A Balance between Testing and Inspections
- An Extended Experiment Replication on Code Verification

Hanna Scott
This thesis is submitted to the School of Engineering at Blekinge Institute of Technology in partial fulfillment of the requirements for the degree of Master of Science in Software Engineering. The thesis is equivalent to 20 weeks of full time studies.

Contact Information:
Author(s): Hanna E. T. Scott
Address: Skepparegatan 26
37135 Karlskrona

University advisor(s):
Professor Claes Wohlin
Department of Systems and Software Engineering

School of Engineering
Blekinge Institute of Technology
Box 520
SE – 372 25 Ronneby
Sweden

Internet : www.bth.se/ipd
Phone : +46 457 38 50 00
Fax : + 46 457 271 25
The empirical investigation method chosen for this thesis investigates the verification of two C programs; through an in depth experimental comparison between the inspections meeting preparation through inspection based reading with checklist based inspection preparation, and structured testing using test cases.

The experiment conducted within this master thesis is an extended replication of an earlier experiment conducted in 2003 by Per Runeson and Anneliese Andrews at Washington state university. The replication experiment was extended with the complementing inspection preparation using a checklist, and was executed with eight subjects, due to a large drop-out rate among participants. The drop-outs created a validity issue of low statistical power in the result, and therefore the statistical analysis was largely replaced by a qualitative investigation with interviews of the participants. The most surprising indication in the result of this qualitative investigation is that there may exist a lower programming experience limit in order to successfully verify code.

The descriptive statistical analysis of the result shows that the replication experiment group could be compared to the original experiment group, thus giving the result a slightly higher validity and showing that the replication was a success. The replication experiment result is contradictory to the original result in that it shows inspections to have a higher detection rate than testing, and also that inspections are a more efficient way of verifying code than testing.

**Keywords:** Inspections, Testing, Scenario-based reading, unit testing
CONTENTS

ABSTRACT .................................................................................................................................. I

CONTENTS ................................................................................................................................... II

1 INTRODUCTION .......................................................................................................................... 1

2 THESIS OBJECTIVE ................................................................................................................... 2

3 VERIFICATION THROUGH TESTING ...................................................................................... 3

   3.1 OVERVIEW OF TESTING .................................................................................................... 3
   3.1.1 Unit Testing ................................................................................................................. 3

VERIFICATION THROUGH INSPECTION .................................................................................. 6

   3.2 THE ORIGIN OF INSPECTIONS ....................................................................................... 6
   3.2.1 Fagan’s Inspections ...................................................................................................... 6
   3.3 INSPECTION VARIATIONS .............................................................................................. 7
   3.3.1 Active Design Review ............................................................................................... 7
   3.3.2 Two-Person Inspections ............................................................................................ 7
   3.3.3 N-fold Inspection ....................................................................................................... 7
   3.3.4 Phased Inspections .................................................................................................... 8
   3.3.5 Inspection without Meeting ...................................................................................... 8
   3.3.6 Gilb Inspections ......................................................................................................... 9
   3.4 READING TECHNIQUES – INSPECTION SUPPORT TOOLS .......................................... 9

4 RESEARCH METHOD ............................................................................................................... 11

   4.1 METHODS AVAILABLE .................................................................................................. 11
   4.1.1 A Large Experiment ................................................................................................. 11
   4.1.2 Small Experiment and Other’s Experiment Data ....................................................... 12
   4.1.3 Medium Sized Experiment through Replication ..................................................... 12
   4.1.4 Post Mortem of an Old Project ................................................................................ 12
   4.1.5 Case Study within a Real Company ....................................................................... 13
   4.2 CHOICE OF METHOD ................................................................................................... 13

5 RELATED WORKS .................................................................................................................... 15

   5.1 EARLY EXPERIMENTS ................................................................................................. 15
   5.2 RECENT EXPERIMENTS ............................................................................................... 15

6 THE EXPERIMENT .................................................................................................................. 17

   6.1 EXPERIMENT PREPARATION ..................................................................................... 17
   6.2 EXPERIMENT PLANNING ............................................................................................ 17
   6.3 EXPERIMENT RE-PLANNING ..................................................................................... 18
   6.4 EXPERIMENT EXECUTION ......................................................................................... 19
   6.5 ORIGINAL EXPERIMENT ANALYSIS ....................................................................... 20
   6.6 ANALYSIS RE-PLANNING ......................................................................................... 20

7 RESEARCH RESULTS .............................................................................................................. 23

   7.1 RESULT VALIDITY ......................................................................................................... 23
   7.1.1 In both the Original Experiment and Replication ................................................... 23
   7.1.2 Validity in the Replication ...................................................................................... 24
   7.2 STATISTICAL COMPARISON OF THE TWO GROUPS ................................................ 24
   7.2.1 Types of Defects ...................................................................................................... 25
   7.2.2 Method Related Patterns ....................................................................................... 26
   7.2.3 Experience Related Patterns .................................................................................. 26

8 LESSONS LEARNED FROM THIS THESIS ............................................................................ 28
1 INTRODUCTION

There are a number of ways to verify software throughout the software development process, and it has been generally agreed that the earlier the software defects are found, the cheaper it is to remove, because there is less rework to be done. The problem is that however good removing defects early may be, there are still plenty left at the end of the implementation phase in a software project. Perhaps this is why software testing has developed along with the ever evolving ways of developing software, and in turn is why software testing has become the natural way of verifying the software code.

However, there is another way of verifying program code, namely that of software reviews and inspections. These techniques consist of different ways of reading through the code, looking for defects, instead of as in testing executing the software program. The usefulness of these methods have from time to time been investigated but this far the results have only clearly shown one thing - that testing detects more defects than the alternative methods. Little is for example known of how testing and inspection methods complement each other in terms of what kinds of defects the different methods can find. It is a large field of comparative investigation that needs to be investigated through empirical studies to see if the expensive task of testing can complemented in such a way that the software verification costs are decreased or the amount of defects found in verification can be increased. This is why this master thesis investigates the balance between one method of code testing and one method of code inspections, with the aim of determining what defect types can found by each of the two methods and the efficiency of each method.

The content of this thesis is structured so that the reader is given basic information on testing and inspection techniques first. This is followed by a section on how the empirical investigation method for this thesis was evaluated and chosen, and an investigation of works related to the method chosen. Only thereafter is the empirical method chosen described with regards to design, execution and analysis. The empirical method chosen for this thesis is an experiment replication, and the last sections of this thesis contain the results and conclusions that can be drawn from the analysis thereof, along with a small section on lessons learned from the experiment that was conducted.

It should also be noted that there is a terminology section present on page 35.
2 THESIS OBJECTIVE

The aim of this thesis is to investigate what defects can be detected using code testing respectively code inspection, and to see if one of the two methods is more efficient in finding defects than the other. The hope for the result is that it will give some idea as to if the two methods chosen for comparison, could actually be combined, to lower the costs of verifying software code.
3  VERIFICATION THROUGH TESTING

Before the empirical investigation method for this thesis is chosen, an overview of the testing and inspections is presented in this and the following chapter. This chapter will focus on the overview of testing, and the testing techniques available within unit testing, in order to give the reader a basic understanding of the testing techniques available, and the testing technique used in the empirical investigation for this master thesis.

3.1  Overview of Testing

The history of testing is not as easily described as the history of inspections. The main reason for this is that while inspections were created for human use in defect detection, testing has evolved based on the limitations of the hardware and software on which the programs created are executed.

Testing methods are usually divided into “white box testing” “grey box testing” and “black box testing”. In black box testing, the program is treated like a black box, and none of the code is visible to the tester. In white box testing all code is visible to the tester, and the grey box technique is when parts of the code are visible, and others are not. These categories describe the visibility of the code being tested, but the areas in which tests are executed within these categories are limited to processing, control, data structures or data validation (Beizer, 1996). This section only focuses on the “white box testing” category, which is sometimes referred to as “unit testing”, because it is the verification of units that is investigated in this thesis along with inspections as is shown later in this thesis.

There is no best way to test software, but there is something that could be called a minimum test requirement, namely the use of the structure-based path testing (Beizer, 1996). Structure-based path testing is in essence the testing technique that is used in the empirical investigation of this thesis.

3.1.1  Unit Testing

Unit testing has four main components (Beizer, 1996):

1. Mechanical analysis
   Syntax testing done by compiler, assembler or interpreter.

2. Manual Analysis
   Things that the computer cannot do such as: cross-referencing variables, labels, functions, variable types, variable value with variable types etc.

3. Audits and reviews
   A review is an examination of program elements with the goal of discovering mistakes made in the code and should be a part of the unit testing. Reviews are conducted in groups, and one of the goals is to guide the implementer to verbalize the intention of his code, in order to find the defects in it. Reviews can also be applied to other software artefacts, such as development documents.

   An audit, on the other hand, is an examination of all kinds of software artefacts, in an attempt to find defects. The implementer has no input to the examination process.

   In short, an audit results in a project state description, while a review results in an overview that indicates the direction of the project.

4. Execution, correction and refinement
The part of unit testing most important to this experiment is the execution phase, because it is here that the different testing techniques are applied. The test technique used in the experiment basically consists of path testing, even though mechanical checking using a compiler is done as well. The more specific techniques available in this phase are:

**Path Testing** which consists of testing a selection of executable series of instructions. The aim is to choose the paths to test in order to show that the programs intended functionality is present and fulfilled (Beizer, 1996).

Path selection criteria for path testing (Beizer, 1996):
1. The set of paths shall ensure that every instruction has been executed once or more.
2. All decision points where the program flow diverges shall be executed so that all divergent path directions have been visited at least once.

If the set of paths tested fulfill the selection criteria the path testing provides complete coverage. Note that this is not the same as having complete path testing.

