SWET2: Swedish Workshop on Exploratory Testing 9-10 april 2011 Göteborg
Peer conference on Test Planning and Status Reporting for Exploratory Testing

Exploratory Testing is often about test execution, but there are many things left to discover about test planning and status reporting when using an exploratory test approach.

Test planning can for instance mean test coverage, time estimation, resource allocation, test project analysis, test strategy, rules of thumb.

Status reporting can be qualitative & quantitative, incremental, final report, test coverage, gut feelings, metrics, oral & written etcetera.

Organizers:

Henrik Emilsson
Martin Jansson
Rikard Edgren

Delegates:

Azin Bergman
Christin Wiedemann
Fredrik Scheja
Henrik Andersson
Johan Jonasson
Klaus Andersson (could not attend, replaced by Robert Bergqvist)
Ola Hyltén
Saam Koroorian
Sigge Birgisson
Simon Morley
Steve Öberg
Torbjörn Ryber

Test Planning and Status Reporting, Dealing with the Unpredictable

Azin Bergman

Test planning for me can mean different things. What is part of test planning I think differs between different perspectives and context?

No matter what I think, test planning is a bit of a paradox since it means you have to do the planning when you know the least about how things will turn out whether it being an exploratory approach or not. That is the case the first time you do it anyway. No matter whether exploratory or not I have also had “limitations” or “project constraints” or “external decisions” with regards to time plan, scope and resources to some extent. I think status reporting is even more difficult than test planning in some cases. How do you communicate to management and other stakeholders that don’t really understand test in depth but is demanding status reports every week or every day.

I have a case when I did it for the first time and I started out with creating a test strategy and a test plan. I did test estimations and estimated test resources since I was obliged to do it, this part didn’t turned out as I planned. Basically my testers didn’t have enough test knowledge and the estimated time to test was too short and also development was taking more and in some functions much more time than estimated. With regards to status reporting I tried to show what areas we had covered and what remained and also the bug status and where we were in terms of correction. During acceptance test with customer, I reported every day both to externally and internally. At the end I wrote a test report and I did not use the S-curve (don’t think it is that good). I rather focused where did we have the majority of bugs and why did the customer find the bugs that were severe and not us. But the truth is management want to hear executed 100% of function test but what does that really mean? What was tested and how, how do you know that the testing was good enough? You don’t, you don’t have a clue. So the truth is that you also need to educate management so that they understand and can ask relevant metrics or status reporting.

The next time around I did the estimations for a project I knew my lessons learned during the first project which I used. The end result was that the total estimation I did was correct (the total difference being 40 hours or something, the total time estimation was thousands of hours) and also the number of test resources. The project manager came by and asked how I did it. Maybe I was lucky but the truth is that I knew each and every test resource and each and every developer and all the other tasks (you can of course argue what tasks and so on and if all of it were necessary, I would say no to some). The time estimation I did was not based on what I thought is should be (if that was the case I would have removed 1/3 of the total number of hours) but based on the knowledge and skill of the test resources and the developers. Now this is a contradiction but it is the way it had to be but was it right? I suppose there is some subjectivity to the answer there is one additional factor to this and it is the fact that I knew the customer and this is a customer that requires perfect quality.

I think there are different kinds of test planning. When you are a vendor providing a product for a customer and you want to do profit (otherwise there will be no test planning, test and no company) you need to do it with regards to the cost it will drive. When I have done my test planning I had to check that towards the project plan for in the end I cannot do test planning that contradicts the project plan (I can of course raise my concerns and issues but the decision is somebody else’s to take)

If it is an internal activity, you need to do the planning and conform to the project plan but it is still easier to cope with unpredictable things and it is easier to add one more person without the impact

that the business case will blow. Once again it all depends but this is my experience when working in different companies. Somehow the whole thing with planning is that is never turns out as expected but I feel that if I have given it some thinking and run through risks and other issues it makes me more prepared for the unseen that will happen.

Right now I'm working in a project where there are 7 parties and the unforeseen will happen and there are lots of things that can turn out different from what I plan. So I have written the test strategy and that will be referred to in the contract and I have also created an excel sheet that contains the test resources needed that I expect. I have done a rough estimation of time needed for test execution or should I rather say the time for this is not really only my decision. But the striking thing is also I can do all these things but there are a number of things that could happen that would change everything for instance if vendors development is delayed, vendor delivers poor quality, or I get resources that don't meet up with my expectations or the vendor doesn't do what we have agreed upon or politics change the scope and so on. And one thing I'm sure of is that it will change. Unpredictable things will happen. My approach here is to do it much more exploratory than I have ever done and for the first time I feel I can do so and it not limited by useless processes and management interfering in a great extent. I feel that I have the confidence I need from the organization and right now there are no requirements how I will report status, more than I should do it once I week. I think it will be up to me to report progress but since there are 7 parties involved it will require me to do so to all parties and so that everybody involved understands what I'm communicating. I will report the truth on how we proceed and focus on what is important information so that management can make the "right" decision.

xBTM – The beauty of combining session reports and mind maps

Christin Wiedemann

In the last couple of projects I have worked on, I have embraced a combination of *Session-Based Test Management* (SBTM) and *Thread-Based Test Management* (TBTM) that I like to denote xBTM. The main difference compared to previous projects has not so much been how the actual testing has been executed, but rather in the test planning and status reporting.

