
When Java
Enterprise
Testing
Attacks!


Can ShrinkWrap
and Arquillian
Save Our Dev?

DAN ALLEN

@MOJAVELINUX

Principal Software Engineer, Seam Community Liaison, JBoss by Red Hat, Inc.

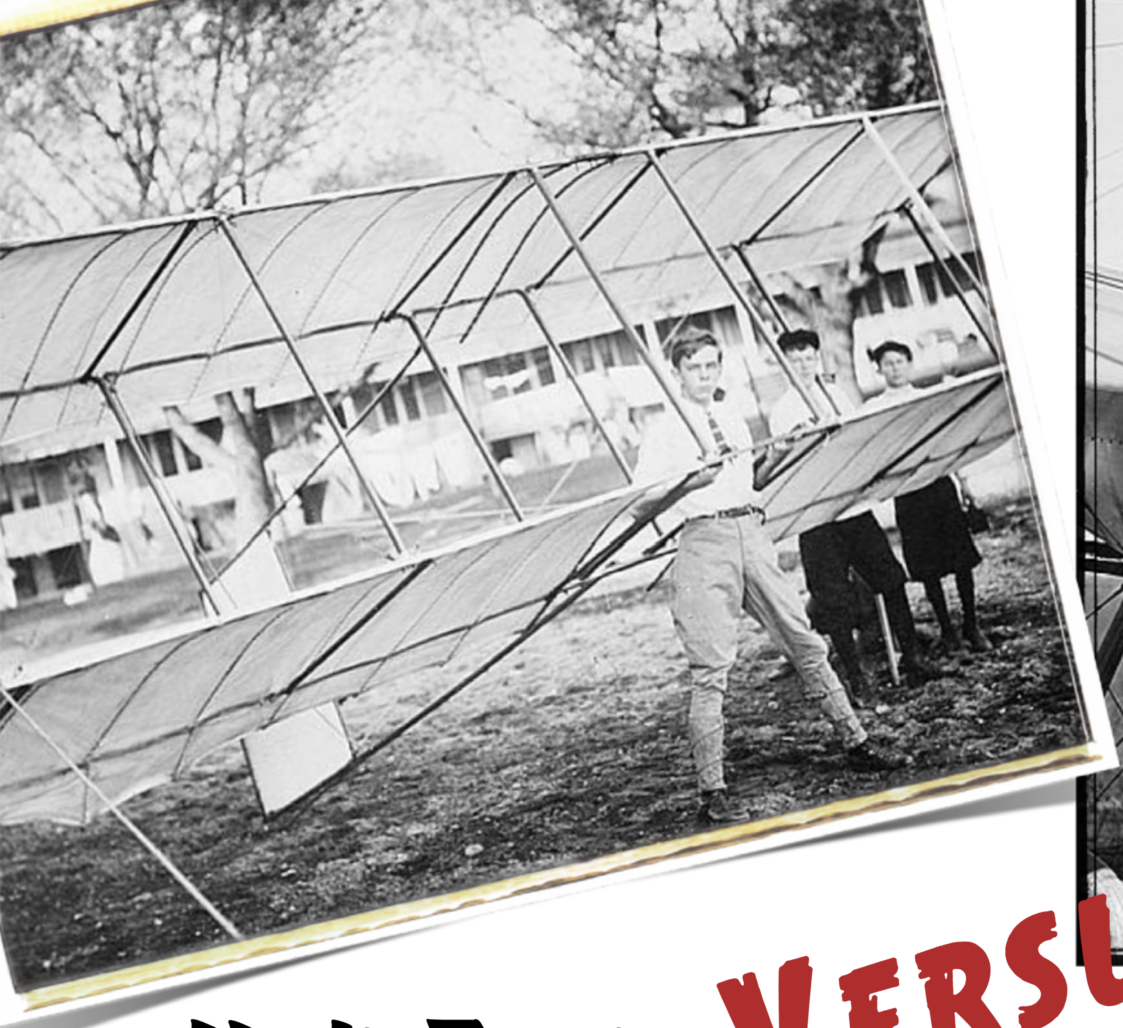
THE RESCUE PLAN

1. What helps prevent application failures?
2. What would help us write tests more efficiently?
3. How can we speed up integration testing?

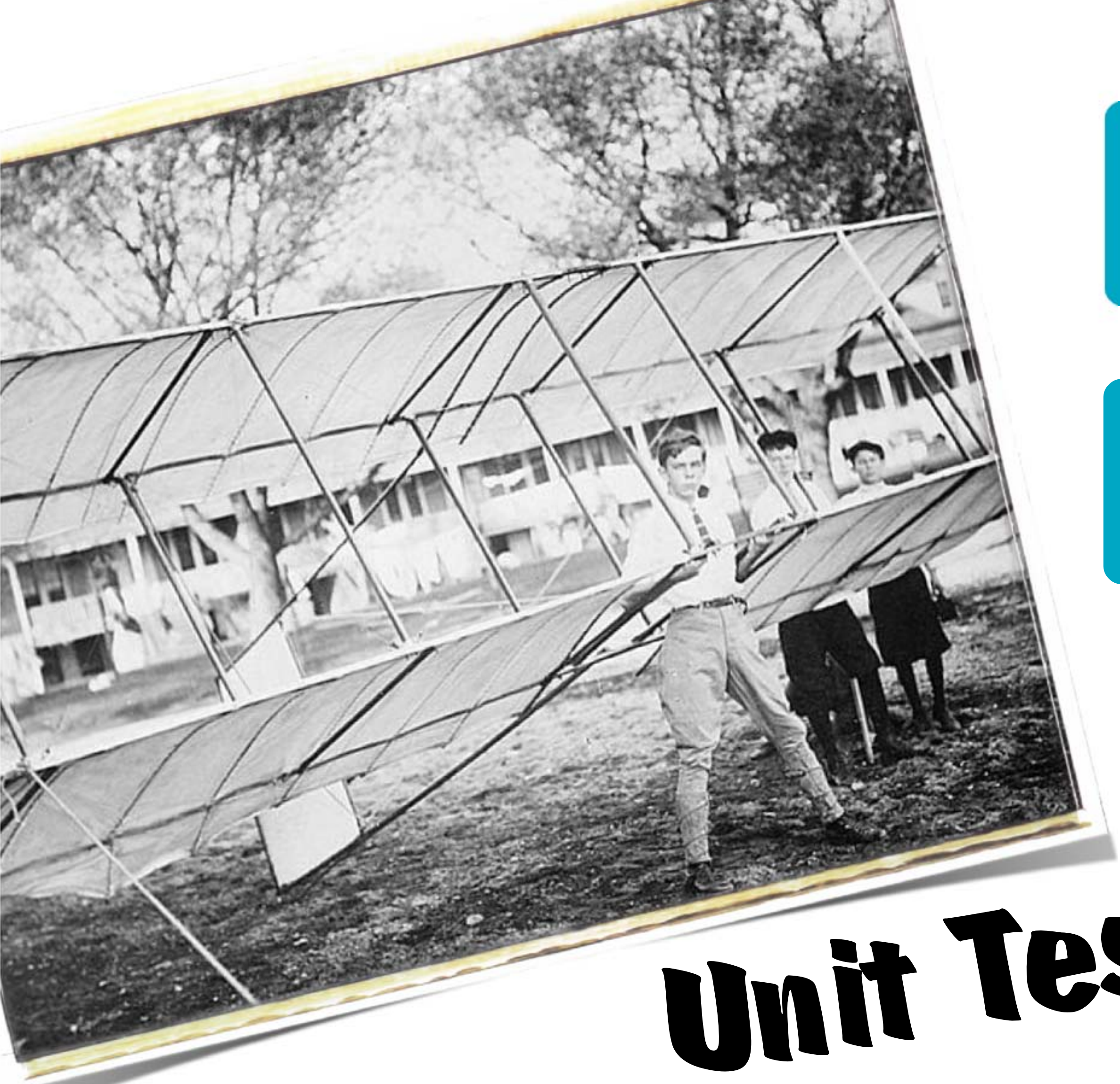
4. How can we simplify integration tests?
5. Can we put these ideas into practice?
6. What's on the horizon for enterprise testing?

What helps prevent application failures?


Let's "weigh" the options.



Unit Test VERSUS Integration Test



Unit Test

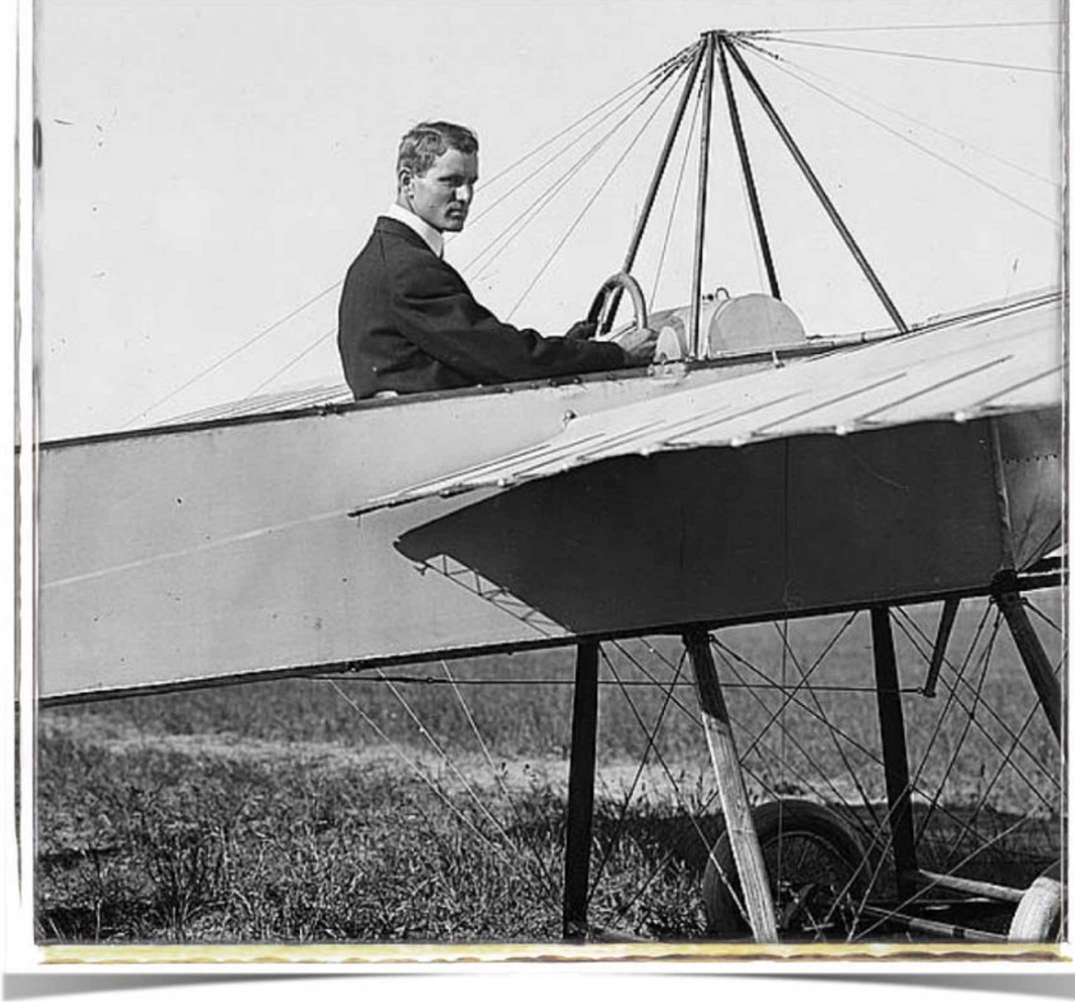
Tests a single API

Runs in isolation


Unit tests are **so** fine-grained.
Fast, simple and easy. Just as testing should be.

