Advanced Java Testing

What's next?

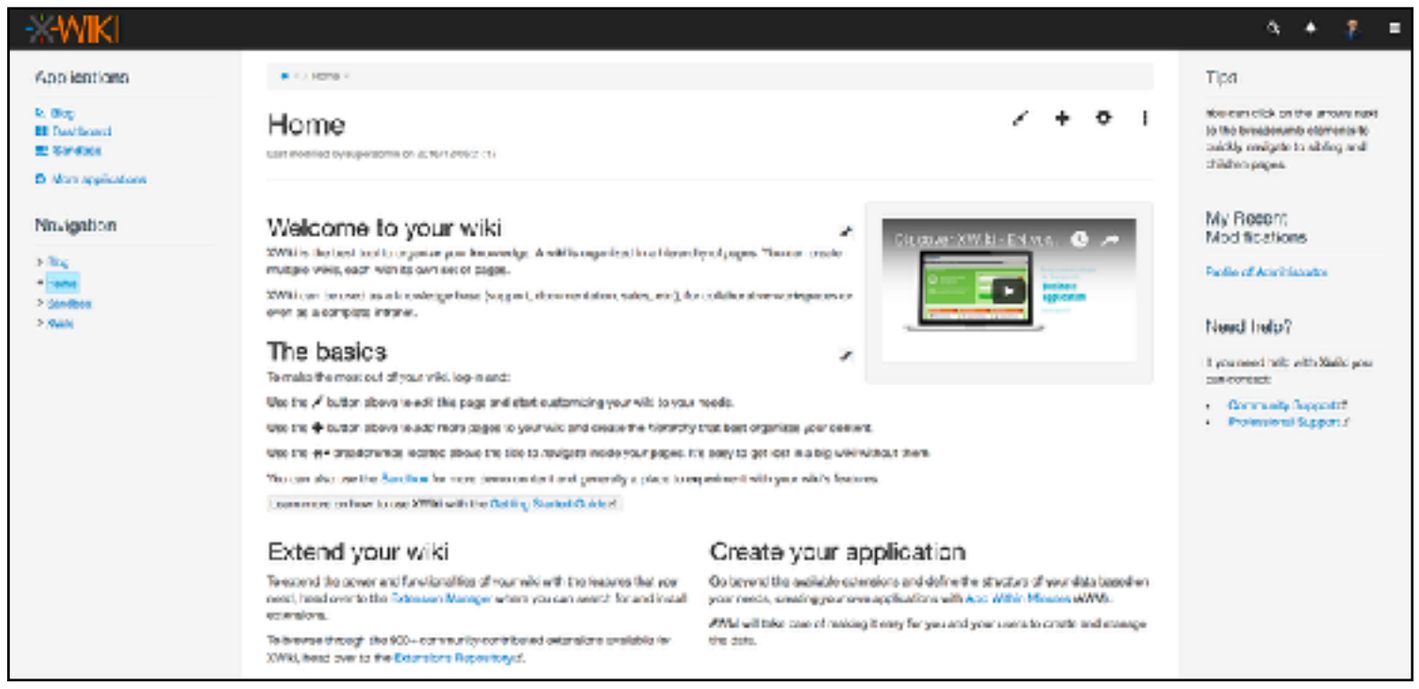
Vincent Massol, February 2018

Agenda

- Context & Current status quo
- Coverage testing
- Testing for backward compatibility
- Mutation testing
- Environment testing