These days I actually use the test plan! And I have read and re-read the test report. Using a mind map to track test status has worked wonders. Keeping the mind map up to date has been much easier than updating any other kind of status document, which means that it actually *has been updated*. As it turned out, even the developers liked it – they preferred looking at the mind map over using our bug tracking tool!

The project

The customer runs an application that generates output which is stored daily in a single XML file and later used by the customer's own applications to derive vital business statistics. There were defects that needed correction, and the customer also asked for new features regarding how log on/log off was recorded. The corrections as well as the new requirements should be implemented in a new file that was supposed to be generated in parallel with the old file for a transition period. It was a small project. I acted as project manager, test manager and tester. Later in the project I was joined by a second tester. There was one single developer.

TBTM

My first step was to make a mind map containing all my test ideas as *threads*. Since I am very fond of open source products I used the software FreeMind[1]. In the mind map in Figure 1, e.g. *File Name* is a thread, *XML file* is the actual product, *ID* denotes a defect, and *CR* denotes a new change request. The test threads are grouped in two ways. Most groups represent *key areas*, e.g. *Generation* which means *file generation*. Other groups are formed based on the *type* of testing, e.g. *Stress testing*. In general I use key areas to group functional tests and test types to group non-functional tests. In some cases I felt short notes were needed to explain the thread, and in those cases I attached text files – *knots* – to the thread. These files are shown as red arrows in the mind map. You click on the arrow to open the text file.

[1] http://freemind.sourceforge.net/wiki/index.php/Main_Page

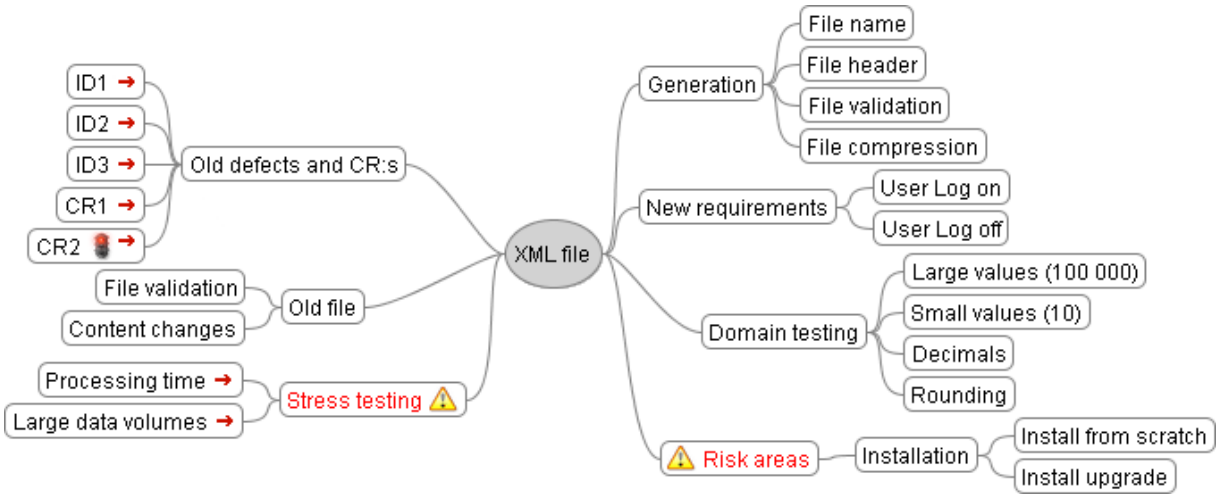


Figure 1: Example of TBTM test plan as mind map.

I tried using colours and icons to make the mind map easier to read. The warning signs mark areas that I judged to be high risk and/or especially important. The stop light marks a thread that was *tied off*. That particular change request was retracted by the customer just before the project started. The initial mind map as it was before I started testing made up my *test plan* and was sent to the customer.

During the test period I would constantly be updating the mind map and it would always give me an accurate picture of the current status of the testing. As soon as I started working on a thread I would mark it with a smiley, see Figure 2. Threads where I found defects were marked with a red cross, and threads where I felt sufficient testing for delivery had been done (not the same thing as claiming to be done testing!) were checked off in green.

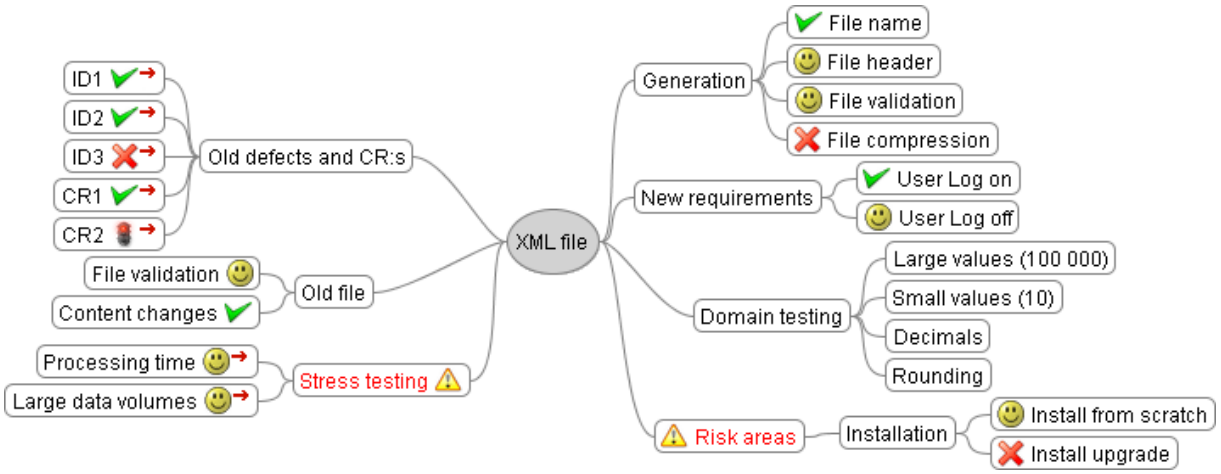


Figure 2: Test status (updated test plan).

