Business Process Testing

Note from the author: because my penchant for irony, stylistic exaggeration and sarcasm, I am afraid this article can be misunderstood, hence the following explanation:

In this article, my intention is ***not*** to claim that BPMN-based testing is a silver bullet, or somehow superior to other approaches. My goal is to present it as a powerful test design method, that has not been yet discovered. I do claim, however, that the negligence of BPMN by contemporary test gurus is a symptom of trendy test specialists’ excessive concentration on fashionable tools and smart methods, but I do not assert that these tools or methods are wrong or useless.

Fin de siècle and software testing

At the end of the 20th century, a phenomenon occurred, that influenced testing profoundly: the emergence of Internet technologies, which made dramatically novel business approaches possible.

1. On one hand, this created the situation, where rapidly changing requirements demanded shorter product life-cycles, and were often incompatible with traditional methods of software development. This meant the advent of XP and, to some extent, of exploratory alias context-driven, alias rapid testing[1].

2. On the other hand, the enormous business potential offered by the new technologies created demand for tools enabling you to describe, optimize, automate, or even obliterate and replace[2], business processes accurately, flexibly and fast. This demand created new, more powerful process modelling methods, including BPMN.

[1] *Testing Computer Software* and *Lessons Learned in Software Testing* (by Cem Kaner, James Bach and others) were both first published in early 1990s, earlier than *fin de siècle* and Internet. Yet, they would never have become as widespread as they are today, if Internet-powered business revolution had not taken place.

[2] Michael Hammer “Re-engineering Work: Don't Automate, Obliterate”, Harvard Business Review, July–August 1990

These were two surprisingly divergent trends: attempts to achieve more discipline in business processes, but the opposite – more *laissez-faire* - in software development and testing.

A short history of modelling

Techniques to **model business process** such as the flow chart, functional flow block diagram, control flow diagram, Gantt chart, PERT diagram, and IDEF have emerged since the beginning of the 20th century.

The rise of computer / software industry resulted in dynamic and comprehensive development of **systems modelling**, providing functional, architectural and structural views of IT systems. In 1980s, database industry developed data modelling methods, including ERD – still going strong.

1990s gave us UML – a family of interrelated system modelling graphical languages, covering all aspects of system and software perspectives (behavioural, data, and structural).

These huge advances in system modelling have **not** been mirrored by corresponding advances in test design techniques. For example, in spite of the unprecedented growth of ISTQB, the most comprehensive test certification scheme ever, using models for test design is given only superficial attention in it[3].

Model-based testing then, and the ISTQB MBT syllabus, expected this year? I would not bet that it'll change much. MBT seems mostly preoccupied, not to say obsessed, with **automatic** test case generation, and focused solely on state transition diagrams / Markov chains. Cute, but far from comprehensive.

At the beginning of 21st century, out of the foundation laid by UML, a new and powerful business process modelling language grew: BPMN. It has revolutionized business analysis, and contributed to, or played an important role in, the birth of new IT branches, described in the following section. However, it went almost totally unnoticed by testing[4].

[3] Out of ca 20-25 useful models that can be used for test design, only use case diagrams, state transition diagrams, and – to some extent – control flow graphs are mentioned; note the word "mentioned", not really covered.

[4] You may imitate a famous statesman by saying "never in the field of IT has so much been so thoroughly neglected by so many". The burning issue is WHY? I try to give a partial explanation of this in the final section of this article, "The *gimbaza* of testing". However, full explanation belongs to sociology, anthropology or group psychology and not IT, so I'd rather not attempt it here.

Scary acronyms: BPM, BPMN and BPMS

BPM – Business Process Management – has always been done one way or another since our ancestors climbed down the trees and started group hunting on savannahs. Only recently, however, it became a scientific, or an engineering discipline. Only recently attempts to create some kind of generic business process theory were started, and its findings are applied to managing, and improving, real processes.

In order to manage a process, you must first of all understand it, then describe it, then distribute your description to its stakeholders so that they know how to work. A very basic, natural-language description is sometimes good enough, especially for process frameworks; scrumguides.org is a good example of such a description. For more complex processes, a formal language is preferable. **BPMN – Business Process Modelling Notation** – is such a language. It was developed quite recently, hardly more than 10 years ago, and today it is controlled by OMG, the same organization that controls UML; quite right, since many BPMN concepts are derived from UML.

What is BPMN used for? Sure enough, you can use it to describe processes in which not an ounce of computer software is involved, provided you can find such business processes today. So, typically, BPMN diagrams describe business processes, or high-level requirements, in which IT systems are, or will be used.