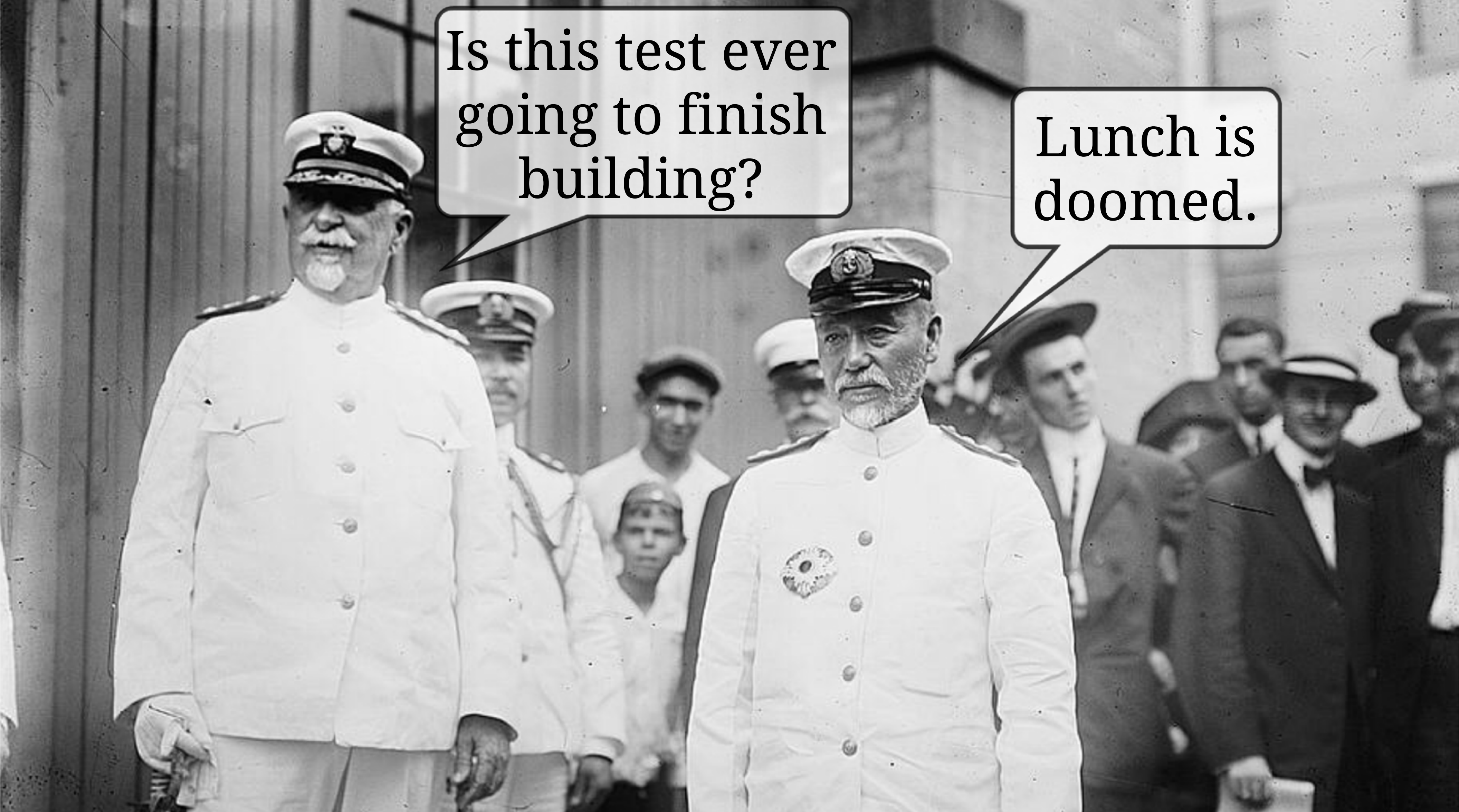
Tests many parts

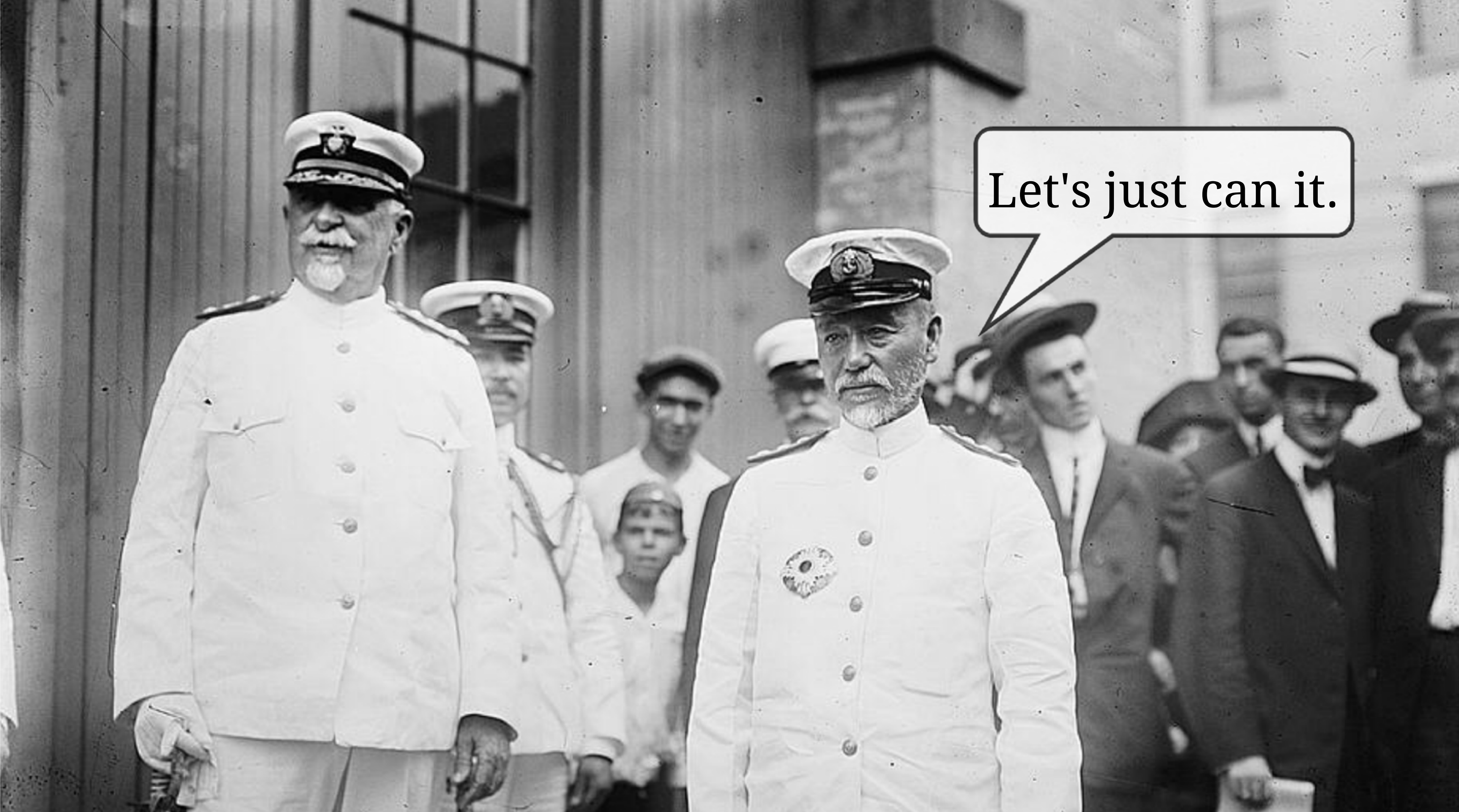
Revs up the engine



Integration Test


Nice if they could run in an IDE.
Ha! In your dreams.
MAN.


Is this test ever going to finish building?
Lunch is doomed.


Let's just can it.

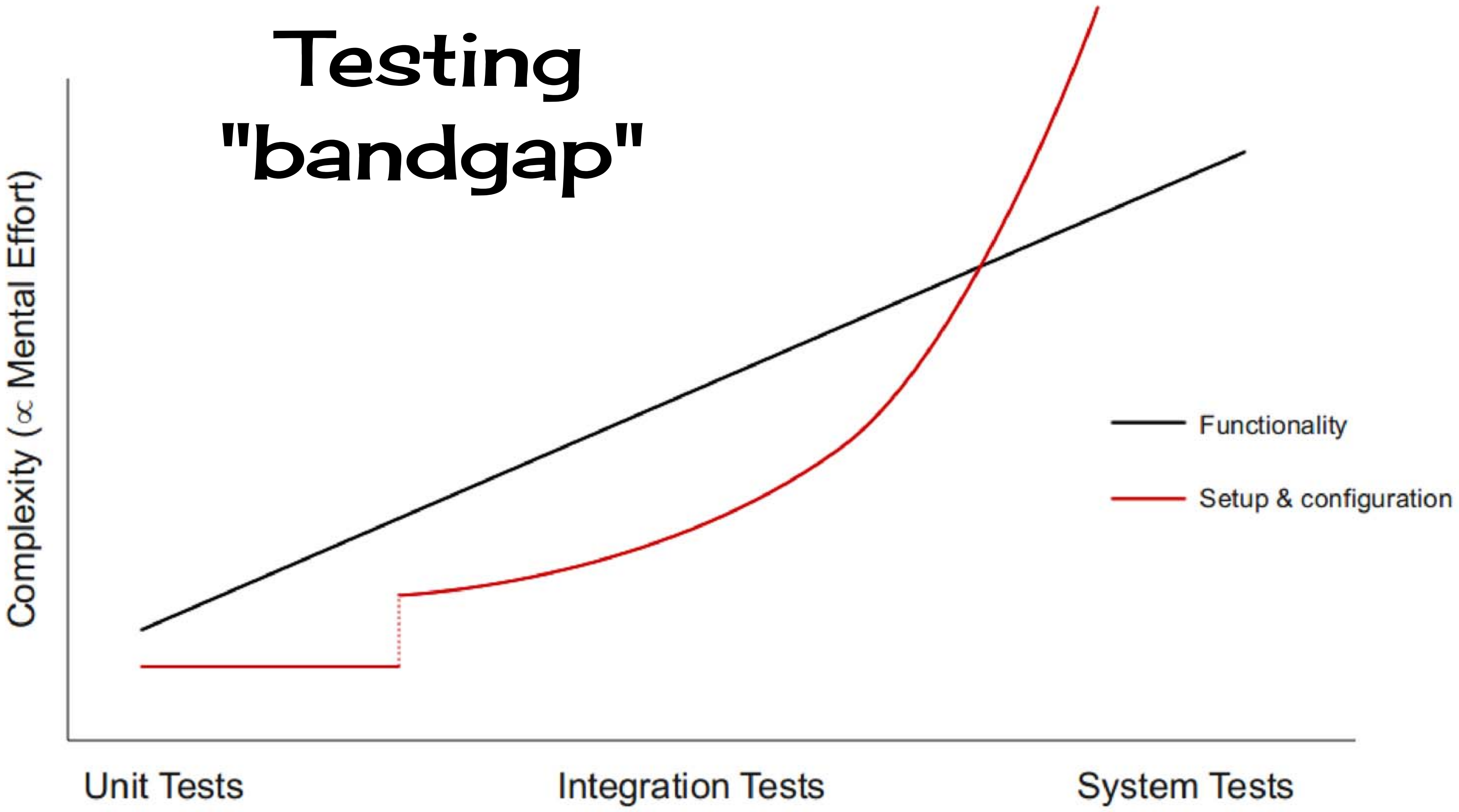

What's the worst that could happen?