Initially my goal was to write daily status reports containing a few short notes on what I had done. In reality I only kept this up for four days. With my constantly updated mind map and the occasional session report (see below) I really did not feel a need for it.

When there was no more time for testing, I took the current status of my mind map and used it as my *test report*, see Figure 3, and sent it to the customer.

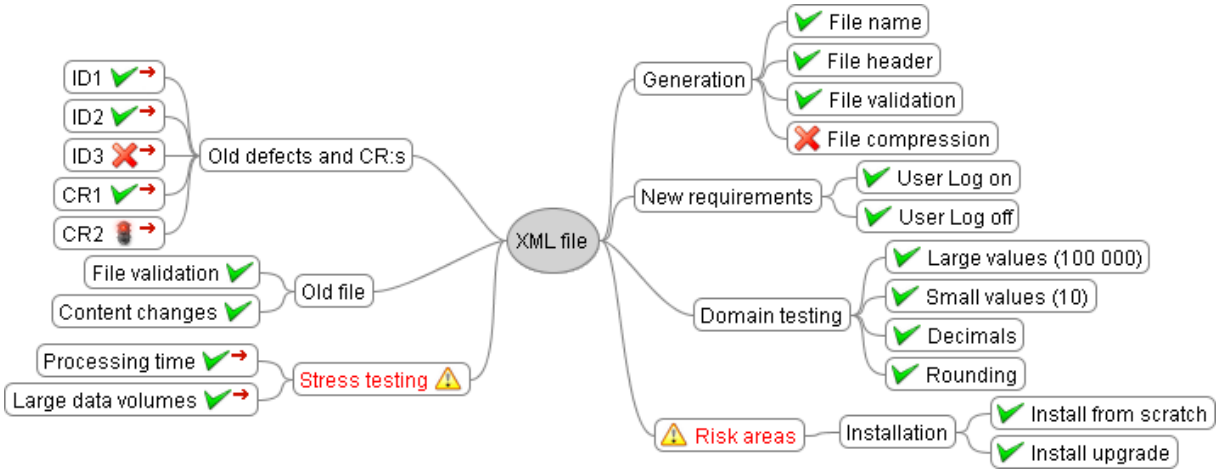


Figure 3: Test report.

SBTM

I do still like SBTM and I find the session reports as well as the time-boxing very useful, so when circumstances would allow, I created test charters and ran time-boxed test sessions. Typically a couple of threads would make one session, but in some cases one thread deserved a session of its own. For example, looking at the test plan in Figure 1 the two threads *File name* and *File header* belonging to the area *File generation* would be tested in the same session, whereas *Install from scratch* and *Install upgrade* belonging to the area *Installation* were tested in separate sessions.

In most cases I would draw a simple diagram – *pattern* – for each test charter, see example below. I prefer using yEd[2] for my diagrams.

[2] http://www.yworks.com/en/products_yed_about.html

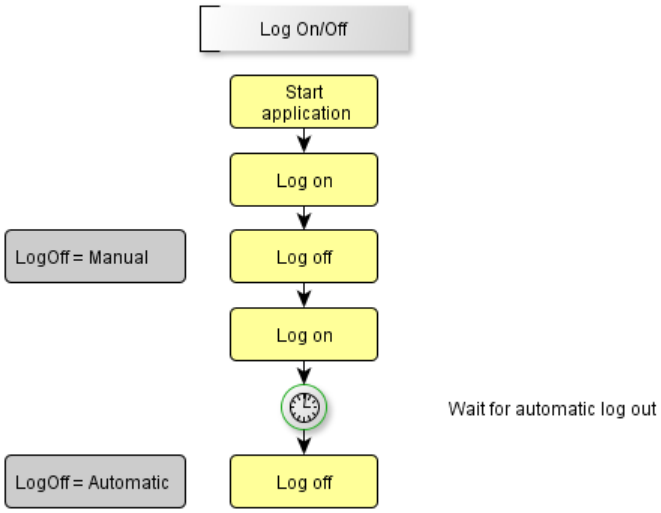


Figure 4: Part of test charter.

I wrote short session reports according to an Excel session report template, see Figure 5.

Log On and Off		Log On and Off	
Charter Name	Log On and Off	**Detailed Notes**	First automatic log off after new installation gave strange error message, should be investigated more.
Key Area	New requirements		
Iteration No	2		
Execution Date	2011-02-28		
Baseline	Version 0.1		
Tester	Christin		
Time Actual Test	10		
Time Total Session	20		
No of Issues	2		
Issue No	ID111, ID222		
Session type	Analysis		
Comments			
Reference	LogOnOff.graphml		

Figure 5: Session report template (Excel).

