Metrics for Automated Tests

Paulo Cheque

Summer 2009

Definitions

- Measurement
  - Assessment on a standard
- Metric
  - Method to ensure an attribute
  - Composed by one or more measurements
- Indicator
  - Variable that interprets a metric

Ex: High coverage **may** indicate that a software is good

Metrics

- Understanding the progress of the project
- *Feedback*
- Communication
- Problems
  - Identify, monitor and solve
- Qualities
  - Identify, monitor and expose
- Manage

GQM

- Goal -> Question -> Metric

  Example:

- G: To develop high-quality software.
  Q: Software has good tests?
  M:
  - Test coverage +
    Number of assertions +
    Number of failed tests

Metrics for Aut. Tests

- To understand:
  - Quality of SUT
  - Quality of tests
  - Code with few tests and code with a lot of tests
  - Tracking tests
  - Strategies of development

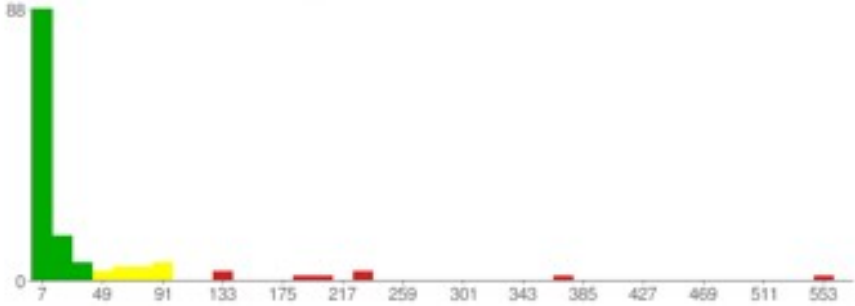
Testability

- OO patterns:
  - Dependency injection
- Anti-patterns:
  - Global variables
  - Public variables
  - Singletons
- TDD

jetty-6.1.9.jar

Class breakdown

```
  Analyzed classes :   128
 Excellent classes :   108   84.4%
      Good classes :    12    9.4%
Needs work classes :     8    6.3%
```



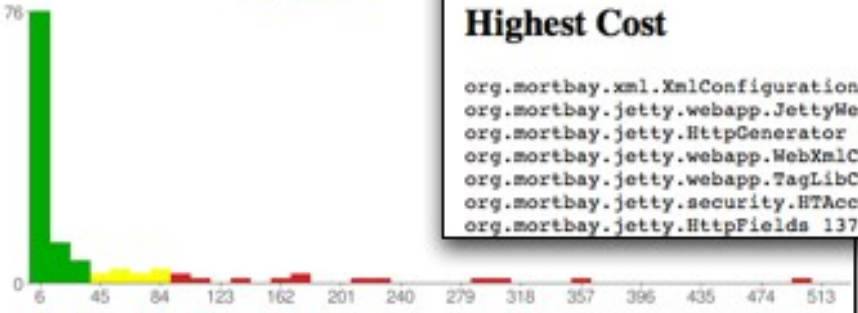
Highest Cost

```
org.mortbay.xml.XmlConfiguration 550
org.mortbay.jetty.webapp.JettyWebXmlConfiguration 371
org.mortbay.jetty.HttpGenerator 236
org.mortbay.jetty.webapp.WebXmlConfiguration 232
org.mortbay.jetty.webapp.TagLibConfiguration 200
org.mortbay.jetty.security.HTAccessHandler$HTAccess 193
org.mortbay.jetty.HttpFields 137
```

jetty-6.0.1.jar

Class breakdown

```
  Analyzed classes :   117
 Excellent classes :    94   80.3%
      Good classes :    10    8.5%
Needs work classes :    13   11.1%
```