Context: XWiki

- Open source wiki
- 14 years
- 10-15 active committers

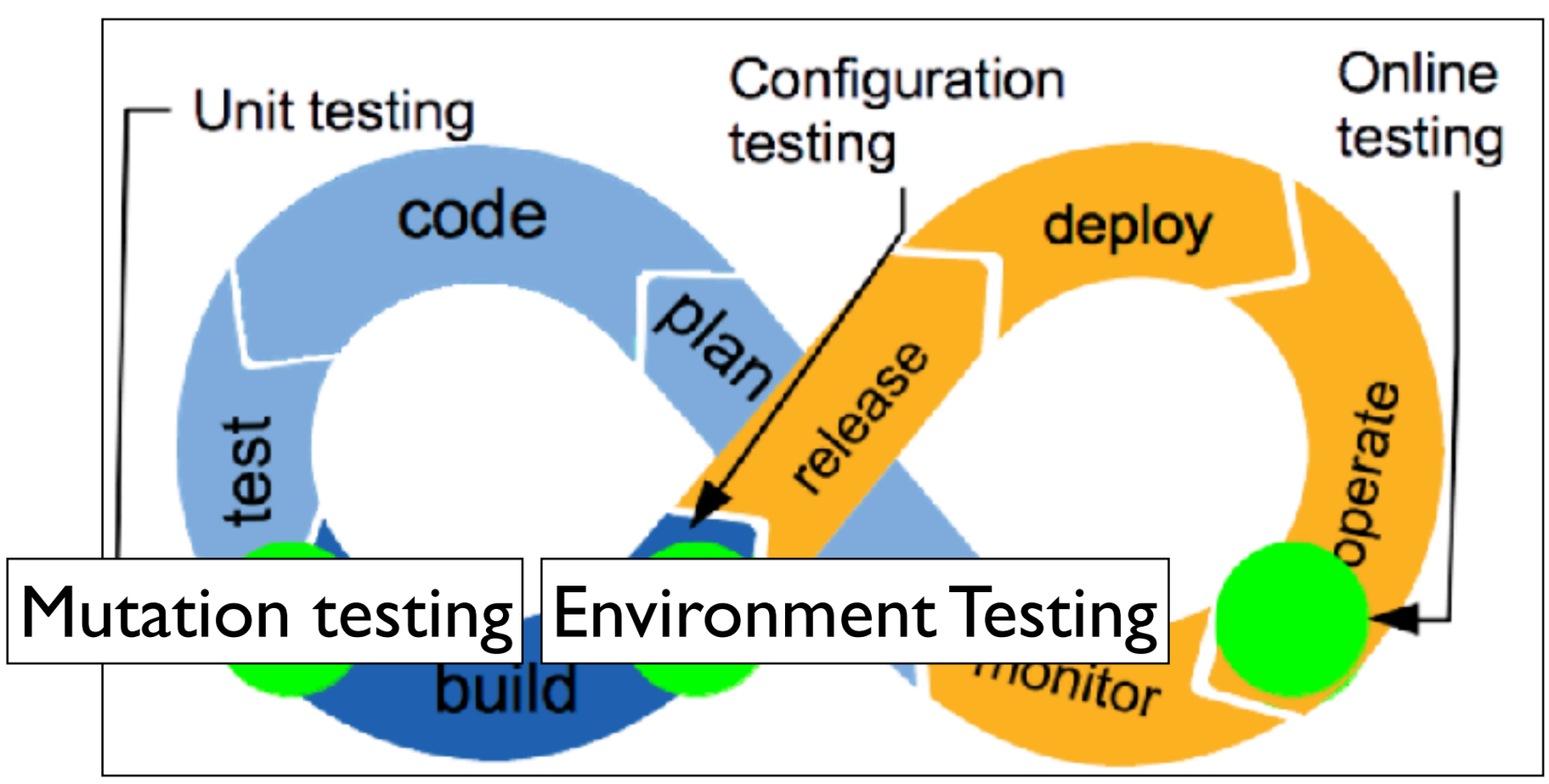


http://xwiki.org

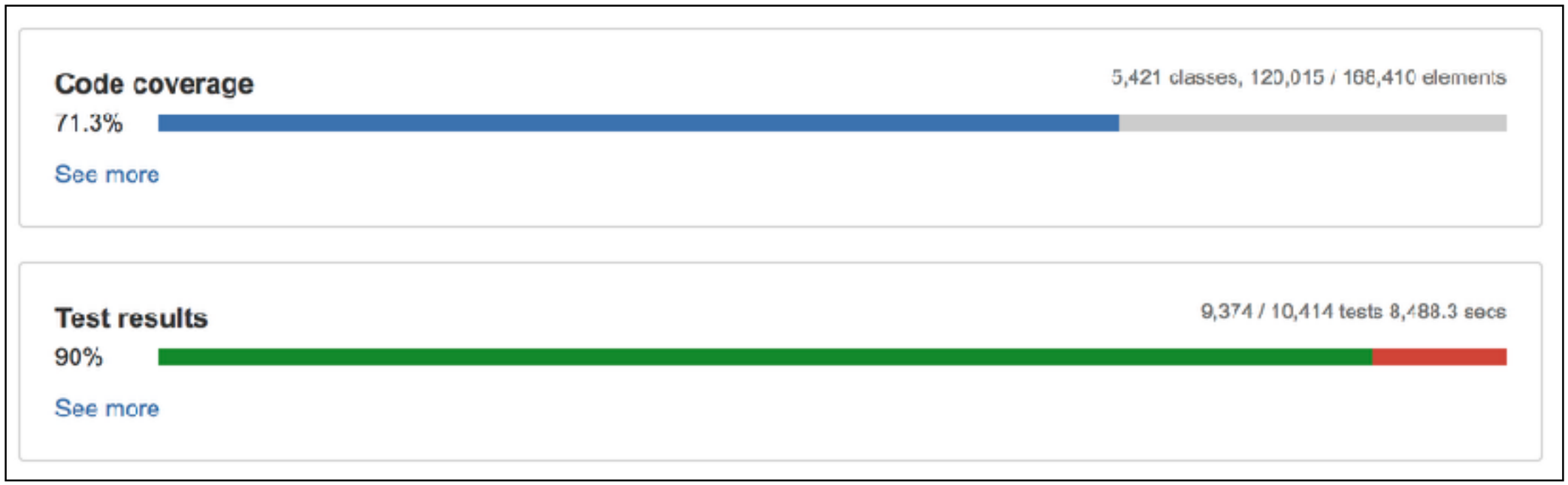
- Very extensible, scripting in wiki pages
- Platform for developing ad-hoc web applications
- Strong build practices using Maven and lots of “Quality” plugins
- Using Jenkins & custom pipeline library for the CI

