TAMPERE
POLYTECHNIC

FINAL THESIS

IMPROVING TEST CASE REVIEW TECHNIQUE IN MESSAGING TEST TEAM

Päivi Mäntylä

Business Information Systems
October 2006
Supervisor: Paula Hietala

TAMPERE 2006

TAMPERE
POLYTECHNIC

Author(s)	Päivi Mäntylä
Degree Programme(s)	Business Information Systems
Title	Improving test case review technique in Messaging test team
Month and year	October 2006
Supervisor	Paula Hietala

Pages: 37

ABSTRACT

Reviews are an important part of the software development process. The main goal is to remove defects from the product as early as possible and this way improve the quality. Reviews can be carried out at all phases of the software development, and any kind of product or document can be reviewed.

Software testing produces different kinds of documents, and test case specifications are one of them. The purpose of this thesis was to find a way to improve the test case review technique in the Messaging test team. The assignment was given by Nokia's Messaging product development team. The aim was also to create a review template for the review meetings and a checklist that helps the reviewers to find more defects from the reviewed document.

The process started by studying the literature and observing the present situation in the Messaging test team. In the theoretical part the basics about reviews are explained. The different kinds of review techniques are presented and after that the focus is on issues that should be noticed in every review. Based on the theory and observation, the solution proposal for improving the test case review technique in the Messaging test team, is generated.

The intention is to take these improvement suggestions into use in the Messaging test team and also follow what kind of influence the changes will bring to the reviews. The review process should also be observed and enhanced in the future.

Keywords Review Quality Software testing

TAMPEREEN
AMMATTIKORKEAKOULU

Tekijä(t) Päivi Mäntylä

Koulutusohjelma(t) Tietojenkäsittely

Opinnäytetyön nimi Testitapausten katselmointikäytännön tehostaminen Messaging-testaustiimissä

Työn valmistumis-kuukausi ja -vuosi Lokakuu 2006

Työn ohjaaja Paula Hietala **Sivumäärä:** 37

TIIVISTELMÄ

Katselmoinnit ovat tärkeä osa ohjelmistokehitysprosessia. Päätarkoitus on poistaa virheet tuotteesta mahdollisimman varhaisessa vaiheessa ja siten parantaa tuotteen laatua. Katselmointeja voidaan suorittaa ohjelmistokehityksen jokaisessa vaiheessa, ja mikä tahansa tuote tai dokumentti voidaan katselmoida.

Ohjelmistotestauksen aikana laaditaan erilaisia dokumentteja, joista yksi on testitapausten määrittelydokumentti. Tämän opinnäytetyön tavoitteena oli kehittää testitapausten määrittelydokumenttien katselmointitekniikkaa Messaging-testaustiimissä. Työ tehtiin toimeksiantona Nokian Messaging-tuotekehitystiimille. Työn tavoitteena oli myös tehdä pöytäkirjamalli katselmointipalavereihin sekä suunnitella tarkastuslista, joka auttaa katselmoijia löytämään entistä enemmän virheitä tarkasteltavasta dokumentista.

Työ aloitettiin tutustumalla kirjallisuuteen ja tarkkailemalla Messaging-testaustiimin tämänhetkistä katselmointikäytäntöä. Opinnäytetyön teoriaosuudessa selvitetään perusasiat katselmoinneista: esitellään erilaiset katselmointitavat ja sen jälkeen keskitytään asioihin, jotka tulisi ottaa huomioon kaikissa katselmoinneissa. Työn lopussa esitetään ratkaisuehdotus katselmointikäytännön parantamiseksi. Tämä ratkaisuehdotus pohjautuu sekä teoriaan että katselmointikäytännön tarkkailuun.

Työssä ehdotetut katselmointikäytännön parannukset on tarkoitus ottaa käyttöön Messaging-testaustiimissä. Tarkoitus on myös seurata, minkälaisia vaikutuksia näillä muutoksilla on, sekä edelleen jatkaa katselmointikäytännön tarkkailua ja mahdollista kehittämistä.

Avainsanat Katselmointi Laatu Ohjelmistotestaus

Table of contents

1 Introduction

Software quality has a huge impact on a product's success. This is a well-known fact in business life, and also Nokia, the world leader in mobile communications, emphasizes the importance of it. To ensure the quality, different kinds of quality assurance activities are taken. One of these activities is software reviews, which are covered in this thesis.

According to Karl Wiegers (2002) there is nothing wrong in making mistakes. He thinks it is the part that makes us human. But he also continues that it is important to catch those errors early, before they become difficult to find and expensive to correct. Software reviews are one of the most effective ways of reducing problems and improving quality in early phases.

Wiegers (2002) also reminds that finding your own mistakes is often hard, because you are too close to your work. To catch the mistakes, you need a fresh perspective and brains that think in a different way. That is also one reason why software reviews with qualified colleagues are important.

There are many different ways of conducting reviews. Review techniques start from disciplined inspections and end up in very informal ad hoc reviews. When people need other peoples' opinions or help, a very informal review will meet the needs. But when the work involves many persons and their approval is required, a more systematic technique is needed.

In this thesis, I will introduce these techniques shortly and try find out the suitable technique for Messaging test team reviews. I will study and analyze the present problems the Messaging test team has and try to find a way to improve the review technique.

1.1 Background

This thesis assignment was given by the Messaging product development team. The assignment originated from the need to improve and obtain a consistent way for reviewing test cases.

At the moment the Messaging team has 12 test engineers and I am one of them. Every test engineer organizes test case reviews quite regularly, but a systematic review technique is missing. Due to that the same mistakes appear in test case specifications time after time.

Test case reviews are a very important tool for catching defects from test case specifications. The specifications have to be of high quality, because besides our team, they are delivered to other teams and customers, too. We can also make proofreaders' work more easier, if we have clear and well-reviewed specifications.

1.2 Objectives

The purpose of this thesis is to define a suitable technique for reviewing test cases in the Messaging test team. I point out the problems we have now and try to find a way to avoid them. The purpose is also to create a checklist that helps focusing on the most important issues while reviewing. In addition to these, I define a review template that is used in the review meetings for collecting data and metrics.

1.3 Structure of the thesis

In this thesis, I first shortly describe how testing is related to the software development process. In chapter 3, theoretical concepts of reviews are introduced. That includes listing advantages and disadvantages of reviews and describing different review techniques. I also point out what kinds of input documents are needed and what kinds of data should be collected from the reviews. Also general guidelines are summarized in the end of chapter 3.

After the theoretical part, the current situation and problems are presented in chapter 4. Also a solution proposal for improving the test case review technique is presented in the same chapter. In chapter 5, the summarization of the work and future improvement suggestions are introduced.

2 Testing as a part of the software development process

Testing is a very important part of the software development process. The goal is to improve quality and verify that software works according to the specifications.

Testing consists of many phases and it is a parallel activity to other software development tasks. Usually the relation between development phases and testing levels is illustrated using the V-model (Figure 1). (Haikala & Märijärvi 2004.)

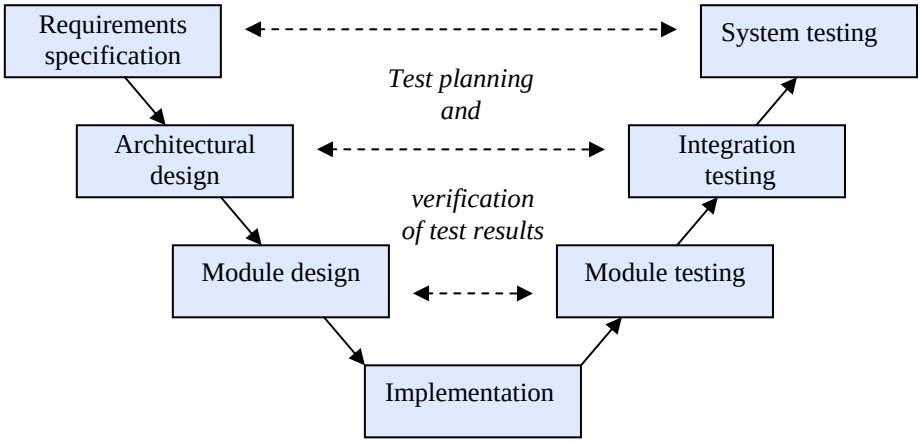


Figure 1. V-model (Haikala & Märijärvi 2004:289)

Test planning for each testing level is done in the corresponding development phase. Also test cases are created based on the documents produced at each development phase. After the test cases have been executed, the results are compared to those documents. (Tamres 2002.)

