Build Automation, Continous Integration

Budapesti Műszaki és Gazdaságtudományi Egyetem
Méréstechnika és Információs Rendszerek Tanszék

Recap: Testing Systems

- Preparing tests
  - Multiple metodologies
  - Goal: increase quality, find issues
- Problem
  - Good testing requires time
  - Developer is prone not to execute it locally

Mozilla Firefox

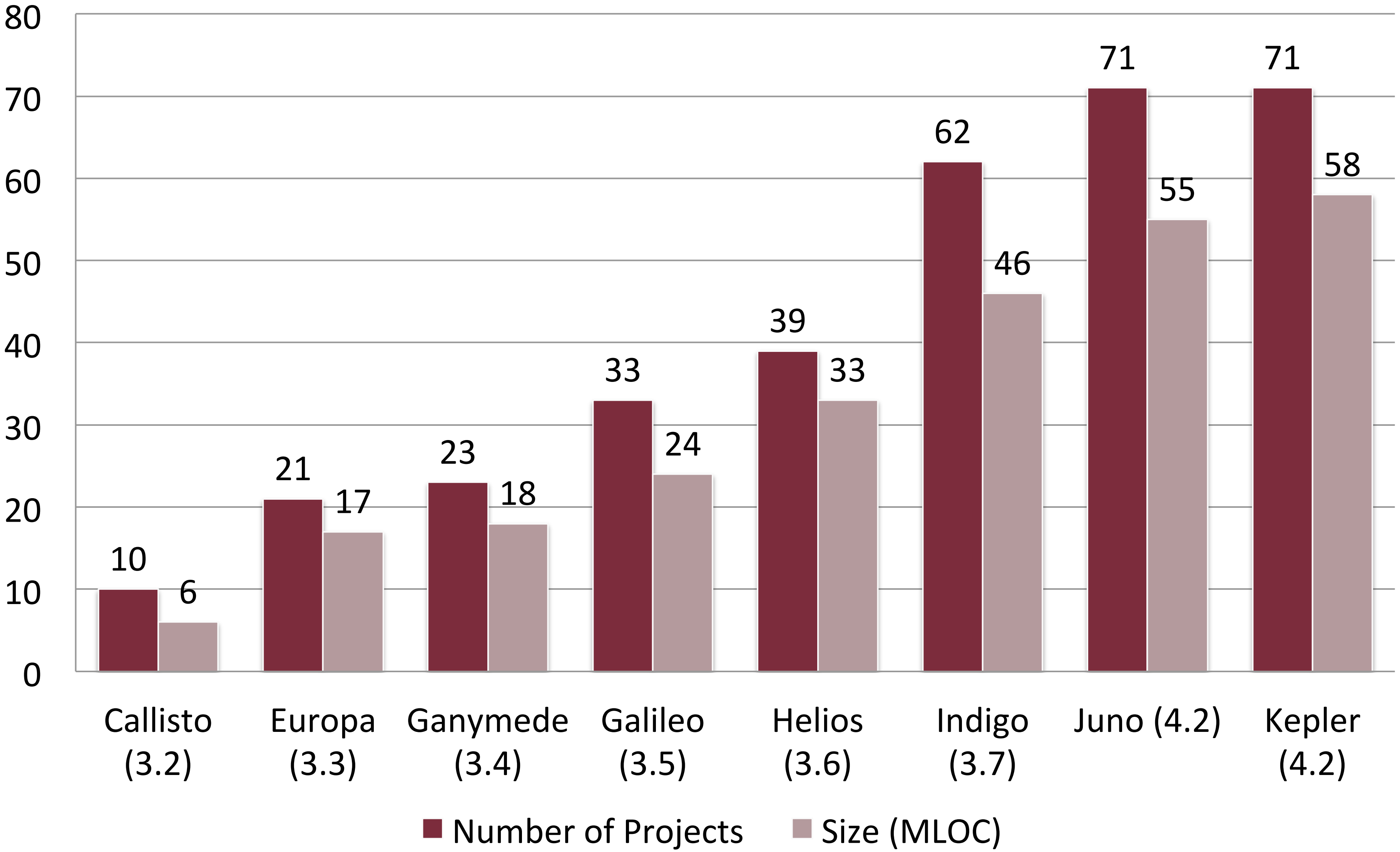
- 17 platform
- 12 source branch
- 1200 build and test machine
  - Compile time: 12.40 hours
  - Testing time: 54.48 hours
  - CPU time: 2.79 days (!)
  - Release testing earlier: 10 days

Forrás: http://relengofthenerds.blogspot.com/2010/11/mozilla-versus-eclipse-build.html

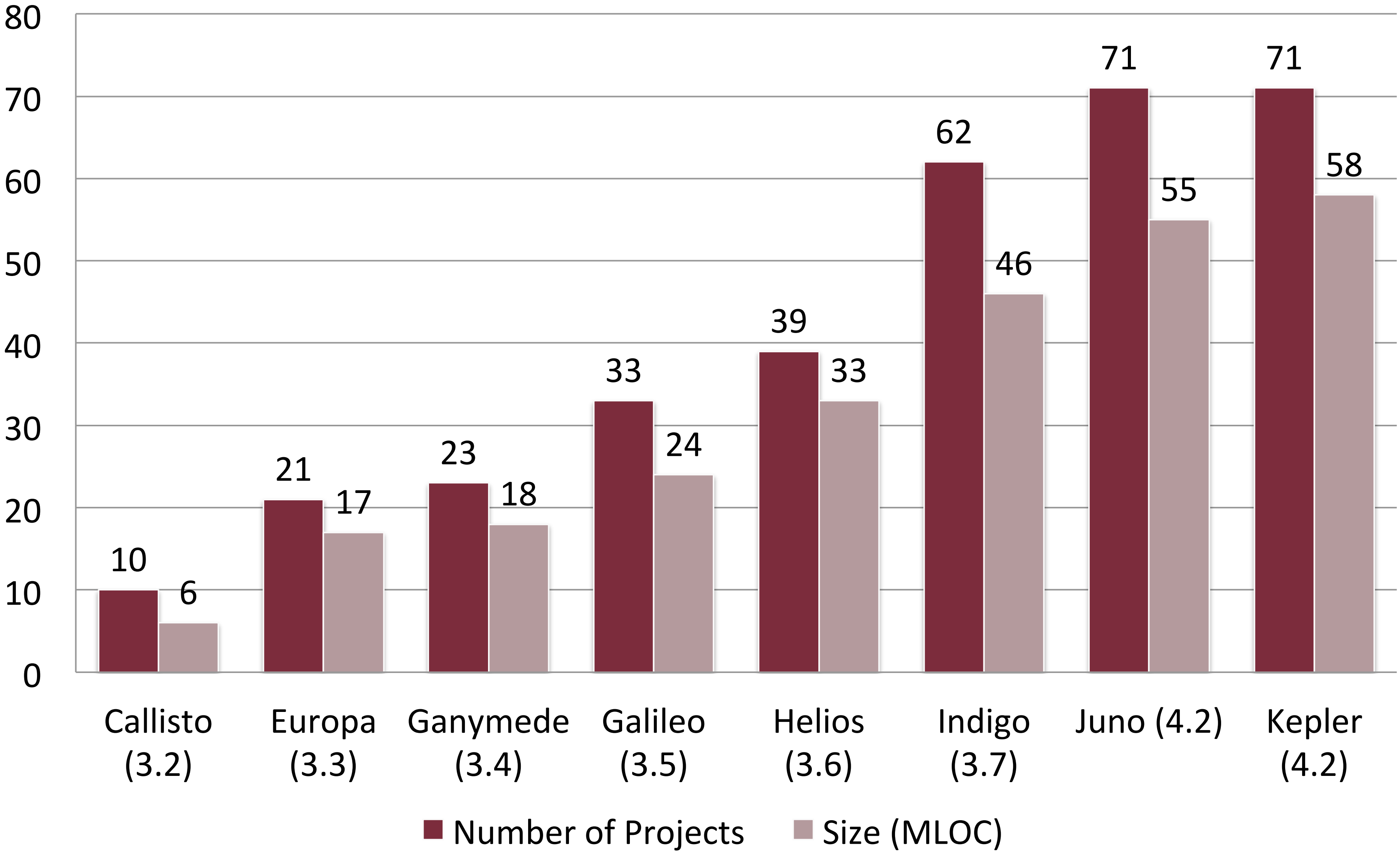
Eclipse Release Train

- Synchronized release of projects
  - Since 2006
  - Yearly
    - 1 main release (new features)
    - 2 service releases (mostly patches)

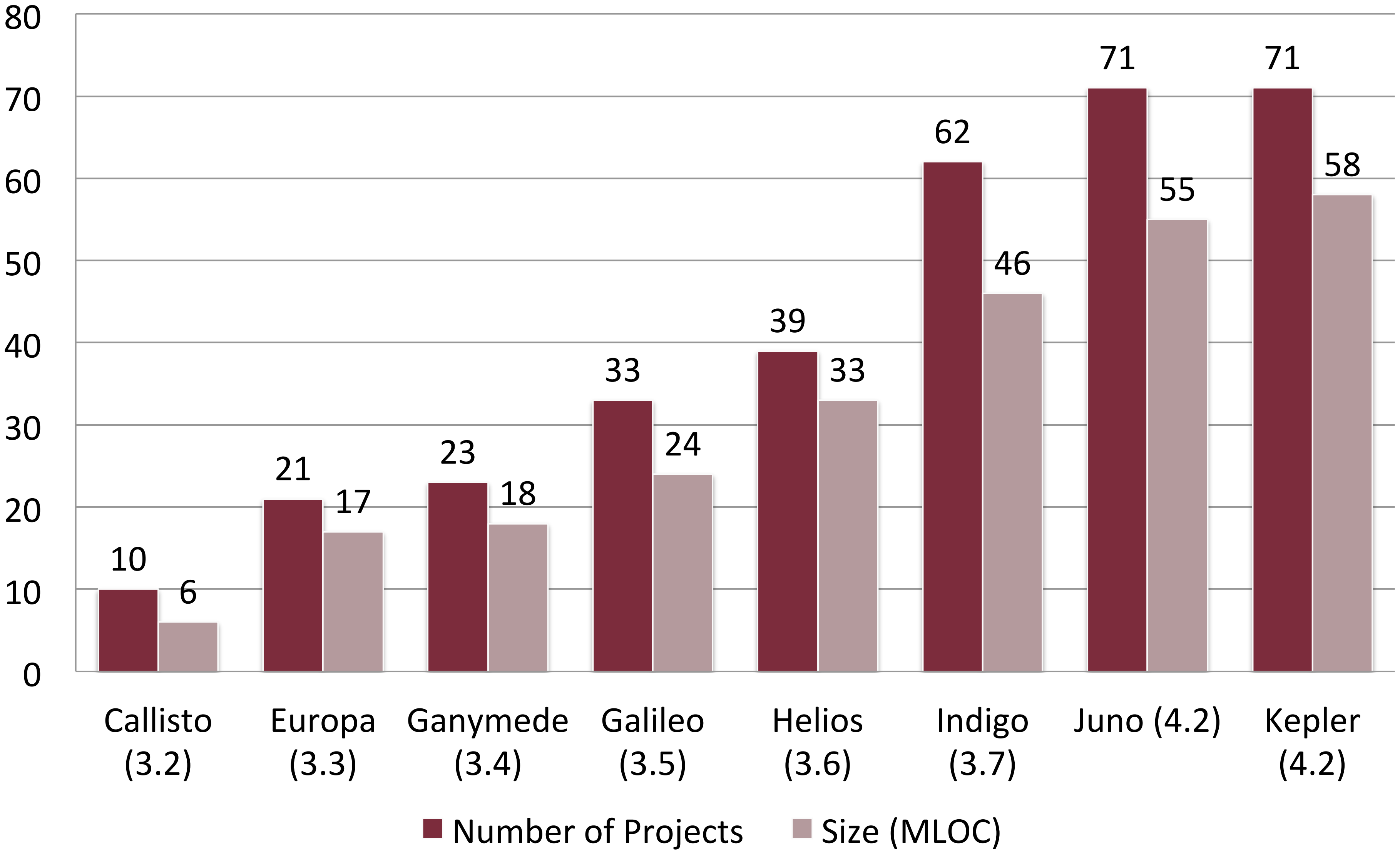
Eclipse Release Train in Numbers



Eclipse Release Train in Numbers



Eclipse Release Train in Numbers



Multiple versions

- Different platforms
  - Windows
    - Win32 32/64 bit
    - There was an early access WPF port
  - Linux
    - GTK 32/64 bit
  - Mac OSX
    - Cocoa 64 bit

Multiple packages

- Package
  - Different set of plug-ins installed together
  - All other plug-ins available for downloads
- Examples
  - Java EE
  - Plug-in developer
  - C/C++
  - Modeling
  - PHP
  - …

Eclipse Platform Build (2009.11.)

Downloading source	20 minutes
Build signing	1 hour 14 minutes
Using p2 Director	20 minutes
Creating p2 repositories	4 minutes
Zipping SDK and platform zips	30 minutes
Running tests	6 hour 40 minutes

Forrás: https://bugs.eclipse.org/bugs/show_bug.cgi?id=293830#c11

Motivation for Release Train

- - Short review of "pre-Callisto" days, to avoid repeating past mistakes; (Names and examples are a fictional melding of several cases). Platform released in June. TPTP and CDT a month or two later, WTP a month or two later. Only at that time, was a bug discovered in the Platform (by WTP nearing release) such that they could not release until SR1. Platform releases SR1 in September, WTP can now release. Only then was it observed that some regression was introduced that prevented CDT from working with the Platform SR1. So, CDT might hurry up with their SR1, or adopters would all have to patch a mix and match of components to make their product schedule

Source: https://wiki.eclipse.org/Planning_Council/May_06_2015#History

Eclipse Release Train

- Many project, complex process
  - Only a single, one-week delay in 8 years
- Frequent release is problematic
- Motto:
  - *"Shipping is hard, that's why we do it 7 times a release."*

Continuous Integration

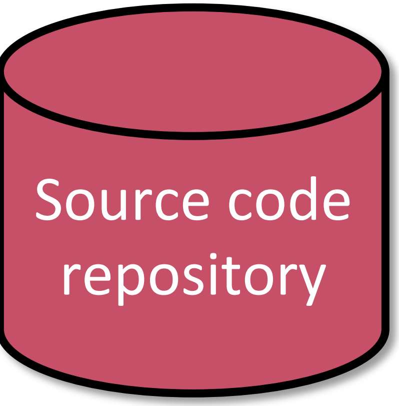
Continuous integration

- *“Continuous Integration is a software development practice where members of a team* ***integrate their work frequently****, usually each person integrates at least daily - leading to multiple integrations per day. Each integration is* ***verified by an automated build*** *(including test) to detect integration errors as quickly as possible.”*

Martin Fowler

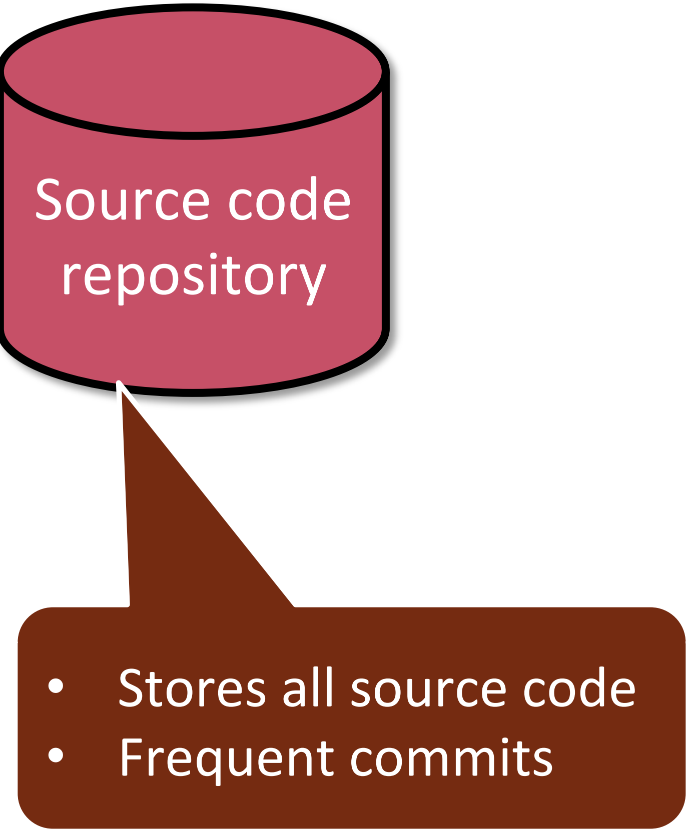
http://www.martinfowler.com/articles/continuousIntegration.html

Tasks in Continuous Integration



Source code repository

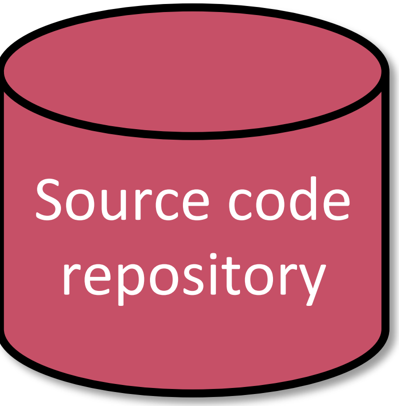
Tasks in Continuous Integration



Source code repository

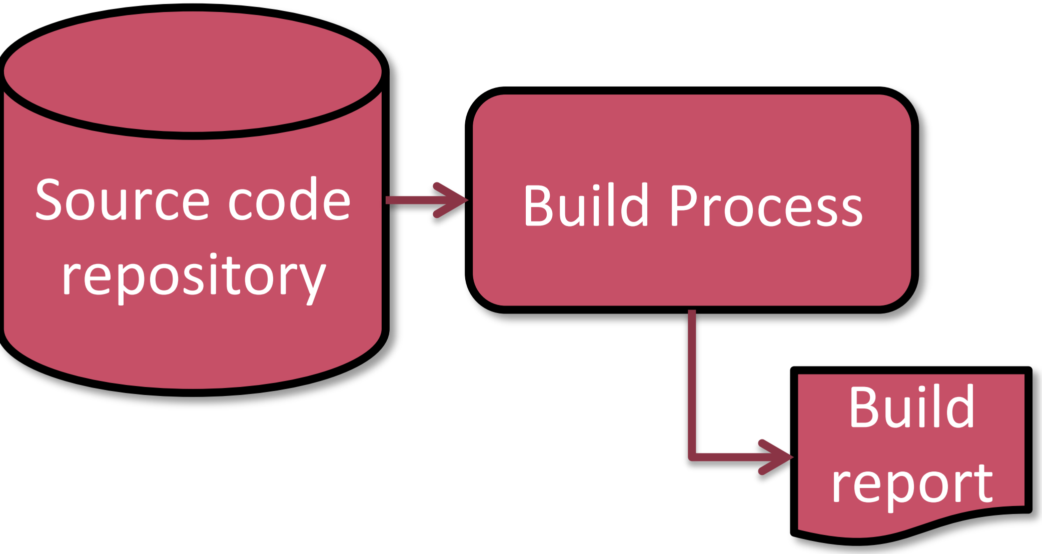
- Stores all source code
- Frequent commits

