3

September, 2008

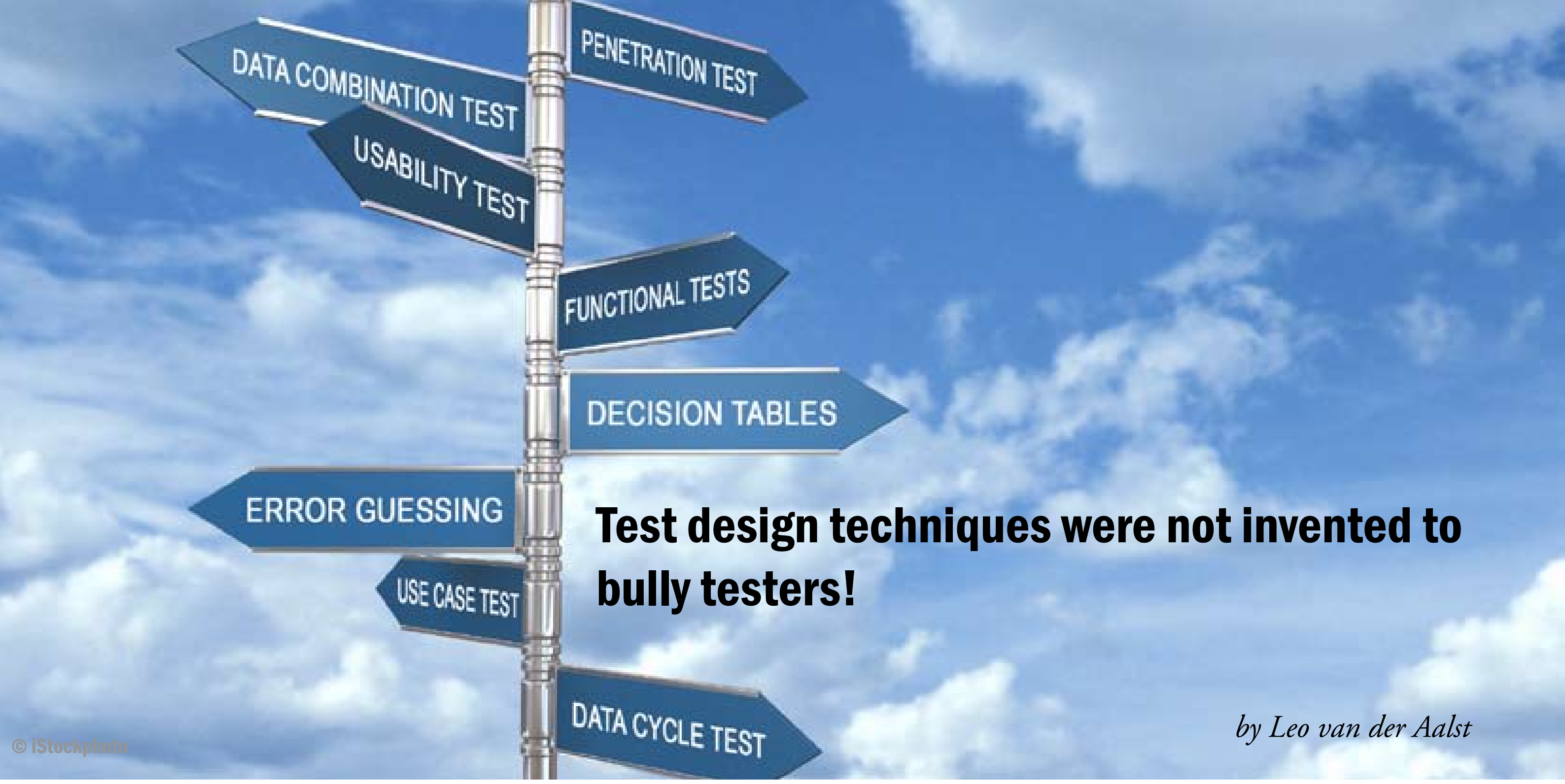
testing experience

The Magazine for Professional Testers

Test Techniques in practice -

Do they help?
Why do we often test without them?

