Master Thesis
Software Engineering
Thesis no: MSE-2004-05
June 2004

Verification and Validation of Object Oriented Software Design

Guidelines on how to Choose the Best Method

Christian Thurn

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

Contact Information:
Author(s):
Christian Thurn
Address: Lindblomsv 86
372 33 Ronneby
E-mail: christian.thurn@ipbolaget.com

University advisor(s):
Miroslaw Staron
Department of Software Engineering and Computer Science

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

ABSTRACT

[The earlier in the development process a fault is found, the cheaper it is to correct the fault. Therefore are verification and validation methods important tools. The problem with this is that there are many methods to choose between. This thesis sheds light on how to choose between four common verification and validation methods.

The verification and validation methods presented in this thesis are reviews, inspections and Fault Tree Analysis. Review and inspection methods are evaluated in an empirical study.

The result of the study shows that there are differences in terms of defect detection. Based on this study and literature study, guidelines on how to choose the best method in a given context are given.]

Keywords: Review; inspection; experiment; Verification and Validation; Unified Modeling Language.

CONTENTS

1 INTRODUCTION

Design verification is the process when the development team makes sure that the design is good enough. This is done through verification and validation of the design with the aid of different methods. A design can contain some errors and still satisfy the requirements, or the design can be completely error free and still fail to satisfy the requirements. But which is the best way to test object oriented software design and how are the verification and validation methods used?

The objective of this thesis is to present verification methods, evaluate the methods empirically, assess these methods and give recommendations on when and how to use the verification and validation methods. The aim for the evaluation is to provide confidence in the assessment of the methods. The expected outcome is guidelines and recommendations on how to choose verification and validation method for different context. For a company, this work is useful because it provides some guidelines if it is worth the extra preparation work to use a structured verification method, or if a general method is good enough.

Four methods are presented thoroughly in this thesis (technical review, perspective based inspection, checklist-based inspection and Fault Tree Analysis (FTA)), and three of those methods are studied in experiment. FTA is not included into the experiment because this method is much more complex than the other methods. The result from the experiment is analyzed and presented.

The result from the experiment and literature studies is used to assess and give recommendations about the verification methods. The result of the thesis is aimed for persons with a background in software design and verification and validation.

The structure of the thesis is as follows. First the verification methods are presented (In the following order: Reviews, inspections, FTA) and then there is a small part about UML design. The next chapter describes the experiment, which is divided into three parts, one part about the experiment design, one part about the experiment operation and one part about the analysis of the experiment data. The chapter after this is about recommendations how to choose the best method and assessment of the methods. The final chapters are about further work and conclusions of the thesis.

2 VERIFICATION METHODS

This chapter presents methods that are suitable for validation and verification of software design documents and other software artifacts. The first method is reviews and though inspections are a kind of review, inspections have its own part (after reviews) in this chapter, since inspections are used in the experiment. The FTA is next and it is a method specially developed for design verification. Finally there is a part about how to estimate the number of remaining faults after the design verification, and the method chosen is capture-recapture. Capture-recapture is a method used to estimate the number of remaining faults, and is presented briefly as an introduction for further work.

2.1 Review

Review is a generic reading technique that is used to examine software artifacts and documents used in a development project.

Reviews are a group of many verification methods. There are management review, technical review, inspection (described in chapter 2.2), walk-through and audit.

Management review is an evaluation of a project level plan or project status relative to that plan, by a designated review team. Technical review is an evaluation of a software artifact by a team of designers and programmers, and the technical review is used in the experiment.

2.1.1 Management review

Management review is a formal way of monitoring progress, determine the status of plans, schedules, confirm requirements and evaluate the effectiveness of management approaches [IEEE 1998: 5]. The reviews support decisions about corrective actions, changes in resource allocation and changes in the project scope.

A management review is performed by, or for the management personal. The review should also identify consistency with or deviations from the plans and adequacies and inadequacies of procedures. Examples of software artifacts suitable for a management review [IEEE 1998: 5] are audit reports, installation plans, progress reports and software quality assurance plans. More information about management reviews can be found in [IEEE 1998: 5-9]

2.1.2 Technical review

A technical review is a formal evaluation of software artifacts like design documents, done by a team of designers and programmers [Kusumoto et al. 1996: 120].

A team of qualified personnel evaluates the software artifact to determine its suitability for its intended use and identify discrepancies from its specification standards. The management is provided with evidence whether:

- The software artifact conforms to its specification.
- The software artifact adheres to regulations, standards, guidelines, plans, and procedures that are applicable to the project.
- Changes are properly implemented and only affect the areas specified by the change request.

Examples of software artifacts that are suitable for a technical review are [IEEE 1998: 9] software requirement specification, software design documents and software user documentation. An example of how to perform a technical review can be found in Appendix A7.

2.1.2.1 Responsibilities

There are responsibilities or roles that must be set before the review.

The review is conducted for a decision-maker and he or she shall determine if the review objectives have been met.

The review leader is responsible for the review. The responsibility includes administrative tasks, ensuring that the review is conducted in an orderly manner and making sure that the review meets its objectives.

There is also a kind of secretary present at the review – the recorder. The recorder shall document anomalies, decisions, action points and recommendations made by the review team.

The technical staff shall actively participate in the review and in the evaluation of the software product.

There are some roles that also might be assigned and considered.

The management staff may participate for the purpose of identifying issues that requires management resolution.

The review leader should determine the role of the customer or user representative before the review takes place.

2.1.2.2 Input

The input to a technical review shall include the following [IEEE 1998: 10-11]:

- A statement of objectives
- The software artifact
- Software project management plan
- Current anomalies or issues list
- Documented review procedures

The input should also include the following [IEEE 1998: 11]:

- Relevant review reports
- Any regulations, guidelines, standards, plans and procedures against which the product is to be examined.
- Anomaly categories

2.1.2.3 Entry criteria

The entry criteria shall be fulfilled before the review is conducted.

Project planning documents shall define the need for conducting a technical review. The technical review can also be required by a specific plan, it may be announced and held at the request of functional management, project management, quality management, systems engineering or software engineering according to local procedures [IEEE 1998: 11]. An evaluation of impacts of hardware anomalies or deficiencies on the software artifact might have to be performed.

The following pre-conditions shall be met before conducting a technical review [IEEE 1998: 11]:

- A statement of objectives is established.
- The required inputs are available.

2.1.2.4 Procedures

This is the procedures for the review and should be followed in this order.

The managers must be prepared and shall make sure that the review is performed as required by standards and procedures. The review must also follow law, contracts and other policies. The managers shall [IEEE 1998: 11]:

- Plan the time and resources required for the review, including support functions according to appropriate standards.
- Provide the funding and facilities that are required to define, plan, execute and manage the review.
- Provide training and orientation on the review procedures.

- Make sure that the review members have appropriate levels of expertise and knowledge to comprehend the software artifact that is being reviewed.
- Make sure that the planned reviews are conducted.
- Act on the recommendations from the review team in a timely manner.

The review leader is responsible for planning the review and this includes activities like these [IEEE 1998: 11-12]:

- Identify the review team.
- Assign specific responsibilities to the team members.
- Schedule and announce the meeting place.
- Distribute review material and allow adequate time for the team member's preparation.
- Set a timetable for the distribution of review material, the return of comments and forwarding the comments to the author.

The review team shall also determine if alternatives shall be discussed at the review meeting. The alternatives can be discussed at the review meeting, at a separate meeting or be left to the author the resolve.

To give the reviewers an overview, a qualified person should present the review procedures, when requested by the review leader. This presentation can be a part of the review meeting or it can be held at a separate meeting.

A technically qualified person should present the software product, when requested by the review leader, to give the reviewers an overview of the software product. This presentation can also be a part of the review meeting or held at a separate meeting.

Preparation

All the members in the review team must prepare by examine the software product and other review inputs prior to the review meeting. If an anomaly is found during this examination, it should be documented and sent to the review leader [IEEE 1998: 12]. The anomaly should be classified by the review leader, and then forwarded to the author of the software product for disposition. The team leader must also verify that each team member is prepared for the technical review, and the team leader should gather the individual preparation times and record the total. If the members are not adequately prepared, then the review leader must reschedule the meeting.

Examination

The examination meeting shall accomplish the following goals [IEEE 1998: 12-13]:

- Decide on an agenda for evaluating the product and its anomalies.
- Evaluate the software product.
- Determine if the product is complete
- Determine if the product conforms to the regulations, standards, guidelines, plans and procedures applicable to the project.
- Determine if changes to the product are properly implemented and only affect the intended areas.
- Determine if the product is suitable for its intended use.
- Determine if the product is ready for the next activity.
- Identify anomalies.
- Generate a list of action items, which are emphasizing the risks.
- Document the meeting.

After the review, documentation shall be generated to document the meeting, list the anomalies and describe the recommendations to management.

The review leader must verify that all the action items assigned during the meeting are closed.

2.1.2.5 Exit criteria

A review is considered complete when the examination activities have been completed and the output described below exits.

