Where Testing Creativity Grows

v1.2.5

Abstract - Good effective software testing requires methodical hard work in combination with a creative effort.

The end users of software will, intentionally or unintentionally, be creative; therefore testing needs to be multi-faceted in order to anticipate potential issues. Manual and automated testing is a lot more than comparing actual result to expected result; it is about learning things, pinpointing functionality gaps, and exposing implicit (non-)functional requirements.

If we test in the same way all the time, we will find less bugs and risk to become less motivated. Therefore, from time to time we need to take one step back and get new perspectives, in order to do new things and the unexpected.

Building a generous environment with trust and tolerance will enable creative ideas; by using group dynamics, we can take advantage of the strong and weak sides of each team member.

It is not always possible to find the most important defects or gather the most useful information, so we all need stimulation in order to have a better chance of great performance.

Besides well-known methods like pair testing, brainstorming, Six Thinking Hats and mind mapping; approaches like analogy, cinema testing, "the opposite-method", provocative operators etc. can inspire your creative mind. Using different and new approaches is a productive way to train creativity in the software testing discipline.

The creativity will be manifested in new test strategies and techniques, ingenious test cases, and fast problem solving.

It will in addition be even more fun to work with software testing.

Rikard Edgren

Qamcom Karlstad

rikard.edgren@thetesteye.com

Software Testing Creativity

This paper is written because I find these ideas interesting and true: *"Testing is an extremely creative and intellectually challenging task"*[1], "*People are the most important asset of any company"*[2], *"Testing is all about information"*.

There are a lot of different aspects of testing, and the appropriate proportion of creativity will vary; but in this paper the focus is on the creative part.

Software testing must be one of the most creative professions. Compare for instance with the idea that programming is creative; "which takes more creativity: creating the error or finding the error?"[3] The comparison is a bit strange, but the essence – it is difficult to find errors you don't know about – is more or less valid regardless of which type of project you are working on.

But there are not a lot written about this topic in the context of software testing; creativity is often seen as a good thing, but seldom described in more detail than 'come up with new ideas' or "generate ideas and see possibilities"[4].

Maybe it's the old Zen wisdom: "those that say do not know, and those that know do not say" that I will have to disregard here.

And sometimes creativity is mentioned as a natural talent for some people, or that exploratory testing only should be done by experts[5]. Or is creativity only for children that would add an imaginary pair of new eyes[6] in her big sisters room?

Creativity should be for the everyday tester that while executing a test script comes to think about an old bug and clicks on the wrong button when trying the idea.

Creativity has been defined in many ways, some of them appropriate for the software testing context:

> "Creativity, it has been said, consists largely of re-arranging what we know in order to find out what we do not know."[7]
>
> "Creativity doesn't create something out of nothing but, rather, recombines ideas that already separately exist."[8]
>
> "The process of having original ideas that have value."[9]

When applied to software testing, the last definition will come in effect very often, since

1. producing or executing test ideas is mostly a new thing, for that specific context
2. the result of the test is useful information regardless if it is exposing a bug or not

The only common testing truth I know of is that all tests can't be performed. To perform really good tests it is important to know which test cases *can* be run. To find out which tests that can be run; and which tests that are most important to run in the given context; the software tester needs to use his or her creativity to generate more test ideas or methods. A tester combining competence and creativity will have a good chance to decide which tests are the most important to run[10].

An example that shows the complexity of a pretty simple program is the classic triangle exercise[11], which has given rise to a lot of different number of suggested test cases, not including the additional parameters noted by Collard[12], and not including parameters like performance, usability, data formats, environment, installability, localization etc. The context defines which tests are appropriate; the creativity is needed to get there effectively.

When defects are encountered, the software tester needs to use his/her creativity to nail down the issue in the best possible way. When verifying a bug fix or an important build, creativity is needed to exploit more chances to destroy the product.

Sometimes it seems assumed that testing is not a lot more than comparing the product with its requirement. If so, I would propose that the testing group should focus on all those implicit requirements that are not written (e.g. combinations of requirements). Many explicit requirements can be verified by unit/system tests, and validated by acceptance tests/use cases.

And in between these levels, there is probably a huge area of unspoken requirements, things that many users expect of the product, but hasn't been specified in detail[13]. Creative testers should try to anticipate the customers (un)intentional ways of using the product.

This is where manual testing[14] often fits best; and for this testing to be effective, it should be creative.

It could be argued that a list could be compiled with all errors that might happen, but besides the fact that this will take an awful lot of time, we need to bear in mind that each software project will have its own specifics that the testers need to adapt to. The most important tests need to be figured out for each occasion. To find out how to perform these tests in the most effective way, I believe creativity is needed, in combination with methodical hard work.