Highest Cost

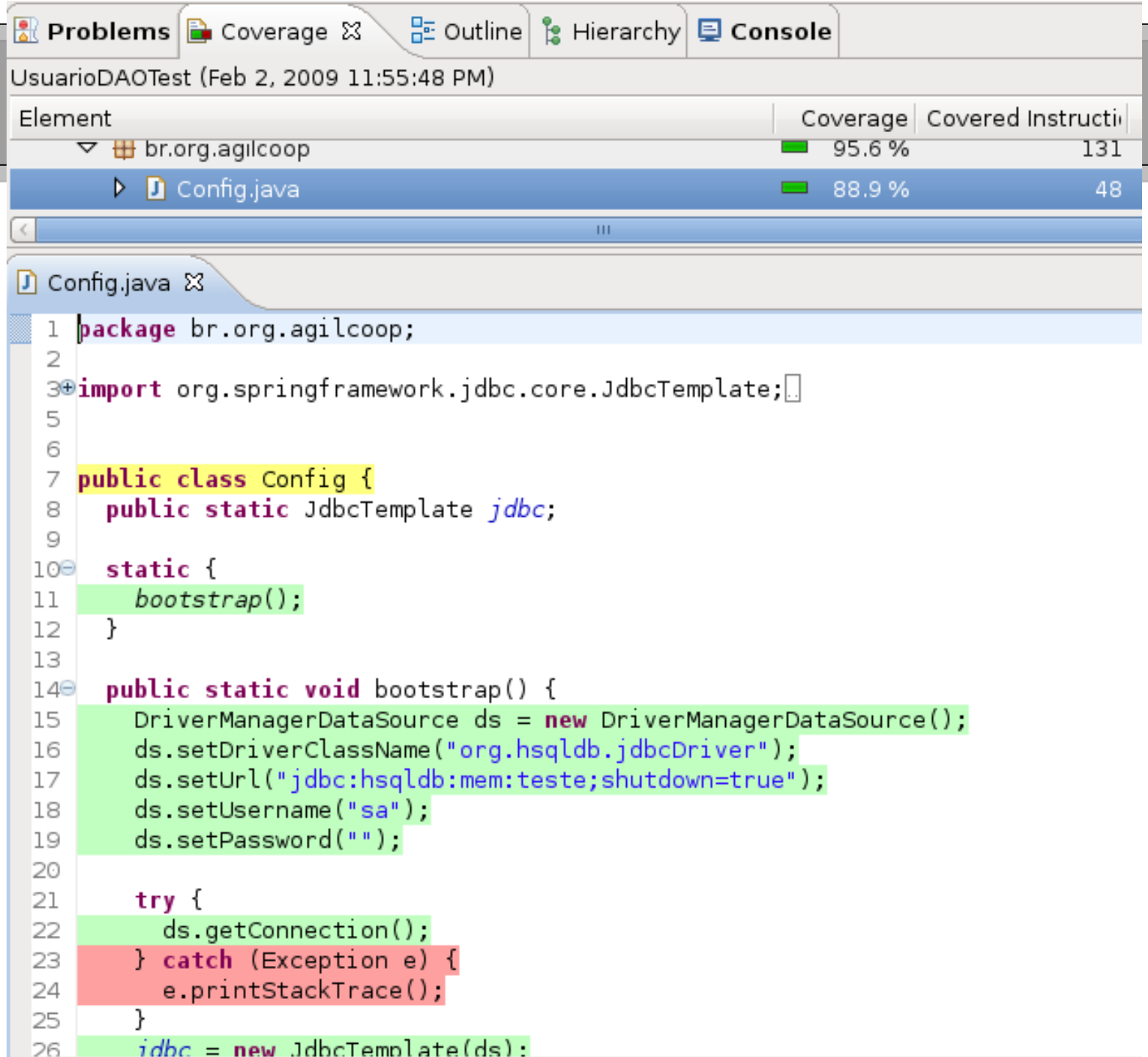
```
org.mortbay.xml.XmlConfiguration 528
org.mortbay.jetty.nio.HttpChannelEndPoint 522
org.mortbay.jetty.HttpConnection 500
org.mortbay.jetty.webapp.JettyWebXmlConfiguration 352
org.mortbay.jetty.HttpConnection$RequestHandler 301
org.mortbay.jetty.MimeTypes 293
org.mortbay.jetty.webapp.WebXmlConfiguration 226
```

Coverage

- Calculates the percentage of code accessed by tests

  Obs: Code accessed by tests doesn't mean lack of errors

- Code not accessed by tests indicates unsafe code (untested code)


Problems
Coverage
Outline
Hierarchy
Console
UsuarioDAOTest (Feb 2, 2009 11:55:48 PM)
Element
Coverage
Covered Instructi
br.org.agilcoop
95.6 %
131
Config.java
88.9 %
48
Config.java
1 package br.org.agilcoop;
2
3 import org.springframework.jdbc.core.JdbcTemplate;
5
6
7 public class Config {
8 public static JdbcTemplate jdbc;
9
10 static {
11 bootstrap();
12 }
13
14 public static void bootstrap() {
15 DriverManagerDataSource ds = new DriverManagerDataSource();
16 ds.setDriverClassName("org.hsqldb.jdbcDriver");
17 ds.setUrl("jdbc:hsqldb:mem:teste;shutdown=true");
18 ds.setUsername("sa");
19 ds.setPassword("");
20
21 try {
22 ds.getConnection();
23 } catch (Exception e) {
24 e.printStackTrace();
25 }
26 jdbc = new JdbcTemplate(ds);

