Catch the Bug
Testaktivitäten erfolgreich messen

Ronald Heimberg
QA Systems GmbH

Übersicht

- Motivation
- Die Herausforderung
- Teststrategien aus 61508, 50128, 26262
- Traceability
- Check Points
- Die Essenz

Historie

- Gödel
  - In hinreichend starken Axiomensystemen gibt es unbeweisbare Aussagen. (-> Lügner Paradoxon)
- Turing
  - Kein Algorithmus beweist, daß die Ausführung eines Algorithmus zu einem Ende gelangt. (-> Turingmaschine)
- Church
  - Turing-berechenbare Funktionen stimmen mit intuitiv berechenbaren Funktionen überein. (-> rekursive Funktionen)
- Dijkstra
  - Die Abwesenheit von Fehlern läßt sich nicht beweisen, nur deren Anwesenheit.

Wieviele Fehler?

- 100 %
  - Defects
  - Faults
  - Failures
  - Errors
  - Bugs
- Absolute Anzahl ist unbekannt
  - Folglich sind Prozentsätze hinfällig

Normen

- Was empfehlen 61508, 50128, 26262?
- Übersicht und Vergleich

IEC 61508

Technique	SIL 1	SIL 2	SIL 3	SIL 4
Dynamic analysis and testing	R	HR	HR	HR
Boundary value analysis	R	HR	HR	HR
Error Guessing	R	R	R	R
Input Partition Testing	R	R	R	HR
Coverage entry points	HR	HR	HR	HR
Coverage statement	R	HR	HR	HR
Coverage branch	R	R	HR	HR
Coverage MC/DC	R	R	R	HR
Functional and black box testing	HR	HR	HR	HR
Input Partition Testing	R	HR	HR	HR
Interface Testing	R	R	HR	HR
Forward Traceability	R	R	HR	HR

EN 50128

Technique	SIL 0	SIL 1	SIL 2	SIL 3	SIL 4
Dynamic Analysis and Testing	–	HR	HR	HR	HR
Boundary Value Analysis	–	HR	HR	HR	HR
Error Guessing	R	R	R	R	R
Input Partition Testing	R	R	R	HR	HR
Traceability	R	HR	HR	M	M
Test Coverage for code	R	HR	HR	HR	HR
Statement	R	HR	HR	HR	HR
Branch	–	R	R	HR	HR
Compound Condition	–	R	R	HR	HR
Data flow	–	R	R	HR	HR
Path	–	R	R	HR	HR
Functional / Black-box Testing	HR	HR	HR	M	M
Boundary Value Analysis	R	HR	HR	HR	HR
Input Partition Testing	R	HR	HR	HR	HR
Interface Testing	HR	HR	HR	HR	HR

ISO 26262

Technique	ASIL A	ASIL B	ASIL C	ASIL D
Methods for software unit / integration testing				
Requirement-based test	HR	HR	HR	HR
Interface test (int. test)	HR	HR	HR	HR
Methods for deriving test cases				
Analysis of requirements	HR	HR	HR	HR
Boundary Value Analysis	R	HR	HR	HR
Equivalence Classes	R	HR	HR	HR
Error Guessing	R	R	R	R
Structural Coverage (unit testing)				
Statement	HR	HR	R	R
Branch	R	HR	HR	HR
MC/DC	R	R	R	HR
Structural Coverage (integration testing)				
Function coverage	R	R	HR	HR
Call coverage	R	R	HR	HR

Teststrategien

- Traceability
- Boundary value analysis
- Error Guessing
- Input Partition Testing
- Functional and black box testing
- Interface Testing
- Coverage

