Tallinna Ülikool
Informaatika Instituut

Error Life Cycle During Test Execution Process. The Case of Ixonos Plc Company

Bakalaureusetöö

Autor: Maxim Kukushkin
Juhendaja: Inga Petuhhov

Autor: ..... „ ..... „ 2009
Juhendaja: ..... „ ..... „ 2009
Instituudi direktor: ..... „ ..... „ 2009

Tallinn 2009

Autorideklaratsioon

Deklareerin, et käesolev bakalaureusetöö on minu töö tulemus ja seda ei ole kellegi teise poolt varem kaitsmisele esitatud.

.....	.....
(kuupäev)	(bakalaureusetöö autori allkiri)

Table of Contents

Introduction

The Mobile phone market was and is still growing and developing very rapidly – new phones with brand-new features are dropped to the market very often. As a result of such fast development, competition between mobile manufacturers becomes very aggressive and intense – each one wants to be on top. One of the main factors in this competition has become the quality of products. Quality can be assured by software testing.

Testing – is one of the best ways to indicate software quality, find errors. Also results of testing can help to evaluate in what shape the product is and help to decide if it is ready to be released to market. The value of software testing has much increased and it became very important part of mobile software development general process – many resources are invested in all testing activities.

Author is working in international company, called Ixonos Plc, which performs software-testing activities with main focus on Nokia smart phones. Head office is situated in Finland, where additional services are also provided – project management and software development solutions. Tallinn subsidiary is fully responsible for software testing and has currently 67 testers, 34 of them have ISTQB / ISEB Foundation Certificate in Software Testing.

Eventually people become professional testers, but most of new employees who start to work in company have no experience in testing and are making first steps in this field. Usually new people do not know much about testing – in this situation it is difficult to orient in large amount of general information and to derive necessary and the most important things. That is the reason why company needs proper start-up material, which can give overview in testing, show and describe certain work-related processes.

Despite the fact that there is much information about testing available in Internet, there is no one united material adopted for company needs that new tester can read and see how business is done, understand his place in testing process, his responsibilities. Material should be as concrete as possible, information in it should refer mostly to company internal environment and such material should have minimum of “water”.

The core of testing performed by Ixonos is searching and finding errors in software. Executing software is the very visible process. But the most important process behind

software execution is maintenance and tracking of discovered errors. It is not much visible as testing, but it would be fundamentally wrong to underestimate its importance. While testing is just shell, error-related activities is a core.

Purpose of the work is to:

- Analyze the process of creating error report and following life cycle of it from Detected to Closed state.
- Use this analysis as a future development idea for creation the reference manual about testing related activities for company internal use for beginners.

To reach this target author will perform following actions:

- Analyze what general information is necessary to present to new tester during starting period.
- Examine usual problems with tools that new tester faces during starting period.
- Discover needs of new tester when he is starting to execute tests and creates new error report about found defect.
- Discover needs of new tester when he is starting to track the status of created error report.

On the basis of these points author will make detailed description of error life cycle, how it is realized on practice.

The work will be structured in the following order.
First Chapter will be analysis of needs of new tester, possible problems. Also author will provide his own experience when he started to work as a tester. This Chapter will be justification of all following chapters and will answer the question: why author presents the material in exactly that order, how it is written in Contents.

Second Chapter will give general knowledge about testing mission and principles, that new tester should know.

In third Chapter author will describe tools and interaction between them during testing. Concept of test case, test set and error will be introduced.

The biggest will be the last forth Chapter about error status changes during life cycle from "Detected" to "Closed". It will be shown how error report is written, all possible statuses

changes, interactions and also described how these interactions affect on test execution process.

Information given in Second, Third and Forth chapters is aimed for beginners in company and will be used as a fundamental base for creation of full reference manual. Work will be ended with conclusion; the evaluation of received material by more experienced specialist from company will be added to conclusion.

1. Analysis of tester's needs and problems

Before starting to create material author needs to analyze the needs of new tester, usual problems and misunderstandings that new tester meets. Analyzing these issues will help to create plan of the work and define the contents. Results and conclusions of this Chapter will be used in all next chapters.

Usually new people who start to work as test engineers have no experience in software testing. That is the reason why it would be wise and useful to give general information about testing. First chapter will give theoretical information about testing. Such theoretical base is mandatory in work. Tester should always keep in head some principles and knowledge of:

- What is testing and what targets it have

Usually testing is understood as a process of software quality improvement. But it is wrong image that author also had when started to work as a tester. There were some other misunderstandings about testing. To disprove most common wrong thoughts about testing first chapter should include:

- Several common myths that usually people have about testing

With help of these myths it will be easier to show testing nature and content.
After tester gains understanding about essence of testing his view changes and he understands that testing as a process is much more complicated than he thought before. On this stage tester should learn the principles of testing. There are some very general ideas that were developed during many years and they really provide very useful knowledge. The final part of first chapter should include:

- Testing principles

First chapter will represent the synthesis of material taken from different sources, because it is pure theory and author will only add some things directly related to company work in such way that reading theoretical material will not be useless and boring process.
The main purpose of the first chapter is to introduce the basics of testing.

Second chapter will be used as instruction for using working tools in every day work. Company employees use mainly two tools for testing. One is for executing software and running tests, other is error database where all errors are written.

Main task of the chapter is to show that these tools are in very close connection, it is impossible to work effectively using only one of them, and both are used for work.
Firstly author will give short description of:

- Tool for tests execution – officially Test Management Tool

When new tester starts to work with Test Management Tool, he faces some terms that he should clearly understand:

- Terms: test case, test set and error

Next critical stage where new problems are met is the point when tester finds wrong behavior in tested software. His acquaintance with errors-related activities should be started from this moment:

- Error database – officially Error Management Tool

From previous experience author remembers how difficult it was to see the whole picture of what he was doing. It was unclear what means test case and how it was connected with error. New tester needs clear understanding of relations between test case, test set and error, between Test Management Tool and Error Management Tool.
It will be explained how test execution is done. Reader will be instructed about passed test case, failed and N/A, how they are marked and what do they mean. Also new tester should receive knowledge of how error affects on software and on test execution process. How error is marked to the test set.

Chapter should be supplied by plenty of screenshots and comments to them. Visualization gives more information and better explains the concept of test case, test set and error. Also this chapter will act a role of smooth transition to the main Chapter about error life cycle. It would be difficult to understand error life cycle without understanding how test execution is done.

Next chapter is the biggest and it will tell about error life cycle. Error-related activities are usually causing most of the questions. Firstly because error related activities are not obvious as test execution itself – it is behind test execution. The second reason – it is hard to find something useful in Internet about error-related activities – it very depends on the tool (database) used.