Using the AddQ tool SBTExecute[3] I could then derive metrics such as *Total session time* (for all sessions), *Number of test charters*, *Number of test charters per key area*, and so on from my session reports. However, since I was still experimenting in this project, as well as working on the template and the tool as I went along, I could not obtain any useful statistics. Also note than any metrics derived would only be valid for my SBTM sessions – and a considerable part of the testing was done according to TBTM.

[3] http://www.addq.se/styr-upp-dina-utforskande-tester-med-sessionsbaserad-testning-sbtm/

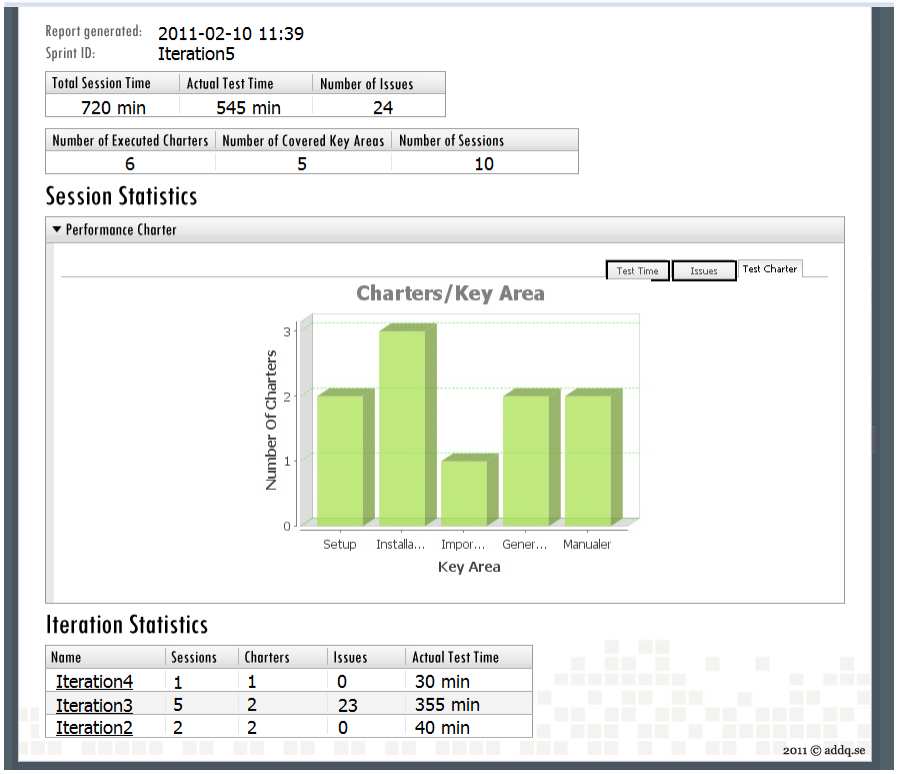


Figure 6: Test report summarizing statistics for one iteration containing several sessions.

Summary and Conclusions

I remember describing at SWET1 how I actually produced more documentation with an exploratory approach to testing than when running scripted tests. Now that I have had more time to try different ideas I have finally reached a stage where the documentation is also actually useful – and gets used!

Case description of how ET was applied in system development within medical related industry

Fredrik Scheja

I would like to present a recent case, a project I have been test responsible for. The assignment is briefly presented below and will hopefully be of interest for the people taking part of the SWET2 conference in Gothenburg, april 2011.

The assignment

The project is a development project for a company in the medical business who has a need for a redesign and further development of their old clinic system, where all patients and treatment data is stored and administrated. The project started the fall of 2009 and has been ongoing until the spring 2011.

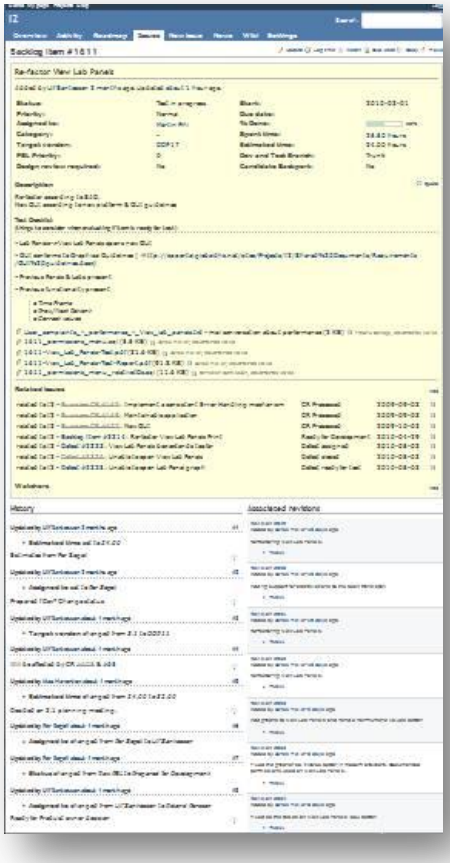
The development team of 10-15 people was set up as a distributed team located in Gothenburg and Lund, Sweden and Denver, USA and the development process has been carried out using a Kanban process framework.

The QA approach

Since this system will operate in the medical business at the company's clinics all around the world, and handle medical data the level of system quality is of high importance. This fact was made clear for every team member from the start and has been However, this particular system was not meant to be classified as a medical related system in the near future and therefore no consideration of the industry stated regulatory requirements (as for example FDA) was necessary to consider.