Context: STAMP

- Automatic Test Amplification
- XWiki SAS participating
- Experiment on XWiki project



Current Testing Status



- 10414 automated tests (in 2.5 hours):
  - Unit tests (using Mockito)
  - Integration tests (using Mockito)
  - Functional (UI) tests (using Selenium/Webdriver)

New questions

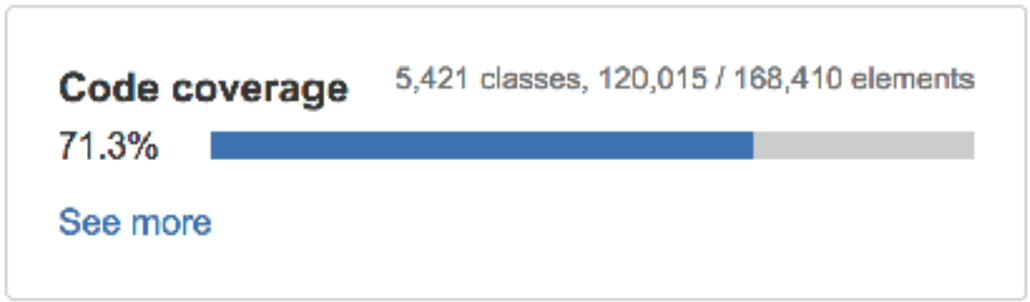
- Are my tests testing enough? Coverage 
- How can I prevent breaking my users (when I expose some APIs)? Backward compatibility 
- How good are my tests? Mutation testing 
- Do my software work in various setups? Environment testing 

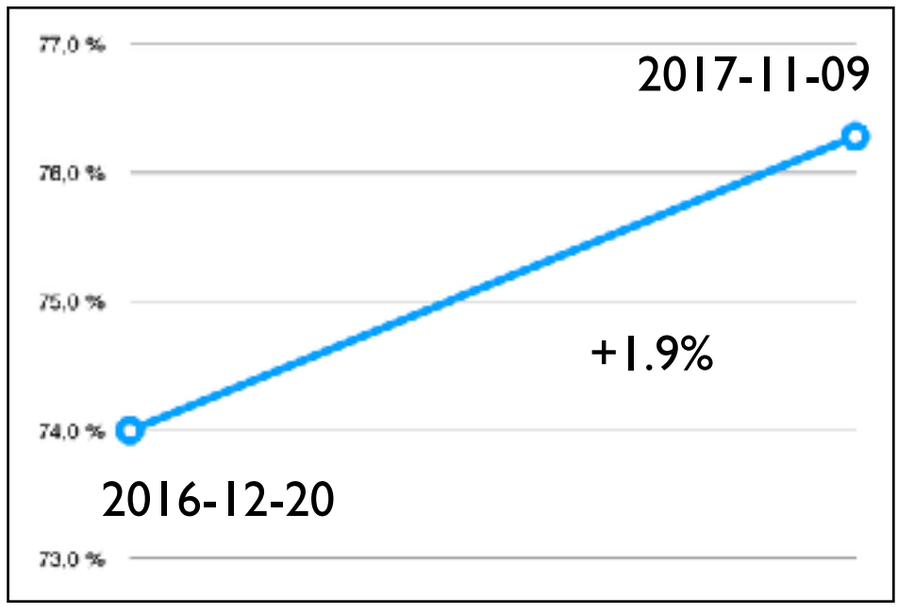
 = in place w/ strategy  = in progress


Test Coverage

Of course TPC is not panacea. You could have 100% and app not working. Also need functional tests. Aim for 80%.

- Using Jacoco and Clover
- Strategy - "Ratchet effect":
  - Each Maven module has a threshold
    - Jacoco Maven plugin fails if new code has less coverage than before in %
    - Dev is allowed to increase threshold
- Global Clover TPC computed automatically once per month on Jenkins for all repos combined