2.1.2.6 Output

The output shall consist of documented evidence that identifies the following [IEEE 1998: 13]:

- The project that have been reviewed.
- The team members.
- The software artifact that have been reviewed.
- The review objectives and whether they are met.
- A list of resolved and unresolved anomalies.
- A list of unresolved system or hardware anomalies. Or a list of specified action items.
- A list of the management issues.
- The status of action items (open, closed), ownership and target date or completion date (if closed)
- Recommendations made by the review team on how to dispose any unresolved issues and anomalies.
- If the product meets the applicable regulations, standards, guidelines, plans and procedures without deviations.

This standard sets the minimum requirements for the content of the documented evidence. Then it is up to the local procedures to prescribe additional content, format requirements and media.

2.1.3 Walk-through

The purpose is to evaluate a software product, but a walk-through might also be held for the purpose of educating an audience about a software product. The most important objectives are to [IEEE 1998: 20]:

- Find any anomalies.
- Improve the product.
- Consider alternative implementations.
- Evaluate conformance to applicable standards and specifications.
- Educate and inform.

Other objectives are an exchange of techniques and style variations and training of the participants. The walk-through might also point out deficiencies like readability problems or modularity problems in the design or source code.

Examples of documents suitable for walk-through reviews are design descriptions, test and user documentation and manuals. Since design is included in the list, this method is suitable for design verification. Additional information about walk-through can be found in for example [IEEE 1998: 20-25].

2.1.4 Audit

The reason for having an audit is to provide an independent evaluation of conformance of software artifacts and processes to certain regulations, guidelines, standards, plan and procedures. Examples of documents suitable for an audit are different plans, like back up, recovery, contingency and disaster plans. Manuals, contracts, source code and different types of reports are also suitable for this procedure [IEEE 1998:25]. Design is not mentioned and audits are therefore not covered any further in this report. More information about audits can for example be found in [IEEE 1998: 25-31].

2.2 Inspection

Inspection is a kind of reading/inspection technique that tries to find in the software artifacts by using formal procedures and a structured reading/inspection of the software artifacts.

Inspections were originally developed by Michael Fagan in 1976 and are an important way to improve the quality in software projects [Fagan 1976]. It is a static verification technique and it can be applied to any artifact produced during the software development process.

The terminology is not always clear when it comes to inspections. Inspection is a kind of review technique, but it is not clear how it differs from other review processes such as walkthrough or a management review. Fagan says that an inspection shall have formal procedures and produce a repeatable result. According to IEEE Std. 1028-1997 [IEEE 1998, cited by Aurum et al. 2001:2] "An inspection is 'a visual examination of a software product to detect and identify software anomalies, including errors and deviations from standards and specifications'".

The purpose of the inspection is to find software product anomalies. It is a systematic peer examination that [IEEE 1998: 13]:

- Verifies that the software artifact satisfies the specifications.
- Verifies that the software artifact satisfies the quality attributes.
- Makes sure the software artifact conforms to regulations, standards, guidelines, plans and procedures.
- Identifies deviations from those standards and specifications.
- Collects software engineering data, like anomaly and effort data.
- Uses this data to improve the inspection process and its supporting documentation, like checklists.

Examples of documents suitable for an inspection are [IEEE 1998: 14] requirement specification, design description, source code and manuals.

2.2.1 Fagan's inspection process

This is a short summary of the original inspection process developed by Fagan in 1976, and has often been used as a base when improving the inspection technique. The inspection involves teamwork and is organized by a moderator. There are three other roles too. These roles are author, reader and reviewer. Fagan stated six different steps to follow during the inspection phase [Fagan 1976]:

1. Planning: The inspection team is formed and the roles are assigned to the team members.
2. Overview: This part is optional. Here the team is informed about the software artifact.
3. Preparation: The reviewers inspect the material independently, and the aim of this stage is to learn the material and fulfil their roles. The material can for example be a requirement specification, design diagrams or some other document.
4. Examination: This is the inspection meeting. The aim of this meeting is to find and pool the defects together. The moderator takes notes and prepares a list of the defects, which are put in a report after the meeting. Any solutions shall not be discussed and evaluated on this meeting.
5. Rework: The author corrects the defects. Some software artifacts must be corrected and re-inspected many times before they are good enough.
6. Follow-up: The moderator inspects and verifies each correction.

The main steps are preparation, examination and rework, but skipping or combining steps is not recommended. Fagan also recommends that the meeting time should be around two hours [Fagan 1976].

2.2.2 Inspection process

Inspection is a well-structured technique that originally began on hardware logic and has now moved on to design, code and other documents [Fagan 1986]. An inspection team with well-defined roles is necessary to perform the inspection. The team members should be familiar with the software artifact, and have good knowledge about the inspection process. Otherwise they must be trained. The members of the inspection team should examine the material individually. Then they attend a meeting with the purpose to effectively and efficiently find defects early in the development process. After this the list of defects are sent to the author, so the documents can be repaired and updated.

This inspection process is general and is applicable to many types of inspections.

2.2.2.1 Roles

The inspection team consists of different roles, which must be assigned. Each team member can play several roles, but each role requires specific skills and knowledge. The roles defined by Fagan are [Fagan 1976]: Author, moderator, reader and reviewer/reviewers. There are a few ground rules to consider [IEEE 1998: 14]. All participants are inspectors. The author shall not be inspection leader, reader or recorder. The individuals that hold a management position over any member of the inspection team shall not participate in the inspection.

- The author is the architect/designer that produced the design. There are different opinions on whether the author shall be active or passive during the inspection process. Gilb and Graham [Gilb and Graham 1993] claim that the author may be the best defect finder.
- The moderator (also called review leader [IEEE 1998: 14]) chairs the inspection activities, facilitates the interaction between the reviewers and may also be responsible for collecting the defects and put them in a list.
- The reader is an architect/designer that explains the design during the meeting.
- The reviewer/reviewers (also called inspectors [IEEE 1998: 14]) are the ones that review the design and try to find faults. The reviewers shall be chosen to represent different viewpoints at the inspection meeting, like sponsor, designer, safety personnel, project or quality manager, and so on. Some inspectors might be assigned specific topics to focus on, like conformance, syntax or overall coherence. The moderator assigns these roles.
- There might also be a recorder [IEEE 1998: 14] that documents anomalies, action items, decisions and recommendations made by the inspection team. The recorder shall also record inspection data. The moderator may be the recorder.

2.2.2.2 Input to the inspection

The input to the inspection shall include the following [IEEE 1998: 15]:

- Statement of objectives for the inspection.
- The product to be inspected.
- The inspection procedure (documented).
- Report forms for the inspection.
- A list of current anomalies and issues.

The input may also include the following:

- Checklists for the inspection.
- Regulations, standards, guidelines, plans and procedures that the software product is going to be inspected against.
- Hardware specifications.
- Performance data for the hardware.

2.2.2.3 Entry criteria

These are demands that should be fulfilled before conducting the inspection.

The authorization demands are basically the same as for the ordinary review (see also 2.1.2.3). The inspections shall be planned and documented in the project planning documentation.

There are also two preconditions that must be met before performing the inspection [IEEE 1998: 16]:

- A statement of objectives is established.
- The required inputs are available.

Minimum entry criteria

The following events shall have occurred or there shall be a documented rationale (accepted by management) before the minimum entry criteria is fulfilled and inspection is conducted [IEEE 1998: 16]:

- The software artifact that is going to be inspected shall be complete and conform to project standards for content and format.
- Automated error-detecting tools (like spell-checkers and compilers) that are required shall be available.
- Prior milestones are identified in the planning documents.
- The necessary supporting documentation is available.
- If you are going to do a re-inspection, then all the items noted on the anomaly list which affect the product must be resolved.

2.2.2.4 Inspection meeting

According to IEEE [IEEE 1998:17], this is how the inspection meeting should be conducted.

The inspection leader has as duty to introduce the inspection participants and describe the roles the participants have. The leader shall state the purpose of the inspection, remind the inspectors to focus on anomaly detection and not resolution, and tell the inspectors to direct their remarks to the reader and comment on the product and not the author. The inspector may ask the author questions about the software artifact. The leader must also resolve any procedural question raised by the inspectors.

The inspection leader must make sure that the inspectors are prepared for the inspection. If the inspectors are not adequately prepared, then the inspection leader must reschedule the meeting. The inspection leader should also gather the preparation time for each inspector, and record the total time in the inspection documentation.

Any anomalies that refer to the product in general shall be presented to the inspectors and be recorded.

The reader presents the product to the inspection team. The inspection team is supposed to examine the software artifact thoroughly and objectively. The inspection leader shall make sure that the focus for this part of the meeting is on creating the anomaly list. The recorder enters each anomaly, its location, description and classification on the anomaly list. You may use IEEE Std 1044-1993 to classify the anomalies. The leader shall answer specific questions and contribute to anomaly detection, based on the special understanding that the author has of the software artifact. If the meeting participants can not agree on an anomaly, then it shall be logged and marked for resolution at the end of the meeting.

At the end of the meeting, the inspection leader should review the anomaly list together with the team to verify the completeness and accuracy of the list. There should be enough time to discuss every anomaly where disagreement occurred. The leader must prevent the discussion from focus on resolution of the anomaly.

The purpose of this decision is to come to an unambiguous closure to the inspection meeting. The decision shall determine if the product meets the inspection exit criteria. This decision shall also prescribe any appropriate rework and verification.