Tasks in Continuous Integration

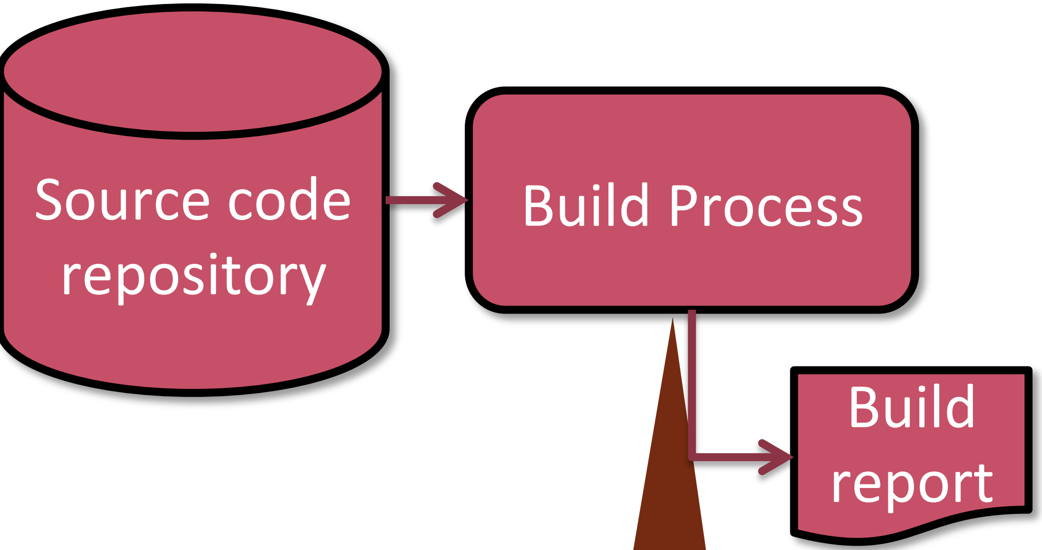


Source code repository

Tasks in Continuous Integration

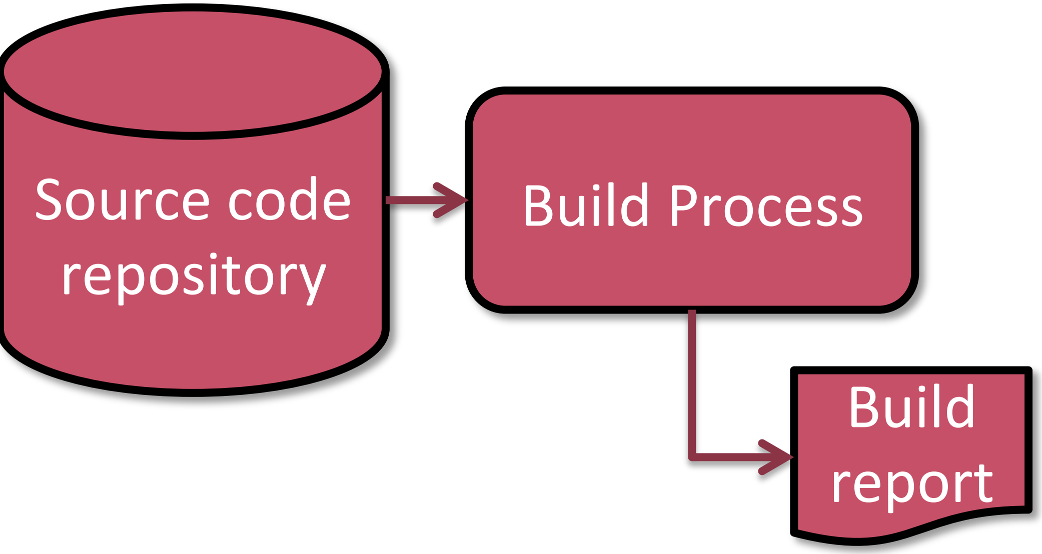


Tasks in Continuous Integration

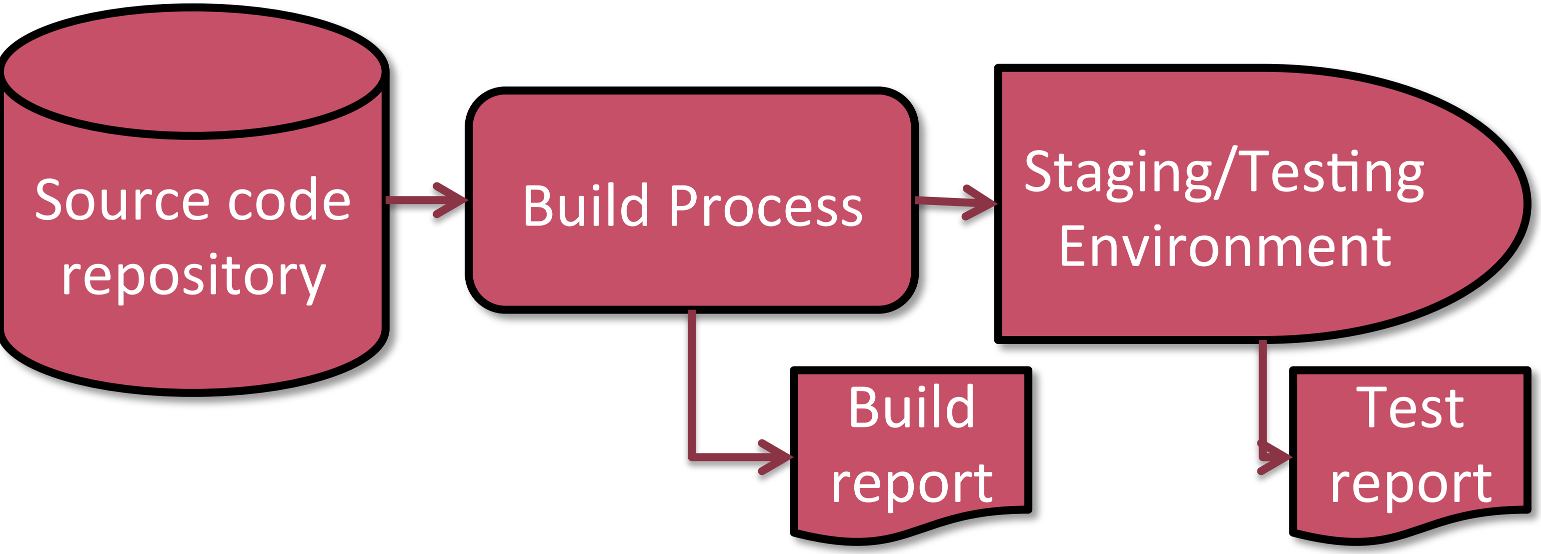


- Fast and automatic!
- Executes unit tests
- Can report failures

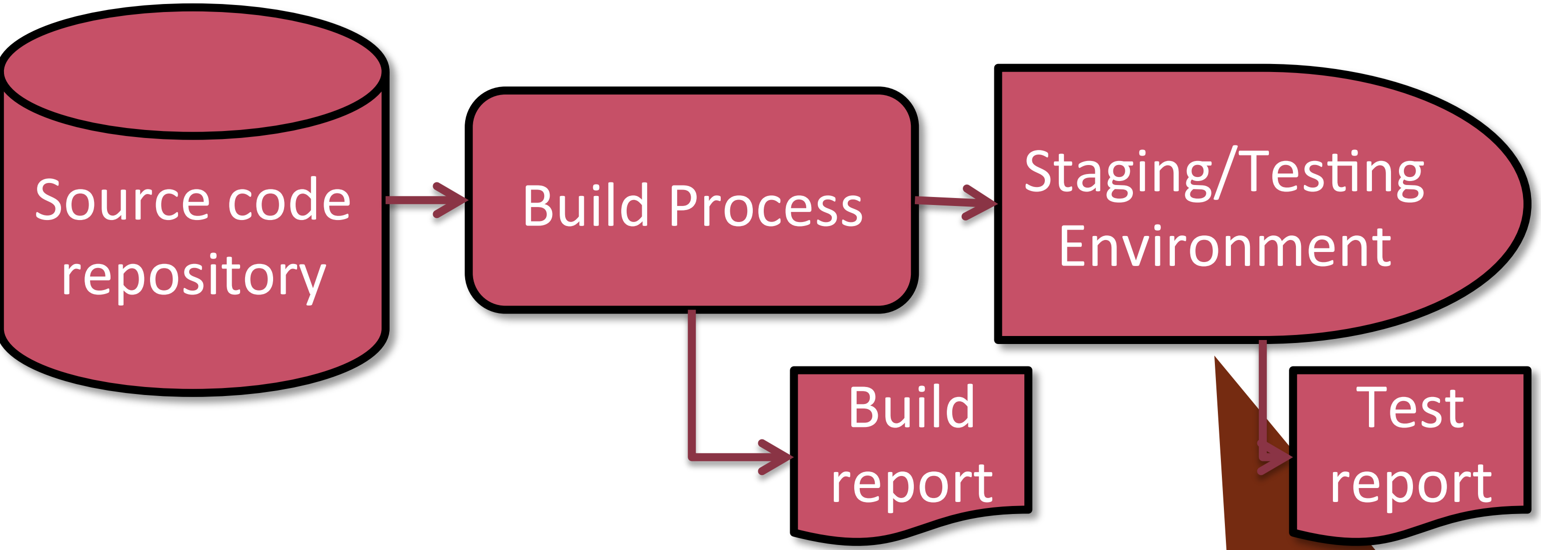
Tasks in Continuous Integration



Tasks in Continuous Integration

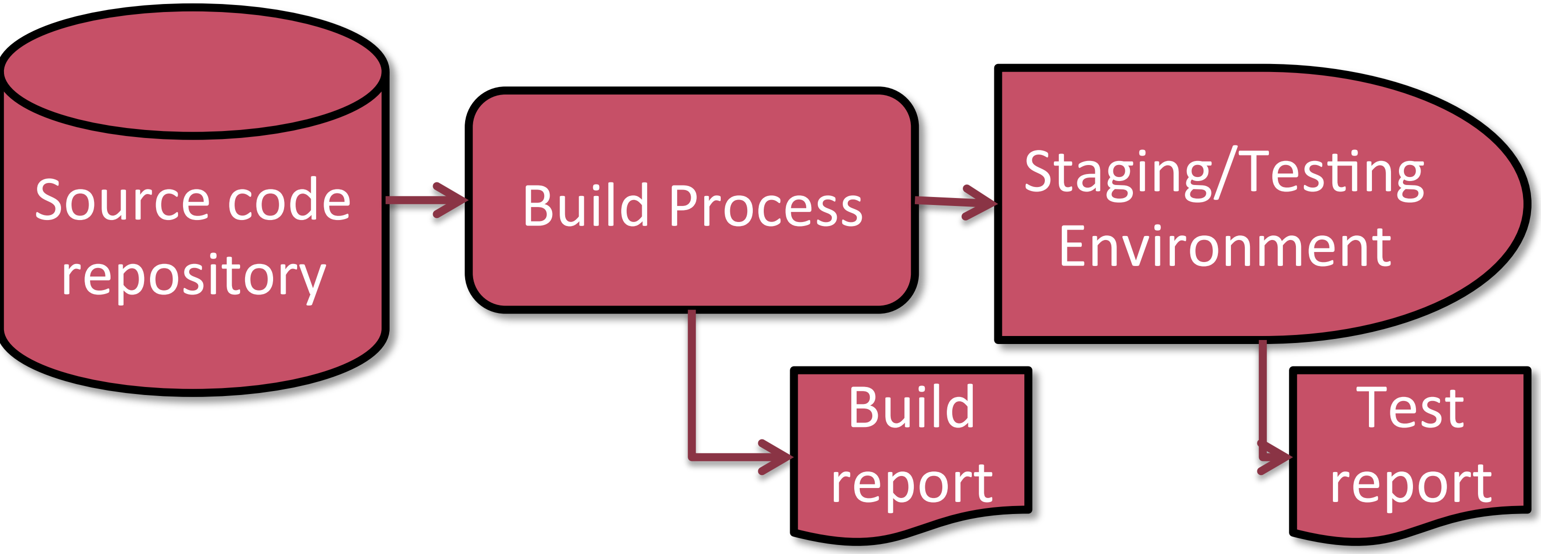


Tasks in Continuous Integration

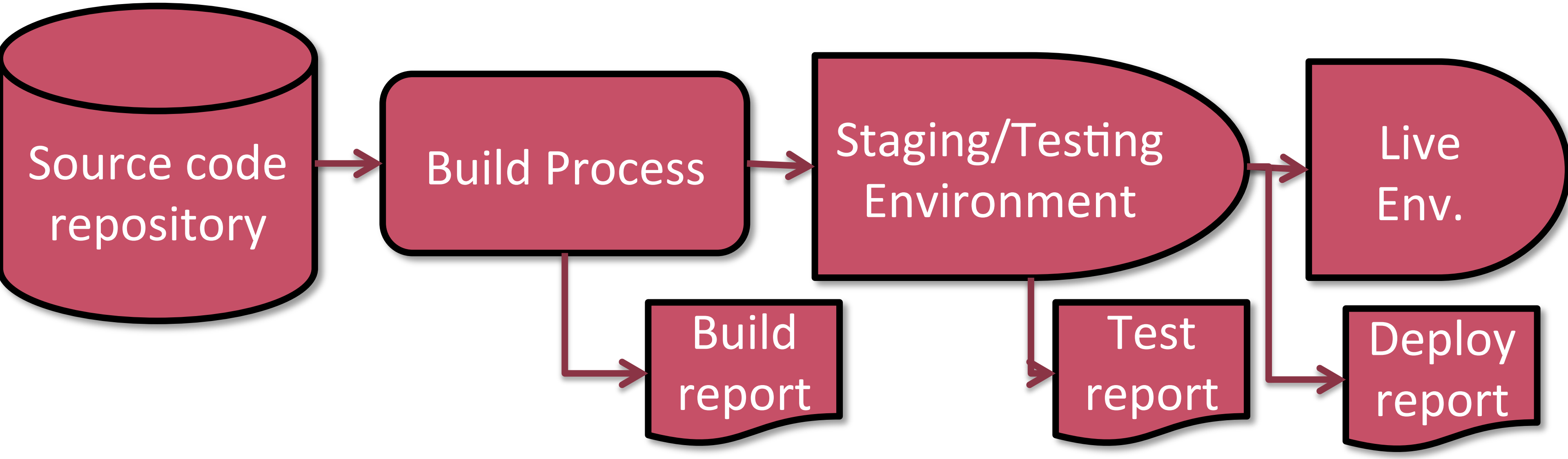


- Copy of the live environment
- Executes integration tests
  - Manual
  - Automatic