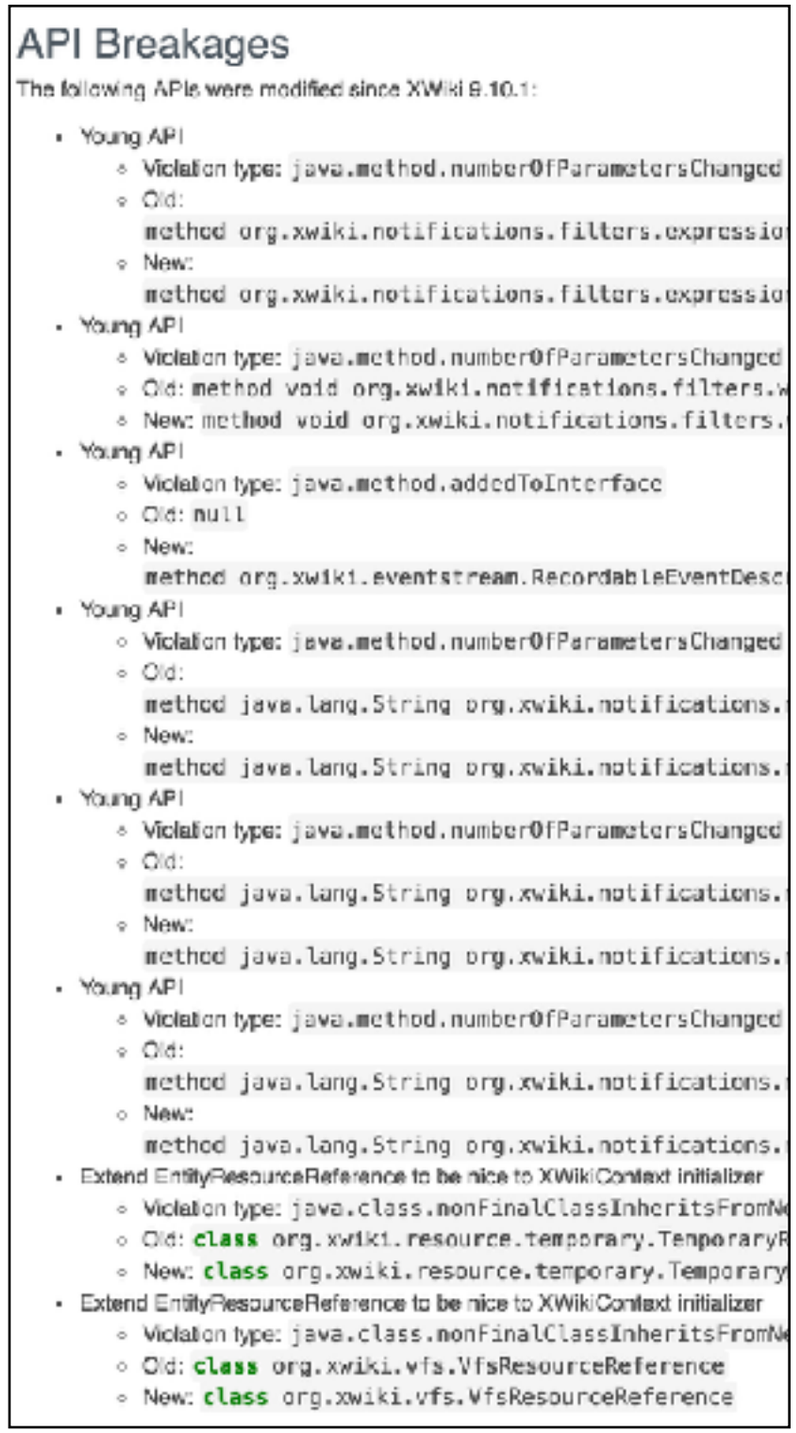


Harmonized TPC

Source: http://massol.myxwiki.org/xwiki/bin/view/Blog/ComparingCloverReports

Backward Compatibility

- For APIs (and SPIs)
- Using the Revapi Maven plugin
  - Supports source and binary compatibility
- Strategy:
  - Break the build on backward compatibility violations
  - Add ignores in pom.xml if really needed and ok
  - Add ignore list in release notes to warn users of your APIs



API Breakages

The following APIs were modified since XWiki 8.10.1:

- Young API
  - Violation type: `java.method.numberOfParametersChanged`
  - Old:
    `method org.xwiki.notifications.filters.expressio`
  - New:
    `method org.xwiki.notifications.filters.expressio`
- Young API
  - Violation type: `java.method.numberOfParametersChanged`
  - Old: `method void org.xwiki.notifications.filters.`
  - New: `method void org.xwiki.notifications.filters.`
- Young API
  - Violation type: `java.method.addedToInterface`
  - Old: `null`
  - New:
    `method org.xwiki.eventstream.RecordableEventDesc`
- Young API
  - Violation type: `java.method.numberOfParametersChanged`
  - Old:
    `method java.lang.String org.xwiki.notifications.`
  - New:
    `method java.lang.String org.xwiki.notifications.`
- Young API
  - Violation type: `java.method.numberOfParametersChanged`
  - Old:
    `method java.lang.String org.xwiki.notifications.`
  - New:
    `method java.lang.String org.xwiki.notifications.`
- Young API
  - Violation type: `java.method.numberOfParametersChanged`
  - Old:
    `method java.lang.String org.xwiki.notifications.`
  - New:
    `method java.lang.String org.xwiki.notifications.`
- Extend EntityResourceReference to be nice to XWikiContext initializer
  - Violation type: `java.class.nonFinalClassInheritsFromN`
  - Old: `class org.xwiki.resource.temporary.TemporaryR`
  - New: `class org.xwiki.resource.temporary.Temporary`
- Extend EntityResourceReference to be nice to XWikiContext initializer
  - Violation type: `java.class.nonFinalClassInheritsFromN`
  - Old: `class org.xwiki.vfs.VfsResourceReference`
  - New: `class org.xwiki.vfs.VfsResourceReference`