So a main reason for testing creativity is speed; a creative tester will find important things a lot faster, especially if using knowledge to prioritize tasks. But we should not see creative testing as something that you do because you are in a hurry, even though it sometimes might be the fastest way to reach your goals. It is about reaching further than the standard tests.
Creativity in testing is a luxury you have to afford.

An Environment for Testing Creativity

It is important to be prepared to be creative. Trust that you will come up with new ideas; and that many of them will be very good. A creative mind-set is the basis.

There has been research performed by Amabile: "The 6 Myths Of Creativity" [15] describes her list of ways that won't give more creativity:

- Creativity Comes From Creative Types
- Money Is a Creativity Motivator
- Time Pressure Fuels Creativity
- Fear Forces Breakthroughs
- Competition Beats Collaboration
- A Streamlined Organization Is a Creative Organization

In order to be truly creative, a creative environment is needed. Sahlin[16] makes a basic set of environment characteristics that isn't mind-gobbling, but makes a lot of sense, also when applied on the software testing profession: generosity, a sense of community, qualifications, cultural diversity, trust and tolerance, equality[17], curiosity, freedom of spirit, small scale[18].

Some characteristics are worth to look more carefully at from a testing perspective:

Qualifications

Knowledge is needed in order to take fruitful leaps into the unknown. Knowledge of test techniques makes it a lot easier to apply effective testing. Knowledge of the technology makes it a lot easier to come up with test ideas. Knowledge of the domain the software will be used in will make it a lot easier to find the most relevant bugs. Knowledge of the product will make the testing faster, and will help you find interesting interacting areas.

Knowing the people producing your software will hopefully make everything better.

Diversity

Different points of view can give radically new ideas when they blend. If all testers found the exact same things, there would be a lot of double work, and no synergies.

By using group dynamics, we can take advantage of the strong and weak sides of each team member. For example it might be good to have a member of the test team that always only does what is stated in the online help.

Can a testing team be too diverse?

Trust & Tolerance

If the testers aren't trusted, they will not do a good job. To be creative involves doing new things, which requires trust.

Patience to wait for exciting things to happen is needed; trust that a lot of time also needs to be wasted. Creativity under pressure is difficult.

Creativity in general is connected to allow oneself to make mistakes. This is especially easy in software testing, since mistakes can mimic end users behaviour, and thereby expose issues. It is not uncommon to start with a mistake and by that come up with a new idea.

"If you're not prepared to be wrong, you will never come up with anything original."[19]

In a test environment with detailed test scripts, it must be OK to side-step the instructions[20], as well as follow them rigidly. Creative testing is about doing things differently, so different approaches should be embraced.

A way to disable the trust would be to perform creativity tests, where you try to measure how creative people are.

Humour[21]

Humour is often about looking at things from a different perspective, to combine things that shouldn't be. Laughter is fun and generates a good atmosphere where creativity can prosper.

That is why you can laugh at developers' mistakes, but also on missed points in a bug report, or the occasional duplicate bug that you already reported yourself.

But most important: If it is fun to work, we do a better job.

Discipline

It must also be noted that the best ideas often come after a lot of hard work:

> "the lightness of inspiration tend to go hand in hand with hard work"[22]

So a creative environment can't be a place where you just do what you feel like. It takes discipline to learn the product good enough, to learn testing techniques; and to execute also the boring test cases needed.

"A total engagement in a complex process"[23], not far from Csikszentmihalyis Flow[24], seems applicable for software testing at its best.

The ingredients in this creativity recipe seem simple, but still there aren't many genuinely creative environments. Sahlin concludes that the unpleasant reason for this is that we, as humans, are driven by the seven deadly sins[25]. I believe that most of us somewhere have lost the most important ingredient: intrinsic motivation[26].

Testing Creativity in Action

After the last fixes before release of a product it might be a good time for regression testing. At the start-up of a project it is probably the right time for creativity; it is too early for dynamic testing, but the right time to define the best test strategy.

All the time in between, there is room for creativity within the testing effort. I believe testing creativity is best used and trained when practiced all the time.
A common case is when trying to reproduce a bug. To see if it still exists for some circumstances (and which those could be) requires creativity.

The right time for creativity can also be answered by examples of instances where creativity might not be appropriate:

- When there are coding conventions for your automatic tests
- When a bug is to be described so it will be found by others
- In a strictly regulatory environment when there is no time besides running specified scripts

Used appropriately, the creativity results in a good bugs or new information that will help the project. And information is what it is all about.