ISSN 1866-5705 www.testingexperience.com free exemplar printed in Germany



Test design techniques were not invented to bully testers!

by Leo van der Aalst

Frequently, clients and even testers complain that using test design techniques is a difficult and time-consuming business. If they can get away with it, they would prefer not using any techniques at all! That's a pity, because these techniques represent the only way to realise the agreed test strategy in a *demonstrable* way. This article provides you with the tools to select one or more suitable techniques.

Substantiate the test strategy with test design techniques

After the test goals and product risks have been established via a product risk analysis, the resulting test strategy should be substantiated, according to the intensity of testing for a specific combination of characteristics and object parts.

> IN MORE DETAIL
>
> - A test goal is a success criterion for the test assignment specified in the customer's language.
> - A product risk is the chance that the product fails in relation to the expected damage if it does so.
> - The test strategy is the distribution of the test effort and test intensity over the combinations of characteristics and object parts aimed at finding the most important defects as early as possible and at the lowest costs.
> - The intensity of testing is light, average or thorough; it is part of the test strategy.
> - Characteristics include amongst others functionality, user-friendliness and security.
> - Object parts are usually the sub-systems of the application software.

Having determined the characteristic to be tested and the test intensity, one or more suitable test design techniques can be selected to create the test cases (See Figure 1: From test goals to test cases, below).

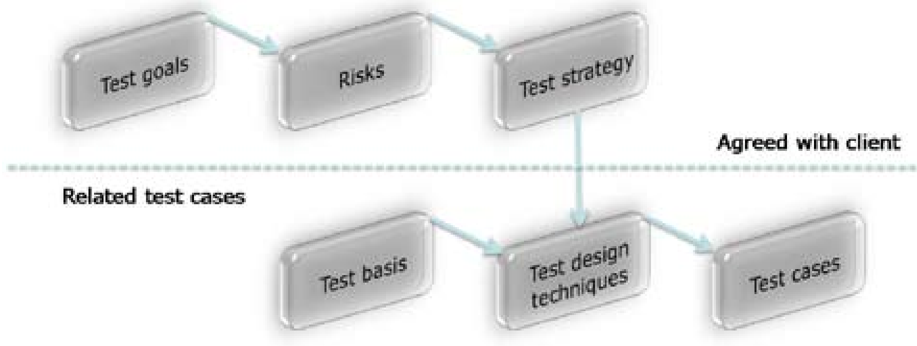


Figure 1: From test goals to test cases

TMap Next [Koomen et al, 2006] describes a large number of techniques and a great many other techniques can be found in books and on the Internet – including some created by testers themselves. Of course, variations of any technique can be also generated.

If an unsuitable technique is selected or no techniques are used, all of the previous steps will have been in vain, and it becomes very difficult to make a judgement as to whether the test goals have or have not been realised – with the attendant risks. For instance, the chance of production disruptions grows - unfortunately still a common situation. And in terms of test governance, test goals cannot be traced through to test cases.

Are test design techniques difficult and time-consuming?

Why is it that clients and testers often consider that using test design techniques is a difficult and time-consuming business? It would seem this is a matter of 'unknown, unloved'!

The quantity and abstract nature of techniques is an important cause of their being 'unloved'.

To avoid overloading the tester such that he 'can't see the wood for the trees', it is better not to teach or explain every single test design technique. Practice has shown that if the techniques are explained well, with examples from a tester's own immediate work environment, the tester 'suddenly' does not feel they are so difficult and sees the benefits of using them. The reason for this is that using examples from the tester's work environment eliminates the abstraction of a technique, and allows the tester to see its practical application and motivates him to put into practice at his workplace what he has learned.