Backward Compatibility

Java 8 default methods helped a lot

- Strategy continued:
  - Use @Deprecated annotation

```
public privileged aspect ApiCompatibilityAspect
{
    @Deprecated
    public boolean Api.checkProgrammingRights()
    {
        return this.hasProgrammingRights();
    }
}
```

  - Once no more code uses deprecated API, move it to Legacy module. We don't break backward compatibility!
  - Use AspectJ in Legacy module to generate and aspectified API (JAR)
  - Makes code clean and not able to use the Legacy modules (build-enforced)
  - Distribute the legacy modules

Backward Compatibility

Mention that as a result XWiki extensions and scripts can still run in XWiki even several years after they were released.

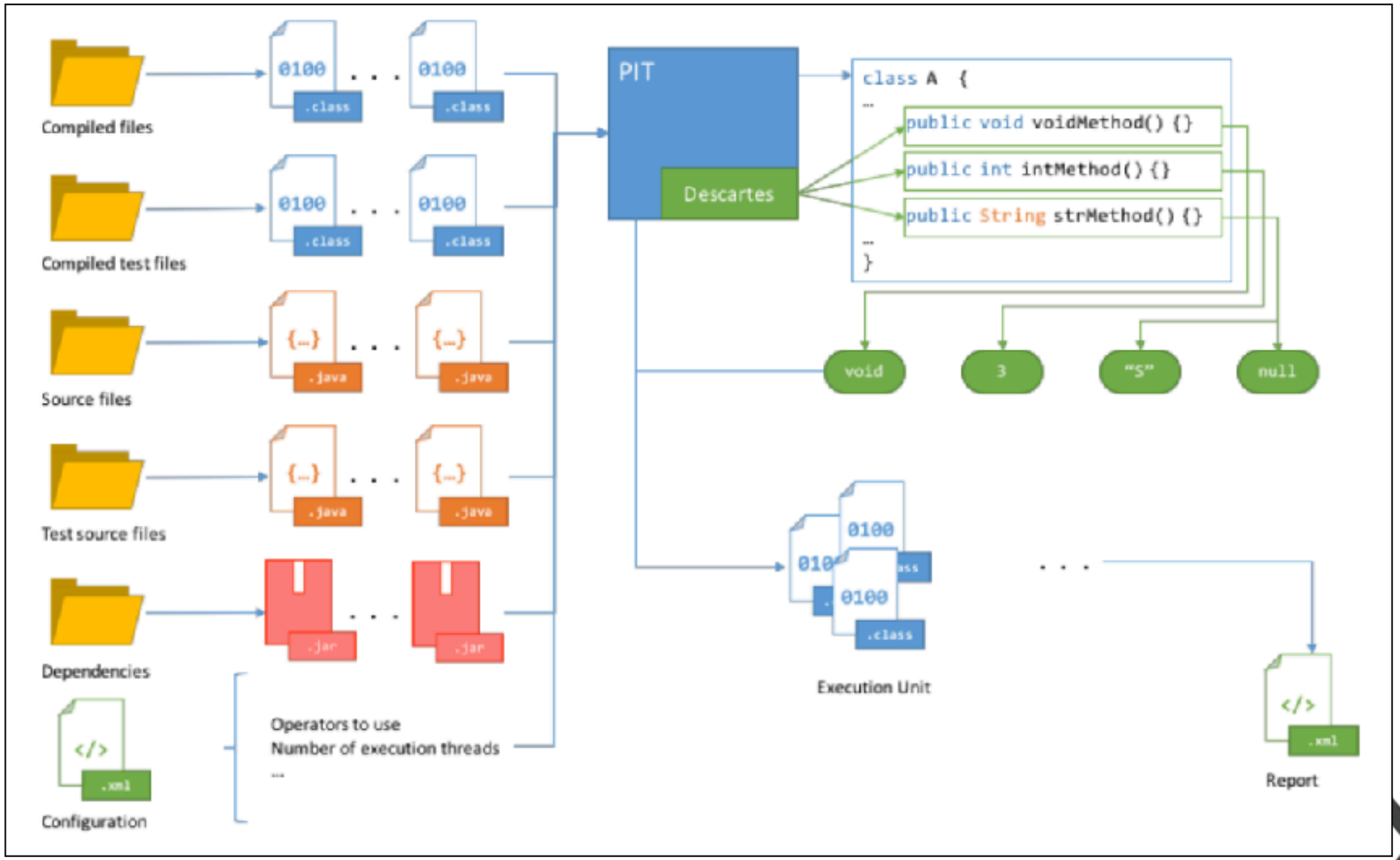
```
/**
 * ...
 * @since 9.7RC1
 */
@Unstable
public List<MacroDescriptor> getMacroDescriptors(Syntax syntax) throws MacroLookupException
{
...
}
```

- Strategy continued:
  - For young APIs, use @Unstable + @Since
  - Enforced by build (to make sure @Since is there). Custom checkstyle rule (or Spoon rule)
  - Max duration is one cycle (i.e. 1 year).
  - Enforced by build (fails the build if beyond).