The test strategy

The system test group of this project was staffed with 2-3 resources and an exploratory test approach was used. This was conducted with strong focus of the individual tester´s freedom and responsibility to perform all test tasks from planning to reporting. The Kanban flow consisted of a swimlane of Backlog Items (BIs) and every one of these were assigned to a tester who took great personal responsibility in preparing the testing and discuss the solution with developer and business representatives, while continuously executing tests on this particular functionality. When the test coverage was considered on an acceptable level and no critical bugs were left to fix, the BI was completed with a test report and sent to the orderer of the project for approval.

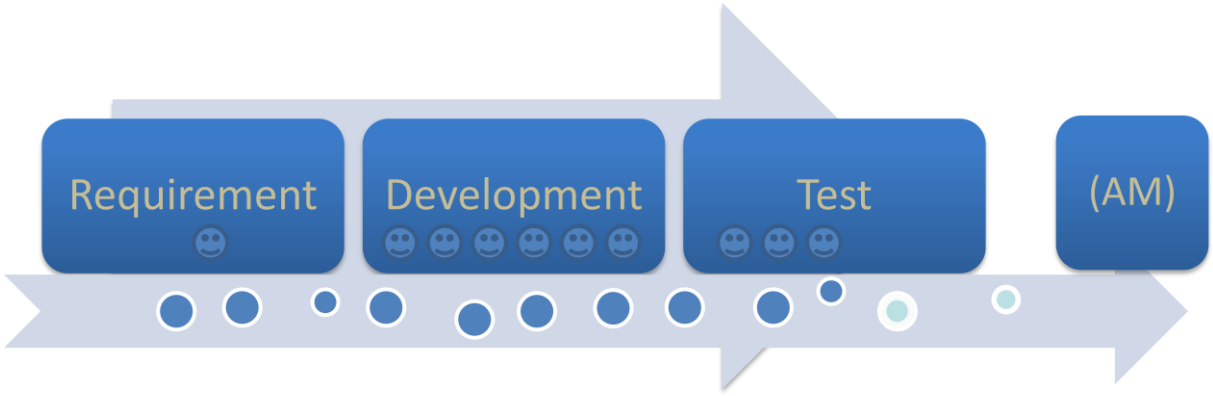


The test planning and reporting

The test planning and reporting has been performed in the same project management web application as the rest of the project team has been worked in, Redmine (www.redmine.org) This is an open source tool and all

different issues in the project; BIs, Change requests, defects reports, product and projects risks has been submitted into this system. On a BI level, the test planning and preparation information has been added as checklists to that particular BI, making the developer aware of how the functionality under development will most likely be tested. This will inevitably influence the developer to deliver implementations of higher quality. The BI in redmine will in this way become a container of everything from business need description, architectural solution discussions, source code references to test checklists and reports and much more. Everything that is relevant to the development of that particular BI.

The test reports are also added directly into the Redmine system and into the BI under test. The reporting is done in free text and contains a short summary, test basis references, test coverage information, and information regarding open issues. The reports are written when the BI is covered with a reasonable amount of testing and when the level of quality is on a reasonable level.



I would like to sum up this case with a quote from one of our developers when he summarized the feeling in the project (freely translated into English by me, Fredrik Scheja):

> *"I even looked forward to see an incoming bug report since the discussions around it really lighted up my day"*

This is interpreted by me as an effect of working with really responsible and professional developers and testers in a creative and efficient development process.

When the tool is more important than the content of your report

Henrik Andersson

What do you do when you “have to” report your Exploratory Testing in a tool not suited for this purpose?

All my team’s testing are based on sessions, that includes performance testing, API testing, E2E scenarios, but also creation of scripts for test automation that will run on our nightly build.

I´m currently spending some days trying to squeeze my reporting into Microsoft Team Foundation Server, Test Manager.
The reason for this is that there is a “corporate” decision that we all shall use this tool.
I´m trying to avoid the situation that I have to do all my reporting twice, so I explore the possibilities to do all my planning, traceability and reporting in TFS.
So far I see some possibilities that we could use what is called test cases in the tool as charters and connect them to user stories (for the required traceability). When we perform a session we can either write our test notes in the test case or attach a test note file from another tool such as session tester or rapid reporter.

One problem that we run into is that what does pass or fail of a session mean. I´m leaning towards setting it to passed when we executed the session and consider it valuable and fail if we don’t accept the session as an accountable session.
So it has nothing to do with if we find a bug or not in a session; it is about the quality of the session.

I believe there is a possibility to customize fields so I can enter my TBS metrics, but as far of today we are not allowed to do any customization, the theme is that everyone shall have the same. This worries me a bit since if all shall have the same set up, I´m now running a great risk that others will look at my project and compare it with others, but not asking me and not knowing that what they see is not even close to the same as other projects.

As you can see I´m not finished, but by SWET2 I will have answers to my questions and will be able to have a full presentation on what I ended up with.

ET + inexperienced testers = problem?

Johan Jonasson

A while back I was asked to be the test manager for a relatively large in-house project at a client. This client had outsourced most of its software development and maintenance to a number of different vendors and had in effect done away with both programmers and testers from their organization. The project in question was to be developed 95% in-house since there wasn't a need for much new code, instead existing systems were used to create new "offers" and combination of services to this company's customers.