The first step is logical and it will show:

- How error report is created

Author will describe all necessary fields, which need to be filled when writing error report. This should be done with a reference to author's experience and explanation of each field. It will be not enough to say that error can have critical, major or minor priority, but each priority should be described, what it means, when tester can set critical priority etc.
Next logical step is description of:

- Error life cycle and error statuses

During life cycle error can be in different statuses and amount of statuses is big enough to get confused amongst them. That is the reason why author will divide all statuses into two groups:

- Main transitions in life cycle
- Possible transitions in life cycle

Author and also other new people had problems with understanding how and why statuses are changing and how those changes affect on test execution. That is why transition should reflect the changes also in test execution.

There are much people with different roles involved in process of error correction and everybody must act according to his responsibilities. In the end of the chapter will be written the list of tester's responsibilities during life cycle. It will give the final accord to description of error life cycle and according to it tester will see his place in this process.

There are some requirements that author would like to fulfill when designing reference manual:

- Material should be concrete as possible, it must have minimum of water
- Material should refer mostly to company internal environment
- Any theoretical information should be possibly supported by experience, practice
- Chapters should be independent and flexible enough to be used afterwards in reference manual without many changes to be introduced in structure

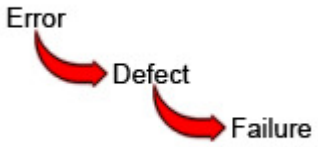
2. Software testing, errors

Software is used in different areas, from simple calculators until extra-complex space programs; depending on software complexity number of errors usually increases. Software containing some amount of errors can lead to many problems – from money loss and even until death.
Focusing on mobile phones – in case of bad software on mobile phones, mobile manufacturer can lose reputation of reliable producer which can also cause lose of profit. Fewer defects the released software has – more successful the phone will be. A typical mobile phone contains millions lines of software code and it would be very difficult to achieve 100% defect-free software, that is the reason why software sometimes does not work as it is expected.

2.1. Error – Defect – Failure

Simple principle of "Error, Defect and Failure" can easily explain why software sometimes can fail.

"*A human being can make an error (mistake), which produces a defect (fault, bug) in the code, in software or a system, or in a document. If a defect in code is executed, the system will fail to do what it should do (or do something it shouldn't), causing a failure.*" [Thomas Müller, 2008]
Picture 1 shows it graphically.



Picture 1. Effect of error

2.2. Testing mission and content

There is much misunderstanding concerning the mission of testing. Most common mistake is to think that testing makes software better and increases the quality.

"*Testing is a way for finding defects in software under development. The main objectives of testing are to gain confidence that software behaves as expected and whether it works without errors. Testing does not give a guarantee for absolute correctness. Instead testing determines the circumstances when deviations from expected outcomes will occur. In any case testing should help to reduce the number of defects in software.*" [Norbert Ruck, 2008]

Other "myth" is that testing consists only of running tests. Test execution is part of testing, but not all of the testing activities. In general, testing activities are represented and described in concept called Fundamental Test Process (FTP). It consists of five points and executing the software is only one point of FTP. [Brian Hambling, 2008]

- Test planning and control
- Test analysis and design
- Test implementation and execution
- Evaluating exit criteria and reporting
- Test closure activities

Under "Testing" is usually understood "Test implementation and execution" stage. But FTP clearly shows that "Testing" is much wider then just running tests: 2 activities before and 2 activities after should be also considered as "Testing".

2.3. Testing principles

Testing as the whole process from the beginning to end is very complicated. That is why some testing principles were formulated during many years from very different sources. They are general and mostly aimed to give a guide to the tester and to prevent from making common mistakes. [Thomas Müller, 2008]

Principle 1 – Testing shows presence of defects
Principle 2 – Exhaustive testing is impossible
Principle 3 – Early testing
Principle 4 – Defect clustering
Principle 5 – Pesticide paradox
Principle 6 – Testing is context dependent
Principle 7 – Absence-of-errors fallacy

First principle means that testing does not increase software (SW) quality – it only shows how mature the SW is, by indicating the presence of defects. Developers will increase the SW quality by fixing errors that were found during testing.

Second principle tells that it is impossible to test everything. Impossible to test all combinations of data inputs – and it means that there should be no 100% confidence in SW. It could be 99,9% but not 100%, because there is always something that was not tested deeply.

Early testing as third principle shows that testing is necessary as early as possible in SW development process. Of course, it would be difficult to test SW that is not yet written. But it is possible to check requirements and specification and to find logical errors there. The cost of correction of errors found as early as possible is very low comparing to latest stages of software development.

Fourth principle explains that most of the errors are found in several modules/areas of SW. And this is true: while testing it was noticed that some features are traditionally work fine, meanwhile some other areas are always spoiled each time by new errors. Tester should always pay more attention to such areas.

Pesticide paradox (Principle 5) tells that running the same test set during some period of time will become useless, because it will not find errors. If SW changes, new functionality is added – proper changes for tests are also necessary. Testing should be well planned and give actual results.

The sixth principle explains that different SW should be tested differently. Mobile SW testing should be different from security-system SW testing. Each SW is specific and requires different approach.

Principle 7 means that there will be no point even in 100% defect-free SW if it does not answer to user requirements. Ideal phone (from point of view of SW quality) will be useless if it is impossible to call from/to it.

Additionally to these principles, author found one great idea in testing magazine, it shortly but fully defines role of tester that it could be mentioned as 8th Principle of testing:

"*It is best if the tester is a pessimist. (A pessimist is an optimist with experience). If something does not work, it is good news, because nobody will have the defect later. Better test forces developers to do better work, it informs management about risks, and it leads to lower cost (for repair). Testers bring bad news, but this is their job.*"[Hans Schaefer, 2008]

Testing process will give better result if it is close to these principles. Testing will be trouble-free for tester if he keeps in head and follows such principles. At least some very common errors will be prevented.

Very general and theoretical information about nature of testing and errors was provided in this Chapter. Many ideas were taken from the book, which is called "Software Testing: An ISEB foundation" – Bible for tester who works in Ixonos Plc. Book is used in Company as the main material for preparation for ISTQB / ISEB exam which gives very valuable certificate in testing community.
With the help of this book Chapter gives the right view of testing process and can be used as a background for the next chapters. Additionally it must prepare reader for practice.

3. Tools: Test management tool and Error management tool

In everyday work tester usually uses mainly two tools: Test Management Tool and Error Management Tool. Both tools are in very close connection, and one tool is useless without other.

First of all author will describe Test Management Tool, because it acts as a primary tool for new tester.