#fail

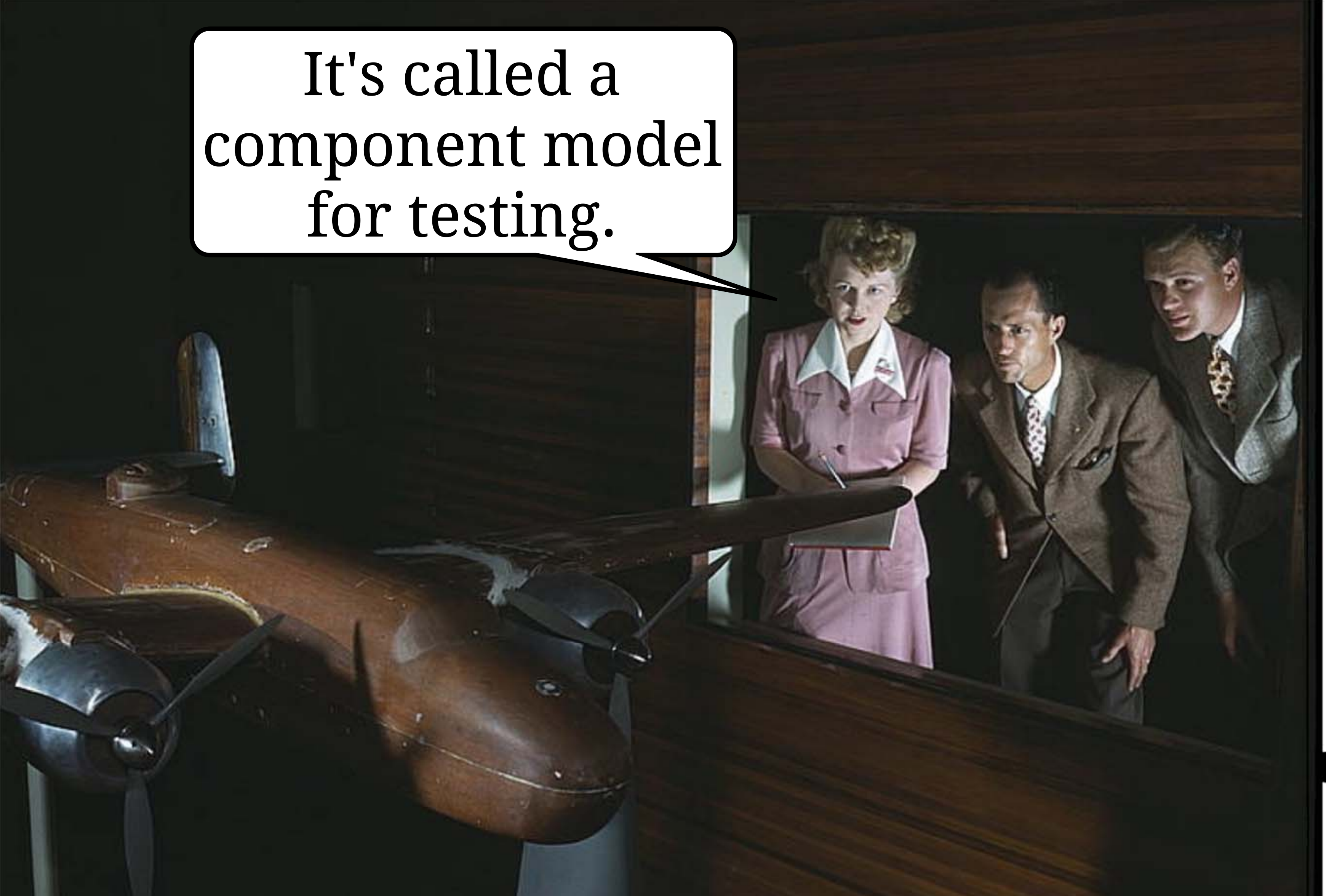

Two words:
intended
environment


We've got to *bridge* this gap.

Testing "bandgap"



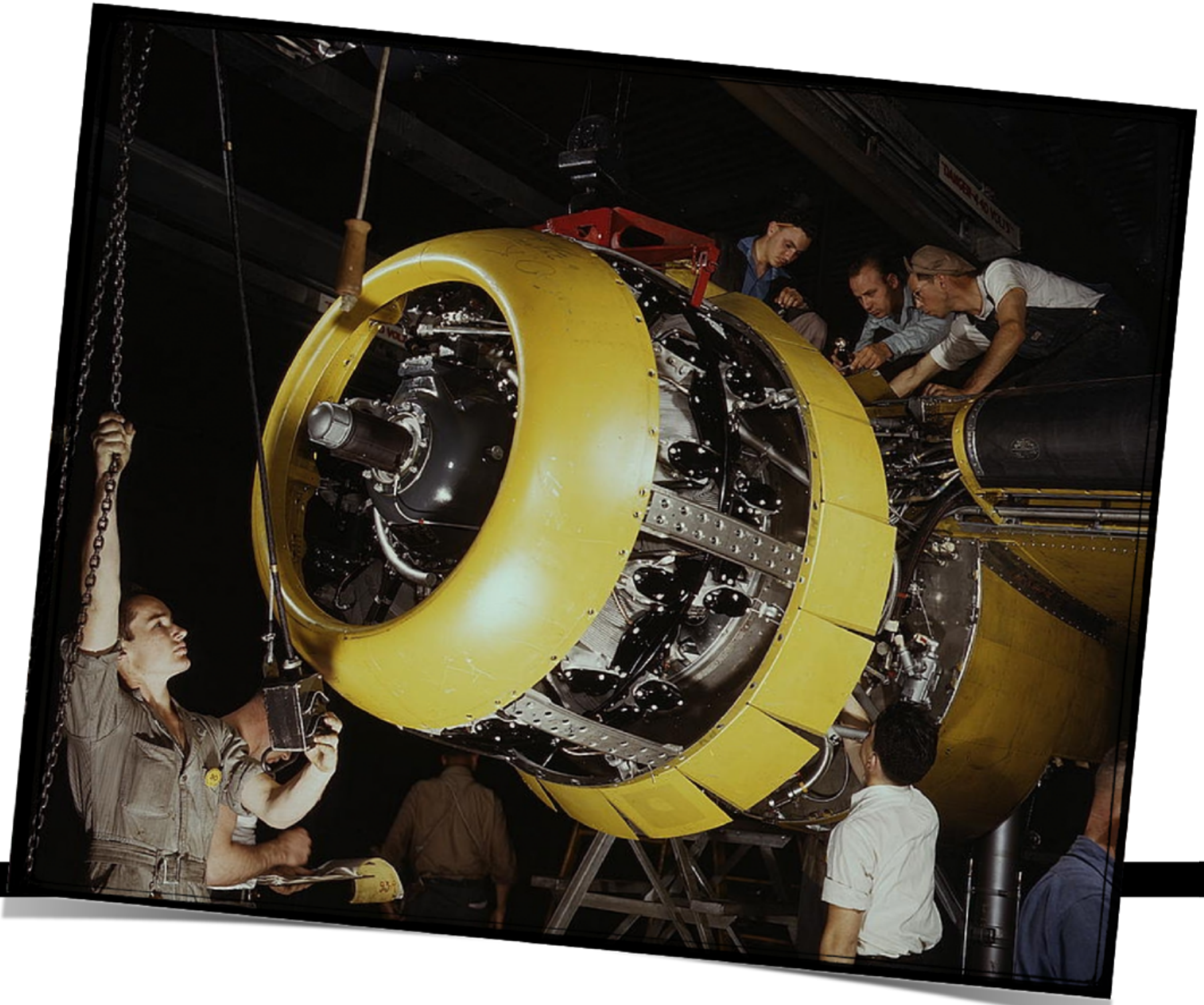
What would help us write tests more efficiently?


It's called a component model for testing.

