Test Design for Reliable Software

ELTE

Thematic Excellence Program

Industry and Digitalisation

Application Domain Specific Highly Reliable IT Solutions

Prof. Dr. Attila Kovács

Dept. of Computer Algebra, Faculty of Informatics,
Eötvös Loránd University

attila.kovacs@inf.elte.hu

Motivation

- With the up-to-date test methodologies we expect 1500 defects after release in 100 000 lines of Java code, where approx. 350 of them are show stoppers [1].
- For safety-critical software it is not acceptable.
- We need better testing.

Concentrate to the specification-based test design techniques

1. Business oriented
   - Use-case testing
   - Business rule-based: decision tables, cause-effect graphs
2. Data oriented
   - EP, BVA, combinatorial methods
3. Behavior oriented
   - State transition testing

Challenges

- Customers expect problem-free software experiences in their preferred channels.
- Business enterprises have to continuously adapt, improve, and deliver competitive, customer-centric solutions.
- IT companies should develop and test with the rise of next-generation technologies: AI, machine learning, IoT testing, big data testing, QA test automation, performance engineering, cyber security testing, etc.

How can we save money and raise SW quality at the same time?

Where to improve?

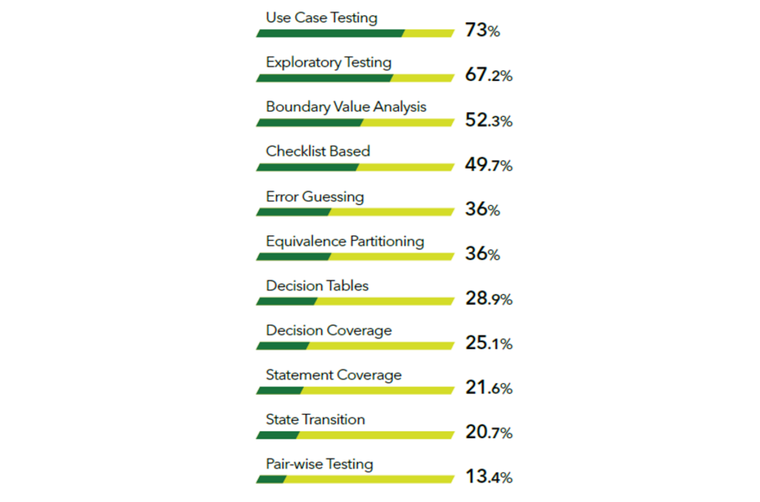


Fig. 1. The testing life-cycle [2].

We need to write more efficient and effective test cases.

- The choice of the right test techniques is critical to achieving a good return on the test investment.

Fig. 2. The mostly used test design techniques [3].

[1] C. Jones and O. Bonsignour, *The Economics of Software Quality*. Addison Wesley, Boston, MA, 2011.

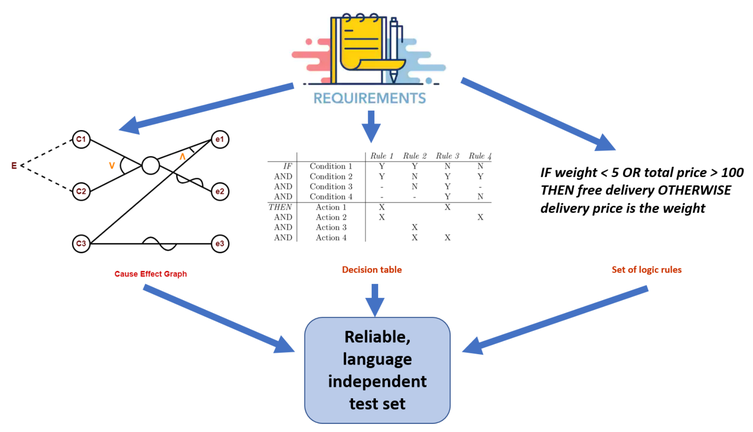
[2] I. Forgács and A. Kovács, *Practical Test Design: Selection of traditional and automated test design techniques*. BCS, 2019.

[3] https://www.istqb.org/documents/, 2018.

[4] I. Forgács and A. Kovács, *Paradigm Shift in Software Testing*. to appear, 2021.

General Predicate Testing [4]

- An extension of equivalance partitioning and boundary value analysis.
- Combines business and data oriented test techniques.



		Rule 1	Rule 2	Rule 3	Rule 4
IF	Condition 1	Y	Y	N	N
AND	Condition 2	Y	N	Y	Y
AND	Condition 3	-	N	Y	-
AND	Condition 4	-	-	Y	N
THEN	Action 1	X		X	
AND	Action 2	X			X
AND	Action 3		X		
AND	Action 4		X	X	

Fig. 3. General predicate testing (GPT).

Advantages:

- Can partly be automated.
- Finds the bugs made by developers or testers.
- In case of a single predicate the number of abstract test cases is linear in the number of atomic predicates.
- In case of a predicate set the abstract tests can efficiently be computed.
- Although the test case minimization is NP-complete, the local optimum is still reliable (has 100% bug revealing capability).
- The resulting test set is not unique and not necessary optimal.
- The method is programming language independent.
- Regarding scalability, the test cases rely strongly on the chosen data decomposition.
- Knowing the software architecture for the tester in advance (functional and data decompositions) much fewer test cases are sufficient.

Example NextDay (Jorgensen)
Given (Day, Month, Year) the next day should be computed. For example,
Nextday(28, 2, 2021) = (1, 3, 2021).

- Pure data decomposition results in 18 test cases assuming that only valid partitions are considered and the library function LeapYear() is correct.
- Knowing the functional decomposition first the function MonthLength(Month,Year) should be developed and tested. GPT results in 4 tests (any month has 28, 29, 30, or 31 days). Then, based on the MonthLength() method, 3 further valid equivalence partitions can be determined. It means altogether 7 test cases for the valid partitions.

GPT is highly recommended for testing logic in safety-critical systems.

Action-state testing [4]

- An extension of use case testing and state transition testing.
- Combines business and behavior oriented test techniques.
- Overcomes the problems of transition infeasibility and state-space explosion.
- It has a textual notation.
- The model building process has two phases: creation and generation.
- Each action step has the form:
  Action => Response STATE newState

Advantages:

- High (over 99%) bug revealing capability.
- Fast model building.
- Visual control with state transition graphs.

Example ATM authentication

```
INITIAL STATE Waiting for Card
valid Card:
insert valid Card => Card recognized STATE Waiting for PIN
    correct PIN:
    enter correct PIN => PIN accepted STATE Authenticated
    incorrect PIN:
    enter incorrect PIN => PIN refused STATE Waiting for PIN
      incorrect PIN second time:
      enter incorrect PIN => PIN refused STATE Waiting for PIN
        incorrect PIN 3rd time:
        enter incorrect PIN third time => PIN refused; card blocked STATE Waiting for Card
invalid Card:
insert invalid Card => Card ejected STATE Waiting for Card
```

Fig. 4. Action-state code for the ATM authentication problem.

- For 0-switch coverage 3 tests are generated, for 1-switch coverage 6 tests.

Together with the all-transition-transition technique [2], the action-state testing is the most effective and most cost efficient technique for testing stateful applications.

Application Domain Specific Highly Reliable IT Solutions project has been implemented with the support provided from the National Research, Development and Innovation Fund of Hungary, financed under the Thematic Excellence Programme TKP2020-NKA-06 (National Challenges Subprogramme) funding scheme.

National Research, Development and Innovation Office Hungary

PROGRAM FINANCED FROM THE NRDI FUND