Alexander Menk

SoSe 17

Unit-Tests?!

Testing in Open-Source Projekten

Motivation

- Softwarequalität verbessern
- Test Automation
- Aus Programmierer-Sicht
- Concurrent Testing

- Eigene Erfahrungen!

Überblick Testing

- Grundlagen
- Exemplarisch
  - Oracle Problem
  - Egoless-Programming
  - Unit Tests
  - Coverage
- Testing in Open Source

Forschung

- Paper I
- Paper II
- Konferenzen

Master?

- Fragestellungen
- Interessen
- Projekte?

Überblick Testing

Definition Testing

> “Software testing consists of the dynamic verification that a program provides expected behaviors on a finite set of test cases, suitably selected from the usually infinite execution domain.”
>
> – Society, I. E. E. E. C.; Bourque, P. & Fairley, R. E.: Guide to the Software Engineering Body of Knowledge (SWEBOK(R)): Version 3.0, Hoboken, New Jersey, USA: *IEEE Computer Society Press.*, 2014, pp 4 – 1 Introduction

Grundlagen: Einige Aspekte

- Formalisierung von Abläufen
  - Vokabular, Klassifikationen
    - White Box, Black Box, Oracle
  - Techniken, Methoden
    - Random Testing, Mutation Testing
  - Auf unterschiedlichen Ebenen
    - **Module**
    - Modules oder Modulgruppen
    - System
- Richtlinien, **Mindset**

```ruby
# do foo on bar
def foo bar
  yield bar # …
end
```

[1]

The Oracle Problem

- Ein Mechanismus der das Ergebnis eines Tests bewertet [2]

```ruby
require "test/unit"

class TestCalculation < Test::Unit::TestCase

    def test_addition
        assert_equal 42, 21 + 21
        assert_equal  1,  0 +  1
        assert_equal  1,  1 +  0
    end
end
```

[Abbildung 2]

[Abbildung 3]

The Oracle Problem

- Wann ist ein Ergebnis korrekt? [2]
  - Mathematisch
  - Innerhalb der Domäne
  - Nicht funktionale Aspekte
- Extrem- und Randfälle
- Ausnahmebehandlung
- Systemtests

- Spezifikationen
- Dokumentation
- Kommentare
- Test-Anleitung
- Statistik
- Gesetze

The Oracle Problem

```
def substr string, start, length
  if string.instance_of? String
    string[start..start + length]
  end
end
```

```
“foo bar” ← substr “foo bar”,0,7
“bar”     ← substr “foo bar”,4,3
“r”       ← substr “foo bar”,6,1
nil       ← substr nil      ,4,2
?         ← substr Date.new ,0,0
```

- Randfälle?
- Ausnahmen?
- Methode < Modul < System

(Testing und Psychologie)

- Der Mensch
- Kommunikation
  - Programmierer
  - Reviewer
  - Tester
- “Egoless Programming” (1971)
  - Code Ownership
  - Who is to blame?

- The 10 Commandments [3]
  - Mistakes
  - Someone else is better
  - Work together
  - …

Unit Test

> A minimal software unit that can be tested in isolation.
>
> - ISTQB Unit[4]