2.1 Testing levels

According to the V-model, testing can be divided to module testing, integration testing and system testing.

Module testing is the first testing phase and it focuses on testing the smallest component that can be compiled. Individual modules are tested in isolation by running tests in an artificial environment, which requires the use of test beds with drivers and stubs. Usually the module testing is done by the developer him/herself. (Haikala & Märijärvi 2004.)

Integration testing verifies that the combined modules function correctly together. Main focus is on checking the interfaces between the modules. Integration testing is usually done partly parallel with module testing and due to that the tests are mainly executed by the developer. (Haikala & Märijärvi 2004.)

System testing verifies the entire product. In this phase, different kinds of things are tested: functionality, reliability, compatibility, performance, security, etc. Due to these different aspects, this testing phase can sometimes be divided into different subphases. The environment in this phase should be as close to the real environment as possible. Also testing should be done by separate testing group (Tamres 2002.) In this thesis things are observed from the system testing and especially from the functional testing point of view.

2.2 Reviews and testing

In every development phase different kinds of specification documents are generated. These documents act as input documents for the next development phase and also for the testing, like a requirement specification acts as a basis for the system testing. To avoid problems in a lower level, each of these input documents should be carefully reviewed before they are used.

Testing activities produce different documents like test plans and test case specifications. Other level testing documents can also be used when planning testing activities for a specific testing level, because checking what kinds of testing has been done in the other levels might help to increase the coverage of the testing.

The testing documents are as prone to defects as any other documents, so also they need to be reviewed before they are used. When, for example, test cases are reviewed carefully, a lot of time is saved when executing actual tests. (Gilb & Graham 1993.) Figure 2 illustrates the relationship between reviews and testing activities.

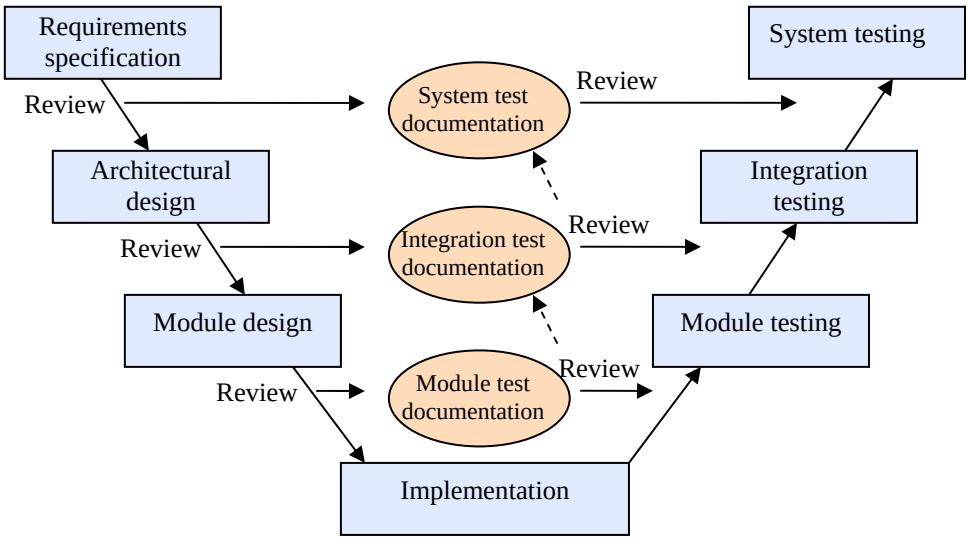


Figure 2. Reviews in software development process (modified from Wiegers 2002:10)

3 Review

Reviews are known as an effective and commonly used quality assurance activity. They are an essential part of different phases in the software development process, and the main goal is to detect and correct problems in early stages. There are no rules on what can or cannot be reviewed. ANSI/IEEE Standard for Software Reviews (1028-1997) lists up to 37 software-related works that are candidates for review. This list includes, for example, requirement specifications, design descriptions, test and user documentation and source code.

Software review was developed by Michael Fagan in 1972-1974. The method was initially used for inspecting computer source code, but it quickly spread to all aspects of software engineering. Fagan called his method ‘Inspection’ and the literature still uses the term ‘Fagan's inspection’ broadly. (Gilb & Graham 1993.)

3.1 Advantages and disadvantages

As mentioned earlier, the main purpose of reviews is to identify and remove defects in a product as early as possible. This is the main idea in reviews, because when defects are found and fixed, the quality of the product increases. Furthermore, when defects are found early, they are easier and less expensive to fix. So eventually well conducted reviews will save time and money.

Besides these advantages, reviews provide other benefits, too. In reviews participants share information and most likely learn something new from others. Freedman & Weinberg (1982) estimate that a programmer, who regularly participates in different reviews, gains experience three times quicker than by working alone.

One advantage is also that reviews can be conducted before the actual work product is even ready. This makes it easier to do some changes and improvements to the product. After the review, participants also share a common understanding of the work product. In addition, the given feedback and improvement suggestions help the author to create a better work product in the future. (Wiegers 2002.)

As usual, the coin has two sides. In this case it means that reviews also have some disadvantages. One of them is that participants often perform overlapping job by finding the same defects twice. This can be avoided by defining specific roles to the participants or by distributing the given comments in real time.

Another downside in reviews is that some people may become lax in their work, because they are relying on someone else to find their mistakes. On the other hand, it is not easy to ask other people to point out errors in your own

work and due to that some people may have a temptation to perfect the product before they allow other people to see it. In these situations reviews are not so valuable and efficient anymore. The author may become resistant to suggestions for changes, because (s)he has already done so much work for the product. (Wiegers 2002.)

Usually the situation is still something in the middle of these two extremes. People do their work quite thoroughly, because they know it will be publicly reviewed. This leads to a better quality already in the beginning and therefore it can be seen as an advantage of the review. (Haikala & Märijärvi 2004.)

3.2 Review techniques

There are different kinds of review techniques which can be classified based on their degree of formality (Figure 3). Almost all review techniques can be seen as derivatives of 'Fagan's inspection' (Tian 2005). Terms for these different techniques are often used as synonyms and also software literature contains conflicting definitions and inconsistent usage for the terms. In this work, I use term 'review' as a general term for all techniques.

Most commonly used techniques are inspection, technical review and walkthrough. Sometimes the term 'technical review' is seen as a general term for all the reviews and in those situations terms like 'team review' or just 'review' are used for a certain kind of technique. Besides these most commonly used techniques, there are plenty of other variations, like buddy check/peer desk check, pass around and ad hoc review. In this chapter, I will describe the main features of these different techniques, mainly focusing on inspection, team review and walkthrough.

