IME-USP

What Do the Asserts in a Unit Test Tell us About Code Quality?

Mauricio Aniche
Gustavo Oliva
Marco Gerosa

University of São Paulo (USP)

Unit Tests and Code Quality

- Great synergy between a testable class and a well-designed class (Feathers, 2012)
- The write of unit tests can become complex as the interactions between software components grow out of control (McGregor, 2001)
- Agile practitioners state that unit tests are a way to validate and improve class design (Beck et al, 2001)

Unit Tests and Asserts

- Every unit test contains three parts
  - Set up the scenario
  - Invoke the behavior under test
  - Validates the expected output
- Assert instructions
  - assertEquals (expected, calculated);
  - assertTrue(), assertFalse(), and so on
- No limits for the number of asserts per test

A little piece of code

```
class InvoiceTest {
  @Test
  public void shouldCalculateTaxes() {
    // (i) setting up the scenario
    Invoice inv = new Invoice(5000.0);

    // (ii) invoking the behavior
    double tax = inv.calculateTaxes();

    // (iii) validating the output
    assertEquals (5000 * 0.06 , tax );
  }
}
```

Why would…

- … a test contain more than one assert?
- Is it a smell of bad code/design?

Research Design

- We selected 22 projects
  - 19 from ASF
  - 3 from a Brazilian consultancy
- Data extraction from all projects
  - Code metrics
- Statistical Test
- Qualitative Analysis

Hypotheses

- H1: Code tested by only one assert does not present lower cyclomatic complexity than code tested by more than one assert
- H2: Code tested by only one assert does not present fewer lines of code than code tested by more than one assert
- H3: Code tested by only one assert does not present lower quantity of method invocations than code tested by more than one assert

Data Extraction

- Test code
  - Number of asserts per test
  - Production method being tested
- Production code
  - Cyclomatic Complexity (McCabe, 1976)
  - Number of method invocations (Li and Henry, 1993)
  - Lines of Code

Heuristic to Extract the Production Method

```
class InvoiceTest {
  @Test
  public void shouldCalculateTaxes() {
    // (i) setting up the scenario
    Invoice inv = new Invoice(5000.0);

    // (ii) invoking the behavior
    double tax = inv.calculateTaxes();

    // (iii) validating the output
    assertEquals (5000 * 0.06 , tax );
  }
}
```

```
class Invoice {
  public double calculateTaxes()
  {
    // something…
  }
}
```

Selected Projects

- Pre-selected all 213 projects from Apache
- Eliminated those because of:
  - Minimum test ratio (unit tests / production methods >= 0.2)
  - Success in production method detection algorithm (higher than 50% in all unit tests that contain more than 1 assert)

Selected Projects

Table I
DESCRIPTIVE ANALYSIS OF SELECTED PROJECTS

Project	Lines of Code	Classes	Methods	Tests
rat	4,582	107	459	117
maven-enforcer	5,273	62	432	92
maven-doxia-sitetools	5,797	41	254	65
Industry CP	5,820	135	763	208
commons-codec	9,885	77	427	383
commons-validator	10,512	131	744	270
Industry WC	10,745	951	1,211	552
log4j-extras	11,503	147	745	498
cxf-dosgi	11,920	185	708	168
commons-compress	16,425	127	937	299
log4j	27,499	354	2,387	621
maven-2	35,676	346	2,182	498
maven-scm	46,804	820	3,048	698
commons-lang	48,096	211	2,222	2,046
maven-sandbox	53,763	646	4,062	1,067
struts-sandbox	54,774	1,001	6,220	2,291
Industry CW	68,690	1,043	6,897	2,354
shindig	76,821	999	6,513	2,623
commons-math	96,793	877	5,478	2,281
maven-plugins	120,801	1,574	7,452	1,605
directory-shared	144,264	1,147	5,866	3,245
harmony	915,930	7,027	59,665	26,73
Minimum	4582.0	41.0	254.0	65.0
1st Quartile	10570.0	132.0	744.2	277.2
Median	31590.0	350.0	2202.0	586.5
Mean	81020.0	818.5	5440.0	2214.0
3rd Quartile	65210.0	987.0	6132.0	2222.0
Maximum	915900.0	7027.0	59660.0	26740.0