The inspection team shall identify the software artifact disposition as one of the following [IEEE 1998:18]:

- Accept the software artifact with no or minor rework.
- Accept the software artifact with rework verification. The rework has to be inspected by the inspection leader or a designated member of the team before it is accepted.
- Re-inspection is required. The re-work has to be inspected. At least shall the areas in the product where anomalies were found in the last inspection be examined, and side effects of those changes must also be identified.

2.2.2.5 Rework/follow-up

It is the inspection leader that has to verify that the assigned action items are closed.

2.2.2.6 Exit criteria

The inspection is considered complete when the meeting activities have been accomplished and the output is completed [IEEE 1998:18].

2.2.2.7 Output

This shall be documented evidence that identifies [IEEE 1998:18]:

- The project that have been inspected.
- The team members.
- The meeting duration.
- The software artifact that have been inspected.
- The size of the inspected material (number of pages, lines of code and so on).
- Specific inputs.
- The objectives for the inspection and whether they are met.
- A list of anomalies, with anomaly location, description and classification.
- A summary of anomalies listing the number of anomalies identified by each anomaly category.
- A disposition of the product.
- An estimation of the amount of rework needed and rework completion date.

The output should also include [IEEE 1998: 18]:

- The total amount of preparation time used by the inspection team.

These are the minimum output according to the IEEE standard. It is left to local procedures to add content and set format requirements and media.

2.2.2.8 Document type

The inspection process can handle a great spectrum of documents. Examples are requirement specifications, software designs, source code, test documents or any other documents related to the development process. The size of the documents can vary from a few pages up to dozens of pages or thousands of lines of code. To prevent information overload, the material must be divided in small chunks. The inspection of a document might take several cycles.

2.2.2.9 Inspection pace

A good pace for the inspection must be found. A slower inspection pace often gives good results [Gilb and Graham 1993], but for economical reasons, an infinite amount of time and resources cannot be used. For documents, one page per hour and participant is an appropriate inspection speed and for source code is the limit 150 lines of code per hour [Aurum et al. 2001:5].

2.2.2.10 The size of the team

The software of today is too complex to be designed and developed by a single person. This complexity has been the driving force behind the creation of teams. It has also been found that the quality of the product increases if it is inspected from many viewpoints [Laitenberger and DeBaud, 1997]. The team size can vary from 1-6 persons in general. For code inspections, 1-2 persons are needed. Two persons are better than just one and this increases the performance on the inspection [Aurum et al. 2001:6]. For requirement inspections five or six persons should be used in the team. Design inspections are between these two extremes, and three to four persons could be an appropriate team size.

2.2.3 Inspection techniques

There are seven different techniques that can be used when performing an inspection. They are based on the original inspection process developed by Fagan, but a few adjustments have been made.

2.2.3.1 Active design review

This technique was presented by Parnas and Weiss in the mid 80's [Parnas and Weiss 1985]. They think that conventional inspection methods fail to find defects because of the information overload on the inspectors during the preparation phase. The inspectors are often not familiar with the goals and may not be competent in the task. In the active design review, there are several brief reviews instead of one big review. Each review focuses on a certain part of the software artifact. Different reviewers with different expertise and specific responsibilities perform each review. The errors are classified in terms of inconsistency, inefficiency and ambiguity. There is also a list of questions to be used as a guideline during the review process. Each review has three stages.

1. The reviewers are presented with a brief overview of the software artifact.
2. The reviewers study the material following the guidelines.
3. The issues raised by the reviewers are discussed during small meetings where only the designer and the particular reviewers attend.

2.2.3.2 Two-person inspection

In this process the moderator is removed, and the team consists of two members, namely the author and the reviewer. The two-person inspection technique were applied by Bisant and Lyle [Bisant and Lyle 1989] to study the productivity of programmers during the design and coding phase. Bisant and Lyle found that this technique had immediate benefits in program quality and productivity. The improvements were even better when novice and less productive programmers were used.

2.2.3.3 N-fold inspection

N small and independent teams are used for this approach [Martin and Tsai 1990]. By using multiple teams, the idea is to find defects that a single team would not have found. Different teams also detect different defects. The size of the teams varies between three and four persons and consists of one author and the reviewer(s). There is only one moderator, which coordinates the teams and collects the inspection data. The teams follow the six steps in Fagan's inspection process and the teams also work in parallel sessions. This process begins with each team member reviewing the documents individually for about two hours. Then each team meets and discusses the defects. This meeting might also last for about two hours. When the inspection process is done, only one team becomes responsible for the rest of the development process.

2.2.3.4 Phased inspection

Knight and Myers developed this inspection technique [Knight and Myers 1993]. Some ideas are taken from active inspections, Fagan's inspection and N-fold inspections. The product is reviewed in a series of partial inspections, called phases. Each phase is conducted in sequential order and there might be up to six phases for a simple inspection. Each phase has a specific goal. For each phase, the reviewers fully examine the product for fulfillment with a specific property and the reviewers do not move on to the next phase until the corrections are completed. Two types of phases are used, single-inspector and multiple-inspector phase. In the single inspector phase, a single person studies the product against a checklist. A technical writer is a good choice for this phase. After the single inspector phase, multiple reviewers individually study the product using different checklists. Then the reviewers compare their findings at a meeting.

2.2.3.5 Inspection without a meeting

In Fagan's inspection the focus is on the meetings where the reviewers meet and discuss their findings. The meetings are supposed to bring synergy to the team. This synergy is accomplished because of the combination of different viewpoints, skills and knowledge from many inspectors. Fagan said that the meeting was crucial and most defects are found during the meeting [Fagan 1976].

Many organizations have radically changed the structure of Fagan's inspection process, and now it contains three stages: preparation, collection (with or without a meeting) and rework. The expected outcome from the two first stages has also been changed. Now the preparation stage is used to find defects and the collection phase is just used to pool out the defects from the reviewers. This technique is used in the experiment.

Votta [Votta 1993] has found five reasons for holding meetings: education, schedule deadline, synergy, requirement and competition. The meetings tend to minimize 'false positives', but may not necessarily bring the synergy. Holding meetings after the preparation stage does not have a significant effect on the inspection. Meetings are more costly than non-meeting methods and do not find that many more defects. Meetings are time consuming, a lot of people are tied up at the same time and often many meetings are needed to complete the inspection, since each meeting only covers a small part of the product.

Another aspect raised by Votta [Votta 1993] is that, although n number of reviewers participates in the meeting, only two reviewers can actually interact at any time. This is because face-to-face communication is established in sequential order. Most of the time, only 30-80% of n-2 reviewers are actually listening to the conversation. So in each meeting, in average n-4 reviewer hours are wasted. Votta also found that most defects are found before the meeting.

2.2.3.6 Structured walkthroughs

There are several techniques that are similar to the process by Fagan. Yourdon has one inspection technique or walkthrough that he calls structured walkthroughs [Yourdon 1989]. The preparation stage and the inspection meeting do not take more than two hours. The reviewers are encouraged to note positive aspects of the document. Walkthrough is defined as "a group of peer review of a product" [Aurum et al. 2001:11].

2.2.3.7 Process brainstorming meeting

Gilb and Graham [Gilb and Graham 1993] have added and extra step to improve the inspection process. This step is added just after the inspection meeting, and is called process-brainstorming meeting.

2.2.4 Reading techniques

There are a few reading techniques to use during the preparation stage. The choice of reading technique has a potential impact on the inspection performance. The techniques are classified as systematic and non-systematic [Porter and Votta 1994] [Porter et al. 1995]. The systematic technique applies a very explicit and structural approach to the process, and it also provides a set of instructions and explanations on how to read the document and what to look for [Shull et al. 2000]. The non-systematic techniques apply an intuitive approach with little or no support for the reviewer.

In this study checklist and perspective based reading is used, and is described more thoroughly.

2.2.4.1 Ad hoc reading

No procedures need to be followed and no training is needed. The reviewer uses his/hers own knowledge and experience to find defects in the documents. No support is offered to the reviewer.

2.2.4.2 Checklist

This approach is more systematic than ad hoc reading and has been a common technique used since 1970s [Sabaliauskaitea et al. 2003]. The reviewer answers a list of questions or checks a number of predefined issues. The questions guide the reviewer through the review process. The checklists are developed from the project itself, and must be prepared for each type of document and for each type of product [Aurum et al. 2001:12]. Checklists often focus on questions that guide the reviewer and the checklist should not be more than one page long for each type of documentation [Gilb and Graham 1993].

The following weak points have to be considered [Sabaliauskaitea et al. 2003].

- The questions are often too general and not tailored to a particular development environment.
- Concrete instructions on how to use the checklists are often missing. It is unclear when and on what information the inspector is to answer a particular checklist question.
- The questions are often limited to detection of defects that belong to particular defect types. Therefore the inspectors might not focus on defect types that are previously undetected and because of this miss whole classes of defects.

An example of how to perform a checklist-based inspection can be found in Appendix A9.

2.2.4.3 Stepwise abstraction

This technique was developed for code reading [Linger et al. 1979]. Each reviewer identifies subprograms in the software and determines the function of those subprograms. After this, each reviewer combines the subprograms to determine the function of the entire program. They also construct their own specification for the program. This derived specification is compared with the original specification for the program. Through this comparison, inconsistencies are found and identified as defects and studied in the next stage. The disadvantage with this approach is that it is only applicable to source code.