As a result of not having any testers, I was randomly given two people from the customer care department to "do the testing" as the mission was described. These two were both 20 years old, zero testing experience and without any computer science degree, or any other degree that I know of for that matter.

I will describe how we used high level, user-centric scenarios, checklists and rules of thumb to successfully manage test planning. I'll also describe how the two would be testers pretty much saved the project by being the exact opposite of the archetypical tester at this company (or their vendors).

Also, for the reporting part of this project, we struggled with outdated metrics and KPIs which we tried to overcome by frequent debriefings, OIM model style conversations and feeling-based "metrics". We were able to resist requests for counting bugs and test cases, partly by ridiculing the requests which in retrospect might not have been the best approach.

All in all, my story on planning and reporting exploratory testing is one where I was surprised by what can be accomplished by testers without formal training who have an open mind and a good spoonful of intrinsic motivation to do a job well.

Time estimating and status reporting

Klaus Andersson

Time estimation is always a challenge when it comes to exploratory test. SBTM has a lot of guidance when it comes to keeping track of time during testing but not much about the planning before testing.

When you count on a new project for a customer your project manager is always asking the question. “How much time do you need for testing?” and “Can tell me when you will be ready”.

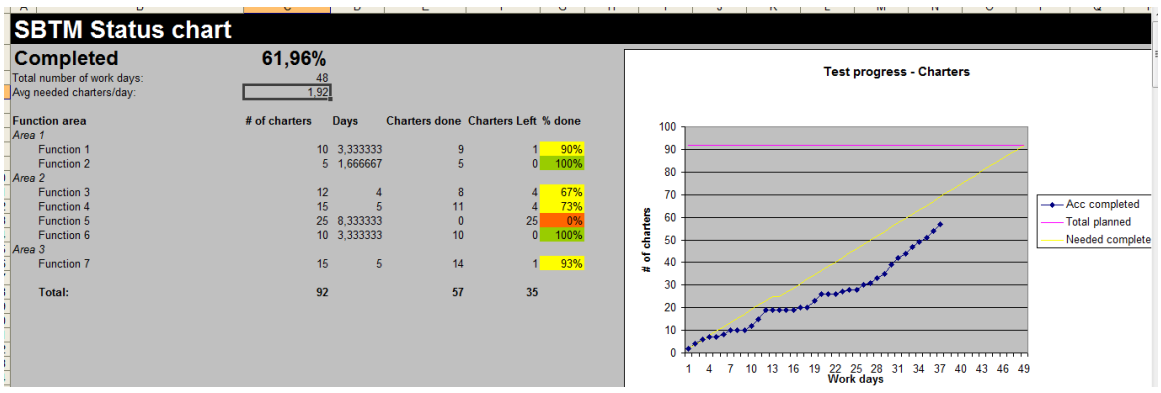
In order to make a good estimation of the time for a specific new function you can use a number of tools.

- Experience (same time as last project, “this project is as complex as the previous one so x hours will probably do”).
- Use a project management time estimation process, like the Delphi method, Lichtenstein method or other similar method
- Time poker/Estimation poker

During my last assignment we made up a way of presenting estimation of the total amount of charters during a complete project. We used a combination of the tools above while influence was taken from the Scrum estimation where areas where broken down into smaller pieces. After each debriefing the estimation was updated to reflect the present status. By doing this the number of charters done for testing corrections also where included in the estimation. Precision using this method was good although some areas where bigger than estimated and others where smaller.

The progress was visible for everyone to see on big screens which provided a good focus on testing progress during the entire project.

It all ended up in good experience of how you can break down areas and also how to present the progress. It also showed the importance of being flexible in the estimation of time.



SBTM Status chart

Completed 61,96%

Total number of work days: 48

Avg needed charters/day: 1,92

Function area	# of charters	Days	Charters done	Charters Left	% done
Area 1					
Function 1	10	3,333333	9	1	90%
Function 2	5	1,666667	5	0	100%
Area 2					
Function 3	12	4	8	4	67%
Function 4	15	5	11	4	73%
Function 5	25	8,333333	0	25	0%
Function 6	10	3,333333	10	0	100%
Area 3					
Function 7	15	5	14	1	93%
Total:	**92**		**57**	**35**	

Did I fail or not?

Ola Hyltén

4 years ago I did my first gig as test manager. I had no experience what so ever managing software testing but I had wanted to try it for some time. It was a small pilot project, an intranet on MOSS 2007 for the clients IT-department. To this day I still ask myself if I did it bad or good. Failure or not?

The cards were stacked against me and I didn't even realize it. My own lack of experience meant that I missed asking the questions I should have asked and my insecurity kept me from questioning things that should have been questioned. I had no testers but me, only the developers in the team to do the testing. I had no contact with the client and all the information I could get from there was through the project manager and the scrum master.

There were risks that were easy to spot in this, besides me being a rookie test manager. The project was a pilot and my company of course wanted it to be successful in order to land a bigger contract on a large scale intranet. Therefore the delivery was important and we risked losing the bigger contract if we did not deliver something that the customer was happy with. The risk of delivering something less than satisfying to the customer was obvious since it was our company's first MOSS project.

I was asked to make sure we had tested all functionality that was built and I did not, my bad, read the requirements documentation. There were a number of webparts and controls to test and the limited time left little time for testing the system as a whole. I tried to communicate this but did not manage to get my point across.