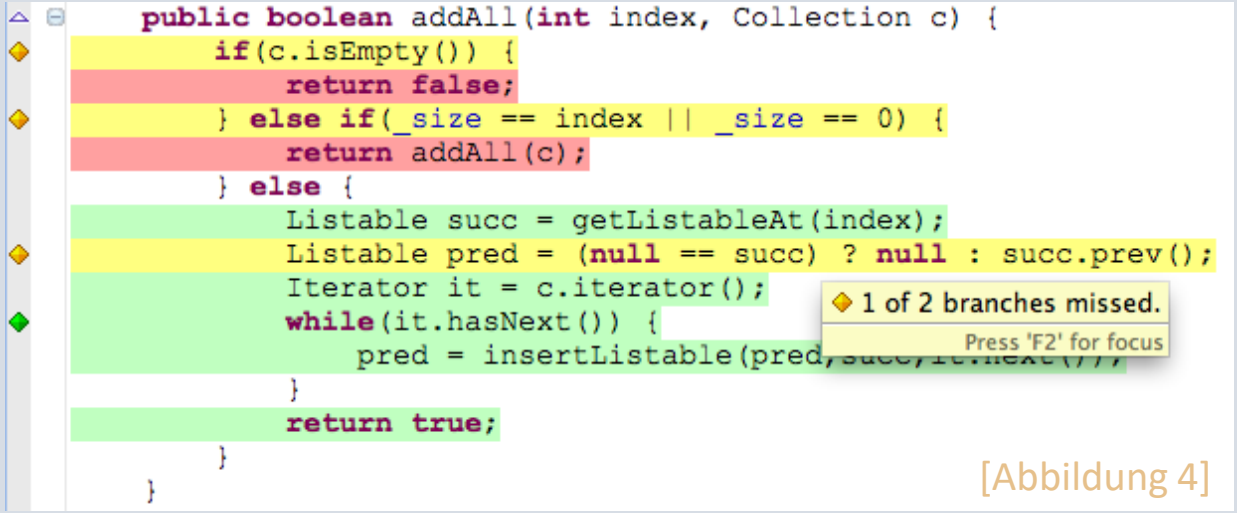
> 1. Testing of individual routines and modules by the developer or an independent tester.
> 2. A test of individual programs or modules in order to ensure that there are no analysis or programming errors.
> 3. Test of individual hardware or software units or groups of related units
>
> - IEEE Unit[5]

Coverage

- Methode!
- Branches
- Erreichbar?

- Branch Coverage
- Statement Coverage
- …

[1]



[Abbildung 4]

$$Coverage = \frac{LOC_C}{LOC}$$

Testing in Open Source

- Aktueller Stand?
- Was?
- Wie?

Testing in Open Source

- Frameworks
- Kategorisierung
- Imports

→ Data Mining

Testing in Open Source

82 447 Open Source Projekte

17% Test Cases

93 genutzte Frameworks

[6]


[clojure.java.io :as io]
[ring.core.protocols :refer :all]))
(deftest test-write-body-defaults
(testing "strings"
(let [output (java.io.ByteArrayOutputStream.)
response {:body "Hello World"}]
(write-body-to-stream (:body response) response output)
(is (= "Hello World" (.toString output)))))
(testing "strings with encoding"
(let [output (java.io.ByteArrayOutputStream.)
response {:headers {"Content-Type" "text/plain; charset=UTF-16"}
:body "Hello World"}]
(write-body-to-stream (:body response) response output)
(is (= "Hello World" (.toString output "UTF-16")))))
:body (list "Hello" " " "World")}]
(write-body-to-stream (:body response) response output)
(is (= "Hello World" (.toString output "UTF-16")))))
(testing "input streams"
(let [output (java.io.ByteArrayOutputStream.)
response {:body (io/input-stream (io/resource "ring/assets/hello world.txt"))}]
(write-body-to-stream (:body response) response output)

Forschung

Papers

I. Gonzalez, D.; Santos, J. C. S.; Popovich, A.; Mirakhorli, M. & Nagappan, M.: A Large-scale Study on the Usage of Testing Patterns That Address Maintainability Attributes: Patterns for Ease of Modification, Diagnoses, and Comprehension. In:: *IEEE Press*. : *Proceedings of the 14th International Conference on Mining Software Repositories*., 2017, S. 391--401

II. Trautsch, F. & Grabowski, J.: Are There Any Unit Tests? An Empirical Study on Unit Testing in Open Source Python Projects. In:: . : *2017 IEEE International Conference on Software Testing, Verification and Validation (ICST)*., 2017, S. 207-218

Paper I

- Data Mining
- Metriken
  - $TP_{Ratio} = \frac{LOC_{Unit}}{LOC_{Prod}}$
  - Aufteilung in Projektgröße
- Gesichtspunkte
  - Qualitätsattribute
  - Test-Entwurfsmuster
  - Frameworks

