Nice words are not enough

Louise Perold
Micro to Mainframe
South Africa
louisep@mtom.co.za

Carsten Feilberg
Strand & Donslund A/S
Denmark
carsten.feilberg@s-d.dk

ABSTRACT
In this paper, we describe our experiences implementing Session based and Exploratory testing in Waterfall type projects.

1. INTRODUCTION

Session-based test management [1]. Exploratory testing. Agile approach. These are all nice words which fortunately are beginning to become known in the IT industry. However, in many situations these words are not enough. Mainly due to fear, insecurity and the lack of knowledge of many IT leaders, project managers, customers, and even test leads are reverting to a manner of testing that is focused more around providing numbers and statistics than actual testing value. When trying to introduce session-based exploratory testing into this context of the fear and insecurity, on both the supplier side and the client side of a project, the tester and the test manager are likely to meet a wall of resistance.

In this paper we would like to investigate the causes of insecurity and the de facto demand for a waterfall approach to both development and testing. Using two different projects as examples, we would like to discuss some attempts to change the situation while also examining some of the skills required.

2. THE PROBLEM

As a community that believes in context and providing valuable information, it is mandatory that we try to not only spread the word and educate people, but also supply the evidence, the substantial facts and the comparisons that enable our stakeholders to believe in us and what we do. Insecurity is often self-fueled. If there is a slight stutter in the tester's voice, it will resonate into the insecurity of the manager. Establishing confidence within stakeholders allows us to communicate more successfully with them.

In our experience, an example of a project type that is most impacted by this insecurity is a fixed-price, fixed-period

contract. Customers want this kind of project because they are worried they might end up paying the supplier endlessly without getting anything useful. They are also terribly nervous that the supplier will cheat them and give them a bad product, or half-a-product, and deliver only after substantial delays. These are real concerns. When we come to them saying, “Let's test it!”, but we don't write down all test cases in advance, they become even more worried that this “hapsy-flapsy” approach won't get them anywhere. To reduce their worry, they measure, and they end up measuring the supplier's progress using numbers that don't necessarily correlate to the health of the project.

Similar to the customer, the suppliers are also scared. They're afraid that the client will never, ever approve anything, so that they will be held in limbo and will never be able to get out of the contract. To prevent this, they react by sticking directly to the requirements and the contract. They don't deliver anything without considering the political value in terms of whether the client will give them credit, or money, for it. This too results in measurements, all focused on when their product is produced. Unfortunately, these measurements, like the measurements of the customer, usually don't directly correlate with project health.

The project manager too is weary. The nature of a fixed price contract results in the need for testing teams to constantly show progress and efficiency and in most project and programme managers minds, this is measured with numbers, usually with defect counting, or numbers of passed test cases. Unfortunately, the more defects that are raised, the more fear project managers have of the ever moving project deadline – which is not a good thing for a fixed price delivery. In some ways, doing good testing can be a very bad thing when trying to manage fixed costs.

The two projects we examine in this paper were both impacted by the fixed price problem in some way.

We believe that as testers, what we *need* to influence is the underlying *model* that is guiding the actions and reactions of the people involved in the project. **What is the cause of the fear?**

3. THE LIKELY CAUSE

For many reasons waterfall is easily grasped. It induces a false sense of security and safety. The majority of people we have come across on projects, believe they can manage numbers. It is a false sense, yes, but people who feel it don't know that. To them it's a true feeling that's so much better than anything which is not readily measurable and complying with their internal, mental model of both product and project.
This model is likely the underlying cause for insecurity and resistance, and also the insatiable demand for numbers.
So how is the model formed?

On the client side, acquiring a new IT system is a necessary task to be ticked off and done with, so that the actual business can continue - preferably with the added value from the new IT system. This resembles the situation of building a new home. It's the result that matters, not the process. And often the client is not competent in the process of developing, testing and implementing a new IT system.
This leads the client to look upon the project as a set of tasks. Once 'done' they have the result. Defining 'done' is the next problem. 'Done' is in many contracts defined as having a system which passes a test. So, the test is crucial.
All in all, someone should sit down, define the test, get it executed and then the project is over.

On the supplier side, the knowledge of the process is not a problem (hopefully). The problem is getting the money for doing the work. This puts the supplier in a state of wanting to measure when the job is done, so that the bill can be sent. Again, this causes the project to be laid out as a set of tasks that should be accomplished and then it's 'done'. Sometimes without consciously making the decision, both client and supplier will easily enter a waterfall process model, as it fits their overall intentions.