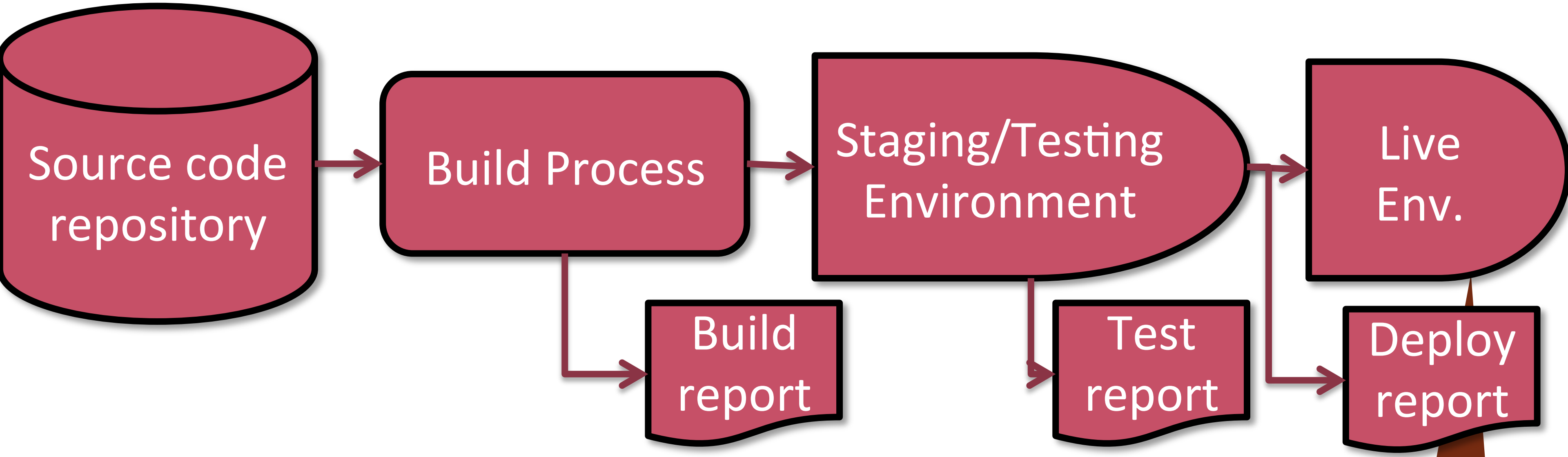
Tasks in Continuous Integration



Tasks in Continuous Integration

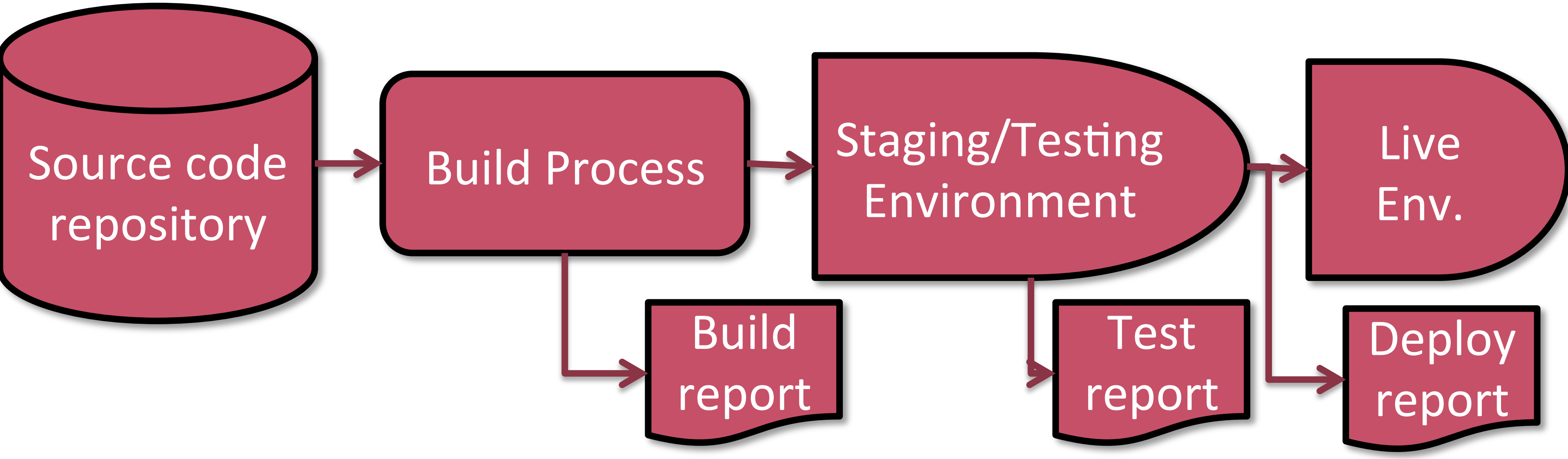


Tasks in Continuous Integration

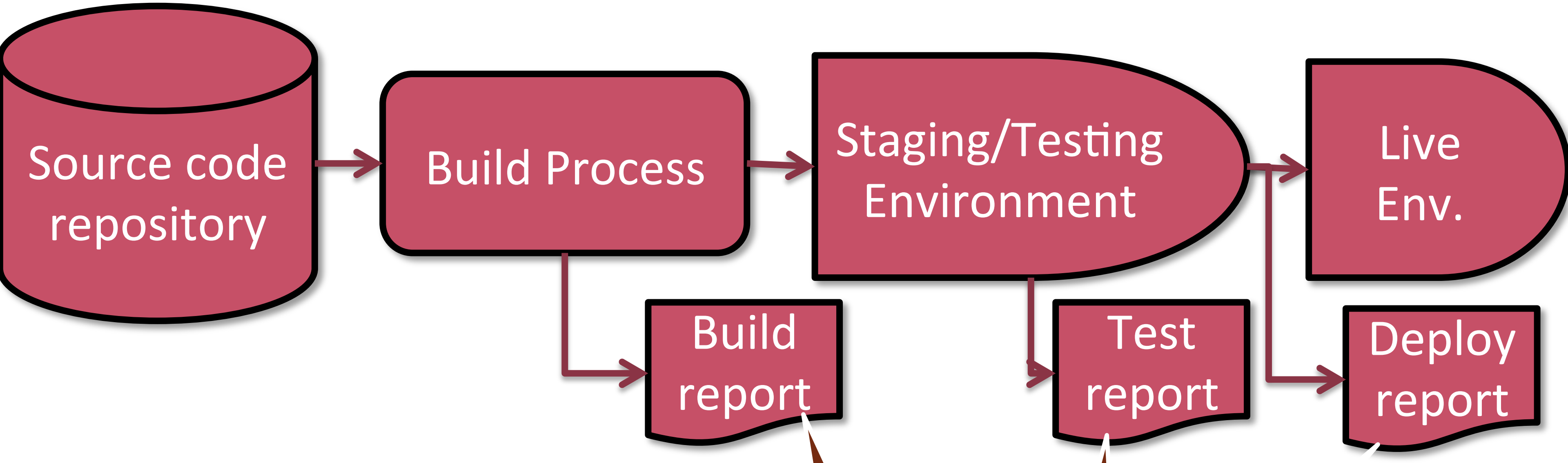


- Possible automatic deployment

Tasks in Continuous Integration

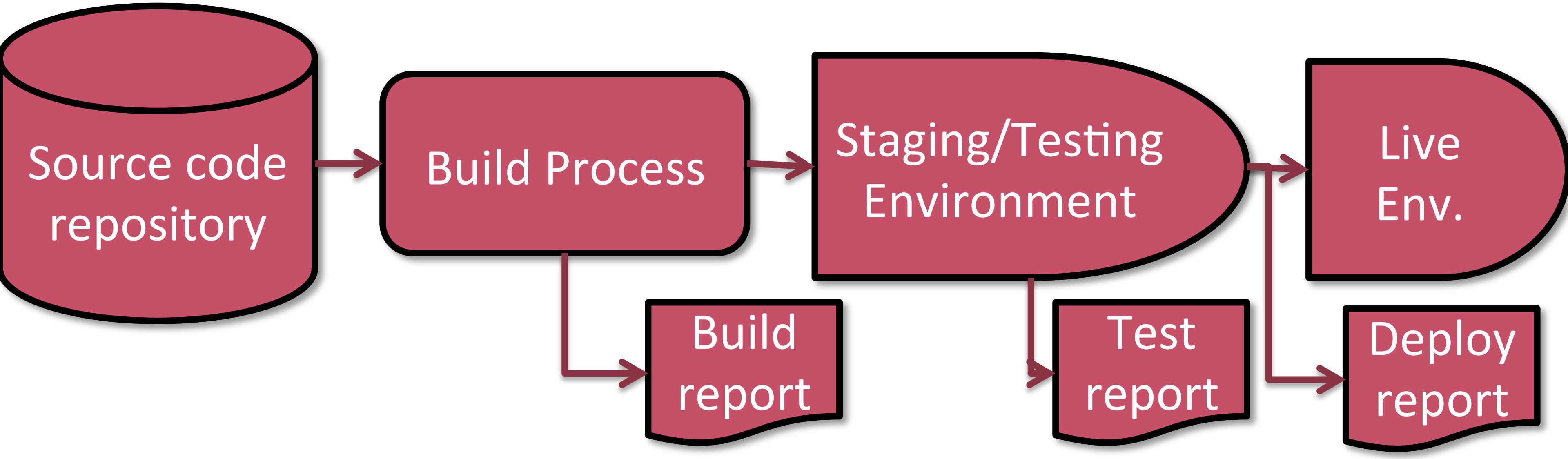


Tasks in Continuous Integration



- Reports and built source code available

Tasks in Continuous Integration



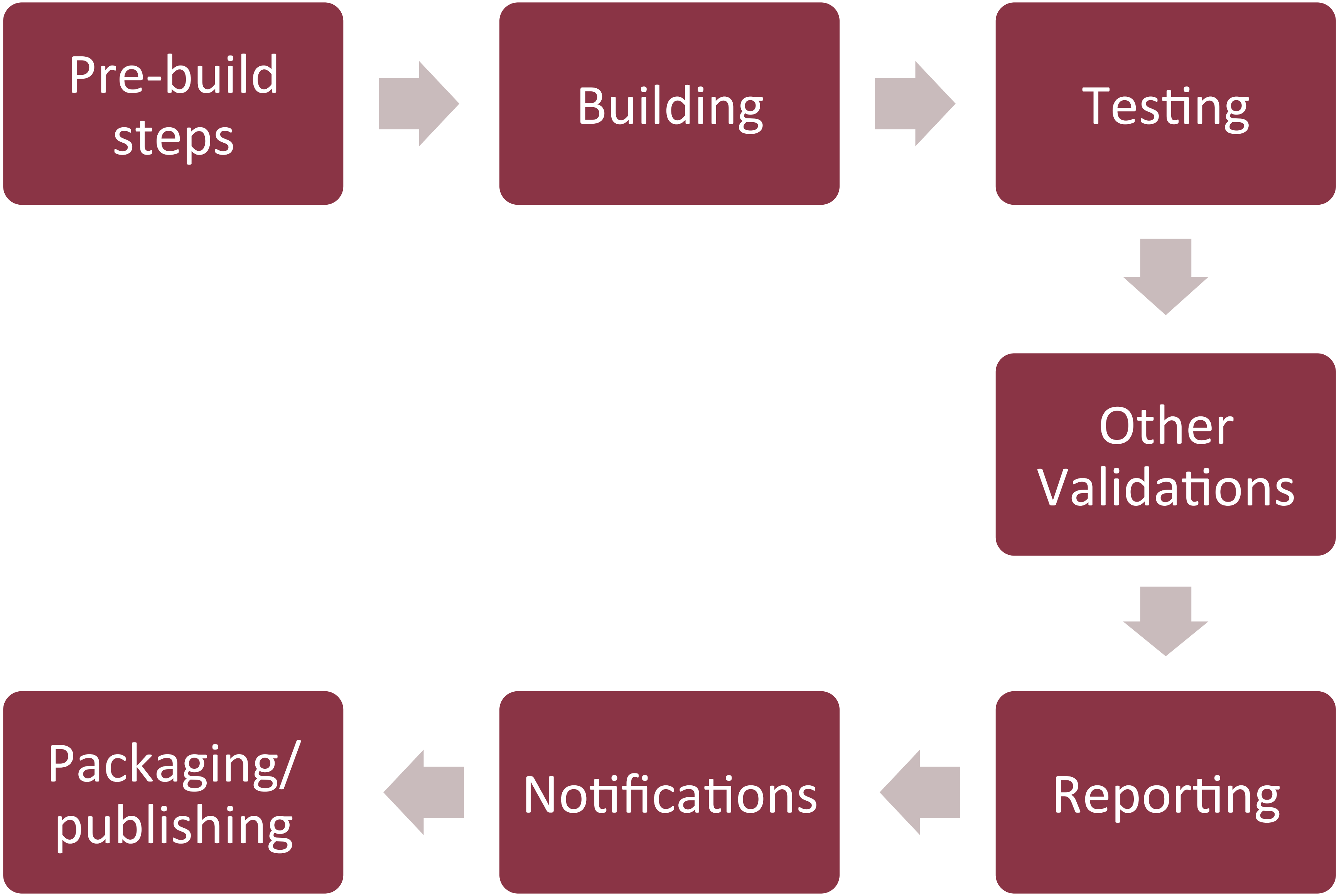
Results

- Builds reproducible
  - Even a year-old build should be repeatable
- Integration phase is short
  - Starts earlier
  - Integration problems become visible soon
- Not a magic bullet
  - Extensive planning required
  - Some changes in development workflow required

Build types

- CI build
  - Executes on every commit
  - Must be fast -> minimal sanity check
- Nightly
  - Executes every night
  - Packaging
  - Should finish in a few hours
- Release build
  - Full testing
  - Might be very long

Most important steps



Pre-build steps

Building

Testing

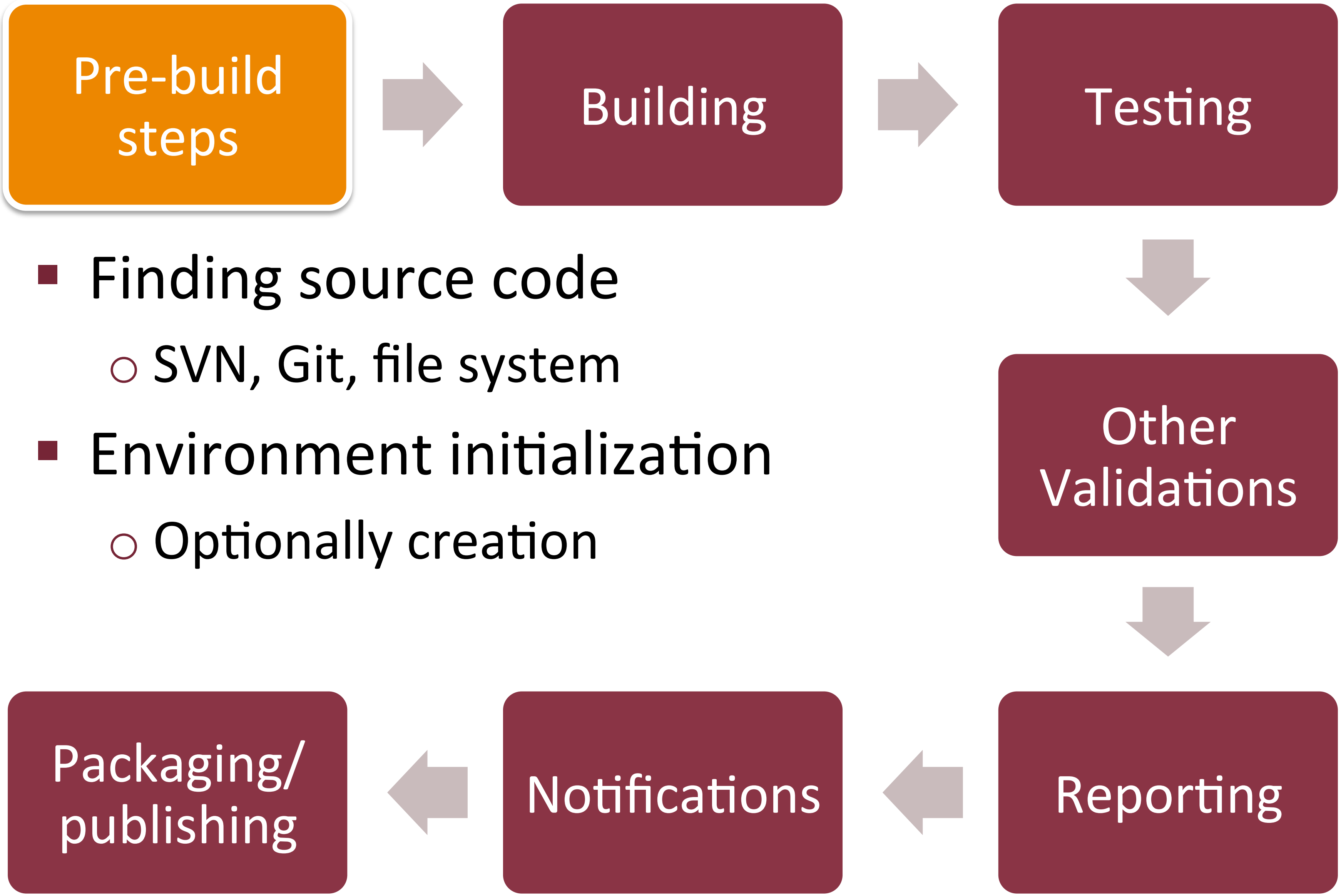
Other Validations

Reporting

Notifications

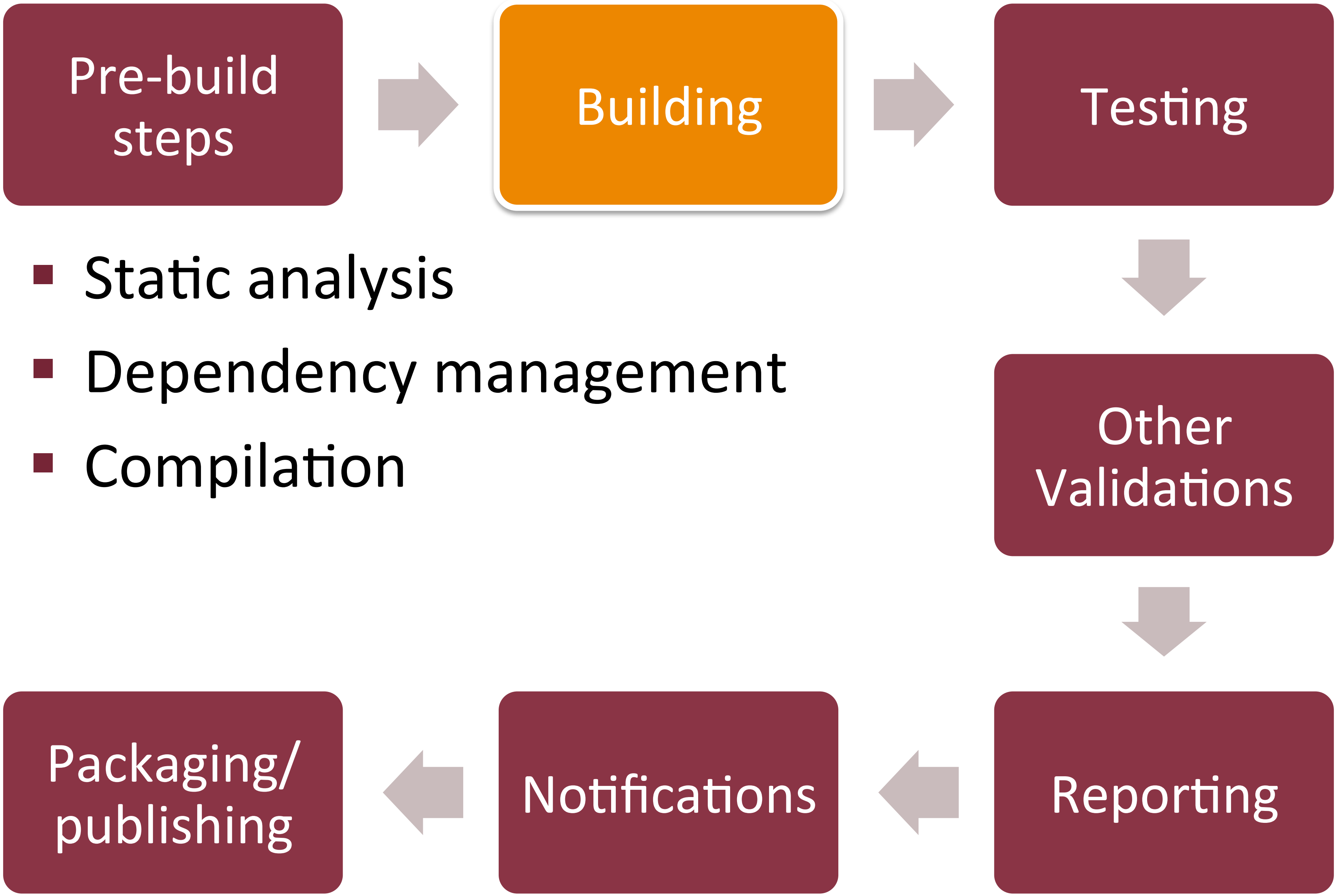
Packaging/ publishing

Most important steps



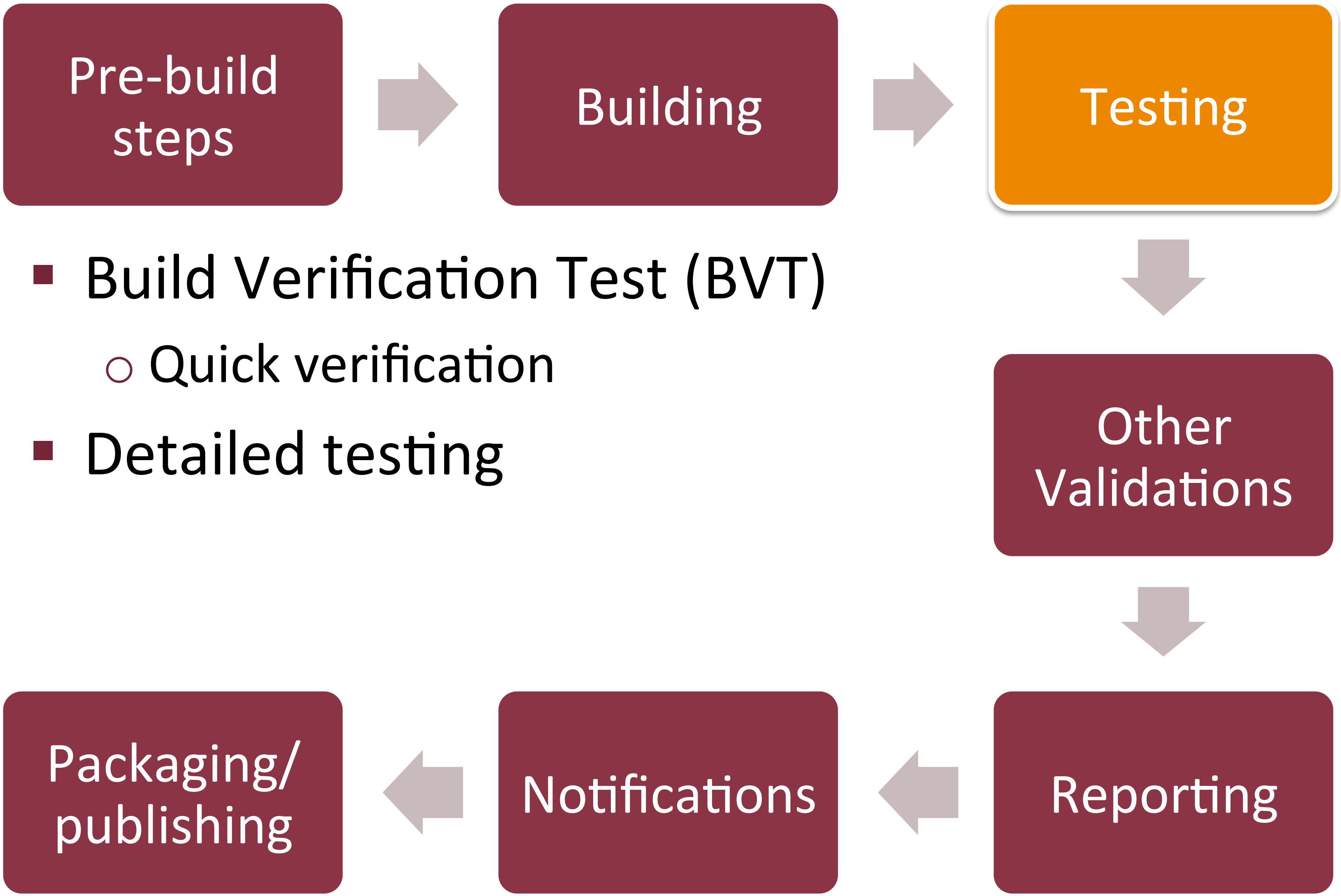
- Finding source code
  - SVN, Git, file system
- Environment initialization
  - Optionally creation

Most important steps



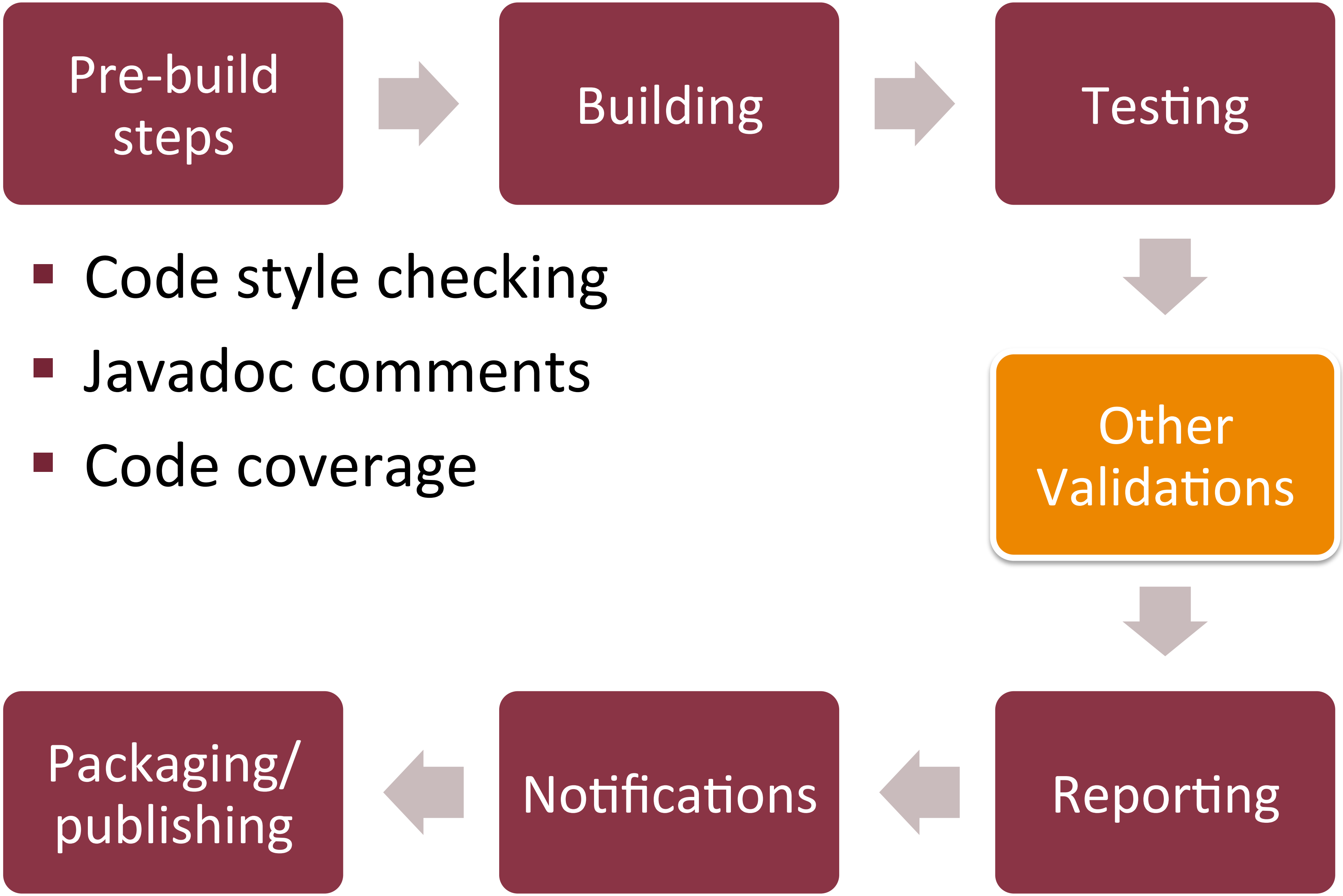
- Static analysis
- Dependency management
- Compilation