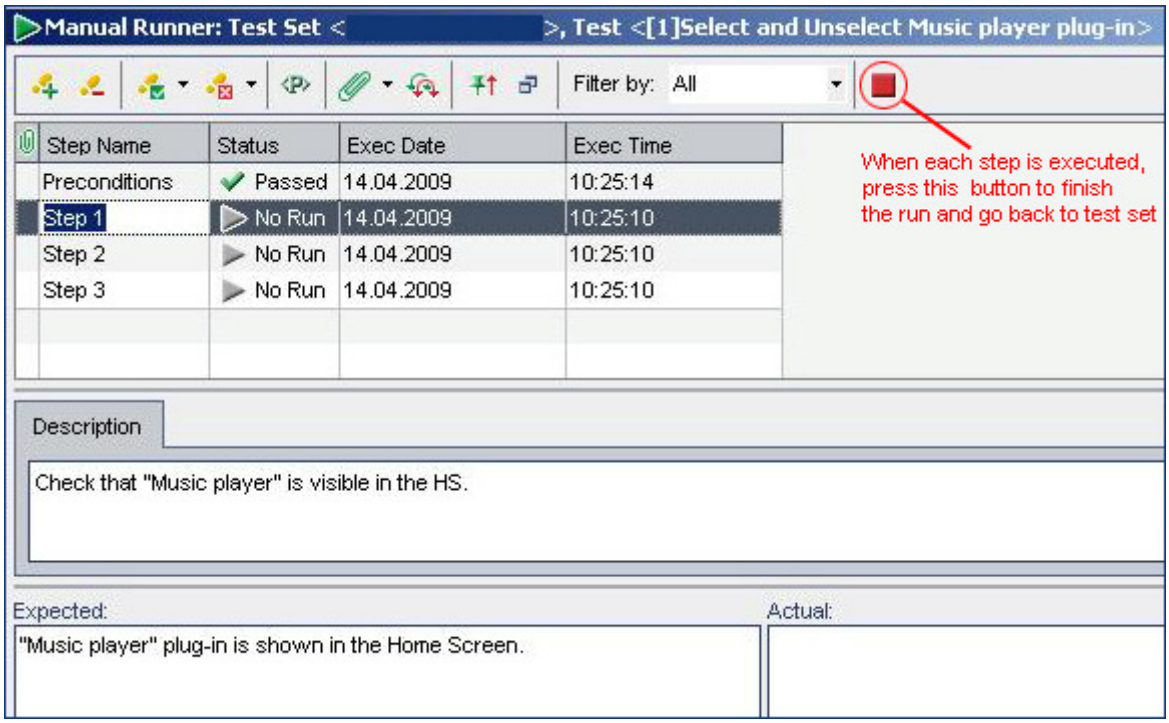
Test management tool – environment that provides support for different testing activities during software development process.

Quality Center (QC) is a Test Management Tool used in Ixonos, product of Mercury Interactive Corporation, which gives control over entire testing process, and includes supporting tools for following activities like Requirements coverage, Test Case Management, Test Execution Reporting, Defect Management, and Test Automation.
But for tester QC mainly provides reach possibilities for designing, maintaining and executing test cases. QC is a program where test cases are written and executed.

During software testing performed in QC, tester faces some basic concepts, which are test case, test set and error. Understanding of those concepts will give a clear picture of relations between test set, test case and error.

3.1. Test case

"*A test case – set of steps (inputs/outputs) that will cover some small part of specification. Test case is prediction of activity that user can/will do with phone. In other words, a test case: gets the system to some starting point for the test (preconditions); then applies steps that should achieve expected result, and leaves the system at some end point.*" [Brian Hambling, 2007]



Picture 2. Test case in QC during execution

Picture 2 shows test case during execution in QC. Test case title is “Select and Unselect Music plug-in” and it has 4 steps (preconditions plus 3 steps). This test case is part of Test Set “Product name_SW version” (hidden on screenshot). Preconditions of the test case have been met (preconditions also considered as step), which can be seen as passed status after the Preconditions field.

Step 1 is now focused and ready to be run. “Description” field gives instruction what to do in this step; “Expected” field shows expected result. “Actual” field was used to describe the possible deviation from expected result (due to organizational changes this field is not used anymore).

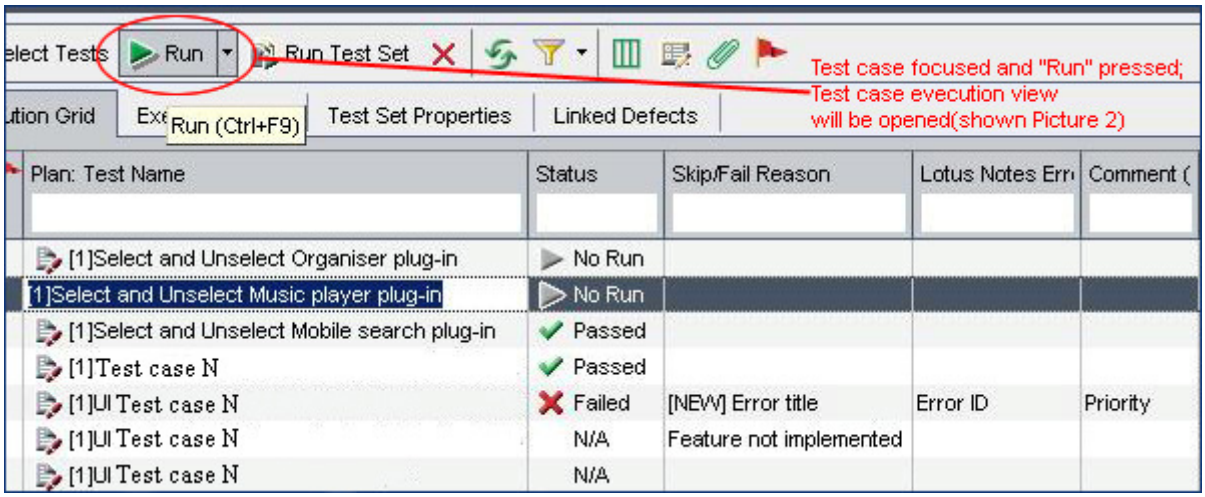
As the actual result (Description) is met for all steps, the whole test case will be marked as passed (after red square button pressed – view goes to test set with this test case marked as passed automatically).

After test case meaning is clear, test set issue can be examined.

3.2. Test set

Test set is a collection of test cases to be executed. Results of test set give overall view in what condition the software currently is. There are many different indicators, like pass rate,

pass rate of total etc., which give very much information to test managers, so that they receive maximum information about software without even trying it by themselves. See Picture 3 that shows part of test set in QC.