Sehr klein < 1k LOC
Klein < 10k LOC
Mittel < 100k LOC
Groß >= 100k LOC

82 450 Projekte
48 Sprachen
17% Tests
93/251Frameworks

Paper I Methodik

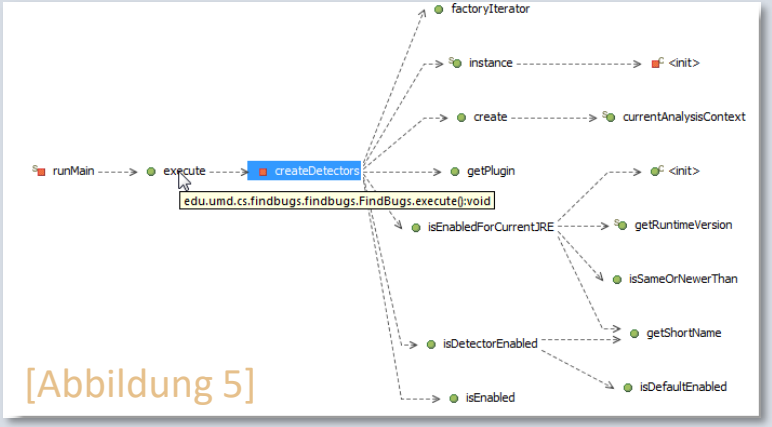
1. Projekte Sammeln und in DB speichern
2. Leere Projekte, Forks filtern
3. Imports, Keywords suchen
4. Test Cases suchen
   - Datei besitzt Framework-Import
   - Keywords des Frameworks vorhanden
5. Patterns ausfindig machen

Sehr klein < 1k LOC
Klein < 10k LOC
Mittel < 100k LOC
Groß >= 100k LOC

82 450 Projekte
48 Sprachen
17% Tests
93/251Frameworks

Paper I Test-Entwurfsmuster

- Simple Test
  - Anzahl an Asserts
- Implicit Teardown
  - `teardown` Prozedur
  - `Destroy`, `remove` Anweisungen
- Assertion Message
  - Regex
- Testcase Class Per Class
  - Call Graph



[Abbildung 5]

Paper I Test-Entwurfsmuster

- 23,75% nutzten Pattern
- Assertion Messages 41%
- Test Case Per Class 33,3%

TABLE VII: Pattern Detection Results

Pattern	# Files	# Projects
Assertion Message	49552	3048
Simple Test	1614	531
Implicit Teardown	920	153
TCCPC	4565	810

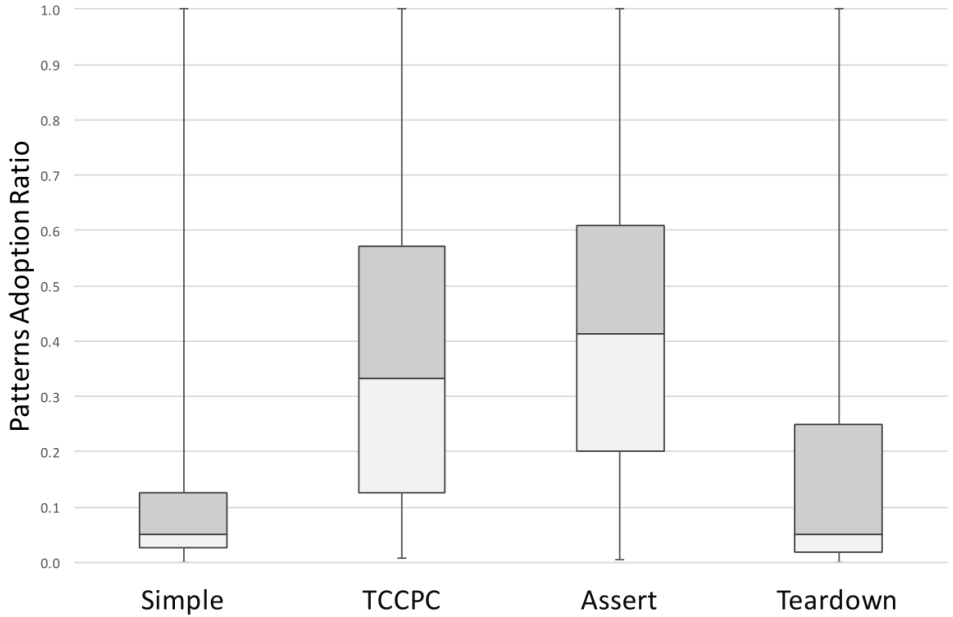
[Abbildung 6]

TABLE IV: Test-Production Code Ratios Across Project Sizes

[Abbildung 7]