**Loop Testing** consists of testing program loops for defects. If path testing is done properly loop testing is redundant, but in reality path testing seldom finds the loop defects – especially if the program being tested contains nested loops (Beizer, 1996).

Test cases that loop testing should contain:

1. Excluding the loop
2. One-loop-execution
3. Two-loop-execution
4. A typical number of loop executions

Another element that needs to be tested is the conditional boundaries for loops in the program. This mainly entails testing below, on and above the maximum number of loop executions.

**Transaction Flow Testing** entails functional testing of what a system user considers a unit of work to be. This unit of work is usually referred to as “a use case”, and are just one abstraction level away from the programs natural language functional requirements – specified by the customer. The use cases can be divided into steps, or tasks, sometimes referred to as functional scenarios. The scenarios in a transaction flow (use case in execution) correspond to the processing steps in a flow chart. Note that these tasks, or scenarios, are abstractions of sets of instructions, and therefore there can be points where the program flow diverges and converges within a task, just as there can be points of execution divergence and convergence between tasks in the flow-chart (Beizer, 1996).

In short, transaction flows are more visual, formal and technical descriptions of a system’s functional requirements and it is created during the design phase of the software development cycle (Beizer, 1996).

However, in transaction flow tests there are inconsistencies between the requirements and their implementation. Path selection criteria should therefore reflect the highest-priority-ranked requirements (Beizer, 1996).

**Logic-based Testing** is when the content of the decision table of a program is converted into boolean algebra-expressions (Beizer, 1996). Through this, the rules of boolean algebra can be used to analyze the programs decision-points to see:
1. That the logic is complete, unambiguous and non-contradictory
2. That the decision-points do not contain unnecessary complexity or redundancy conditions.
3. That the decisions do not lead to infinite loops.

Logic-based testing is used in addition to path testing to test parts of programs that contain a lot of TRUE-FALSE alternatives (Beizer, 1996).

**State-transition Testing** is a way of testing the computation that leads the program from one state to another. Just as in path testing it is too time consuming to test all states and transitions in a program, so a set of tests are selected, preferably so that all transitions are executed at least once (state transition coverage). A state test set should always begin and end in the same state for easy testing, and all test sets shall include (Beizer, 1996): input, corresponding transitions and output.

**Input validation and syntax testing** is a functional testing technique that tests program input through trying to give both valid and invalid syntax input to the program. This testing method is part of the functional tests of the system or part of the acceptance tests, and it aims to demonstrate that (Beizer, 1996):

1. The program does not fail because of bad inputs.
2. Bad input does not cause failure in another element, even if the first one does not fail itself.
3. The element does not corrupt data as a result of bad input.
4. The element rejects all bad input and accepts all good inputs.
5. The element performs correct processing on good inputs

This kind of testing is a test of the systems integrity as well as its compliance with specifications.

Having given an overview of the testing area, the next section moves on to give an overview of the inspection area, along with a description of the most common inspection techniques available.
In order to complement the testing overview of the previous chapter, this section gives an overview of the inspection area, and the inspection techniques available, before moving on to how the empirical method was chosen for this thesis.

3.2 The Origin of Inspections
Inspecting software is a structured way of examining a software artefact of some kind. The artefacts range from requirements specifications and design documents to the actual implementation of a system.

One of the great advantages of inspections is that they can be applied early in the software development process hence finding the defects earlier, which in turn leads to lower defect removal costs (Gilb and Graham, 1993). So why do we not only use this cost effective method of verifying software artefacts? Because there are still things that cannot be done using inspections such as to visibly show that the system implementation seems correct in execution. This can only be done through executing the actual code of the system to be verified (Gilb and Graham, 1993) (Beizer, 1997).

The method of software artefact inspection was officially introduced by M.E. Fagan in 1976 (Fagan, 1976). Since then, the concept of inspections has been dissected, shortened, extended and changed a number of times.

This section presents Fagan’s original suggestion for the inspection method. The major method variations on the inspection method are also presented, with focus on the criticism that caused the changes.

Fagan’s Inspections
The technique that Fagan introduced back in 1976 contained five phases and four specific roles needed within the inspection team (Fagan, 1976). The roles are: Moderator, Designer, Implementer and Tester. These roles each have a set of work tasks within the phases of the inspection. The phases and tasks are briefly described like this:

- **Overview**: The designer presents an overview of the area in which the inspection is to take place (Design, requirements etc). The designer distributes and explains the design of the specific area to inspect.
- **Preparation**: Participant homework is a way to get the inspection participants to understand the design and the intention the part of the system to be verified. They study anything that gives hints to what the most defect prone areas are, such as previous inspection results etc.
- **Inspection meeting**: A reader is appointed from the inspection group, to explain the use of the artefact, to make sure that all participants have interpreted the artefact correctly. The artefact is then searched for defects. Defects found are documented by the moderator.
- **Re-work**: All problems encountered during the inspection are resolved by the Implementer.
- **Follow-up**: The moderator makes sure that all problems discovered during the inspection has been dealt with by the implementer and the designer.

How this original suggestion has been modified over the years is described in the following section.
3.3 Inspection Variations
This original proposal has been criticized and changed many times, by several researchers. To give an overview of what method changes have resulted in which variations, the following sections will describe the most prominent variations of Fagan’s inspection technique.

3.3.1 Active Design Review
The Active Design Review was created by Parnas and Weiss in 1985. The Active Design Review method contains several small meetings instead of one large inspection meeting. The small meetings only have the preparation- and inspection meeting phases.

This method’s main aim is to focus the inspection participants’ attention on the material being reviewed. This is partially done by selecting inspection team members carefully and partially by having a specific questionnaire for each artefact in which the participants must answer questions on the inspected artefact and system. The questionnaire is created in such a way that answering the questions require a thorough study of the artefact to be inspected. Another part of the inspection that Parnas and Weiss has changed from the inspection method suggested by Fagan, is the removal of the large meeting, in favour of several small meetings between the designers and each reviewer (Parnas and Weiss, 1985).

The main problems that Parnas and Weiss aim to solve by these changes are that:

- The reviewer is swamped with information
- The reviewers are most often not familiar with the goals of the artefact
- The individuals have no clear individual responsibility
- The large meeting only gives two people the ability to interact at any given time
- The wrong people are present during the inspection
- The reviewers are asked to examine issues beyond their competence

3.3.2 Two-Person Inspections
The Two-person Inspection consists only of an author and a reviewer, but it still has all the five phases that Fagan originally suggested (Aurum et al., 2001). The two-person inspection method was presented as an empirical investigation in an article by Bisant and Lyle in 1989.

There have been investigations done on the effect the number of inspection participants has on the effectiveness of inspections, and one example is that of Bisant and Lyle, presented back in 1981, which showed no difference in performance between 3, 4 and 5-person inspection teams (Bisant and Lyle, 1989). This is why Bisant and Lyle decided to make a study of two-person inspection teams through eliminating the moderator role. The results of their investigation showed that the two-person inspection method, in particular, improved the performance of the less skilled programmers.

Possible application areas of this kind of inspection, presented by Bisant and Lyle, are in small software companies or as a transition method when a change from using reviews to using inspections is made.

3.3.3 N-fold Inspection
The N-fold inspection method suggested by Martin and Tsai in 1990 consists of a replicated effort, where several different teams separately studied the same artefact. The results of all teams were then compiled into one artefact rework session. The efforts of the different teams also needed to be coordinated, so there was a need for an extra moderator role (Martin and Tsai, 1990).

The basic theory behind the creation of this method is that two or more teams are likely to find more defects in an artefact than a single team. Martin and Tsai conducted a pilot study to verify their theories of the method’s effectiveness, where ten teams studied the same user requirements document. The result of the study showed that just two teams found
significantly more defects in the artefact than a single team did, so the indication is that the N-fold inspection method is much better at finding defects than the single team inspection method originally suggested by Fagan.

There is however no question that the N-fold inspection is more costly than Fagan’s original suggestion, so Martin and Tsai limit their application area recommendation to the initial development phase of “mission-critical software”.

3.3.4 Phased Inspections
The Phased inspection method was suggested by Knight and Myers in 1993, and is a more rigorous approach than the Fagan inspection- and N-fold inspection methods. The idea behind the creation of the Phased Inspection method is that it is not possible to search all defect categories in a single, large inspection, because the human mind does not have the capability to inspect an artefact from that many angles in the traditional two hour inspection meeting with the preceding one to two hour preparation (Knight and Myers, 1993).

The largest problems that Knight and Myers are trying to address with their inspection methods are as follows:

- There is little focus on quality aspects of software in traditional inspections
- Existing inspection methods are not applied consistently
- The result of reviews is not dependable because they are not thorough enough. There is no assertion that a reviewed product has certain properties – just because it has been reviewed.
- Existing inspection methods do not use human resources in an effective way. In meetings only one person can speak, while the others listen, so there is no parallel activity, and there is no required active participation, which in turn gives the individual the opportunity to be silent during the meeting if he or she has not prepared properly. There is also a problem with single person dominance in such meetings.
- No team is capable of checking for all types of defects simultaneously in a two hour meeting.
- No computer support is recommended in existing inspection methods.

Phased inspections can either be performed by several inspectors who use a structured pre-inspection phase to examine the artefact before the inspection meeting, or by a single inspector, where the focus lies on “a rigidly formatted process, driven by a list of unambiguous checks” (Knight and Myers, 1993).

3.3.5 Inspection without Meeting
In the beginning of the 1990’s criticism of the effectiveness of inspection meetings started to appear. The most famous critic of the inspection meeting phase was perhaps Lawrence G Votta Jr. who in 1993 presented an article, on the myths of inspection meeting benefits (Votta, 1993).