The testers/developers did their best to test and I tried to give them fairly loose reins and wrote scripts, that was a requirement from my PM because he wanted that in his system documentation, that gave them a fair amount of freedom and I told them to explore. "You are not machines. Think." I told them.

Since I had no information about the customer's needs or expectations I had no way of knowing if we tested the right things, or rather if we tested the right way. It was in that respect a shot in the dark.

I had to give all parts of the intranet the same amount of attention since I didn't know what was most important to the customer and the users. This way I'm sure we spent too much time on trivial stuff and not enough on the truly important parts.

When we found bugs, and we did, they were instantly reported to the responsible developer to fix. This was not a good idea I discovered since that meant that I lost one tester during the time it took him/her to fix the bug. The result of this was that when we started out and found a lot of bugs testing came close to a standstill. This meant that the parts we tested first got the most attention since we re-tested them after the bug fixing was done and testing the rest of the system got pushed further into the future. The problem with this is that in that future was our deadline so we got less time to deal with the parts that were at the end of the list. This wouldn't have been a problem if we had a prioritized list that indicated what parts of the intranet were the important ones and which were the nice-to-haves.

Some of the planning problems I have just mentioned and the one thing that I should have seen but didn't was the fact that the number of available testers shrunk a lot when they were asked to go back to their roles as developers and fix the bugs.

The testing was placed as a separate phase when most of the development was done and therefore the testing could not give feedback to the developers while developing. I'm sure that testing early would have meant that the developers had learnt lessons from testing that would have helped them be better developers.

I was given a template for the test report that was very large and very detailed and I ended up delivering a test report that was huge, obviously the client liked thick documents, and it was far between any useful information. The useful information that was in there went missing amongst all the meaningless information that "was required". I was only given feedback on this from the PM and he liked it. I would have liked to hear what the customer thought of it and if they read it at all. In the end the report basically said "we tested this and that and after a few fixes it worked for us." I tried to put in a word here and there about risks in the functionality but I was not allowed to elaborate on them. I guess our PM wanted to look good and to make our company look good by saying everything works, at least on our servers, with our clients.

We lost the deal on the large scale intranet. The customer was dissatisfied and from what I was told there were a whole lot of things that didn't meet the expectations. The expectations sometimes being way higher than anything we could possibly deliver. Communication was a big problem in this project and there was little I could do about it and that was not my role.

Why do I feel like I failed in my mission? Probably the biggest contributor to this feeling is the fact that I did not get enough information and did not communicate enough with the PM and demanded more information about what was important to the customer. What was important to our company I was aware of but meeting a deadline and not spending more time and money than we had was our problem, not our clients problem. I didn't get any feedback from the client, or much feedback from my PM. That didn't help me make up my mind about whether I did a good job or not. The feedback I got from the testers/developers was primarily positive. They told me when I was being unclear and gave me a chance to be clearer. Some of them enjoyed being testers for a while but far from all.

I learned a number of things:

- I don't want testing done as a separate thing after the development is supposedly done
- I liked trying to inspire developers to be testers, let them get a first person experience of the craft of testing and help them understand testers
- I will demand information and ask questions that will get me the information I need to plan the testing.
- I will challenge required documentation and force the one requiring them to convince me of the necessity and value of it.
- I will ask questions about my own performance if that information is not given to me spontaneously. I can't getter better if I don't know if what I do is good or bad.

Status reporting in a diverse test landscape

Saam Koroorian

We have many testers working with the same set of products. The organizational setup of our test operation includes several departments and managers, spans several countries, across time zones and cultures. As people we have different views on testing due to different backgrounds, experiences, education, values, and expectations. Nevertheless, we work together, test the same systems and report to the same stakeholders.

This is where it becomes complex for us. One team talks about requirement coverage, another counts test cases, and a third is talking about test sessions. The aggregated status message easily becomes unclear. What was tested? How well was an area covered? What is the system health? What about risks & problems?
The questions are common, but the way we answer them differs. Does this really need to depend on how we test or approach testing?

We are currently piloting a method similar to James Bach's "testing dashboard" that would allow us to continue to have a diverse way of testing, including exploratory testing, but align the way of reporting. This method currently encompasses a system map, test scale and assessment guidelines.

Delivering traditional test reporting with exploratory means - Two case studies

Sigge Birgisson

I have worked in two different projects within a big organization. These projects have been very different in contexts and size, but one thing that has been the same is the way testing /is expected/ to be carried out and reported in a more traditional way. I will describe how I managed to adapt to the environment and with an exploratory mindset still deliver the correct and accepted test reporting.

Exploratory test specification

The project was running a form of Scrum like process. In the backlog, there was already for every release a template story for what needed to be done before release. Two of these tasks were something that I as a new tester got in my lap. “Create test specification” and “Run through test specification” were these horrible tasks, that were not even worth a days of work together in the time estimates. First set of these tasks took me about a month to finish with an acceptable test result. During the following 8 release cycles I developed this to a more exploratory approach, still delivering the same type of required documentation and reporting. Part of this was the exploratory test specification which was fitted into the Scrum process.

By dividing the presumed areas of testing from the test specification into their own stories/ tasks in the Scrum tool, I adapted the testing to the process like all the development tasks.