	All Proj.	Very Small	Small	Medium	Large
Total # Projects	82447	38379	25425	13322	5321
Avg. TPRatio	0.06	0.05	0.06	0.08	0.08
Max. TPRatio	12.20	12.20	11.05	7.92	4.62
#Proj. With Tests	15119	3207	4203	4131	3578

Fig. 1: Pattern Adoption Ratios (Excludes No-Pattern Projects)



[Abbildung 8]

Paper I Qualitätsattribute

- Ease of Modification
  - Nur 4,32% mit Teardown
  - ¼ aller Projekte (mit Tests) modifizierbar
- Ease of Diagnosis
  - Viele Assertion-Messages
- Ease of Comprehension
  - 2% verständlich nach Kriterien

Sehr klein < 1k LOC
Klein < 10k LOC
Mittel < 100k LOC
Groß >= 100k LOC

82 450 Projekte
48 Sprachen
17% Tests
93/251Frameworks

Paper I Frameworks

TABLE II: Number of Projects included in the study

Language	Freq.	Language	Freq.	Language	Freq.
JavaScript	20201	Lua	343	F#	21
Java	14493	Haskell	338	Processing	21
Ruby	8558	Rust	241	Dart	19
Python	7180	CoffeeScript	230	FORTRAN	17
PHP	6889	Matlab	214	Objective-C++	16
C++	4758	Groovy	161	Haxe	15
C	4369	Puppet	99	Julia	15
C#	4365	TypeScript	63	Elixir	13
Shell	1879	Emacs Lisp	62	Pascal	13
Objective-C	1493	PowerShell	55	Prolog	13
Go	858	Arduino	46	Visual Basic	13
Swift	782	ASP	44	Common Lisp	12
Scala	633	OCaml	29	Mathematica	11
VimL	440	Assembly	27	AGS Script	9
Perl	399	Erlang	27	BlitzBasic	9
Clojure	386	ActionScript	26	Scheme	9
R	366	D	23	Cuda	8

*Total number of projects: 82,447

[Abbildung 9]

TABLE VI: Top 20 Most Used Frameworks

Framework	Language	Freq.	Normalized Freq.
Mocha	JavaScript	6714	0.33
JUnit	Java	3841	0.27
unittest	Python	1663	0.23
PHPUnit	PHP	1000	0.14
PHP Unit Testing	PHP	867	0.13
Test::More	Perl	49	0.12
NUnit	C#	443	0.10
Autotest	Python	466	0.06
Check	C	251	0.06
SimpleTest	PHP	348	0.05
Nose	Python	336	0.05
Mockito	Java	573	0.04
Moq	C#	164	0.04
JSUnit	JavaScript	758	0.04
Enhance JS	JavaScript	737	0.04
Sinon.js	JavaScript	638	0.03
py.test	Python	213	0.03
Chai	JavaScript	523	0.03
Vows	JavaScript	517	0.03
Boost Test Library	C++	113	0.02

Normalized Freq= Framework Freq. Divided by the Total #Projects in that Programming Language

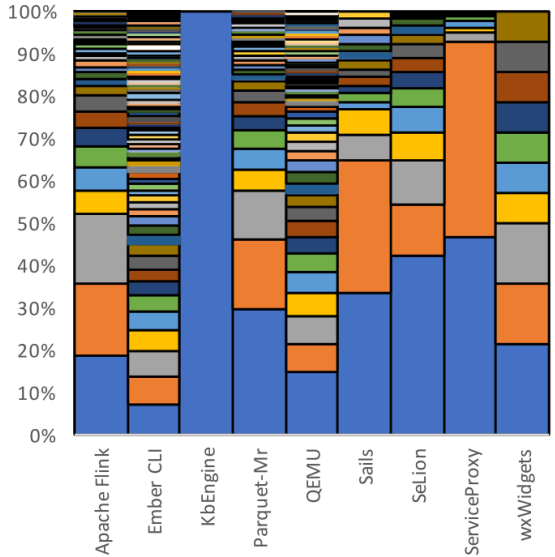
[Abbildung 10]

Paper I Weitere Ergebnisse

- Nur **neun** Projekte setzten alle Patterns um
- Wenige Programmierer setzen die meisten Patterns um
- Kein Einfluss durch Industrie erkennbar

→ Eigenentscheidung der Programmierer

Fig. 2: Proportion of Contributions to Test Files with Patterns Across the Investigated Projects that Implemented All Patterns (Each Color is a Contributor)



[Abbildung 11]

Paper II Are There Any Unit Tests?

