From Here to Eternity: Pragmatic, Independent Testing

The Evolution of Software Engineering

There is an evolution in everything, including software quality engineering[1]. Even though you may be forgiven for believing software engineering is purely rational, scientific[2] and engineering-wise, in reality its development has over years followed a bumpy, turbulent road, like any other social phenomenon[3].

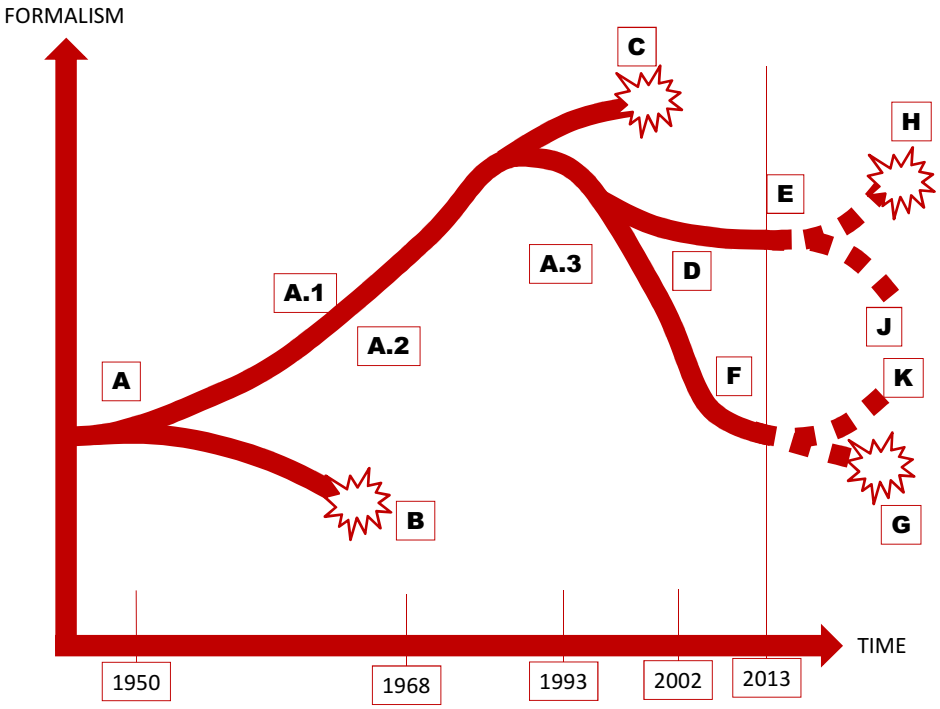

Figure 1: The evolution of Software Engineering

During early, pre-transistor, pre-integrated circuit days[4] of computing [**A**], so beautifully described by Richard Feynman in his reminiscences from Manhattan

[1] Or software engineering, or system engineering…
[2] Is software testing scientific? – Bogdan Bereza, EuroSTAR 1998, or victo.eu/ENG/Papers/STAR_1998_paper.pdf
[3] Software Engineering as a Social Science. Bereza, Ethicomp 2001, Gdansk: http://www.ccsr.cse.dmu.ac.uk/conferences/ccsrconf/ethicomp2001/programme.html
[4] Even though I guess that many readers, especially young ones, are not very keen to learn what happened long ago, there is no better way to look into the future than by studying past trends and interpolate form them.

Project[5], software projects were very few and – by today's standards – very small. So, there were enough geniuses and Nobel prise laureates to people them and to conduct them competently without special processes, organisation, or tools.

This changed; computers grew more powerful and numerous, projects grew larger and more complex. There were not enough geniuses to people them all, so more ordinary programmers had to take over. When confronted with daunting tasks, people have an ability – almost unique among world species - to tackle them not only with the power of their individual minds, but with organization and co-operation, processes and procedures[6]. This was the course taken by programming – although the name "software engineering" did not even exist then - in the late fifties and in the sixties [**A.1**].

As software continued to grow in complexity, but pre-manufacturing project methods remained, the situation deteriorated. Words about "software crises" were spoken, and, finally, talking about software *engineering*, instead of *just programming*, became standard[7]. Companies, who refused to mend their ways, are mostly extinct now [**B**].