2.2.4.4 Scenario based reading

This is a systematic reading technique with specific responsibilities for the reviewers [Porter and Votta 1994] [Porter et al. 1995]. It was originally developed to find defects in requirement specifications. The defects are classified, and a set of questions is developed for each defect class. The scenarios are a collection of procedures for detecting particular types of defects and they are also focused on a particular viewpoint. To answer the questions, the reviewer must follow the specific scenario.

2.2.4.5 Perspective based reading

This is an enhanced version of scenarios [Basili et al. 1996]. The main idea with this technique is that the software product should be inspected from the perspective of different stakeholders. The perspective depends on the roles the participants have within the software development process. The focus is on the needs and point of view of a particular stakeholder. Each scenario consists of a set of questions, and is developed based on the viewpoint of the stakeholders. The perspective based inspection technique provides guidance for the inspector in the form of a scenario on how to read and examine the document.

The reviewers read the document from a particular viewpoint, and can also produce a physical model that can be used to answer the questions for that particular viewpoint. The aim is that this structural approach reduces any gaps or overlaps between the reviewers during the review process.

Perspective based reading is believed to work better on generic documents [Aurum et al. 2001:14]. To use this technique, the reviewers should have a certain range of experience. There are also beneficial qualities, because it is focused, systematic, goal-oriented, customizable, and transferable via training.

This scenario consists of three major sections:

- Introduction which describe the quality requirements.
- Instructions that describe what kind of document to use, how to read and how to extract information.
- Questions which the inspector has to answer during the inspection.

An example of how to perform a perspective based inspection can be found in Appendix A9.

2.3 FTA

Fault Tree Analysis (FTA) is a verification method that tries to avoid software design fault by deriving safety assertions using the fault tree analysis. This method also computes a behavioral graph of the specification and analyses statistically if this graph satisfies the safety assertions derived earlier.

Safety has become an important factor in software development, especially in safety critical systems. If there is a design fault lingering in such a system, then a serious failure could occur. Many techniques have been developed to assure safety in a software system, like verification and validation, but they all have their limitations [Fukaya et al. 1994:208]:

- It is hard to define safety assertions from the initial product requirement specification.
- Formal verification is often limited to the source code and can therefore not address the origin of the problem.

These issues are something that FTA tries to solve. The FTA approach has two main features [Fukaya et al. 1994:208]:

- There is a fault analysis method that includes both hardware and software. This method can derive safety assertions that must hold in the software specification. Logical notations that represent the safety assertions are also provided.
- An automatic verification procedure is also provided, which computes a reachability graph of the specification. Then it analyses statistically if this graph satisfies the safety assertions. This procedure also localizes design fault and calculates the fault level. The fault level is the worst situations that can happen because of software design faults.

2.3.1 Software design steps

This part describes the design steps used in the FTA. A microwave oven with built in software is used as an example from real life [Fukaya et al. 1994].

2.3.1.1 Definition of software primitive objects

Extract Software Primitive Objects (SPO). SPO's are essential function elements in a product. These objects are abstract representations of items, like control devices or control data that are closely related to the target product. For the microwave oven, there are SPO like "Thermometer object" and "Cooking-timer object" as an example [Fukaya et al. 1994:208]. The SPO's are based on the product characteristics and extracted with the aid of object-oriented approaches.

Define modes and methods for each SPO. Modes are sets of states. In these states, the objects can change.

Define a specification of each SPO's activities. Methods are sets of operations that each SPO provides. There are two kinds of methods, namely events and actions. Events are used to detect the mode of a SPO and actions are used to change the mode.

2.3.1.2 Description of a product specification

A product behavioral specification describes the activities of a product. This specification is described through a state transition diagram. This diagram consists of states and transitions and each transition has actions and events. Events and actions can be described like this [Fukaya et al. 1994:209]: [a primitive object name, method]. Example: [Timer, START]. Product behavioral specifications are the representation of the communication of objects.

2.3.2 Verification method

This method detects logical errors during the design phase and takes two steps [Fukaya et al. 1994:209]:

- Generate safety assertions, which must hold in the product specification.
- Verify if the product specification satisfies the defined safety assertions. This is done automatically.

2.3.2.1 Generate assertions

One method to generate assertions is to use HSFTA (Hardware and Software Fault Tree Analysis). It can be applied to systems that contain both hardware and software and it can generate assertions to avoid faults [Fukaya et al. 1994:209].

- Use HSFTA to analyze the fault factors of a product, including software.
- To avoid software fault factors, derived safety assertions must hold in the software specification. These assertions must be translated into logical constraints between SPO.

2.3.2.2 Hardware and Software Fault Tree Analysis

FTA is based on the question [Fukaya et al. 1994:209] "what can be dangerous and what can cause danger". This method which is top-down analysis, localizes the causes of undesirable situations and the sequence of such situations.

FMEA (Failure Mode and Effect Analysis) is based on the question [Fukaya et al. 1994:209] "what happen if…" and is a bottom-up analysis method. It discovers incompleteness and a potential failure of design.

When fault analysis is applied to software, the top-down approach (FTA) is more suitable, since it is difficult to find out primitive fault factors. So FTA can be applied to software fault/failure analysis.

To be able to avoid analyze fault factors, the software must satisfy a number of assertions. If it is possible to define those assertions formally, then it becomes possible to verify if the assertions hold in the specification. This can be done automatic and it is possible to assure that the software is free from design faults. But the conventional FTA can not be applied to the source code of the software. HSFTA can be applied to systems that include both hardware and software and it can also generate assertions. The reason for applying HSFTA to a product is to derive safety assertion, as a complement to the requirements. HSFTA unfolds top-level fault factors into hardware and software factors and has the following two stages [Fukaya et al. 1994:209]:

- Unfold fault factors to system fault factors, using FTA.
- Unfold fault factors to software design fault factor, using HSFTA.

The first stage: Unfold to primitive object level

In this step, conventional FTA is used. There are three steps in this stage

1: All undesirable situations of the product are enumerated.

2: The primary factors that could cause the above factor and related factors are analyzed.

3: The logical combination of analyzed factors is defined.

Step two and three are repeated until the objects appear as the lowest factors. All factors must be unfolded to three kinds of items [Fukaya et al. 1994:209-210]:

- Faults of Primitive Software Objects.
- Physical faults.
- Unexpected human behaviors.

The second stage: Unfold to mode and method of primitive object

Now the fault factors are unfolded to modes and methods of primitive objects and their relations. The designers enumerate those primitive factors that cause faults of SPO and analyze the relation between those factors.

Unfolding is repeated until the following factors appear [Fukaya et al. 1994:210]:

- Software design fault factors.
- Destruction of software, ROM, RAM by external factors.
- Unexpected human behavior.
- Fatigue of hardware by physical factors.

A microwave oven is heated by a kind of heater. Heating time depends on factors like cooking patterns, user operations and the temperature of the oven. If you consider this mechanism, overheating can be caused by a fault in one or in several objects. The factors that can cause faults in the cooking-timer object are analyzed in the following way [Fukaya et al. 1994:210].

Design faults in product specification.

The timer is unable to control the heating time. It is also unfolded that:

- When the heater is switched on, the timer does not start counting.
- When the timer reaches zero, then the heater does not turn off.

These are cases where design faults are present in the product behavioral specification that controls "timer object" and "heater object".

Faults in primitive objects.

The user cannot program the timer correctly or the timer cannot count down in a correct way. Such fault might occur if there are bugs in the "timer object" implementation.

Software destroyed.

External factors might disturb the product in spite of a correctly implemented specification.

When this fault analysis has been completed, then the designers can define assertions to avoid software design fault factors and such assertions are invariant properties of a product specification. If HSFTA is used, it is possible to specify software design fault factors that are unsafe.

Translate assertions into logical forms

To be able to verify if safety assertion should be in a product specification, the assertions must be defined as formal notations. The assertions can be represented as constraints on several SPO. To avoid the factor "overheating situation", there are many assertions on SPO in the behavioral specification. Examples of constraints on the heater and timer object are [Fukaya et al. 1994:210]:

- When the heater is turned on, the timer should start immediately or it has already started.
- When the heater is on, the timer should be checked if the countdown of the timer has reached zero.
- When the timer countdown reaches zero, then the heater should be turned off immediately.

These constraints must be translated into a logical form, and it looks like this [Fukaya et al. 1994:211]:

"(1) Heater [ON] $\Rightarrow$ Timer [START] V Timer [counting]

(2) Heater [on] $\Rightarrow$ Timer [OVER]

(3) Timer [OVER] $\Rightarrow$ Heater [OFF]"

When all these constraints hold in the product behavioral specification, then design faults are avoided and safety is assured of the software.

2.3.3 Verification procedure

This verification procedure computes a reachability graph of the product behavioral specification. Then this method analyzes statistically if this graph can hold the safety constraints derived from HSFTA.

2.3.3.1 Compute reachability graph

If a behavioral specification is considered as a directed graph a reachability graph is generated, by simulating a state transition diagram from its initial state. The reachability graph shows the sequences of a product's activities.

Given: A product behavioral specification, one set of SPO definitions and a set of constraints.

Compute: The reachability graph is composed of nodes and directed branches, which is a connection between adjacent nodes. There is a model for each node. The model (M) is a set of mode of SPO that constitute a product.