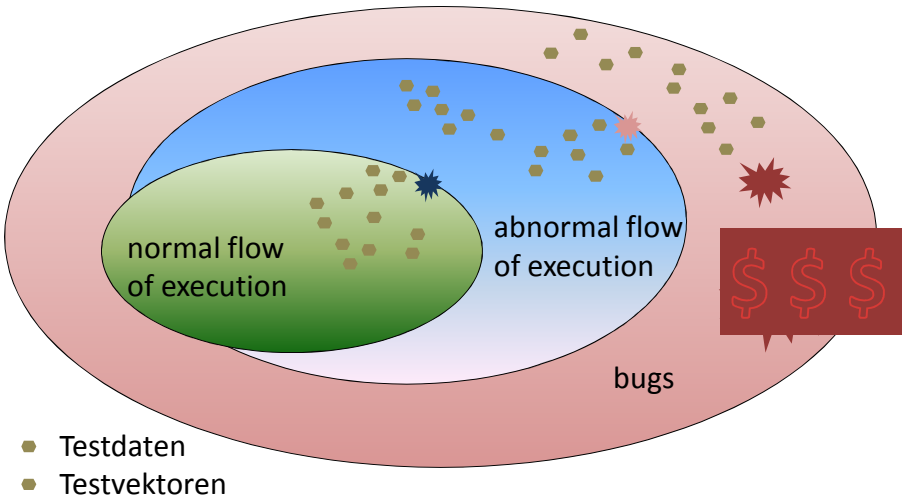
Inhalte der Normen

- Anforderungen und Sicherheitsanforderungen sind elementar und allgegenwärtig
- Rollen für Codieren und Testen
- Eingabe- und Ausgabe-Dokumentation
- Im Detail subtile Unterschiede

Traceability

- Is your requirements repository prepared?
  - get
    - reqID
    - short description
    - software baseline
  - set
    - testID
    - short description
    - software baseline
    - N:M links to reqID
    - test result
- Ideally a data exchange based upon xml

The Sea Of Bugs



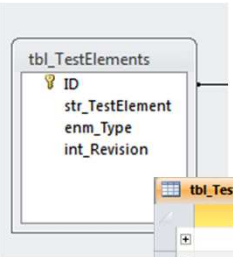
- Testdaten
- Testvektoren
- Eingabedatensätze

Testbeobachtungen

- Tests werfen Fragen auf
  - alle Fragwürdigkeiten sammeln
- Falls kein Bug:
  - Ist die Dokumentation zugänglich?
  - Verständlichkeit der Dokumentation
  - Vollständigkeit der Dokumentation
  - Korrektheit der Dokumentation
  - Konsistenz der Dokumentation
- Falls doch ein Bug: dann erst recht!

Testelement

- Elemente eines Tests:
  - Translation Units
  - Klassen
  - Funktionen
  - Methoden
  - Datenobjekte

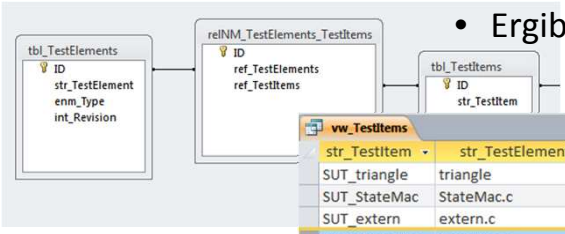


tbl_TestElements

ID	str_TestElement	enm_Type	int_Revision
1	triangle	Function	1
2	StateMac.c	Translation Unit	1
3	extern.c	Translation Unit	1
4	getStateX	Function	1
5	getStateY	Function	1
6	g_State	Data Object	1
7	StateMac::doSomething	Function	2

Testgegenstand

- Gegenstand eines Tests:
  - einzelnes Testelement
  - Menge von Elementen
- Ergibt N:M Relation

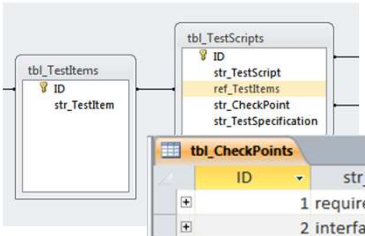


vw_TestItems

str_TestItem	str_TestElement	enm_Type	int_Revision
SUT_triangle	triangle	Function	1
SUT_StateMac	StateMac.c	Translation Unit	1
SUT_extern	extern.c	Translation Unit	1
MUT_StateMac	StateMac.c	Translation Unit	1
MUT_StateMac	extern.c	Translation Unit	1
DDT_StateMac	StateMac.c	Translation Unit	1
DDT_StateMac	extern.c	Translation Unit	1
DDT_StateMac	g_State	Data Object	1

Testskript

- Testskripte:
  - Ausführbare Anweisungen an den Tester
  - Eindeutiger Bezug zum Testgegenstand
  - Teststrategie als Check Point



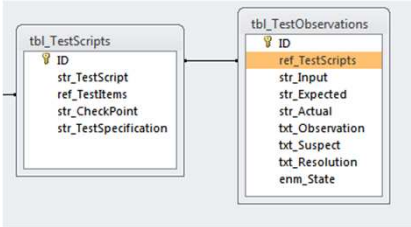
tbl_CheckPoints

ID	str_CheckPoint
1	requirement based test
2	interface test
3	boundary value analysis
4	input partition test
5	extreme values check

tbl_TestScripts

ID	str_TestScript	ref_TestItems	str_CheckPoint	str_TestSpecification
1	test_triangle	1	output partition test	alle möglichen Rückgabewerte hervorrufe