- Fragen
  I. Kategorisierung
  II. Wie viele Test sind Unit Tests?
  III. Mocking Mechanismen
  IV. Entwicklung auf Zeit
- Vorraussetzungen
  - Projekt besitzt Tests
  - > 200 Revisionen
  - Unterscheidung zwischen Unit-/ Tests

- 10 Python Projekte
- 70 000 Revisionen

Project	#Revisions	#Tests	#File Changes
ansible [34]	19,992	396,857	49,514
aws-cli [35]	3,958	476,644	42,042
nose [36]	1004	75,087	3726
nose2 [37]	784	29,294	2,764
nupic [38]	5,825	549,029	346,371
pip [39]	4,745	161,506	16,661
robotframework [40]	11,083	823,056	39,150
scikit-learn [41]	20,969	2,025,313	105,271
vcrpy [42]	657	12,520	1,503
warehouse [43]	1,937	87,926	6,872
Overall:	**70,953**	**4,637,232**	**613,874**

Paper II Methodik

- Projekt in DB Speichern
- Imports, Mocked Imports sammeln
  - Anzahl Imports, Ort des Imports
  - Dateiname enthält „test“
- Mock Nutzung via Regex feststellen
- Kategorisiere Tests nach Imports

- 10 Python Projekte
- 70 000 Revisionen

Project	#Revisions	#Tests	#File Changes
ansible [34]	19,992	396,857	49,514
aws-cli [35]	3,958	476,644	42,042
nose [36]	1004	75,087	3726
nose2 [37]	784	29,294	2,764
nupic [38]	5,825	549,029	346,371
pip [39]	4,745	161,506	16,661
robotframework [40]	11,083	823,056	39,150
scikit-learn [41]	20,969	2,025,313	105,271
vcrpy [42]	657	12,520	1,503
warehouse [43]	1,937	87,926	6,872
Overall:	**70,953**	**4,637,232**	**613,874**

Paper II Test Kategorien

- ALL
- DEV
  - Unit Test laut Ort, Dokumentation
- ISTQB
  - Nur ein Import
- IEEE
  - Imports nur im selben Package
- Weitere Kategorien ISTQBD, ….

- 10 Python Projekte
- 70 000 Revisionen

Project	#Revisions	#Tests	#File Changes
ansible [34]	19,992	396,857	49,514
aws-cli [35]	3,958	476,644	42,042
nose [36]	1004	75,087	3726
nose2 [37]	784	29,294	2,764
nupic [38]	5,825	549,029	346,371
pip [39]	4,745	161,506	16,661
robotframework [40]	11,083	823,056	39,150
scikit-learn [41]	20,969	2,025,313	105,271
vcrpy [42]	657	12,520	1,503
warehouse [43]	1,937	87,926	6,872
Overall:	**70,953**	**4,637,232**	**613,874**

Paper II Ergebnisse

I. Großer Unterschied zwischen DEV und ISTQB & IEEE
II. Basierend auf Definition und Projekt 0 – 68%
III. Mocking wird je nach Projekt intensiv eingesetzt
IV. Vier Strukturen, keine Aussage möglich

- 10 Python Projekte
- 70 000 Revisionen

Project	#Revisions	#Tests	#File Changes
ansible [34]	19,992	396,857	49,514
aws-cli [35]	3,958	476,644	42,042
nose [36]	1004	75,087	3726
nose2 [37]	784	29,294	2,764
nupic [38]	5,825	549,029	346,371
pip [39]	4,745	161,506	16,661
robotframework [40]	11,083	823,056	39,150
scikit-learn [41]	20,969	2,025,313	105,271
vcrpy [42]	657	12,520	1,503
warehouse [43]	1,937	87,926	6,872
Overall:	**70,953**	**4,637,232**	**613,874**

