"If it walks like a duck"
Software testen

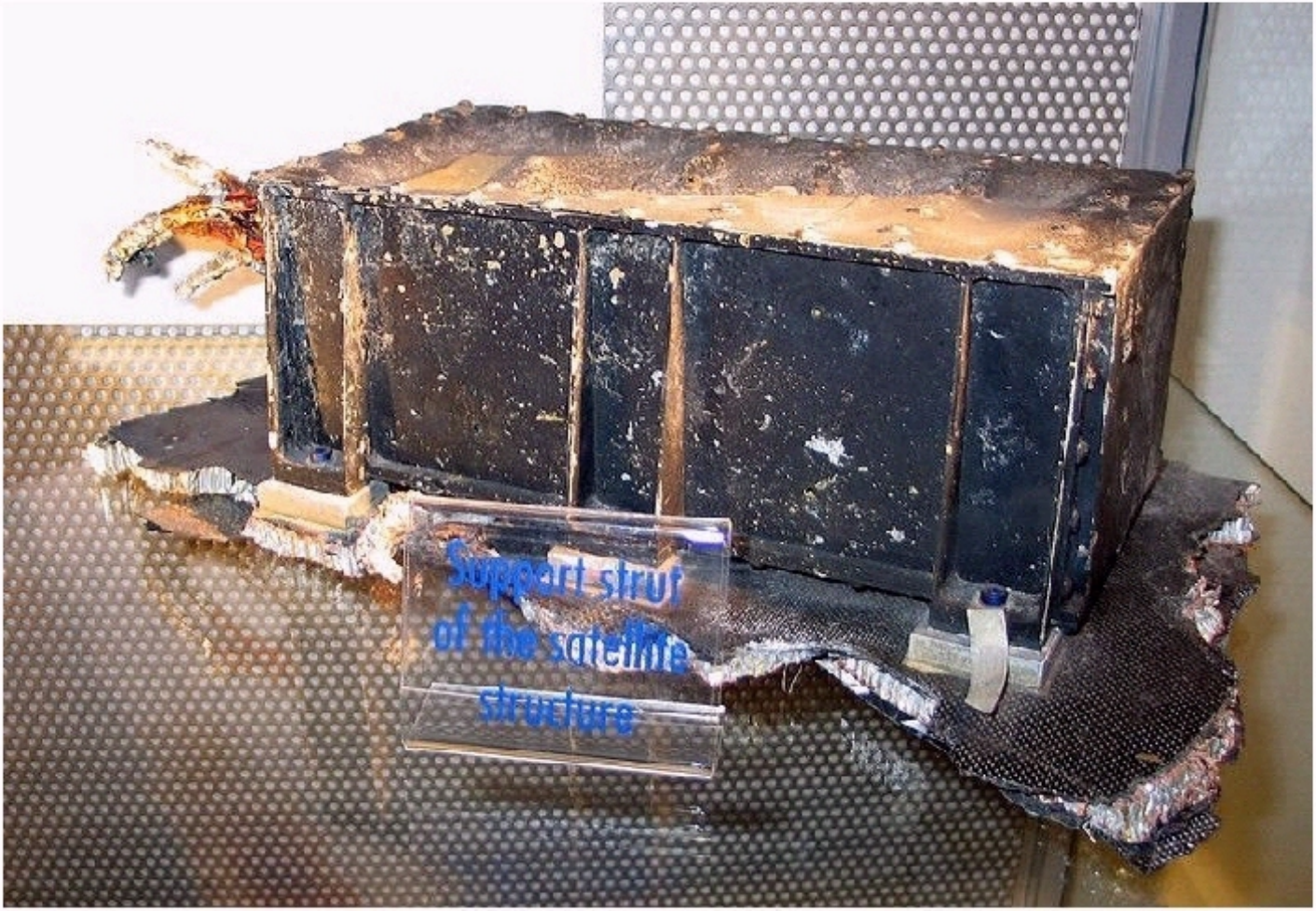
Dr. Hauke Reddmann

Seminar
"Softwareentwicklung in der Wissenschaft"
(C. Hovy), Uni Hamburg, SoSe 2016

Gliederung

1. Einleitung
2. Theorie
3. Fallbeispiele
4. Literatur

Warum?
Was?
Wann?
Wer?
Wo?
Wie?