Adheres to a
STANDARD
programming
model

Encapsulates test logic

Manages the runtime and orchestrates test execution

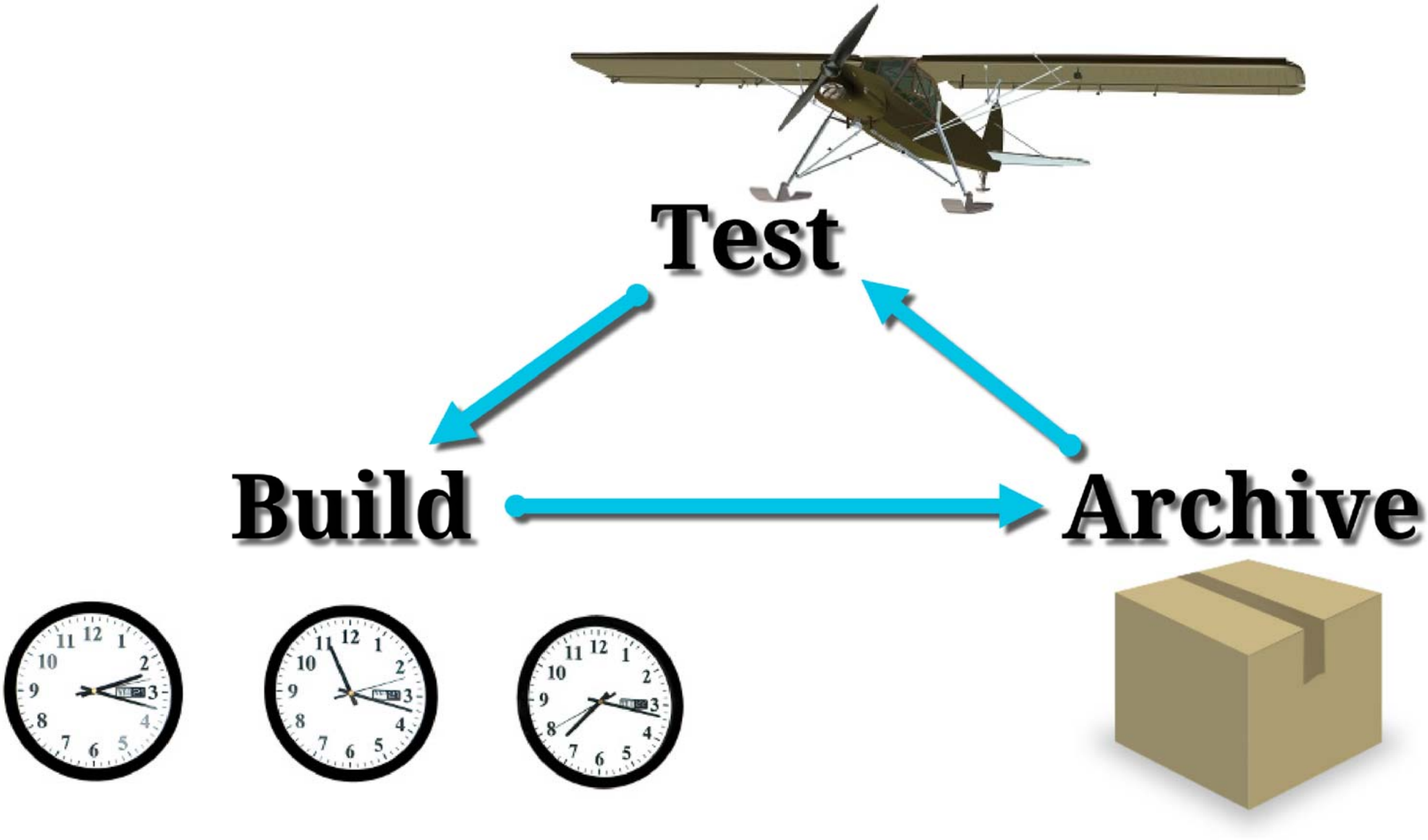


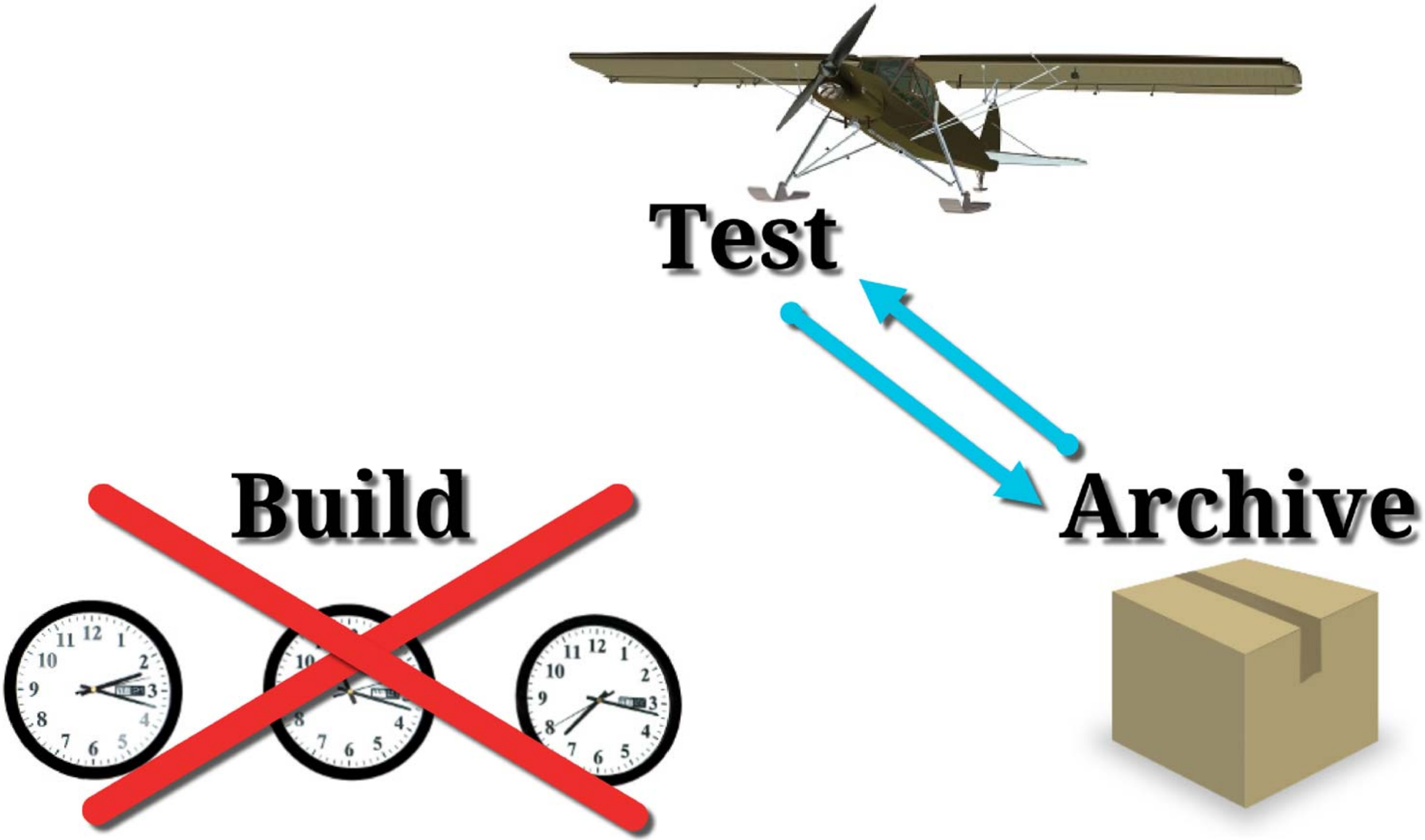
Gives you
POWERFUL
MECHANISMS
for free

Keeps you focused on the test at hand

How can we speed up integration testing?


Skip the build.


Test
Build
Archive


Test
Build
Archive

SHRINKWRAP

http://jboss.org/shrinkwrap

SHRINKWRAP

An API for creating archives (e.g., JARs, WARs and EARs).

...in Java! ^

Andrew Rubringer

@alrubringer

```java
JavaArchive jar = ShrinkWrap.create(JavaArchive.class, "beans.jar")
        .addClasses(Greeter.class, GreeterBean.class)
        .addAsManifestResource(EmptyAsset.INSTANCE, "beans.xml");
```

Simple and fluent API

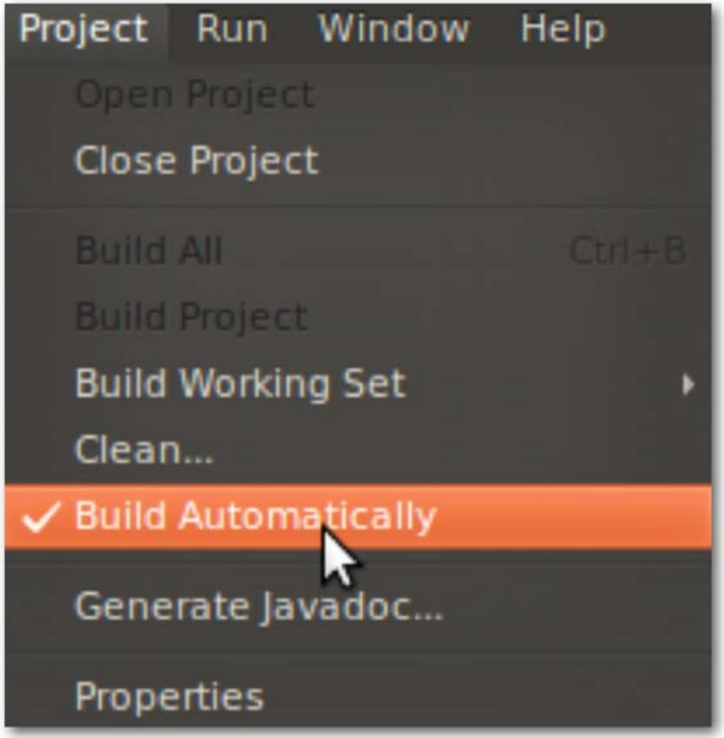
* So what can ShrinkWrap do for us?

Skip the build!

```
System.out.println(jar.toString(true));
```

```
/META-INF
/META-INF/beans.xml
/org
/org/example
/org/example/Greeter.class
/org/example/GreeterBean.class
```

Incremental IDE compilation



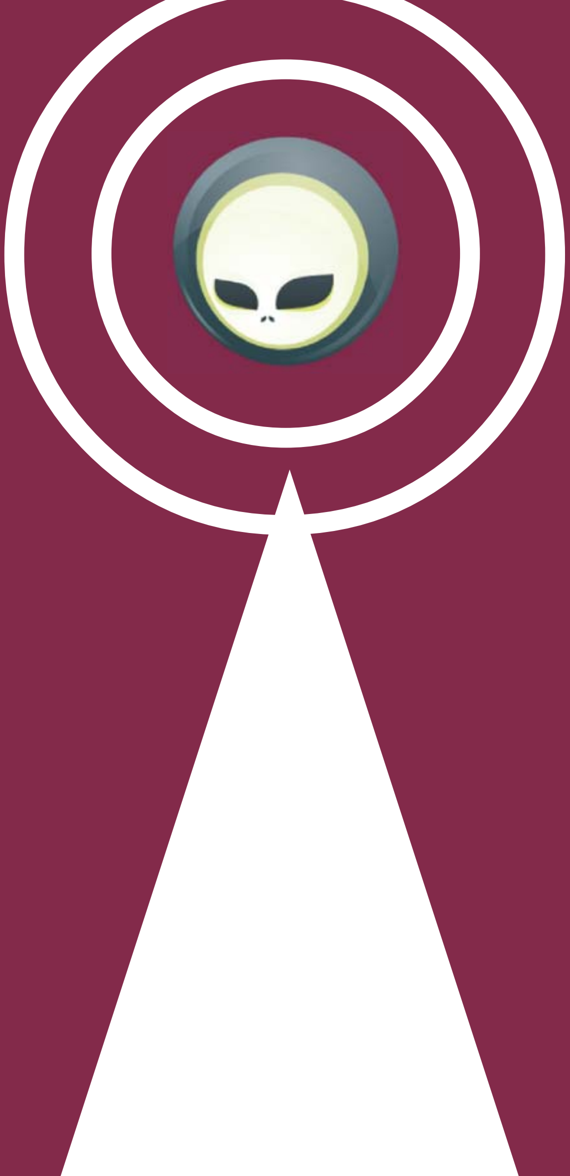
Micro deployments

```
.addClasses(Greeter.class, GreeterBean.class)
   .addClasses(Greeter.class, MockGreeter.class)
```