There are a number of reasons why an inspection meeting is held, and the top five according to Votta’s investigations are:

- Synergy – The group finds more defects together than all the individuals in the group can find separately.
- Education – People with little experience of inspections learn from people with more experience of inspections.
- Deadline – The inspection is a planned event towards which people work.
- Competition – It gives the inspection meeting participants a chance to show their worth.
- Requirement – The inspection process definition requires a meeting.
Votta shows, with convincing empirical evidence, that the synergy effect is too small in comparison to the cost of the meetings for the synergy effect to be a convincing argument for holding inspection meetings. Further he shows that the well disciplined meetings do not put the individual participants’ skills to work and that only two participants can interact in a meeting at any given time, which shows that these issues make waste of resources.

Votta does not however, suggest a complete removal of meetings as one of the alternatives to holding meetings. He argues that there are some features of the meetings that are too important to dismiss, such as the necessity to collect the individual reviewer’s findings, to be able to determine progress and to give all inspection participants a clear deadline to work towards. Votta simply suggest meetings with fewer participants.

There is a large amount of empirical evidence that favours removing the inspection meeting from the inspection process, but so far, no one has made a concrete suggestion of what an inspection without a meeting would actually look like. The closest resemblance found to a form of inspection without meeting is the single inspector inspection suggested by Knight and Myers in 1993.

3.3.6 Gilb Inspections
The Gilb inspection technique was presented in 1993 by Tom Gilb and Dorothy Graham. It is a very detailed version of Fagan’s original suggestion, with one exception. Gilb and Graham add a phase to the inspection process after the meeting, which is actually a second meeting, but with focus on problem root causes, not just findings. This second meeting, called Process Brainstorming, was suggested as a way to make the inspections a part of the overall process improvement in projects, through identifying where the problems originated, and go back and change the process that produced the defect (Gilb and Graham, 1993).

3.4 Reading Techniques – Inspection Support Tools
There are some support tools for inspections, referred to as Reading Techniques, which are used when the individual inspector is reading through the material before the inspection meeting (Aurum et al., 2001).

The application of different reading techniques, are attempts to improve the effectiveness of inspections. This is done through forcing the participants to view the artefact in different ways, or to force them to look for certain things – as is the case of active design reviews.

The Reading Techniques available today are:

- Ad-hoc
- Checklists
- Stepwise Abstraction
- Scenario-Based Reading
- Perspective-Based Reading

The Ad-hoc approach, is simply reading the artefact through, using your experience and skill to discover defects or inconsistencies (Aurum et al., 2001).

The Checklist approach was suggested by Fagan in the same article as the original inspection method, and is a list of items, such as defect categories or common defects, that the participant uses to keep his focus on the problem areas (Fagan, 1976).

The Stepwise Abstraction technique is a bit more elaborate. It is when the programmer starts by determining the function of each and every subsystem, and then determines the function of the entire system by putting the sub functions together. This interpretation of the function of the system is then checked against the actual system specification (Aurum et al., 2001).
The Scenario-Based Reading technique consists of creating a set of system usage scenarios that cover all defect categories. This is mostly done by creating a set of questions for each defect category, and then creating scenarios that are likely to answer the questions (Aurum et al., 2001).

Perspective-Based Reading is an extended version of the Scenario-Based Reading technique, where scenarios are developed seen from the different stakeholders’ perspectives. During the inspection meeting preparation physical models are constructed based on the scenarios for each stakeholder. The idea is that they will be cross referenced for gaps and overlaps. This gives a more visual image of what is missing or wrong (Aurum et al., 2001).

The inspection technique used in the experiment is only the individual preparation stage to the actual inspection meeting. In this individual preparation the scenario based reading technique is used, and the participants are the ones deriving scenarios from use cases provided. However, before moving on to the details of the actual empirical investigation method chosen, the following chapter describes how the selection of that method was made.
4 Research Method

When making an empirical investigation of a field such as comparing testing and inspections, there are a number of empirical investigation methods available. In this chapter, a selection of empirical investigation methods suitable for the aim of this thesis is described. The chapter also contains information on how these methods can be compared, and finally, an explanation as to how the selection of a method was made.

4.1 Methods Available

There are different ways of gathering information to reach the objectives of this thesis, but only one can be used because of the time limitations. The alternative method categories available here are empirical methods or purely theoretical studies. The purely theoretical, meaning the non-empirical approach of deduction and induction based on literature, have been discarded on account of literature shortage within the research area. The three empirical methods considered suitable for this empirical investigation are as follows:

Experimental methods aim to investigate "causal relationships using tests controlled by yourself" (Dawson, 2000).

Case Studies are used to study a single, typical situation (Dawson, 2000). When studying problems in software projects case studies are most often done through studying a particular project, within a particular company.

Survey techniques entail the investigation of an issue through interviews or questionnaires (Dawson, 2000).

An initial study of the different empirical methods showed that the survey approach would not be able to provide the data needed since it would require finding a set of software developers that already had an intimate knowledge and experience of both testing methods and inspection methods; hence it was discarded early on.

After having discarded the survey approach, the focus was put on the possibilities within the Experimental and Case-Study methods. The alternatives considered within these areas are listed below, together with relevant advantages and disadvantages, seen from the perspectives of result validity, result applicability and study feasibility.

4.1.1 A Large Experiment

In this method, a section of code is created. All defects are removed from it, followed by a re-introduction of different types of faults. After that, a large number of people is chosen, as many as 40, with a background in software engineering. These people will be divided into two groups, one inspection group and one testing group. Each group is divided into groups of, for example, four people, which will then try to find as many faults as possible in the code section, using their specified method, while being observed. The times noted are: start, when faults are found, and stop time. Where the faults were found is also documented.

Advantages: The experiment would contain a large number of subjects, making the result of the study more valid.

Disadvantages: The experimental subjects would be students, because it is not considered possible to find participants from industry within the time limit, and with no way to compensate the participants economically. And when having students, participants, most of them would not have worked within the industrial setting that the result is directed toward. Even finding 40 students is actually considered to fall outside the time frame of this thesis.
4.1.2 Small Experiment and Other’s Experiment Data
In this method, a search for similar experiments to “The Large Experiment” described above would be conducted, and get hold of the experiment data from those experiments. To complement that data, a small experiment would be conducted, using approximately ten people with a background in software engineering. The experiment would be conducted in the same way as the large experiment only with a slightly different focus than the existing experiments, from which the data was collected.

**Advantages:** The result could be more valid seen from the point of view of originality in the contribution.

**Disadvantages:** The result would be less valid than in the case of a large experiment, precisely because it is smaller and thus has fewer subjects. Something that could endanger the feasibility of this investigation is that the probability of getting hold of data sets from another experiment is low.

4.1.3 Medium Sized Experiment through Replication
This method consists of replicating an already completed experiment within the field, through copying that experiment’s approach at the lowest level possible. The experiment approach would be borrowed from the people who conducted the first experiment, in order to assure that the replication is done as thoroughly as possible. The main focus of the experiment would be the comparison of the two existing sets of data once the experiment has been conducted.

**Advantages:** The result of both the original, and the second experiment, would be more valid due to the increased number of subjects, as well as the diversity increase among the subjects. This investigation is also feasible within the time frame of the study.

**Disadvantages:** The experimental subjects would be students, because it is not considered possible to find participants from industry within the time limit, and with no way to compensate the participants economically. And when having student, participants, most of them would not have worked within the industrial setting that the result is directed toward.

4.1.4 Post Mortem of an Old Project
This method entails getting documents, from one or several old projects, and investigating the documented balance between dynamic testing methods, and static inspection methods, used.

**Advantages:** This kind of empirical investigation is based on real industrial projects; hence the result of such an investigation would be more valid in industrial settings than the results of an empirical experiment conducted with students as subjects. The result would also be generally applicable to software projects, since the diversity of the studied projects would be controlled.

**Disadvantages:** The feasibility of this study is somewhat limited since it is unusual that companies document the balance between the testing or inspection methods used - the most common approach is to only document the number of faults found within a given time frame. The feasibility is further limited by the time frame of the study to be conducted.
4.1.5 Case Study within a Real Company

If using this method, a number of companies would have to be contacted to get hold of a project which is entering its testing phase, and convince the company that has the appropriate project to participate in a case study. The project participants would then be educated in both methods, and use both methods as they see fit. Information would be collected through observing the project.

**Advantages:** The result would be based on reality within a company, and would hence be relevant to other companies conducting project of the same type as the one studied.

**Disadvantages:** The result would be typical, not general, because only one project can be studied within the time frame of the study. The feasibility is limited because of the difficulty of finding a project that is suitable for the study, both in methods, as well as lead time.

Having presented all of these alternatives, the problem of choosing one still remained. In order to make a good choice, the advantages and disadvantages have to be considered with the context of this thesis in mind. How this was done is described in the following section.

### 4.2 Choice of Method

To make the final selection of the empirical method, the advantages and disadvantages stated above have been summarized. What is important to keep in mind here, is that the feasibility analysis of each method always has precedence, since a disadvantage within the feasibility of an investigative method means that the method is not applicable to this study.

An advantage given within applicability of the result is important, in the sense that the original aims of this thesis is to answer a question on the balance between testing and inspections for projects within an industrial setting – but an advantage within result validity is more important, because a disadvantage in result validity will make an analysis of applicability of the result, completely redundant.

Based on the analysis above, it is concluded that the most important advantages are, ordered from high to low, as follows: Feasibility, result validity, and result applicability. The alternative that comes closest to having an advantage in all areas, is "The Medium sized Experiment through Replication", which has the advantages of feasibility and result validity. This is why the chosen empirical method for this study is "The Medium sized Experiment through Replication".