Warum?

Bild: Wikipedia (Deadpan)

Fehler in Zahlen

1%

0.3%

16%

9%

Warum?

aus: Sneed et al., "Der Systemtest"

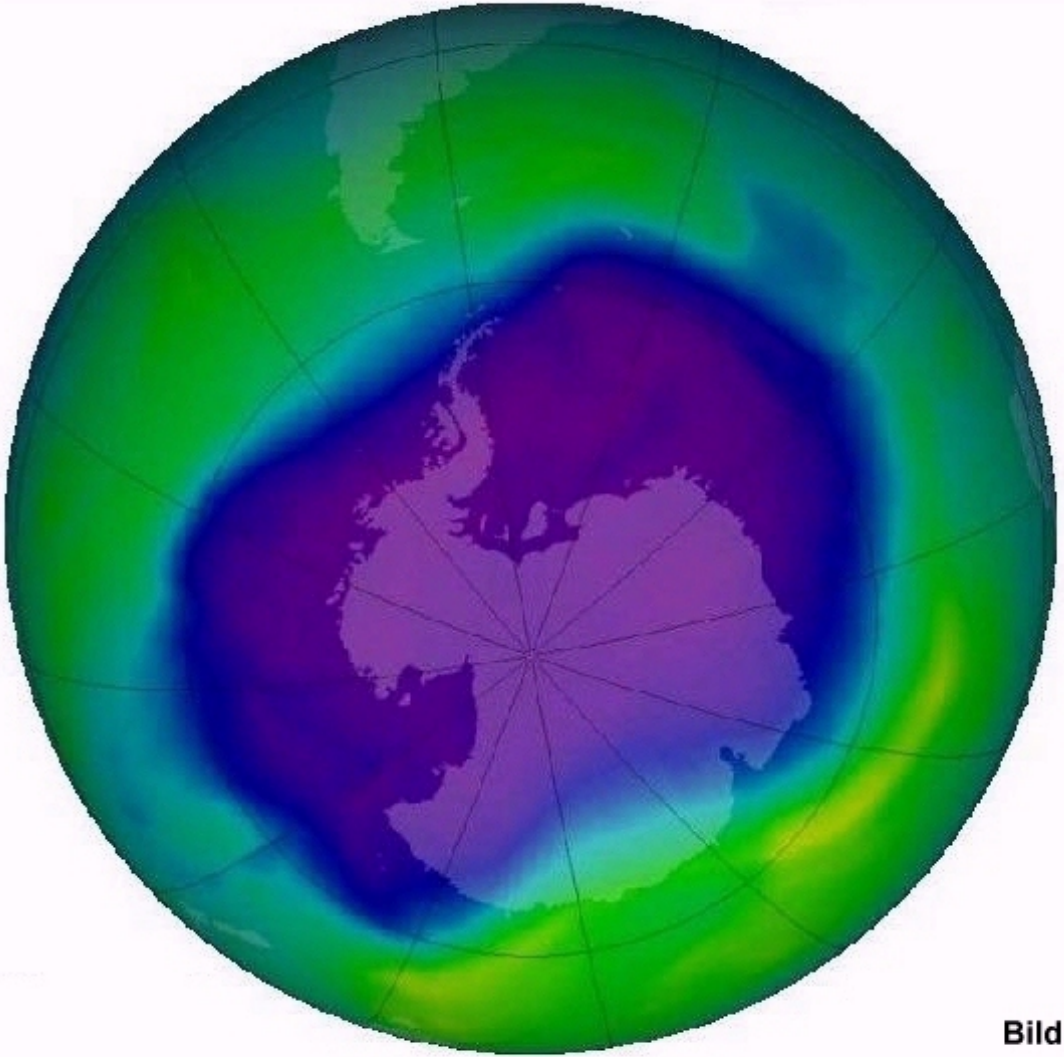
Typische Quellen

- Ungetesteter Code
- Unterschiedliche Reihenfolge
- Ungetestete Inputkombinationen
- Umgebungsvariablen

Betriebssystem
Bibliotheken
Peripherie
Netzwerk

Warum?