Mutation Testing

- Using PIT and Descartes
- Concepts
  - Modify code under test (mutants) and run tests
  - Good tests kill mutants
  - Generates a mutation score similar to the coverage %
- Descartes = extreme mutations that execute fast and have high values

Mutation - Descartes



Mutation - Example

Pit Test Coverage Report

Project Summary

Number of Classes	Line Coverage		Mutation Coverage	
14	80%	294/369	71%	110/155

Breakdown by Package

Name	Number of Classes	Line Coverage		Mutation Coverage	
org.xwiki.rendering.internal.macro	4	81%	96/118	76%	42/55
org.xwiki.rendering.internal.transformation.macro	3	83%	85/102	65%	26/40
org.xwiki.rendering.macro	2	87%	48/55	69%	20/29
org.xwiki.rendering.macro.descriptor	4	57%	36/63	55%	11/20
org.xwiki.rendering.transformation	1	94%	29/31	100%	11/11

Report generated by PIT 1.2.3

Mutation - Example

```
106     @Override
107     public boolean equals(Object object)
108     {
109         boolean result;
110
111         // See http://www.technofundo.com/tech/java/equalhash.html for the detail of this algorithm.
112 2       1. equals : All method body replaced by: return true → SURVIVED
113         2. equals : All method body replaced by: return false → KILLED
114         } else {
115             if ((object == null) || (object.getClass() != this.getClass())) {
116                 result = false;
117             } else {
118                 MacroId macroId = (MacroId) object;
119                 result =
120                     (getId() == macroId.getId() || (getId() != null && getId().equals(macroId.getId())))
121                         && (getSyntax() == macroId.getSyntax() || (getSyntax() != null && getSyntax().equals(
122                             macroId.getSyntax())));
123             }
124         }
125         return result;
126     }
127 }
```

```
result =
    (getId() == macroId.getId() || (getId() != null && getId().equals(macroId.getId())))
    && (getSyntax() == macroId.getSyntax() || (getSyntax() != null && getSyntax().equals(
    macroId.getSyntax())));
```

Mutation - Example

```java
@Test
public void testEquality()
{
    MacroId id1 = new MacroId("id", Syntax.XWIKI_2_0);
    MacroId id2 = new MacroId("id", Syntax.XWIKI_2_0);
    MacroId id3 = new MacroId("otherid", Syntax.XWIKI_2_0);
    MacroId id4 = new MacroId("id", Syntax.XHTML_1_0);
    MacroId id5 = new MacroId("otherid", Syntax.
    MacroId id6 = new MacroId("id");
    MacroId id7 = new MacroId("id");

    Assert.assertEquals(id2, id1);
    // Equal objects must have equal hashcode
    Assert.assertTrue(id1.hashCode() == id2.hash

    Assert.assertFalse(id3 == id1);
    Assert.assertFalse(id4 == id1);
    Assert.assertFalse(id5 == id3);
    Assert.assertFalse(id6 == id1);

    Assert.assertEquals(id7, id6);
    // Equal objects must have equal hashcode
    Assert.assertTrue(id6.hashCode() == id7.hash
}
```

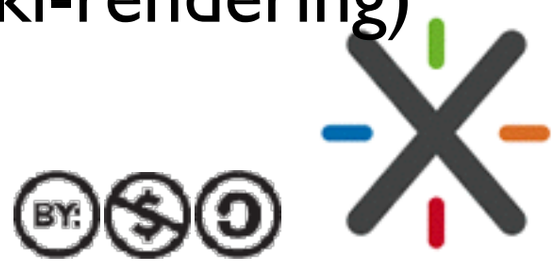
Not testing for inequality!

```java
@Test
public void testEquality()
{
    MacroId id1 = new MacroId("id", Syntax.XWIKI_2_0);
    MacroId id2 = new MacroId("id", Syntax.XWIKI_2_0);
    MacroId id3 = new MacroId("otherid", Syntax.XWIKI_2_0);
    MacroId id4 = new MacroId("id", Syntax.XHTML_1_0);
    MacroId id5 = new MacroId("otherid", Syntax.XHTML_1_0);
    MacroId id6 = new MacroId("id");
    MacroId id7 = new MacroId("id");

    Assert.assertEquals(id2, id1);
    // Equal objects must have equal hashcode
    Assert.assertTrue(id1.hashCode() == id2.hashCode());

    Assert.assertFalse(id3 == id1);
    Assert.assertFalse(id3.equals(id1));
    Assert.assertFalse(id1.equals(id3));

    Assert.assertFalse(id4 == id1);
    Assert.assertFalse(id4.equals(id1));
    Assert.assertFalse(id1.equals(id4));

    Assert.assertFalse(id5 == id3);
    Assert.assertFalse(id5.equals(id1));
    Assert.assertFalse(id1.equals(id5));

    Assert.assertFalse(id6 == id1);
    Assert.assertFalse(id6.equals(id1));
    Assert.assertFalse(id1.equals(id6));

    Assert.assertEquals(id7, id6);
```

Improved thanks to Descartes!