The selection of what experiment to replicate was limited because of two factors. The first limitation is that all the experiment artefacts would have to have been kept after the experiment. This limited the selection to fairly recent experiments. The second limitation, was there was a perceived need of having the experiment artefacts explained, which meant that a lot of problems would be eliminated if the search for an experiment was limited to experiments where a person, preferable an experiment designer, was available for questioning face to face. After some research, it was decided that the replication was to be done on the experiment done by Per Runeson and Anneliese Andrews at Washington State University in 2003 (Runeson and Andrews, 2003), because Per Runeson volunteered to be available for questioning and agreed to send the artefacts needed. The details of how the experiment was conducted, on a more detailed level, can be seen in the “The Experiment” section below.

Having chosen replicate the Runeson and Andrews experiment, it was important to form a view of what other, similar, experiments have been conducted within the field of comparing structured code testing and scenario based reading in inspection preparation, since these are the two methods compared in the Runeson and Andrews experiment. The next chapter summarizes the experiments done in comparing structured code testing and
inspections, prior to this experiment, since there did not seem to be any experiments available on the specific inspection preparation technique used in the original experiment for this replication.
5 RELATED WORKS

There have been several experiments over the years comparing code testing and code inspections in different ways. This chapter presents a brief description of related experiments, in a chronological order, to give the reader an idea of previous results as well as the status of this research field today.

5.1 Early Experiments
In 1976 Hetzel performed what seems to be the first experiment comparing some kind of code reading, functional testing and structural testing. His findings showed that testing had the highest detection rate. The detection rate average using any of the three techniques was 33 percent when the subjects worked individually, but when working in pairs the detection rate average got as high as 50 percent (Laitenberger, 1998).

Two years later, Myers performed an experiment with 59 subjects, of whom a clear majority (49) were employed as programmers or program testers, testing a small PL/I program (Myers, 1978). He compared three different methods of verification namely structural testing, functional testing and inspections.

There are a few things in the Myers experiment that could be questioned, such as that there were no time boundaries set for the testing methods, but for the inspection method there was a time boundary of 90 minutes. Further, all the subjects using structural testing as well as functional testing were working individually, while the inspections were conducted in teams of three subjects. These differences add uncertainty factors to the comparison of the results.

The result of the Myers’ experiment was that structural testing had the highest average detection rate - namely 38 percent, while the inspection method landed on an average detection rate of 30 percent (Myers, 1978).

In 1987 Basili and Selby set out to compare functional testing using equivalence partitioning, boundary value analysis, code reading (by stepwise abstraction) and structural testing. The result showed that the code reading technique detected slightly more defects (Laitenberger, 1998).

According to Laitenberger, the Basili and Selby experiment was replicated twice by Kamsties and Lott, where the first replication showed that inspection had the highest detection rate average, while the second replication showed that functional testing had the highest detection rate average.

The two earliest of these experiments show that testing had a higher detection rate than inspections, and two out of three of the later of these experiments show that inspection had a higher detection rate. The information available on the kind of inspection used in these experiments is very limited, so it is possible that the inspection methods used varies, hence there are uncertainties as to how relevant these early experiments really are to this experiment. To give a more updated view of the experimentation within the field of comparing testing and inspections, the next section describes experiments conducted after 1998.

5.2 Recent Experiments
In 1998 Laitenberger made an experiment encompassing 20 subjects, where he sequentially uses inspection and structural testing to verify a C-code module. He found inspections to be more effective than structural testing in defects found per hour, but it also had a higher detection rate than testing. This was the case both when inspecting and testing individually and in groups of two. Another conclusion that Laitenberger drew was that the two methods find the same kinds of defects, and thus does not complement each other (Laitenberger, 1998).
This experiment also raises questions as to uncertainty factors created through the use of sequential application of the methods with respect to learning and how the two methods can be compared when used sequentially on the same C-code.

Another experiment done in comparing testing and inspection was done by So, Cha, Shiemeall and Kwon in 2002, where they compared Fagan’s inspection method with a structural testing method. The result of this experiment is very interesting, in that it actually contradicts Laitenberger’s result in that they found performance to be better when using both methods together, because defects not found in testing were found in inspections and the other way around (So et al., 2002). They also found that testing has a higher detection rate than inspections.

The latest experiment done in the field of comparing testing and inspections, was done by Runeson and Andrews, and is the experiment that is replicated in this master thesis. The Runeson and Andrews experiment encompassed 31 subjects, and compared structural testing with the inspection meeting preparation using scenario-based reading. The 31 subjects were divided into two groups, which were balanced in randomization through the use of a pre-test where all subjects mainly stated their experiences of development, testing and inspections. The experiment was divided into two sessions, where both groups were verifying the same C program in parallel, but using different verification techniques. The groups switched techniques for the second session, and during the second session a second C program was verified, once again by both groups in parallel – as to counteract any uncertainty factors of learning when comparing the results of the groups.

One element that separates the Runeson and Andrews experiment from the other experiments within this field is that they chose to separate detected defects and isolated defects. Detected defect means that the defect has been detected but its origin has not been located in the code, and isolated defect means that the defect has been detected and the origin has been located in the code. The results of their experiment showed that testers detected more defects than inspections, but that inspectors were better at isolating defects they found. It was also indicated that testers found a wider range of defect types than inspectors. The defects detected per hour of verification also showed that testing was more efficient in finding defects than inspections.

If considering all experiments in this chapter, it seems that testing has a higher detection rate than inspections, but it is unclear which method is more efficient. Hopefully this experiment will contribute to answering that question, but first a thorough description of what this experiment actually entails is presented, in the following chapter, to avoid the kind of uncertainties of methods seen in the early experiments.
6 THE EXPERIMENT

The aim of this thesis is to investigate what defects can be detected using code testing respectively code inspection, and to see if one of the two methods is more efficient in finding defects than the other. This chapter shows how these aims were to be achieved, through describing the planning, execution and analysis of the experiment replication.

The experiment design was largely built on the original experiment design, created by Per Runeson and Anneliese Andrews. The original experiment contains much like other experiments, three main phases:

1. Preparation and Planning
2. Experiment
3. Experiment Result Analysis

The preparation phase aims at supplying the experiment with information for group divisions, and makes sure that all experiment participants have understood how to use the experiment methods investigated.

6.1 Experiment Preparation

The preparation phase consisted of preparing and sending out a pre-test with questions on the participants’ experience level. The questions can be viewed in the Pre-test of Appendix B. To make sure that the participants understood how to use the experiment artifacts and methods, two homework assignments were sent out as well.

The participants that were recruited for this experiment was master year students at Blekinge Institute of Technology, and were hence the classmates of the experiment conductor. Their motivation for agreeing to sign on for doing the experiment, were that they knew the experiment conductor.

6.2 Experiment Planning

This section describes how the experiment was planned to be, and thus should have been, executed.

The participants were to be divided into two groups based on their knowledge and experience within the software engineering area, but the people in both groups were to work individually.

The pre-test was constructed so that the participants had four options of experience length for each question of experience. The four options were:

1. No experience,
2. Experience from courses,
3. Less than 3 years industrial experience and
4. More than 3 years of industrial experience.

The questions all regarded experience in development, programming, inspections and testing, and can all be found in the Pre-test of Appendix B.

The option numbers on all experience questions were then added to give each experience stated a randomized importance in the group division, and then the sum of the experience questions worked as a group division factor, so that the sum of all experiences within both groups were randomized but balanced.

The experiment was to be executed so the order, in which the techniques were used, was reversed between sessions - meaning that one group would start off using the inspection method, and the other group would start off using the test method, then both groups would switch techniques before entering the second verification session. The two C programs to be
verified during the sessions were both collected from Watts Humphreys book on the Personal Software Process (Humphrey, 1995), and the same code was to be verified by both groups at the same time – to remove the uncertainty factor of students from different groups talking to each other about the code during the recess that separates the two sessions. A visualization of how the experiment was to be conducted is found in Table 6.2.1 below:

<table>
<thead>
<tr>
<th></th>
<th>Group 1</th>
<th>Group 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Session 1 - Counter Program verification</td>
<td>Inspection</td>
<td>testing</td>
</tr>
<tr>
<td>Session 2 - Correlation Program verification</td>
<td>Testing</td>
<td>Inspection</td>
</tr>
</tbody>
</table>

Table 6.2.1

One of the biggest changes to the original Runeson and Andrews experiment, an addition of a C code review checklist, in order to keep the new experiment comparable to the original one. This checklist was purposely created for the C language by Per Runeson, and is based on the checklist described by Watts Humphrey in his book on the Personal Software Process (Humphrey, 1995). The reason why the original experiment was not extended through having the actual inspection meeting instead of the original approach of inspection meeting preparation, is twofold: first of all, the results from the replication would not be directly comparable to the original results, because the inspection meeting preparation is done individually while the actual inspection meeting is done in groups, and secondly, there exists material that presents the meeting as having questionable benefits to the verification process – a few such examples are discussed in the “3.3.5 Inspection without Meeting” section.

The information collected during both sessions, from all participants, was to be:

- time for start of verification,
- time it took to read all material supplied
- time to generate test data ( test cases from the use cases supplied )
- what defects each participant has found,
- where the defect was found
- time when each defect was found,
- what category each defect was considered to belong to,
- the level of confidence that each defect the individual has found is actually a defect,
- time to fix the defects found ( in case of testing )

After the experiment would have been executed, all participants would have filled out a post-test. The post test would primarily aim at finding out how the participants were thinking during the experiment execution, what artifacts they had most use of and what they learned from the experiment and preparation. Of course, this section describes what the original experiment plan looked like, but along the way, unforeseen events made it necessary to change the plan. The next section describes how this was done.

6.3 Experiment Re-planning

This section describes how unforeseen events forced the original experiment planning to be revised in two major ways, the first being the moving of a deadline, which caused the
correction of the homework assignments to be cut back, and the second was an addition to the experiment introduction held before the first experiment session.