Aus: Whittaker, "What is software testing? And why is it so hard?"



Warum?

Bild: Wikipedia (NASA)

Fehler

Qualität

Testen

Orakel

Was?

Bild: Wikipedia (Tilla)

Fehler

Anforderung

Was?

Bild: Internet-Mem

Fehler

Bug - Feature

Sichtbar - Unsichtbar

Finden - Ausschließen

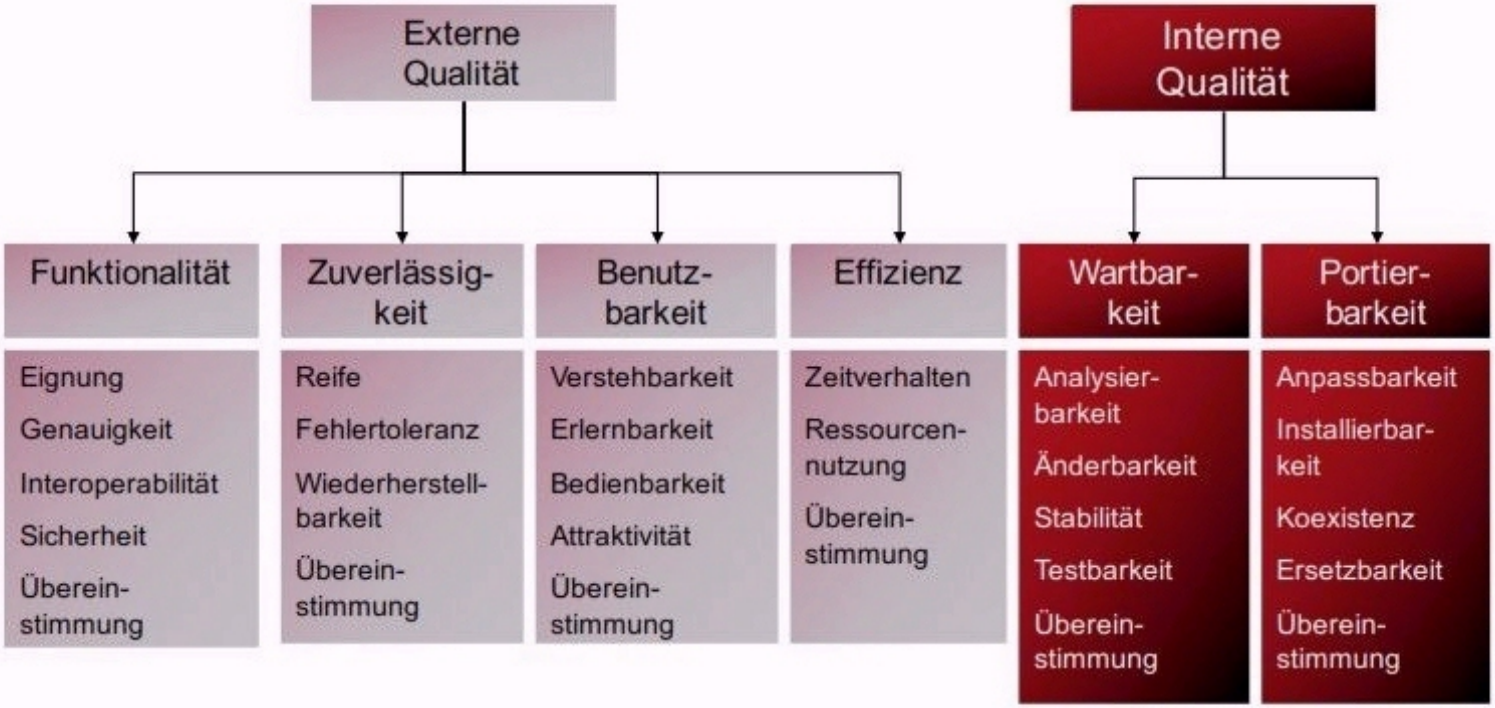
Pro - Kontra

Hardware - Software

Was?

Externe und interne Qualität in der ISO 9126

Nachfolger: ISO 25010



Was?

