Continuous Integration

Multi-Stage Builds for Quality Assurance

Dr. Beat Fluri
Comerge AG

comerge

ABOUT

MSc ETH in Computer Science

Dr. Inform. UZH, s.e.a.l. group

Over 8 years of experience in object-oriented software engineering with Java

Special focus on Java EE 6

Roadmap

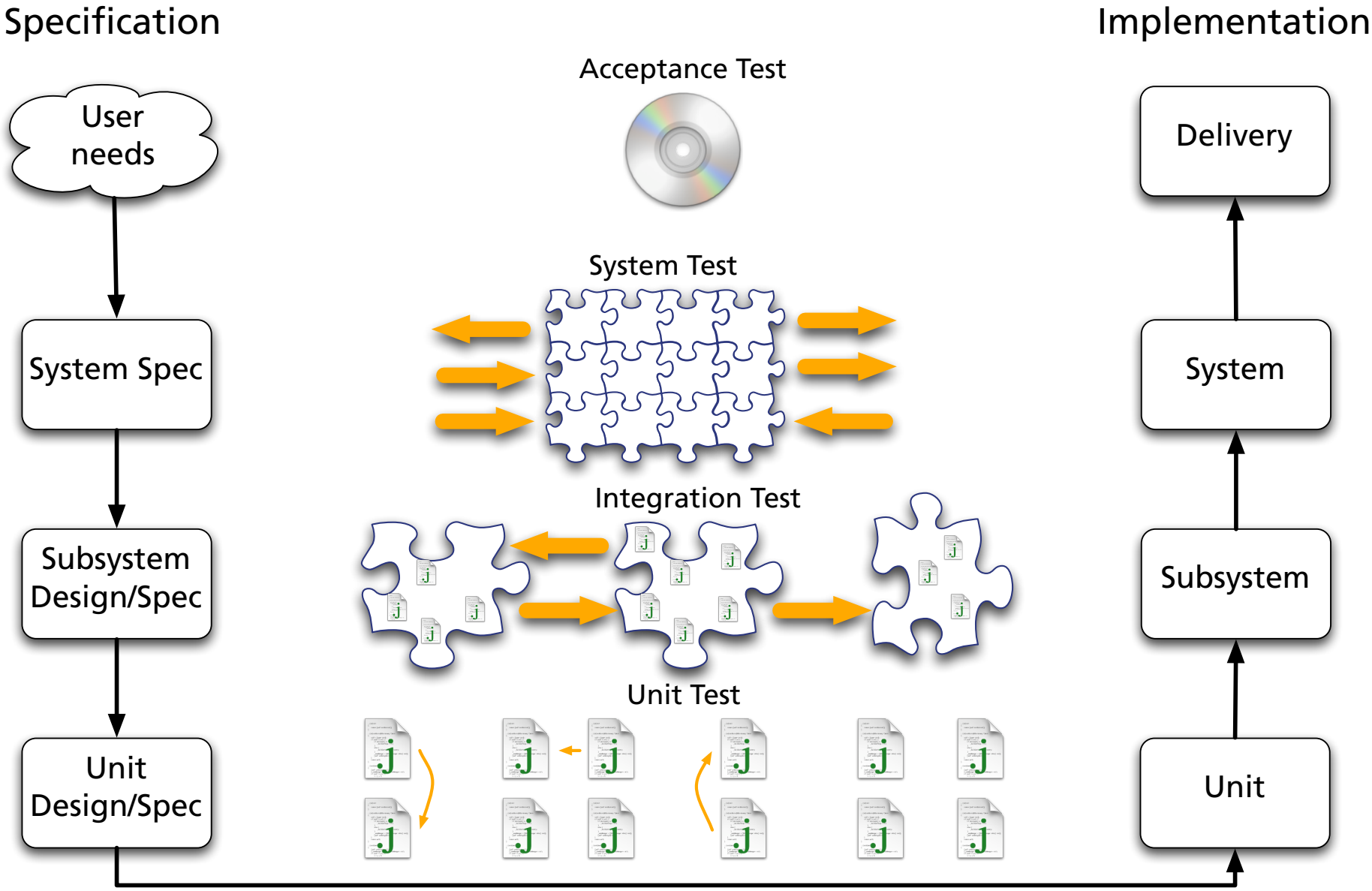
> V-Model

> Continuous Integration

> Example JEE RESTful application

> Multistage Builds for V-Model

V-Model


Specification
Implementation
Acceptance Test
User needs
Delivery
System Test
System Spec
System
Integration Test
Subsystem Design/Spec
Subsystem
Unit Test
Unit Design/Spec
Unit