Four days before the experiment, when the deadline for the homework assignments and the pre-test passed, only two of 21 participants had handed in their homework. Due to this, a lot of time had to be spent trying to convince as many of the original participants to hand in their homework and not drop out of the experiment. This was done through spending a day helping anyone, still willing to participate, with understanding the homework assignments, and then prolonging the deadline for the homework assignments. This in turn resulted in a shallow correction of the homework assignments that basically just allowed for a superficial assurance that the participants had understood how to use the tools.

The correction of the homework assignments showed some discrepancies in the participants’ derivation of scenarios from use cases. This issue was handled by adding information to the experiment introduction presentation, on how to derive scenarios from use cases.

After the plan had been revised to fit the new needs of the experiment, the day for the experiment execution had arrived. To give a good view of all factors that may have influenced the results, the experiment execution is described in detail in the following section.

6.4 Experiment Execution

This section describes what the actual execution of the experiment looked like – what happened, how it happened, what went wrong and what went according to plan. In short, everything went according to plan, except for the numerous drop-outs among the participants.

The two preparing homework assignments were sent out along with the pre-test via e-mail to all participants 19 days before the experiment was executed. The homework assignments and the pre-test were handed in one day before the experiment. The homework correction indicated that all participants had understood the use of both techniques that they would be using in the experiment, but there were some minor differences in viewing the derivation of scenarios from the use cases provided.

13 participants handed in their pre-tests in time, and based on the pre-test the participants were divided into two groups, group A and B, on the day of the experiment, before the first experiment session, so that such uncertainty factors as planned participant absence due to sickness or simply drop-out, could be balanced out.

Eight participants showed up on the day of the experiment, so therefore they were divided into two groups of four. The mean calculated for each group, based on the experience stated in the pre-test, was 23.25 for group A, and 23.50 for group B. One member of group A arrived at the experiment two hours later than the rest of group A – this was however notified to the experiment executor the week before the experiment and was thus expected.

The experiment started with an introduction meeting, where the discrepancies in the derivation of scenarios from use cases in the homework assignments were addressed through an example of scenario derivation. All participants had previously received a general schedule for the experiment sessions, and information regarding the background of the experiment and the plans for its result.

After the introduction, the two groups were separated. Group B, was relocated to a computer room, where they were to conduct verification through unit testing. All participants had the options of either using Microsoft visual studio for testing, or using a Secure Shell client to connect to a GNU test environment.

Group A stayed in the room where the introduction had taken place, where they conducted verification through inspection using scenario based reading. Session one lasted for two hours, and was followed by a 30 minute break.

For session two, group A was placed in the computer room to conduct verification through unit testing, and group B was placed in the introductory room, and conducted
verification through inspection using scenario based reading. The second session also contained the post-test questionnaire, which was handed in along with each participants session result.

Questions on the contents of the experiment were directed to the experiment executor directly in the inspection-room, and during a brief question session in the test-room. The questions and answers were only heard by the individual participant and the experiment executor except for one instance, where a misunderstanding had to be cleared up.

Both experiment rooms were being monitored throughout both sessions, to make sure that there was no inter-participant-communication during the experiment, and all participants were free to hand in their session result and leave the experiment room at a time of their own choice.

That is the account of what the experiment execution looked like, but just as the experiment execution had an original plan, so did the analysis of the result from the experiment execution. To give an accurate description of what the analysis plan ended up looking like, the original plan is first described in the following section.

6.5 Original Experiment Analysis
This section describes the analysis as it was done in the original experiment (Runeson and Andrews, 2003). The plans for this experiment were to extend the original analysis methods, described below, with a comparison between the original experiment group and this experiment group.

Data collection
The data to be collected was that of defects found per time unit i.e. efficiency and defects found in total, i.e. effectiveness.

Data Purification
First, the data was to be purified, through isolating outliers with respect to time consumption and Pre-test stated skills and experiences.

Once the data would have been purified, the median, mean and standard deviation of the following values would have been calculated: number of defects detected, number of isolated defects, detected defects per hour, detected defects per hour, defects and isolated defects per hour. The Wilcoxon signed rank test would also have been performed to test the hypotheses, which were directly used from the original experiment (Runeson and Andrews, 2003).

Defect Distribution
After testing the hypothesis, the distribution of defects found by different techniques would have been analyzed by using bar plots and \( \chi^2 \)–test. The isolation rate of the two techniques, as well as the defect severity in defects found by the two techniques would also have been analyzed, though the isolation rate would have been shown as box plots instead of bar plots.

Post-test Analysis
The contents of the post-test would also have been analyzed, but only to give a sense of the opinions within the group.

Due to the high drop-out rate among the participants of this experiment, the analysis had to be changed as well. How this affected the analysis and how the analysis was changed, is described in the section below.

6.6 Analysis Re-planning
This section describes how unforeseen events forced the original experiment analysis plan to be revised. Due to the low turn-up rate among the signed-on participants the result of this experiment had too low statistical power of the participant group to perform the statistical analysis used in the original experiment. The statistical analysis for this experiment result
was therefore re-planned and confined to comparing the analysis of the experiment groups from this experiment and the experiment group from the experiment conducted by Runeson and Andrews at Washington State University in 2003. This will be done through calculating median, mean and standard deviation for defect detection and defect isolation in both testing and inspections.

These figures indicate if the result of this experiment is close to the result of the original Runeson and Andrews experiment.

Since the quantitative analysis in the form of a statistical analysis was out of the question due to the low statistical power of the group, a new analysis was designed with the intent of making the result of this experiment add to the body of knowledge in comparing testing and inspections, even though it had such low statistical power, through finding patterns in this small group, that would be applicable to the original experiment group.

There are essentially two ways of analyzing the result of an experiment: the quantitative way, and the qualitative way, so this basically left the option of the qualitative investigation for the design of the analysis phase.

A qualitative analysis is done when the issue being investigated is how people reason or react, or to separate various ways of acting. Understanding behavior and finding patterns in the reasoning of the group is in other words a key element in qualitative investigations. There are five common traditions of qualitative research (Creswell, 1998):

- **Biography** – creating a study based on stories and specific events with focus on an individual, and then viewing these stories and specific events from different viewpoints and against different backgrounds.
- **Phenomenology** – the study of a single phenomenon, through studying a number of descriptions of the phenomenon, extracting significant statements and compiles these statements into a summary of the phenomenon.
- **Grounded theory** – the aim of the grounded theory approach is to generate a theory, through creating a well described procedure of result analysis along with a visual model of the result.
- **Ethnography** – a highly detailed study of roles and behaviours related to the culture the role or behaviour is found in.
- **Case study** – the observation and interpretation of a phenomenon with as little interaction as possible.

There are different ways to extract data in a qualitative investigation, regardless of tradition. You can for example choose to: only observe, have focus groups, conduct interviews with an individual or in groups, or combine these data extraction methods to increase the validity of the result, through making sure that the data sets indicate the same result regardless of collection method (Creswell, 1998).

Choosing a tradition is not very difficult, since they all have such clear aims. The tradition with the goal type closest to that of examining an event such as the experiment is that of phenomenology. To choose the method of data extraction, the advantages and disadvantages of each of the collection methods were considered. Observation alone was ruled out from the beginning, due to the fact that drop-out rate became known only the day before the experiment, so there was no time to prepare. The focus groups could have been an option, but the problem with focus groups is that one should follow the participants, as a group, for a longer period of time, and the time boundary for the thesis did not allow for that option. Then there is the risk, mentioned by Trost (Trost, 1993), of having one or two individuals dominate the discussions if meeting the participants in groups. Another problem with using a focus group in this particular case is that the information that was going to be discussed was of a sensitive nature, meaning that perhaps people would become mentally hostile when feeling exposed to the entire group. This reasoning also excluded the option of a group interview. The conclusion is that the most preferable method of data collection is individual interviews.
The design of the interview questions was open ended – to direct the answers of the participants as little as possible (Trost, 1993). The issues further investigated through the interview questions were mainly: the reasons for why the participants missed defects, reasons for why the participants found false positives, a further investigation of their experience with C and C++ and finally an investigation of their testing and inspection experiences. False positives are statements that the participants have identified as defects, even though they are not defects. The interview questions can be viewed in full in the qualitative questions section of Appendix B.

The plans for the result of the interviews were to find similarities in the answers to why the participants did not find certain defects, and why they found false positives. Where these answers turned up independently in four cases or more, a pattern was considered found. This limit was chosen because it represents half the group of participants.

A serious problem, that could be labeled as a validity question, is that the questions in this investigation were retrospective – meaning that the questions were asked at a point in time separated from the actual event (Trost, 1993). Factors such as memory weaknesses or interpretations of what happened are likely to have occurred between the time of the experiment and the time of the actual interviews – which were conducted about two weeks after the experiment. One technique that can be used to improve the validity of the answers is to somehow stimulate the memories connected to the event of the experiment session. This technique was applied through bringing the material each participant had handed in at the end of the experiment to the interview, and through a review of each error, as viewed in the code. The thought was that seeing the artifacts could relate to the memory of the event (Trost, 1993).

If the quantitative comparison between this entire experiment group and the original experiment group indicated that this experiment group is comparable to a subset of the original experiment group, the patterns found in the qualitative investigation of this experiment group could perhaps, with caution, be applied to at least part of the original experiment group – and thus create an opportunity to draw conclusions between and within both groups based on that application.

Hopefully, this chapter has been able to provide an overview of the design and execution of the experiment replication, so it is time to move on to the result and the actual analysis thereof. The following chapter therefore presents the result analysis, together with the proper validity issues that follows from factors such as methods used in the execution.


7 Research Results

This section describes what the result of the analysis was. It is divided into three major sections, where the first one discusses the validity of the result, the second compares this experiment group with the original experiment group, to see what the connection points between the groups are and how they relate to each other. The last section describes the patterns found in the qualitative analysis of this experiment group.

