Issue 20, Free Print Edition

QUALITY MATTERS

WWW.QUALITY-MATTERS.ORG

CONVERSATION WITH AN AI

By OLIVIER DENOO

CRACKING THE CODE OF SMART TESTING: STRATEGIES THAT WORK

By RENE VAN VELDHUIJZEN

TEST DESIGN, ITS VALUE AND THE AI SOLUTION

By Drs. ERIK VAN VEENENDAAL

IMPLEMENTING THE FIVE MOST IMPORTANT BEST PRACTICES OF QUALITY ASSURANCE

By GERIE OWEN

TEST DESIGN, ITS VALUE AND THE AI SOLUTION

By Drs. ERIK VAN VEENENDAAL, Bonaire

ERIK VAN VEENENDAAL
Improve IT Services BV, Bonaire

Drs. Erik van Veenendaal (www.erikvanveenendaal.nl) is a leading international consultant and trainer, and a recognized expert in the area of software testing. He is the author of a number of books and papers within the profession, one of the core developers of the TMap testing methodology and the TMMi test improvement model. Erik is a frequent keynote and tutorial speaker at international testing and quality conferences. For his major contribution to the field of testing, Erik received the European Testing Excellence Award, the ISTQB International Testing Excellence Award and holds a roll of honor of the TMMi Foundation.

Test design techniques have been around for decades. Despite the clear benefits they bring, such as increased defect finding capability and levels of coverage, their uptake is still relatively limited. What are the reasons that we are not all using test design techniques? How can AI support the testing community during test analysis and design and ensure that we will finally start using the available test design techniques?

THE ART OF TESTING

Software testing has been referred to as an "art", amongst other things, for many years (Myers, 1979). Test techniques, also referred to as test design techniques, make a part of the exercise mechanical in that the production of the test cases follows a defined process and set of rules. However, the overriding need for test techniques, is the need to achieve our purpose. The purpose being to identify defects and provide objective and measurable tests and results to allow users to make an objective decision about the likely impact of taking the system into live operation.

This paper has been produced to discuss some of the issues surrounding the need for, the use of, advantages and disadvantages of test techniques and ultimately how AI can play a role in their uptake. Test design is the activity where test cases are derived and specified, preferably using test techniques. Test techniques come in many shapes and sizes, some formal some not, some dynamic and some static. Inevitably, the focus will be drawn to the dynamic test techniques and activities. With them the identification of test conditions and specification of test cases is most strongly supported.

SYSTEMATIC VS. NON-SYSTEMATIC TECHNIQUES

As the heading suggests, the distinction is in the level of formality. Systematic techniques (e.g., black-box techniques and white-box techniques) are based around a set of rules that make the activity mechanistic, repeatable and measurable. Non-systematic techniques are based on the experience of the tester, either at domain/product level or within the test function. Identifying test cases using non-systematic techniques has gathered credibility using techniques such as exploratory testing. These techniques seek to put some process rigor around testing using experienced testers carrying out the testing. The biggest drawback with these techniques is the lack of documentary evidence and therefore re-usability and repeatability of the tests.

WHY USE TEST TECHNIQUES?

By using test techniques consistently, they will become an integral part of the test process. Why use test techniques at all? They typically provide the following benefits:

- Increased defect finding capability
- Objectivity, guarantees a certain level of coverage
- Ability to reproduce tests

The decision regarding which test techniques to use is largely dependent on level of risk. Risk identification, analyses and mitigation are a fundamental part of testing. The identification and analyses of risk provides a focus to the testing, and mitigation is achieved when the tests, with the correct level of coverage, are executed. The simplest techniques to apply, equivalence partitioning and boundary value analysis, will provide a start point for the test scoping activity. These are the easiest to understand, and are perhaps the most directly relevant to most users. Exploratory testing and error guessing (experienced-based techniques) can be utilized to provide a wider range of tests quickly. Many of the test design techniques have been developed to find defects of a certain type. The test techniques chosen should be focused on the type of defects that the testers are expecting to find, or that should be looked for at any given testing. The early use of test techniques can also act as a great static testing technique as they will identify defects in the specification.

ADVANTAGES/DISADVANTAGES

The decision how to use techniques and which ones is directly related to the risk of the products being tested. Initially a basic view of both the advantages and disadvantages must be taken.

ADVANTAGES	DISADVANTAGES
Objectivity	Requires training
Formal coverage measures	Time to implement - culture change
Early defect finding	Everyone must be 'bought in"
Traceability	Not seen as useful for all applications
Coverage independent of the tester	Take more time than less formal test design
Way to differentiate test depth based on risks using different techniques	Do not cover all situations (experienced-based testing still useful)
High level of re-use, including re-usable testware for regression testing	Little use of domain and product knowledge of tester
Repeatability and Reproducibility	
Audit trails	
Higher defect finding capability	

Table 1: Advantages / Disadvantages Test Design Techniques

Much is espoused in testing about the need to ensure the activities are measurable, controllable and repeatable. Many techniques and supporting tools have been developed to allow testers to meet those objectives. The use of test techniques is methodical and mechanical for the core tests, but used properly they are flexible. Used together formal and informal techniques create an entire test package. Testers typically have to document test cases whichever approach is taken, why not take advantage of what is already in place? Re-inventing the wheel is a tedious and time wasting exercise.