Most important steps



- Build Verification Test (BVT)
  - Quick verification
- Detailed testing

Most important steps



Pre-build steps

Building

Testing

Other Validations

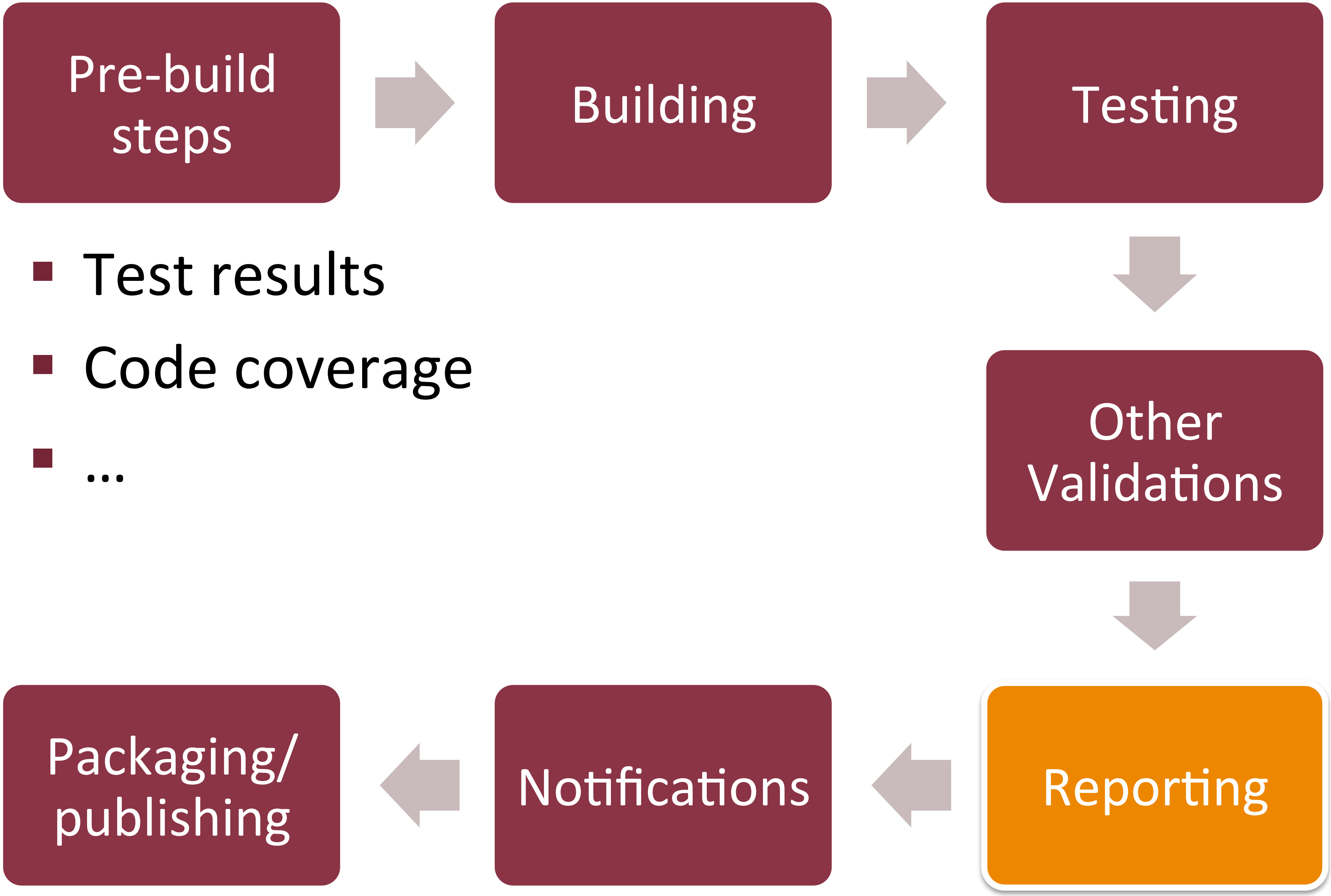
Reporting

Notifications

Packaging/ publishing

- Code style checking
- Javadoc comments
- Code coverage

Most important steps


Pre-build steps

Building

Testing

Other Validations

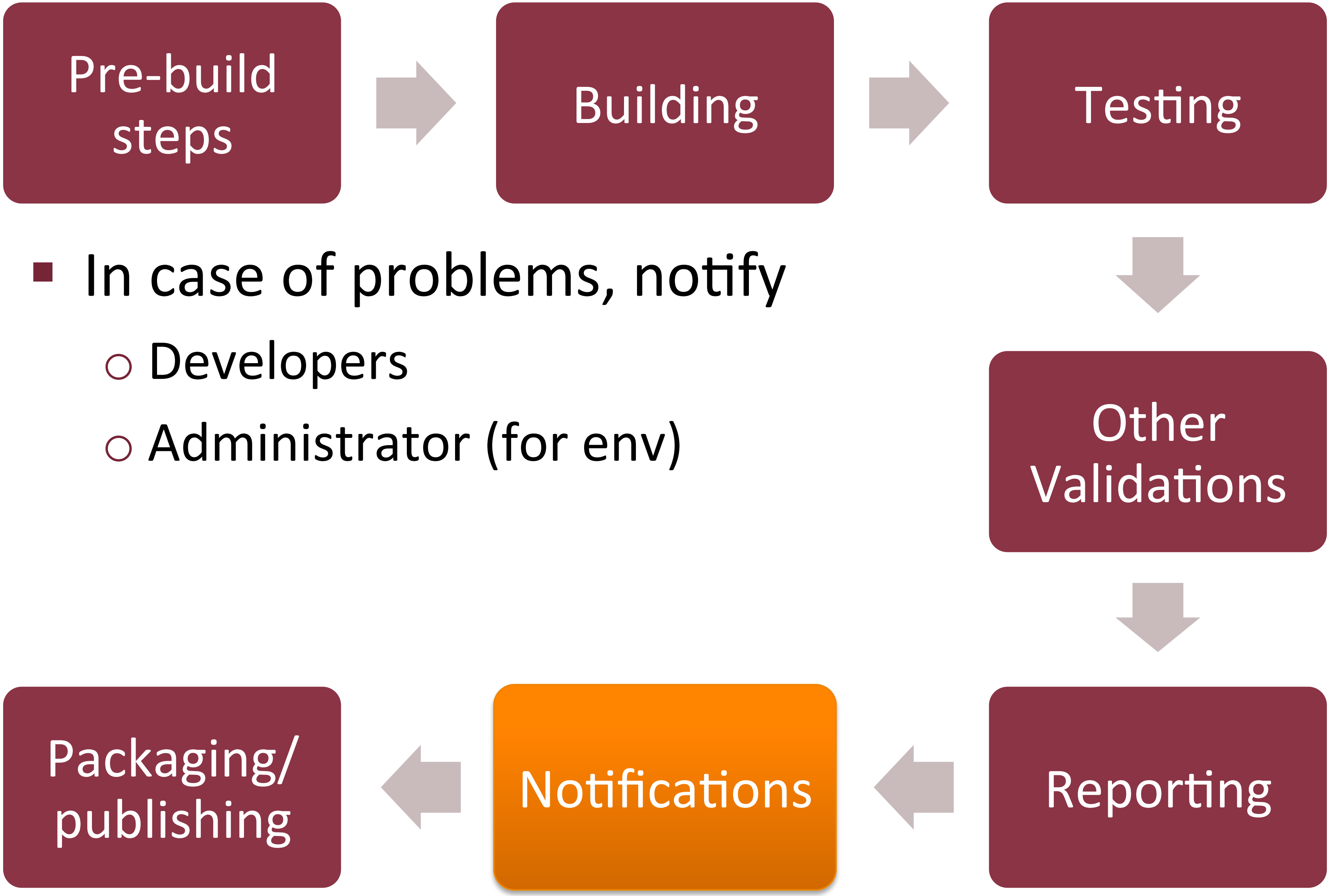
Reporting

Notifications

Packaging/ publishing

- Test results
- Code coverage
- …

Most important steps



Pre-build steps

Building

Testing

Other Validations

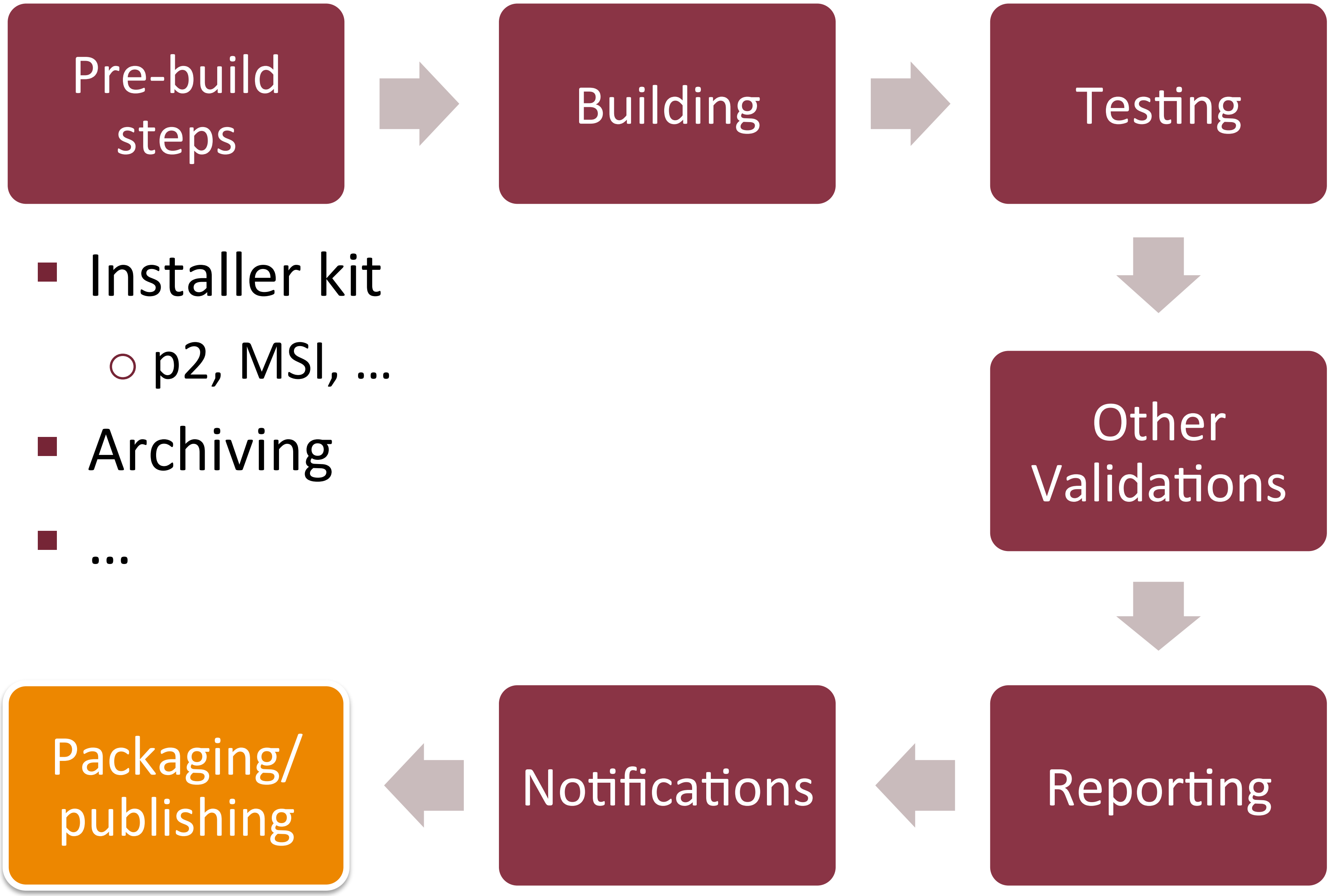
Reporting

Notifications

Packaging/ publishing

- In case of problems, notify
  - Developers
  - Administrator (for env)

Most important steps



Pre-build steps

Building

Testing

Other Validations

Reporting

Notifications

Packaging/ publishing

- Installer kit
  - p2, MSI, …
- Archiving
- …

Build Executor Engines

Build Tools

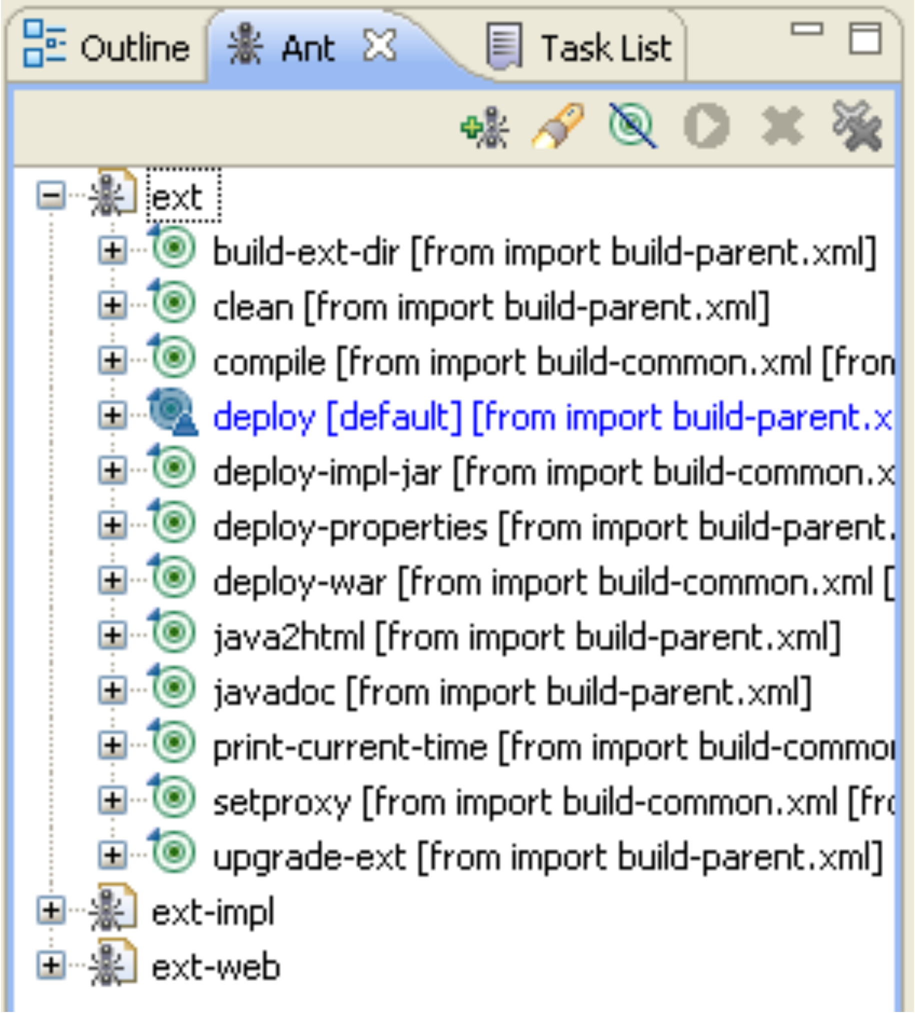
- Make
  - C/C++
- Apache Ant
  - Make files for Java
  - XML dialect
- Apache Maven
  - Uniform source and dependency management
  - Declarative build descriptors
    - Functionally similar to Ant
- …

Ant

- Java library and command line tool
- Versatily, extensible
- Main application: Java application compilation

Ant Basics

- Project
  - Represented by a single build descriptor
- Target
  - A set of executable tasks
  - **May depend on other targets**
  - E.g., compile, deploy
- Task
  - Executable code
  - E.g., javac, copy, junit, exec, signjar, mail…



```
ext
  build-ext-dir [from import build-parent.xml]
  clean [from import build-parent.xml]
  compile [from import build-common.xml [fron
  deploy [default] [from import build-parent.x
  deploy-impl-jar [from import build-common.x
  deploy-properties [from import build-parent.
  deploy-war [from import build-common.xml [
  java2html [from import build-parent.xml]
  javadoc [from import build-parent.xml]
  print-current-time [from import build-commo
  setproxy [from import build-common.xml [fro
  upgrade-ext [from import build-parent.xml]
ext-impl
ext-web
```

Additional Options

- Properties (key-value pairs)

```
<property name="build" location="build"/>

<target name="init">
    <mkdir dir="${build}"/>
</target>
```

- Paths, classpath

```
<classpath>
   <pathelement path="${classpath}"/>
   <pathelement location="lib/helper.jar"/>
</classpath>
```

- Every element can have an optional ID
  - Everything can be referenced

Example: Testing with Ant

- Required:
  - junit.jar
  - ant-junit.jar
    - Default location: `ANT_HOME/lib`
- junit.jar location:
  - In `ANT_HOME/lib` directory, or
  - Set via `–lib` argumentum, or
  - Set via the `classpath` element of the `junit` task

Example: Testing with Ant

```
<project default="test" >
    <path id="classpath.test">
        <pathelement location="x/y/junit.jar" />
        <pathelement location="${build}" />
    </path>
...
    <target name="compile-test">
        <javac srcdir    tst-dir}" >
            <classpath refid="classpath.test"/>
        </javac>
    </target>
...
```

Example: Testing with Ant

```
...
<target name="test" depends="compile-test" >
    <junit printsummary="yes"
  haltonfailure="yes">
        <classpath refid="classpath.test" />
        <formatter type="plain" />
        <test name="hu.bme.mit.junit.
          bookstore.book.test.BMListTest"
                   haltonfailure="no"
  outfile="result" >
            <formattertype="xml"/>
        </test>
    </junit>
</target>
```

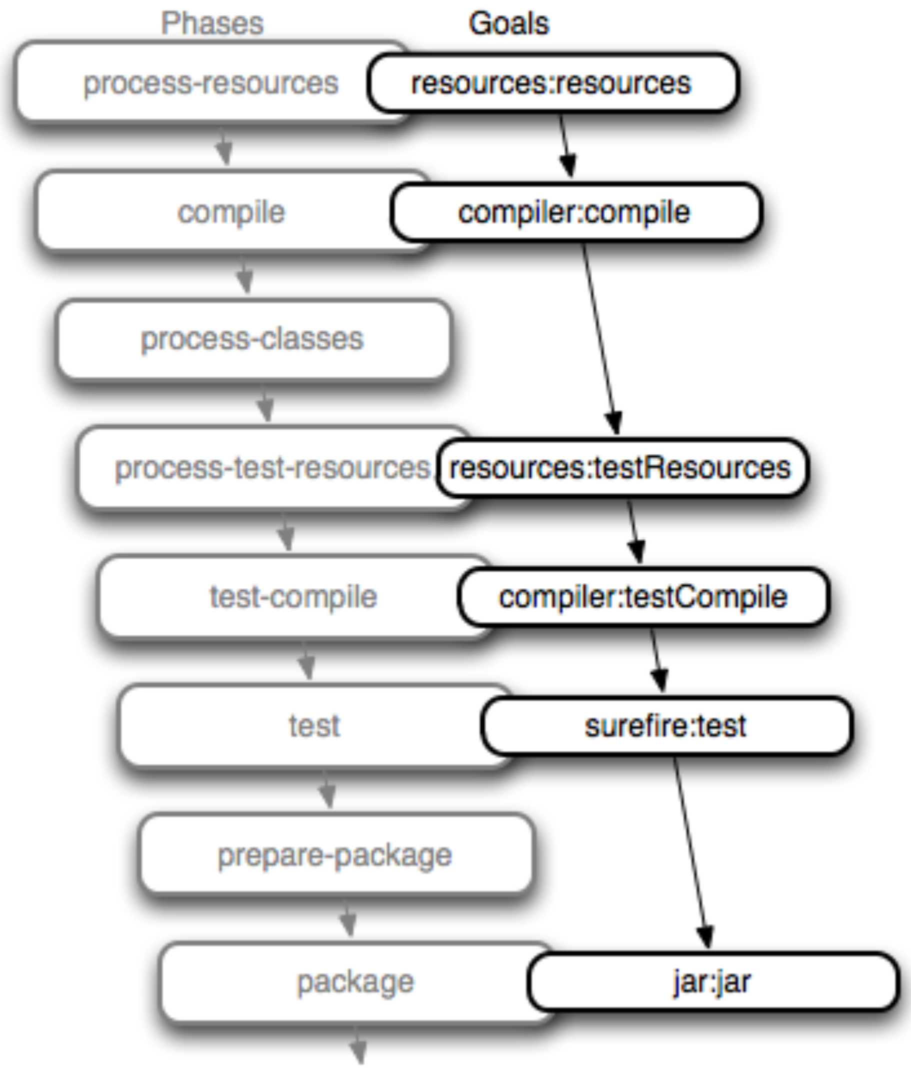
Maven

- More complex build tool
- Build process predefined
  - Usually less configuration required
    - Convention over configuration
    - BUT: If conventions need to be
      - Understood
      - Followed (or the differences described)
- Dependency management!