Continuous Integration

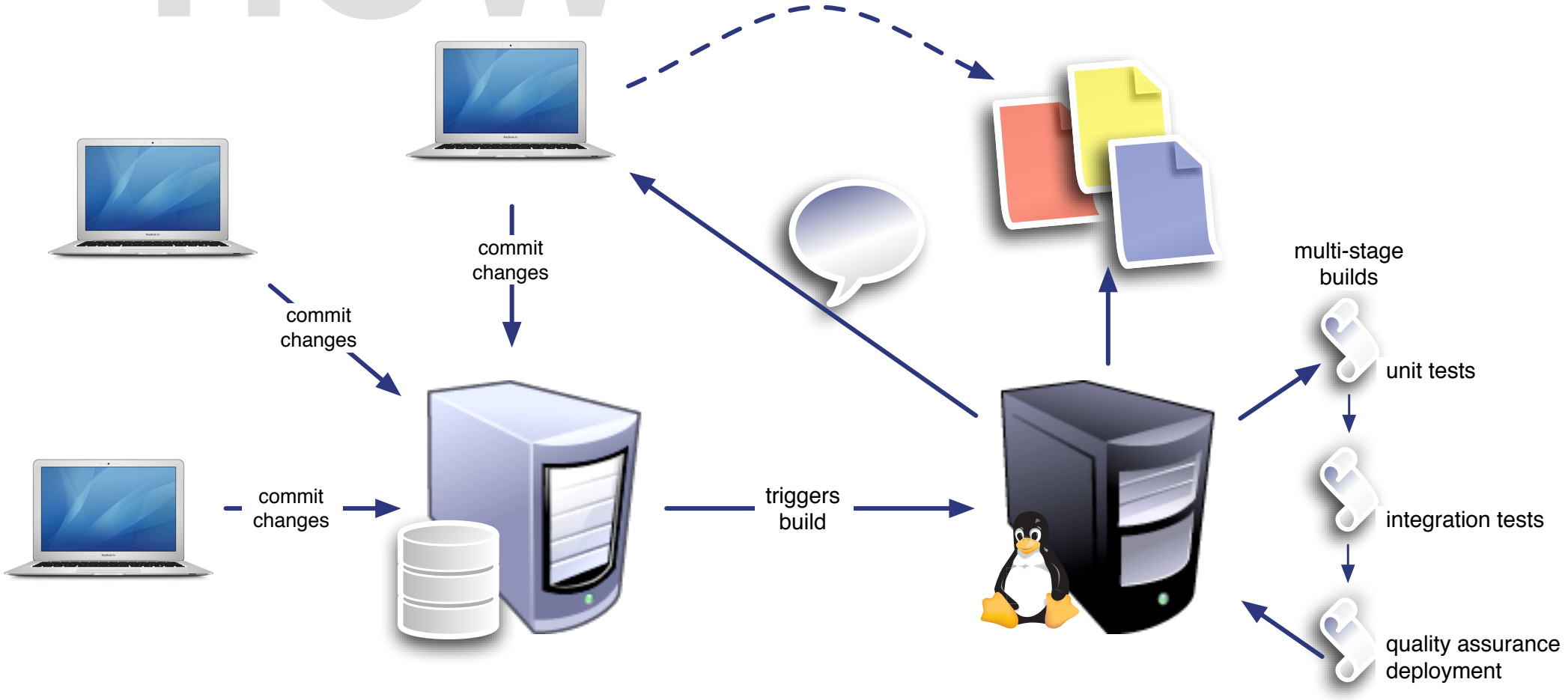
WHAT

... 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. Many teams find that this approach leads to significantly reduced integration problems and allows a team to develop cohesive software more rapidly.