Test Factor

- = Lines of tests / Lines of SUT
- To compare modules of a project
- Obs: Not recommended for tracking tests

of TA / SUT lines of code

- Tracking evolution of AT

- Obs: As long as the growth of a project doesn't have drastic changes in complexity

Lines of tests

- Analog to lines of SUT

  Identify tests that need refactoring

- May be useful for management of maintenance efforts

Amount of tests

- Tracking of AT evolution

- Especially during the beginning a project

Metric	Total	Mean	Std. De	Maximu	Resource causing Maximum	Method
Number of Overridden Methods (avg/m	0	0	0	0	/ExemplosTestesBD/src/main/java/br/org/agilcc	
Number of Attributes (avg/max per typ	2	0.667	0.471	1	/ExemplosTestesBD/src/main/java/br/org/agilcc	
Number of Children (avg/max per type	0	0	0	0	/ExemplosTestesBD/src/main/java/br/org/agilcc	
Number of Classes (avg/max per pack	3	1.5	0.5	2	/ExemplosTestesBD/src/main/java/br/org/agilcc	
Method Lines of Code (avg/max per m	57	4.75	4.567	17	/ExemplosTestesBD/src/main/java/br/org/agilcc	bootstrap
Number of Methods (avg/max per type	11	3.667	2.625	6	/ExemplosTestesBD/src/test/java/br/org/agilcoc	
Nested Block Depth (avg/max per met		1.083	0.276	2	/ExemplosTestesBD/src/main/java/br/org/agilcc	bootstrap
Depth of Inheritance Tree (avg/max pe		1	0	1	/ExemplosTestesBD/src/main/java/br/org/agilcc	
Number of Packages	2					
Afferent Coupling (avg/max per packag		0	0	0	/ExemplosTestesBD/src/main/java/br/org/agilcc	
Number of Interfaces (avg/max per pa	0	0	0	0	/ExemplosTestesBD/src/main/java/br/org/agilcc	
McCabe Cyclomatic Complexity (avg/m		1.083	0.276	2	/ExemplosTestesBD/src/main/java/br/org/agilcc	bootstrap
Total Lines of Code	119					
Instability (avg/max per packageFragm		1	0	1	/ExemplosTestesBD/src/main/java/br/org/agilcc	
Number of Parameters (avg/max per m		0.667	0.943	3	/ExemplosTestesBD/src/main/java/br/org/agilcc	atualizaUsuario
Lack of Cohesion of Methods (avg/max		0	0	0	/ExemplosTestesBD/src/main/java/br/org/agilcc	
Efferent Coupling (avg/max per packag		1.5	0.5	2	/ExemplosTestesBD/src/main/java/br/org/agilcc	
Number of Static Methods (avg/max p	1	0.333	0.471	1	/ExemplosTestesBD/src/main/java/br/org/agilcc	
Normalized Distance (avg/max per pac		0	0	0	/ExemplosTestesBD/src/main/java/br/org/agilcc	
Abstractness (avg/max per packageFra		0	0	0	/ExemplosTestesBD/src/main/java/br/org/agilcc	
Specialization Index (avg/max per type		0	0	0	/ExemplosTestesBD/src/main/java/br/org/agilcc	

Amount of assertions

- Useful to verify that tests are actually testing something (useless tests)
- May indicate tests that need refactoring

Number of pending tests

- Tracking for AT
- May indicate problems with time (pressure)
- May indicate TDD anti-patterns

Number of tests that failed

- Fragility of tests
- Tracking the repair of testing

Assertions per method

- May indicate tests that need refactoring
- May identify classes and methods with a lot of responsabilities

Test code replication

- Tests need refactoring
- Indicate SUT code also has replication

Amount of defects

- Software quality

- Lack of automated tests
  - Unit tests
  - Acceptance tests

Suite of tests execution time

- Find SUT bottlenecks (not mocked tests)
- Find tests that need to be optimized, refactored or moved to another suite

Tools

- Eclipse Metrics:
  - http://metrics.sourceforge.net/update
- Eclipse Eclemma:
  - http://update.eclemma.org
- Testability Explorer:
  - http://code.google.com/p/testability-explorer

Contact

http://www.agilcoop.org.br

http://ccsl.ime.usp.br

http://qualipso.org

agilcoop@agilcoop.org.br

paulocheque@agilcoop.org.br