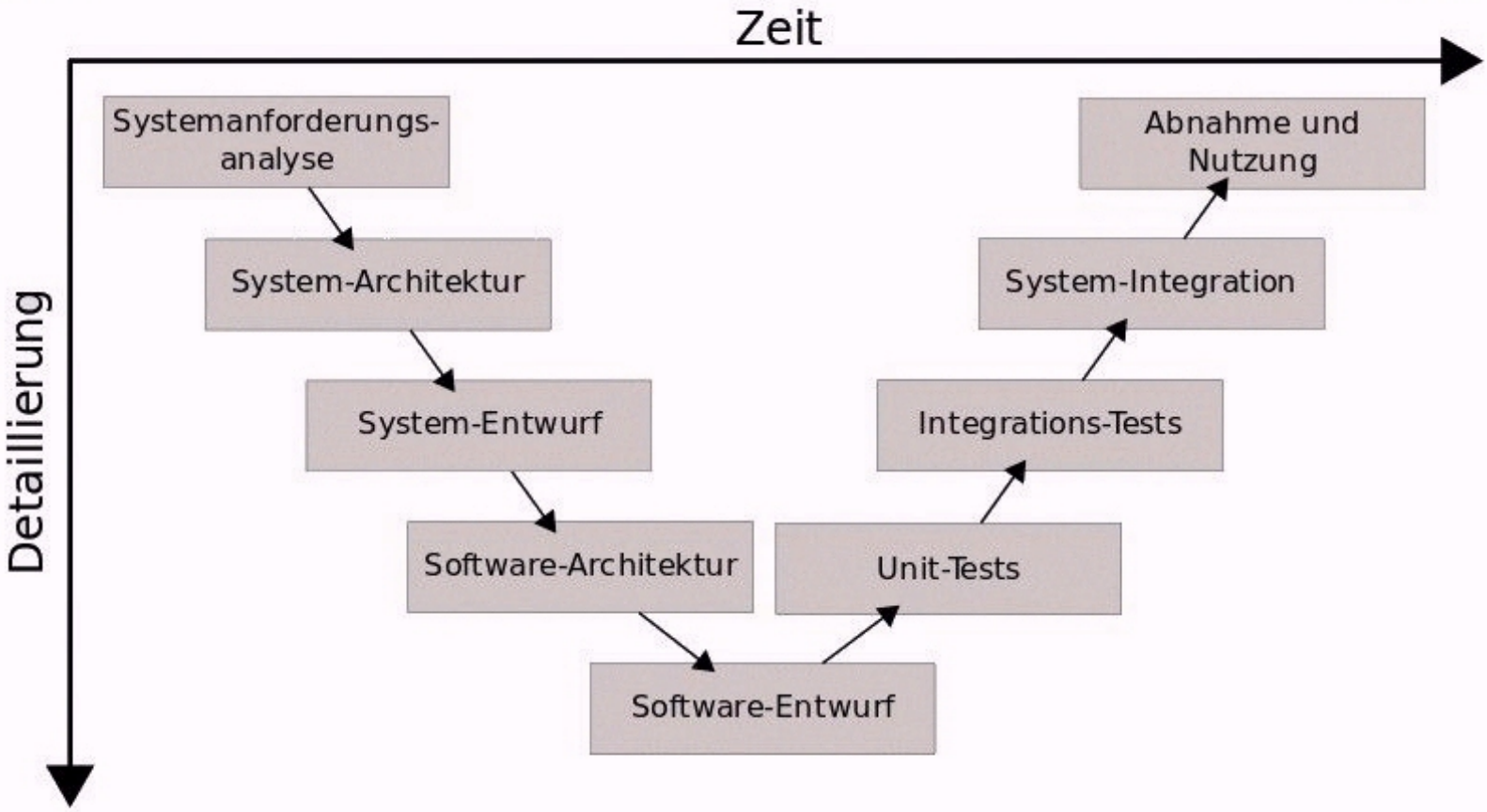
Bild: Vorlesung SE2 HH 2016

Testgrundsätze

- Anwesenheitsbeweis
- Unvollständig
- Früh beginnen
- Clustern von Fehlern
- Resistenz von Fehlern
- Gesamtsystem testen
- Fehlerfrei ≠ funktioniert

Wie?