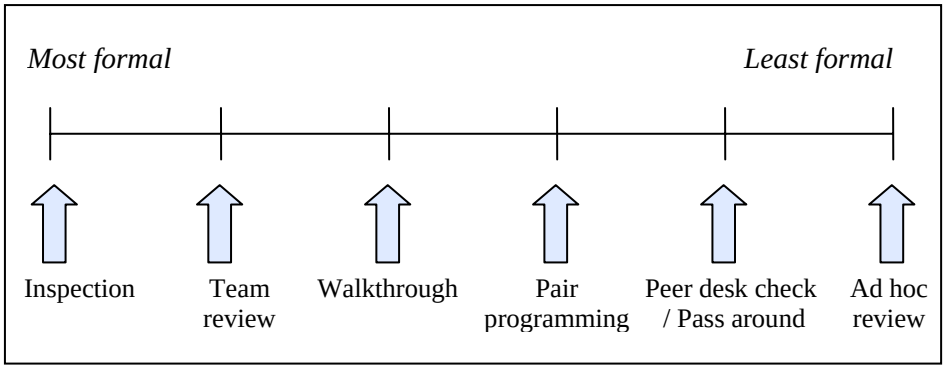


Figure 3. Formality spectrum of reviews (Wiegers 2002: 32)

3.2.1 Inspection

Inspection is the most systematic and most formal review technique. It is also identified as the most effective technique to find defects. During the review

process, specific steps are followed under the control of a trained Inspection Leader (Gilb & Graham 1993).

As mentioned earlier, it was Michael Fagan who first developed the inspection technique in the 1970's. His inspection process included six steps: planning, overview, preparation, inspection, rework and follow-up (Tian 2005). Variations of Fagan's inspection have been developed afterwards, but differences to the original inspection technique are quite minor. The most famous and most used variation is probably the technique developed by Tom Gilb and Dorothy Graham, which I also describe in this chapter. Their inspection process has similar steps as Fagan's, but they are labeled differently (Figure 4). Gilb and Graham also added one step called "process brainstorming meeting" right after the inspection meeting. The purpose of this step is to improve the whole inspection process continuously.

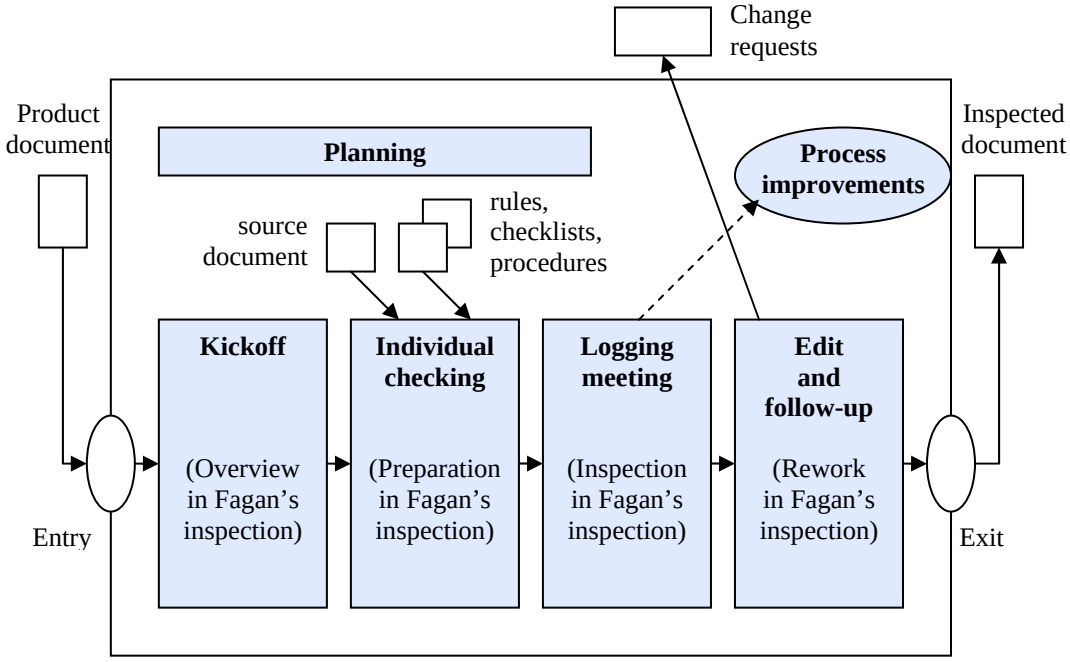


Figure 4. Inspection steps (Gilb & Graham 1993: 34)

Before the actual inspection process can even start, a request for inspection is needed. This means that the author of a document wants to get his or her work inspected and (s)he asks the Inspection Leader to organize the inspection. The Inspection Leader is a trained and certified person, who will control the whole process. Sometimes there is only one trained Inspection Leader in the organization or then there is a certain sphere of responsibilities among the leaders. (Gilb & Graham 1993.)

Planning

The Inspection process starts with planning. The first activity is to check that the entry criteria for the inspection have been met. The purpose of this is to reduce the possibility that the inspection group will waste time by inspecting a document that is unfinished. If the Inspection Leader finds for example a large number of defects in a document only by looking at it briefly or (s)he notices

some other faults, the entry fails and the author of the document has to correct his or her work. (Gilb & Graham 1993.)

After the document passes its entry criteria, a more detailed planning process starts. The planning process includes many activities, which are displayed in Figure 5. Based on these activities, a master plan that guides the execution of the inspection process is created. (Gilb & Graham 1993.)

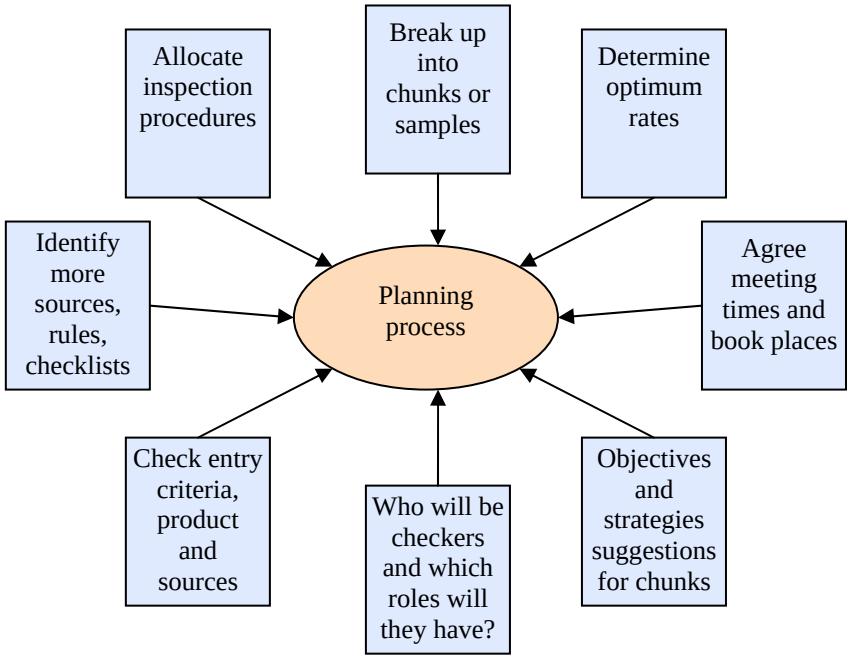


Figure 5. Components of the planning process (Gilb & Graham 1993: 44)

Kick-off meeting

A kick-off meeting is a voluntary step in the inspection process, but it is often seen as a helpful tool for training and motivating the inspection group. The purpose is to share information to everyone at the same time and clarify the individual checking tasks. It may include activities like distributing documents, assigning specific roles to the participants and giving instructions on how to do the inspection work. The kick-off meeting also gives the possibility to ask any questions about the document that is being inspected. (Gilb & Graham 1993.)

Individual checking

In the individual checking step inspectors find their own time and place, and check the document through alone using the source documents, rules, procedures and checklists provided. The main purpose is to find defects or other faults from the document. Usually the Inspection Leader assigns specific roles to individual inspectors, so that they can focus on identifying particular

type of defects (Figure 6). This way the possibility of finding many unique issues is maximized. (Gilb & Graham 1993.)

User:	Concentrate on the user or customer point of view
Tester:	Concentrate on test considerations (testability, test requirements, etc.)
System:	Concentrate on wider system implications
Quality:	Concentrate on all aspects of quality attributes
Rules:	Pay special attention to rules for this product

Figure 6. Examples of roles (Gilb & Graham 1993:77)

Individual inspectors mark all the defects they find in the document and also categorize them to major or minor. In case they encounter any difficulties during the checking, they contact the Inspection Leader right away. The inspectors should work within the recommended optimum working rate, which is calculated from the inspection experience metrics. For keeping these metrics up to date, the inspectors also have to record the time they spend doing the checking. (Gilb & Graham 1993.)