Model M calculation [Fukaya et al. 1994:211]

"M = {Object_α (mode_i), Object_β(mode_j),…}"

Graph computation:

1: The verifying procedure sets the target node to the initial state of the behavioral specification.

2: A set of transitions T is gathered from the target node.

3: The validity of each transition t of T is checked.

4: If the checking procedure proves that each transition t can occur safely, then a new branch and node is computed. The model can be recognized automatically, because events and actions of each transition correspond to the methods in SPO definition and the mode of the object can be searched through the behavior. The target node is moved to the next state that is the terminal state of t.

5: If the node satisfies one of the following conditions, then this procedure terminates and selects another transition of T:

- The target node reaches the final state on the behavioral specification.
- The node is already included by the graph. The graph reaches a previously visited state of the behavioral specification, and the new model is equal to the old model that was generated at a previous visit. If the new model is different from the previous model, then a new node is added to the graph.

2.3.3.2 Checking the validity of transitions

When the graph is being computed, this procedure checks if the graph satisfies the safety constraints.

Checking of events

This procedure verifies if a set of transitions T satisfies the safety constraints. A constraint that only includes mode conditions is selected in the left-hand side of constraints, and on the right side are constraints that only include method conditions. When the model of a node satisfies the mode conditions, this procedure verifies if events of at least one transition of T satisfy the method conditions.

Example [Fukaya et al. 1994:212]: "Heater (on) $\Rightarrow$ Timer [OVER]". If model contains "heater (on)", this procedure verifies if at least one of the gathered transitions T contains the event "[Timer [OVER]]".

Checking of actions

First the new model M' is computed when a transition m might occur. Actions of each transition correspond to the method in a primitive object description. So when an action might occur, the mode of an object can be recognized from the behavior in a primitive object description. This procedure verifies if actions of a transition t satisfy the safety constraints. When the actions of transition t and the model M' satisfies the left-hand side of a constraint, then this procedure verifies if the actions of transition t and model M' satisfies the right hand side of this constraint. This is described in example 1 and 2.

Example 1 [Fukaya et al. 1994:213]: "Cooling-Fan (on) $\Rightarrow\neg$ Heater (on)"

If M' contains "Cooling-fan (on)", then this procedure verifies that M' does not contain the mode "Heater (on)".

Example 2 [Fukaya et al. 1994:213]: "Heater [ON] $\Rightarrow$ Timer [START] $\vee$ Timer (counting)"

If a transition t contains the action "Heater [ON]", then this procedure verifies that this transition t contains the action "[Timer, START]" or that M' contains "Timer (counting)". When this checker determines that the formulas are invariable true, then it can be said that a safety assertion holds in a reachability graph and also in a product behavioral specification.

2.3.3.3 Verification results

Fault sequence

When this verification procedure finds an assertion that cannot hold in a reachability graph, the designers can localize the sequence of transitions in the product specification that could cause a fault [Fukaya et al. 1994:212]. This is achieved by tracing the graph from root node to fault branch and node in the reachability graph. By repeating this correction and verification process, the designers can remove all design faults from the product specification.

Fault level

When all assertions have been verified, a set of assertions that cannot hold in a product specification is derived. By using this set and the information from HSFTA, this method can identify the fault level. The fault level means the most undesirable situations that can be caused by several primitive faults [Fukaya et al. 1994:212]. Primitive faults are located on the lowest layer in a FTA diagram. If an assertion does not hold in a behavioral specification, then primitive faults might occur corresponding to this assertion. When several assertions cannot hold, primitive faults that correspond to these assertions are identified. These primitive faults are propagated to an upper layer of faults in the HSFTA diagram. If occurrences of factors is propagated depends on the logical combination of lower factors.

2.4 Estimate the remaining faults

This section describes a method used for estimation of the remaining faults, and this method is described very briefly as an introduction to be used for further work (see chapter 5). The method chosen is capture-recapture.

2.4.1 Capture-recapture

Complex software systems sometimes fail, because of faults introduced in the design stage of the development process. Most of these faults can be removed with a design review or design inspection, but often a few faults remain undiscovered. The number of undiscovered faults can be estimated by using capture-recapture methods. The ultimate goal would be to prevent all design faults. The first step is to develop methods to find design of low quality, which is done by estimating the number of remaining faults after a design review or inspection. This part focus on two approaches, Maximum Likelihood Estimator (MLE) and Jackknife Estimator (JE).

2.4.1.1 Number of undiscovered faults estimators

These two estimators are used by biologist for capture-recapture studies of wildlife populations, but these estimators can also be used for estimations in software development.

MLE is derived from assumptions that accommodate different reviewer's probability of detecting a fault, and all faults have the same probability of detection.

JE is the opposite of MLE. The reviewers are equally good at finding faults, and the faults have different detection probability.

The problem description and notation can be found in [Scott et al. 1993:1046].

Maximum likelihood

This kind of MLE is based on a probability model. In this model, the events that reviewer *j* discovers fault *i* are independent with probability $p_{ij} = p_i$ that depend only on the reviewer. This is described by saying that all faults are probabilistically independent and identical. The reviewers act independently and the different reviewers might have different probability for discovering faults. How to calculate this can be found in [Scott et al. 1993:1046].

Jackknife

Burnham and Overton [Burnham and Overton 1978] developed an estimator of population based on the generalized jackknife. Their JE should do well when the faults have varying difficulty. Their derivation uses a probability model where the difficulty of each fault is described by its discovery probability p_i (i = 1,...,N). The p_i's are a random sample from an unknown distribution. The p_i's are given, the events that reviewer j discovers fault *i* are independent with the probabilities $p_{ij} = p_i$, and the events only depends on the fault index *i*. Reviewers are probabilistically independent and identical. The difficulties for faults (p_i's) are differing but the difficulty for a fault is the same for all reviewers. The JE derived from this model and how to make the calculations can be found in [Scott et al. 1993:1046]:

Grouping faults

Since MLE treats all the faults the same, it can do poorly if there are several types of faults with different discovery probability. That is why these fault have been pooled together to estimate f_0 (Number of undiscovered faults). It is helpful to classify the faults into small number of groups and estimate the faults within each group separately. These groups need to be formed in a special way. For a given reviewer and a given group, all the issues should have the same capture probabilities.

Confidence bounds

For estimators with small bias, it could be interesting to compare the coverage probabilities for confidence bounds that are constructed using various approximations. Three different methods have been selected by [Scott et al. 1993:1046] for construction upper confidence bounds on f_0. The upper confidence bound is the highest value of f_0 that the data can support with credibility. If the upper bound is far greater than $f^\wedge_0$, then this indicates that the data supports a wide range of values for f_0.

Lower bounds are equally easy to calculate, but upper bounds are more useful for a software developer that wants to be confident about the number of remaining fault in the design.

There are two methods that can be used to calculate the confidence bounds. The first method is Wald confidence bounds for f_0, which is based on the asymptotic normal distribution of the MLE. The calculations can be found in [Scott et al. 1993:1046].

The next method is likelihood-ratio confidence bounds. They are based on the asymptotic distribution of the likelihood-ratio statistic. How the likelihood-ratio is calculated is described in [Scott et al. 1993:1047].

3 SOFTWARE DESIGN METHODS

This chapter describes design techniques and languages used when making the design diagrams in this thesis. The design language described is UML. Stereotypes are also described since it is a technique to improve the readability of UML design diagrams.

3.1 Unified Modeling Language

This part describes UML which is used for when making design diagrams in object oriented design. A language specification can be found at the Object Management Group web site [OMG].

3.1.1 Introduction

UML is a language that is used for specifying, visualizing and constructing the artifacts in an object oriented software system. It is very popular and is an industry standard and has been adopted by CASE tool vendors.

Object oriented design is about assigning responsibilities to different parts of the software. The different parts are divided into classes, and each class has responsibility and functionality. After this is done, it is time to decide and describe how these parts collaborate. All this can be described by using UML.

When the software is executed, objects are created from these classes. The objects are the classes in an executable form.

3.1.2 Using UML

This part describes very brief how UML is used when creating an object oriented software design.

3.1.2.1 Planning and elaboration phase

In this phase, the requirements and use cases are prepared.

The requirements must be examined, understood and stored in a logical and understandable way in the requirement specification.

Use cases are a technique to improve the understanding of the requirements. Use cases are narrative documents that describe the sequence of events of an actor (external entity, like a user) that uses a system.

3.1.2.2 Analyze phase

In this phase, the software system is analyzed in an abstract level.

The conceptual model describes things in a real world problem domain, and it does not consider the software design domain. It describes concepts, relationship (how the concepts interact) and attributes for the concepts. There is also multiplicity for the attributes. For example, one concept A can interact with n number of concept B.

System sequence diagrams illustrate events from actors to systems. They describe how the actor uses the system and which system events the actor request. The events must be in chronological order.

3.1.2.3 Design phase

Here the implementation of the software system is considered and the design is on a very concrete level.

Collaboration diagram (see Appendix A3 and Figure 3-1). This diagram describes how the objects interact with each other. It usually contains one or many instances (often named) with links (relationship) between the instances. There is also a message for each link that has parameters if applicable.

