIPELINE 1
CHANGE 2 PIPELINE 2
CHANGE3 PIPELINE 3
• ANY CHANGE IS A TRIGGER • FAST • ACT ON IT
BENEFITS
EMPOWERED - IN CONTROL
LOW STRESS - SMALL RELEASES
REDUCING ERRORS
- CONFIG MGT.
- VERSION CONTROL
DEPLOYMENT FLEXIBILITY
- EASY TO START APPLICATION IN NEW ENVIRONMENT
PRACTICE MAKES Perfekt
SEEMS LIKE THE AUTHORS CAN'T STRESS IT ENOUGH. IT'S EVERYWHERE THROUGHOUT THIS BOOK.
VERSION CONTROL
AUTOMATE ALMOST EVERYTHING
“ ENCOURAGING GREATER COLLABORATION BETWEEN EVERYONE INVOLVED IN SOTFWARE DELIVERY IN ORDER TO RELEASE VALUABLE SOFTWARE FASTER AND MORE RELIABLY. ”
If it hurts, do it more frequently
(cc) BY-SA
Nhan Ngo

Cloud Services for Continuous Delivery



Shippable

FEATURES PRICING DOCS ABOUT US BLOG LOGIN

elivery,
zed
otprint

Shippable Docs Support delors

118.1

Project	delors/opal
Started at	3 hours ago
Duration	12 minutes
Allow Failure	false

Branch	master
Commit SHA	7622f5e
Matrix Values	runtime=2.11.2 jdk=oraclejdk8

Image	shippable/minv2
Commiter	delors
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>

Console Tests Coverage Script

1759 Passing | 4 Failures | 0 Errors | 34 Skipped

class org.scalatest.exceptions.TestFailedException: expected: MetaInformationUpdate; actual: NoUpdate

ppable Docs Support delors

lors / OPAL

tensible library for the static analysis of Java bytecode.

History Pull Requests Notifications

Build History Page: 1

ld up	Status	Triggered	Duration	Changeset	Branch	Committer	Actions
8	success	Today at 12:50 PM	12 minutes	7622f5e	master	Michael Eichberg	
7	success	Today at 10:23 AM	8 minutes	5c48f82	master	Michael Eichberg	
+ 116	success	Yesterday at 3:21 PM	7 minutes	0e8616f	master	Michael Eichberg	
+ 115	success	Yesterday at 2:33 PM	6 minutes	15230dd	master	Michael Eichberg	

Badge build shippable

Queued/Running

No Queued/Running Builds

Permissions 1

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 hans of the It
  (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.

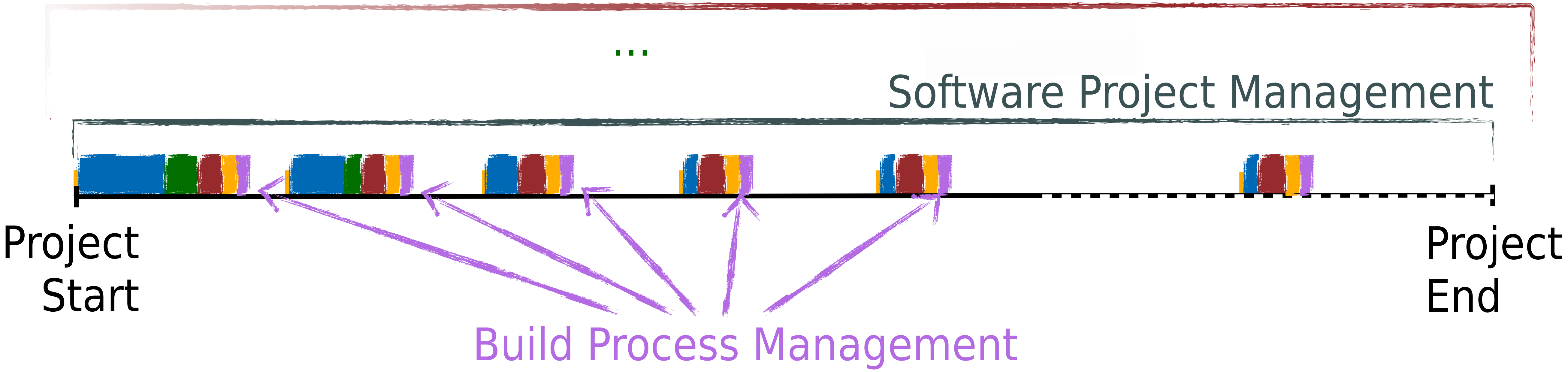
Summary

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.



🟦 Requirements Management

🟩 Domain Modeling

🟧 Modeling

🟥 Testing

🟪 Build Process Management