And this trend continued, creating growing bastions of techniques, procedures, documentation, rules and standards as defences against the powers of chaos and entropy.

The Revolution

But there is such a thing as overdosing, and all methods, if used without reflection and due consideration, can easily create effects opposite to their initial intentions. Sensible documentation and order more and more often degenerated into stifling bureaucracy. Good formal techniques were sometimes used without regard to real business needs. When the cost of quality exceeds the cost of the lack of quality, overall cost increases instead of decreasing, and quality is no longer free[8], see figure below.

---

[5] Richard Feynman, "What Do You Care What Other People Think Further Adventures of a Curious Character", 1988

[6] Bogdan Bereza "Przyszłość IT tworzą geniusze" (The Future of IT Is Made By Geniuses), in memoriam of Adam Kolawa, 2011, http://www.computerworld.pl/artykuly/374129_2/Przyszlosc.IT.tworza.geniusze.html

[7] NATO Software Engineering Conference, 1968 [**A.2**]
http://en.wikipedia.org/wiki/History_of_software_engineering#1945_to_1965:_The_Origins

[8] Philip Crosby "Quality is Free", 1979

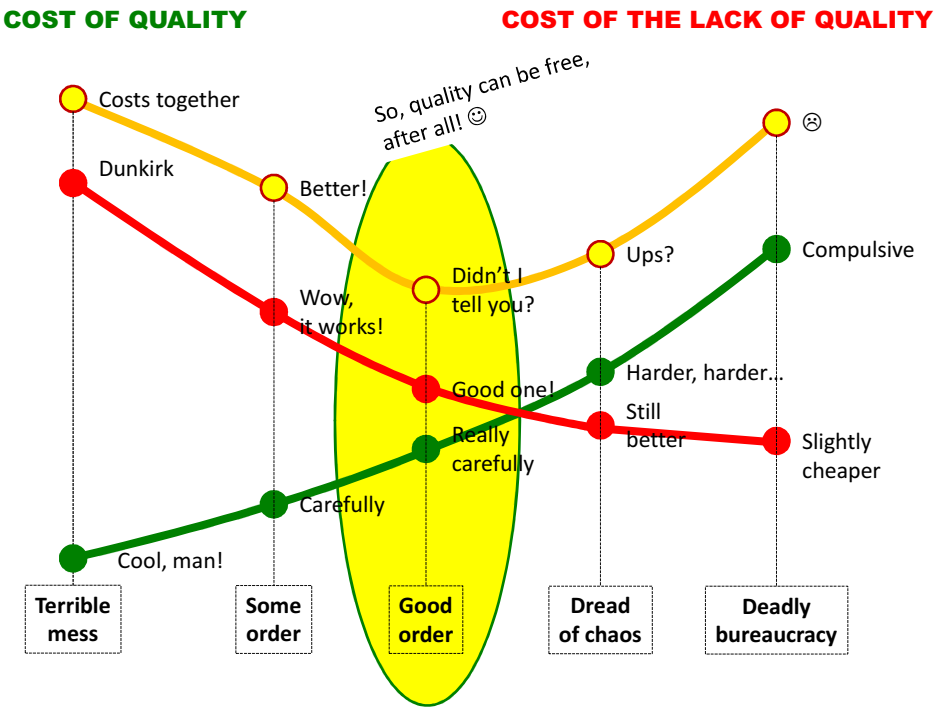

Figure 2: Optimal investment in quality assurance

When *ancient régime* of ISO, COBIT, ITIL and PRINCE 2 had started to crumble and loose some of its previous appeal, new, revolutionary trends appeared: agile, extreme, exploratory, rapid, dynamic, spiral, context- or test-driven [**A.3**]. Please forgive me that this last sentence is not fully compatible with the actual timing of some events; my intention is to capture the spirit, not the exact chronology, of this process.

Companies, who continued their blind confidence in protective armour of excessively stiff rules and torpid bureaucratic ways against *horror vacui*, are mostly very dead now [**C**], or reborn in a new, flexible shape, of which IBM is a good example.

The Great Divide

While downward slide down the curve of decreasing formality continued, a surprising bifurcation, or split, appeared (**D**).