Mutation - Limitations

- Takes time to find interesting things to look at and decide if that's an issue to handle or not. Need better categorisation in report:
  - *Strong pseudo-tested methods*: The worst! No matter what the return values are the tests always fail
  - *Pseudo-tested methods*: Grey area. The tests pass with at least one modified value.
- Multi module support - PITmp
  - Slow on large projects (e.g. 7+ hours just for xwiki-rendering)



Mutation - Strategy

- Work in progress, no feedback yet! 
- Fail the build when the mutation score of a given module is below a defined threshold in the pom.xml
- The idea is that new tests should, in average, be of quality equal or better than past tests.
- Other idea: hook on CI to run it only on modified code/tests.

Ideally: replace coverage check by mutation check(*)

(*) But too slow for now to replace coverage, can be done in addition (in a Maven profile for example, or executed on CI). Timeouts are a problem for example.

Mutation: Going further

- Using DSpot 
  - Uses PIT/Descartes but injects results to generate new tests
    - Adds assertions to existing tests
    - Generate new test methods

Mutation: Dspot Example

Generated test

New test

```
public void escapeAttributeValue2() {
    String escapedText = XMLUtils.escapeAttributeValue("a < a\' && a\' < a\" => a < a\" {");
    // AssertGenerator add assertion
    Assert.assertEquals("a &#60; a&#39; &#38;&#38; a&#39; &#60; a&#34; =&#62; a &#60; a&#34; &#123;", escapedText);
    // AssertGenerator create local variable with return value of invocation
    boolean o_escapeAttributeValue__3 = escapedText.contains("<");
    // AssertGenerator add assertion
    Assert.assertFalse(o_escapeAttributeValue__3);
    // AssertGenerator create local variable with return value of invocation
    boolean o_escapeAttributeValue__4 = escapedText.contains(">");
    // AssertGenerator add assertion
    Assert.assertFalse(o_escapeAttributeValue__4);
    // AssertGenerator create local variable with return value of invocation
    boolean o_escapeAttributeValue__5 = escapedText.contains("'");
    // AssertGenerator add assertion
    Assert.assertFalse(o_escapeAttributeValue__5);
    // AssertGenerator create local variable with return value of invocation
    boolean o_escapeAttributeValue__6 = escapedText.contains("\"");
    // AssertGenerator cre
    boolean o_escapeAttrib
    // AssertGenerator add
    Assert.assertFalse(o_e
    // AssertGenerator cre
    boolean o_escapeAttrib
    // AssertGenerator add
    Assert.assertFalse(o_e
}
```

Original test

```
@Test
public void escapeAttributeValue()
{

    String escapedText = XMLUtils.escapeAttributeValue("a < a' && a' < a\" => a < a\" {");

    assertFalse("Failed to escape <", escapedText.contains("<"));
    assertFalse("Failed to escape >", escapedText.contains(">"));
    assertFalse("Failed to escape '", escapedText.contains("'"));
    assertFalse("Failed to escape \"", escapedText.contains("\""));
    assertFalse("Failed to escape &", escapedText.contains("&&"));
    assertFalse("Failed to escape {", escapedText.contains("{"));
}
```

Mutation: Dspot Strategy

- DSpot is very slow to execute.
- One strategy is to run it on CI from time to time and in the pipeline commit generated tests in a different source root.
- Configure Maven to add a new test directory source using the Maven Build Helper plugin.
- Another idea: run it as GitHub commit hook so that it only executed on the modified code.

Environment Testing

- Environment = combination of Servlet container & version, DB & version, OS, Browser & version, etc