7.1 Result Validity

When making experiments, there are a number of uncertainty factors that can influence the result, and in order to draw relevant conclusions from the result of this experiment, all results should be viewed with the particular uncertainty factors for this experiment in mind. In this section, the validity of the result is discussed. There are four different kinds of validity to consider in all experiments (Wohlin et al., 2000):

- **Conclusion validity**: Making sure that there is a statistically significant relationship between treatment, in this case the verification method, and outcome which in this case means the experiment result.
- **Internal validity**: Making sure that the relation between the cause and effect is causal, which in the case of this experiment means assuring that the verification technique is responsible for the result.
- **Construct validity**: Making sure that the technique is applied as described, and that the result is equivalent to the result expected, in this case assuring that the technique and result both are connecting the theory of how the experiment construct should work, with the way the experiment construct actually works.
- **External validity**: Is the cause and effect separable from the environment dependent variables so that it can be applied to situations outside of the experiment scope. In this experiment the issue would be if the result is applicable in industry or to the result of the original experiment.

7.1.1 In both the Original Experiment and Replication

In the original experiment a lot of potential validity issues were ruled out, but there were a few validity issues that even though they were counter-acted, they were not eliminated, and these need to be considered for the replication experiment as well.

**Conclusion validity threats:**
- The human judgement in interpreting what defects were detected respectively isolated.
- That there is no way to check if the subjects are really using the techniques specified when finding faults.
- The risk of random heterogeneity in the groups

**Internal validity threats:**
- Learning caused by the specific order in which the techniques are applied.

**Construct validity threats:**
- Social threats – like hypothesis guessing among the subjects may influence the way they perform.
External validity threats:

- That the subjects are students influences the results applicability in industry.
- The programs from an academic environment, which might make them different from industrial programs.

These problems all apply to both the original experiment and the experiment replication of this thesis, but there are validity issues that are new to the replication experiment.

7.1.2 Validity in the Replication

There are a few additional validity issues that are related to the replication experiment only, and to make sure that none are missed, the comprehensive lists from Wohlin et al. (2000) were used to search for additional validity issues:

- The conclusion validity threat of low statistical power is a problem in this group, meaning that the group is so small, in particularly when it is divided into two subgroups.
- The construct validity threat of the checklist not having been tested in a pilot-study.

These specific validity issues need to be kept in mind when comparing this experiment group with the original experiment group in particularly, because they do separate the groups. However, there are a number of values and factors connecting the two groups as well, and what these are, and how they connect the original group and the replication group, can be seen in the section that follows.

7.2 Statistical Comparison of the Two Groups

In this section the result of this experiment group has been summarized for both methods with regards to (Table 8.2.1):

- defect detection, meaning defects detected but where the origin of the defect is or is not found
- defect isolation, meaning that the defect origin was isolated in the code,
- detection efficiency, meaning defects detected per hour
- isolation efficiency, meaning defects isolated per hour.

This experiment showed that inspection had a higher detection defect rate than testing, but that testing was more efficient, meaning found more defects per hour, in both detection and isolation of defects.

In the Runeson and Andrews experiment, the detection rate was found to be an average of 33 percent. This group had, as a whole, a significantly lower detection rate of only 22 percent.

That these groups are not comparable with regards to average values was expected because of the many uncertainty factors stated, in the section above, on result validity. A very important question in regards to the result of the replication group, is weather this experiment group is comparable to any subset of the original experiment group. Since the raw data is not available for the original experiment group, the range of isolation and detection rate in the original experiment is best described by taking the mean value, and adding and subtracting the standard deviation from it. This forms ranges where most of the isolation and detection rates lie. When doing the same to the replication experiment (raw data can be found in Tables 1-4 in Appendix A), and compare the ranges, found in Table 2.8.1, we find that they overlap quite nicely.

Note that in inspection, it was possible to separate the time of checklist inspection from the inspection done by scenario based reading. This separation was not possible within testing, because the checklist was used, if needed, to isolate each defect detected in the code. Since all participants stated that they had no use of the checklist in testing, the defect
detection and defect isolation (effectiveness) rates are still comparable to the Runeson and Andrews experiment, but the efficiency (defects detected or isolated per hour) of testing is not comparable to the Runeson and Andrews experiment, since the time consumption of attempts to use the checklist is unknown.

<table>
<thead>
<tr>
<th>This Experiment (replication) – mean range</th>
<th>Original Experiment – mean range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inspection</td>
<td>Testing</td>
</tr>
<tr>
<td>Defect detection</td>
<td>-0.50 – 4.00</td>
</tr>
<tr>
<td>Defect Isolation</td>
<td>-0.62 – 3.38</td>
</tr>
<tr>
<td>Defects detected per hour</td>
<td>-0.29 – 2.79</td>
</tr>
<tr>
<td>Defects isolated per hour</td>
<td>- 0.47 – 2.41</td>
</tr>
</tbody>
</table>

Table 8.2.1

The reason that the lower parts of the ranges in the columns for this experiment, in Table 2.8.1, are lower is that more participants found zero defects in this experiment, as can be seen in the two last columns of Table 6 in Appendix A. This deviation can partly be understood by the logical deviations of participant A and D in column 3 of Table 6 in Appendix A. This deviation indicates that these two participants had considerable more time since their last programming experience than the rest of the group as a whole. In Table 6 Appendix A, the deviation of participant D is logical, because of the low programming experience in total, but the deviation of participant A indicates that perhaps the time since the last programming experience affects the result more than hours of programming experience, after a certain point.

Table 8.2.2 clearly indicates that what separates this experiment group from the original group is that a lot of these participants found no defects at all. However, the groups are still comparable in average range – both with regard to detection and isolation, and when considering both effectiveness and efficiency (of inspection but not testing) of the verification.

The similarity between the two groups in regards to time consumption is not as clear as that of the defect rate. Based on the raw data in Tables 1 and 4 regarding time consumption an average time consumption of 68.5 minutes was derived for the scenario based inspection. This is significantly lower than the comparable number from the Runeson and Andrews experiment which is 103.8 minutes. The time consumption for the testing is not directly comparable to the Runeson and Andrews values, just as in the case of the efficiency, because the time spent using the checklist could not be separated from the rest of the testing session.

The similarities between the groups continue through to detected and isolated defects grouped per program. The mean values found in Table 10 Appendix A clearly shows that the trend of detecting and isolating slightly fewer defects on average in the counter program than in the correlation program is the same in the replication experiment as in the original experiment. What could be said about differences to the original experiment result compared to Table 10 in Appendix A, is that the differences between the average number of isolated defects and detected defects are slightly larger in the replication experiment.

Having concluded that the groups do connect on several points in the descriptive statistical analysis, it is time to analyze patterns found within the replication group only. The next section describes the patterns found in types of defects, methods and experiences.

7.2.1 Types of Defects
The classification of defects can be found in Table 8 and Table 9 in Appendix A, and one correlation found between the defects not found, was that only one of three class 5 defects was detected at all, and that one was detected using testing by one out of eight participants
7 out of 8 thought that functional defects were easier to detect using testing. This was concluded from the answers to Post-test question 10 in Appendix B, and the result from the original experiment group also indicated that participants thought functional defects were more easily detected through testing.

7.2.2 Method Related Patterns

7 out of 8 participants preferred testing over inspections “to identify that there is a defect” in the software (see Appendix B, Post-test question 13). This pattern is also found in the original experiment group. Testing was said to include both structured testing as well as individual ad-hoc debugging according to answers given to question 12 in the qualitative questions of Appendix B. The reason stated by 5 out of 8 participants was that testing is more fun. 5 out of 8 stated that testing was more interesting because one could more easily see that the program was correct or incorrect. These last statements were found in answers to question 11 of the qualitative questions in Appendix B, and complement the results from the original experiment group by providing an answer to why testing was preferred.

All participants that found two defects or less in their inspection said that they experienced information overload when trying to inspect what they found to be too much code for the time frame provided. This parallel was drawn between Tables 1 and 4 in Appendix A, and the answers found to question 14 of the qualitative questions in Appendix B. This pattern is probably not directly applicable to the original experiment group, because the existence of the checklist may have affected this factor as described in the following pattern.

The time consumption variation in the scenario based reading inspection, is hard to explain, but the indications from the qualitative investigation (question 13 in the qualitative questions of Appendix B) suggests that the inspection using scenario based reading was rushed through because the presence of the checklist was known.

7.2.3 Experience Related Patterns

Five participants found two defects or less using inspections and they all independently stated that the lack of knowledge of the C language was a contributing reason to why they did not find some defects (question 14 of the qualitative questions in Appendix B). This is an interesting pattern, because it is not consistent with the experience of C stated by each participant in the pre-test, which indicates that the amount of C experience did not affect the result, as can be seen in Table 7.2.3.1.

<table>
<thead>
<tr>
<th>Participant ID</th>
<th>Hours of C programming experience</th>
<th>Defects found in inspections</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1200</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>400</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>760</td>
<td>5</td>
</tr>
<tr>
<td>D</td>
<td>65</td>
<td>0</td>
</tr>
<tr>
<td>E</td>
<td>105</td>
<td>1</td>
</tr>
<tr>
<td>F</td>
<td>640</td>
<td>4</td>
</tr>
<tr>
<td>G</td>
<td>120</td>
<td>0</td>
</tr>
<tr>
<td>H</td>
<td>0</td>
<td>5</td>
</tr>
</tbody>
</table>

Table 7.2.3.1

If removing the outliers (participants A and D) based on last C or C++ programming experience, the results look like they form a pattern except for participant H, but on the other hand this deviation can not be explained, and thus the pattern must be disregarded.

All participants (three) that found four defects or more using both methods, all had more than 6200 hours of coding experience (6240-7280), while those who found two defects or
less (five) using both methods, had 5200 hours of coding experience or less (200-5200). This can be seen in Table 6 in Appendix A, and is a very interesting pattern that seems to show that there is a lower experience pre-requisite limit for successful verification both through using testing and inspections. This is definitely something that should be investigated further.