There are no specific rules on how to go through the document, so the inspectors can choose the working style that suits them best. The main issue is that the checking work is completed in the given time and it is done properly. (Gilb & Graham 1993.)

Logging meeting

The logging meeting is a place, where items found in the individual checking are reported and logged. There are three types of items which will be recorded: potential defects, questions of intent to the author and process improvement suggestions. The recording is done by a person called scribe, who usually does not belong to the inspection group. (Gilb & Graham 1993.)

The logging meeting is strictly controlled by the Inspection Leader or the moderator that the leader has appointed. The meeting starts by collecting checking data, like time spent during the checking and the amount of found defects, from each checker. If someone of the checkers has not prepared properly, (s)he will be sent off from the meeting. If there are more than one checker that has not completed the individual checking, the logging meeting is postponed. (Gilb & Graham 1993.)

The main activity in the meeting is to identify the found items aloud to the scribe. The checkers should express themselves as clearly and concisely as possible. Every item should only be brought up once to avoid duplicate issue logs. This means that if one person reports an item that some other checkers have found too, the others should remain silent. (Gilb & Graham 1993.)

Besides reporting and logging earlier found items, the checkers will continue a quiet checking activity during the logging meeting. The purpose is to find even more items than those found during the individual checking. Due to this checking activity, the checking rate should be slow enough to allow new items to be found. But again, these meetings also have a calculable optimum rate, which limits extra slowness. (Gilb & Graham 1993.)

To keep the meetings effective, no further discussion about the items is allowed. Any debate, criticism, explanations, etc. are consciously excluded. Because of human tiredness, the meeting cannot last more than two hours. This means that a large document has to be divided into two or more chunks. (Gilb & Graham 1993.)

Process brainstorming meeting

The Process brainstorming meeting is an optional activity that will take place right after the logging meeting. It will last for at most 30 minutes and it will be held with the same persons that attended the logging meeting. Sometimes also additional interested parties can be invited. (Gilb & Graham 1993.)

The purpose of this meeting is to analyze root causes for the found defects and generate ideas on how to improve the software development process or the inspection process. Usually there is not enough time to go over all the found defects, so few most important defects have to be selected. Each defect is presented quickly to remind all the attendees where the defect was found and what it is all about. After that people can freely come up with ideas for the root causes. (Gilb & Graham 1993.)

After the root causes have been analyzed, process improvement ideas that could prevent the defect from occurring again, are brainstormed. For each defect, at least one improvement suggestion that can be easily carried out by one or more attendees, should be generated. Besides those easily accomplished improvement ideas, there can also be some bigger proposals, which require actions for example from the management side. A simple example of the root cause analysis is presented in Figure 7. (Gilb & Graham 1993.)

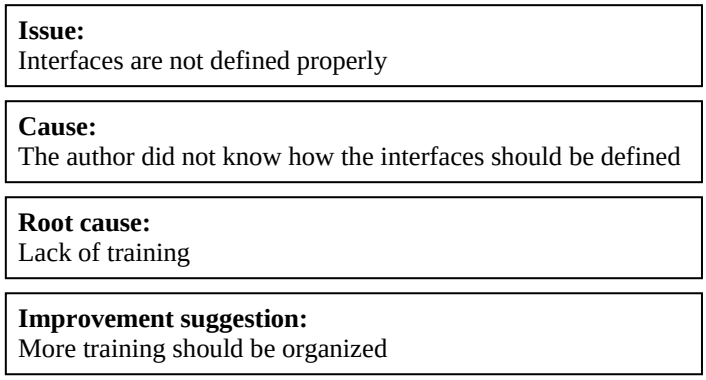


Figure 7. Example of root cause analysis

All the root cause issues and improvement ideas are logged. As in the logging meeting, criticism and evaluation of an idea are not allowed. The logged items are saved to the quality assurance database, where they can be monitored and followed-up. (Gilb & Graham 1993.)

Edit

The next step after the logging meeting – and possible process brainstorming meeting – is to start the edit process. This is usually carried out by the author of the inspected document. (S)he will go through the list of found issues and make a correct action for each item logged. This can mean making corrections to the product, sending change requests to other people's documents or making some further improvement suggestion. (Gilb & Graham 1993.)

The edit process begins by giving the final classification to the issues found. The author decides if the issues are genuine defects or improvement suggestions. (S)he also determines the severity and action for each issue. After editing the product or analyzing the found issue more, the author is also authorized to reclassify the issues, if necessary. (Gilb & Graham 1993.)

Follow-up

After the author has completed the editing task, (s)he gives all the edited data to the Inspection Leader. The Inspection Leader then checks that some action has been taken on every item logged. This checking phase is called follow-up. (Gilb & Graham 1993.)

It is not reasonable that the Inspection Leader examines all the actual edit actions, but (s)he will check that all the listed issues are acted on in writing and that the improvement suggestions are sent to the appropriate process owners. Reporting final inspection metrics for defects by severity and hours used, is also a part of the follow-up process. (Gilb & Graham 1993.)

After the Inspection Leader has checked the listed items through, (s)he will decide if the exit criteria for the inspected product has been met. The exit may fail, for example, if some listed items are not acted on or the estimated number of remaining defects is too high. The official exit criteria should be set by the organization. (Gilb & Graham 1993.)

If the product fails to exit, the Inspection Leader decides what is the best way to continue. This can mean, for example, repeating the inspection process after the edit phase or asking the author to do a massive cleanup or re-write for the document. Otherwise, if the product exits successfully, it will be a releasable, fully approved product. (Gilb & Graham 1993.)

3.2.2 Team review

Team reviews are also planned and structured review techniques, but less formal and less rigorous than inspections. In team reviews, there is also a group of qualified persons that will check the document and identify possible defects or other issues (Wiegers 2002). Many steps are similar as in

inspections, but for example the kick-off meeting and the follow-up phase are missing.

The team reviews are organized by the author of the document. When (s)he decides that the document is ready for review, (s)he will make a meeting reservation and send the material for the reviewers. This should happen several days prior to the meeting, so that the participants have enough time to study the material on their own. (Wiegers 2002.)

There is no separate leader in the team reviews, but a moderator is still needed for keeping the meetings effective and on course. Unlike in inspections, the moderator is usually the author of the reviewed document. The meetings are proceed so that the moderator asks the participants if they have any issues on a specific section or page. The scribe writes down the issues that arise, using the standard forms that the organization has adopted. (Wiegers 2002.)

Although there is no specific follow-up step in the team reviews, it is assumed that the author checks through all the found issues and makes the needed corrections. If a lot of issues were found during the review meeting, the reviewers can also suggest that a re-review will be conducted after the author has edited the document. (Wiegers 2002.)

3.2.3 Walkthrough

Walkthroughs are classified as an informal review technique, because they usually do not follow any specific procedure. Different people can hold different kinds of walkthroughs, ranging from casual to disciplined. As in other reviews, one of the purposes is to find defects and other problems, but another goal is also to achieve a shared understanding and agreement about the functionality of the presented product. Walkthroughs can also be seen as a good training tool for people to learn more about the specific product. (Wiegers 2002.)

In walkthroughs, there is usually a larger number of participants compared to inspections or team reviews. In a typical walkthrough, the author leads the other participants through the document or code (s)he has written. The material should have been sent to the participants in advance, so that they can examine it before the meeting. In the meeting the author explains the purpose of each module, how it is structured and how it performs its tasks. Other participants listen and give feedback and suggestions. (Patton 2001.)

Usually no specific record about the walkthrough is kept (Wiegers 2002). It is important though, that the author records the given suggestions, so that (s)he can make improvements to the product, if necessary.