Picture 3. Part of the test set with some test cases as "No Run", some are "Passed", one test case is "Failed" and several test cases are "N/A"

Picture 3 shows part of the test set. Before the test case is run, it is shown in "**No run**" status. Pressing **"Run"** button starts test case execution (was shown on Picture 2) and after test case is completed, the status of the run is updated into the test set view. If the actual result is the same as expected results, the test case status will be **"Passed"**. If it doesn't meet the expected results the status will be **"Failed"**. Sometimes test case cannot be run at all and the status is marked as **"N/A"** (Not Available).

Test set execution round is considered as completed after all test cases in a test set are run (no matter with what result) and results are collected into a test results report.

Test case and test set terms are the first things that new tester will meet in work with QC. Soon he will face the error and failed test case.

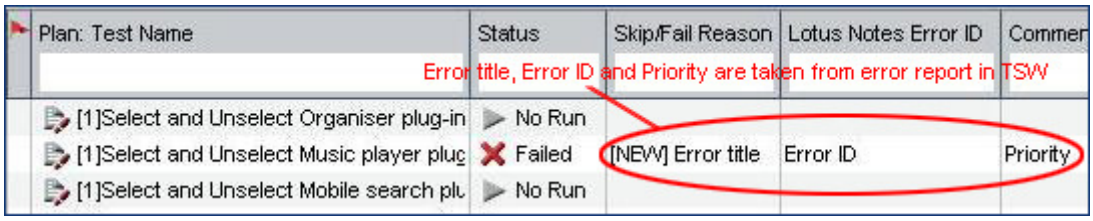
3.3. Error

Errors (bug, incident) – the main object of testing, appear when there is a deviation from specified expected result. Simplifying: when something goes not as it is expected, then probably it is an error.

If tester finds new error during particular test case execution he should fail the test case in QC and write error report about it to Error Management Tool.
Error Management Tool – error database that is designed to help testers and developers in keeping track of reported software errors.

TSW is an Error Management Tool used in Ixonos, based on Lotus Notes environment. TSW allows create error reports and filter them with different conditions (by Priority, by Location etc.). It provides all necessary options for creating, searching and maintaining errors. TSW slowness and inconvenient interface are the main minuses of the tool.

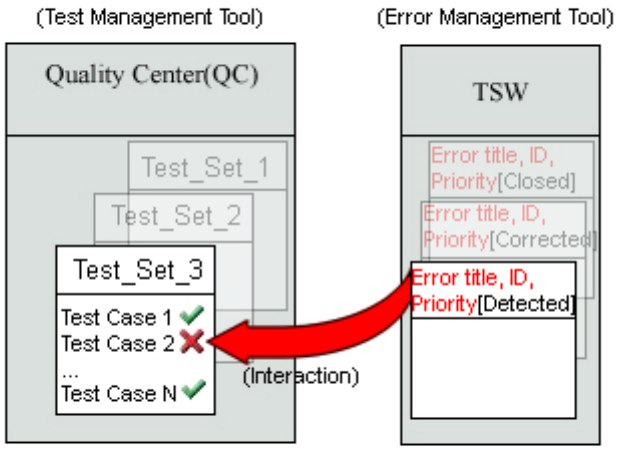
When error report created and saved in TSW – there are 3 main things that must interest the tester: Error title (Actually, tester gave this name to the error), Error ID and Priority. These fields will be necessary when failing test case in QC. They should be inserted to proper fields in QC.



Picture 4. Failed case with all required details taken from error report in TSW (<u>Error title</u>, <u>Error ID</u>, <u>Priority</u>)

Picture 4 shows the test case in QC, which is failed by new error ([NEW]-tag is added before error title). Error was just created in TSW and Error Title, Error ID and Priority were copy-pasted from there.
Error title should be inserted to “Skip/Fail Reason” field in QC.
Error ID should be inserted to “Lotus Notes Error ID” field in QC.
Priority should be marked to “Comment (free text)” field.
When this error will be fixed and expected result of test case will be met, test case will be marked as passed.

On picture 5 is graphically shown the interaction of QC and TSW.



Picture 5. QC and TSW relations

Picture 5 shows cooperation between two tools: QC and TSW. It shows that if test cases are executed without errors – tester does not need TSW and only QC is necessary. But when deviation from expected result is found and test case is failed – reference to error database is mandatory: error report created in TSW and test case is failed with this error in QC. And some of the details in error report are marked in QC for failed test case.

From this moment is started the core of the work, because reached the point where close acquaintance with errors-related activities is necessary. Tester faced the wrong behaviour and have to create error report, that is the reason why author would like to start the new separate chapter about it. Behind the test set execution in QC there is other activity – error tracking. Executing test cases in QC is only “top of iceberg”. The most important things are done in error management tool – in TSW.

4. Error status from "Detected" to "Closed"

"*At a basic level incident management tools are used to provide two critical activities: creation of an incident report; and maintenance of details about the incident as it progresses through the incident life cycle.*" [Brian Hambling, 2007]

Second activity (maintenance of details about the error) will be mostly examined in this Chapter, but firstly few words about first activity from which error life cycle starts – creation of error report. In the end of previous chapter several words were told briefly about it, but in this chapter it will be done in more details.

4.1. Creating error report

When error is found during testing, then it should be reported to error management tool (TSW database), immediately without delays. But before reporting it, it is necessary to use search option to see if somebody else already reported such error.

"*Fields in the database structure normally include:*

- *Priority (e.g. high, medium, low)*
- *Severity (e.g. high, medium, low)*
- *Assignee (the person to whom the incident is currently assigned, e.g. a developer for debugging, a tester to perform confirmation testing)*
- *Status in the incident life cycle (e.g. New, Open, Fixed, Reopen, Closed)*"

[Brian Hambling, 2007]

These are standard fields that should be present in any error database. Now author wants to compare and see from what points consists real error report in database that is used in Ixonos every day.

To open new report template 

-button should be pressed.

Error report in TSW includes:

- Error title – tester decides how to name the error. Title should be informative and short as possible. Ideally, title should give understanding of error even without reading steps how to reproduce it. For example, good title could be the following: "Search in Contacts application does not give any results". After report is saved, error title should be copied to QC next to failed test case in "Skip/Fail Reason" field (refer to Picture 4).