Mainstream IT trends during the last 10-15 years include corporate architectures designed to automate, or semi-automate, whole business processes. Such system are often called **BPMS, Business Process Management Systems**. Names like SOA, Web Services, BPEL, end process engines appear here. The need for testing them is obvious. The need to understand BPMN to test them is equally obvious.

BPMN overview

Really learning BPMN will take a few days, and learning to use it in practice, some more. For this, a book like Bruce Silver’s “BPMN Method and Style” will come handy.

The simplest explanation is a picture, an example diagram:

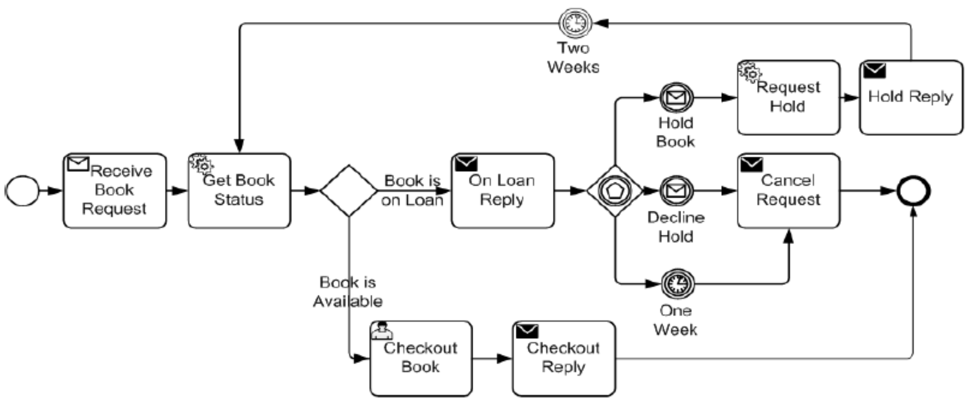

Figure 1. Example BPMN diagram (Source: brsilver.com/wp-content/uploads/library_spec.png)

You can see a diagram, that quite resembles a control flow diagram, or an UML activity diagram, but contains some different or additional symbols. Without actually learning BPMN, you can intuitively guess its meaning, especially if you’re already familiar with other diagram types.

Another example shows how BPMN diagrams map onto organizational structures:

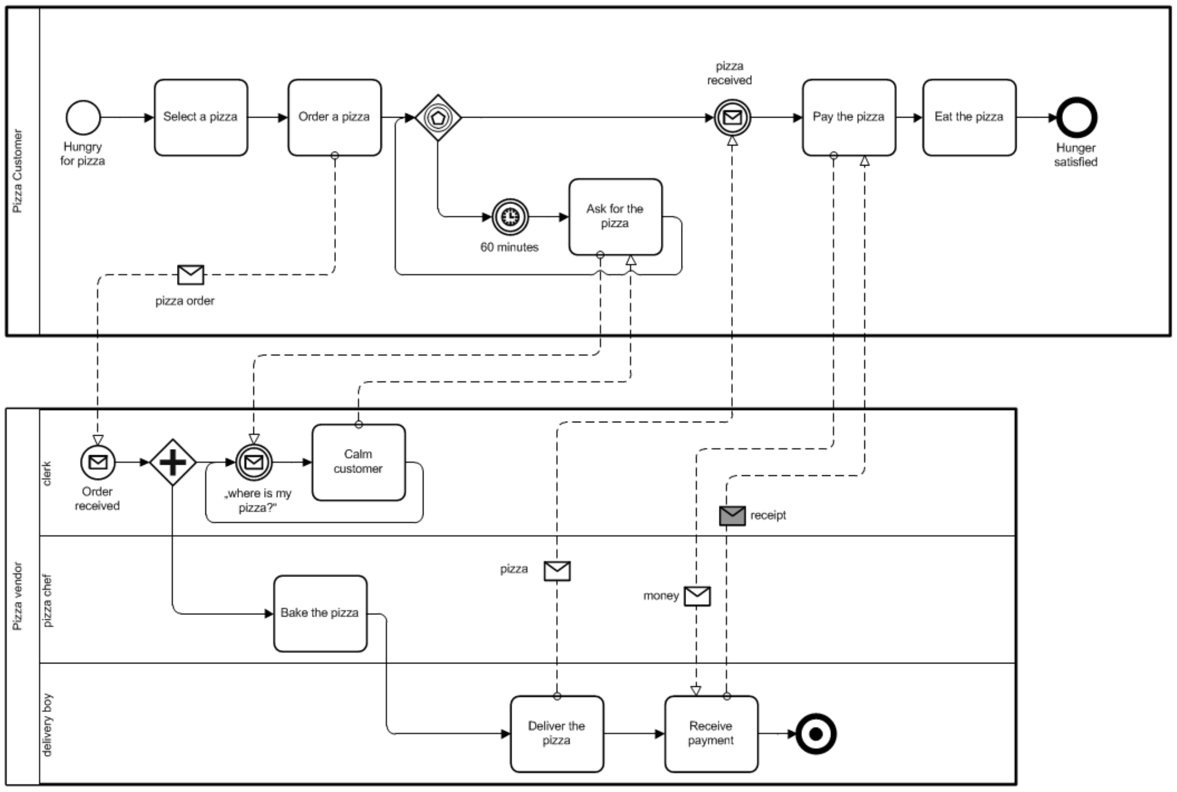

Figure 2. Popular BPMN “pizza example” illustrates pools and swim lanes (Source: bpmn.info/wp-content/uploads/2011/12/BPMN-Order-Process-for-Pizza-1024x688.png)

Pools and lanes are not very surprising, either: they can be used, though seldom are, with all UML behavioural diagrams, too.

This is it!

BPMN-testing

Actually, nothing more is needed. BPMN exists and is widely used by business analysts, and is a useful and popular way to describe / model both business and system requirements, so **of course BPMN-based testing is a must**.

James Bach tells us that testing requires “maximum use of skill” and that exploratory testers shall be ready to learn software stakeholders’ real needs. BPMN diagrams, ***if available***, can provide both.

Many Scrum practitioners recommend user stories[5] as a good way to describe requirements, because they decrease the risk of misunderstanding; BPMN-diagrams have exactly the same benefits.

Only very conservative, or exceedingly Selenese[6] person can deny this.

Static testing of BPMN diagrams

A valid and important argument in favour of models (all models, not only BPMN-diagrams) instead of natural language descriptions is that formal languages can be verified, if they are syntactically correct. This does not replace validation, but removes some possible requirements bugs. ISTQB syllabi explain – quite rightly – the benefits of code review and static analysis, but fail to mention much greater benefits of static model testing.

If BPMN diagrams are used, testers shall rather spend some of their time static testing them, providing very valuable help to business analysts with little test skills, than spend much, much more time (remember Boehm’s curve?) exercising co-dependent behaviour[7] lower down the food chain.

[5] By the way, does using BPMN fit in agile framework? Why not? For example, you can use a BPMN diagram as a high-level epic, and then describe its tasks (rectangles with rounded corners) as user stories.

[6] I like Selenium! But, as is the case with all test automation tools, excessive preoccupation with it sometimes results in forgetting other aspects of testing, including its business sense.

[7] Lee Copeland: http://www.stickyminds.com/article/when-helping-doesnt-help (20 May 2015).

Static business process testing

Testers are used to it: requirements change halfway thorough software project, and, unless you work according to agile framework, nobody tells the testers. The great prophets of Agile Manifesto knew it, and therefore they wrote "we value responding to change over following a plan".

Why is that so? One of the reasons is, that while you attempt to optimize your process by giving it IT-support, you often discover how bad, costly, illogical, complex, counter-intuitive, error-prone, tiring and irritating for customers your process is in the first place, and that automating it may actually make matters worse. Then you start re-designing your process on the fly, and requirements change.

An idea – just an idea – why not introduce a business-process-testing-and-optimization-process not only as part of business analysis preceding an IT project, but as continuous activity[8]? If you prefer maven + jbehave + PageObject + Selenium, stay away from it. Otherwise, go for it!

Remember, testing is not only about finding bugs, but very much about measuring, and helping to improve, UX (James Bach calls it "charisma"[9]). A chaotic process is never charismatic, it's annoying. Test it, using BPMN diagrams, to find some process usability bugs, and prevent this.

Test design from BPMN

Please have a look again at BPMN diagrams in the previous section, then open your ISTQB syllabus at random. Instruction coverage, branch coverage? Of course, only "instruction" need be replaced with a BPMN-name "task" (they are the rectangles with rounded corners). Path coverage, too (yes, BPMN does support loops). The squares (called "gates" in BPMN) are tempting places to design decision and decision-condition testing.

Messages and other events (circles on BPMN diagrams) are good candidates for the type of testing used as well on state transition diagrams. And, provided valid and invalid values for messages and events are defined, domain testing beacons, too.