Also some organisations want more 'certainty' and assurance that their testers have an adequate knowledge of test design techniques. Solutions include encouraging or even mandating their in-house testers to acquire formal certification or asking for external certified testers when testing.

Often, when the use of test design techniques is described as time-consuming, people forget that the techniques may be used 'incorrectly'. The aim is not for the tester to design every possible test case, but rather that he selects a specific technique in relation to the selected test strategy - aiming to achieve the highest possible 'defect-finding chance' with the least possible number of test cases. In practice, we find that testers frequently make the wrong choice. As a result, an excessive number of test cases are designed. This is an important cause for considering the use of techniques to be time-consuming.

IN MORE DETAIL

Defect-finding chance
Let's say that you have a 'travel reservation system' with the following parameters and equivalence classes:
Number of days : 8; 15; >15
Amount (euros) : <500; 500-1000; >1000
Membership card: none; silver; gold; platinum
Departure date : workday; weekend; bank holiday
You need 3x3x4x3=108 test cases to test all possible combinations (the complete decision table). If you use the technique 'pairwise testing', you only need 13 test cases (if using the 'Allpairs' tool[1]).

Research conducted by the National Institute of Standards and Technology [Kuhn, 2000] shows that 98% of all defects are found when pairwise testing is used. This is because just 2% of all defects are caused by a problem in the combination of three parameters or more! In other words, 12% of all possible test cases will be enough to detect 98% of all defects in the above example.

Choosing the best possible technique

After the test strategy is determined, suitable techniques must be chosen. This is not always easy - after all, we must take a large number of variables into account:

- characteristic
- test intensity
- test basis
- knowledge and skills of the testers
- labour-intensiveness of the technique.

The flow chart illustrated in Figure 2: Technique selection diagram is a valuable means of making the technique selection, in conjunction with the information in Table 1: Proposed technique for a specific combination of characteristic and selected test intensity, and Table 2: The required test base for a proposed technique.

In Figure 2, the activity 'Take measures' may include:

- Required test basis not available
  - ask designers to adapt the test basis so that the technique can be used
  - ask testers to adapt the test basis so that the technique can be used
  - organise information sessions to achieve a usable test basis.
- Inadequate knowledge and skills on the tester's part
  - train the testers in the proposed technique
  - select another technique because it is a better match with the tester's knowledge and skills.
- Labour-intensiveness disproportionate to the time available
  - make the test less intensive
  - make more time available.

Clearly, the proposed measures must be agreed with the client, since they may have an impact on the agreed result, the risks to be covered, the estimated costs, and/or the planned time.

The technique that is the result of using this method of selection is a suggestion only; there may, of course, be reasons for selecting another technique. The selection diagram is just a tool. Finally, sometimes, a certain technique is imposed on a tester, for reasons of industry regulation or standardisation.

IN MORE DETAIL

Technique is imposed
Not every company, industry or specific application allows a tester to 'freely' select a technique; it may be prescribed. An example from the aviation industry illustrates this situation. Aircraft can only use software that aviation authorities have found to be "safe" for aviation purposes. To this end, the American Radio Technical Commission for Aeronautics (RTCA) and the European Organization for Civil Aviation Equipment (Eurocae) have developed a standard: DO-178B for America and ED-12B for Europe. These standards classify software systems according to the consequences for the aircraft and its passengers if a system should fail. The consequences range from 'no negative impact' (level E) to 'catastrophic' (level A). DO-178B and ED-12B require the coverage type "decision points modified condition/decision coverage (MCDC)" for level A testing. The American and European aviation authorities, the Federal Aviation Administration (FAA) and the European Aviation Safety Agency (EASA), accept this standard for certifying aviation software systems.