3.2.4 Other techniques

Other techniques are classified as informal review techniques and they do not usually involve so many persons as the inspections, team reviews or walkthroughs. The advantage of these techniques is that they are quick and cheap, and do not require planning in advance (Wiegers 2002).

Peer desk check (or buddy check) is a review technique, where only one person examines the document besides the author. This is one of the cheapest review techniques, because it only takes one person's time. The downside is that the effectiveness of the review depends entirely on a single reviewer's knowledge, skills and self-discipline. (Wiegers 2002.)

The Pass around technique is similar to the peer desk check, but the difference is that the author gives a copy of the document to several people instead of just one person. This usually means that the document is reviewed more carefully. The reviewers are also able to see the comments that the others have already written, which reduces redundancy. (Wiegers 2002.)

Ad Hoc reviews are the most informal type of reviews. This review takes place when one person asks another to spend a few minutes helping to track a difficult problem in their work. This may be an ordinary occasion in everyday work and usually people do not even think that they are doing a review. (Wiegers 2002.)

In the following table (Table 1) activities that are typically included in different types of reviews are illustrated.

Table 1. Typical activities in different types of reviews (Wiegers 2002:33)

Review type	Planning	Preparation	Meeting	Correction	Verification
Inspection	Yes	Yes	Yes	Yes	Yes
Team Review	Yes	Yes	Yes	Yes	No
Walkthrough	Yes	No	Yes	Yes	No
Peer Desk check /Pass around	No	Yes	Possibly	Yes	No
Ad Hoc Review	No	No	Yes	Yes	No

3.3 Input documents for reviews

Although the main focus in the reviews is on the reviewed document, also other documents are needed. Gilb & Graham (1993) list four types of input documents that inspectors should have:

- Source document(s)
- Rules
- Checklists
- Procedures

Source documents

The product under the review is generated based on source documents like contracts, requirements, plans, etc. It is important that all the inspectors have access to those documents and that they check the reviewed product carefully against them. (Gilb & Graham 1993.)

As mentioned in chapter 2.2, also source documents should be carefully reviewed before they can be used as sources. Despite the review, it is always possible that the source documents have some defects, too. Because of that the inspectors should read the source documents critically and report any potential defects found in them. (Gilb & Graham 1993.)

Rules

Rules are statements that define how the document should be constructed. They might define, for example, conventions for naming objects, formatting source code or organizing a document (Wiegers 2002). Because they specify what is required in a written document, they are the key in finding defects (Gilb & Graham 1993).

Rules also help to increase the objectivity in reviews (Gilb & Graham 1993). It is easier to point out potential defects by referring to some specific rule number than by just saying “I don’t think this is a good way to do it”. This way the focus stays on the document and not on the person. (Wiegers 2002.)

Rules should be updated and improved as often as needed. The inspectors can give improvement suggestions at any time during the review process, as well as outside of it, but the changes should be formally approved. The rules have a specific owner, who will do the updating. (Gilb & Graham 1993.)

Checklists

Checklists are an essential part of the review process. Gilb & Graham (1993) define the checklist as a specialized set of questions designed to help the inspectors in finding more defects. According to Gilb & Graham, the checklist items are derived from rules - they are just less formal interpretations (Figure 8).

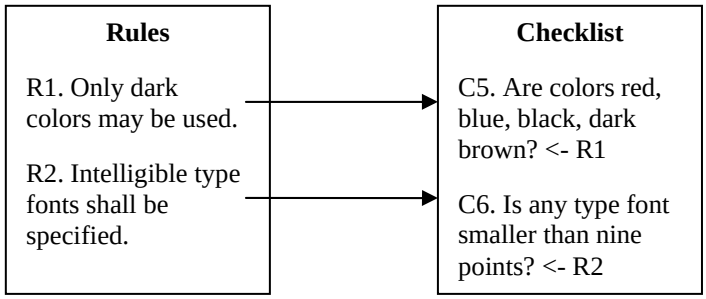


Figure 8. Checklists are extensions of rules (Gilb & Graham 1993:60)

Gilb & Graham (1993) require that every checklist item should include a reference to the rule tag which they are interpreting. They also remind that checklist questions cannot be used to make new rules. On the other hand, Wiegers (2002) sees checklists and rules as a supplement or even an alternative to each other.

Checklists should be kept short and they should never exceed one page. If some checks can be done using automated tools, there is no point in putting these kinds of questions on a checklist. Also too general questions should be avoided. (Brykczynski 1999.)

It is not necessary to have every possible question on a checklist, but to focus on questions which will turn up major defects. Copying a checklist from another environment is not recommended, because the checklists should be built based on experience. Of course an example checklist can be used when starting to practice reviews, but the checklist should be updated and tailored later on to meet specific needs. (Gilb & Graham 1993.)

Checklists are not necessarily owned by a specific person, but someone should take the responsibility for updating them whenever needed (Gilb & Graham 1993). When checklists are updated regularly, it is more likely that the inspectors find additional defects by using them (Brykczynski 1999).

Procedures

The purpose of procedures is to describe how to do the review process: what is the expected behavior and activity during different phases. There are procedures for example for meeting-etiquette. It describes how to act in a meeting, including how to report issues. Procedures are best-known practices and they should also be updated whenever needed. (Gilb & Graham 1993.)

3.4 Collecting and using review metrics

Many organizations hold reviews without collecting any data about the review process. According to Gilb & Graham (1993) it is the same as working blind, because you have no idea how well the process is working or is it working at all. By collecting review data it is possible to understand the development and

quality process better and to improve the process based on the metrics (Wiegers 2002).

Collecting data from multiple reviews gives a historical perspective and allows organizations to base decisions on facts instead of assumptions. Planning time allocation for future reviews is also easier when historical data is available. (Wiegers 2002.)

Regular detailed metrics can also be used to estimate how many additional defects will be found in the remaining life-cycle phases or through customer (Wiegers 2002). Furthermore, the metrics reveal what things are actually costing the organization time and money. However, the most important thing is that with the review metrics it is possible to find out if reviews are even profitable to justify their existence. (Gilb & Graham 1993.)

While collecting data and calculating metrics, it is important to remember that metrics must never be used to reward or penalize individuals. The data collection should be kept objective and impersonal to avoid any distortion. (Wiegers 2002.)

3.4.1 Basic data items

Basic data items, which should be collected from every review, can be classified to four different categories. These categories are size, time, effort and quality.

- *Size* covers the size of the reviewed document. The unit used for measuring it depends on what type of document is being reviewed. It can be lines of code, document pages, number of test cases, etc.

- *Time* covers the duration of the review meeting. If a re-review is needed to complete the process, both meetings will be calculated together.

- *Effort* covers the total labor hours that are spent during the review process. It can be subdivided into the different review phases (planning, kick-off meeting, individual checking, logging meeting, edit and follow-up).

- *Quality* covers the number of defects found and corrected. The defects are usually classified to major or minor based on their severity. Wiegers (2002) defines that major defects include inconsistencies between the work product and the source documents, and could cause wasted time through rework or customer problems. Minor defects, on the contrary, include cosmetic problems such as incorrectly spelled text and functionality or usability problems.

Besides these four categories, there are also some other review items that should be recorded. These are at least the number of reviewers and the appraisal of the reviewed document. (Wiegers 2002.)

3.4.2 Minutes of the review

It is easier to use specific forms for collecting data from reviews. Literature provides different kinds of templates for forms, but organizations should tailor their own forms with their own local terminology and content.

The most important forms are a summary report and an issue log. The summary report describes the reviewed document, identifies the review participants and their roles, displays the preparation hours, etc. The issue log is used to record details about the found defects. In addition to these forms, there can be a different form for minor items such as typographical errors. (Wiegers 2002.)

3.4.3 Analyzing the review metrics

There is no point of collecting these review data items, unless they are used and analyzed afterwards. Although it is possible to make an analysis starting from the first review, it is more valuable to collect data from multiple reviews to be able to calculate averages and sense trends.