Maven

- Descriptor
  - pom.xml: project model
  - Archetype: a description of a project type
    - It is enough to list the differences wrt an archetype
    - Default archetype is Java project
- Build process
  - Name a goal (e.g., test, package)
  - Manages all phases until that point

Mavaen Lifecycle Phases and Goals



Source: Maven by Example, http://books.sonatype.com/mvnex-book/reference/simple-project-sect-simple-core.html

Note: There are more phases than shown above, this is a partial list

Example: Testing with Maven

- Project structure:
- my-app
  - pom.xml
  - src
    - main
      - java
        - com |
          - mycompany
            - app |
              - App.java
  - test
    - java
      - com
        - mycompany
          - app
            - AppTest.java

Example: Testing with Maven – pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://
maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.mycompany.app</groupId>
  <artifactId>my-app</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>Maven Quick Start Archetype</name>
  <url>http://maven.apache.org</url>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.8.0</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
</project>
```

Ant vs Maven

- Real “crusade”
  - See also .Net or Java, etc.
- Ant
  - Everything can be (is) hand-managed
  - Useful for unique projects
- Maven
  - “Convention over configuration”
  - Every Maven plug-in is similar…
  - Dependency management
    - “Maven downloads the entire Internet”

Build Scheduling

Jenkins (a.k.a. Hudson)

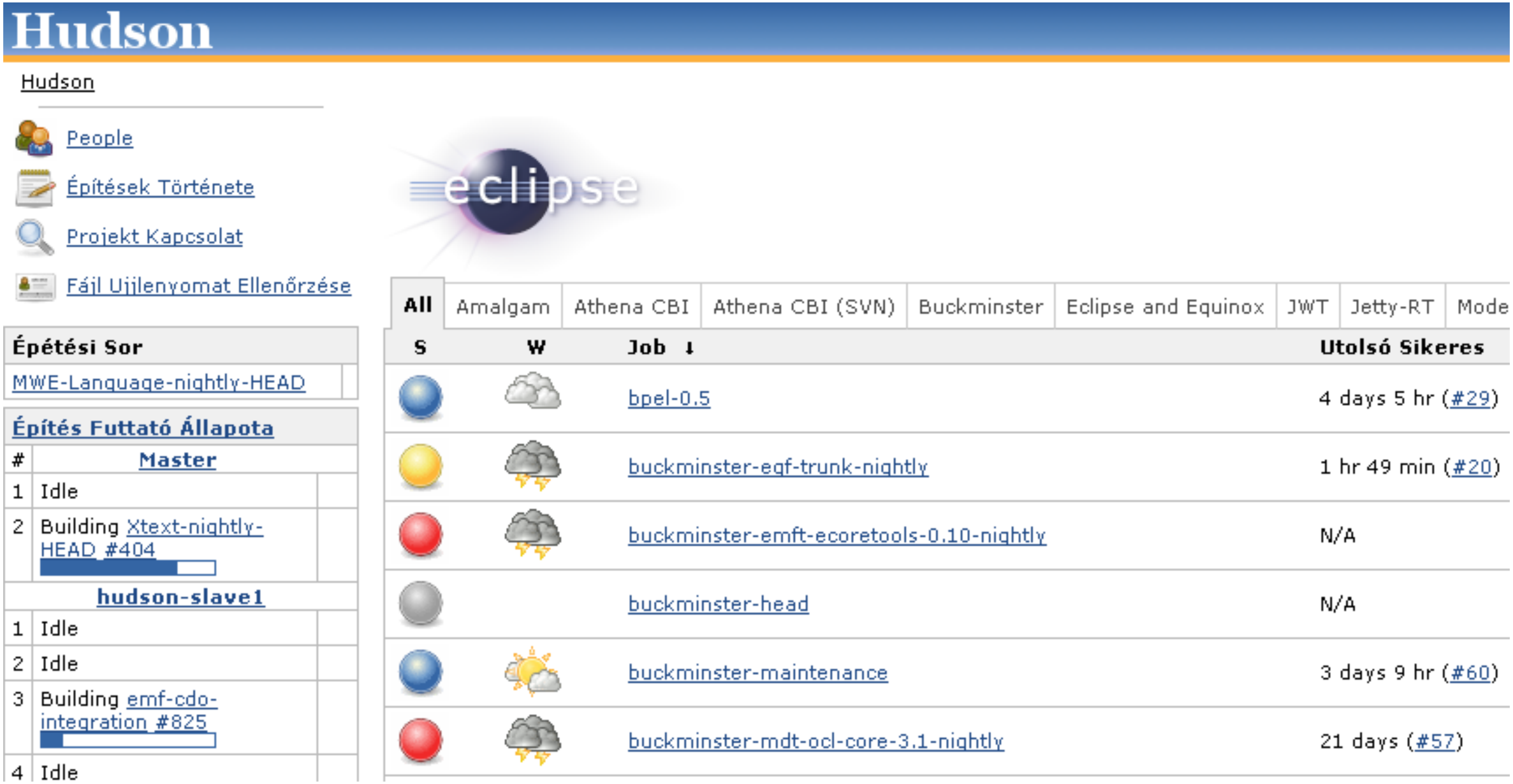
CI Servers

- Apache Continuum (Java)
  - XML-based configuration + web UI
- CruiseControl (Java, .NET, Ruby)
  - XML-based configuration
- Jenkins/Hudson (Java, extensible)
  - Web UI
- TeamCity (Java, .NET, Ruby)
  - Commercial
- …

Hudson/Jenkins

- Java servlet based
  - Every application server is useable
- Plug-in based, **extensible**
  - Plug-ins can be updated
- Easy to learn
- Does not determine build tool, only
  - Scheduling and
  - Management
- Multiple build jobs with **dependencies** between them

https://hudson.eclipse.org/hudson/



Hudson

Hudson

- People
- Építések Története
- Projekt Kapcsolat
- Fájl Ujjlenyomat Ellenőrzése

Épétési Sor

MWE-Language-nightly-HEAD

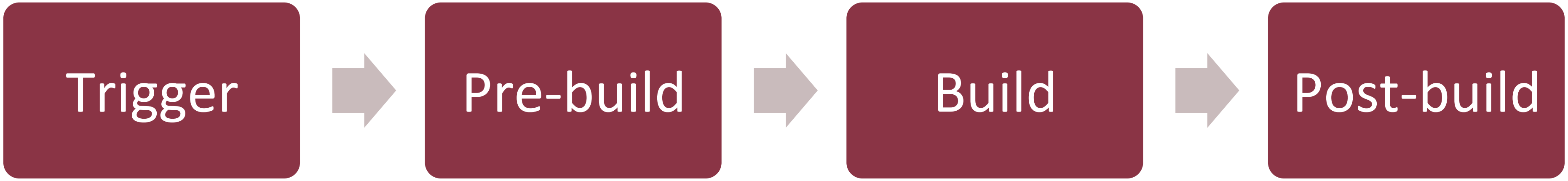
Építés Futtató Állapota

#	Master
1	Idle
2	Building Xtext-nightly-HEAD #404
	hudson-slave1
1	Idle
2	Idle
3	Building emf-cdo-integration #825
4	Idle

All | Amalgam | Athena CBI | Athena CBI (SVN) | Buckminster | Eclipse and Equinox | JWT | Jetty-RT | Mode

S	W	Job ↓	Utolsó Sikeres
●		bpel-0.5	4 days 5 hr (#29)
●		buckminster-eqf-trunk-nightly	1 hr 49 min (#20)
●		buckminster-emft-ecoretools-0.10-nightly	N/A
●		buckminster-head	N/A
●		buckminster-maintenance	3 days 9 hr (#60)
●		buckminster-mdt-ocl-core-3.1-nightly	21 days (#57)

Hudson Workflow



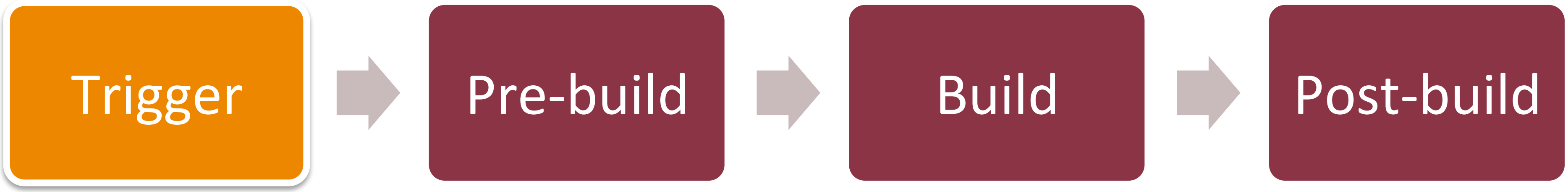
Trigger

Pre-build

Build

Post-build

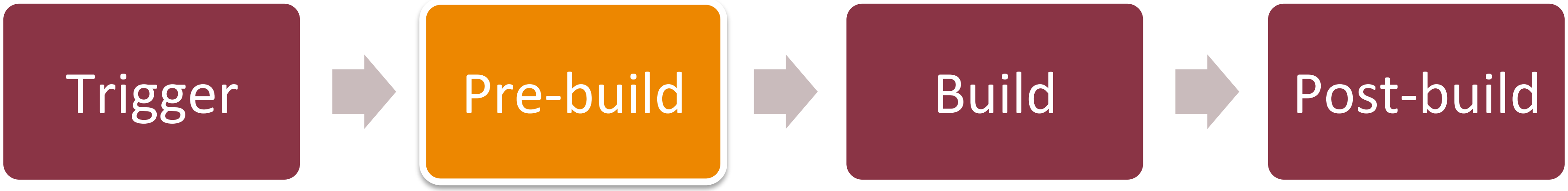
Hudson Workflow



Trigger → Pre-build → Build → Post-build

- Manual
- Timed
- Change in version control
- Dependencies built in another job
- Custom (extensible)

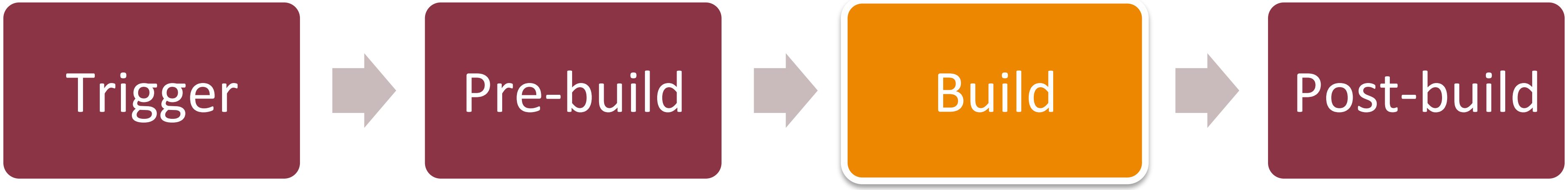
Hudson Workflow



Trigger → Pre-build → Build → Post-build

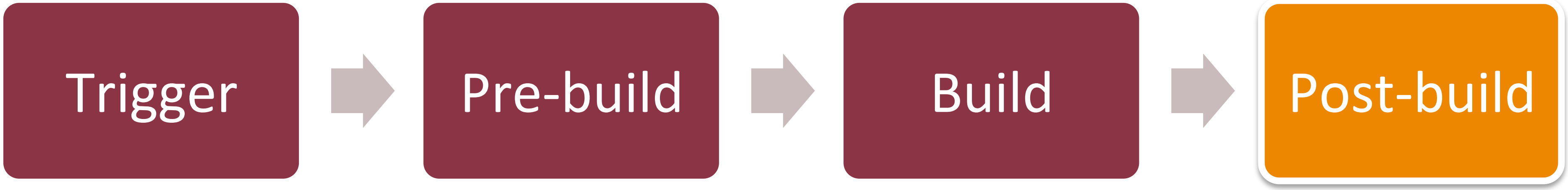
- Optional
- Collect sources
- Set up environment

Hudson Workflow



- Build steps
- Build tools supported by default
  - Ant
  - Maven
  - Command line
- Additional tools
  - Buckminster
  - .Net compiler
  - …

Hudson Workflow


Trigger

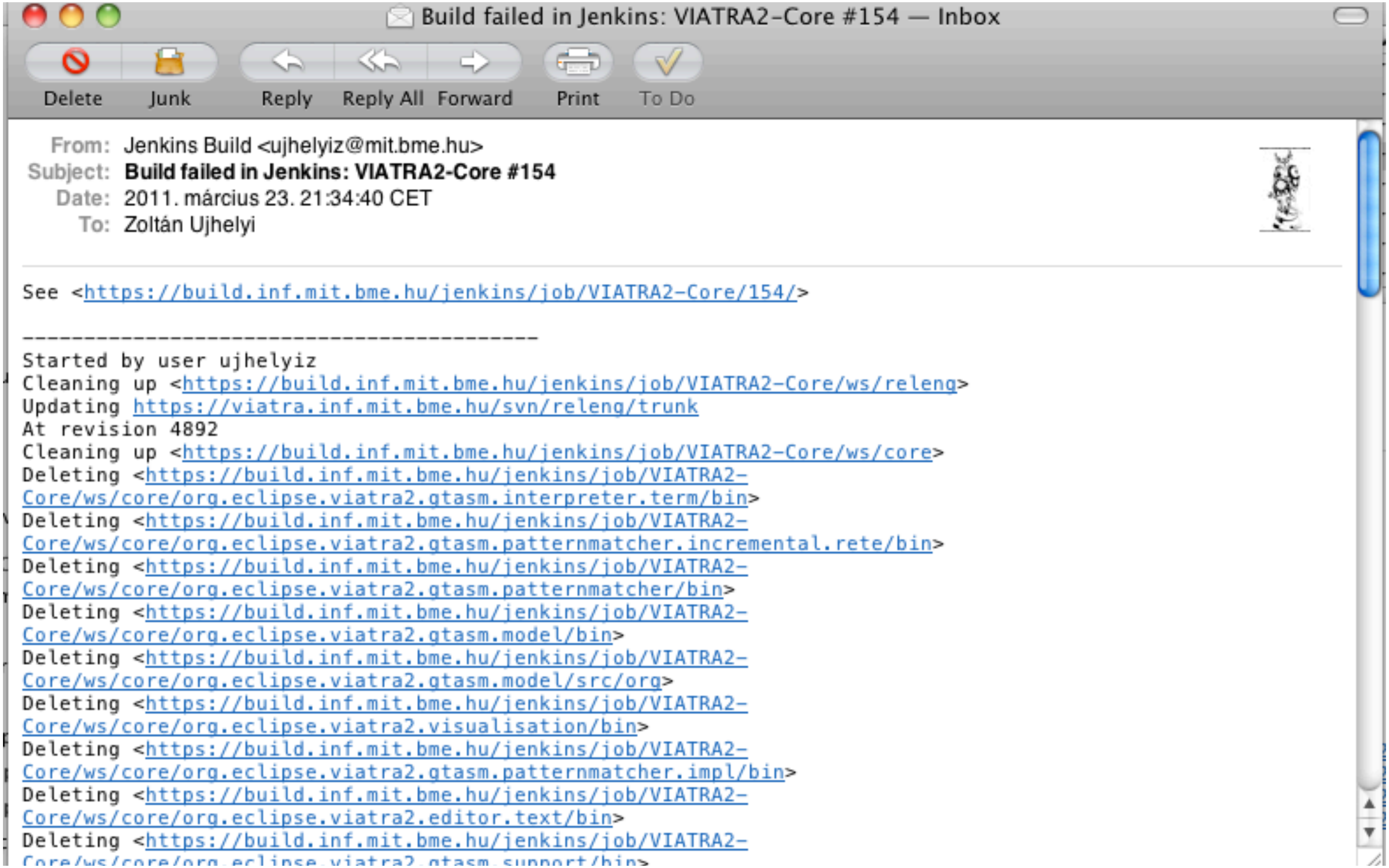
Pre-build

Build

Post-build

- Optional step
  - Archiving
  - Publishing
  - Start follow-up builds
  - Notifications
  - …

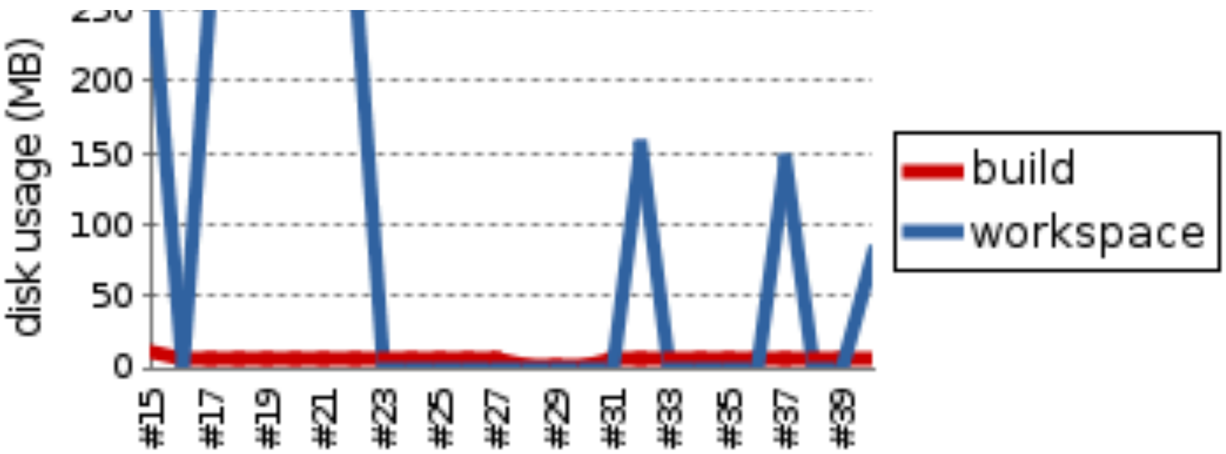
Blame mail

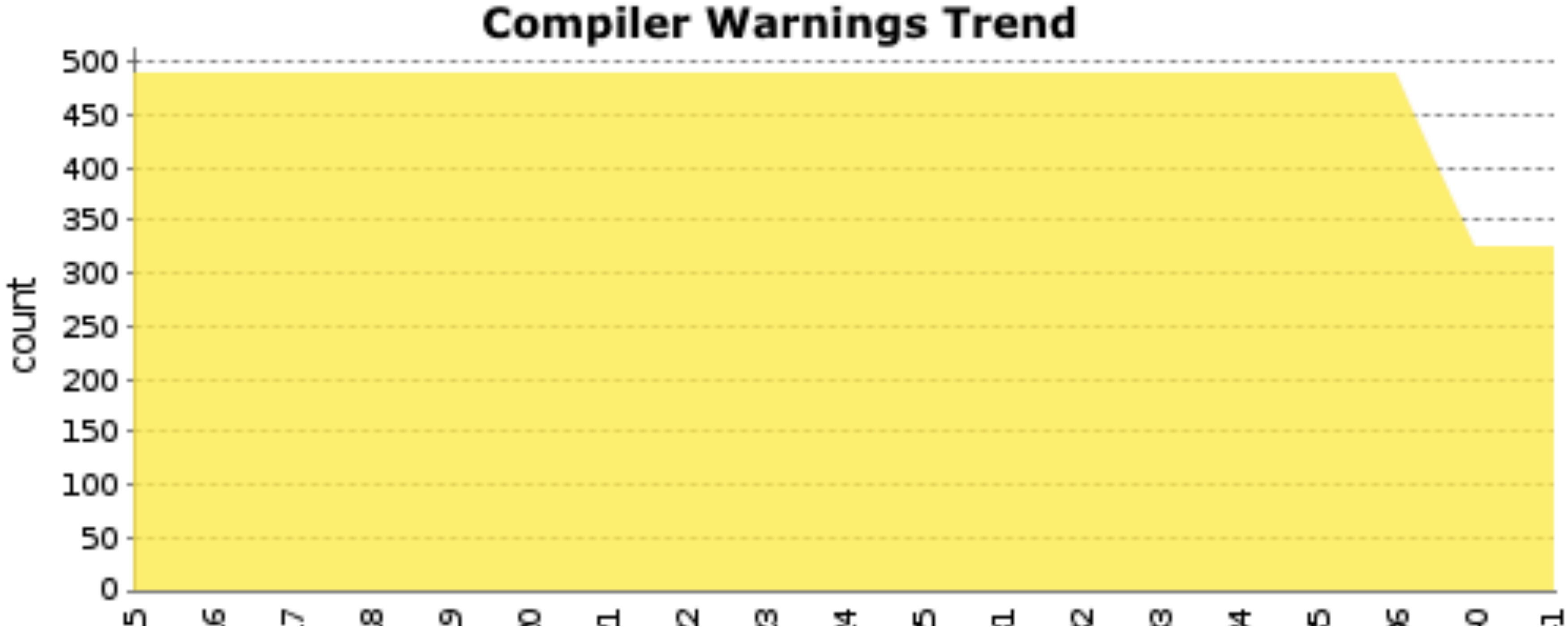


From: Jenkins Build <ujhelyiz@mit.bme.hu>
Subject: **Build failed in Jenkins: VIATRA2-Core #154**
Date: 2011. március 23. 21:34:40 CET
To: Zoltán Ujhelyi

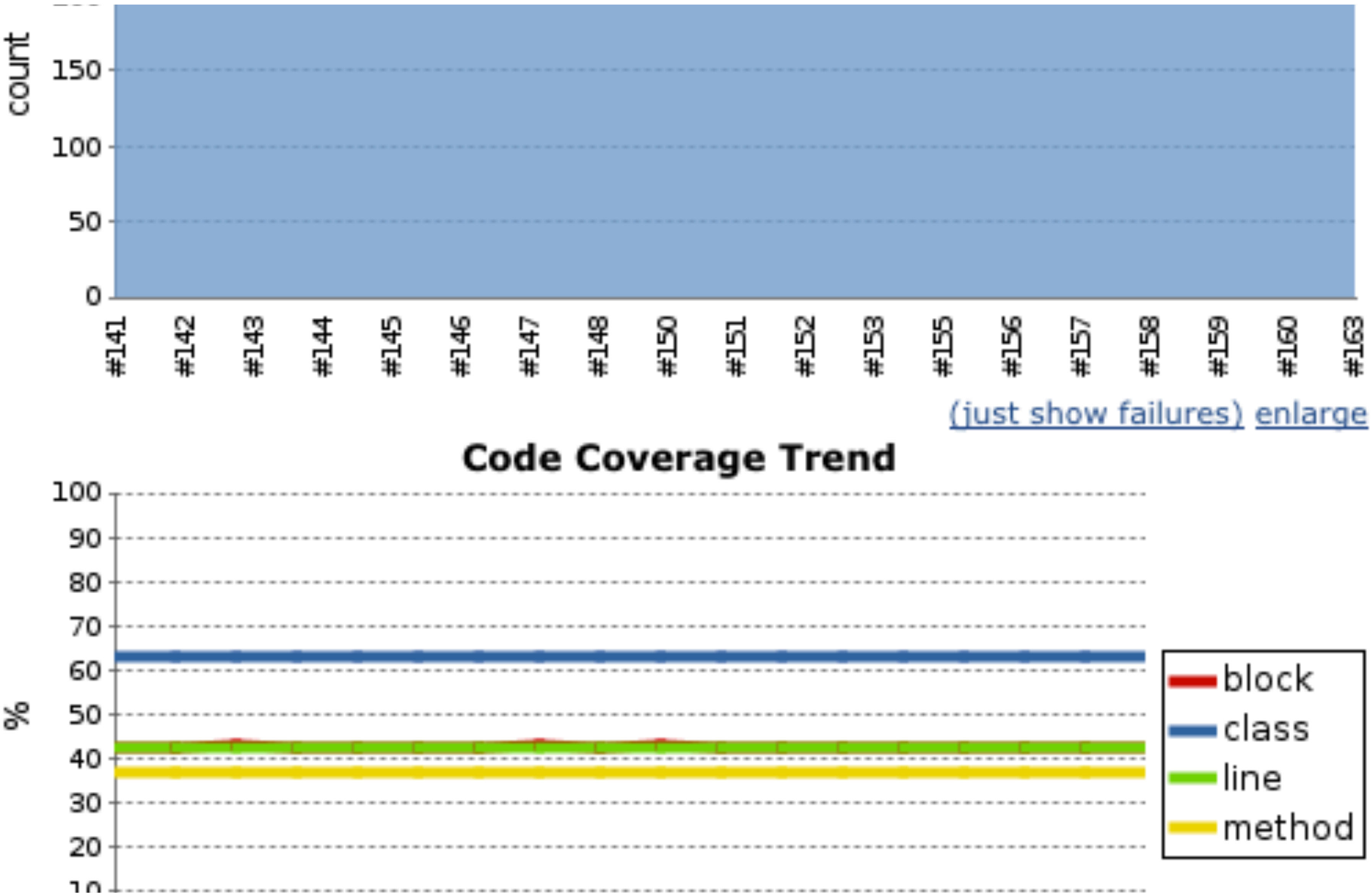
```
See <https://build.inf.mit.bme.hu/jenkins/job/VIATRA2-Core/154/>