Martin Fowler, http://martinfowler.com/articles/continuousIntegration.html

Continuous Integration

HOW



Continuous Integration

HOW

- Version control repository: CVS, **Subversion**, **Git**, Mercurial, etc.
- Build tools: **Ant**, **Maven**, Make, Gant, Grails, Rake, etc.
- Continuous integration environment (server): CruiseControl, Continuum, **Hudson/Jenkins**

Continuous Integration

WHY

- **Reduce risks**
  - Defects are detected when they are introduced
  - Measure the health of a software
  - Environment always the same and build starts clean => no assumptions

Source: Paul M. Duval. Continuous Integration. Pearson Education, Inc., 2007

Continuous Integration

WHY

- **Reduce repetitive manual processes (safe time and costs)**
  - Process runs the same every time
  - Ordered process: compile, unit tests, integration tests, qa, etc.

Source: Paul M. Duval. Continuous Integration. Pearson Education, Inc., 2007

Continuous Integration

WHY

- **Generate deployable software at any time and at any place**
  - Going back in the build history to deploy older, maybe stable, builds

Source: Paul M. Duval. Continuous Integration. Pearson Education, Inc., 2007

Continuous Integration

WHY

- **Enable better project visibility**
  - Decisions on quality improvements and tests are shown immediately
  - Ability to notice trends in various quality metrics (# bugs; # checkstyle, findbugs, pmd violations; code coverage of test cases)

Source: Paul M. Duval. Continuous Integration. Pearson Education, Inc., 2007

Continuous Integration

WHY

- **Establish greater confidence in the software product from the development team**
  - Making progress is visible and encourages developers
  - Confidence increases if the increase of overall product quality is visible

Source: Paul M. Duval. Continuous Integration. Pearson Education, Inc., 2007

Continuous Integration

WHY

- **Same code basis for every developer: reduces communication overhead**
  - Go to build #x, do you see...

Source: Paul M. Duval. Continuous Integration. Pearson Education, Inc., 2007

Continuous Integration

MEET JENKINS

Version Control

GIT

- **Version control is the management of changes to documents, programs, and other information stored as computer files.**
  - History of changes to a file visible at any time.
- **Maintain a master repository on a server and all developers get a consolidated copy of the project**

Version Control

GIT

- **Create a repository**
  - `$ mkdir project`
  - `$ cd project`
  - `$ git init`
- **Create a Java class with an empty method**

Version Control

GIT

- **Add the file to the repository and commit**
  - `$ git add SimpleClass.java`
  - `$ git commit`
- **Make a change to the method**

Maven

BUILD AUTOMATION

- Maven is a build and software project management tool
- Maven works with Project Object Models (pom.xml) files to specify project settings
- Plugin architecture to include tools into the build cycle

Maven

MAIN PRINCIPLE

- **Convention over configuration**
  - ./src/main/java
  - ./src/main/resources
  - ./src/test/java
  - ./src/test/resources
  - ./target/classes
  - ./target/* (*.jar, reports, etc.)

Maven

BUILD LIFECYCLE

> compile

> test

> package

> integration-test

> verify

> install

> deploy

> $ mvn <phase>

Maven

DEPENDENCY MANAGEMENT

- **Maven works with repositories where a huge number of jars are stored**
- **Specify a dependency in the POM file**
  - maven downloads the necessary jar file