Testbeobachtungen



- Testbeobachtung:
  - Bezieht sich auf die konkrete Ausführung eines Testskriptes
  - dokumentiert die Abweichung, den Verdacht, den Status, die Auflösung

ons

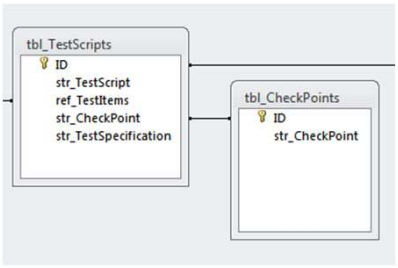
ref_TestScript	str_Input	str_Expect	str_Actual	txt_Observation	txt_Suspect	txt_Resolution	enm_Sta
1	(3,3,4)	isosceles	scalene	der vierte Aufruf der Funktion gibt immer scale	Bug im Algorith	Routine war eine Mock	verified

Check Points

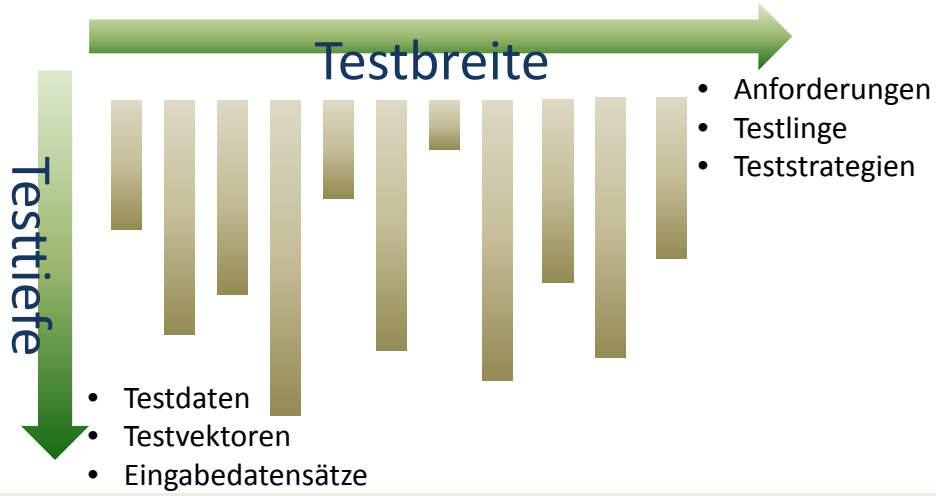
tbl_CheckPoints

ID	str_CheckPoint
1	sanity check
2	functional test
3	input partition test
4	output partition test
5	equivalence classes test
6	black box test
15	requirement based test
16	interface test
17	boundary values analysis
18	robustness test
19	negative test
20	extended functional test
21	error guessing
22	error seeding
23	fault injection test
24	enumerative test
25	extreme values check
26	back-to-back test
27	stand-alone coverage che

- CheckPoints:
  - werden durch Testskripte erreicht



CheckPoints



CheckPoints

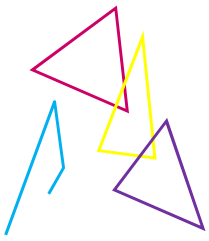
- Dienen dazu, die erforderliche Testbreite zu erlangen
- Veranlassen dazu, die notwendige Testtiefe zu erhöhen

CheckPoints

- Testbreite bspw.:
  - Mehr Anforderungen
  - Mehr Translation Units
  - Mehr Teststrategien
  - …
- Testtiefe bspw.:
  - Mehr Repräsentanten der Äquivalenzklassen
  - Mehr Grenzfälle einer Grenzwertanalyse
  - Mehr Abhängigkeiten zu Daten und Funktionen
  - …

Test Design - Beispiel

- Es liegt ein Programm "triangle" zum Test vor.
  - Input: drei int-Parameter
  - Output: mit den angegebenen Längen läßt sich
    - ein gleichseitiges Dreieck,
    - ein gleichschenkliges Dreieck,
    - ein allgemeines Dreieck konstruieren.