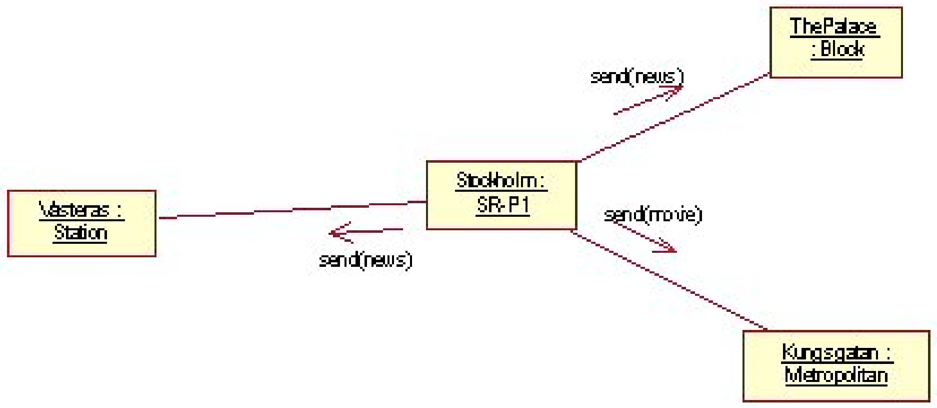


Figure 3-1

Class design diagrams (see Appendix A2, and Figure 3-2) contain the classes needed for the software system. There is multiplicity between the classes, and there can also be associations between the classes. Each class can contain attributes and methods. Attributes are variables and the attributes and methods are visible to other classes.

Inheritance is also specified in the class diagrams. In Figure 3-2, SR-P2, NRG, RadioMatch and SR-P1 inherit from the class Radio. Inheritance means that the functionality in the class 'Radio' is also present in the sub-classes (for example SR-P2 and NRG).

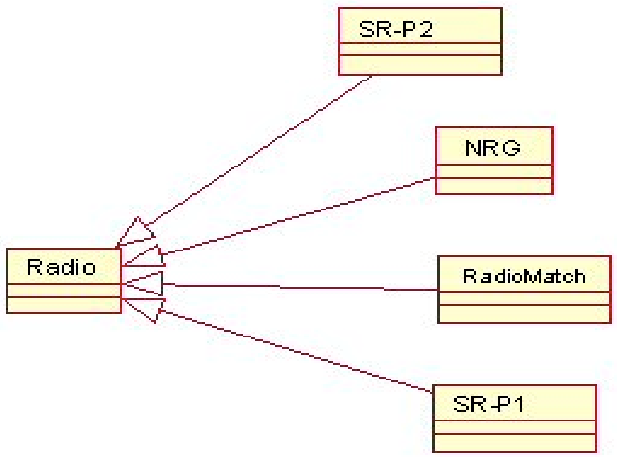


Figure 3-2

3.2 Stereotypes

This part describes stereotypes, which are a way graphically of improving the readability of UML design diagrams.

Collaboration diagram with stereotypes (see Appendix A4 and Figure 3-3). This diagram describes how the stereotyped objects interact with each other. It usually contains one or many instances (often named) with links (relationship) between the instances. There is also a message for each link that has parameters if applicable.

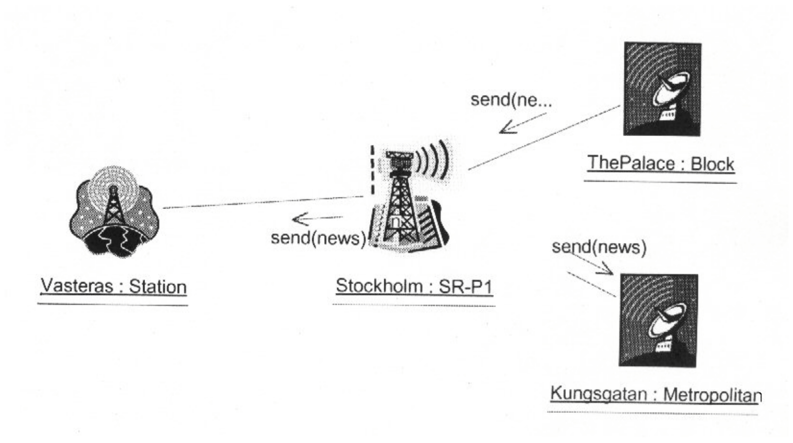

send(ne...
ThePalace : Block
send(news)
Vasteras : Station
Stockholm : SR-P1
send(news)
Kungsgatan : Metropolitan

Figure 3-3

4 THE EXPERIMENT

This chapter describes the experiment. It contains the necessary theoretical background and this is followed by a description on how that theory is applied on this experiment- i.e. how the experiment is designed. Then there is one part about the experiment operation and how the experiment is executed. Finally there is one part about the analysis of the data gathered during the experiment.

4.1 Experiment design

The experiment design describes how the test is organized and run [Wohlin et al. 2000: 52-62]. The experiment is designed based on the statistical assumptions that have been made and which objects and subject that are included in the experiment.

The aim of the experiment presented in the thesis is to compare three different verification methods in the context of investigations of OO (Object Oriented) design documents. The methods chosen for the experiment are technical review, checklist based inspection and perspective based inspection (described in section 2). The reason for choosing Technical review are that it is the review method most suited for design verification and it is interesting to compare a generic method with the more structured inspection techniques. The inspection methods are compared in a similar experiment and therefore the result of this experiment can be compared with the result of the similar experiment. For the perspective based inspection, the designer's point of view is chosen because the other methods focus on the correctness of the design.

Although presented previously, the FTA method is not used in the experiment, because it is much more complex compared to the other methods. Its presence requires a different experimental setting than the one used in this study. The large complexity of the FTA method poses a danger that the results of it might not be comparable to the other methods. Therefore it is justified to remove FTA from the study.

4.1.1 Experiment definition

This is the goal definition summary for this experiment according to the goal template presented by [Wohlin et al. 2000: 42]):

This experiment analyzes technical reviews, perspective based inspections and checklist based inspections for the purpose of assessing the methods with respect to their effectiveness from the point of view of the design personal in the context of master level students examining UML design documents.

4.1.2 Context selection

The most general result in an experiment is achieved with professional personal in a large real software project. But there are risks with this approach and it costs a lot of money and resources. A cheaper approach is to populate the experiment with students, which often also gives a more homogenous group of subjects, with a more equal knowledge base. [Wohlin et al. 2000: 48-49]. Therefore this approach is used in this experiment. There are four dimensions that can be used to characterize the experiment [Wohlin et al. 2000: 49]: Off-line vs. on-line, student vs. professional, toy vs. real problems and specific vs. general.

This experiment is populated by master level students, which have the same basic education in computer science and software engineering. This gives a uniform subject group. The subject's professional background varies a little, but this does not affect this experiment, because of the background questionnaire (see Appendix A1).

The characteristics in this experiment can be described like this:

- Off-line. This experiment is performed outside a real project.
- Student. The participants are students (see also 4.1.6).
- Toy. The problem is of toy character.

- General. The result of this experiment can be applied generally on more than one design, but the design must be based on UML.

4.1.3 Variables

In the experiment, dependent and independent variables must be selected [Wohlin et al. 2000: 51].

Independent variables: These are the variables that can be controlled and changed in the experiment. When choosing those variables, the measurement scale, the range for the variables and the level at which test is made should also be chosen.

Dependent variables: There are two dependent variables in the study that are derived from the hypotheses.

In this experiment the independent variables are:

- The design diagrams (one class diagram and two collaboration (object) diagrams) and the requirement specification.
- The verification methods (technical review, checklist-based inspection and perspective based inspection).

In this experiment the dependent variables are:

- The number of faults found by the subjects.
- The time required for finding the faults.

The independent variables are manipulated in the course of the study and the dependent variables are measured. Based on those measurements, the hypotheses are evaluated.

4.1.4 Hypotheses

The basis for the statistical analysis is hypothesis testing based on the gained data. The hypothesis is stated formally and the data collected during the experiment is used to reject the hypothesis if possible. If the hypothesis can be rejected, then you can draw conclusions based on the hypothesis [Wohlin et al. 2000: 49-50].

Hypothesis statement

Two hypotheses have been formulated during the planning phase, the null hypothesis (H_0) and an alternative hypothesis (H_a, H_1, etc).

Null hypothesis: This is the hypothesis that should be rejected with as high confidence as possible. An example is that all the verification methods find the same number of faults.

Alternative hypothesis: This is the hypothesis that is the counterpart to the null hypothesis. An example is that one verification method finds more faults than the other methods.

Hypotheses in this experiment:

1 = Technical review.
2 = Perspective based inspection.
3 = Checklist-based inspection.
μ = Mean value.
NoF = No of Faults.
TtFF = Time to Find Fault

Hypotheses:

$H_{0\,A}$: $\mu_{\text{NoF }1} = \mu_{\text{NoF }2} = \mu_{\text{NoF }3}$

$H_{0\,B}$: $\mu_{\text{TtFF }1} = \mu_{\text{TtFF }2} = \mu_{\text{TtFF }3}$

Alternative hypotheses:

$H_{1\,A}$: $\mu_{\text{NoF }1} \neq \mu_{\text{NoF }2} \neq \mu_{\text{NoF }3}$

$H_{1\,B}$: $\mu_{\text{TtFF }1} \neq \mu_{\text{TtFF }2} \neq \mu_{\text{TtFF }3}$

4.1.5 Objects

The objects used during the experiment consist of the following:

- One class diagram (see Appendix A2).