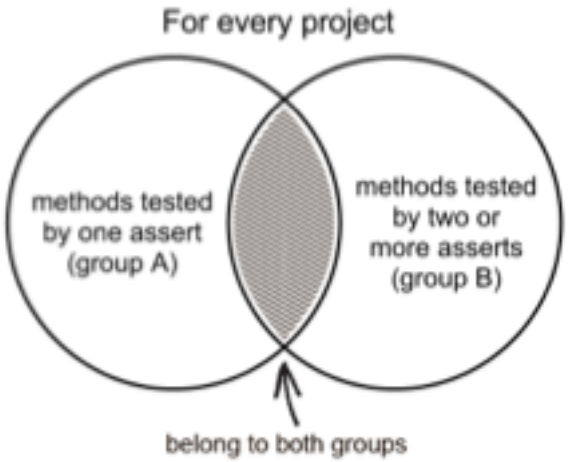
Asserts Distribution in Selected Projects

Table II
DISTRIBUTION OF ASSERTS IN PROJECTS

Project	Zero Assert	One Assert	More Than One Assert
commons-codec	112	125	146
commons-compress	172	26	101
commons-lang	156	397	1,488
commons-math	416	719	1,132
commons-validator	69	29	170
cxf-dosgi	22	60	80
directory-shared	284	1,250	1,704
harmony	5,318	7,063	14,263
log4j	39	464	118
log4j-extras	33	398	67
maven-2	67	249	175
maven-doxia-sitetools	5	9	51
maven-enforcer	39	24	29
maven-plugins	470	377	583
maven-sandbox	96	645	311
maven-scm	242	180	276
Industry CW	327	1269	758
Industry CP	74	66	68
Industry WC	220	242	90
rat	29	27	61
shindig	362	1,098	1,163
struts-sandbox	2,049	157	81

Statistical Test

- Separation in two different groups
  - The same method can appear in both groups
  - No duplicates
- Wilcoxon Signed-Rank Test



Results of the Test

Table III
RESULTING P-VALUES FROM APPLYING WILCOXON TEST TO THE GROUP OF ONE ASSERT AND THE GROUP OF MORE THAN ONE ASSERT

Project	Cyclomatic Complexity	Method Invocations	Lines of Code
commons-codec	0.1457	0.0257*	0.2991
commons-compress	0.7899	0.5374	0.9122
commons-lang	0.6766	0.3470	0.8875
commons-math	0.9230	0.9386	0.9369
commons-validator	0.9477	-	0.9635
cxf-dosgi	0.8445	0.9567	0.9463
directory-shared	0.9518	0.1298	0.9972
harmony	0.0174*	0.2822	0.5676
log4j	0.9489	0.6789	0.9885
log4j-extras	0.4811	0.6339	0.5703
maven-2	0.2532	0.9490	0.4427
maven-doxia-sitetools	0.9561	0.9213	0.9595
maven-enforcer	0.9371	0.4064	0.9727
maven-plugins	0.9607	0.6946	0.9847
maven-sandbox	0.4277	0.6499	0.9133
maven-scm	0.9324	0.9846	0.9948
Industry CW	0.9999	0.2567	0.9999
Industry CP	0.2850	0.0003*	0.4425
Industry WC	0.5380	0.5381	0.0561
rat	0.3162	0.4153	0.3263
shindig	0.9968	0.0252*	0.9998
struts-sandbox	0.9758	0.2942	0.9649

Why more than 1 assert?