- … Eine einprägsame Spezifikation mit nicht-trivialen Testfällen …

aus:
Glenford J. Myers: "The Art of Software Testing", John Wiley & Sons, 2. Ed. 2004
übrigens: Erstauflage von 1979 …

CheckPoints

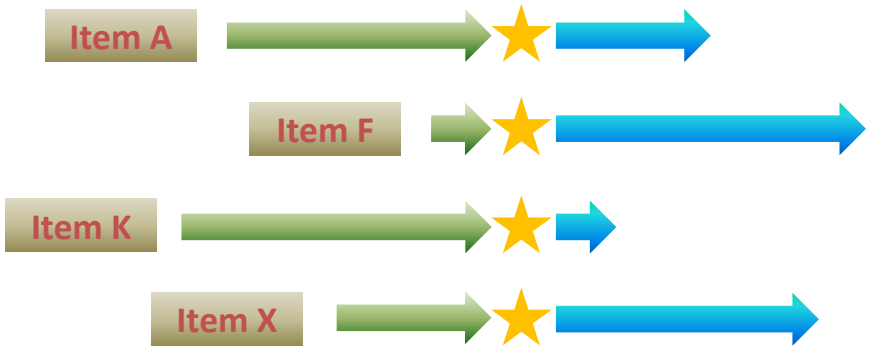
- CP 1: Sind die Voraussetzungen erfüllt?
  - Tools evaluiert, ausgewählt, beschafft, installiert, konfiguriert, geschult
  - Dokumentationen, Anforderungen, Spezifikationen zugänglich und verständlich

CheckPoints

- CP 2: Sind die Testlinge reif für einen SUT?
  - Testling im Single Unit Test ist stabil: behält seine Aufrufschnittstelle, seine Struktur in etwa bei, und ist testfähig

CheckPoints

- Reife der Test Items



CheckPoints

- CP 3: Übersteht der Testling einen *normal flow of execution*?
  - Basisfunktionalität lt. der Anforderungen ist vollständig gegeben

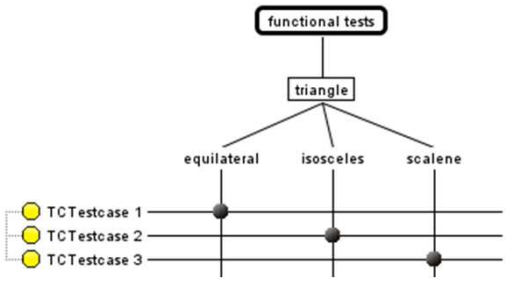
Normal flow of execution

- Eine Funktion, die für drei Seitenlängen bestimmt, ob das Dreieck
  - gleichseitig,
  - gleichschenklig oder
  - allgemein ist.

```
10 typedef enum
11 {
12     scalene,        /* allgemeines Dreieck */
13     isosceles,      /* gleichschenkliges Dreieck */
14     equilateral,    /* gleichseitiges Dreieck */
15 }
16 T_TRIANGLE;
17
18 T_TRIANGLE triangle( int A, int B, int C);
```


Output partition test

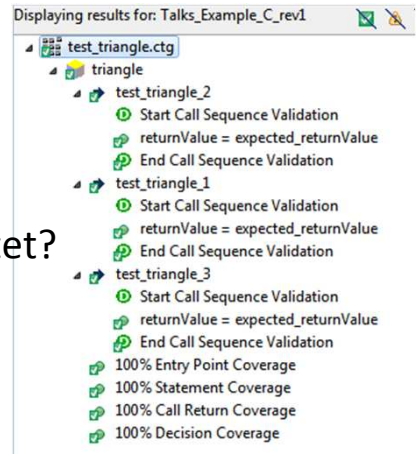
- Anwendung der Äquivalenzklassenmethode:
  - für jeden Rückgabewert einen Testfall



Output partition test

- Testvektoren
  - (4, 4, 4)
  - (2, 4, 4)
  - (2, 3, 4)
- Doch was wurde getestet?

```
10
11 T_TRIANGLE triangle( int A, int B, int C)
12 {
13     int retVal = rand() % 3;
14
15     return (T_TRIANGLE)retVal;
16 }
17
```



Testbreite vs. Testtiefe

- Testtiefe
  - Besser oder schlechter?
  - Testvektor (4, 4, 4)
  - Testvektoren { (i, i, i) | i = 1..INT_MAX}
- Testbreite
  - Besser oder schlechter:
  - Testvektoren (4, 4, 4); (2, 4, 4); (2, 3, 4)