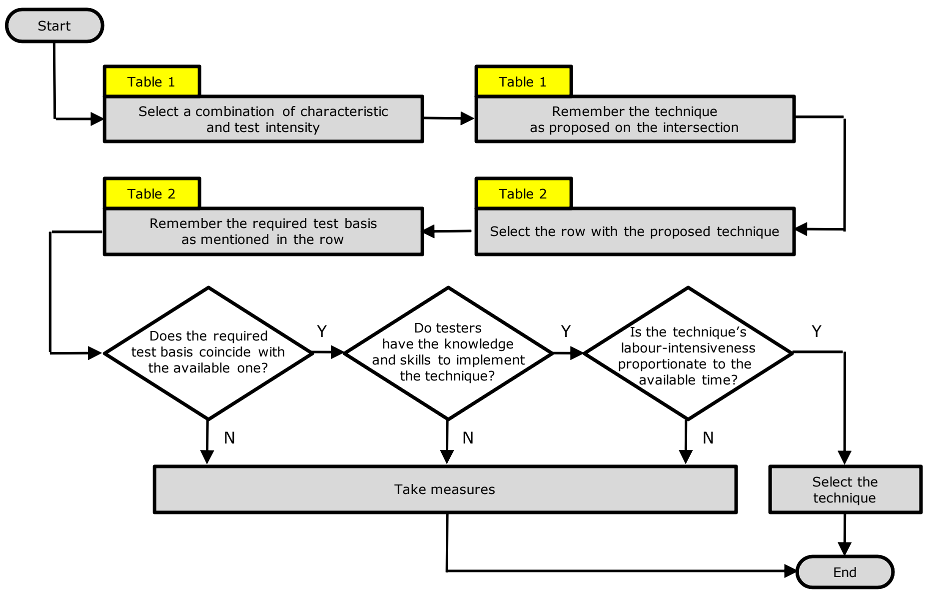


Figure 2: Technique selection diagram

[1] The 'Allpairs' tool was created by James Bach and can be downloaded from http://www.satisfice.com. Another tool is 'Pict33' by Microsoft®, which can be downloaded from http://www.pairwise.org. The DaimlerChrysler tool, 'Classification tree editor', can also be used; this can be downloaded from http://www.systematic-testing.com.

Characteristic	Test intensity		
	Light coverage	Average coverage	Thorough coverage
Manageability	Checklist PCT-test depth level 1 UCT-checklist EG	DCoT-equivalence classes PCT-test depth level 2 ET	DCoT-pairwise testing PCT-test depth level 3
Security	Checklist EG	DCoT-equivalence classes SEM-modified condition/decision coverage ET	DCoT-pairwise testing Penetration test
Usability	UCT-checklist EG	PCT-test depth level 2 UCT-paths	RLT-operational/load profiles UCT-decision points
Continuity	EG	RLT-operational/load profiles ET	RLT-operational/load profiles
Functionality - detail	DTT- condition/decision coverage DCoT-equivalence classes ECT- condition/decision coverage EG	DCoT-pairwise testing ECT-modified condition/decision coverage ET	DTT- multiple condition coverage (+ boundary values) DCoT-N-wise testing ECT-multiple condition coverage
Functionality - overall	DCoT-equivalence classes SYN-checklist (limited) UCT-checklist EG	DCoT pairwise testing DCyT (life cycle of the data) CRUD DCyT (integrity rules) decision coverage PCT-test depth level 2 SYN (prioritised list) SEM-condition/decision coverage UCT-paths ET	DCoT-N-wise testing DCyT (life cycle of the data) CRUD (extra Rs) DCyT (integrity rules) modified condition/decision coverage RLT-operational/load profiles SEM-modified condition/decision coverage UCT-decision points
Functionality - validations	SYN-checklist (limited) EG	SEM-condition/decision coverage SYN (prioritised list)	SEM-modified condition/decision coverage
User-friendliness	SYN-checklist (limited) EG	PCT-test depth level 2 SYN (prioritised list) UCT-checklist	Usability test (possibly in lab)
Infrastructure (suitability for)	EG	RLT-operational/load profiles ET	RLT-operational/load profiles
Suitability	UCT-checklist DCoT-equivalence classes PCT-test depth level 1 UCT-checklist EG	UCT-paths PCT-test depth level 2 DCoT-pairwise testing DCyT (life cycle of the data) CRUD DCyT (integrity rules) decision coverage ET	RLT-operational/load profiles UCT-decision points DCoT-N-wise testing DCyT (life cycle of the data) CRUD (extra Rs) DCyT (integrity rules) modified condition/decision coverage PCT test depth level 3
Performance	EG	RLT-operational/load profiles ET	RLT-operational/load profiles
Portability	Checklist Random sample functional tests Random sample environment combinations EG	Functional regression test Important environment combinations ET	All functional tests All environment combinations
Efficiency	EG	RLT-operational/load profiles ET	RLT-operational/load profiles