Maven

DEPENDENCY MANAGEMENT

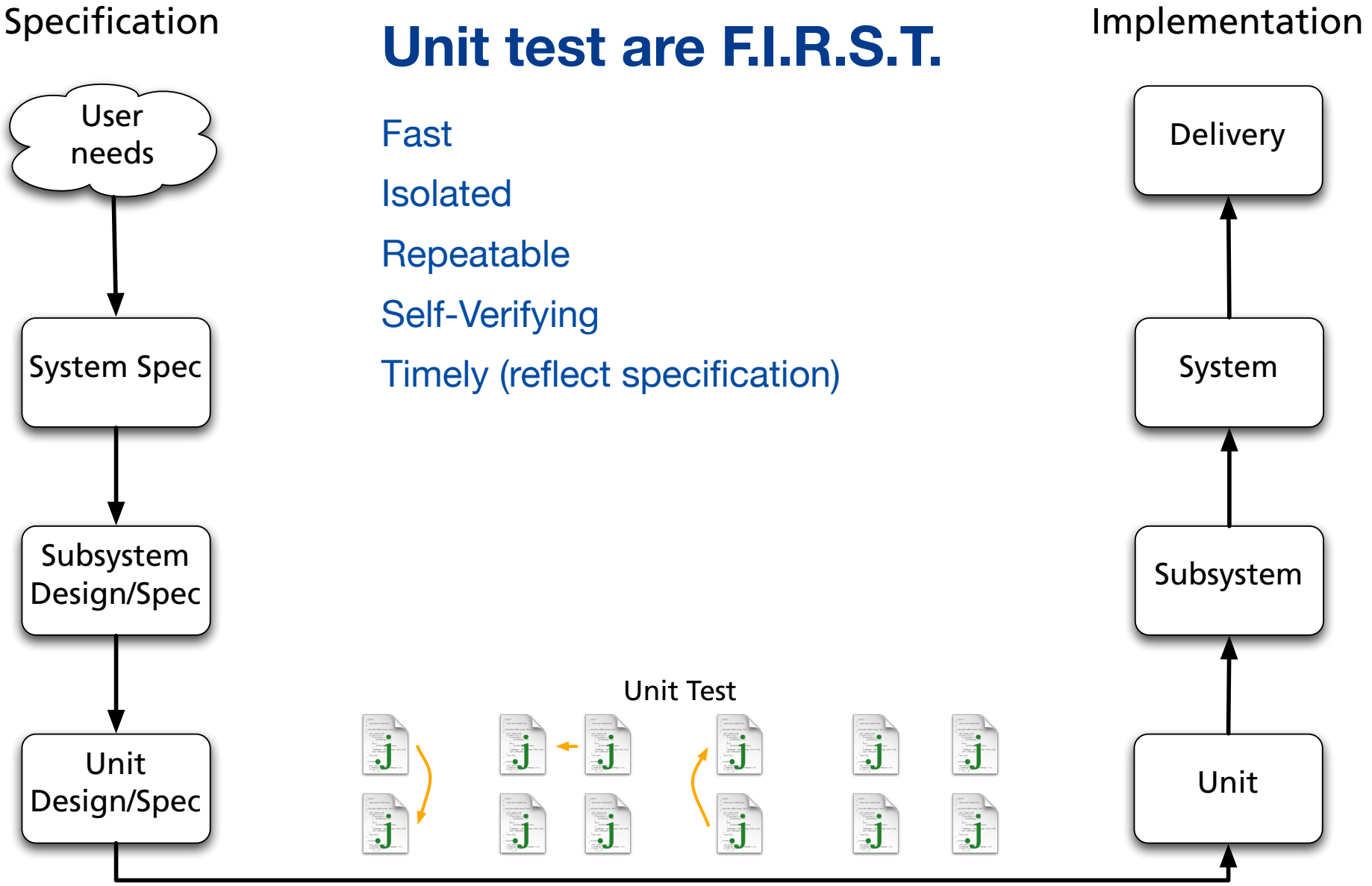
```
<dependencies>
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.8</version>
        <scope>test</scope>
    </dependency>
</dependencies>
```

Sports Ground

EXAMPLE

- **Java Enterprise Edition 6 (JEE 6)**
- **RESTful API**
  - REpresentation State Transfer (JAX-RS)
- **JBoss Application Server 7.1**
- **https://github.com/kraftan/ci-showcase**

V-Model



Unit test are F.I.R.S.T.

- Fast
- Isolated
- Repeatable
- Self-Verifying
- Timely (reflect specification)