Aus: Spillner et al., "Basiswissen Softwaretest"



Was?

Bild: Wikipedia (M. Pätzold)

Wann?

Vor dem Programmieren testen!

Wer?

Nicht vom Programmierer testen!

Wo?

Nicht von Organisation testen!

Wann? Wer? Wo?

Aus: Myers, "The Art of Software Testing"

Anweisungstest

Abnahmetest

anwendungsfallbasierter Test

Akzeptanztest

Back-to-back-Test

Alpha-Test

Bedingungstest

anforderungsbasierter Test

Beta-Test

Blackbox-Test

blockierter Test

codebasierter Test

Wie?

statisch
dynamisch

strukturbezogen
änderungsbezogen

funktional Was?
nichtfunktional Wie?

agil

numerisch instabil
In/Output zu groß
Antwort unbekannt
Mißverständnis

Wie?

Bild: Wikipedia

Orakel

Debakel

Pseudo-Orakel

Null-Orakel

Was?

Orakel

Spezifikation
Dokumentation
Mensch
Modell

Ohne

Sampling
Heuristik
Konsistenz
Statistik
Metamorph
Parallel
KI-Lernen
Ducktesting

Was?

Spezifikation

- je formaler, desto besser
- Runtime Assertions

Mensch

- genauso, nur per Hand

Modell

- Analyse des Programms
- int ObskurPlus(int a, int b)

Was?

Heuristik

- Ausgewählte Daten: Sin(x)<=1

Statistik

- Zufällige Daten: 239, 240, 238, 0

Sampling

- Spezielle Daten: Sin(0)=0

Konsistenz

- Wiederholte Daten

Was?

Metamorph

- Bestandene Tests sind nützlich
- Sin(-x)=-Sin(x)

Parallel

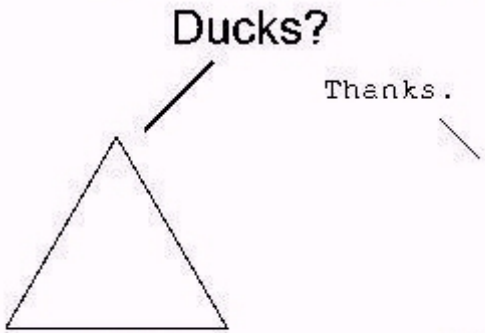
- Verschiedene Programme: 239

KI-Lernen

- Muster in Output

Was?

Ducktesting



Nullorakel

Was?

Bild: P.Shaughnessy, "Triangle & Robert"

Kenndaten

- Vollständigkeit
- Genauigkeit
- Unabhängigkeit vom Programm
- Unabhängigkeit von Umgebung
- Geschwindigkeit
- Brauchbarkeit
- Änderbarkeit des Programms
- Nötiges Vorwissen

Wie?

Aus: Hoffman, "A Taxonomy for Test Oracles"

Table 1: Five Approaches to Oracles

The heuristic approach is only one implement in a tool chest for SQA—the decision about which approach is best for you depends on your test situation. The table below gives descriptions of five approaches to oracles that have been successfully employed across the software industry to verify automated software tests.

	TRUE ORACLE	HEURISTIC ORACLE	SAMPLING ORACLE	CONSISTENT ORACLE	NO ORACLE
Definition	■ Independent generation of all expected results	■ Verifies some values, as well as consistency of remaining values	■ Selects a specific collection of inputs or results	■ Verifies current run results with a previous run (Regression Test)	■ Doesn't check correctness of results (only that some results were produced)
Advantages	■ No encountered errors go undetected	■ Faster and easier than True Oracle ■ Much less expensive to create and use	■ Can select easily computed or recognized results ■ Can manually verify with only simple oracle	■ Fastest method using an oracle ■ Verification is straightforward ■ Can generate and verify large amounts of data	■ Can run any amount of data (limited only by the time the SUT takes)
Disadvantages	■ Expensive to implement ■ Complex and often time-consuming when run	■ Can miss systematic errors (as in following *sine* wave example)	■ May miss systematic and specific errors ■ Often "trains the software to pass the test"	■ Original run may include undetected errors	■ Only spectacular failures are noticed

Wie?