On one hand, organizations like ISTQB [**E**], that were once bulwarks of common sense and good technical and engineering ways, while not abandoning them in their *syllabi*, slowly but surely abandoned them in their *ways*: they became politicised, closed, monopolistic, not open for audit, deaf to any criticism and unforgiving towards their, real or imagined, enemies. The monopoly they enjoy give them

power to stop or at least cripple potential competitors, and their sheer size makes them believe they are always right. So, while their syllabi remain relatively sensible (at least in those places where they are not full of absurdities some people have in vain pointed out for years), they no longer address real need of real professionals. They will either mend their ways, become more open and transparent, both in the way they develop syllabi, and in the way they manage and administer themselves (**J**), or perish (**H**).

On the other hand, the left-wing drive started by exploratory and context-driven approaches, became exceedingly totalitarian. Already 2004 [**F**], I wrote the following[9]: "It [exploratory testing] is a sensible, intelligent approach, application of good practices even in nonstandard situations. But it is NOT an "exploration", it is NOT alternative, revolutionary, green nor better – and there is no use pretending it is. Grandiose names do not harm directly, but they may create false goals and false gods, and make us waste time on them – instead of testing".

Maximum Hysteria

The programme of EuroSTAR 2013[10] seems to be the ultimate expression, in some respects very much like 1968 riots, of the exploratory zeal[11]. Have you heard about cargo cult[12]? Or NLP[13]? If not, it is high time you did, then, because they have one thing in common with exploratory testing: "the balance of evidence reveals them to be a largely discredited pseudoscience; scientific reviews show they contain numerous factual errors, and fail to produce the results asserted by proponents." Sorry, guys: no amount of interaction, creativity, social science, networking, "boosting your powers", "using sociology to examine testing expertise", coaching, "context driven mind mapping" (!), peer conferences, "testing machines as social prostheses" (!) and other mumbo-jumbo[14] will make you a good tester. Mastering reasonable test management methods and provable test design techniques will.

[9] http://victo.eu/ENG/Papers/Extremely_Agile_Exploratory.pdf
[10] http://www.eurostarconferences.com/
[11] http://gazeta.testerzy.org.pl/nr/2013-04-27
[12] http://en.wikipedia.org/wiki/Cargo_cult
[13] http://en.wikipedia.org/wiki/Neuro-linguistic_programming
[14] http://www.eurostarconferences.com/conferences/2013/conference-at-a-glance

Back to Basics[15]

Many people dare not challenge exploratory nonsense because they are afraid to look old-fashioned, outdated, or dull. Even serious companies use it, in addition to other methods, thinking this will let some revolutionary steam off, but do no or little harm. Some[16] do dare to challenge them, though. Sooner or later, we'll learn how to retain what's good in exploratory ways, and go back to using your rational brains, too [**K**]. The exploratory hysteria will subside [**G**].

What Testers Should Do Now

Keep the Evolution in Mind

The chapter above, it is not only of historical interest. On the contrary, *it is mostly about the present, and the future*. Knowing the fact that SQA methods change and evolve, and that this evolution is very much a social or even anthropological phenomenon, not – as you may believe – a technological progress[17], prevents you from being seduced by transient, trendy fashions, and allows you to distinguish the important form unimportant. Software engineering is, alas, as fertile ground for charlatans as psychology and psychological therapy are, and knowing it you profit both by not falling for them, and by being able to pretend you believe in them, if it is good for your career ☺.

Methods and techniques are important – names are not.

Please note that EVO – *Evolutionary Project Management* (1970) and *cleanroom software engineering* (1980), and *daily build* (1990) are old names for the same methods that are better known today as *agile* and *extreme programming*.

Please note that *exploratory testing* was once simply called *experience-based test design*, and did not pretend to replace everything else in testing, nor to be better.