- One collaboration (object) diagram (non-stereotyped objects) (see Appendix A3).
- One collaboration (object) diagram (stereotyped objects) (see Appendix A4).
- One fault report-form which contains a description of the experiment, a description of the fault form and the form itself (see Appendix A5).
- One requirement specification (see Appendix A6).

4.1.6 Selection of subjects

If the result should be generalized to a desired population, then the selection must be representative for that population. In this case the population is people with an education in software engineering or with similar knowledge about verification and UML gathered during a professional career. This selection process is called sampling.

Examples of probability sampling are [Wohlin et al. 2000: 51-52]:

- simple random sampling
- systematic sampling
- stratified sampling

Examples of non-probability sampling are [Wohlin et al. 2000: 51-52]:

- convenience sampling
- quota sampling

Convenience sampling is used for this experiment when selecting the participants. This means that the closest and most convenient persons are selected. The subjects are students from the software-engineering program at Blekinge Institute of Technology in Ronneby and are attending the master year. Besides from being very convenient, the knowledge base for the participants is known and equal between the subjects. This is the most important factor, because the subjects have studied design verification and UML approximately the same amount.

For assigning the subjects to experiment groups, another approach is used. The subjects had to answer a few questions in the background questionnaire (see Appendix A1 for the questions and for the result of the questionnaire) to determine the previous knowledge in the area of the experiment. According to the answers received, the subjects are divided into groups in the following way. First subjects with the highest knowledge grade for the methods are chosen. One subjects for each group/verification method. If more than one subject has the highest knowledge grade about one method, then random selecting is used. Then one subject with the lowest grade is chosen for each group. After this the subjects with medium grades are selected and divided into groups, by randomization. (An example of the grades in a group could look like this: 1,3,5). The goal is to have as equal knowledge base between the groups as possible and also to simulate real life. In a real verification process, you might most likely have both verifiers with low method competence and verifiers with very high method competence.

4.1.7 Experiment execution plan

This experiment consists of one pilot study and one major test occasion. There are three groups of students that use one verification method each to verify the three design documents (objects).

This is how the experiment is executed (see Table 4-1).

Action	**Group 1**	**Group 2**	**Group 3**
Introduction	T1	T1	T1
Class diagram (verification)	T2	T2	T2
Collaboration (object) diagram no 1 (verification)	T3	T3	T3
Collaboration (object) diagram no 2 (verification)	T4	T4	T4

Table 4-1

The assignment of treatments to subjects is as follows:

Group 1 uses the technical review.

Group 2 uses the perspective based inspection.

Group 3 uses the checklist-based inspection.

The times (T1, T2, T3 and T4) are equal for each group and are estimated through a pilot study.

All the inspection and review steps are not used in this experiment. The focus in this experiment is to see how the methods perform when used by a subject that is examining the design according to the method instructions. The desired result is to see how each subject perform with a certain verification method. Therefore the meeting is removed. Rework is not used since this experiment is performed outside a project and therefore there is no need to update the design.

General design principles

Balancing [Wohlin et al. 2000: 53-54] is used in this experiment and in this technique, the number of subjects in each group is the same. For each object that is tested, there are an equal number of subjects. All subjects verify every object.

Standard design types

There are four types of standard designs and the one used in this experiment is called 'one factor with more than two treatments' [Wohlin et al. 2000: 54].

The factor for this experiment is the design verification method and the treatments are technical review, checklist based inspection and perspective based inspection.

Example of hypothesis:

See part 4.1.4.

The analysis to be performed in this experiment is the ANOVA (Analysis Of VAriance) [Körner, Wahlgren 2000: 329-334, 355-361] procedure This is further described in chapter 3.3 Analysis and Interpretation.

In this experiment the same UML design documents are used for each treatment (verification method) but the subjects (participants) are not assigned randomly to each treatment. The assignment depended on the result of a background survey performed on the subjects (see 4.1.6. and Appendix A1). Each subject uses only his or hers treatment on all the three UML design diagrams.

4.1.8 Instrumentation

There are three types of instruments for an experiment, namely objects, guidelines and measurement instruments [Wohlin et al. 2000: 54]:

The objects in this experiment are described in the section 3.1.5 Objects.

Guidelines are necessary to guide the participants and in this case it is checklists and process descriptions. In this experiment, there are instructions on how to use each verification method and checklists to follow during the verification.

Measurements are conducted via data collection. In human intensive experiments, the data is collected via forms. A special form is used to collect the data from the participants (see Appendix A5). The background questionnaire (see Appendix A1) is another tool used to gather information from the subjects.

The forms used in this experiment to collect data can be found in Appendix A5, the objects and guidelines can be found in Appendix A2-A4 and A6-A9.

4.1.9 Threats to validity

One important aspect is how valid the results are and the issue of their validity must be considered already in the planning phase. The results shall be valid for the population of interests (external validity). If as general conclusions as in the case of this experiment is to be drawn then the validity must be more general. There are four validity steps to follow [Wohlin et al. 2000: 64]:

4.1.9.1 Conclusion validity

The concern here is the relationship between the treatment and the outcome. There should be a statistical relationship with a given significance. Threats in this group are:

Threat: Low statistical power due to low sample size.

Minimize threat: This threat is not possible to remove. The study is voluntary, so the number of subjects is limited. The study is exploratory and the aim is to identify the initial differences between the methods in the context of OO design. This allows the sample size to be a little smaller. The cost to pay is a lower statistical power in the tests.

Threat: Low statistical power due to low effect size.

Minimize threat: There are three different design diagrams used in the study, which increases the statistical power. The three design diagrams are of different type, which makes sure that the result is not bound to a single type of design diagram.

4.1.9.2 Internal validity

If the relationship exists between the treatment and the outcome, then we must be sure that it is a casual relationship. It shall not be the result of a factor of which we have no control or have not measured (a confounding factor). The treatment shall cause the outcome. The verification methods shall be the reason that the faults are found. Threats in this group are:

Threat: The subject groups might not be equally constituted, which leads to an unequal knowledge base for the groups.

Minimize threat: The subjects answer the questions in the background questionnaire. Based on these answers, the subjects are divided into different groups, with as equal knowledge base as possible.

Threat: Subjects might drop out before the study. This might cause problems with the group constitution and give a skew knowledge base.

Minimize threat: The group division is done the day before the study takes place.

Threat: Imitation of other subjects. The subjects might look at each other's methods and apply their findings on the study, i.e. applying the verification strategy from another method or even copy the faults that another subject has found.

Minimize threat: The subjects are placed in such a way that they cannot look at each other's methods or fault report form. The experimenter is also monitoring the study to prevent any cheating.

Threat: A subject might find false negatives – i.e. faults that are not faults.

Minimize threat: The experimenter examines the fault report forms and only the correct faults are counted.

4.1.9.3 Construct validity

Here the focus is on the relationship between theory and observation. If there is a casual relationship between cause and effect, then we must ensure two things. The first is that the treatment reflects the construct of the case well. The second is that the outcome shall reflect the construct of the effect well. Threats in this group are:

Threat: Too few dependent variables so called mono-method bias [Trochim].

Minimize threat: There are two dependent variables that are measured in the study.

Threat: Hypothesis guessing. The participants might try to guess the purpose of the study and then base their behavior on that assumption.

Minimize threat: The subjects receive an explanation of the purpose of the study before starting to verify the designs. They now that the purpose is to evaluate and assess the verification methods. Therefore they can focus on doing their best instead of guessing.

Threat: The time set for the study might be insufficient, causing the subjects to run out of time before having a chance to find all the faults.

Minimize threat: A pilot study is used to get a time-estimation. The main study time is even larger than the obtained time.

Threat: The time set for the study might be too long, causing the subjects to get bored and lose their motivation.

Minimize threat: A pilot study is used to get time estimation. The main study time is larger than the obtained time, but in relation to the number of faults.

The subjects can leave a little earlier on the verification of diagram three (collaboration diagram 2 (with stereotypes)), if the subjects think all faults have been found.

4.1.9.4 External validity

This factor is concerned with generalization. If the casual relationship between the construct of the cause and the effect exist, can the result be generalized outside the scope of the study? Can a relation between the treatment and the outcome be observed? Threats in this group are:

Threat: The people participating might not be representative for the desired population that the result should be generalized to.

Minimize threat: The subjects are master level students in software engineering. The desired population is software engineering students and people with similar knowledge in software engineering and verification.

Threat: Some of the people participating might not be Swedish software engineering students. Since they are exchange students, they might have a little different educational background.

Minimize threat: The UML design is on a general level and should be understandable by all software-engineering students. All the participants have taken the same course in verification and validation.

Threat: The location where the experiment takes place might be very unusual.

Minimize threat: The experiment takes place in an ordinary classroom at the university – i.e. the usual place for the students.

Threat: The time when the experiment takes place might be awkward.

Minimize threat: The experiment takes place after lunch, which is not too early or too late in the day.

4.1.10 Data analysis methods

This part describes the analysis methods used for analyzing the data gathered during the study.