Different metrics can be calculated from the collected data. Here are a few examples:

- *Defect density,* which describes the number of defects found per unit of the reviewed material.
- *Effort per defect,* which gives the total labor hours spend to find a defect.
- *Percentage of major defects,* which tells if the review focus has been on finding minor or major errors.
- *Review rate,* which counts the quantity of reviewed material per meeting hour.

(Wiegers 2002)

The collected data items can also be used to judge the general value of the reviews. This value can be measured through calculating the effectiveness or the return on investment. (Wiegers 2002.)

The effectiveness means the percentage of defects found in the review compared to the total amount of defects in the product. This requires that also the defects found in later testing stages or by the customers should be recorded. The effectiveness can be used for example to estimate how many defects remain in a document in the following review. (Wiegers 2002.)

The return on investment (ROI) is a cost/benefit analysis, which can be calculated as follows:

$$\text{Return on investment} = \frac{\text{Net savings}}{\text{Detection cost}}$$

Net savings is an estimated cost of fixing a defect in the future minus the actual cost of fixing it when it was found in a review. It may be pretty difficult to estimate the cost of fixing a defect in later phases, but usually some average cost is known and can be used in calculations. The detection cost is the actual cost of the review. ROI should be a little over 1.0 to justify the reviews. (Wiegers 2002.)

3.5 Review guidelines

Some guidelines for conducting reviews are already described earlier in this chapter, but the key factors for successful reviews are summarized here.

The most important guideline is probably to review the product, not the producer. The reviewers should select their words carefully and point out the errors gently. This way the tone of the meeting stays constructive and the reviews are more effective. (Pressman 1997.)

It is important to set an agenda for a review and maintain it. Meetings should be kept on track and on schedule, and extra debate should be limited. The main focus is on finding defects and other problems, but solving the problems should be postponed until after the review meeting. It is also essential to keep a record about the reviews. (Pressman 1997.)

The review teams should be kept small, usually between three and seven participants. Studies have proved that the optimum number of people in a review is ~3.14, so in practice this means three or four persons. (Freedman & Weinberg 1982.) The review material should be received several days prior to the meeting and participants must be well prepared in advance (Wiegers 2002).

Besides these guidelines, the basis for good reviews comes from the management side. As Wiegers (2002) says, even motivated team members will struggle to perform the reviews, if the management commitment is not obtained. This means that time for reviews and rework has to be allocated in the project plan.

4 Reviews in the Messaging test team

The purpose of this thesis was to define a technique for reviewing test cases in the Messaging test team. In the Messaging test team it was noticed that there is no systematic way for conducting reviews. This leads to a situation where the same mistakes appear in the test case specifications time after time.

4.1 Present situation

Considering the testing levels described in chapter 2, the Messaging test team focuses on system testing and especially on functional testing. As illustrated in Figure 2, the input documents for this testing level come from requirements specification phase. In practice this means that the test cases are created based on requirements specifications and UI specifications that extend the requirements.

The test cases have to be of high quality, and there are many reasons for that. First of all, we have several test engineers in our own team and it is not uncommon that we execute test cases that some other team member has specified. Although it is easy to go and ask a colleague for help with an unclear test case, it still takes extra time.

It is not enough that our own team members understand how the test cases should be executed, because the same test cases are used in many other teams inside the company, too. These other teams can be situated in a different city or country, so asking for help is not so easy anymore. Furthermore, these other teams are probably not as familiar with the Messaging application as we are. Because of that, the test cases have to be specified so that almost anyone is able to execute them.

This same problem can be broadened out even more, because some of the test cases are delivered also to our customers outside the company. And it is obvious that no-one wants to deliver unfinished or faulty documents to customers. Customers receive test cases also from other teams besides our team. Due to that it is important to create consistent test cases using general guidelines for style, outlook and content.

We do have separate proofreaders who will check our test case specifications before they are delivered further. But that does not give us an excuse to write test cases any worse – quite the contrary. If our test cases are unclear, proofreaders have to contact us before they are able to do their work. And this again leads to the situation, where many people have to do extra work to get things done. In addition, proofreaders focus mainly on spelling issues, so actual defects have to be caught before delivering the test case specifications to the proofreaders.

4.2 Survey

4.2.1 Methods

I started to study the present situation by participating into different reviews as often as possible. During five months, I attended about 20 reviews, most of them being test case reviews. Usually I acted as a reviewer myself, but a few times I also acted as a silent observer.

In a typical test case review, there are at least three participants besides the author: a developer, an UI designer and another test engineer. The test case specification is sent to the participants few days before the review meeting and everyone should read the specification carefully and mark any defects found in it. In the review meeting, the specification is gone through one test case at a time. The author leads the meeting and (s)he also writes down the comments that the participants point out.

When I observed the reviews, I mainly focused on what kinds of defects are typically pointed out. I also paid special attention to what kind of defects different persons point out: is there a difference between issues that a developer, an UI designer and a test engineer found. Besides focusing on defects, I also observed general issues: how people are prepared, does the meeting keep on track, is the pace of the meetings suitable, etc.

I also thought that other test engineers could have useful information and opinions on our reviews. Due to that I organized a short group discussion among the test engineers. The discussion was hold as a free-form discussion in the end of June 2006 and 9 persons were participating. The purpose was to find out what kind of problems other people had noticed in our reviews and do they already have some improvement suggestions on their mind. I also wanted to get feedback on the review practices that I had gathered from the literature.

This session took place conveniently a few days after quite an unsuccessful review meeting and due to that most of the participants were very anxious to get some improvements. The subject matter focused first on identifying problems in the reviews. After that we tried to find and list improvement suggestions to the most common problems. Every now and then I also presented some questions like “What do you think if we would assign specific roles to reviewers?”.

Besides observing the reviews myself and gathering information from other test engineers, I also noticed that proofreaders could be a useful source when figuring out typical problems in our test case specifications. Due to that I received and studied the list of the most common issues they have to correct in our test specifications.

4.2.2 Results

When I observed the reviews, I noticed that the following problems were repeated:

- Lots of typographical errors or wrongly spelled terms
- People were not prepared for the review
- The discussion drifts to another subject
- No data was collected from the reviews (the author only wrote down the found defects)
- No checklist / writing rules were used

Sometimes the reviews also lasted over two hours and people started to feel too tired. Also one time the reviewed document was clearly not ready for the review and due to that many people's time was wasted when trying to go through a document full of defects.

I noticed that different persons found somewhat different issues from the test case specification. Developers focused mainly on functionality and testability: does the product really work that way, is it possible to execute the test, etc. Test engineers had often noticed those kinds of issues, too, but they also paid a lot of attention to writing issues: are the terms spelled correctly, are there some words that should not be used, etc. The role of the UI designer seemed to be to check that there is no inconsistency between the UI specification and the test case specification.

Other test engineers had noticed pretty much the same kind of problems as I did. Besides these actual problems, there was some kind of uncertainty about organizing reviews: who should be invited, when the invitation should be sent, what is the number of test cases that can be reviewed in one meeting.

It was not so easy to figure out the solution proposals for the problems. Some improvement suggestions were still discovered in the group discussion and we already agreed about few issues, too. It was decided that the maximum number of test cases in one review should be 25. If the number is higher, test cases have to be divided into chunks and two or more review meetings should be organized. In these cases the titles of all the test cases still have to be added to every invitation, so that the reviewers get the whole picture.

One improvement suggestion was also that if someone is specifying test cases for a major requirement, it would be a good idea to send the first five test cases to one person for pre-review. This way the author would get comments in advance, and at least the same mistakes would not appear in all the test cases.

People also agreed that it would be important to keep minutes of the review. Some people discussed though, that for most of us it means doing the job twice: if you do not have a laptop, in reviews you have to write the comments down on a paper and after the review copy those comments to an excel sheet.