------------------------------------------
Started by user ujhelyiz
Cleaning up <https://build.inf.mit.bme.hu/jenkins/job/VIATRA2-Core/ws/releng>
Updating https://viatra.inf.mit.bme.hu/svn/releng/trunk
At revision 4892
Cleaning up <https://build.inf.mit.bme.hu/jenkins/job/VIATRA2-Core/ws/core>
Deleting <https://build.inf.mit.bme.hu/jenkins/job/VIATRA2-
Core/ws/core/org.eclipse.viatra2.gtasm.interpreter.term/bin>
Deleting <https://build.inf.mit.bme.hu/jenkins/job/VIATRA2-
Core/ws/core/org.eclipse.viatra2.gtasm.patternmatcher.incremental.rete/bin>
Deleting <https://build.inf.mit.bme.hu/jenkins/job/VIATRA2-
Core/ws/core/org.eclipse.viatra2.gtasm.patternmatcher/bin>
Deleting <https://build.inf.mit.bme.hu/jenkins/job/VIATRA2-
Core/ws/core/org.eclipse.viatra2.gtasm.model/bin>
Deleting <https://build.inf.mit.bme.hu/jenkins/job/VIATRA2-
Core/ws/core/org.eclipse.viatra2.gtasm.model/src/org>
Deleting <https://build.inf.mit.bme.hu/jenkins/job/VIATRA2-
Core/ws/core/org.eclipse.viatra2.visualisation/bin>
Deleting <https://build.inf.mit.bme.hu/jenkins/job/VIATRA2-
Core/ws/core/org.eclipse.viatra2.gtasm.patternmatcher.impl/bin>
Deleting <https://build.inf.mit.bme.hu/jenkins/job/VIATRA2-
Core/ws/core/org.eclipse.viatra2.editor.text/bin>
Deleting <https://build.inf.mit.bme.hu/jenkins/job/VIATRA2-
Core/ws/core/org.eclipse.viatra2.gtasm.support/bin>
```

Trends, Metrics

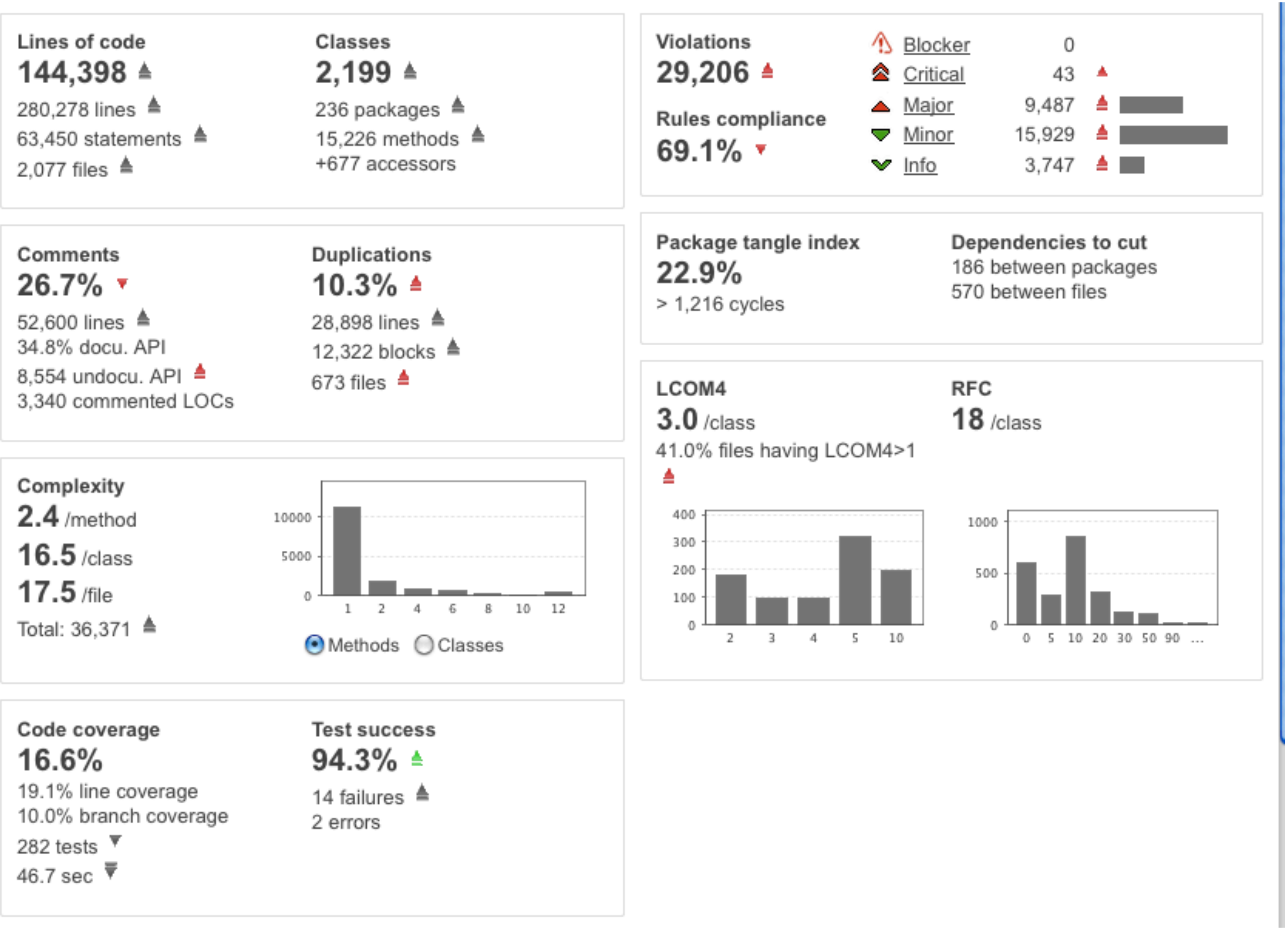




Coverage trends



Other Metrics (using Sonar)



Lines of code
144,398 ▲
280,278 lines ▲
63,450 statements ▲
2,077 files ▲

Classes
2,199 ▲
236 packages ▲
15,226 methods ▲
+677 accessors

Violations
29,206 ▲

Rules compliance
69.1% ▼

Blocker	0		
Critical	43	▲	
Major	9,487	▲	▬
Minor	15,929	▲	▬
Info	3,747	▲	▬

Comments
26.7% ▼
52,600 lines ▲
34.8% docu. API
8,554 undocu. API ▲
3,340 commented LOCs

Duplications
10.3% ▲
28,898 lines ▲
12,322 blocks ▲
673 files ▲

Package tangle index
22.9%
> 1,216 cycles

Dependencies to cut
186 between packages
570 between files

Complexity
2.4 /method
16.5 /class
17.5 /file
Total: 36,371 ▲

● Methods ○ Classes

LCOM4
3.0 /class
41.0% files having LCOM4>1
▲

RFC
18 /class

Code coverage
16.6%
19.1% line coverage
10.0% branch coverage
282 tests ▼
46.7 sec ▼

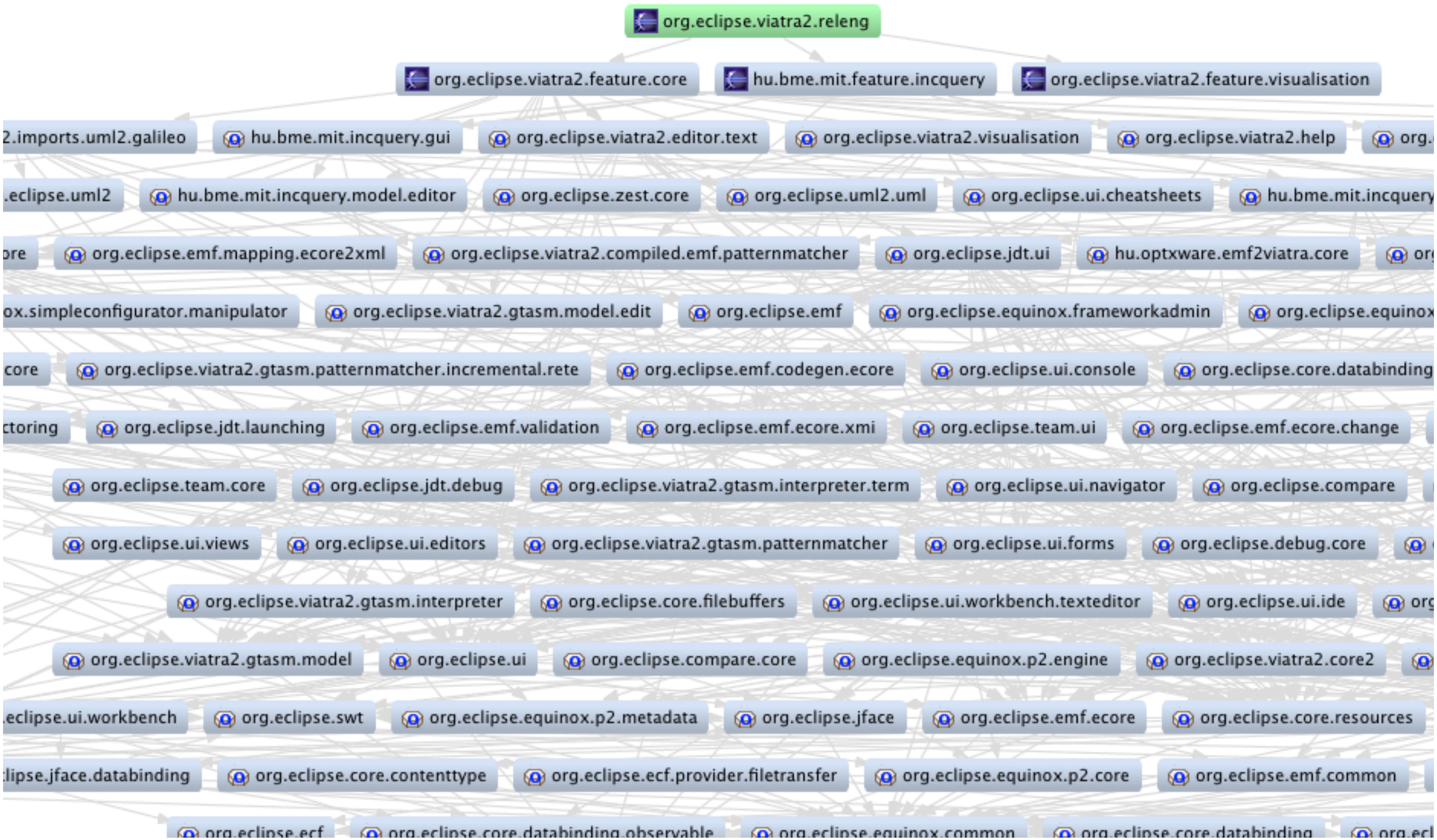
Test success
94.3% ▲
14 failures ▲
2 errors

Results

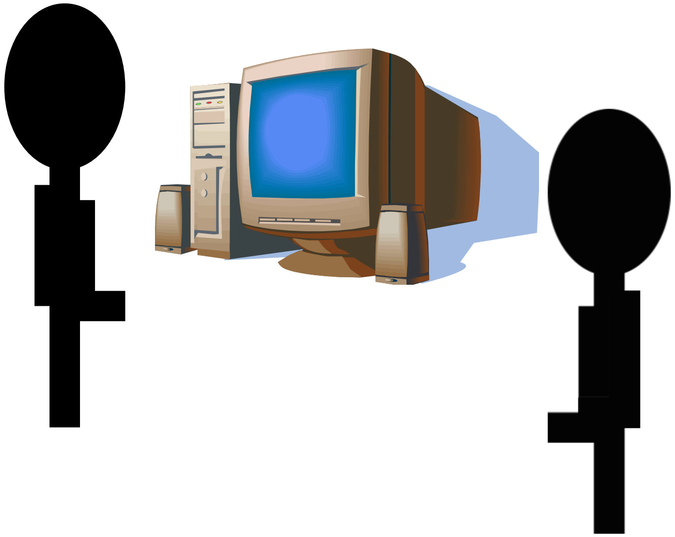
- What is required?
  - Automatic compilation
  - Automatic integration
  - Automatic testing
- What is provided?
  - Source code collection
  - Scheduling
  - Publishing
    - Reports
    - Results

CI Builds of Eclipse Plug-ins

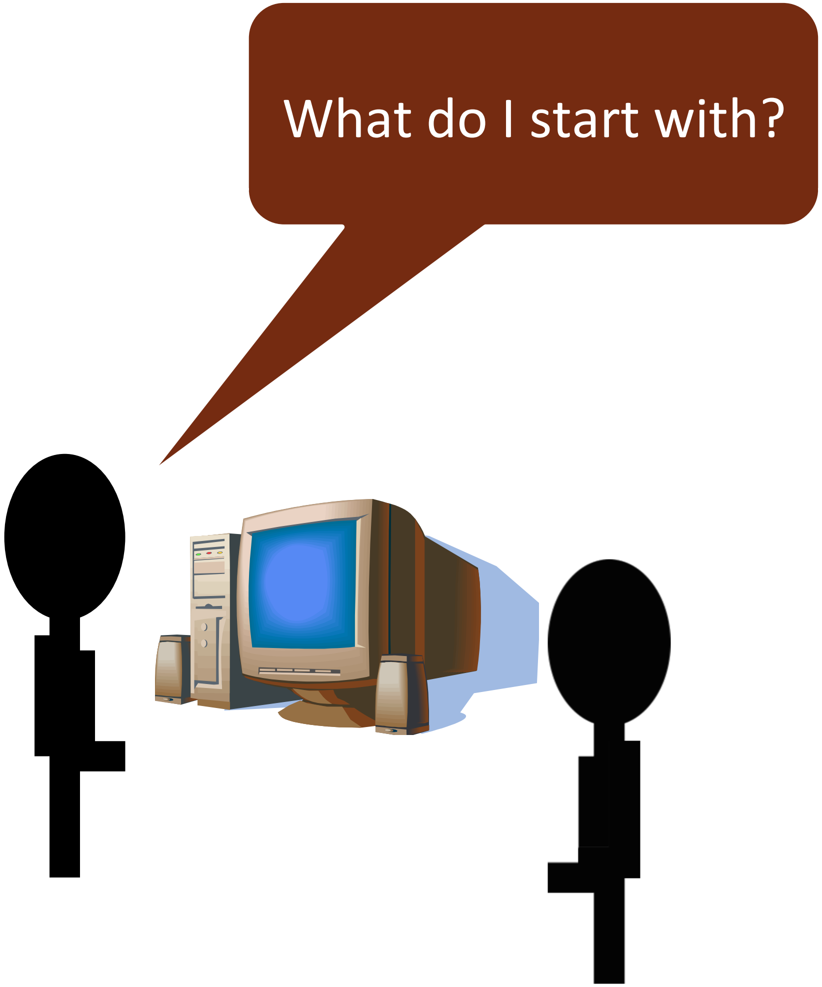
Problem


org.eclipse.viatra2.releng
org.eclipse.viatra2.feature.core
hu.bme.mit.feature.incquery
org.eclipse.viatra2.feature.visualisation
2.imports.uml2.galileo
hu.bme.mit.incquery.gui
org.eclipse.viatra2.editor.text
org.eclipse.viatra2.visualisation
org.eclipse.viatra2.help
org.
.eclipse.uml2
hu.bme.mit.incquery.model.editor
org.eclipse.zest.core
org.eclipse.uml2.uml
org.eclipse.ui.cheatsheets
hu.bme.mit.incquery
ore
org.eclipse.emf.mapping.ecore2xml
org.eclipse.viatra2.compiled.emf.patternmatcher
org.eclipse.jdt.ui
hu.optxware.emf2viatra.core
or
ox.simpleconfigurator.manipulator
org.eclipse.viatra2.gtasm.model.edit
org.eclipse.emf
org.eclipse.equinox.frameworkadmin
org.eclipse.equinox
core
org.eclipse.viatra2.gtasm.patternmatcher.incremental.rete
org.eclipse.emf.codegen.ecore
org.eclipse.ui.console
org.eclipse.core.databinding
ctoring
org.eclipse.jdt.launching
org.eclipse.emf.validation
org.eclipse.emf.ecore.xmi
org.eclipse.team.ui
org.eclipse.emf.ecore.change
org.eclipse.team.core
org.eclipse.jdt.debug
org.eclipse.viatra2.gtasm.interpreter.term
org.eclipse.ui.navigator
org.eclipse.compare
org.eclipse.ui.views
org.eclipse.ui.editors
org.eclipse.viatra2.gtasm.patternmatcher