Bild: Hoffman, "Heuristic Test Oracles"

Parallel (1)

Grundsätze:

- unabhängige Programme
- schnell geschrieben
- später häufig benutzt
- eindeutige Spezifikation
- gute Testdaten

Wie?

Aus: Davis et al., "Pseudo-oracles for non-testable programs"

Parallel (2)

- Kontext: Flugzeug-Kabelbremse
- Genetische Algorithmen
- Prinzipiell funktionsfähig
- Fehlerdichte noch zu hoch

Aus: Feldt, "Generating diverse software versions with genetic programming: an experimental study"

Parallel (3)

- Kontext: Dynkin-Indices
- Berechnung einfach...
- wenn neues Programm korrekt
- Widerspruch zu altem Programm
- altes Programm auch korrekt
- Hardwarefehler!

Wie?

Metamorph (1)

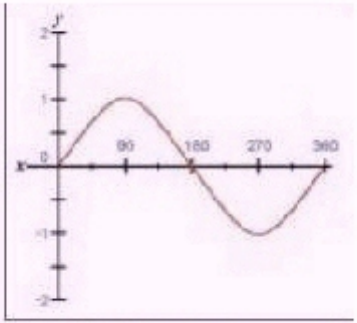


FIGURE 1 A *sine* wave

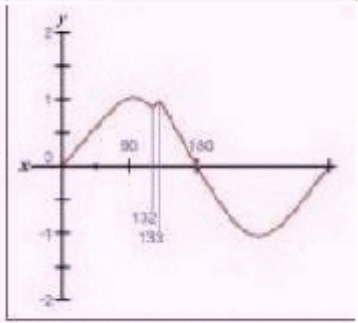


FIGURE 2 A *sine* wave with errors that could be detected by a heuristic oracle

Sin(-x) =-Sin(x)

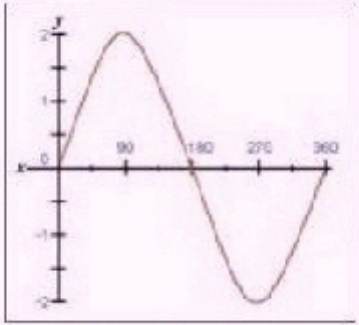
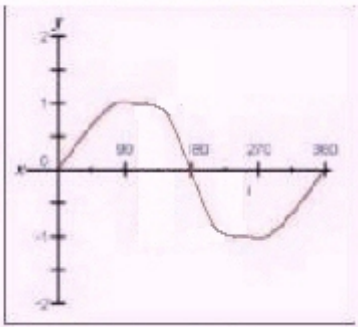


FIGURE 3 Another incorrect *sine* wave



Wie?

Bild: Hoffman, "Heuristic Test Oracles" (modifiziert)

Metamorph (2)

- Kontext: Differentialgleichung
- zu speziell für Bibliotheken
- zu kompliziert für "per Hand"
- Künstlicher Fehler unentdeckbar
- Ungleichungen werden verletzt

Aus: Chen et al., "Metamorphic testing of programs on partial differential equations: a case study"

Metamorph (3)

- Konzept einfach
- trotzdem effektiv
- aufwendig im Input
- aufwendig im Output
- Rundungsfehler

Testautomatisierung!

Wie? Aus: Murphy et al., "Automatic system testing of programs without test oracles"

Statistik

- Kontext: Bildverarbeitung
- große Datenmengen
- nur Testen von Mittelwerten
- deckt trotzdem viele Fehler auf
- Problem: Diskretisierung
- Problem: Werte nahe 0

Wie?

Mayer et al., "Test Oracles Using Statistical Methods"

KI-Lernen

- Kontext: Mesh-Simplifizierung
- Große Datenmengen
- Training der KI mittels vereinfachten Modells
- Genauigkeit vs. Robustheit

Wie?