- Using Docker

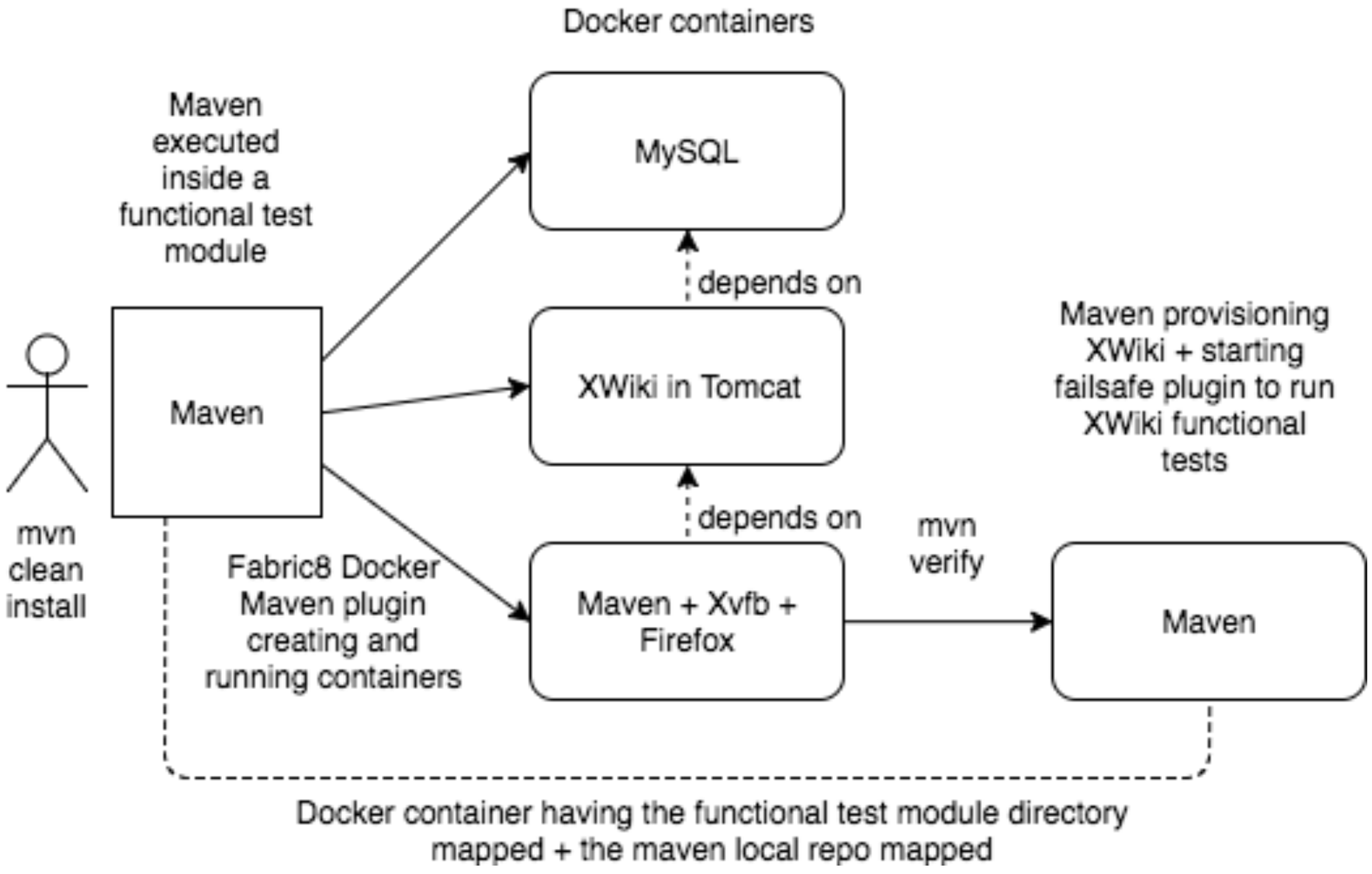
- Need: Be able to run functional tests on local dev machines as well as on CI

  - Lead to using Fabric8 Docker Maven Plugin (DMP) ⚠

Google Chrome 48	HyperSQL 2.3.3
Internet Explorer 11	PostgreSQL 9.5
Internet Explorer 11	HyperSQL 2.3.3
Mozilla Firefox 44	HyperSQL 2.3.3
Mozilla Firefox 43	HyperSQL 2.3.3
Mozilla Firefox 34	MySQL 5.7
Google Chrome 51	MySQL 5.7
Internet Explorer 10	HyperSQL 2.3.4
Internet Explorer 11	HyperSQL 2.3.4
Microsoft Edge 40	MySQL 5.7
Internet Explorer 11	PostgreSQL 9.6.2
Google Chrome 58	PostgreSQL 9.6.2
Google Chrome 59	Oracle 11.2
Internet Explorer 11	HyperSQL 2.4.0
Internet Explorer 11	MySQL 5.7

Environme

- One maven module to generate the XWiki Docker image for the official distribution
- Another maven module to generate the XWiki Maven Docker image (Maven + Browsers installed)
- In each functional test module, use the DMP to start the DB Docker image + the XWiki image + the XWiki Maven image.
- Execute Maven's verify goal inside the XWiki Maven image



Parting words

- Experiment, push the limit!
- Some other types of tests not covered and that also need automation
- Performance/Stress testing
- Usability testing
- others?

Q&A

Me

Vincent Massol

vincent@xwiki.com
skype: vmassol
http://about.me/vmassol

http://xwiki.org
http://xwiki.com