org.eclipse.ui.forms
org.eclipse.debug.core
org.eclipse.viatra2.gtasm.interpreter
org.eclipse.core.filebuffers
org.eclipse.ui.workbench.texteditor
org.eclipse.ui.ide
org
org.eclipse.viatra2.gtasm.model
org.eclipse.ui
org.eclipse.compare.core
org.eclipse.equinox.p2.engine
org.eclipse.viatra2.core2
.eclipse.ui.workbench
org.eclipse.swt
org.eclipse.equinox.p2.metadata
org.eclipse.jface
org.eclipse.emf.ecore
org.eclipse.core.resources
lipse.jface.databinding
org.eclipse.core.contenttype
org.eclipse.ecf.provider.filetransfer
org.eclipse.equinox.p2.core
org.eclipse.emf.common
org.eclipse.ecf
org.eclipse.core.databinding.observable
org.eclipse.equinox.common
org.eclipse.core.databinding
org.ecl

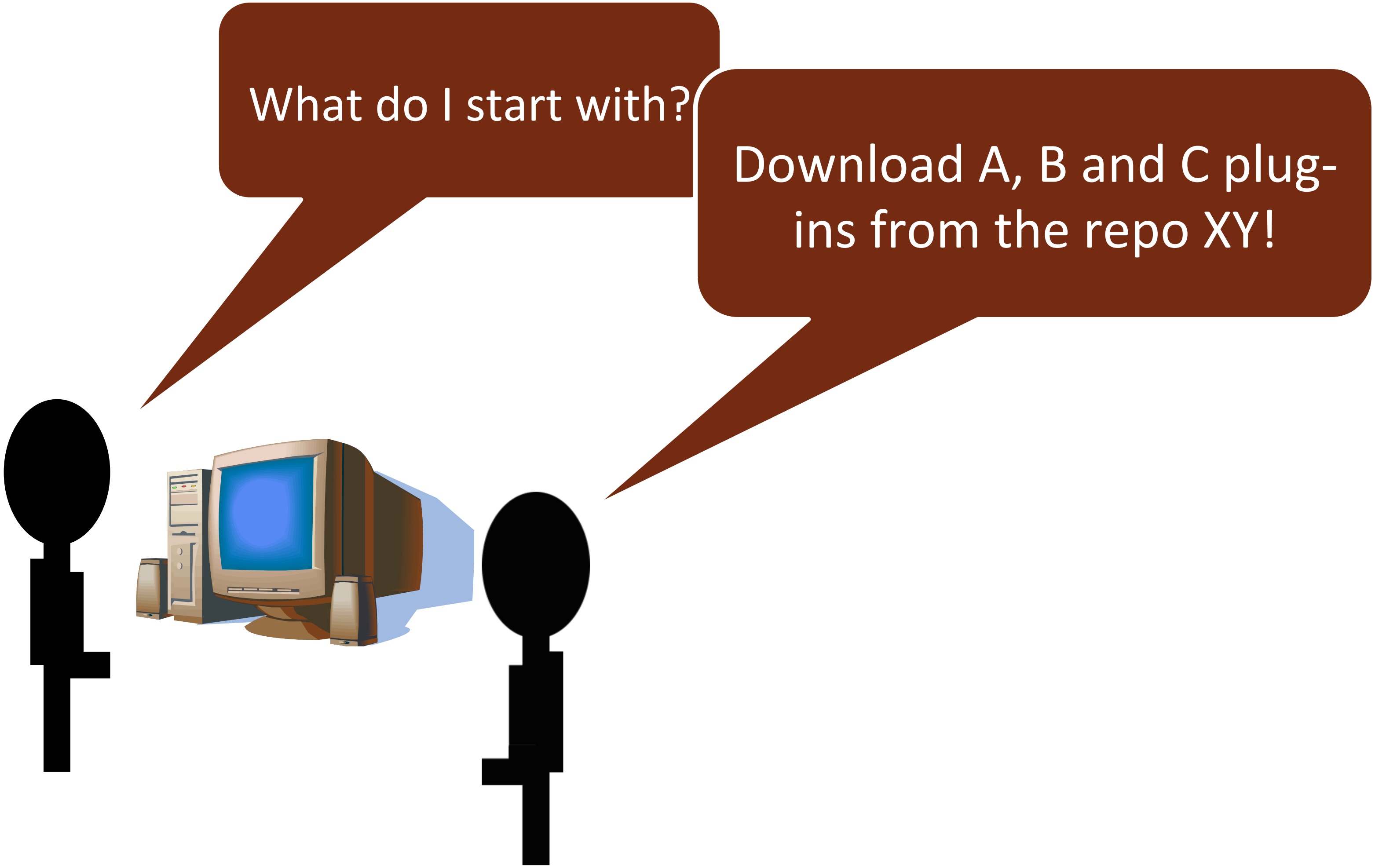
Problem



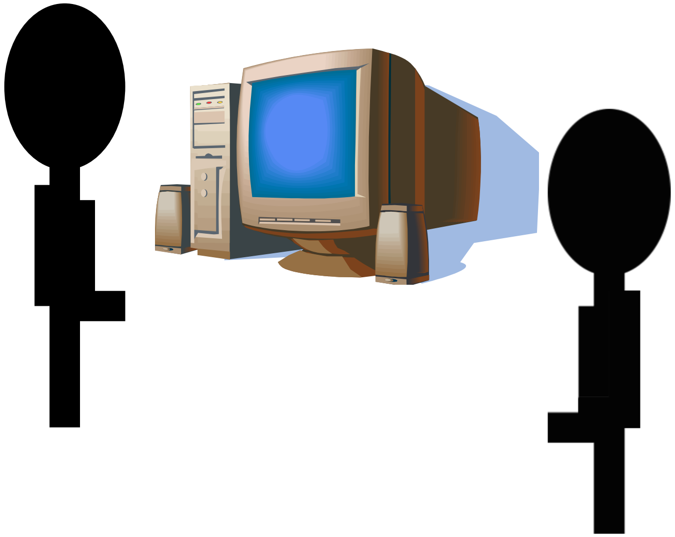
Problem



Problem

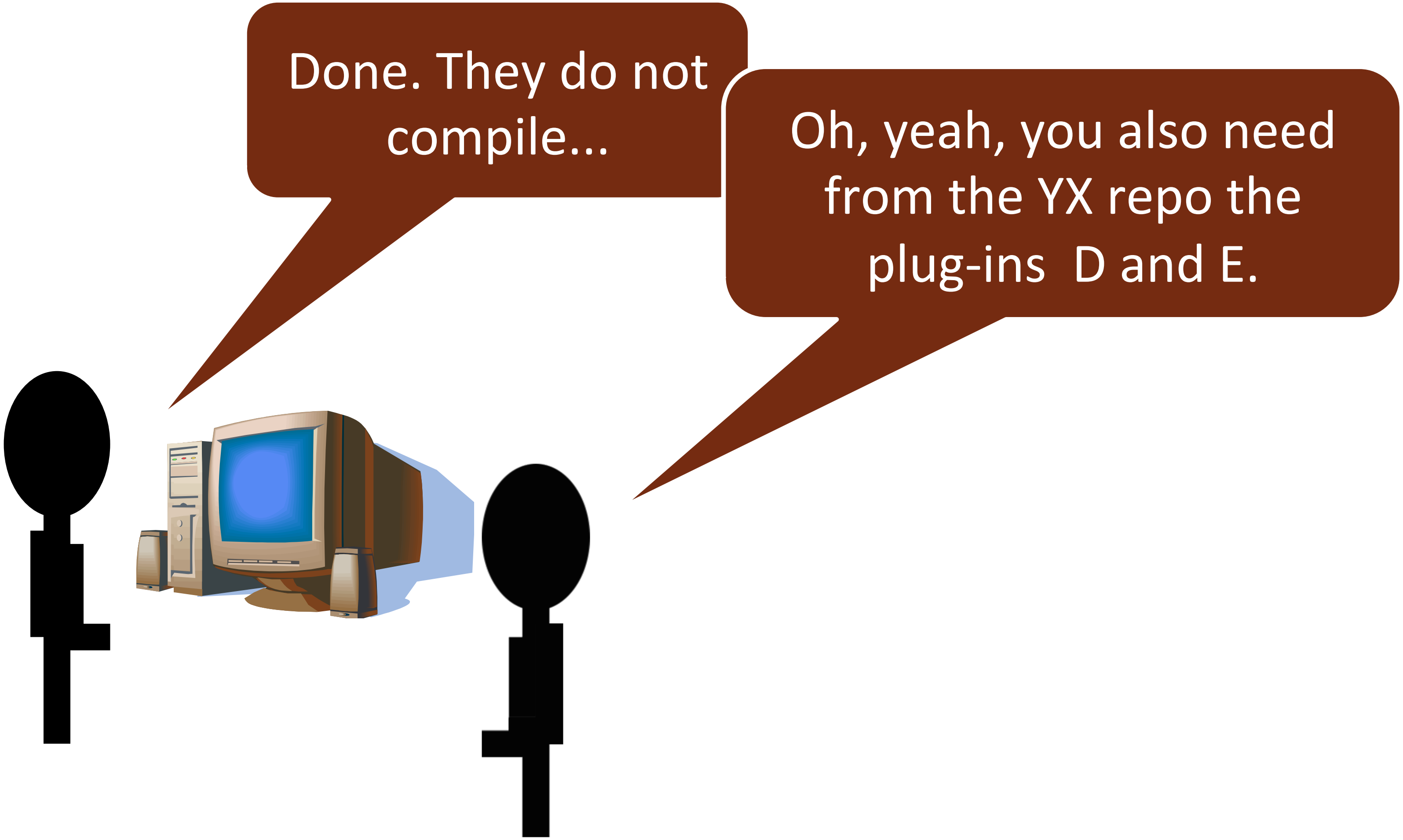

What do I start with?
Download A, B and C plug-ins from the repo XY!

Problem

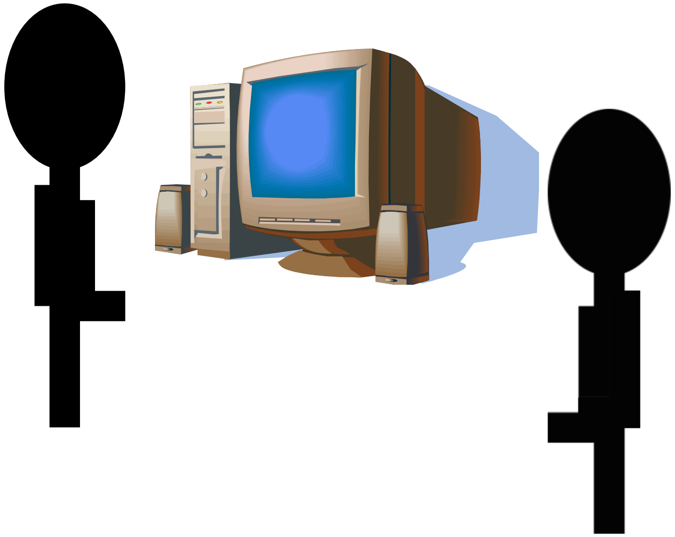


Problem

Problem


Done. They do not compile...
Oh, yeah, you also need from the YX repo the plug-ins D and E.

Problem

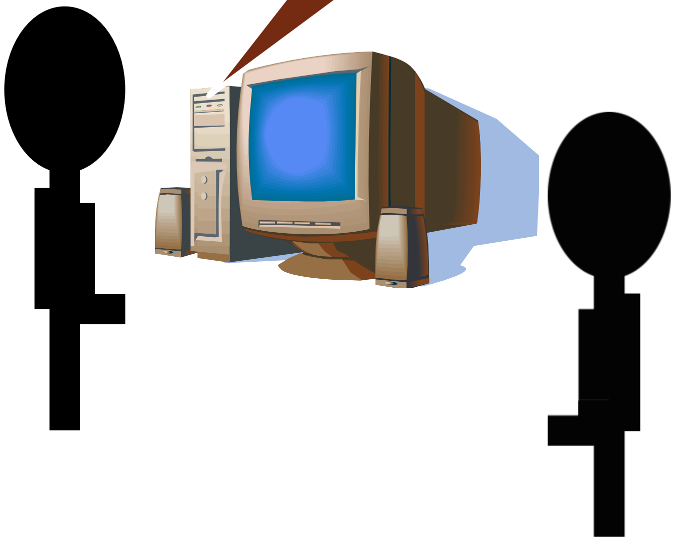


Problem

Problem

Something is still missing...

Oh, of course, from plug-in D only version 1.2 works. Also copy F to the plug-ins folder...



Automatic Compilation of Eclipse Plug-ins

- Headless execution required
  - E.g. from command line
  - Without manual steps
- Target platform
  - Handcrafted, or
  - Created during the build

Dependency Management

- Ant4Eclipse
  - Avoids using PDE/Build
- Pax, **Tycho**
  - Extends Maven with OSGi dependencies
- PDE headless build
  - Generates Ant scripts
    - Basically non-understandable by humans
- **Buckminster**

Buckminster

Buckminster

- Eclipse Tools Project
- High-level tool
  - Re-uses Eclipse builders
    - Buildable in Eclipse -> Buildable with Buckminster
  - Defines **descriptors**
    - XML documents
    - Partially generated
    - Editing support for other
  - **Dependency** management

Usage Profiles

- IDE support
  - Editing descriptors
  - Build execution
  - Collects dependencies
- Headless mode
  - Requires providing an Eclipse instance
- Hudson/Jenkins plug-in
  - Uses headless mode
  - Provides easier configuration

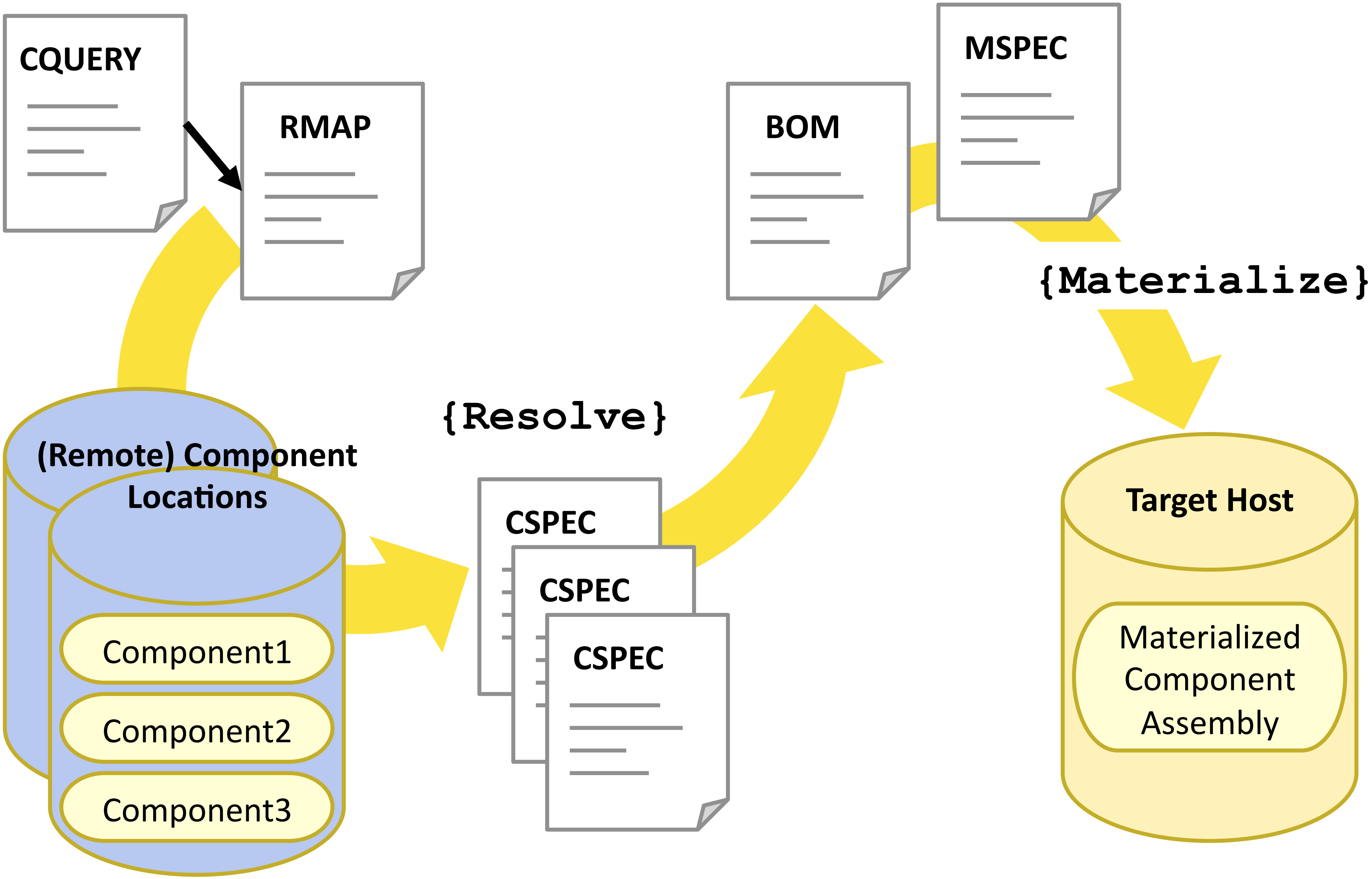
Capabilities

- Collecting source files
- Building
  - **PDE/Build**, Ant, Maven
- Packaging
  - P2 update site
  - Target platform

Basics: Component

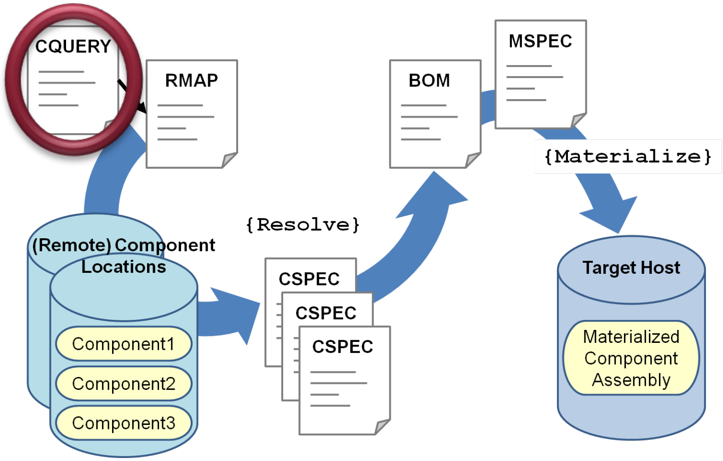
- A component is a buildable element
  - Feature, plug-in…
  - Has name, type, version
- Can execute operations
  - Some predefined (pl. site.p2, bundle.jar)
  - Custom operations

Descriptors

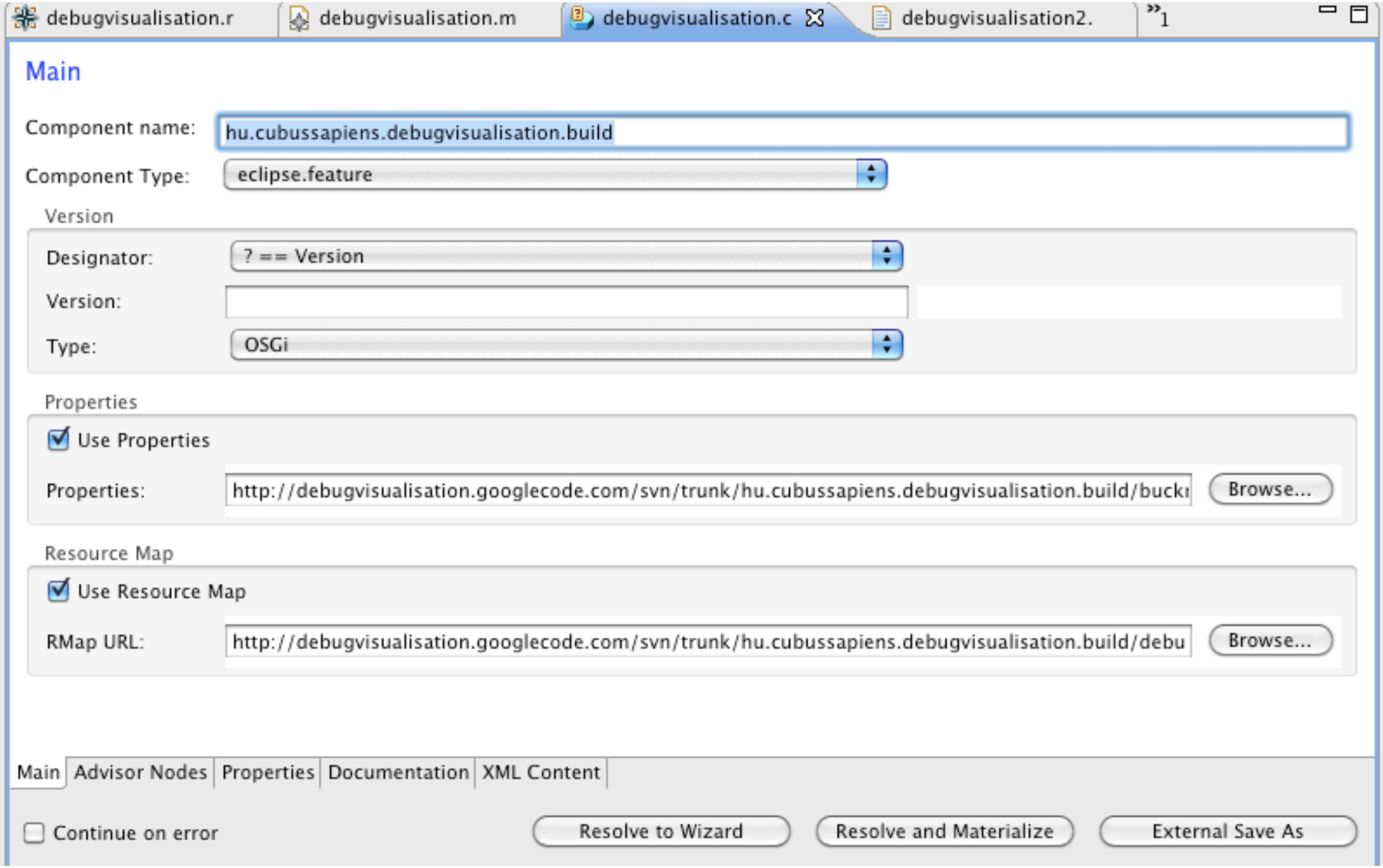


CQuery

- *Component Query*
- Describes what to collect/build
  - Only top-level element listed
  - Dependencies are resolved by Buckminster



CQuery Editor



Main

Component name: hu.cubussapiens.debugvisualisation.build

Component Type: eclipse.feature

Version

Designator: ? == Version

Version:

Type: OSGi

Properties

☑ Use Properties

Properties: http://debugvisualisation.googlecode.com/svn/trunk/hu.cubussapiens.debugvisualisation.build/buck

Resource Map

☑ Use Resource Map

RMap URL: http://debugvisualisation.googlecode.com/svn/trunk/hu.cubussapiens.debugvisualisation.build/debu

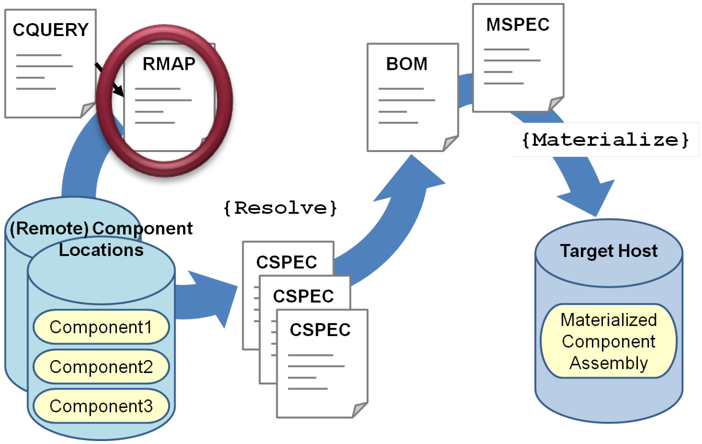
□ Continue on error

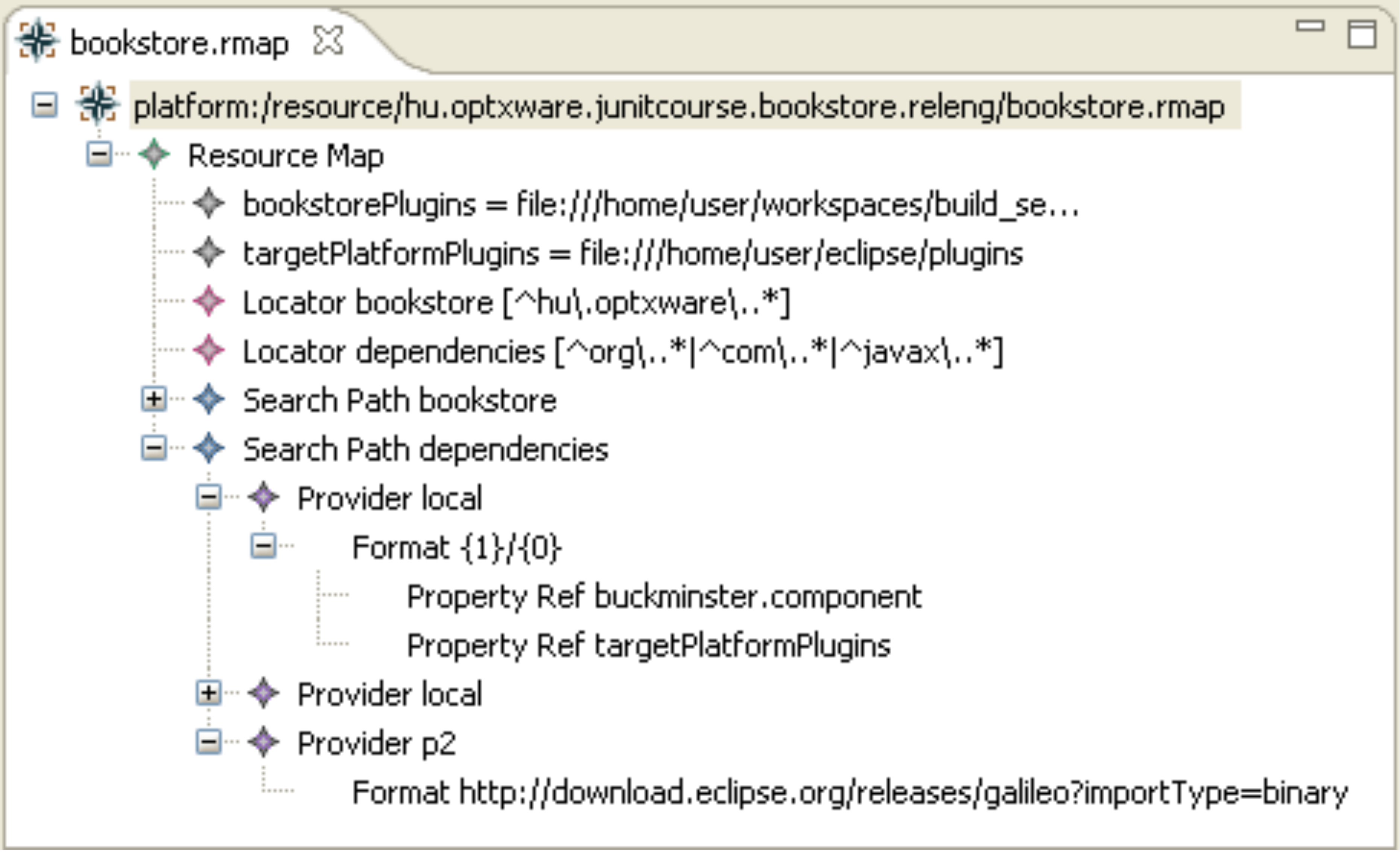
Component Query

- Query descriptor
  - What to collect?
  - Identifier + Resource map
- Optional paramters
  - Source or binary?
  - Branches/tags, etc..
  - Release/Nightly build repository

RMap

- *Resource Map*
- Where to collect stuff?
  - P2 repository
  - Local folder
  - SVN, CVS, Git…
  - Maven
  - Target platform
  - Workspace
  - URL

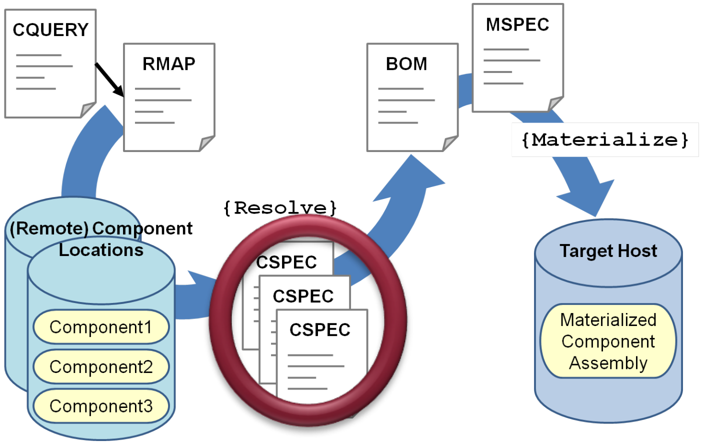