The amount of false positives is slightly larger in inspections than in testing, as can be seen in Tables 11 and 12 in Appendix A, but a large proportion of the false positives in inspection consist of complaints on the code standard, the looks of the code and lack of file checking routines within the programs. Code standard, meaning not initialized variables, were responsible for 9 out of 36 false positives. Complaints that the code contained ugly indentation and superfluous spaces were responsible for 7 out of 36 false positives. File checking was responsible for 5 of 36 false positives. 11 false positives could be explained as the participant having misunderstood, or misread, the code, and the last 4 were clear cases of the participant having misread the specification. 7 out of 8 participants had at least one real false positive in inspection. This pattern may indicate that in a group with a common code standard to work with the amount of false positives would be significantly lower than in the replication experiment.

That was the compiled and analyzed result of the empirical investigation of this thesis, but there is yet another dimension of lessons learned connected to the investigation, and this is the lessons learned from the work with this thesis as a project in its own right. To add this shell of premises for the experiment replication, the following chapter briefly describes some features of this master thesis project.
8 LESSONS LEARNED FROM THIS THESIS

With the hope of providing the reader with some ideas of the most prominent premises for the experiment replication, this chapter describes the lessons learned in different areas of this experiment. This is done with the aim of separating the thesis as an educational form, from the investigation, meaning the experiment replication, this form of education entails.

8.1 Experiment Planning

The biggest lesson learned from planning the experiment, is the understanding of why these kinds of experiments keep reappearing with a common subset of validity threats, namely that of using smaller software programs than most encountered programs in industry, or using students for participants in experiment that could give answers that the industry needs. The reason is quite simply that of motivation. It is very hard to get hold of people to participate in an experiment and especially in an extensive experiment. Of course it is preferable to have large programs – meaning extensive experiment, and people from industry to participate in the experiments, but most people work for a living, and do not see participating in an extensive experiment during their precious free time as something positive. The lesson learned is thereby that it is better to conduct investigations using small programs and students as participants than to not conduct investigations at all. What can be questioned are perhaps the methods we choose to perform these investigations through.

8.2 Experiment Replication

The replication element of this experiment is connected to a set of lessons learned, because the original experiment contains a lot of uncertainty factors that may influence the relationship between cause and effect of the verification techniques, and the replication just adds to the amount of uncertainty factors in that not everything can be replicated. Further, the replication extended the original experiment in such a way, that in the end, the extended parts’ effect on the result was hard to separate from the original parts’ effect on the result. One example of an uncertainty factor due to an un-replicable condition was that of not being able to replicate the motivation for why people should participate in the experiment. The original experiment had the motivation of being one of two alternatives to a mandatory element of a university course.

Another element of difficulty in replication is the interpretation of the artefacts the experiment contained. In this case, one of the creators of the original experiment was available for questioning, and above all, for assuring that the perception of what the original artefacts and experiment entailed was correct, so the replication did not alter the central parts of the experiment – which could have rendered the result useless for comparison between the replication participant group and the original participant group. In other words – the artefacts need to be accompanied by explanations of their intended use and the meaning of their contents.

8.3 Experiment Result

The result of this experiment contained a lot of surprises, such as false positives not being as simple as someone misreading the code or specification. Another surprise was how impaired ones ability to successfully verify code is by the time that has passed since the last coding experience. These surprises all contributed to an understanding of the expectations that build up along with reading the results of other people’s experiments. It truly gives a new dimension to the need of stepping back, and reclaiming the role of researcher, instead of investing in expectations of the result.
9 CONCLUSIONS

The aim of this thesis was twofold: to find out if the two methods had different sets of defect types in defects found, and if the result would benefit from a combination of the two methods. The result was, that little difference could be seen in comparing the sets of types of defects found, and hence there is little evidence in the result of this thesis, that the methods would benefit from being combined.

The conclusion drawn from the comparison between the groups is that the replication of the central parts of the experiment was a success, and this is largely attributed to the presence of one of the original experiment designers.

There are a few conclusions that can be drawn from this experiment alone, such as that it is not a good idea to combine inspection using scenario-based reading with inspection using a checklist within the same two hour interval. Another conclusion of the same kind is that a lot of false positives are eliminated if using a common code standard when implementing software. One of the most exiting indications in the result from the replication group is the possible existence of a lower experience limit in verifying code, and hopefully someone will challenge this indication in the future.

As a whole, this master thesis has shown, that empirical investigations such as replications are hard to do, because of the need for artefacts to replicate, the problem in understanding the artefacts once they are found, problems with finding people to participate in the investigation if needed and last but not least, the problems of validity in the result. However, this thesis has also shown that there is value in these empirical investigations, and in particular, in replications, because with the increasing number of times investigations, and especially the same investigations, are conducted, the validity of their collective results increases.

For future work, it would be nice if someone investigated the indications of a lower experience limit for verifying code. Another issue that would be nice to see investigated in the future, is the correlation between last general programming experience, the amount of programming experience and code verification performance. On a more general level, it would be nice if the issue of replication prerequisites were investigated further, to try and present a basic level of when a replication attempt can be treated as a success. This would simplify the interpretation of the results of replications enormously.
10 \hspace{1cm} \textbf{TERMINOLOGY}

\begin{itemize}
  \item \textbf{Bugs} \hspace{1cm} \text{Faults (Fenton and Pfleeger, 1997)}
  \item \textbf{Defects} \hspace{1cm} \text{Faults and Failures (Fenton and Pfleeger, 1997)}
  \item \textbf{Error} \hspace{1cm} \text{When humans make errors (Fenton and Pfleeger, 1997)}
  \item \textbf{Failure} \hspace{1cm} \text{When a system does not behave in the way it is required to (Fenton and Pfleeger, 1997)}
  \item \textbf{False positive} \hspace{1cm} \text{Where a defect has been marked as found in a place where there is no defect present.}
  \item \textbf{Fault} \hspace{1cm} \text{When a human error results in a mistake in some software (Fenton and Pfleeger, 1997)}
\end{itemize}
11 ACKNOWLEDGEMENTS

Special thanks are hereby given to Per Runeson for supplying this experiment with the material needed for replication, and for explaining the material and thus assuring that this replication was as good as could be. Special thanks are also given to Anneliese Andrews, for supplying documents that were both lost and found.
12 REFERENCES


Lawrence G Votta Jr (1993): ‘Does Every Inspection Need a Meeting?’, ACM.


## APPENDIX A

### Group 1 – Counter inspection (without checklist)

<table>
<thead>
<tr>
<th>ID</th>
<th>Time</th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
<th>C4</th>
<th>C5</th>
<th>C6</th>
<th>C7</th>
<th>C8</th>
<th>C9</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>58</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>42</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>99</td>
<td>I</td>
<td>D</td>
<td>D</td>
<td>I</td>
<td>I</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>95</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1 – Defects detected (I and D) and isolated (I) in scenario-based reading inspection meeting preparation, session 1.

### Group 2 – Counter Testing

<table>
<thead>
<tr>
<th>ID</th>
<th>Time</th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
<th>C4</th>
<th>C5</th>
<th>C6</th>
<th>C7</th>
<th>C8</th>
<th>C9</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>120</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>120</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>D</td>
</tr>
<tr>
<td>G</td>
<td>104</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>115</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>D</td>
</tr>
</tbody>
</table>

Table 2 – Defects detected (I and D) and isolated (I) in testing, session 1.

### Group 1 – Correlation Testing

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>89</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>62</td>
<td>I</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>87</td>
<td>I</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>90</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3 – Defects detected (I and D) and isolated (I) in testing, session 2.

### Group 2 – Correlation Inspection (without checklist)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>58</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>83</td>
<td>I</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>33</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>80</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4 – Defects detected (I and D) and isolated (I) in scenario-based reading inspection meeting preparation, session 2.

<table>
<thead>
<tr>
<th>Participant ID</th>
<th>Time spent on checklist in inspection</th>
<th>Isolated defects using checklist</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>45</td>
<td>-</td>
</tr>
<tr>
<td>B</td>
<td>31</td>
<td>-</td>
</tr>
<tr>
<td>C</td>
<td>13</td>
<td>-</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>E</td>
<td>49</td>
<td>-</td>
</tr>
<tr>
<td>F</td>
<td>20</td>
<td>corr7</td>
</tr>
<tr>
<td>G</td>
<td>32</td>
<td>corr2</td>
</tr>
<tr>
<td>H</td>
<td>11</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 5 – Results from the checklist-based inspection meeting preparation
<table>
<thead>
<tr>
<th>Participant ID</th>
<th>Hours of programming experience</th>
<th>Years since last programming experience</th>
<th>Total amount of defects detected in inspection</th>
<th>Total amount of defects detected in testing</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>5200</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>3000</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>C</td>
<td>6240</td>
<td>1</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>D</td>
<td>200</td>
<td>7</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>E</td>
<td>1900</td>
<td>0.5</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>F</td>
<td>6400</td>
<td>0.5</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>G</td>
<td>1280</td>
<td>1.5</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>H</td>
<td>7200</td>
<td>1</td>
<td>5</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 6 – Comparing programming experience and verification results

<table>
<thead>
<tr>
<th>Fault Class</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Initialization</td>
<td>The fault is due to wrong or missing variable initialization.</td>
</tr>
<tr>
<td>2</td>
<td>Computation</td>
<td>The fault is related to computational issues such as calculations that assign a variable a wrong value.</td>
</tr>
<tr>
<td>3</td>
<td>Control</td>
<td>The fault is related to the control flow of the program, e.g. an incorrect predicate in an if-statement.</td>
</tr>
<tr>
<td>4</td>
<td>Interface</td>
<td>The fault is related to an interface, e.g. wrong parameters are passed in a function call.</td>
</tr>
<tr>
<td>5</td>
<td>Data</td>
<td>Faults related to structure or content of a data structure, e.g. indexing an array.</td>
</tr>
<tr>
<td>6</td>
<td>Cosmetic</td>
<td>Faults which are cosmetic only, e.g. a spelling mistake in an error message.</td>
</tr>
</tbody>
</table>