This is a list of important things that are difficult to find without being creative:

- A test strategy that fits with time schedule and resources[27]
- Simplifications of test case coverage (Pairwise only works sometimes)
- Usage paths that aren't covered by specified use cases
- Usage of product in a completely different domain
- Bugs appearing under very special circumstances
- Uncommon platform configurations that end users will have
- Combinations of seemingly unrelated functionality

Another type of combination is the ability to combine details with "the whole picture", and to do it repeatedly.

"Knowledge seems to be the raw material for creativity"[28], because without knowing anything about the product to test, or the system it is running in, or test methods, or typical bugs; there won't be a lot of good test ideas executed.

In most cases the creative ideas comes by combining knowledge from different areas[29]. Thinking "what if..." renders a test that will reveal new information. But sometimes we need stimulation in order to find the most important defects and gather a lot of useful information. This is a list of techniques that often generate more creative ideas:

Improvisation

This is the basic creativity method used for many, many years in software testing. The skilled improvisation is the basis for Exploratory Testing[30]; and its least formal and most improvisational part: Ad Hoc Testing[31].

The only thing that is needed is freedom to extend tests and try out things. To do that better, you need a lot of other testing skills as well[32]. There is a lot of room for learning things with this approach, and learning things gives creativity.

Suitable for: anytime.

Pair testing[33]

Two testers join at one machine. One runs the program, the other one comments and documents tests and bugs.

If it works well for programmers, why shouldn't it work well for testers?

Suitable for: exploratory testing, testing education.

Cinema testing[34]

This is basically an extreme of pair testing, where several testers sit in the same room and look at the same screen.

Suitable for: UI prototype review, education, team building.

Brainstorming

A group of people generates ideas; all ideas are welcome[35].

Suitable for: generating test ideas, high-level review of test effort.

Six Thinking Hats[36]

In group, use six different ways of approaching an area: white – information & facts; red – intuition and feelings; black – logically negative; yellow- positive, possibilities; green – creative; blue – process, meta-perspective.

Suitable for: analyzing and discussing problems and solutions, reviews/inspections.

Provocative operators

In group, say "po", and then state a sentence that might have nothing to do with current subject. This is Edward deBono's term for helping the mind by giving some starting points to build your knowledge around.

A similar way is to use random stimuli, e.g. by using "Random article" at Wikipedia.

Suitable for: occasions when testing is boring and doesn't result in finding a lot of issues.

Analogy

Analogy can be used descriptive or as inspiration.

Is software testing a lot like “Find Five Faults”, but with the big difference that you don’t know how many faults there are; and you don’t have a key.

One can look at a different area to see how problems are solved; maybe a football huddle isn’t appropriate when the Beta build arrives, but what about …?

Suitable for: understanding things or generating new ideas.

“The opposite” method

Take the opposite view of the standard.

For instance you could state “How can our software be as bad as possible” (Can’t start it; Can’t Install it; Can’t Find it) or “How can we test as bad as possible” (Don’t Test; Don’t Look for Errors) has a good chance of generating insight of what is really important.

Suitable for: trying to find new solutions to problems.

Mind mapping

Draw words and symbols in a diagram with lines between connected items. This can help you create a mental model of test objects.[37]

FreeMind[38] is freeware software that works well for many occasions.

Suitable for: test planning, test execution.

Something New

New things are inspiring to the mind. Besides using totally new methods, also do small things that still are new.

Run FileMon[39] during a day or two; perform test actions very slow; imagine that you are a specific persona for a few days etc.

Even really bad ideas can help as the day when I only wanted to report bugs with even ID number.

There are a lot of other ways; the best ones are the ones you make up yourself. Just using different and new approaches to your everyday work is a productive way to train the creativity.

For the methods to produce results, the tester also needs to be vigilant; she needs to have “The Eye” that will want to spot errors, that sees many types of errors, that will look in many places, and that will be trained primarily by performing a lot of tests.

Endnotes

This is my list at this moment. It is up to you to test your own methods, to see if and when they work out fine. Allow your colleagues to do the same.

It is natural to be creative; the most important leadership ability might be to not de-motivate people. Creativity is not a gift, but rather the result of discipline and patience together with an open and curious mind.

Everyone is creative.

---

[1] Glenford Myers (1979), *The Art of Software Testing*. Wiley.

[2] Tom deMarco and Tim Lister (1997), *Peopleware.* Dorset House Publishing Company.

[3] Interview with Robin Goldsmith
http://webloadtesting.typepad.com/web_performance/2004/05/the_interview_w.html

[4] Part of Lesson 21 in Cem Kaner, James Bach, Bret Pettichord (2002). *Lessons Learned in Software Testing.* Wiley.