- Description – usually there is a template that tester should fill. In description tester should mark the following points:
  - "Preconditions" – should include software version, describe some special states of the phone (applications opened, SIM card, profiles, contacts defined, wallpaper set etc.).
  - "Test steps how to reproduce the error" – description of action that developer should perform to reproduce the error.
  - "Expected output vs. monitored output" – description of expected correct result of performed actions against actual results that were received.
  - "Testers sophisticated guess about the failing area" – optional field but if tester has any ideas why error occurred and in what area it can be corrected, it would be not bad to write.
  - "Additional info" – anything else that can be helpful for developers to eliminate the error (for example, link to related errors that can provide more information).

- Error ID – generated by TSW when report is saved, looks like this - AMKN-9H2L8P. 2nd and 3rd letters of A**MKN** are initials of reporter. For example all reports created by author (**M**axim **K**ukushki**N**) will have x**MKN** letters. After report is saved, Error ID should be copied to QC next to failed test case to "Lotus Notes Error ID" field (refer to Picture 4).

- Severity – shows how permanent and serious is the effect of the error on the phone. The finder of the error sets severity from the list: Low, Medium and High.
  - Low – used if error is annoying but does not require any user interaction to recover – normally all UI issues, localization, etc.

- Medium – used if error affects working or requires minor user interaction to recover like application boots itself after pressing some button again.
- High – used if phone is unusable because of an error or software requires a heavy user interaction to recover e.g. removing battery or restoring factory settings.

- Priority – a subjective criterion, which indicates the urgency of error from the business point of view. After report is saved, Priority should be copied to QC next to failed test case to "Comment (free text)" field (refer to Picture 4). Tester should define it by himself from available list: Minor/Major/Critical and Show Stopper.
  - Minor problem – is used if error is not noticed by end user or does not cause remarkable problems. The error with priority minor *may* be fixed.
  - Major problem – is used if error causes remarkable problems but the situation is not common or frequency of the error is rare. The error with major priority *should* be fixed.
  - Critical problem – is used if error does not prevent sales but has an impact to customer satisfaction. For example, an error that occurs all the time can be a critical or error causing heavy impact to the system. Critical can be used also if error prevents part of testing. The error with Critical priority *must* be repaired immediately.
  - Show stopper – is used if error is really preventing something, e.g. sales to certain region or operator, preventing the program to proceed. In other words – error has maximum bad affect on SW. The error with Showstopper priority *must* be repaired immediately. All possible effort should be addressed to solve the error.

- Status – when error report is just created, it automatically receives status Detected. There are several statuses that are indicating error correction progress: from Detected to Closed. It will be described later in details.

- Affected parties – error can affect several products that are marked to this field (same software/components on different phones). Affected Party field is used for setting target for each product.

- Also there are some fields that should be filled: Area, Location, Error type (crash, data lost, incorrect result etc.), Reference hardware, Reproducible (every time, easy to repro, hard to repro), Platform, SW Release, Symbian OS version, Language variant, Affected parties, Testing type (Functional testing (shortly – FuTe), Basic Acceptance Testing (shortly – BAT), ad-hoc etc.), Found at gate, Test Environment (emulator, prototype hardware etc.), Responsible Person(s). But they are not important in context of the topic.

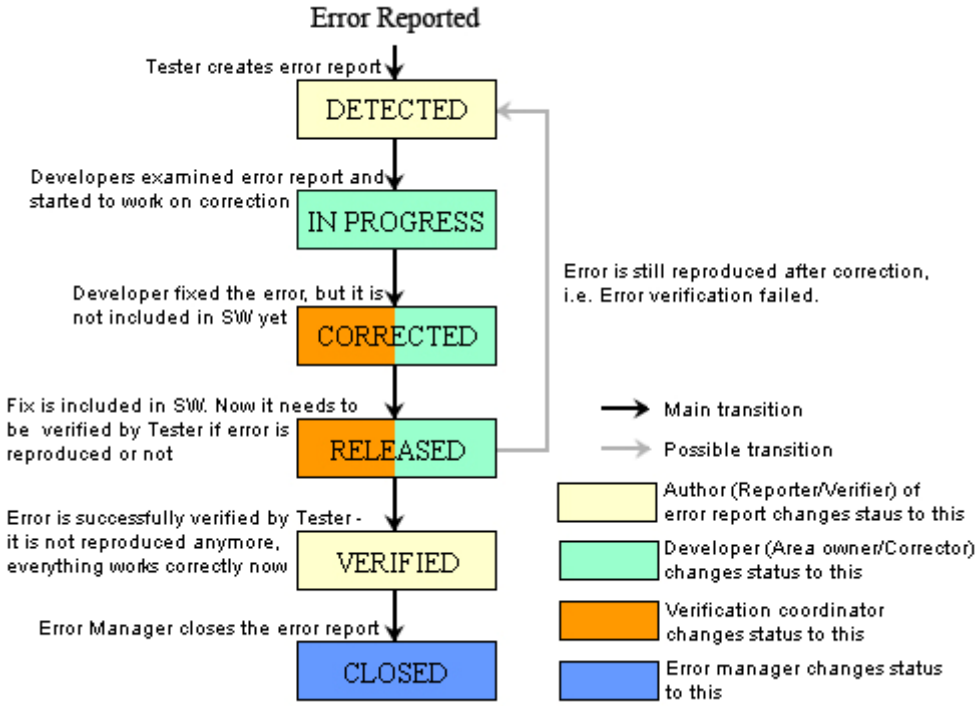
All necessary fields are started with red capital letter, if some mandatory field is not filled, it will be impossible to save report.

When new error report is saved, it appears in TSW with status Detected. The first life stage of error is successfully started.

4.2. Error report maintenance

From the moment when error report is saved until the moment when it is fixed and closed can pass very much time. From experience of author, he created error report that lived about 5 month until it was corrected. There were several tries by developers to fix it, but it was always reproduced. During this time error was changing its status rapidly.

There are some general rules for statuses changing and also scheme was designed to show **the main** (note: main, but not all possible) movements and interactions of error (Picture 6).



Picture 6. Error general life cycle

Picture 6 shows the main (but not all) possible movements in error life cycle. Black bold arrow shows main mandatory transition, i.e. to reach Released status error should walk through Detected, In Progress and Corrected statuses. Also there is a possible transition that is marked by thin gray arrow, i.e. from Released status error must go to Verified status, but it is also possible to go back to Detected status. Each status has its own color, which responses to particular responsibility. For example Reporter/Verifier (tester) can set statuses only to Detected and Verified. He cannot change status of error to Corrected or Released, because it is responsibility of Verification coordinator or Corrector. Error manager is the only person who can close error after it was Corrected and Verified.

There are some responsibilities that correspond to certain roles in TSW.