Chan et al., "PAT: A pattern classification approach to automatic reference oracles for the testing of mesh simplification programs"

Ducktesting

- INVEIG (am LBL entwickelt)
- Nicht von Informatiker
- 4000 Zeilen FORTRAN
- Nur für Spezialisten verständlich
- Standard-Bibliotheken
- Obskure Input-Flags
- 100 peer-reviewte Publikationen
- Vom Autor migriert

Wie? Amberger et al., "Zur Elektronenstruktur metallorganischer Komplexe der f-Elemente"

Take-Home-Messages

- Testen ist nötig, aber manchmal unmöglich.
- Dies gilt gerade in der Wissenschaft:
  - Unbekannte Antwort
  - Instabiler Algorithmus
  - Riesige Datenmengen
- Dennoch wurden Methoden entwickelt, die auch "untestbare" Programme testen.
- "Metamorphes Testen" ist eine gute Methode, die wenig Fachwissen voraussetzt.

Warum?

"Warum sparen wir nicht Zeit und testen einfach nur die Programmteile, die Fehler enthalten?"

Nach: Baumgartner et al., "Agile Testing"

a) Bücher: Testen generell

[Ammann08] P. Ammann, J. Offutt, „Introduction to Software Testing“, Cambridge University Press, Cambridge (Großbritannien), 2008.
[Baumgartner13] M. Baumgartner, M. Klonk, H. Pichler, R. Seidl, S. Tanczos, „Agile Testing“, Hanser, München, 2013.
[Binder00] R. V. Binder, „Testing Object-Oriented Systems: Models, Patterns and Tools“, Addison-Wesley, Boston (USA), 2000.
[Daigl16] M. Daigl, R. Glunz, „ISO 29119“, dpunkt.verlag, Heidelberg, 2016.
[Hendrickson14] E. Hendrickson, „Explore It!“, dpunkt.verlag, Heidelberg, 2014.
[Mili15] A. Mili, F. Tchier, „Software Testing“, Wiley, New Jersey (USA), 2015.
[Myers04] G. J. Myers, „The Art of Software Testing“, 2. Auflage, Wiley, New Jersey (USA), 2004.
[Seidl12] R. Seidl, M. Baumgartner, T. Bucsics, „Basiswissen Testautomatisierung“, dpunkt.verlag, Heidelberg, 1. Auflage, 2012.
[Sneed12] H. M. Sneed, M. Baumgartner, R. Seidl, „Der Systemtest“, Hanser, München, 3. Auflage, 2012.
[Spillner12] A. Spillner, T. Linz, „Basiswissen Softwaretest“, dpunkt.verlag, Heidelberg, 5. überarbeitete und aktualisierte Auflage, 2012.

b) Bücher, Artikel, Konferenzen: Testen speziell

[Chan09] W. K. Chan, S. C. Cheung, J. C. F. Ho, T. H. Tse, „PAT: A pattern classification approach to automatic reference oracles for the testing of mesh simplification programs“, Journal of Systems and Software **82**(3), 2009, pp. 422-434.

[Chen98] T. Y. Chen, S. C. Cheung, S. M. Yiu, „Metamorphic Testing: A New Approach for Generating Next Test Cases“, Technical Report HKUST-CS98-01, Hong Kong University of Science and Technology, 1998.

[Chen02] T. Y. Chen, J. Feng, and T. H. Tse, "Metamorphic testing of programs on partial differential equations: a case study." Computer Software and Applications Conference (COMPSAC 2002), Proceedings. 26th Annual International. IEEE, 2002, pp. 327-333.

[Clune11] T. L. Clune, R. B. Rood. „Software testing and verification in climate model development“, Software, IEEE **28**(6), 2011, pp. 49-55.

[Cox99] M. G. Cox, P. M. Harris, „The design and use of reference data sets for testing scientific software“, Analytica Chimica Acta **380**(2), 1999, pp. 339-351.

[Davis81] Davis, Martin D., Elaine J. Weyuker. "Pseudo-oracles for non-testable programs." Proceedings of the ACM'81 Conference, ACM, 1981, pp. 254-257.

[Feldt98] R. Feldt, „Generating diverse software versions with genetic programming: an experimental study.“ Software, IEE Proceedings, IET, **145**(6), 1998, pp. 228-236.

[Hoffman98] D. Hoffman, "A Taxonomy for Test Oracles", Quality Week, **98**, 1998, pp. 52-60.