Think for Yourself

ISTQB boasts that it has more than 250.000 certificates, and plays now in the same league as PRINCE 2 and PMI. Is it good or bad? Neither – it is irrelevant. To some extent, it is looks good, because ISTQB syllabi *mostly* make sense, and because

[15] Lee Copeland, The Nine Forgettings, CzechTest 2013 (http://czechtest.com/nine-forgettings)
[16] Martin Pol, The Evolution of Testing, CzechTest 2013 (http://czechtest.com/evolution-testing-what%E2%80%99s-next-contribute)
[17] Rather, it is spiral, which shows nicely if you remove the time axis from Figure 1: The evolution of Software Engineering

testing *is* important. To some extent, it is bad, because ISTQB has secured its dominant position not by being better, nor transparent, nor wise, but *in spite of* not always being that. Exam questions still contain prime examples of absurd, downright wrong or at least disputable muck, but nothing can be done about. Some national ISTQB boards are still fiefdoms without any openness, public control or democracy, but nothing can be done about it. Changes in the syllabi, examination rules etc. are still planned behind closed doors and come as surprise to all except the privileged few, and… nothing can be done about it, either!

What should you, TESTER, do then? Use your head is the best piece of advice I can give you. Do not let you be overly impressed by either the political clout of ISTQB, or the seeming revolution of exploratory; learn from them all, and think for yourself!

Figure 3. Stride proud and independent between the two extremes

Keep Full SQA in Mind

Testing is not a separate kingdom, it is just a part of software quality assurance. However impressive and sexy the names of test techniques and tools sound[18], they are not more important than good old plain ways like requirements modelling and validation, or configuration and change management.

[18] Especially tools – I often refer to many test discussions as "capybara – ruby – fitness – selenium-level discussions"

An anecdote will show my point more clearly: a company once addressed a tool vendor, asking about code coverage tools. The ostensible reason was a huge amount of bug recurring during regression testing. A thorough analysis of the company's development process showed, that real culprit was… wrong make-files! Fixing them removed the problem, and measuring code coverage would not be the right medicine at all.

Learn the Requirements

Testing hardly makes sense unless there are the requirements. It is simply not possible to check if software is wrong or right, without knowing what is wrong and what is right, that is, knowing the requirements.

Is that really true? How come so many reasonably successful projects have little or no requirements, and still perform testing? The answer is simple: in them, testing becomes both requirements elicitation and validation. It is a way; by far the most expensive one. Using requirements engineering is much more effective, efficient and cheaper.

We testers should help our RE sisters and brothers in getting more acceptance. The most widespread RE certificate, IREB CPRE[19], has got approximately 13.000 certificates, compared to ISTQB 250.000. Worse still, even their name is insecure: RE is notoriously called "business analysis" and mixed up with project management. Changing this, is one of the most important challenges for SQA during coming years.

Learn Test Design

Yes, you know: decision table, all-pairs, state transition, equivalence partitioning, boundary value analysis, black-box, exploratory, TDD, regression, confirmation, configuration, UAT, extreme programing (testing in), agile, user story, checklist, context-driven, static, review, model-based, OAT, contract and regulation, alpha and beta, BDD, scenario, domain, boundary, equivalence class, syntax, fuzz, random, penetration, vulnerability, metasploit, sanity, basic, requirements(-based), risk(-based), characteristics, negative, smoke, white-box, grey-box, back-box, security, stress, load, volume, error injection, UML, verification, validation, coverage, cause-effect graphing, domain analysis, defect-based, error-guessing, experience-based, checklist-based, accuracy, suitability, interoperability, usability, accessibility, root cause analysis, dynamic analysis, reliability, recoverability, maintainability, data model based, data flow based, transaction, testing based on

[19] www.ireb.org

various code coverage measures, testing based on various graph coverage measures, TDD.

When you know them, come back and let's then talk about being "certified tester".

Learn Risk and Profits

That is easy: just have a look again at Figure 2: Optimal investment in quality assurance and wonder where your organization is on this diagram? Learn to estimate business benefit of more (and, sometimes, less) testing, and to follow the **secret golden rule of testing**: it is easier to convince CFO than CIO about the benefits of testing, because for CFO, good testing saves money.

And, last but not least, if your country, or your company, or your colleagues are among latest casualties of exploratory disease, you are fully justified to call doing this something like "contextually-driven exploratory mind-map-based risk assessment" (which it essentially is, provided you like to play around with misleading words), and get their acceptance, too!

Have fun!