- Verification coordinator – responsible for setting Corrected and Released statuses, he is dealing with fix task lists, where every fix is indexed and respond to error report. Actually this responsibility is shared with Corrector.
- Error manager – the most informed person, who know everything about current situation with errors in project. He prioritizes errors if needed (changes Priority field), makes sure that error correction speed is in acceptable level. Also he closes errors after successful verification. Additionally in his power to ignore error, set as Duplicate and postpone the error if necessary.

- Reporter/Verifier – tester is acting this role. Usually testers create most of the error reports and also they are responsible for verification of errors.
- Area owner or Corrector – developer. Developers set error status to In Progress and then to Corrected, possibly to Released. Developer's main target is to fix the code according to error reports and add fix task ID.

4.2.1. Main transitions in life cycle: from Detected to Closed

Now author wants to walk through all possible states of error and focus on the most important ones. Some statuses are intuitively clear, but others need some clarification.

***Error reported* – Detected**

When error report is filled with all necessary details and saved, it automatically receives status Detected. Detected status means – The beginning. Detected is alarm for developers – it means error occurred and some actions are necessary to solve the problem as soon as possible (shortly - ASAP). Also Detected state is a sign for tester: if error report was written some time ago and still status is Detected and new SW version came for testing – tester can easily mark this error to test set, because nobody checked it yet and almost 100% that error still occurs in new SW version.
Also Detected status is set after invalid Verification (see Picture 6 with thin gray arrow leading back from Released to Detected), i.e. error was fixed and released, but fix didn't have proper effect and error is still reproduced. In this situation Detected is set again and everything starts from the beginning.
Other case when Detected status is set already during life cycle is after providing additional information to developers when they request it by setting error report in More Info status.

Status meaning in context of QC – TSW interaction:
If error is Detected, then it should be mandatory marked in QC test set with properly failed test case. And comment should be added to error report in TSW that "Error is reproduced with *Name of device_SW version*"

Detected – In Progress

In progress status means simply that developers examined Detected error report and started to work on it. Usually there are many errors in TSW that are In Progress state, because error-fixing process can take much time.

Status meaning in context of QC – TSW interaction:
If error is In Progress status, then it still should be marked by failed case in QC test set (because it is still reproduced), but no comments necessary to error report itself in TSW.

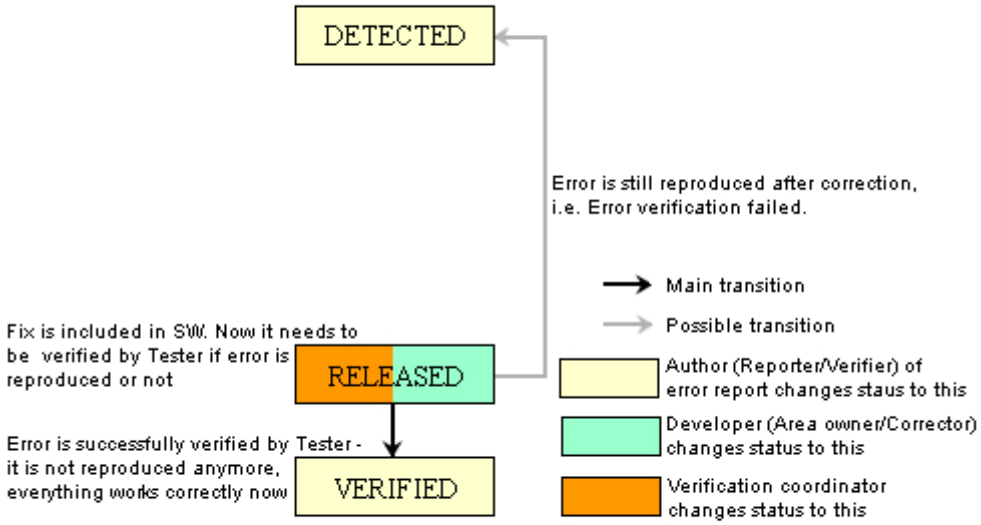
In Progress – Corrected

Corrector sets status to Corrected. It means that error is fixed. Appropriate developers do the error correction. Information about the correction is first updated to the error database.

Status meaning in context of QC – TSW interaction:
Corrected error also should be marked in test set with failed test case. No comments necessary in error report.

Corrected – Released

Released is set when error correction is included in product SW that is released officially. Released is the most temporary status. Error is not stored for long time with this status. When error receives Released status, Error Verification process is started – check Picture 7 for more information.



Picture 7. Error Verification

Picture 7 demonstrates Verification process. Tester should check if error is reproduced or not and depending on result of test, set it back to Detected (fix didn't work and error is still reproduced) or set status to Verified (really fixed and not reproduced any more). If there are errors in Released status – this is sign for tester that he should take some actions. Tester who created error report is set as Responsible Person when status set to Released. It means that creator of error must verify his errors.

Status meaning in context of QC – TSW interaction:
As it was said, Released is temporary status, which goes immediately either to Detected or Verified. And in test set, accordingly test case would be failed (if error goes back to Detected) or test case is passed (if error is fixed and Verified).

Released – Verified

Verification process continues. If error changes status from Released to Verified, then Verification process considered as successful. Tester who created error report is responsible for verification. He marks error report as Verified when he tested error on proper SW version and fix really affected on error so that it is not reproduced any more.

Status meaning in context of QC – TSW interaction:
When error is verified then it means that error is eliminated and test case that was previously failed by it will be passed this time.

Verified – Closed

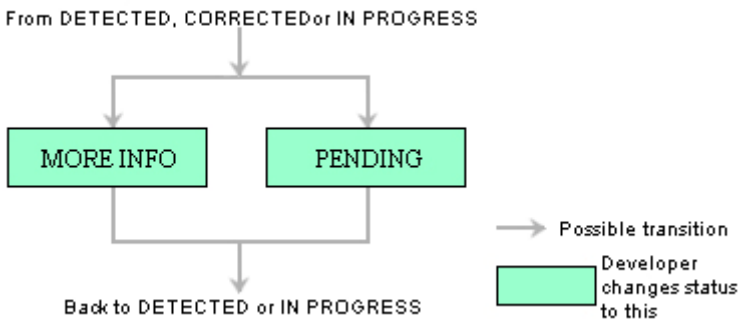
Error manager closes error report after it is fixed and Verified. This is one possible end of error life cycle.

Status meaning in context of QC – TSW interaction:
No error – no failed test case. Test case is passed.

General life cycle of error was presented. Error report is only interesting for a tester if it is accepted and repaired, i.e. it passed through the general life cycle. Each error mandatory passes through those stages, if it was not Ignored or considered as Duplicate in the beginning. But also there are some optional statuses that are used very often. They are optional but can be very important same time. For example, More Info status, which developers usually set to ask some questions or additional information from testers. Such optional statuses should be introduced also in frames of this work.