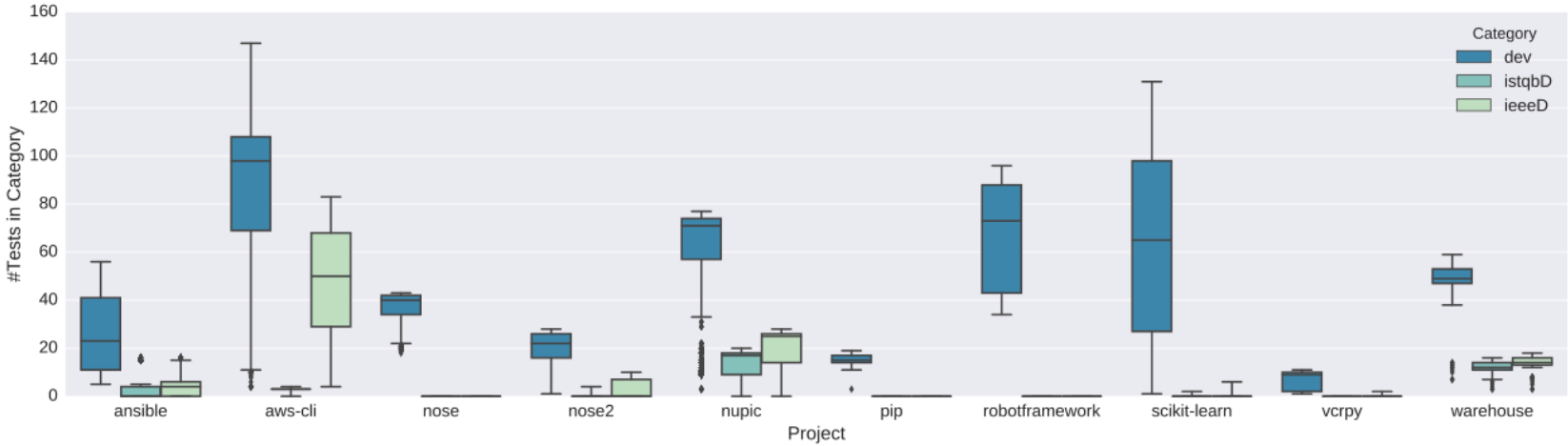


Fig. 3. Boxplot of the categories *dev*, *istqbD*, and *ieeeD* for the different projects.

Konferenzen

- ISSTA 2017, 10 – 14 Juli, Santa Barbara, California, USA
  www.conf.researchr.org/home/issta-2017
- ASE 2017, 30 Oktober – 3 November, Urbana-Champaign, Illinois, USA
  www.ase2017.org/

- CAST 2017, 16 – 18 August Nashville, Tennessee, USA
  www.associationforsoftwaretesting.org/conference/cast-2017/
- STAREAST 2018, 29 April – 4 Mai, Orlando, Florida, USA
  www.stareast.techwell.com/


[clojure.java.io :as io]
[ring.core.protocols :refer :all]))
(deftest test-write-body-defaults
(testing "strings"
(let [output (java.io.ByteArrayOutputStream.)
response {:body "Hello World"}]
(write-body-to-stream (:body response) response output)
(is (= "Hello World" (.toString output)))))
(testing "strings with encoding"
(let [output (java.io.ByteArrayOutputStream.)
response {:headers {"Content-Type" "text/plain; charset=UTF-16"}
:body "Hello World"}]
(write-body-to-stream (:body response) response output)
(is (= "Hello World" (.toString output "UTF-16")))))
Master
:body (list "Hello" " " "World")}]
(write-body-to-stream (:body response) response output)
(is (= "Hello World" (.toString output "UTF-16")))))
(testing "input streams"
(let [output (java.io.ByteArrayOutputStream.)
response {:body (io/input-stream (io/resource "ring/assets/hello world.txt"))}]
(write-body-to-stream (:body response) response output)

Eigene Schlüsse und Fragen

- Zu wenig (Unit-) Tests
- Fehlende Grundlagen der Testautomatisierung
- Definitionen werden nicht angewendet
- Richtige Anwendung scheint an Personen zu liegen

- Umsetzung von Grundlagen nötig – oder abwegig?
- Vermeidung von Psychologie
- Unit Tests anhand von Spezifikationen geniereren?