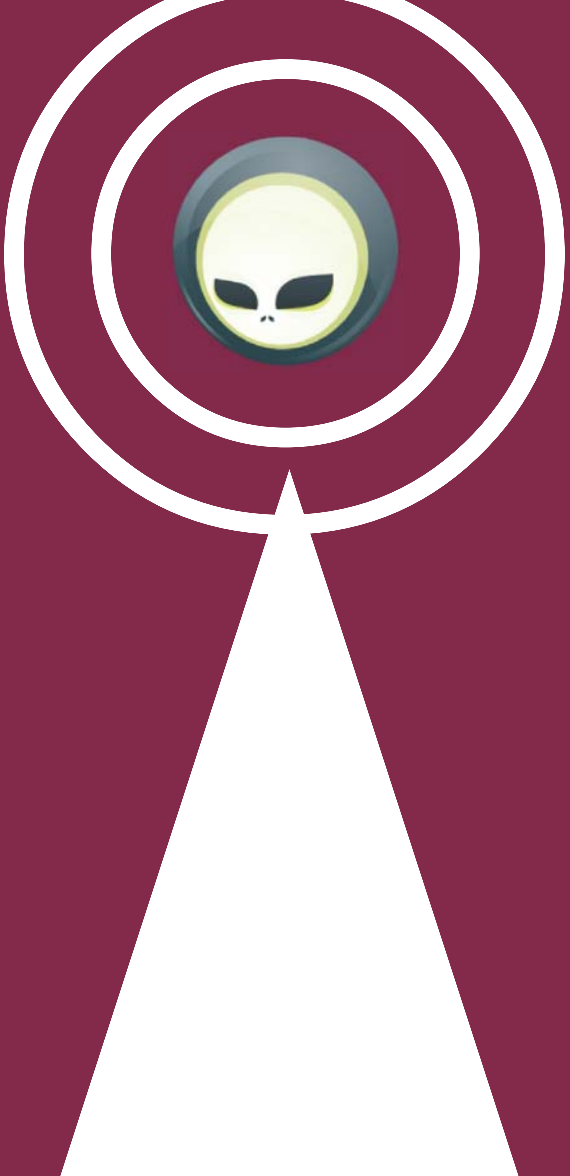
Container deployment adaptors

How can we simplify integration tests?

We interupt this workshop to bring you breaking news!



We've picked up
an alien signal
in Oslo!



It's an ultimatium:

Don't mock me.


Hm. Test in container?

http://jboss.org/arquillian

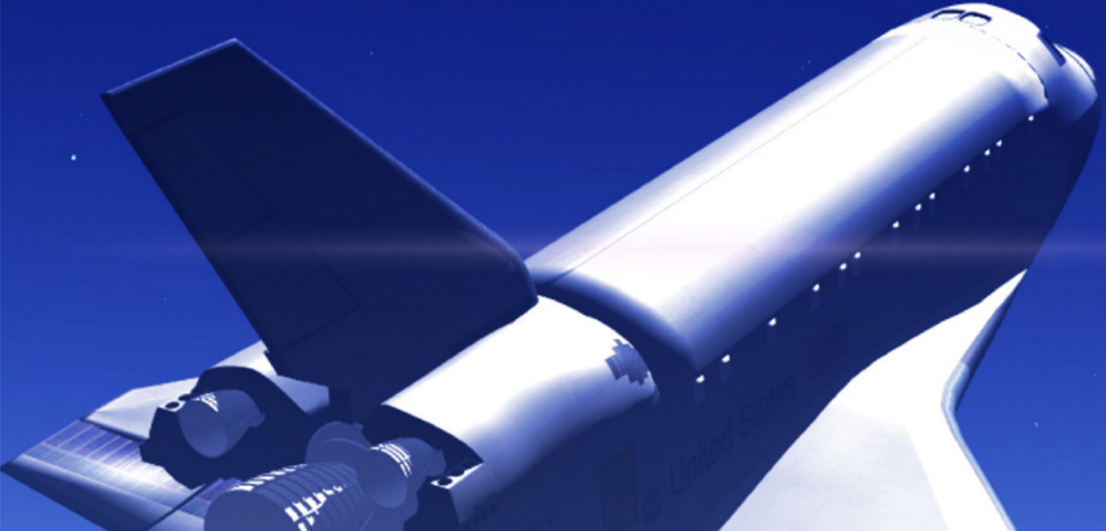
arquillian

A container-oriented testing framework

Aslak Knutsen

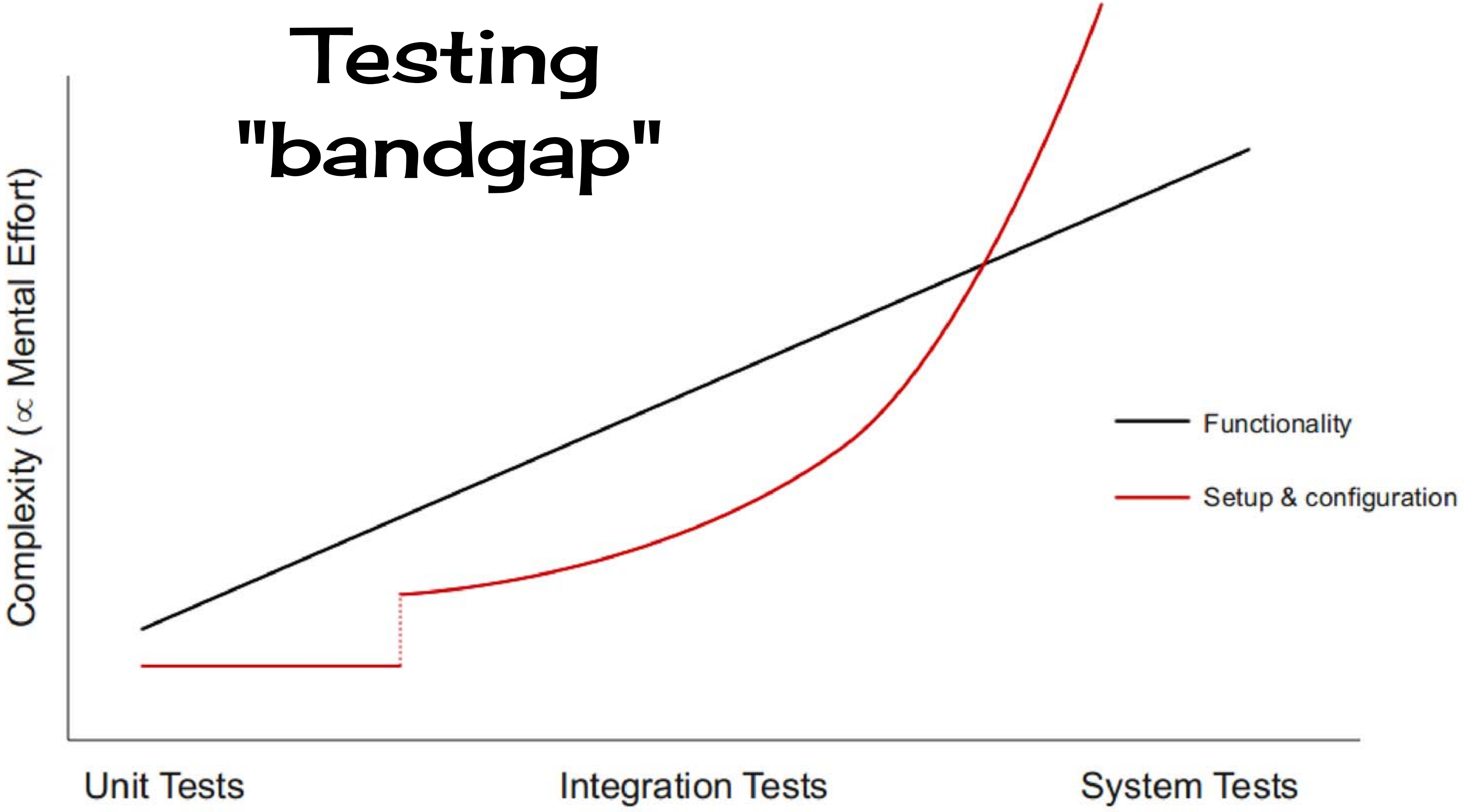
@aslakknutsen

Bring your test to the runtime...

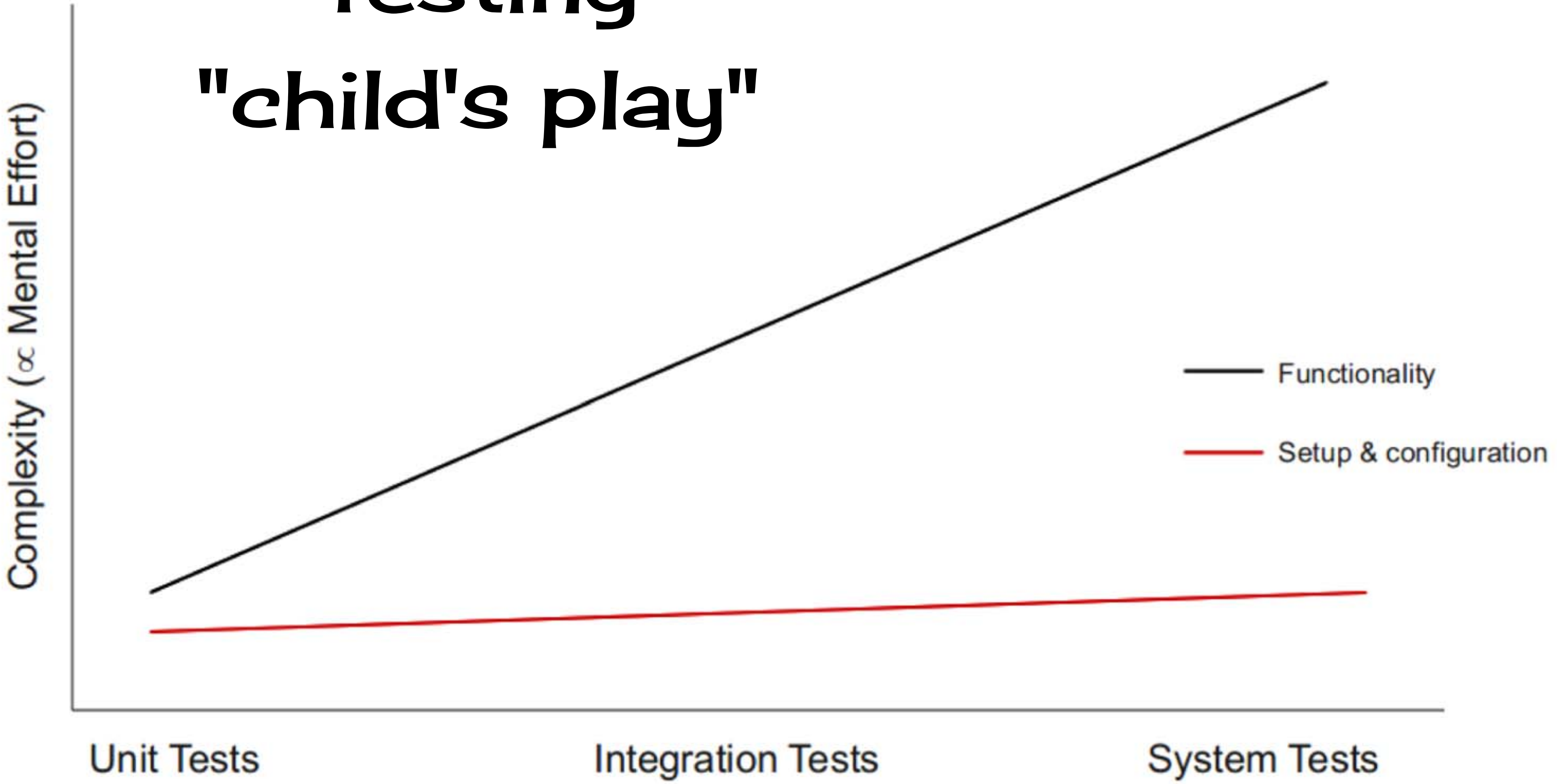


...instead of managing the runtime from your test.