In contrast, agile approaches are completely different. Instead of having a house built with a fixed period and for a fixed price, some money is set aside and the building process continues as long as there's money. Constantly choices

are made as to what the next step should be.
Most clients are resistant to this scheme. Although they might get a very good product from it, they don't feel convinced they can play their part well. This approach requires room for experimentation. Although waterfall does also need some experimentation, that part of it is hidden away. If you only got a few hard-earned bucks to buy a system for, you want it's full money's worth.

Testing in the waterfall process is looked upon as being just another task: setup, do, done. And it's true that once the foundation for a house is put down and checked, this verified result will last. Only severe damage, like accidentally dropping a wall on it, will change it. Alas, in software it's not so and this is just one deviation from the model.

4. ATTEMPTS TO ALTER THE MODEL

Changing the model that people work by to incorporate the elements of session-based testing and exploratory testing is not easy, since these elements are in conflict with crucial elements of the waterfall and 'building a house'-models. We would like to provide a number of examples drawn from our experiences of some projects describing how we have dealt with introducing exploratory testing, recording and running test sessions, documentation and reporting - and what the reaction has been.

4.1 Exploratory testing

4.1.1 Carsten on introducing exploratory testing

"Testing is easy - just create the appropriate amount of test cases, linked to each requirement, and you're done". That is the doctrine I have faced in fixed-time, fixed-price projects. The value of finding bugs in the code, bugs in data, bugs in usage etc. is *okay*, but still of lesser importance. Even the client puts this view forward, which surprises me.

There was always a demand for rigid documentation of test cases and estimation of the testing work, so 'someone' could 'approve' that the testing was done right.

This is part of the fear scheme. Fear of not getting what was wanted. Fear that something important is left out. And I am convinced it *was* fear, because the work done to question the selection of tests, the coverage and the methods could only serve to be making sure that someone had *thought* of these things, not to thoroughly investigate whether anything actually was missing.

Another general resistance I am confronting is the assumption that exploratory testing is difficult and it would be much easier if the testers would get a script to follow.

There is also a practical concern, that if test cases are not known until the last minute, test data cannot be prepared in good time. Countered with exactly what the problem is with that reveals a lot of the insecurity, but it may be handled by assigning some people to work on ways to provide data, like generators and contacting 3rd party providers to set up data in their systems (if applicable).

Then there is the concern about coverage - 'how can one be sure that everything is covered when the tester can decide the next step on the fly? Something is bound to be overlooked!'

Despite giving an explanation of how session-based testing works - that debriefing will catch this and thus working on coverage is indeed ensured, the doubt remains until the very end.

An alternative plan is to start interviewing the doubtful people about how they would feel certain that the appropriate coverage is met by testing. However that is not working either, as the fear they feel is a feeling, not a logical deduction. Hence they can't tell themselves what is needed. They will always ask for time to think about it.

The best thing that can happen is that testing starts and the results show them that work is done, bugs are found and that coverage is met.

Until then I have had best results with communicating my intentions specifically and starting to collect test ideas openly. This seems to **SOOTHE** the doubt a bit.

4.2 Recording and running test sessions

4.2.1 Louise's project context

The project involved testing an off-the-shelf-customised package for a smaller bank in South Africa. The initial brief was for a team of six testers to join the project for six weeks and assist the team of Business Analysts in testing base functionality of the product with bank specific configurations. The team eventually grew to nineteen people and the project continued for a year up until implementation.

4.2.2 Louise on recording and running test sessions

On the project, we made a determined effort to start testing and logging bugs as quickly as possible. Deadlines were extremely tight and we needed to demonstrate quickly that we were capable of delivering value. We setup an initial briefing session with the business analysts and found out about the base functions that they wanted us to focus on. We asked some questions around what they were expecting to happen and essentially created a very high-level checklist of functional scenarios that we should test. This list was used as a starting point to brainstorm further with the test team. Functional areas were then divided amongst us. Each "scenario" became an initial session charter. These were typed into HP's Quality Center as "test cases". In the execution part of the same tool, we had instances of these "test cases" that were really scenarios that we would execute against. For execution, we would include our session notes, and screenshots as attachments to the test. During execution, if we came up with new test ideas, these would be added as additional "test cases" for execution later. (Or if it was part of what we were doing, then we could create some new notes and simply add and attach at the same time as executing).

4.2.2.1 Some side effects of this approach

Given that we now had our sessions in a tool, we had created a situation where they could immediately be viewed by everyone on the team. Session charters would be categorised with all the other "test cases" already captured by business analysts. It also meant that each "test case" had to have a status. We did manage to add some status types - passed, failed, not complete, not applicable, or not executable. (So we were not only limited to passed and failed)

If a "session" didn't pass - we called it a "failed test case". If a "session" didn't result in us finding any bugs, we called it a "passed test". If something was out of scope, it was "not applicable" and if we were unable to get to the functionality at all because of blocking defects, the scenario was "not executable". To try and communicate that we were actually working with charters and test sessions and not "test cases", was difficult.