Test techniques provide an understanding of the complexities imposed by most systems. The use of techniques forces testers into thinking about what they test and why they are testing it. In many cases, techniques identify a level of coverage that would otherwise be a mystery. Techniques will not provide a mechanism to exercise 100% test coverage. However, if information regarding the level of coverage to be achieved is available, then objective decisions can be made, based on the risk of the system under test and on which tests to leave out.

Test techniques, both formal and informal, should be an integral part of everyday testing life. If testing is to be taken seriously, the testing function must take itself seriously and ensure its achieving its objectives. Test techniques provide the framework to increase defect finding capability and measure coverage more objectively.

STATE-OF-THE-PRACTICE

It has been 30 years since I co-authored TMap introducing test techniques to the Dutch speaking testing community. ISTQB Foundation, teaching a set of test techniques, has been around for more than 20 years and today more than 1.000.000 are ISTQB certified. Based on this, one would expect that almost every organization and almost every tester is developing test cases using test design techniques. However, based on personal observations, all the books and training have not led to a large and successful uptake. At best, testers understand and use the principles of the test techniques. Many claim to the apply exploratory testing, but in reality they are just doing error guessing.

The main reasons for this situation are twofold. The first one being time. Testing almost always lacks resources, is pressured and taking place within strict deadlines. With testing, time is simply a scarce resource. Test techniques simply take (a lot of) time to implement, it is a change from the current way of working. Also applying them typically takes more time than test design based on domain and product knowledge. The second reason is knowledge and skills. Applying test techniques as not as easy as it looks at first instance; the small exercises in ISTQB Foundation do not resemble the complexity of real-life projects. Practical hands-on training is needed where one learns how to design test cases using test techniques based on the real project requirements and specifications. Not many organizations invest in this second step of learning and as a result practical skills on how to apply test techniques, including experienced-based techniques, are most often lacking.

THE SOLUTION: ARTIFICIAL INTELLIGENCE

Today, at least according to some so-called experts, AI is the solution for almost everything. For sure, AI is great new technology, that will Impact our way of working and provides the option to make testing more efficient and when used wisely also more effective. Maybe we will finally get the time to do all the testing we want to do, and we can better manage the ever growing complexity and requirements for product quality. However, and of utmost importance to always remember, AI is "just" a technology. In business management, a highly popular framework is the People, Process, Technology framework (also known as the PPT-framework). It refers to and exhibits how the balance of people, processes, and technologies drive successful organizational change, improvements and re-engineering. In other words, to apply AI (technology) successfully, you need to know and understand what you are doing (process) and be able to evaluate and use it wisely (people). The process part in this case refers to the test (process) maturity of the organization. It makes sense to only apply AI to test process areas that you have already mastered. "If you don't know what you doing, don't just do it with AI."

There are many testing areas where AI can play in a role in the future, but it is recommended to focus on where it is easy to start and get added value today. Test analyses and design is an area where most organizations have an understanding and possess knowledge and skills. Can AI support the implementation of test techniques and finally bring the listed advantages, that we were not able to (fully) achieve in previous decades?

TEST ANALYSIS AND AI

The test design activity typically starts with test analysis where the test basis (requirements, process flows, user stories, acceptance criteria, etc.) is analyzed. The test basis is reviewed by the tester to determine its testability; can test cases including expected results be derived from the test basis. This is an activity where AI can provide a huge support, both in terms of efficiency and effectiveness. Basically by just "asking" your AI LLM tool, e.g., ChatGPT, to do this for you. Asking in the world of AI means prompt engineering. Prompt engineering is the process of crafting and refining instructions (prompts) given to generative AI tools to elicit the desired output. It is new to most testers, but acquiring this type of knowledge and skills is essential for testers in today's AI-world. Prompt engineering is rarely a one-and-done task. Often, the initial prompt may not yield the desired result, requiring multiple iterations to fine-tune it. An effective AI prompt generally includes these key elements: task, context, input data, output Indicator, and optionally, examples, role, tone, and formatting. The prompt should clearly define what action the AI tool should perform (task), provide relevant background information (context), specify the data to be processed (input), and indicate the desired format of the output. Examples, in the format of attachments, can guide AI, while role and tone can help shape the output's style.

Back to test analysis and the task that a tester typically performs: reviewing the test basis, e.g., as part of a backlog refinement session. Below is an example of a prompt that can be used for test analysis.