4.2.2. Possible transitions in life cycle: More Info, Pending

More Info and Pending were excluded from main life cycle, because they are optional and error report can pass the cycle without changing to such statuses. But as practice shows rare error report can avoid of being in these statuses – Picture 8 gives an overview.



Picture 8. Optional statuses: More Info and Pending

Picture 8 shows two possible statuses that error can get after Detected, In progress and Corrected statuses. They are called “floating”, because they can be set in almost any point of error life cycle. Corrector/Area owner usually sets these statuses.

Detected/Corrected/In Progress – More Info

One of the main responsibilities of tester is to check error reports with More Info status. Usually developers use this status when description of error is unclear, error is hard to reproduce and some more details needed how to reproduce the error, e.g. logs or screenshots. Reaction to More Info reports should be as fast as possible from tester’s side. Because all delays with it cause delay of error correction and affect on testing process badly. More Info status should never be left unanswered. If the developer is asking more details about the error, then tester should just provide answers. After it is done, status is set back to “Detected” and responsible person developer who asked this.

Status meaning in context of QC – TSW interaction:
Error goes to Detected state after answer is provided; in QC error is marked with failed test case as usual.

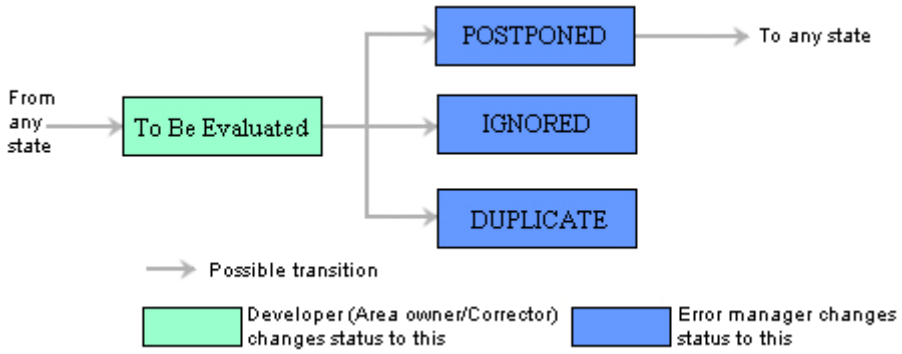
Detected/Corrected/In Progress – Pending

Pending means that correcting or releasing of this error is waiting for something, for example missing localizations. But if error has highest priority (Show Stopper), then everything will be done to fix it and hardly it will have status Pending. Although for Minor errors it is not critical, and usually they can be in Pending status during long period. But in practice this status is rather rare.
After Pending, error report can be set to either Detected or In Progress. Corrector is usually dealing with it.

Status meaning in context of QC – TSW interaction:
Error should not be marked to QC test set when it is in Pending state. Test case should be passed.

3.2.3. Possible transitions in life cycle: To Be Evaluated, Postponed, Ignored, Duplicate

Additional optional but very important interaction is shown on Picture 9. This can affect and change general error life cycle very noticeably.



Picture 9. Optional status: To Be Evaluated leading to Postponed, Ignored or Duplicate statuses

Picture 9 shows that status To Be Evaluated can be used in any possible point of error life cycle by Corrector/Area owner. Three additional statuses are also represented on picture: Postponed, Ignored and Duplicate. Ignored and Duplicate statuses are end-states (same as Closed), but Postponed still can lead to any state further. This is Error manager responsibility to set these statuses.

From any status **– To Be Evaluated**

The name of this status explains everything by itself. To Be Evaluated is set when it is unclear what to do with error from first sight. Usually it takes some time for developers or test managers to decide what actions should be performed with error, for example, should it be ignored or it is major enough and needs to be fixed.

Status meaning in context of QC – TSW interaction:
Test case should be failed with such error (until there is no decision made it is considered that error is active) and no comments necessary to error report.

Ignored, Duplicate

Author decided to group these two statuses under one section. Reason is that they are alternative for common final status of error – for Closed status.

Ignored and Duplicate states are finishing the life cycle of error the same way as Closed status. But in case of Closed state, error was handled, fixed, verified and released. On the other hand, Ignored and Duplicate finish life cycle of error without fixing it. Usually, tester will not be happy to see his carefully created error report as Ignored or Duplicate, because it means he made useless job.

Ignored state forces to forget about the error, because it is actually not an error from developer's point of view. For example, error is ignored when tester considers correct behavior as false (different reasons for it: usually because of lack of information about particular behavior in specifications and requirements, or just such view of tester) and creates faulty error report.
The second reason when some error reports are ignored is that error can be too insignificant or hard to repair.

Duplicate means simply, that somebody before, created similar error report and false behavior is already known. Actually, this is one of straight responsibilities of tester to check that error is not already in TSW before creating error report.

Statuses meaning in context of QC – TSW interaction:
If error is ignored, test case logically should be marked as passed (no error – no failed test case).
If error is Duplicate – then there should be original error, and test case should be handled according to status of original error.

Postponed

Postponed means that error will be handled later and it does not mean the end of errors life cycle (but it can lead to Ignored). Usually Postponed is set when deadline for releasing product to market is close and management needs to prioritize what errors should be fixed as soon as possible and what errors are not much important. Not important errors go to

Postponed state. But when time comes such errors will be active again and error can go to any possible state. Error manager is dealing with it.

Statuses meaning in context of QC – TSW interaction:
In test set Postponed error is ignored at the moment and no need to fail any case with such error.

As a last word before conclusion of this Chapter author would like to describe shortly the responsibilities of tester when dealing with errors in TSW. In frames of this chapter it will give clear view what role tester plays in error life cycle.

4.3. Tester's responsibilities in error life cycle

There are several responsibilities of tester when working with TSW. Tester must:

- Create error reports about wrong behavior of SW
- Verify that the error is not a duplicate before reporting the error
- Check errors related to own project on daily basis and follow the statuses
- Pay additional attention to More Info statuses:
  1. Reproduce error when needed if developer cannot reproduce and give clear instruction how to do it.
  2. Answer the questions that developer asks.
- Verify the error correction (Status of error is set to Released)
- Follow the reasons why errors are ignored and learn from them

On this moment Chapter is finished. Error management tools are used to perform two activities: error report creation and error report maintenance. It was briefly looked through the first activity and introduced and analyzed the second one.
All possible statuses and their interactions were mentioned. Also was shown each statuses meaning in context of QC – TSW interaction, that will be very useful information for tester. There are several different roles for people who are involved in error correction process, e.g. Corrector, Error Manager etc. But in the end of the chapter author stopped only on testers responsibilities in error maintenance. It made final accord to the chapter and also underlined the role of tester in this rather complicated (from first sight) process.