Table 1: Proposed technique for a specific combination of characteristics and selected test intensity

Please refer to Table 2 for the meaning of the abbreviations. See TMap Next [Koomen et al, 2006] for comments on the test design techniques.

Comments on the terms used in Table 1:
Portability - functional tests
When testing portability, a random sample of functional tests, the regression tests or all test cases can be executed in a specific environment with increasing test intensity.
Environment combinations
Testing portability determines whether the system runs in various environments. Environments may consist of different parts, such as hardware platform, database system, network, browser and operating system. If the system needs to be able to run on 3 (versions of) operating systems under 4 browsers (or browser versions), you already have 3 x 4 = 12 environment combinations to test.
Penetration test
The penetration test aims to find gaps in the system's security. It is usually executed by a so-called 'ethical hacker'.
Usability test
A test in which the users simulate business processes and test the system. Statements about the test object's user-friendliness are made by observing the users during the test. A specifically configured and controlled environment, which includes e.g. video cameras and a room with mirrored glazing for observers, is also called a usability lab.

Technique	Test basis								
	All types of test basis	Individual conditions or decision tables, without structure	Structured functional specification (pseudo code)	CRUD matrix, data integrity rules	Structured description of business or operating processes	Operational profiles, load profiles	Input and output specifications, business rules	Input and output specifications, attribute descriptions	Use cases
Checklist	x	x					x		x
decision table test (DTT)	x	x	x						
data combination test (DCoT)	x	x	x	x					
error guessing (EG)	x	x	x	x	x	x	x	x	x
exploratory testing (ET)	x		x			x	x		
elementary comparison test (ECT)		x	x						
functional tests		x					x		
Data cycle test (DCyT)		x		x					
Environment combinations		x					x		
Penetration test									x
process cycle test (PCT)		x		x	x	x		x	x
real life test (RLT)		x				x			x
semantic test (SEM)							x	x	x
syntactic test (SYN)							x	x	x
usability test								x	x
use case test (UCT)					x	x		x	x

Table 2: The required test base for a proposed technique

References

- [Koomen et al, 2006]
  Koomen, T., Aalst van der, L., Broekman, B., Vroon, M. (2006), TMap® Next for result-driven testing, Tutein Nolthenius, 's-Hertogenbosch, Netherlands, ISBN 90-72194-80-2, www.utn.nl
- [Kuhn, 2000]
  Kuhn, D.R., Wallace, D.R., (2000), Failure modes in medical device software: an analysis of 15 years of recall data, National Institute of Standards and Technology, Gaithersburg, MD 20899 USA, http://csrc.nist.gov/staff/Kuhn/final-rqse.pdf
- TMap® is a registered trademark of Sogeti Nederland B.V.

Biography

Leo van der Aalst has more than 20 years of testing experience and developed amongst others services for the implementation of test organisations and for test outsourcing.

He is co-author of TMap Next, designed the EXIN tasks for TMap Next certification and holds a readership position for quality&testing at the 'Fontys Hogescholen'.

Besides all this, Leo is a much sought-after teacher of test training, a regular speaker at national and international conferences, and he is the author of several articles.