Giving context to the scenarios that were being executed also proved challenging. I did however have the daily debrief with the team, which gave me an opportunity to understand the detail, and we were updating a dashboard daily. This gave me enough information to know where the issues were and I was able to communicate this by using the dashboard, as well as regular email updates around specific risks and issues.

4.2.2.2 What I learned and need to work on next time

Having the charters in a central place worked in that everyone had access to ideas and the scenarios could be linked to bugs. It did also allow a view of functional areas covered and not covered.

I didn't make enough use of the Session Based Testing metrics [2] to help me to build historic data to predict trends from. I think this was also a side effect of managing the sessions and execution through Quality Center as I became focused on providing other types of metrics.

4.2.3 Some notes on skills for recording and running sessions

Communicating effectively within the team and using the debrief to understand the critical issues was incredibly important. This meant being able to ask the right questions. As a test lead, being able to effectively communicate to stakeholders in such a way that trust is established is also key. Of importance here is to be open and frank, not trying to hide bad situations for example if the test environment is down or a really ugly bug has been found. Also, qualifying with stakeholders what this information means to them and whether there is anything they are missing is

a good idea. It makes them feel more part of the process, and not just anxious spectators.

4.3 Documentation

4.3.1 Louise on note-taking

For evidence of our sessions, we took notes in notepad and had screenshots using Timesnapper (which automatically takes screenshots at a pre-defined interval). We only attached critical screenshots to Quality Center of evidence where applicable (and kept the full shots separately). Where appropriate, we also kept spreadsheets of calculations with the corresponding screenshots included within the spreadsheet to show a particular scenario involving calculations.

4.3.1.1 Some side effects of this approach

Not everyone took good notes. In some cases, team members didn't even attach logical screenshots - this seemed to be a side-effect of having the test charters in Quality Center. It was more familiar to some team members to revert back to a more traditional way of trying to create and follow a repeatable script - especially because all of the team had used Quality Center extensively on previous projects.

4.3.1.2 What I learned and need to work on next time

Making sure that team members are taking notes properly and managing their testing evidence in a logical way is a continuous and ongoing challenge when using Session-based testing. I would like to work a lot harder on establishing the correct habits within the team (regardless of what tool we use). This includes ensuring that no matter what format the notes are taken in, they are stored electronically if this is important for the context.

4.3.2 Carsten on note-taking and documenting

I like to keep records on paper and have made room for a PROOF-like summary of the session on the charter itself. The actual documentation resides in the test log, which is a point I always debate with the testers. Some like notebooks, and then I buy those for them. I prefer notebooks myself. Others insist that notepad or another editor is the right thing for them and I let them have a go with it, but whatever they choose I always follow up on how it works for them. On one recent project I have tried to have WINK, a screen capture program, running to capture a screendump every time a button is pressed or the mouse is moved or clicked. This additional documentation can easily be saved and stored on DVD's and provides a very extensive documentation which further makes reporting bugs extremely easy.
The logs - in whatever format they have - are stored somewhere safe, but I have never had to go back to them, so they are preserved mainly because it proves that testing was done - if anybody *should* ask.

For most of the documenting I keep the charters with the summary, and for every session I make sure the testers print out a new charter. This makes the tool dependency low, as it's mainly a question of allocating charters around and categorising them into 'undone' and 'done'.

Where the client or supplier is hungry for 'passed test cases', there is some extra work for the test lead to make them understand the concept of sessions. Sometimes I have been showing them some of the charters, going through them in detail and perhaps even going through a quick tour of the test log.
The result varies - some get really interested and *hooked*, and others appear to be thinking of their next lunch or something equally important while I explain.

4.3.3 Some notes on skills for documentation

Keeping a constant focus on mentoring and coaching the team in taking good notes is a special skill. A test lead needs to be able to provide constructive and specific feedback as well as good examples for the team to learn from.

4.4 Reporting

4.4.1 Louise's approach to reporting

I decided to try James Bach's dashboard [3] as the primary reference for progress reporting. There was a nice large whiteboard which was divided up into the functional areas. Legends next to it showed everyone what was meant by each column. It was updated every couple of days.
Initially, it attracted some positive attention. The supplier development team head would use it to see which areas seemed to have a lot of bugs as well as checking the blocked areas, so that he could focus on getting those specific bugs fixed as soon as possible. This worked well. However, most other people (specifically project managers) were confused: "But can't I get some numbers?" and "I need to see how far you are and I just can't tell from this" and "I want a place from which I can start asking questions, and numbers will give me that. I want a barometer of progress".