V-Model

Specification

Implementation

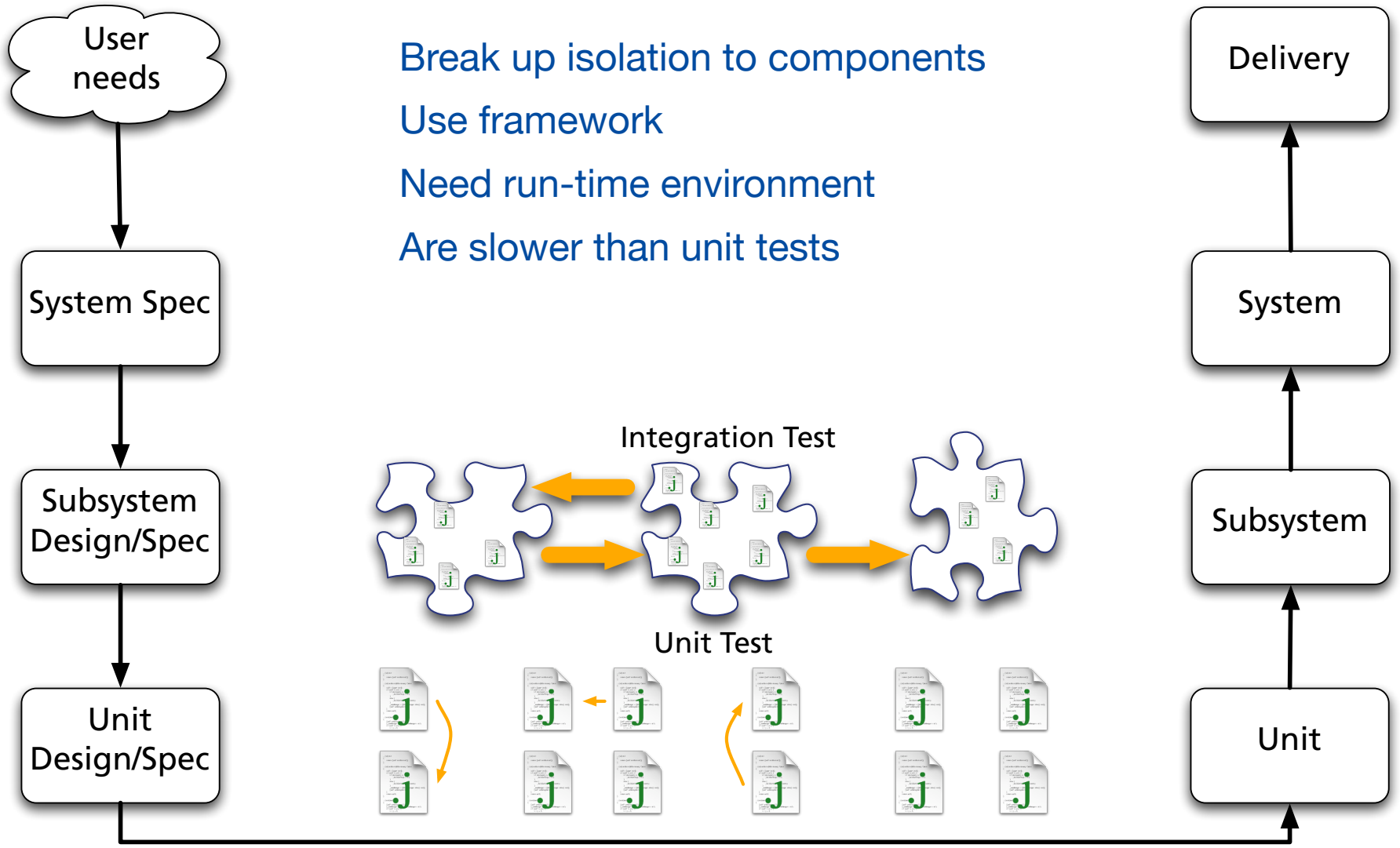
Integration tests

Break up isolation to components

Use framework

Need run-time environment

Are slower than unit tests



V-Model

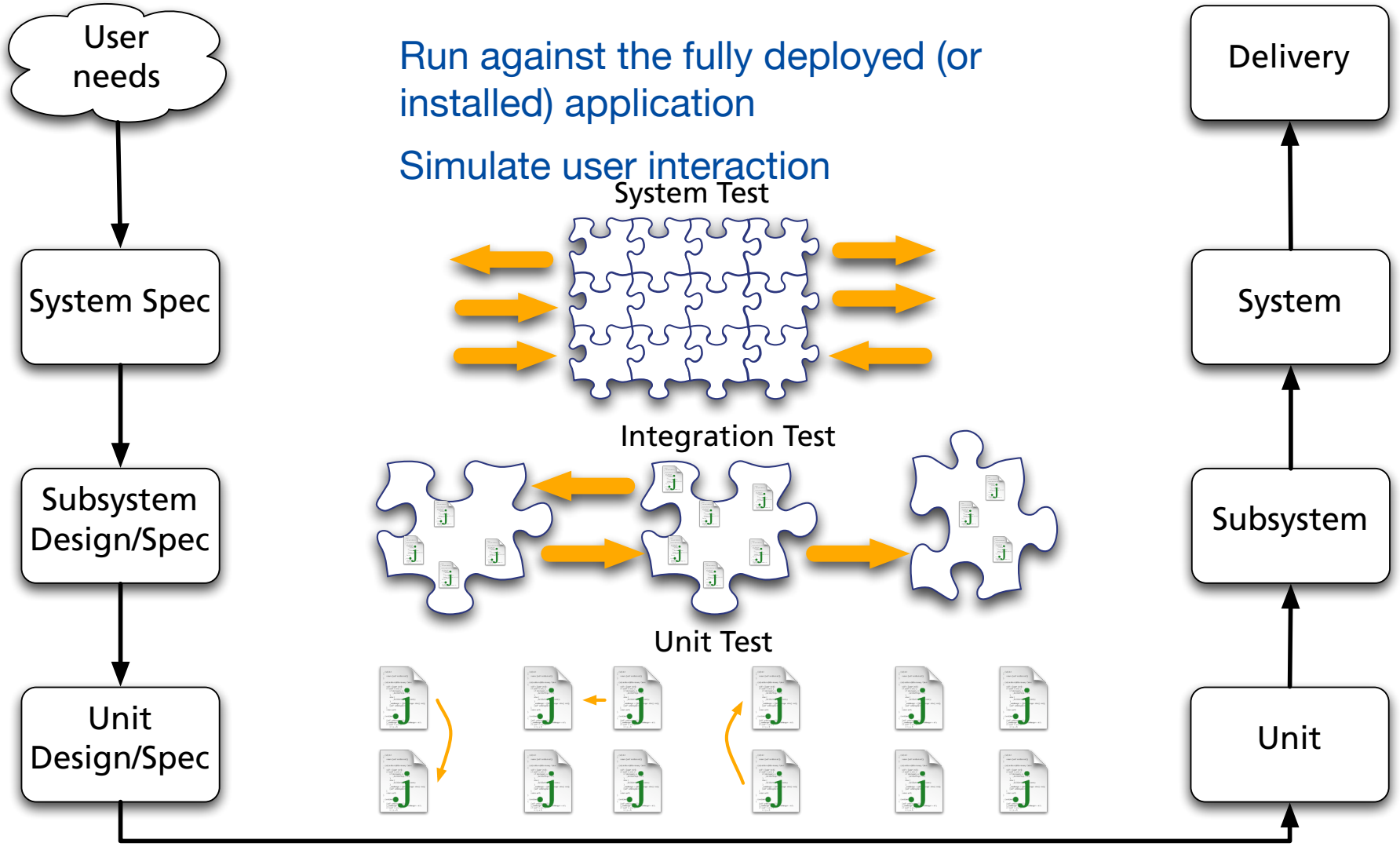
Specification

Implementation

System tests

Run against the fully deployed (or installed) application

Simulate user interaction



Non-Functional Req