[Hoffman99] D. Hoffman, „Heuristic Test Oracles“, Software Testing & Quality Engineering Magazine, (3-4), 1999, pp. 29-32.

[Kanewala13] U. Kanewala, J. M. Bieman, „Techniques for Testing Scientific Programs without an Oracle“, Proceedings of the 5th International Workshop on Software Engineering for Computational Science and Engineering, IEEE Press, 2013, pp. 48-57.

[Kelly10] D. Kelly, S. Thorsteinson, D. Hook, „Scientific Software Testing – Analysis in Four Dimensions“, IEEE Software, **28**(3), 2010, pp. 84-90.

[Mayer04] J. Mayer, R. Guderlei, "Test Oracles Using Statistical Methods", in: Proceedings of the First International Workshop on Software Quality (SOQUA 2004), Ilmenau, Germany, September 2004, pp. 85-95.

[Murphy09] C. Murphy, K. Shen, G. Kaiser, „Automatic system testing of programs without test oracles“. In: Proceedings of the eighteenth international symposium on Software testing and analysis,. ACM, 2009., pp. 189-200.

[Peters94] D. K. Peters, D. L. Parnas, „Generating a Test Oracle from Program Documentation“, Proceedings of the 1994 International Symposium on Software Testing and Analysis (ISSTA), ACM Press, 1994, pp. 58–65.

[Richardson92] D. J. Richardson, S. L. Aha, T. O. O'Malley, „Specification-based test oracles for reactive systems“, Proceedings of the 14th international conference on Software engineering, ACM, 1992, pp. 105-118.

[Robinson99] H. Robinson, "Finite State Model-Based Testing on a Shoestring", Talk, STAR West 1999

[Staats11] M. Staats, M. W. Whalen, M. P. Heimdahl, „Programs, tests, and oracles: the foundations of testing revisited“, 2011 33rd International Conference on.Software Engineering (ICSE), IEEE, 2011, pp. 391-400.

[Weyuker80] E. J. Weyuker, "The Oracle Assumption of Program Testing", in: Proceedings of the 13th International Conference on System Sciences (ICSS), Honolulu, HI, January 1980, pp. 44-49.

[Whittaker00] J. A. Whittaker, „What is software testing? And why is it so hard?“, Software, IEEE **17**(1), 2000, pp. 70-79.

c) Bücher, Artikel, Konferenzen: Diverses
[Cristian85] F. Cristian, "A Rigorous Approach to Fault-Tolerant Programming", IEEE Transactions on Software Engineering, **11**(1), 1985, pp. 23-31.
[Gödel31] K. Gödel, „Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I.“, Monatshefte für Mathematik und Physik, **38**, 1931, pp. 173-198.

d) Websites (letzter Test: Juni 2016)
[Bolton] http://www.developsense.com/blog/2012/05/oracles-cant-guarantee/
[MS] https://www.microsoft.com
[V] http://www.v-modell-xt.de
[Wiki6W] https://de.wikipedia.org/wiki/Nachricht_%28Journalismus%29#Stil_und_Inhalt
[WikiAriane] https://de.wikipedia.org/wiki/Ariane_V88
[WikiBug] https://en.wikipedia.org/wiki/Bebugging
[WikiDuck] https://de.wikipedia.org/wiki/Duck-Typing
[WikiDynkin] https://de.wikipedia.org/wiki/Dynkin-Index
[WikiOzon] https://en.wikipedia.org/wiki/Ozone_depletion#Antarctic_ozone_hole

e) Sonstiges, Fußnoten, Anmerkungen
[ISO9126] ISO/IEC 9126, International Standard for the Evaluation of Software Quality
[LB] Diese Programmierweisheiten sind so alt, daß die Herkunft nicht mehr feststellbar ist. Vgl. auch den satirischen Webcomic „Lovelace and Babbage“, http://sydneypadua.com/2dgoggles/
[INVEIG] Lawrence Berkeley Laboratory, Programmierer und Datum nicht bekannt
[Schellekens16] Prof. Bert Schellekens (NIKHEF, Niederlande), persönliche Mitteilung, 2016.