The programme and project manager kept asking for numbers as a demonstration of progress. At the time, I couldn't figure out another way to communicate it satisfactorily and so, I decided to give them what they wanted.

I did attempt to give the numbers context - I plotted the total number of test ideas (scenarios, "cases", charters), against the number executed, and number "passed". I also included the total number of open and closed bugs per day. Updates to the figures were made daily and put on the wall next to the dashboard in the hope that it would show a clear contrast between meaningful and non-meaningful information. My experience however was that the project and programme managers in this case, preferred to only look at the numbers and then start asking questions.

4.4.1.1 Some side effects of this approach

Due to the fact that the numbers of tests were increasing every day for the first part of the project, at one point I was told to "please stop adding tests". "You need to get 'permission' for any new tests to be added". We then had to go through a process of justifying any new charters that we added.

Another very serious side effect of showing the numbers was that we ended up biassing them. The minute that we started to look at "progress" as depicted by tests passed, there was a lot of concern displayed by the programme manager when it wasn't deemed sufficient. We ran "easier, quicker" test scenarios, which we knew were in the more stable parts of the application, so that we could show some "progress". This was not always intentional, but definitely a side effect of measuring by numbers passed.

4.4.1.2 What I learned and need to work on next time

I am still struggling with this part of Session based testing and moving successfully away from measuring by numbers. There is an element of progress that needs to be communicated successfully. The question is how to do it in a language that project managers understand. This seems to be especially challenging when their main objective is to have an easy way of communicating progress to Executives without having to go into any kind of detail.

4.4.2 Carsten on "Progress"

"Progress" in a waterfall model is seen as the number of undone tasks decreasing over time. However, with Session Based Test Management you have a constant number of undone tasks, as charters done are replaced by new charters that either change direction or deepen the test scenario. This results in conflict.
Project managers with a waterfall model in their heads look at their 'Progress-o-Meter', and if it's not showing the expected velocity *something needs be done*.

In both Louise's and my own projects, the numbers that went into the 'Progress-o-Meter' rarely showed an appropriate velocity. The number of ideas increases and the number of executed tests will increase as well, but more steadily. The number of 'passed' charters does not necessarily represent areas of the system that need no more

testing, but may be interpreted that way by a waterfall impregnated mind. In conclusion, the 'Progress-o-Meter' readout will be confusing and fuel insecurity and in turn, dismay with the test manager. Although testing *is* done properly and professionally, revealing lots of bugs, it *looks* like a process running wild when only judged by these metrics.

Communicating this to the project manager is as difficult for me as for any other. The ultimate problem is, that though progress is shown every day, it looks like progress in the wrong direction, so the remedial action, fit for a waterfall model but unfit for session-based exploratory testing, would prove disastrous, essentially preventing adding new information or value to the process.

5. SUMMING UP

In our experience, the problem with introducing and working with exploratory testing and session-based test management in a fixed-price, fixed-time environment has to do with the underlying model, which is usually a waterfall-like model of the project. The demands for progress, measured by the amount of tasks done per time unit being checked off against a finite list, do not correlate with the evolutionary nature of Session based testing.

In introducing exploratory testing and session-based test management, our facts to persuade both supplier and client should center around the working method and the opportunities that are offered by these::

- That all testing need not be carved in stone prior to the test
- It is okay to collect test ideas as early as possible, and not waste time to flesh them out as test cases
- That problems can be pursued and that changes can easily be absorbed on the fly
- That important information and bugs can be uncovered quickly

Any mention of not writing test cases in advance will spawn huge debates and discussions and may lead to an unfortunate disapproval of using these methods at all.

Using tools to record and keep track of the test often can lead to confusion, as sessions are usually not well supported in the mainstream tools available. There is also the problem of getting past the ready made progress reports within the tool.

Documenting the tests is in fact not much different from any other kind of testing, although there's no ticking off test scripts.

Reporting can be much improved by using the low-tech dashboard, but reporting is often met with a demand for a 'Progress-O-Meter'. If not provided by the test team, the project manager will invent his or her own, probably based on the wrong model.

6. ACKNOWLEDGMENTS

Our thanks to Tim Coulter for his patient review of our paper.

7. REFERENCES

[1] Bach, James Session Based Test Management summary http://www.satisfice.com/sbtm/index.shtml

[2] Bach, Jonathan Session Based Test Management (first published in *Software Testing and Quality Engineering* magazine, 11/00)

http://www.satisfice.com/articles/sbtm.pdf

[3] Bach, James A Low Tech Testing Dashboard *First presented:* STAR '99 East, 5/99

http://www.satisfice.com/presentations/dashboard.pdf