These stories were just “Test area X”, with the underlying tasks of “Perform testing” and “document in test specification”. With this planning setup, I could execute testing using an “Explore > Learn > Adapt” approach, just like the developers doing TDD. At the same time, I always kept delivering the correct test documentation. My biggest opportunity from doing this was actually that I was able to deliver the test specification in the end of testing, together with the test evaluation summary which was just a protocol for referencing the specification of the testing that had been done for that release.

Investing in a metrics reporting debt

The other project was a waterfall process project altogether. Here I was a part of a test team of 6 including a test manager. He is very much the rapid testing type, but with the project context in mind, there is a need to deliver the traditional test documentation and metrics upwards in hierarchy. These include the number of run test cases and the requirements coverage of these, which is possible to visualize in our test management tool. With my background knowledge of the systems environment, my position quickly went towards working closer to the developers, testing on a lower level than the written test cases. This enabled me to dig in more on the product without too much of an interference of test cases. In my own perspective this also gave me the possibility to quickly find some real show stoppers in a short time, probably saving quite some test time later on.

This approach did create a debt towards the test team and process, which had to be managed somehow and sometime. That time came two months into the test phase when we got a delivery installed on our test environment with a show stopper which really stopped me from doing any tests on my areas. This is when I delivered some metrics and coverage, based on the already recorded bug reports which just had to be converted into metrics (test cases with requirement coverage) in our test management tool.

Analysis & Planning for Exploratory Testing in Large / Complex Systems

Simon Morley

1. The use of Feature Walkthroughs to plan and analyse test possibilities, including:
    - First parts of Test Feasibility/Analysis (including test scope)
    - Test Strategy to adopt

    To include:

    - What is a feature walkthrough?
        - How use of a stand-up discussion & whiteboard with flow & control diagrams can help understand what's happening in the system.
    - How seeing the flow of new and existing features helps understand the new feature – see problem areas and start forming test ideas.
    - How test idea formation in the bigger picture can help understand the areas that need more analysis and what the test strategy should look like.

2. It will also look at how the scope and plan forms.
    - Decisions about follow-up investigation & planning.
    - Pinpointing the initial scope.
    - Getting to estimates and discussing this with project/stakeholders

    To include:

    - How the stand-up meetings help break down a big problem into smaller manageable chunks and then following those up – whilst still feeding back to the stakeholders and product owners the ongoing findings.
    - Examples of the importance of dialogue and how test planning is based on an ongoing discussion.
    - Discussions around how lightweight documentation helped planning.

Optional topics

3. Some problems experienced/to consider with exploratory testing in large systems, including
    - Making use of scripting - for example for simulation aspects
    - "Walking in the woods"

4. Lessons learned

Transparency instead of metrics

Steve Öberg

As testers we are often asked to report the progress of our work.

More often than not the request is to show some kind of metrics of how the work is progressing.

In a project I recently worked in I got just such a question from the project manager.

The question that immediately arouse was, in what way do you want the progress reported; metrics, values or status smilies?

We made an inventory of what we already had;

- Long-term plans in the form of test plans and prior knowledge/strategies
- White board with test areas and dependencies
- Short-term plans in the form of simple text files as task lists that we updated on a daily or weekly basis

Setting up long-term detailed plans for the testing in the beginning of the project is tempting, but rarely pays off. They usually need to be updated whenever the scope of a feature or functionality changes.

Short term plans does not demand as much of rework when reality comes knocking on the door.

We used MS Project create a plan that covered the rest of the project, with bars to incorporate each testing effort. The goal was to easily explain what we needed to do and in what order. The end result was visually appealing, but in the end not very successful. The overall impression was too cluttered since we were testing many of the features in parallel. And it needed to be updated whenever functionality took longer than planned to test or develop.

After the initial talk with the Project Manager we realized that he was not at all looking for metrics.

He was interested in whether our work was going according to our plans or if there were any issues that prevented us from doing our work.

We ended up sending him the, what used to be internal, task lists in combination with the MS Project image, to describe the testing progress and what needed to be done.

In the end it was enough to be transparent in what and how we performed testing and showed what we already had in the form of planning and follow up.

Metrics might not be needed if one has got information and we communicate more openly.

Do not underestimate the people you are reporting to, they can take more information than only a few numbers.

Structure of test strategy in a volatile and loosely specified project

Torbjörn Ryber

I have worked for a year in a relatively small project where we applied the models I mention below.

Started out in the project as test manager / tester. Missing useful requirement specifications. Wondered how I would plan testing.

Test Strategy based on how to satisfy the needs of users:

1) Analysis of business processes: -> the basis for overall scenario tests
2) Effect Mapping of user groups and their specific needs -> scenarios and specific functions - data for testing
3) Focus on how the system works technically, created a state map and state diagram: -> this became the basis for both systems design, requirements assessment and exploration test cases of flows
4) Detailed mapping of business rules for when what e-mails should be sent out and to whom -> this is also the basis for testing, no separate test specification
5) Other planning elements as the type of tests to be run, and by whom and when. Included think-aloud scenarios, the pilot test
6) Introduction of a scrum-like approach to control development and testing. Decision to be more responsible for project management on the expense of less testing!
7) The decision to bring in professionals to do interaction design and interface design after conducting user testing.
8) Summary of the different models that I used to get a good test planning / design.
   An extremely simple process for getting basic procedures and cooperation.
   Test strategy that has emerged in parts for a long time.
   No test cases at all but only a number of different maps at different levels, etc.
   Reflections on what parts I think other roles / persons should be responsible for.