This problem was quickly solved, because it was noticed that our team has an extra laptop that can be borrowed for this purpose.

The list of common issues proofreaders usually correct in our test case specifications did not reveal anything new. They listed things like terminology, copy-paste errors, wrongly marked internal information, missing full stops in sentences and wrongly used references.

4.3 Solution proposal

4.3.1 Review technique

After learning about different kinds of review techniques, I soon noticed that Gilb & Graham's style inspection process is far too heavy for us to practice: it is too bureaucratic and disciplined with a certified and trained Inspection Leader and six inspection phases. It would require more resources and that would most probably lead to a situation where test case reviews are not even profitable anymore.

On the other hand, in most of the cases the walkthrough and other informal review techniques are too informal and unsuitable for our purposes. It is not enough that test cases are just walked through by explaining the purpose of different cases, because the main idea is to find defects. I also think that test cases should be reviewed by more than one person, so this excludes the peer desk check and ad hoc reviews. Anyway, these two techniques are extremely suitable for the pre-review that was mentioned in chapter 4.1.2.

The pass around technique could be suitable for our purposes in situations when it is difficult to organize a review meeting, for example when the reviewers are located in different cities. Anyway, it is much easier to clarify issues face to face than trying to explain them in writing. That is why I think we should always organize a separate review meeting, if possible. So under the circumstances I think that the most suitable technique for our purposes is the team review. It is a planned and structured technique, but sufficiently flexible for us.

When I observed our test case reviews, I noticed that we are carrying out reviews mainly as described in chapter 3.2.2. It means that we do not have to make any major changes to our reviews, because we are already using the most suitable technique for our team. We should just tailor and improve the way we practice it.

4.3.2 Collecting data and metrics

I think the biggest weakness in our reviews is that we do not collect any data from them. This is the main reason why the similar defects appear in our test

case specification time after time – we just do not pay any attention to what kinds of defects have been found in the earlier reviews. As Gilb & Graham say, carrying out reviews without collecting any data can be considered as working blind. So the first improvement we have to make is to start taking minutes of our reviews.

We had already discussed about the issue in our group discussion, and also other test engineers thought that taking minutes of our reviews is important. It was agreed that the review minutes would be saved to our network drive so that they are available whenever needed. My task was to define a suitable template for reviews.

As mentioned in chapter 3.4.2, literature offers different kinds of templates for collecting data from reviews. I studied a few of these templates, and considering our own needs, I selected suitable data items and created a template based on them. The template is created using Microsoft Excel and it can be found from Appendices 1 - 3. Next I will go through the main issues from it.

I divided the template into three different worksheets: summary, issue list and metrics. This kind of division was also used in example templates and I think the division is clear and logical. As mentioned in chapter 3.4.2, it is also possible to have a separate form for typographical errors, but I do not think it is necessary. On the contrary, I think it is easier to make corrections based on just one list.

The summary-sheet has general information on the review. It first identifies the reviewed document and shows the details from the review meeting. After that the reviewers are listed. If some roles or responsibilities have been assigned to the reviewers, it is also mentioned here. There are also different fields that are used to calculate metrics and to sum up the spent hours: the meeting duration, preparation hours and rework hours. At the end of the summary-sheet, the product appraisal is marked.

The issue list focuses on the found defects and other issues. First there is a field for general comments. The reason for putting the field here is that sometimes we notice a general improvement idea in the review meeting and it should be written down somewhere. We can, for example, agree that some specific names should be bolded, so that they are more visible in our specifications.

After the general comments there are short instructions on how to use the issue list. The columns in the issue list are defined so that they suit for test case reviews. The error type of the found defect has to be defined and defects also have to be classified as major or minor. The classification should be made as described in chapter 3.4.1: a defect is major if there is inconsistency between the test case and the requirement/UI specification or it could cause wasted time

for example through an unnecessary bug report. A minor defect means that something could be done better, but the test case can be executed as it is.

The last sheet is for metrics. The metrics are calculated automatically from the other sheets, so the user does not have to fill any field in this sheet. I chose to calculate a few basic metrics that could be useful for our purposes.

4.3.3 Checklist

Another thing that could improve our test case reviews is a suitable checklist. We do have other input documents (source documents, procedures, writing rules) available in our database, but a concrete checklist is missing.

As was told in chapter 3.3, Gilb & Graham require that every checklist item has to be derived from rules. I started to create us a checklist based on that idea, but after a while I noticed that there are also other kinds of defects in our test case specifications than what the writing rules cover. Due to that I adopted Wiegers point of view and created a checklist as a supplement to rules.

I divided the checklist to four categories: test objectives and test steps, functionality, style and other issues. I took care that the checklist does not exceed one page and finally I had 15 questions on the list. Those questions are mainly selected based on the data I gathered when observing different reviews. Although Gilb & Graham emphasize that checklist questions should focus on issues that will turn up major defects, my list focuses also on minor defects. The reason for this is that at the moment it seems that most of the defects in our test cases are minor. The checklist can be found in Appendix 4.

4.3.4 Other issues

Participants

In the group discussion it was mentioned that it is not always clear who should be invited to the review. The invitation should be sent for at least four persons: to the developer who implements the requirement (or developers if there are many persons implementing the same requirement), to the UI designer who specifies the requirement to the UI specification, to another test engineer in own team and to the test team leader. Sometimes it is also useful to send the invitation to more than one test engineer, so the specification is checked more carefully or the responsibility can be shared.

For the test team leader the invitation is more like a status report, so (s)he knows that the test cases are ready. But other persons are mandatory and I suggest that the author also requires an approval from everyone, even if someone is not able to participate in the review meeting.

When to send the invitation

There was also uncertainty about how much earlier the test case specification should be sent to the reviewers. The literature did not give an exact guideline for that, it was only said “several days before the review meeting” (chapter 3.5).

In our team the practice seemed to vary a bit: sometimes the specification was sent one week before the review meeting and sometimes it was sent only two days before the meeting. First I thought that this could be the reason why people are not prepared for the reviews – the specification is sent so late that the reviewers do not have enough time to read it properly.

Anyway, I noticed that even if the specification is sent more than one week before the meeting, people read it only a day or two before the meeting. I also noticed that if I checked the specification through already one week before the meeting, it was not so easy to remember what I had meant with my comments. So I came to the conclusion that the earlier the better, but the deadline for sending the specification to reviewers is two workdays before the review meeting.

Number of test cases

It was already decided in our group discussion that the maximum number of test cases in one review is 25. I think this also improves our reviews, because going through 25 test cases most probably will not exceed the two hour limit that was mentioned in chapter 3.2.1.

Roles and responsibilities

In chapter 3.2.1 it was explained that it is useful to define specific roles or responsibilities for the reviewers. I wondered if this could be a good idea in our situation, but came to the conclusion that defining specific roles does not improve our reviews. The reason for this is that we kind of have them already, because the developer, the UI designer and the test engineer check the test cases from their own point of view.

But if there is, for example, two test engineers participating in the review, the responsibility can be shared among them. This can be done, for example, so that the other test engineer checks test cases 1-10 and the other one test cases 11-20. Or perhaps the other test engineer focuses on style issues and the other one on testability and other issues.

Anyway, even if the roles or responsibilities are not defined, it could be useful to ask some persons to start reading from the beginning of the specification and the others from the middle of the specification. This is because usually there are lots of comments on the first test cases, but none on the last ones. So most probably the last ones are not read so carefully anymore.

Review meeting

One of our problems in reviews was also that people are not prepared for the review meeting. In these situations I suggest that we would be more strict and act as mentioned in chapter 3.2.1: postpone the review meeting if there are persons who have not checked the test case specification. I think that people rather read the specification than reschedule the meeting.

5 Conclusion

A review is an effective way of removing defects from a product in the early phases. It can be seen as a systematic evaluation process, during which the product is examined by a team of qualified persons.