---

[8] Making models often helps find bugs in natural-language descriptions, and in the processes or in the presumptions, on which these processes are based. This is not only the case with BPMN-modelling, of course. I have, for example, many times found bugs in requirements simply by attempting to translate natural language descriptions to state transition diagrams. Dead states, or states that cannot be reached were quite common. Using modelling – any modelling - simply disciplines ones thinking.

[9] http://www.satisfice.com/tools/htsm.pdf (20 May 2015).

The blah-blah-blah[10] so often used to describe strategies for testing use-case scenarios ("begin with main scenario, then try alternative scenarios, then error cases") are of course valid experience-based strategies for BPMN, too. Only better, because BPMN diagrams are more commonly understood by business people, and more suitable for describing business processes than use case scenarios are.

Just a minute ... what **are** use case scenarios? This question raises another argument I have sometimes heard as a (poor) explanation of why BPMN is ignored by testing. Apparently, BPMN is for high business-level process descriptions, while ISTQB-endorsed test design techniques are for (lower) system level. To this, I have three counter-arguments:

1. Sometimes, it is true; very often, it is not.
2. Have you ever heard about black-box testing? Well, this is it.
3. Many experts argue that test cases should not only be small, artificial steps, but rather long sequences of such steps, mirroring real business system usage. They are right, and BPMN-based test scenarios help you achieve this.

In bullet "1" above I say that BPMN diagrams can describe system-level behaviour, too. Use case scenarios are perfect examples. Use case gurus have invested a lot of energy into telling you how to describe use cases scenarios using... natural language, "goals" and such. Cute, but not very helpful for testing. I recommend using activity diagrams or even control flow diagrams, instead, but, first of all, BPMN diagrams. In other words, BPMN diagrams can be components of higher-level use case diagrams – good news for testers!

Agile says, that user stories should be completed by acceptance criteria (or acceptance scenarios, or examples – "specification by example"). In my opinion, the art of deriving adding such example acceptance criteria to BPMN diagrams will be an important trend of the future. We will attempt it, or have already attempted it, depending on when this article is published, during my CzechTest 2015 tutorial[11] "Business Process Testing" 24 June in Prague.

Test process in BPMN?

Last but not least, the absolute majority of process and process maturity / improvement standards (PRINCE 2, PMI, CMMI, SPICE, Automotive SPICE), including test standards (TMM, TPI, ISTQB "fundamental test process"), use words, not BPMN diagrams, to

[10] "Blah-blah-blah" does not imply that I think this strategy is wrong – it simply states my opinion that it is trivial.

[11] http://czechtest.com/business-process-testing (19 May 2015)

communicate their contents. No wonder – they were created in pre-BPMN era. **It is time for change**.

Summary: the *gimbaza*[12] of testing

Since 1980s, software testing has grown enormously. From the days of pioneering books by Boris Beizer, Bill Hetzel and Glenford Myers to 2015, with zillions of testers and with ISTQB – the world's greatest totalitarian empire since the days of the Soviet Union, a lot has happened.

Testing has become a (pseudo-) science, a profession, a calling, a you-name-it – even an ideological battle-ground. It is now dominated by very young people. Testing, far from being a VIP profession, has become another hacker business, in parallel with programming. Instead of learning to see the whole business-process-project-QA-stuff, it keeps itself, or is kept, firmly within childish "maven + jbehave + PageObject + Selenium" framework.

Hey, please, do not get upset! I do not claim here that maven is useless – on the opposite, good build control can replace a lot of mindless testing. I do not think jbehave is bad – as all BDD approaches or tools, I think it is great. PageObjects are useful for test automation, and Selenium is a fantastic tool. But I have met too many people, and listened to too many conference presentations, that firmly believe, that possessing these particular "test hacking skills" is all there is to know about QA and testing, which is a very, very wrong belief – hence my sarcastic style. Testing sometimes mimics the worst programming mistakes, such as valuing technology and tools over budget and business goals, and I am against it.

Bosses, and older testers who has advanced to become bosses, get certified in ITIL / PMI / IPMA / IIBA / COBIT / BPM and forget testing, because it is hardly mentioned there – just like the situation was 40 years ago. Testing, instead of becoming, together with requirements engineering, part of the duopoly of power in IT projects, is still kept in power antechamber, and relegated to "maven + jbehave + PageObject + Selenium" spectrum.

Business process testing may change this.

[12] Gimbaza: a Polish slang word, connoting "immaturity, obnoxiousness and stupidity" (http://en.wiktionary.org/wiki/gimbaza, 19 May 2015)