- 130 tests randomly selected
- Qualitative analysis:
  - More than one assert for the same object (40.4%)
  - Different inputs to the same method (38.9%)
  - List/Array (9.9%)
  - Others (6.8%)
  - Extra assert to check if object is not null (3.8%)

“Asserted Objects”

- We coined the term “asserted objects”
  - It counts not the number of asserts in a unit test, but the number of different instances of objects that are being asserted

```
assertEquals(10, obj.getA());
assertEquals(20, obj.getB());
```

Counts as 1 “asserted object”

Distribution of Asserted Objects

Table IV
DISTRIBUTION OF ASSERTED OBJECTS IN PROJECTS

Project	Zero Assert	One Assert	More Than One Assert
commons-codec	117	212	54
commons-compress	172	91	36
commons-lang	284	1,308	449
commons-math	568	1,247	453
commons-validator	69	131	68
cxf-dosgi	22	109	31
directory-shared	337	1,875	1,027
harmony	5,337	14,699	6,606
log4j	39	538	43
log4j-extras	33	419	46
maven-2	67	348	77
maven-doxia-sitetools	5	46	14
maven-enforcer	39	46	7
maven-plugins	471	628	331
maven-sandbox	450	548	54
maven-scm	244	327	127
Industry CW	449	1,586	319
Industry CP	116	67	25
Industry WC	263	249	40
rat	29	57	31
shindig	506	1,625	492
struts-sandbox	2,049	202	36

Results of the Test

Table V
RESULTING P-VALUES FROM APPLYING WILCOXON TEST TO THE GROUP OF ONE ASSERTED OBJECT AND THE GROUP OF MORE THAN ON

Project	Cyclomatic Complexity	Method Invocations	Lines of Code
commons-codec	0.0376*	0.1376	0.0429*
commons-compress	0.0385*	0.0445*	0.4429
commons-lang	0.9918	0.7207	0.9611
commons-math	0.0262*	0.1873	0.0325*
commons-validator	0.0329*	0.6256	0.0223*
cxf-dosgi	0.5351	0.9986	0.3919
directory-shared	0.9998	0.1408	1.0000
harmony	3.5300E-05*	0.0001*	0.0043*
log4j	0.2749	0.6789	0.3574
log4j-extras	0.0658	0.6339	0.0594
maven-2	0.2912	0.6228	0.1611
maven-doxia-sitetools	0.8012	1.0000	0.8986
maven-enforcer	0.2670	0.8219	0.1348
maven-plugins	0.1461	0.1972	0.0953
maven-sandbox	0.9821	0.1400	0.9815
maven-scm	0.9726	0.9193	0.9999
Industry CW	0.9999	0.0089*	0.9986
Industry CP	0.9103	0.7909	0.3055
Industry WC	0.1274	0.4718	0.0606
rat	0.2213	0.1533	0.1373
shindig	0.0006*	0.0238*	0.0002*
struts-sandbox	0.9994	0.6910	0.9995

Results of the Test

Summary of the Tests

Table VII
SUMMARY OF THE RESULTS OF THE STATISTICAL TESTS: QUANTITY OF PROJECTS THAT REFUTED EACH HYPOTHESES

Hypotheses	Qty of Projects (Asserts)		Qty of Projects (Asserted Objects)	
H1 (Cyclom. Complexity)	01	(05%)	06	(27%)
H2 (Lines of Code)	00	(00%)	05	(22%)
H3 (Method Invocations)	03	(13%)	04	(18%)

Findings

- Counting the number of asserts in a unit test does not give valuable feedback about code quality
  - But counting the number of asserted objects may provide useful information
  - However, the difference between both groups was not “big”
- A possible explanation:
  - Methods that contain higher CC, lines of code, and method invocations contains many different paths, and developers prefer to write all of it in a single unit test, rather than splitting in many of them

Threats to Validity

- Many unit tests variations that are not supported by our tool
- The production method detection algorithm
- Before/After methods
- The projects' filtering process
- Integration and system tests

Related Work

- Most similar to our work: Bruntink and van Deursen (2006).
  - Significant correlation between class metrics (fan out, size of a class, response for a class) and test metrics (size of the class, number of asserts).

Conclusions

- The number of asserts does not tell us anything about the production code quality
  - But the number of asserted objects does.
  - "More than one asserted object per test" smell
- The need of replicating the study with a larger amount of industry projects

Contact Information

- Mauricio Aniche
  aniche@ime.usp.br / @mauricioaniche
- Gustavo Oliva
  goliva@ime.usp.br / @golivax
- Marco Gerosa
  gerosa@ime.usp.br

Software Engineering & Collaborative Systems Research Lab (LAPESSC)
http://lapessc.ime.usp.br/