This thesis process showed that there are several different ways for conducting reviews. These techniques differ in their formality and amount of phases. People should always choose the technique that suits best for their purposes and tailor it according to their needs.

The Messaging test team had accustomed to a certain kind of review technique. During this thesis process it was noticed that the technique itself is suitable for the team, but some improvements should be made, though.

5.1 Analysis

The purpose of this thesis was to find means to improve the test case review technique in the Messaging test team. The purpose was also to create a checklist for the test case reviews and define a review template for the review meetings.

I think these objectives were quite well met. I studied the literature and observed the present review technique in the Messaging test team. Based on the literature and observation, I created a suitable checklist and a review template. I also considered other ways to improve the review technique, but those improvement suggestions were quite minor.

In my opinion, the most important improvement is that data from the review meetings will be collected. This way it is possible to see what kinds of mistakes are repeated time after time and pay special attention to them. I also think that people start to be more prepared for the review meetings, because they know that preparation times are collected.

Defining the review template was a pretty easy task, because the literature offers different templates and reasons why each data item should be collected. But creating the checklist was much more difficult, because every checklist item had to be tailored for the Messaging test team’s purpose. I still think that I managed to list at least some important questions on the list.

When I received this thesis topic, I found it very interesting and also reasonably challenging. Unfortunately my motivation came somewhat down at the end of this process, and I could not keep the schedule I had originally planned. Anyway, as a whole, I am quite satisfied with this process, because I learnt lots of new issues about reviews.

5.2 Future plans

The improvement ideas presented in this thesis are just a start for the improvement process. After data from review meetings is accumulated, it is important to use and analyze it. So, in the future there should be a separate template for calculating the review metrics.

When the review template has been used for a while, we might notice that some items are missing or there are items that are not valuable. In those situations the review template should be updated.

Also the checklist should be updated whenever necessary. When the checklist is used, people start to pay special attention to the issues mentioned on the list and most probably those mistakes will be reduced. In this case, new questions should be generated based on the data collected from the reviews.

References

ANSI/IEEE 1028-1997. Standard for Software Reviews.

Brykczynski, Bill. 1999. A survey of Software Inspection Checklists. Software Engineering Notes vol 24 no 1, 82-89.

Freedman, Daniel P. & Weinberg, Gerald M. 1982. Handbook of Walkthroughs, Inspections, and Technical Reviews. Toronto: Little, Brown and Company.

Gilb, Tom & Graham, Dorothy 1993. Software Inspection. London: Addition-Wesley.

Haikala, Ilkka & Märijärvi, Jukka 2004. Ohjelmistotuotanto. Hämeenlinna: Karisto Oy.

Patton, Ron 2001. Software Testing. Indiana: Sams Publishing

Pressman, Roger S. 1997. Software engineering: a practitioner's approach. New York: The McGraw-Hill Companies, Inc.

Tamres, Louise 2002. Introducing Software Testing. London: Addition-Wesley.

Tian, Jeff 2005. Software Quality Engineering: Testing, Quality Assurance, and Quantifiable Improvement. New York: Wiley.

Wiegers, Karl E. 2002. Peer Reviews in Software: a practical guide. London: Addition-Wesley.

Appendices

Appendix 1: Review template – Review summary sheet

Review summary

Document information	
Document name:	
Release	
Author:	
Number of test cases:	

Meeting information	
Meeting date:	
Meeting time:	
Meeting duration (h):	
Meeting place:	

Reviewers					
#	Name	Preparation time (h)	Present in review	Comments by e-mail	Role / Responsibility
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					

Effort hours		***Product appraisal (and comments if not accepted)***		
Rework hours (h)		Accepted as it is		
Total preparation time		Accepted with actions		
Total effort (preparation, meeting, rework)		Re-review needed		

Appendix 2: Review template – Issue list sheet

Issue list

General comments

Instructions	
Step #:	Step number / Objectives / Preconditions / Title
Field:	Description / Expected
Error type:	Missing / Wrong / Extra / Typo / Style / Clarification /Question
Severity:	Major / Minor
Status:	Open / Corrected / Ignored

Issue #	Test case #	Step #	Field	Description of issue	Error type	Severity	Status	Comments
1								
2								
3								
4								
5								
6								
7								
8								
9								
10								
11								
12								
13								
14								
15								
16								
17								
18								
19								
20								

Total :
0

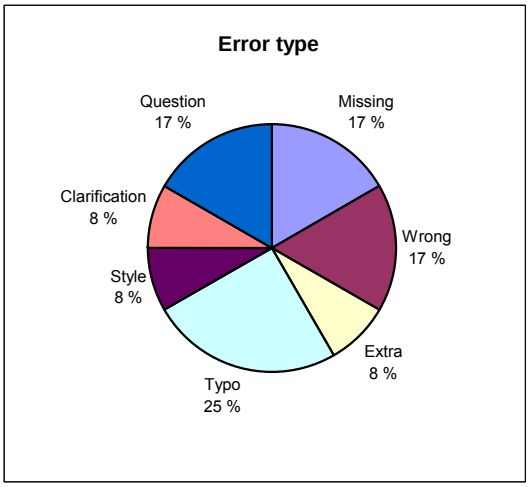
Appendix 3: Review template – Metrics sheet

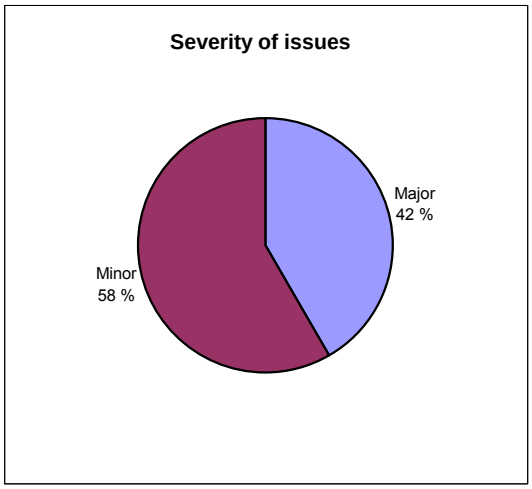
The data in this sheet is unreal, but it was added to visualize the use of the diagrams.

Metrics

Error type	Amount	%
Missing	2	17
Wrong	2	17
Extra	1	8
Typo	3	25
Style	1	8
Clarification	1	8
Question	2	17
Total	12	100,0

Severity	Amount	%
Major	5	42
Minor	7	58
Total	12	100,0





OTHER METRICS

Defect density *(defects/test case)*	0
Effort per defect *(hours)*	0
Review rate *(reviewed test cases per hour)*	0

Appendix 4: Checklist

CHECKLIST FOR TEST CASE DESIGN

Test objectives and test steps:

- □ Are the test objectives specified properly:
  - They contain all the items that are tested in the test steps.
  - There are no logical strings mentioned.
  - They start with words “Test objectives: Test that”.
- □ Is every necessary item mentioned in the Preconditions, but nothing that is not used in the test steps?
- □ Are all the test steps unambiguous, so that the tester has no alternatives (no “if”, “probably”, etc. words)?
- □ Is the number of steps in one test case less than 20?
- □ If the expected result is the same for many test steps, is the result defined in detail only once (so that the other steps have a more general description)?
- □ Is the Options menu and the submenu tested in their own test steps?

Functionality:

- □ Is also the functionality of the software tested, not only the UI (that a button also works correctly, not only that it has a correct label)?
- □ Is all the functionality mentioned in the requirement covered?
- □ Is it possible to execute all the test cases?

Style:

- □ Are the logical strings written correctly with the §-marks, small letters and underscores between the words?
- □ Are all the terms and names written correctly (using capital letters, if needed)?
- □ Is the internal data marked correctly?
- □ Are the test case titles as descriptive as possible, using the “-ing” form when beginning with a verb?

Other:

- □ Are all the test cases necessary?
- □ Are test cases with new logical strings (or old logical strings in a new context) marked as Language variant cases?