Conclusion

Purpose of the work was to analyze needs and problems of new employees in Ixonos Plc. And using this analysis and conclusions made in it as a basis, start designing the learning material for Company internal use. Author tried to provide answers to most frequently asked questions related to error maintenance in most readable way. Selecting the error-related activities as the core of this thesis was not accident. There are at least two reasons for it:

- Object of testing is finding errors; tester must know his “enemy” face and characteristics.
- The way, in which error report handled is Company-specific and it is highly important that new tester will refer to specific internal materials, not to general found in Internet.

Author tried to provide material logically, step-by-step from theory to practice, from basics to complex.

The main difficulty that was faced when creating this material was to provide clear and logical structure. First of all testing theory was given, then testing tools and interactions between them were introduced. Finally, error life cycle and its influence on testing process were shown. This is the exact template of how new tester starts to study the working process. New tester faces the problems in exactly this order – tried by author and many people who started to work as testers.

During review of the material when it was already written, author found some points to be improved – ideas for future implementation.
For example, additional information about test case design (good test case title, description, amount of steps is test case, requirement coverage etc) would be necessary for new tester. How test case is derived from specification, why it is necessary – common questions about test design.
Next point is to provide more information about testing types. They can be different, from Functional Testing (FuTe) until smoke testing. Each testing has own principles and targets. Even different types of errors can be discovered during different types of testing. But this is general topic and lots of material can be found in Internet also.
Also it could be useful for new tester to have the standard plan of how separate testing project is done: from e-mail letter with new testing task until test results report creation.

All these points will be written and added to learning material. Also some places will be modified according to confidential policies of the company. Then several specialists will check it and after proper corrections, material can be allowed for use internally.
Currently author got feedback-impression about this material:

"It is obvious that testing is vital in software life cycle. To find mistakes and remove defects in code might save someone's life not to mention the potential material losses. But how we can be sure that mistakes are not made while searching mistakes? To minimize such scenarios different standards and processes are developed that companies are following: e.g. BS 7925-1 (Glossary of Software Testing Terms), BS 7925-2 (Software Component Testing Standard) or IEEE829 (Standard for Software Test Documentation). Thus it is very important to understand the basic of SW lifecycle, learn to handle different tools and follow the processes implemented before actual testing can be started. Testing is not just pressing the buttons and marking the cases passed or failed but interpreting the causes why something happened or not."(Mirko Spenk, Line Manager, works in Ixonos Plc since 2006).

Kokkuvõte

Mobiiltelefonide turg kasvab jõudsalt – uued mobiiltelefonide mudelid ilmuvad tihti. Mobiiltelefonide tootajate konkurents on väga tihe. Üks oluline faktor sellises olukorras on mobiiltelefonide tarkvara kvaliteet. Protsess, mis vastutab kvaliteedi eest, on testimine.

Bakalaureusetöö autor töötab firmas, mis tegeleb tarkvara testimisega Nokia nutitelefonide jaoks – firma nimi on Ixonos Plc. Firmas töötab umbes 70 inimest. Paljud inimesed alustavad oma testija karjääri eelneva kogemuseta selles vallas. Sellepärast on firmal vaja õppematerjale, mida võiksid kasutada uued testijad töö algusel.

Töö eesmärgiks on esiteks analüüsida, mida uuele testijale vaja on – millised probleemid ja küsimused tavaliselt tekivad. Kui see on saavutatud, on vaja luua õppematerjal, mis katab kõik need küsimused. Põhiline tähelepanu materjalis on vea elutsüklil testimisprotsessis, sest kõige olulisem testimise eesmärk on vigade leidmine.

Esimeses peatükis on kirjeldatakse probleeme, mis võivad tekkida kogemusteta testijal. Selles peatükis on kokkuvõte töö sisukorrast ja struktuurist. Töö autor alustas samuti testimise vallas eelneva kogemuseta ja ta on ise kõik need probleemid läbi elanud.

Teine peatükk räägib testimise eesmärkidest ja tutvustab testimise printsiipe. Praktika ei ole võimalik ilma vastava teooriata ja see peatükk sisaldab teoreetilist informatsiooni. Järgmine peatükk annab ülevaate sellest, kuidas toimub tööprotsess firmas. Igapäevases töös kasutab testija kahte tööriista - Quality Center, kus testjuhtumeid käivitatakse. Quality Center on tööriist, mis toetab testimist. Kui testija hakkab kasutama QC, siis ta puutub kokku selliste terminitega, nagu “test case”, “test set” ja “error”.

Pärast vea leidmist peab testija tutvuma vigade andmebaasiga, mille nimi on TSW. TSW-s kirjutab testija vea aruande. Viimane ja kõige olulisem peatükk on sellest, kuidas toimub vea elutsükkel.

Materjal ei ole veel täiesti valmis. Autoril on mõned ideed kuidas seda täiendada. Näiteks võiks lisada informatisooni testide koostamise ja testimistehnikate kohta. Seda materjali ei saa pidada lõplikuks ja kindlasti on veel erinevaid mooduseid selle täiustamiseks.

Literature

Brian Hambling, Peter Morgan, Angelina Samaroo, Geoff Thompson, Peter Williams. 2007. Software Testing: An ISEB Foundation. Swindon: The British Computer Society.

Thomas Müller, Dorothy Graham, Debra Friedenberg, Erik van Veendendal. 2008. Certified Tester Foundation Level Syllabus. URL= http://www.istqb.org/downloads/syllabi/SyllabusFoundation.pdf (viimati vaadatud 12.04.2009).

Erik van Veenendaal. 2007. Standard glossary of terms used in Software Testing. URL= http://www.istqb.org/downloads/glossary-current.pdf (viimati vaadatud 12.04.2009).

Norbert Ruck, Ralf Lulei, Thorsten Geiselhart, Gerd Rockweiler. 2008. Requirements and test management: More safety in vehicles through traceable testing. *Testing Experience. The magazine for professional testers*, 2, 38-45.

Hans Schaefer. 2008. What a Tester Should Know, At Any Time, Even After Midnight. *Testing Experience. The magazine for professional testers*, 1, 40-46.

Abbreviations

ISTQB – International Software Testing Qualifications Board

ISEB – Information Systems Examination Board

N/A – Not Available

FTP – Fundamental Test Process

SW – Software

QC – Quality Center

TSW – name of the database, not abbreviation

FuTe – Functional Testing

BAT – Basic Acceptance Testing

ASAP – As Soon As Possible

OS – Operating System