As a software tester [Role] you are reviewing a set of requirements specification [Task]. Review the requirements specification "name spec" using the rule set as attached [Input]. Organize the issues found by requirement in a tabular format (state requirements number, the issue found, the rule violated and indicate major or minor) [Output]. Once the identified issues are clarified, the tester shall be to design appropriate test cases and test scenarios [Context].

And within seconds, we are provided with an answer (issue list):

Req. No	Issue Description	Rule Violated	Severity
01	Business goal stated as system requirement	Testable	Major
01	Not under system or team control	Technically Achievable	Major
01	Value not directly tied to implementation	Valuable/Purpose	minor
02	Subjective language ("more smoothly than and other watch")	Unambiguous	Major
02	Not measurable or verifiable	Testable	Major
02	Multiple ideas in one statement	Compound	Major
02	Incorrect grammar ("waked up")	Language	minor
03	No business or technical justification provided	Rationale	minor
03	No priority assigned	Priority	minor
04	Uses "will" instead of "shall"	Language	minor
04	No priority specified	Priority	minor
04	No rational given	Rationale	minor
..	...	.....	....

Figure 1: ChatGPT output example for test analysis

The AI tool (technology) provides us with an answer, the rule set (process) makes the output less ambiguous and as a tester (people) you will evaluate and possibly adapt the output. Overall an efficiency and effectiveness gain. Note that based on this review log, one may not be entirely happy with the severity classification and as a result improve the rule set by adding the typical severity to each rule. As such based on the AI output, the process was improved and thereafter better input was provided to the AI tool to perform its task.

TEST DESIGN AND AI

Now to test design and the task to develop test cases using test techniques. Below is an example of a prompt to be used for test design. From practical experience, it can basically be used this for any given test technique. Providing a description or reference for the technique to be used to get consistent output is recommended. The same goes for providing an output test specification template or a previous test specification example.

As a software tester **[Role]** you are developing test cases **[Task]**. Design test cases based on the requirements specification "name spec" using equivalence partitioning (description attached) **[Input]**. List the test conditions (both valid and invalid) and the test cases required to cover all partitions. (including objective and expected results). Use the test specification template (attached) [Output]. Once the test cases are designed, you shall be able to executed them in a structured and repeatable manner **[Context]**.

And again within seconds, we are provided with an answer (table showing the EC's to be covered and set of test cases). Quite often, despite using a more or less standard prompt, one will have to iterate to achieve the desired output.

Attribute	Type	Valid ECs	Invalid EC's
Card type	Set	EC1: Visa Card, Mastercard, American Express, Visa Delta	EC2: Blank (E109), EC3: Other value (E100)
Card Number	Numeric	EC4: Numeric	EC5: Blank, EC6: Non-numeric, EC7 Fails algorithm (E102)
Cardholder Name	Alpha	EC8: Non-blank, alpha ≤ 30 chars	EC9: Blank (E109), EC10: non-alpha (E300): >chars (E300)
Valid To Date	Numeric YY	EC 12: Numeric current year ≤ YY ≤ 2030	EC13: Blank (E109), EC14: Non-numeric (E103), EC15: < current year, EC16: > 2030 (E104)
Company ID	Alpha	EC17: Non-blank, alpha, registered	EC18: Blank (E109), E19: non-alpha, E20: Unregistered (E200)

Figure 2: ChatGPT output example for test conditions (based on equivalence partitioning)

TC ID	Objective (EC's covered)	Card Type	Card Number	Cardholder name	Valid To Date	Company ID	Expected Result
01	Verify all valid inputs (EC1, EC4, EC8, EC12, EC17)	Visa Card	4111 1111 1111 1111 (valid)	JohnSmith	2025	98775	Processing continues, R02
02	Verify blank card type rejected (EC2)	(blank)	4111 1111 1111 1111	JohnSmith	2025	98775	E109 (blank not allowed)
03	Verify unknown card type rejected (EC3)	ABC Card	4111 1111 1111 1111	JohnSmith	2026	98775	E100 (card type unknown)
04	Verify blank card number rejected (EC5)	Master card	(blank)	JohnSmith	2027	98775	E109
05	Verify non-numeric card number rejected (EC6)	American Express	4111 ABCD 1234 1111	JohnSmith	2028	98775	E101 (must be numeric)
06	Verify incorrect card number rejected (EC7	Visa Delta	4111 1111 1111 1112 (fail)	JohnSmith	2029	98775	E102 (incorrect number)
07	...	.....	.....	.....	.....	.....	....

Figure 2: ChatGPT output example for test cases (based on equivalence partitioning)

The AI tool develops and designs test cases using the rules of a specific test technique. Of course, the testers need to evaluate the output and will often change or add one or more test cases, e.g., based on domain/product knowledge. Can AI support the implementation of test techniques, and finally bring the listed advantages that we have not been able to (fully) achieve in previous decades? Based on personal experiences in real-life projects: **YES!**