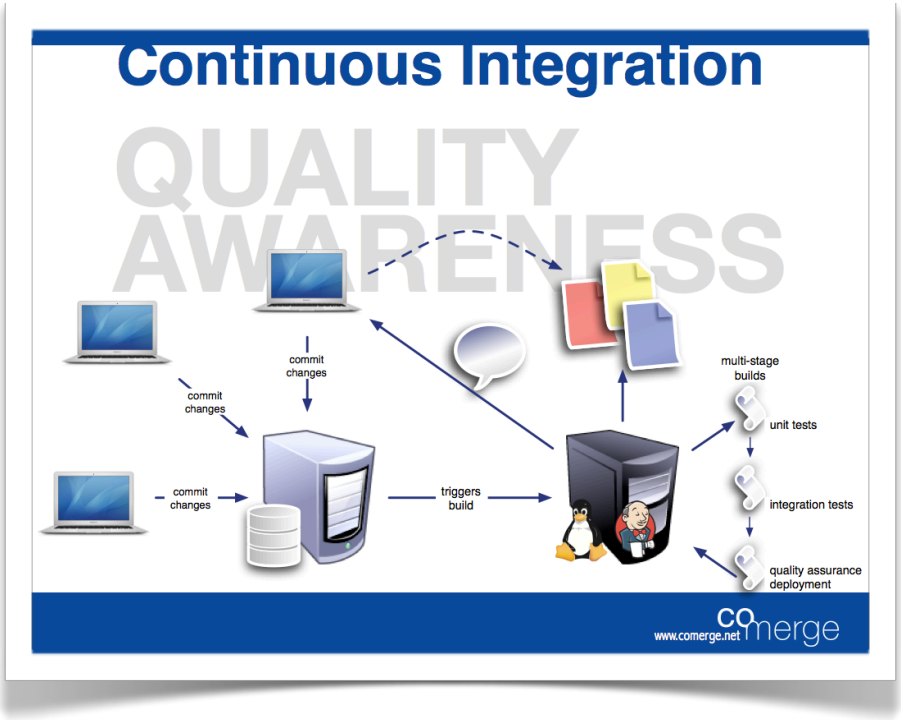
EFFICIENCY

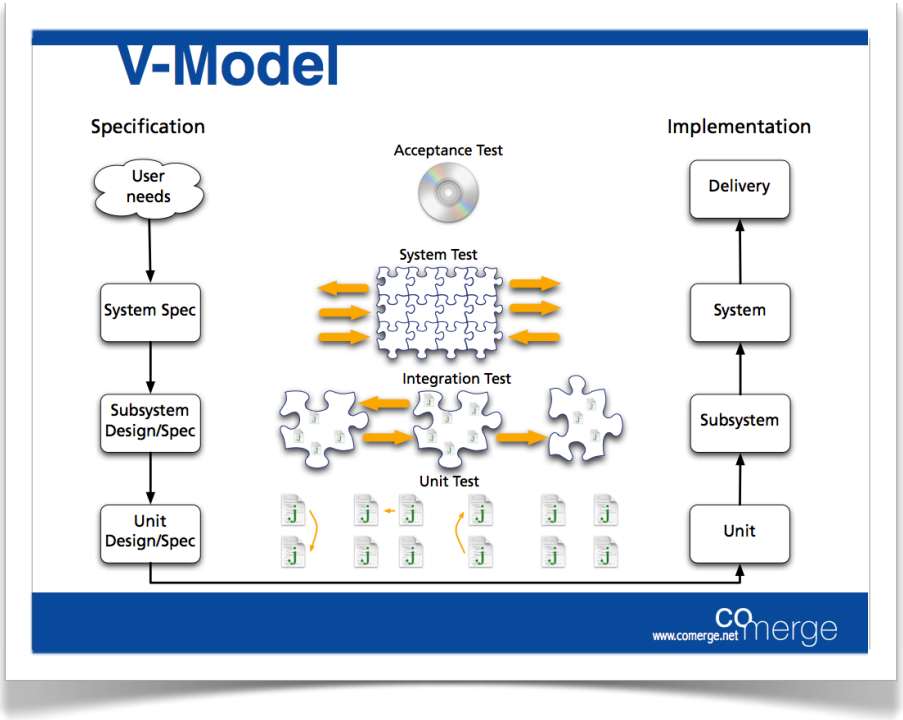
- **Performance testing**
  - Which component have a long response time?
  - For instance, certain DB query may take noticeably longer than others
- **Load testing**
  - How does the system work under heavy load?
  - How long does it take to complete a request if specific number of persons interact with the system
- **Stress testing**
  - How does the system react if it is overloaded?

Used Tools

- **Git** - Version Control
- **Maven** - Build Automation
- **Jenkins** - Continuous Integration
- **Java EE 6 on JBoss AS 7** (Example on github)
- **JUnit** - Unit testing
- **mockito** - Unit testing in isolation
- **Arquillian** - In-container integration testing
- **Sonar** - Quality Assurance
- **JMeter** - Load testing

Wrap Up





Design to Test

Use CI to Check