Testing "bandgap"



Testing "child's play"

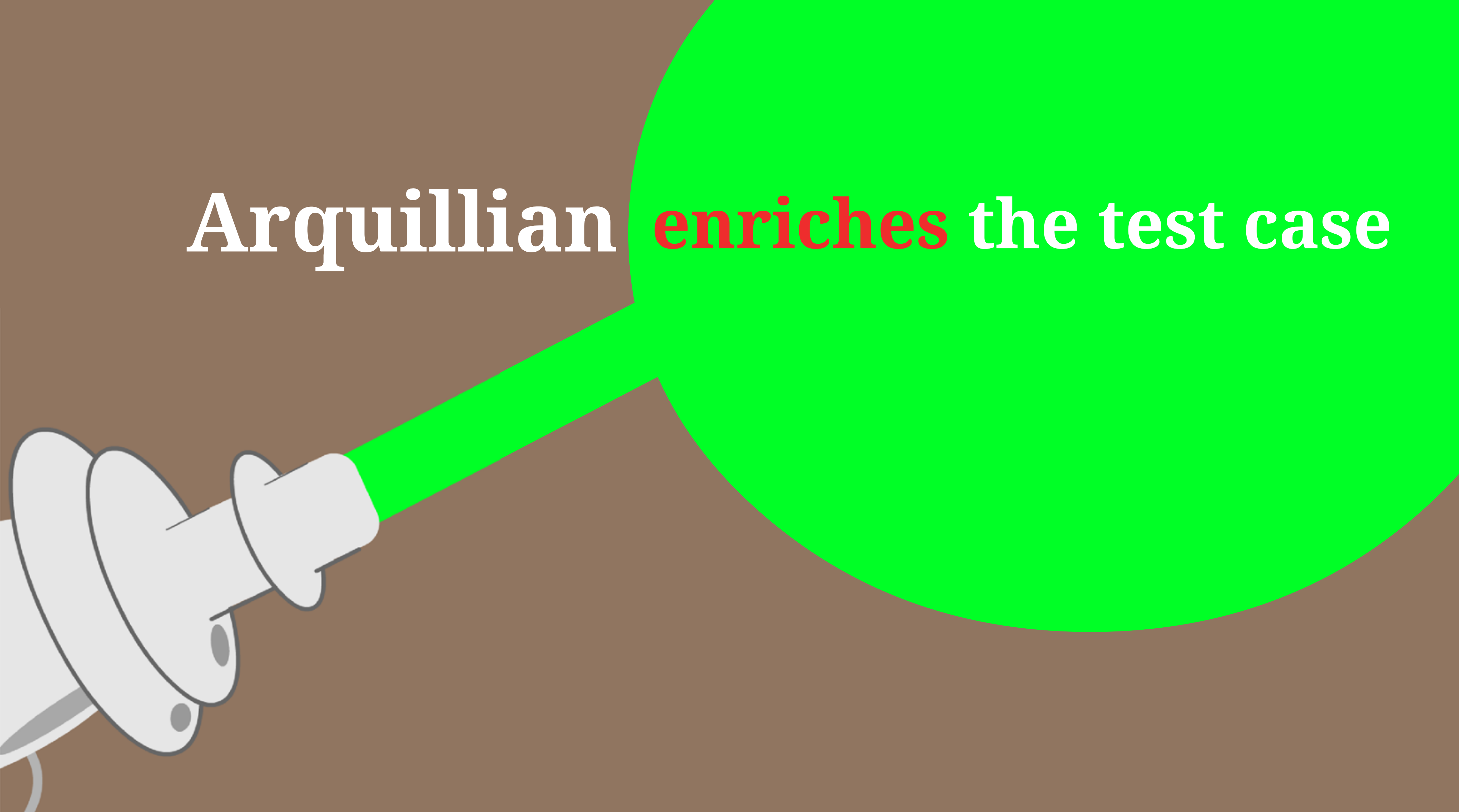


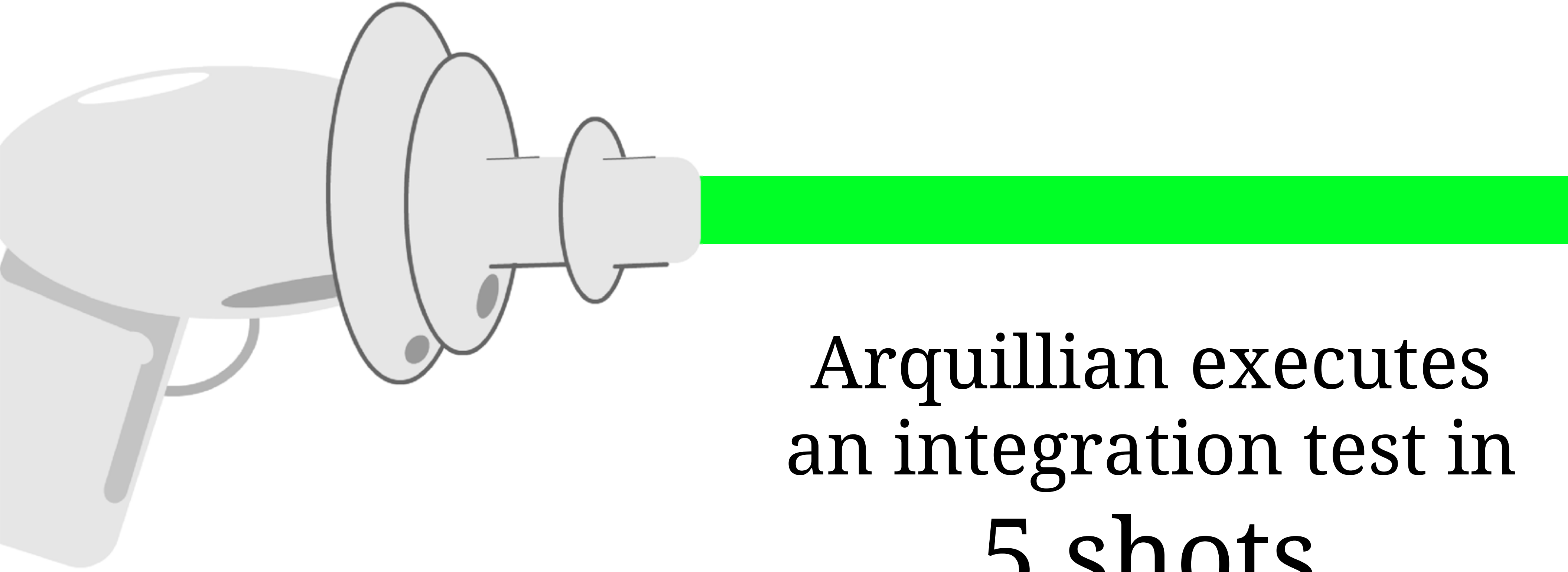
Arquillian
enables

portable integration tests
for enterprise applications

the container's lifecycle

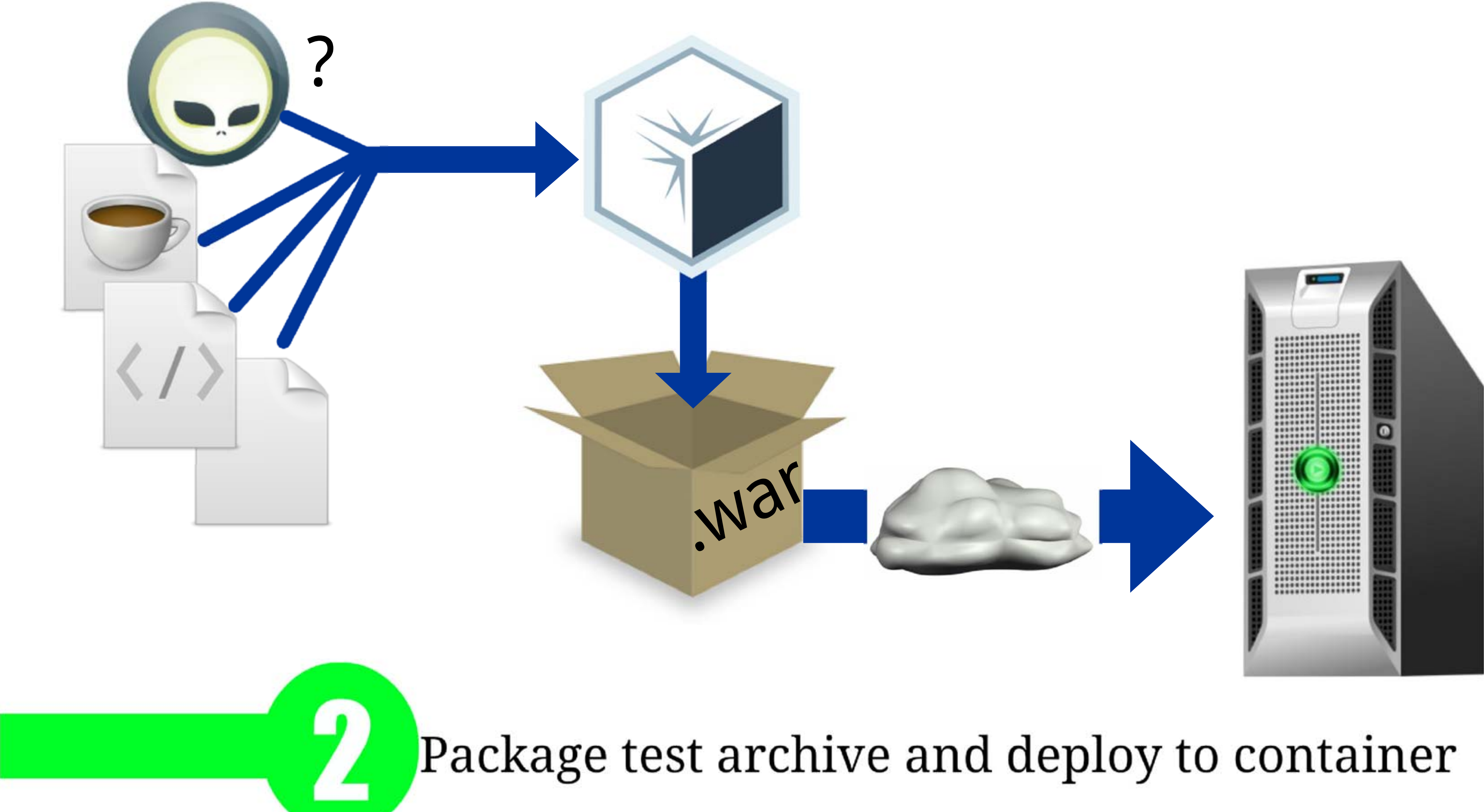
Arquillian
manages


Arquillian enriches the test case



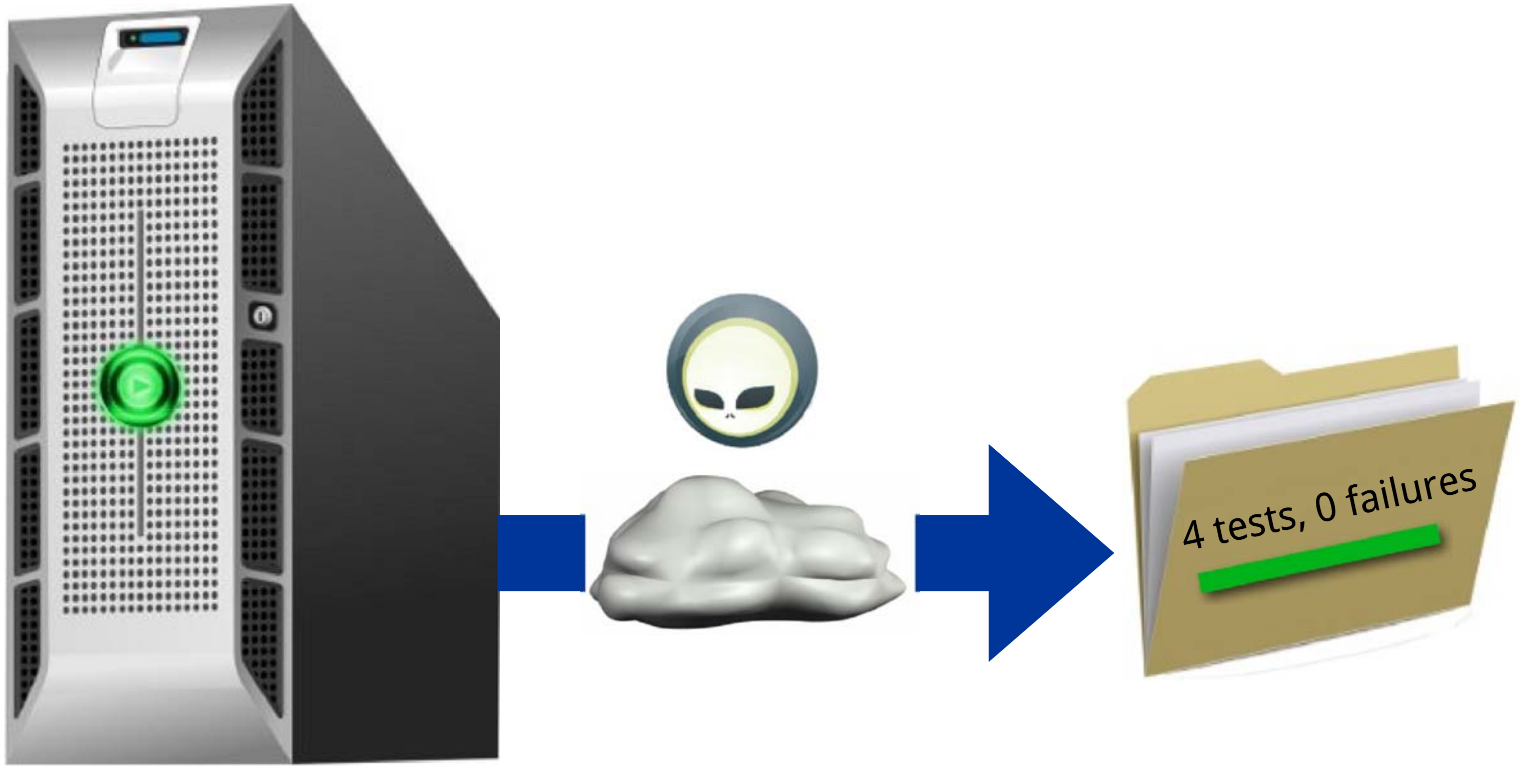
Arquillian executes
an integration test in
5 shots

1 Start or connect to a container



Package test archive and deploy to container

3 Run test in-container



4 Capture and report results

5 Undeploy test archive and stop or disconnect from container

Can we put these ideas into practice?


Prove it.

Experience Ike

```
public class GreeterTestCase {









}
```

```java
@RunWith(Arquillian.class)
public class GreeterTestCase {









}
```

```java
@RunWith(Arquillian.class)
public class GreeterTestCase {
    @Deployment
    public static JavaArchive createDeployment() {
        return ShrinkWrap.create(JavaArchive.class)
            .addClass(Greeter.class)
            .addAsManifestResource(EmptyAsset.INSTANCE, "beans.xml");
    }




}
```

```java
@RunWith(Arquillian.class)
public class GreeterTestCase {
    @Deployment
    public static JavaArchive createDeployment() {
        return ShrinkWrap.create(JavaArchive.class)
            .addClass(Greeter.class)
            .addAsManifestResource(EmptyAsset.INSTANCE, "beans.xml");
    }

    @Inject Greeter greeter;







}
```

```java
@RunWith(Arquillian.class)
public class GreeterTestCase {
    @Deployment
    public static JavaArchive createDeployment() {
        return ShrinkWrap.create(JavaArchive.class)
            .addClass(Greeter.class)
            .addAsManifestResource(EmptyAsset.INSTANCE, "beans.xml");
    }

    @Inject Greeter greeter;

    @Test
    public void shouldBeAbleToInvokeBean() throws Exception {
        Assert.assertEquals("Hello, Earthlings", greeter.greet("Earthlings"));
    }
}
```