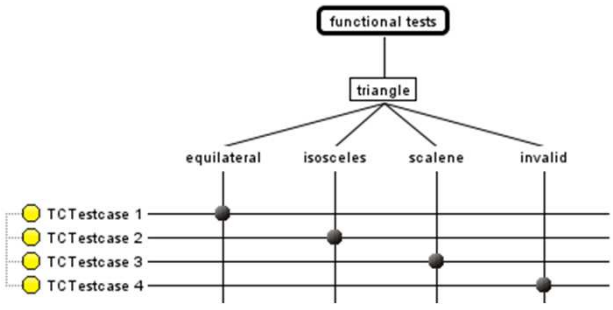
CheckPoints

- CP 4: Übersteht der Testling einen *abnormal flow of execution*?
  - Ungewöhnliche, ungültige Eingabedatensätze werden zulässig verarbeitet

Abormal flow of execution

- Immer noch Äquivalenzklassenmethode:
  - für jeden Rückgabewert einen Testfall

- (2, 3, -1)
- (2, 3, 6)



CheckPoints

- CP 5: Verwendet der Testling anderweitige Daten korrekt?
  - Unerwartete Datenwerte und Datensätze werden robust verarbeitet, korrigiert, propagiert

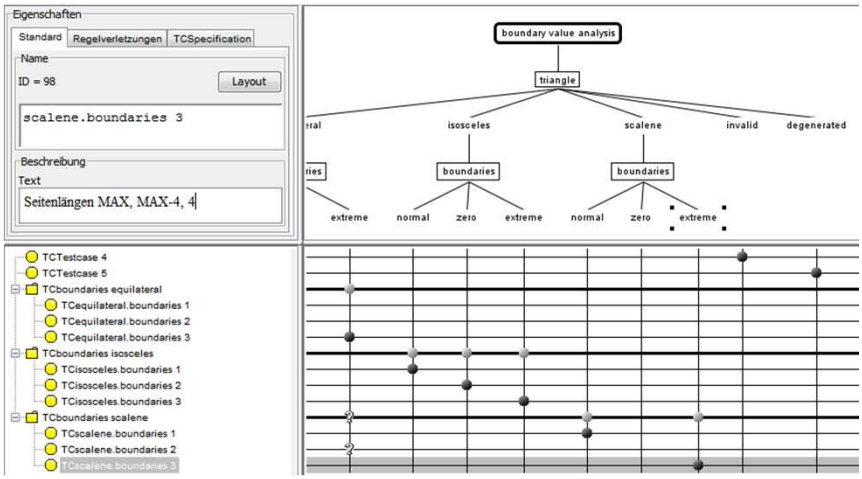
Boundary Value Analysis

- Anwendung der Boundary Value Analysis:
  - grenzwertige Eingabedatensätze

- (0, 0, 0)
- (8, 4, 4)
- (7, 3, 4)

- Permutationen steigern die Testtiefe

Boundary Value Analysis



CheckPoints

- CP 6: Verwendet der Testling anderweitige Module korrekt?
  - Keine ungewöhnlichen, ungültigen Übergabedatensätze, Rückgaben korrekt interpretiert

Interface test

- Aufrufabhängigkeiten müssen eruiert werden
- Sind Grenzfälle des einen Moduls störend für ein anderes (verursacht Fehlfunktion?)

```
33 int isPythagoreanTriple( int A, int B, int C)
34 {
35     if( scalene != triangle(A, B, C)) /* Plausi-Check */
36         return 0; /* kann ja kein rechtwinkliges Dreieck sein */
37
38     if( A*A + B*B == C*C)
39         return 1;
40     if( A*A + C*C == B*B)
41         return 1;
42     if( B*B + C*C == A*A)
43         return 1;
44
45     return 0;
46 }
```


CheckPoints

- CP 7: Spielen mehrere Testlinge korrekt zusammen?
  - Datenzentrierte Tests unter Beteiligung mehrerer Module mit Schreib- und Leserecht auf den Datenobjekten

Die Essenz

- Es braucht kreative, intuitive Tester
- mit Programmierkenntnissen
- Fallstricke der Programmierung sind bekannt
- Dokumentation im Original zugänglich
- Testen in die Breite, dann ggf. in die Tiefe
- Testbeobachtungen dokumentieren
- CheckPoints steuern die Abfolge