Eigene Interessen

- Lösung für die Informatik entwickeln
- Test First, Property Testing
- Abstraktionen
- Funktionale Programmierung
- Untersuchungen Nutzen, nicht Durchführen

Grundprojekt: Untersuchung

- Repositories minen
- Fragen:
  - Testing Framework in Sprache mitgeliefert?
  - Werkzeuge bieten Testing Möglichkeiten?
- Beispiel, Vergleich:
  - Java → Eclipse/Intellij, JUNIT
  - Java → Maven, JUNIT & Mockito
  - Clojure → Leiningen, Clojure.test

Hauptprojekt und Master

- Ergebnisse des Grundprojekts nutzen:
  - Testing vereinfachen
  - Testing ‘motivieren’
- Entwicklung einer Bibliothek oder eines Werkzeugs
  - Regeln erzwingen?
- Einen weiteren Ansatz bieten?

- Untersuchung des Prozess’ im Master

Abbildungen

- [Cover] GREENFIELDBOYCE, Nell. *How Do You Lift A Million Pounds Of Stainless Steel? Very Carefully*. 2016, http://www.npr.org/sections/thetwo-way/2016/05/20/477926381/how-do-you-lift-a-million-pounds-of-stainless-steel-very-carefully. Abgerufen. 13.6.17
- [Abbildung 1] NASA. *Journey to Mars*. https://www.nasa.gov/exploration/systems/sls/multimedia/gallery/srmtest1.html. Abgerufen. 13.6.17
- [Abbildung 2] Screenshot
- [Abbildung 3] Manufacturer, T. *Man or machine? Robots on the rise* https://www.themanufacturer.com/articles/man-or-machine-robots-on-the-rise/ Abgerufen. 13.6.17
- [Abbildung 4] M. R. Hoffmann. *EclEmma – Java Code Coverage for Eclipse* http://www.eclipse.org/community/eclipse_newsletter/2015/august/article1.php. Abgerufen. 13.6.17
- [Abbildung 5] Chris Chedney. Call Graph https://stackoverflow.com/questions/8813344/what-tools-are-available-to-visualize-what-methods-call-other-methods-for-java-c Abgerufen 13.6.17
- [Abbidlung 6..11] GONZALEZ, Danielle, SANTOS, Joanna C. S., POPOVICH, Andrew, MIRAKHORLI, Mehdi y NAGAPPAN, Mei. *A Large-scale Study on the Usage of Testing Patterns That Address Maintainability Attributes: Patterns for Ease of Modification, Diagnoses, and Comprehension*. Piscataway, NJ, USA: IEEE Press. 2017.p. 391–401.

Abbildungen

- [Abbidlung 11…] TRAUTSCH, F. & GRABOWSKI, J.: Are There Any Unit Tests? An Empirical Study on Unit Testing in Open Source Python Projects. In:: . : *2017 IEEE International Conference on Software Testing, Verification and Validation (ICST)*., 2017, S. 207-218

References

- [1] COMPUTER SOCIETY, IEEE, BOURQUE, Pierre y FAIRLEY, Richard E.. *Guide to the Software Engineering Body of Knowledge (SWEBOK(R)): Version 3.0*. 3rd ed. Hoboken, New Jersey, USA: IEEE Computer Society Press, 2014.
- [2] BERTOLINO, Antonia. *Software Testing Research and Practice*. Berlin, Heidelberg: Springer-Verlag. 2003. .P. 9.
- [3] WEINBERG, Gerald M.. *The Psychology of Computer Programming*. New York, NY, USA: John Wiley & Sons, Inc., 1985
- [4] . *International Software Testing Qualification Board Glossary*. https://www.astqb.org/certified-tester-resources/glossary-software-testing-terms/. Abgerufen 13.06.17
- [5] "Systems and software engineering – Vocabulary". *ISO/IEC/IEEE 24765:2010(E)*. 2010, p. 1-418.
- [6] GONZALEZ, Danielle, SANTOS, Joanna C. S., POPOVICH, Andrew, MIRAKHORLI, Mehdi y NAGAPPAN, Mei. *A Large-scale Study on the Usage of Testing Patterns That Address Maintainability Attributes: Patterns for Ease of Modification, Diagnoses, and Comprehension*. Piscataway, NJ, USA: IEEE Press. 2017.p. 391–401.