Table 7 – Fault Classification

<table>
<thead>
<tr>
<th>Counter Defect ID</th>
<th>Fault Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count1</td>
<td>3</td>
</tr>
<tr>
<td>Count2</td>
<td>3</td>
</tr>
<tr>
<td>Count3</td>
<td>4</td>
</tr>
<tr>
<td>Count4</td>
<td>3</td>
</tr>
<tr>
<td>Count5</td>
<td>3</td>
</tr>
<tr>
<td>Count6</td>
<td>3</td>
</tr>
<tr>
<td>Count7</td>
<td>6</td>
</tr>
<tr>
<td>Count8</td>
<td>5</td>
</tr>
<tr>
<td>Count9</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 8 – Defect classification of the Counter program defects
### Table 9 – Defect classification of the Correlation program defects

<table>
<thead>
<tr>
<th>Correlation Defect ID</th>
<th>Fault Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Corr1</td>
<td>1</td>
</tr>
<tr>
<td>Corr2</td>
<td>2</td>
</tr>
<tr>
<td>Corr3</td>
<td>2</td>
</tr>
<tr>
<td>Corr4</td>
<td>2</td>
</tr>
<tr>
<td>Corr5</td>
<td>5</td>
</tr>
<tr>
<td>Corr6</td>
<td>2</td>
</tr>
<tr>
<td>Corr7</td>
<td>6</td>
</tr>
<tr>
<td>Corr8</td>
<td>6</td>
</tr>
<tr>
<td>Corr9</td>
<td>5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Median Count</th>
<th>Mean Count</th>
<th>Std. deviation Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Defects detected</td>
<td>0</td>
<td>1.5</td>
<td>1.63</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.63</td>
<td>2.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2.00</td>
<td>2.26</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1.85</td>
</tr>
<tr>
<td>Defects isolated</td>
<td>0</td>
<td>1.5</td>
<td>1.13</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.13</td>
<td>2.13</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2.13</td>
<td>1.55</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1.64</td>
</tr>
</tbody>
</table>

Table 10 – Detected and isolated defects grouped by program.

### Table 11 – False positives in group 1 during both sessions

<table>
<thead>
<tr>
<th>Participant</th>
<th>Counter inspection</th>
<th>Correlation testing</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>4</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 11 – False positives in group 1 during both sessions

### Table 12 – False positives in group 2 during both sessions

<table>
<thead>
<tr>
<th>Participant</th>
<th>Counter testing</th>
<th>Correlation inspection</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>F</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>G</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>H</td>
<td>4</td>
<td>6</td>
</tr>
</tbody>
</table>

Table 12 – False positives in group 2 during both sessions
### APPENDIX B

#### Pre-test

<table>
<thead>
<tr>
<th>ID</th>
<th>Question</th>
<th>Alternatives</th>
</tr>
</thead>
</table>
| 1  | What courses have you taken/attended in computer science and software engineering? | a) Program design and development (BTH: Object oriented systems development)  
b) A course in data structures (BTH: Computer structures and Algorithms in JAVA)  
c) course in programming tools  
d) course in automata (BTH: Automata and formal languages)  
e) course in Systems programming  
f) course in Compiler design (BTH: Compiler technique)  
g) course in database systems  
h) course in database techniques (BTH: Database techniques)  
i) course in Software engineering principles (BTH: Small software engineering project and Large software engineering project)  
j) course in program language design (Such as the 10 point JAVA course held at BTH during the first semester)  
k) A course in advanced data structures  
l) A course in design and analysis of algorithms (BTH: Computer structures and Algorithms in JAVA)  
m) A course in software verification and validation (BTH: Software Verification and Validation) |
| 2  | How long has it been since you coded something in C or C++?                 | None                                                                                                                                         |
| 3  | When did you begin your studies in software engineering at university level? | None                                                                                                                                         |
| 4  | Software Engineering “State of the art knowledge”                        | 1) No knowledge  
2) Studied courses  
3) Industrial experience ( less than 3 years )  
4) Long industrial experience ( more than 3 years ) |
| 5  | Software Engineering practice                                            | 1) No knowledge  
2) Studied courses  
3) Industrial experience ( less than 3 years )  
4) Long industrial experience ( more than 3 years ) |
| 6  | C coding experience                                                       | 1) No knowledge  
2) Studied courses  
3) Industrial experience ( less than 3 years )  
4) Long industrial experience ( more than 3 years ) |
|   | Use Cases Experiences                             | 1) No knowledge  
2) Studied courses  
3) Industrial experience ( less than 3 years )  
4) Long industrial experience ( more than 3 years ) |
|---|-----------------------------------------------|-----------------|
| 8 | Testing experience                             | 1) No knowledge  
2) Studied courses  
3) Industrial experience ( less than 3 years )  
4) Long industrial experience ( more than 3 years ) |
| 9 | Inspection experience in general ( Any kind of document ) | 1) No knowledge  
2) Studied courses  
3) Industrial experience ( less than 3 years )  
4) Long industrial experience ( more than 3 years ) |
| 10| C code inspection experience                  | 1) No knowledge  
2) Studied courses  
3) Industrial experience ( less than 3 years )  
4) Long industrial experience ( more than 3 years ) |
| 11| Compiler Construction experience               | 1) No knowledge  
2) Studied courses  
3) Industrial experience ( less than 3 years )  
4) Long industrial experience ( more than 3 years ) |
| 12| Statistics experience                          | 1) No knowledge  
2) Studied courses  
3) Industrial experience ( less than 3 years )  
4) Long industrial experience ( more than 3 years ) |
| 13| Personal Software Process experience           | 1) No knowledge  
2) Studied courses  
3) Industrial experience ( less than 3 years )  
4) Long industrial experience ( more than 3 years ) |
| 14| Experience of checklists in general            | 1) No knowledge  
2) Studied courses  
3) Industrial experience ( less than 3 years )  
4) Long industrial experience ( more than 3 years ) |
| 15| Experience of checklists in inspections ( Any kind of document ) | 1) No knowledge  
2) Studied courses  
3) Industrial experience ( less than 3 years )  
4) Long industrial experience ( more than 3 years ) |
## Experiment Post-test

### Name

<table>
<thead>
<tr>
<th>Statements</th>
<th>Strongly agree</th>
<th>Agree</th>
<th>Neutral</th>
<th>Disagree</th>
<th>Strongly disagree</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. The instructions were easy to follow</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. The testing and inspection tasks were trivial to perform</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3. I learned something new from performing the experiment</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4. I learned something new from the homework assignments</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5. I learned something new from the experiment introduction</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Statements</th>
<th>Strongly for testing</th>
<th>For testing</th>
<th>Neutral</th>
<th>For inspection</th>
<th>Strongly for inspection</th>
</tr>
</thead>
<tbody>
<tr>
<td>6. Before the experiment, I preferred one of the techniques</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7. After the experiment I prefer one of the techniques</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8. I think I performed better using one of the techniques</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9. For interface defects, I think one is better</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10. For functional defects, I think one is better</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11. For logical defects, I think one is better</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12. To identify mismatch between specification and code, I think one is better</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13. To identify that there is a defect, I think that one technique is better</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Statements</th>
<th>Strongly for counter</th>
<th>For counter</th>
<th>Neutral</th>
<th>For correlation</th>
<th>Strongly for correlation</th>
</tr>
</thead>
<tbody>
<tr>
<td>14. I think one for the programs was easier to understand</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15. I have more experience with similar programs</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16. I think I performed better</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
for one of the programs
# Qualitative Questions

<table>
<thead>
<tr>
<th>ID</th>
<th>Question</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>How many work weeks have you spent programming since you became interested in computers?</td>
</tr>
<tr>
<td>2</td>
<td>How many work weeks have you spent programming in C or C++?</td>
</tr>
<tr>
<td>3</td>
<td>What percentage of that is in C respectively C++ programming?</td>
</tr>
<tr>
<td>4</td>
<td>Do you easily find faults when reading code?</td>
</tr>
<tr>
<td>5</td>
<td>Do you easily find faults when reading text?</td>
</tr>
<tr>
<td>6</td>
<td>What kinds of, if any, structured testing have you used before the experiment (Structured testing is defined as testing with previously defined test cases)?</td>
</tr>
<tr>
<td>7</td>
<td>How many hours or work weeks have you spent on more structured testing before the experiment?</td>
</tr>
<tr>
<td>8</td>
<td>How long has it been since you last used a structured testing technique before the experiment?</td>
</tr>
<tr>
<td>9</td>
<td>Have you ever used inspections?</td>
</tr>
<tr>
<td>10</td>
<td>How long has it been since you last used inspections?</td>
</tr>
<tr>
<td>11</td>
<td>You stated that you preferred testing in the post-test – why is that?</td>
</tr>
<tr>
<td>12</td>
<td>When you answered the post-test, did your answers refer to the experiences from the experiment, or from experiences before the experiment, or both?</td>
</tr>
<tr>
<td>13</td>
<td>What did you think of the checklist?</td>
</tr>
<tr>
<td>14</td>
<td>Why do you think you missed this defect? (Question was applied to all defects not found by each participant in both programs)</td>
</tr>
<tr>
<td>15</td>
<td>Why did you mark this row as containing an defect? (Question was applied to all false positives found by each participant in both programs)</td>
</tr>
<tr>
<td>16</td>
<td>What would you have needed to find the defects you did not find? (referring to the inspected program of each participant)</td>
</tr>
</tbody>
</table>