```
platform:/resource/hu.optxware.junitcourse.bookstore.releng/bookstore.rmap
  Resource Map
    bookstorePlugins = file:///home/user/workspaces/build_se...
    targetPlatformPlugins = file:///home/user/eclipse/plugins
    Locator bookstore [^hu\.optxware\..*]
    Locator dependencies [^org\..*|^com\..*|^javax\..*]
    Search Path bookstore
    Search Path dependencies
      Provider local
        Format {1}/{0}
          Property Ref buckminster.component
          Property Ref targetPlatformPlugins
      Provider local
      Provider p2
        Format http://download.eclipse.org/releases/galileo?importType=binary
```

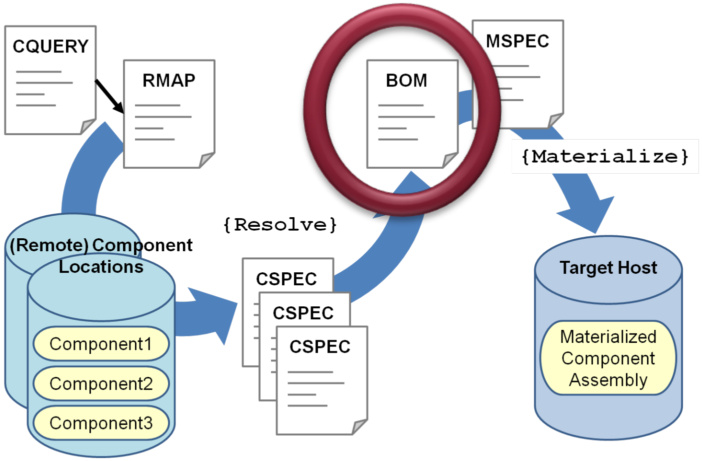
CSpec

- *Component Specification*
- Generated
- Contains executable operations
- Custom extensions:
  - *CSpeX (CSpec eXtension)*



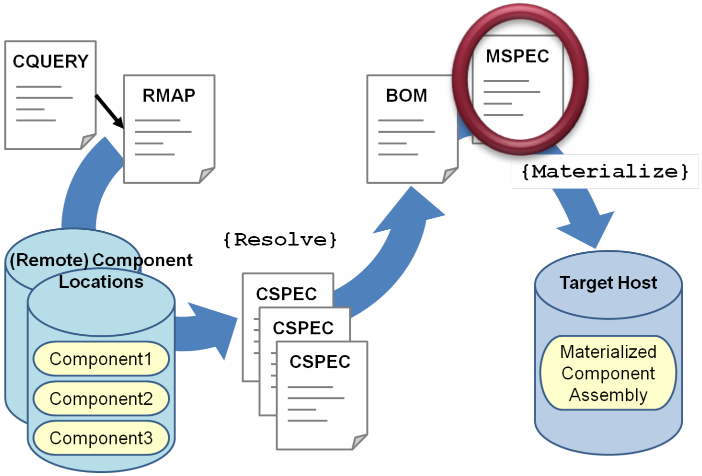
BOM

- *Bill Of Materials*
- Generated
- List of elements to download and steps to execute



MSpec

- *Materialization Specification*
- Where to put found stuff?
  - Workspace
  - Target platform
  - Selected folder
- Defaults:
  - Source into workspace
  - Binary to target platform
  - Good default



Buckminster - Summary

- Collects the components
- Defined operations
- From here
  - Execute build
  - Run tests
  - …

Further Reading: BuckyBook

- Eclipse Buckminster, The Definitive Guide
  - http://www.eclipse.org/downloads/download.php?file=/tools/buckminster/doc/BuckyBook.pdf
  - 271 page long „draft”

Maven/Tycho

Maven Tycho

- Maven POM is simple
  - “Manifest-first” approach
    - Develop in Eclipse normally
    - Add minimal descriptors for builds
  - Minor settings duplication ☹

Tycho Packaging Types

- eclipse-plugin
  - Plug-in projects
- eclipse-test-plugin
  - Plug-in tests
- eclipse-feature
  - Feature projects
- eclipse-repository
  - RCP applications and p2 repositories projects
- eclipse-target-definition
  - Target platform definition projects

Tycho Sample: Minerva project

- Minerva project
  - Target platform
  - Building
  - Tests
- Links
  - http://wiki.eclipse.org/Minerva
  - https://github.com/caniszczyk/minerva

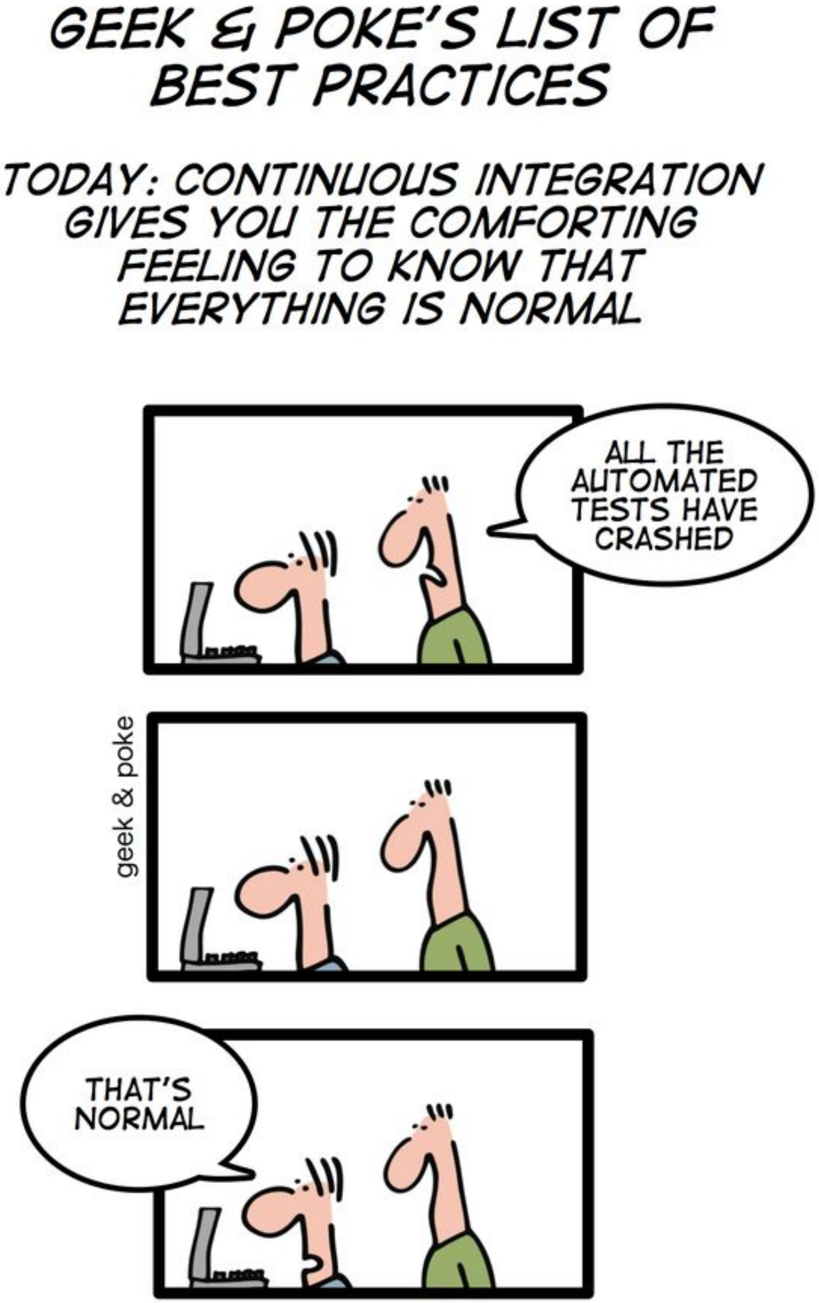
Tycho: Try it out

- Three steps:
  - Install Maven
  - git clone git://github.com/caniszczyk/minerva.git
  - mvn -Dskip-ui-tests=true clean install

Summary

Summary

- Test automatization
  - Complex process
  - Many steps
  - Automatization can happen one-by-one
- Build process
  - Required
  - Goal: reproducibility
  - Good tool support
  - BUT: It has to be created at first


GEEK & POKE'S LIST OF BEST PRACTICES
TODAY: CONTINUOUS INTEGRATION GIVES YOU THE COMFORTING FEELING TO KNOW THAT EVERYTHING IS NORMAL
ALL THE AUTOMATED TESTS HAVE CRASHED
geek & poke
THAT'S NORMAL

http://geekandpoke.typepad.com/geekandpoke/2010/10/geekpokes-list-of-best-prac