[5] Stated as common misconception as stated in Kaner, *The Ongoing Revolution in Software Testing*
http://www.kaner.com/pdfs/TheOngoingRevolution.pdf

[6] The question about a pair of new eyes has been investigated by Fredrik Härén, www.interesting.org

[7] George Keller

[8] Arthur Koestler (1964). *The Act of Creation.* Macmillan.

[9] Sir Ken Robinson, TED Talk: *Do schools kill creativity?* - http://www.ted.com/index.php/talks/view/id/66

[10] A complicating factor is the speed of a test, e.g. if you're executing a test and see that some more can be performed really fast, then do them.

[11] The triangle exercise is defined in Myers, *The Art of Software Testing:* "A Software is given, which receives three inputs (numbers) that define the size of a triangle's sides. The SW can, then, categorize the triangle in one of these categories: Equilateral, Scalene, Isosceles or Invalid. How many test cases can you build for testing this triangle? The more heterogeneous the cases the better, and: The more effective, better yet."

[12] Ross Collard (2004), Developing Effective Test Cases

[13] To specify all implicit and unspoken requirements would take insanely long time.

[14] Automated testing could move the creativity elsewhere, but with current technologies it costs too much to automate all tests, and they won't cover all the things a human can see. Rather, we should use automation to perform things otherwise impossible; load testing being a good example.

In a couple of hundred years total automation might be possible, but only for software that is implemented and used by computers.

[15] Teresa Amabile in interview with Bill Breen. *The 6 Myths Of Creativity*
http://www.fastcompany.com/magazine/89/creativity.html

[16] Nils-Eric Sahlin (2001). *Kreativitetens filosofi.* Nya Doxa. The chapter about creative environment is available at http://www.fil.lu.se/sahlin/kreativitet/content.html

[17] Sahlin, page 170, states that environments with a guru show signs of stagnation.

[18] Small scale seems dubious to include as a necessity for software testing. The larger the project, the larger need for both systematic and creative testing?

[19] Sir Ken Robinson, TED Talk: *Do schools kill creativity?* - http://www.ted.com/index.php/talks/view/id/66

[20] If your test effort requires only the exact following of pre-define test scripts, maybe you can't do creative testing, and should read other articles.

[21] Humour and discipline are not included in Sahlins list, but I think they are needed, at least for software testers.

[22] My translation of Burton (2002). *Det som muser viskat*. Symposion. p. 83: "*Inspirationens lätthet brukar också gå hand i hand med ett hårt arbete*"

[23] Howard Gardner according to Burton p. 28

[24] Mihaly Csikszentmihalyi (1990). *Flow.* Harper and Row.

[25] Sahlin, p. 174

[26] "Intrinsic motivation -- people who are turned on by their work often work creatively -- is especially critical." - Teresa Amabile in interview with Bill Breen. *The 6 Myths Of Creativity* http://www.fastcompany.com/magazine/89/creativity.html

[27] Result or quality is omitted, since quality is free if you pay for it.

[28] Burton, p. 21

[29] A huge list of different type of errors is available in appendix of Cem Kaner, Jack Falk, and Hung Q. Nguyen (1999). *Testing Computer Software, 2nd Edition.* Wiley.

[30] James Bach, *Exploratory Testing Explained* - http://www.satisfice.com/articles/et-article.pdf

[31] Chris Agruss & Bob Johnson , *Ad Hoc Software Testing* - http://www.testingcraft.com/ad_hoc_testing.pdf

[32] Jonathan Kohl, *Getting Started with Exploratory Testing - Part 2* - http://www.kohl.ca/blog/archives/000186.html

[33] Paired Exploratory Testing is described by Kaner and Bach in http://www.testingeducation.org/k04/documents/bbst18_2004.ppt

[34] This is a new term that might not be the best.

[35] A large collection of brainstorming (and other creativity) techniques can be found in Michael Michalko (2006).*Thinkertoys: A Handbook of Creative-Thinking Techniques (2nd Edition).* Ten Speed Press, or Roger von Oech (1998). *A WHACK ON THE SIDE OF THE HEAD: How You Can Be More Creative.* Business Plus.

[36] Edward de Bono (1985). *Six Thinking Hats.* MICA Management Resources.

[37] See EuroSTAR 2007 presentation "T4 - Grand Testing Master Theory - Mind Mapping Workshop" by Graham Thomas.

[38] FreeMind Wiki http://freemind.sourceforge.net/wiki/index.php/Main_Page

[39] One of many free system tools available at http://www.sysinternals.com (acquired by Microsoft in 2006)