Unit test framework

■ ■ ■

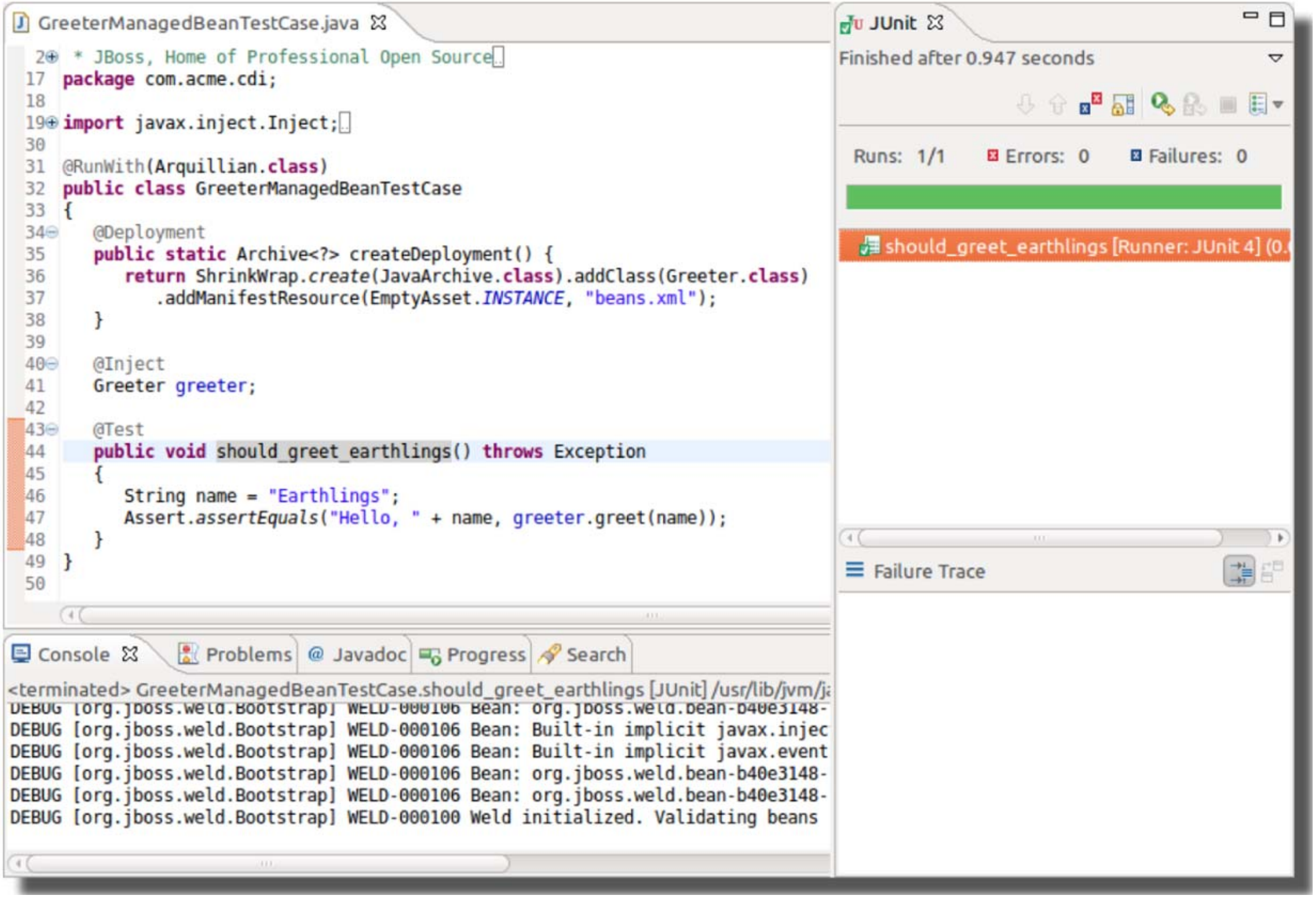
Arquillian unit test framework adapter

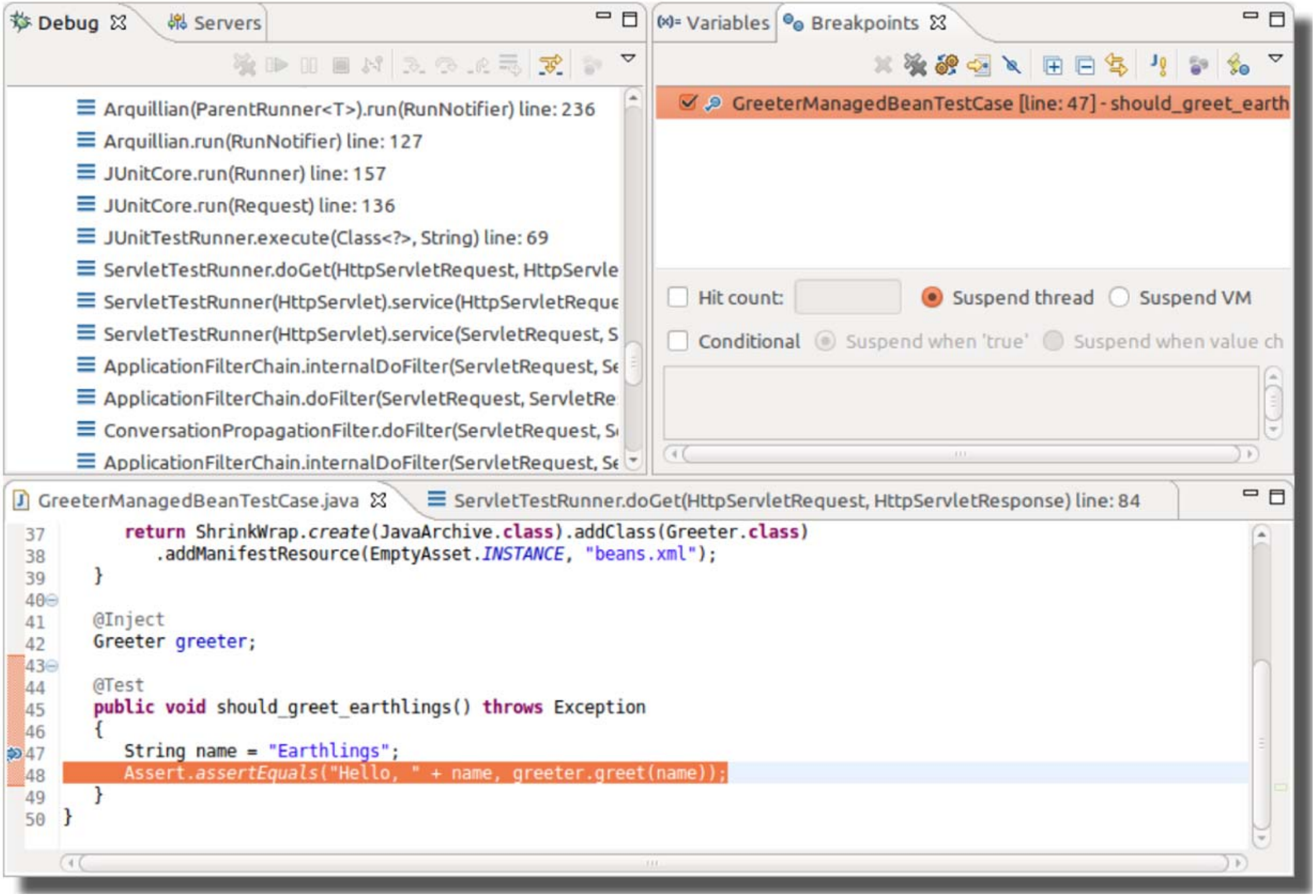
■ ■ ■

Arquillian container adapter

■ ■ ■

Container client or runtime


GreeterManagedBeanTestCase.java
* JBoss, Home of Professional Open Source
package com.acme.cdi;
import javax.inject.Inject;
@RunWith(Arquillian.class)
public class GreeterManagedBeanTestCase
{
@Deployment
public static Archive<?> createDeployment() {
return ShrinkWrap.create(JavaArchive.class).addClass(Greeter.class)
.addManifestResource(EmptyAsset.INSTANCE, "beans.xml");
}
@Inject
Greeter greeter;
@Test
public void should_greet_earthlings() throws Exception
{
String name = "Earthlings";
Assert.assertEquals("Hello, " + name, greeter.greet(name));
}
}
JUnit
Finished after 0.947 seconds
Runs: 1/1
Errors: 0
Failures: 0
should_greet_earthlings [Runner: JUnit 4] (0.
Failure Trace
Console
Problems
Javadoc
Progress
Search
<terminated> GreeterManagedBeanTestCase.should_greet_earthlings [JUnit] /usr/lib/jvm/j
DEBUG [org.jboss.weld.Bootstrap] WELD-000106 Bean: Built-in implicit javax.injec
DEBUG [org.jboss.weld.Bootstrap] WELD-000106 Bean: Built-in implicit javax.event
DEBUG [org.jboss.weld.Bootstrap] WELD-000106 Bean: org.jboss.weld.bean-b40e3148-
DEBUG [org.jboss.weld.Bootstrap] WELD-000106 Bean: org.jboss.weld.bean-b40e3148-
DEBUG [org.jboss.weld.Bootstrap] WELD-000100 Weld initialized. Validating beans


Debug
Servers
Arquillian(ParentRunner<T>).run(RunNotifier) line: 236
Arquillian.run(RunNotifier) line: 127
JUnitCore.run(Runner) line: 157
JUnitCore.run(Request) line: 136
JUnitTestRunner.execute(Class<?>, String) line: 69
ServletTestRunner.doGet(HttpServletRequest, HttpServle
ServletTestRunner(HttpServlet).service(HttpServletReque
ServletTestRunner(HttpServlet).service(ServletRequest, S
ApplicationFilterChain.internalDoFilter(ServletRequest, Se
ApplicationFilterChain.doFilter(ServletRequest, ServletRe
ConversationPropagationFilter.doFilter(ServletRequest, S
ApplicationFilterChain.internalDoFilter(ServletRequest, Se
Variables
Breakpoints
GreeterManagedBeanTestCase [line: 47] - should_greet_earth
Hit count:
Suspend thread
Suspend VM
Conditional
Suspend when 'true'
Suspend when value ch
GreeterManagedBeanTestCase.java
ServletTestRunner.doGet(HttpServletRequest, HttpServletResponse) line: 84
return ShrinkWrap.create(JavaArchive.class).addClass(Greeter.class)
.addManifestResource(EmptyAsset.INSTANCE, "beans.xml");
}
@Inject
Greeter greeter;
@Test
public void should_greet_earthlings() throws Exception
{
String name = "Earthlings";
Assert.assertEquals("Hello, " + name, greeter.greet(name));
}
}

“With tools like Arquillian, Shrinkwrap, [and] in-memory databases, [we] are busting through the old barriers of terms like unit and integration tests. Perhaps we need a new term for them: "real tests" or "useful tests" are two possibilities that come to mind.”

- Brian Leathem

- Less (boilerpate) test code
- As much or as little integration as you need
- Looks like a unit test, but executes in a true environment
- Same test can be run in multiple environments
- It's a *learning* environment

What's on the horizon for enterprise testing?

jclouds
multi - cloud library

Infinispan

JSFUnit

Container	Mode	Compliance	Artifact Id
JBoss AS 5.0	remote	Java EE 5	arquillian-jbossas-remote-5
JBoss AS 5.1	remote	Java EE 5	arquillian-jbossas-remote-5.1
JBoss AS 5.1	managed	Java EE 5	arquillian-jbossas-managed-5.1
JBoss AS 6.0	remote	Java EE 6	arquillian-jbossas-remote-6
JBoss AS 6.0	managed	Java EE 6	arquillian-jbossas-managed-6
JBoss AS 6.0	embedded	Java EE 6	arquillian-jbossas-embedded-6
GlassFish 3.0	remote	Java EE 6	arquillian-glassfish-remote-3
GlassFish 3.0	embedded	Java EE 6	arquillian-glassfish-embedded-3
GlassFish 3.1	remote	Java EE 6	arquillian-glassfish-remote-3.1
GlassFish 3.1	embedded	Java EE 6	arquillian-glassfish-embedded-3.1
Tomcat 6	embedded	Servlet 2.5	arquillian-tomcat-embedded-6
Jetty 6.1	embedded	Servlet 2.5	arquillian-jetty-embedded-6.1
Jetty 7	embedded	Servlet ~3.0	arquillian-jetty-embedded-7
Weld SE 1.0	embedded	CDI 1.0	arquillian-weld-se-embedded-1
Weld EE 1.1	embedded	CDI 1.0	arquillian-weld-ee-embedded-1.1
IronJacamar 1.0	embedded	JCA 1.6	org.jboss.ironjacamar:ironjacamar-embedded-arquillian
OpenEJB 3.1	embedded	EJB 3.0	arquillian-openejb-embedded-3.1
OpenWebBeans 1.0	embedded	CDI 1.0	arquillian-openwebbeans-embedded-1
JBoss OSGi 1.0	embedded	OSGi 4.2	arquillian-osgi-embedded-4.2

Calling all containers!

#arquillian

Write for us!

github
SOCIAL CODING

Fork us!

#jbosstesting
freenode

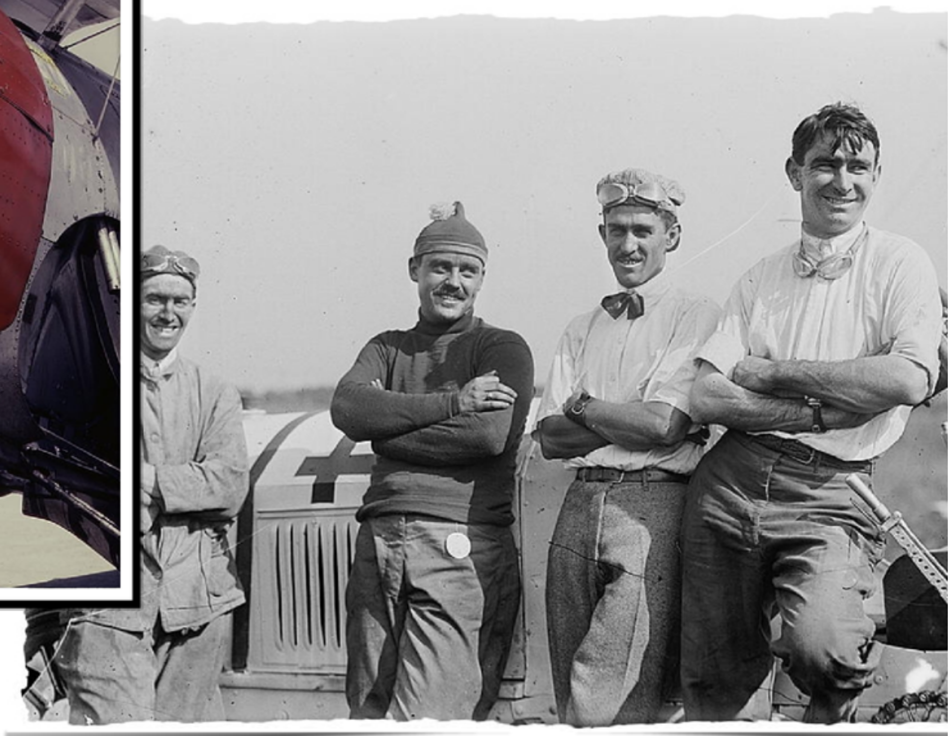
Chat with us!

ShrinkWrap
http://jboss.org/shrinkwrap

arquillian
http://jboss.org/arquillian


Go forth and test!
build free and in
container!

Historical photographs courtesy of the Library of Congress



www.flickr.com/photos/library_of_congress/