4.1.10.1 Descriptive statistics

SPSS [SPSS] (SPSS is a software used for statistical calculations) is used to calculate the descriptive statistics. The mean value, standard deviation and confidence interval is for example presented.

The data is also presented in histograms.

A scatter plot is used to find out if any outliers exist in the data.

4.1.10.2 Hypotheses testing

The hypotheses are tested with the method ANOVA (ANalyze Of VAriance) [Körner, Wahlgren 2000: 329-334, 355-361] [Wohlin et al. p105]. SPSS is used in this case too, when performing the calculations. ANOVA is applicable when there is one factor with many treatments as in this case. The factor is the verification methods and the treatments are technical review, perspective based inspection and checklist based inspection. ANOVA is a parametric method and as such it has higher statistical power. ANOVA also requires less input data than other methods applicable to this kind of experiment design. Two calculations are performed, one for each hypothesis.

4.2 Experiment operation

This experiment consists of two steps - a pilot study and the main experiment occasion.

4.2.1 Pilot study

A small pilot study has been performed to verify the quality of the experiment documents and to get a time-estimation on the different steps in the main study. The pilot study was performed like this (3.1.1.1). A certain number of faults should be found, and the time for doing this was measured by the pilot subject. The pilot subject also returned valuable feedback on the quality of the experiment instruments.

4.2.1.1 Pilot study plan

This is how the pilot study was performed (see Table 4-2). There are three tasks to perform and the time was measured for each task. An evaluation of the quality of the experiment documents was also performed.

Task	Time
Find 5 fault in the class diagram (diagram no 1)	?
Find 6 faults in the non-stereotype collaboration (object) diagram (diagram no 2)	?
Find 5 faults in the stereotype collaboration (object) diagram (diagram no 3)	?

Table 4-2

The following results were returned, from the pilot study, and the checklist-based inspection was used for this assignment (see Table 4-3):

4.2.1.2 Pilot study result

This is the result from the pilot study.

Task	Time
Find 5 fault in the class diagram (diagram no 1)	8 min
Find 6 faults in the non-stereotype collaboration (object) diagram (diagram no 2)	7 min
Find 5 faults in the stereotype collaboration (object) diagram (diagram no 3)	6 min

Table 4-3

4.2.2 Main experiment occasion

When the experiment has been planned and designed, it is carried out in order to collect the data that shall be analyzed. This is the part where the verification methods are applied to the subjects and also the part where the experimenter meets the subjects.

This phase consists of three different steps [Wohlin et al. 2000: 75]:

Preparation - where subjects are chosen and forms are prepared.

Execution - where subjects perform their tasks according to the experiment instructions and the data is collected.

Data validation - where the data is validated which in this experiment is done in the analysis phase.

4.2.2.1 Preparation

The first preparation done before the experiment is selecting and informing the participants and the other is to prepare the material (forms and tools).

It is important that the participants are motivated and willing to participate throughout the whole experiment. When then right people have been found and convinced to take part in the experiment, then there are additional aspects to consider [Wohlin et al. 2000: 77]:

- *Obtain consent*: The participants have to comply with the research objectives. They should know the intention of the experiment, and this is accomplished through a document describing the research objectives, and a part describing the intention with the experiment. The reason for the experiment is also explained by the experimenter in the introduction part of the experiment
- *Sensitive results*: If the result of the experiment is sensitive to the participants, then it is important to assure them that the result is kept confidential. In this case the result could be a little sensitive. Therefore the result is kept confidential and only the experimenter has access to the result for each person.

In this thesis numbers are used to represent the subjects, and only the experimenter knows who the real person behind the subject number is.

- *Inducement*: An inducement can be good to motivate the participants a little, and a small inducement was used in this experiment. During the experiment the participants was offered Coca-Cola and when the experiment was done they received a small bag of candy.
- *Deception*: Using deception or betraying the participants should not be used in the experiment, and if used, it should not affect the willingness of the subjects to participate. No deception was applied to the experiment.

4.2.2.2 Execution

A simple inspection experiment as this is carried out during one single meeting. The advantages are for example that the data can be collected direct and the experimenter is present during the meeting and can answer questions that arose. The experimenter can also monitor the experiment and prevent any cheating.

The experiment was performed in a classroom at Blekinge Institute of Technology between 13.00-15.00. There were 11 subjects (master level students in software engineering) and most of them were Swedish students. Two subjects were exchange students, but still on the master level.

Data collection: For this experiment, data was collected manually, by having the participants filling in forms (see Appendix A5).

Experimental environment: If performed within a regular development project, the experiment should not affect the project more than necessary. If the project is affected too much, then the effects you are looking for might be lost. But this experiment was performed outside a regular development project, so there was not any project disturbance.

Time plan: This is how the time was planned during the experiment (see Table 4-4. Each group of subjects had the same amount of time. These time estimations are made according to the result of the pilot study (see 4.2.1.2).

Action	**Time**
Introduction	15 min
Verification of the class diagram (diagram no 1)	20 min
Verification of the first collaboration (object) diagram (diagram no 2)	25 min
Verification of the second collaboration (object) diagram (diagram no 3)	25 min

Table 4-4

For the second collaboration diagram, the subjects were free to leave after 15 minutes, if they felt that there were no more faults to be found (some of the faults were the same in collaboration diagram 2 and 3). All the subjects left after 15-20 minutes.

4.3 Analysis and interpretation

In this phase, the data is interpreted and conclusions are drawn from the data collected during the experiment. This can be done in three steps [Wohlin et al. 2000: 81]. Descriptive statistics, data set reduction and hypothesis testing.

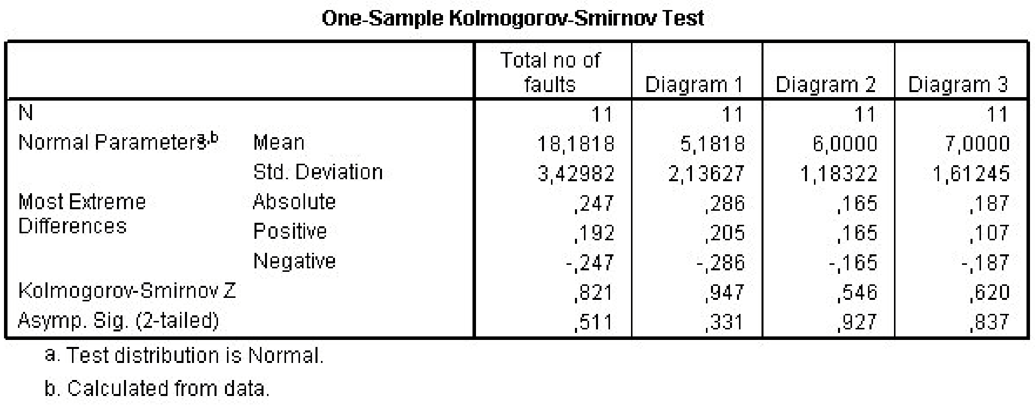
4.3.1 Normality test

For many statistic calculations, the input data should be normally distributed. ANOVA is not an exception from that rule. It is assumed that the input data used for ANOVA calculation is normally distributed [Archambault 2000].

One way to test the data is to use the One-Sample Kolmogorov-Smirnov Test, which is done on the data (number of faults found and the mean time to find a fault) in this experiment.

4.3.1.1 Normality test – number of faults found

This test verifies that the data set for number of faults found is normally distributed. The result for the normality test is as follows (see Figure 4-1).



One-Sample Kolmogorov-Smirnov Test

		Total no of faults	Diagram 1	Diagram 2	Diagram 3
N		11	11	11	11
Normal Parameters[a,b]	Mean	18,1818	5,1818	6,0000	7,0000
	Std. Deviation	3,42982	2,13627	1,18322	1,61245
Most Extreme Differences	Absolute	,247	,286	,165	,187
	Positive	,192	,205	,165	,107
	Negative	-,247	-,286	-,165	-,187
Kolmogorov-Smirnov Z		,821	,947	,546	,620
Asymp. Sig. (2-tailed)		,511	,331	,927	,837

a. Test distribution is Normal.

b. Calculated from data.

Figure 4-1

Table data:

The columns stand for the following: Total no of faults is the total number of faults found by each subject in all the diagrams. Diagram 1 is the class diagram, diagram 2 is the collaboration diagram without stereotypes and diagram 3 is the collaboration diagram with stereotypes.

N: This is the number of subjects in the study.

Normal parameter - Mean: This is the mean value of the number of faults found by each subject. According to this calculation, the data is normally distributed.

Normal parameter – Std deviation: It is a measure of dispersion around the mean value and is further described in 3.3.2.3.

Most extreme - Absolute: This is the absolute value of highest value of the values presented under differences. It is the largest difference from normality.

Differences - Positive: This value is the point at which the empirical CDF exceeds the theoretical CDF with the largest value. It is the largest positive difference from normality.

Differences- Negative: This value is the opposite of the positive value. This is the point where the theoretical CDF exceeds the empirical CDF. It is the largest negative difference from normality.

Kolmogorov-Smirnov Z: This is the product of the square root of the sample size (N) and the largest absolute difference between the empirical and theoretical CDFs.

Asymp. Sig (2-tailed): This probability is presented as a number between 0 and 1. A value of 0.05 means that only 5% of normally distributed data sets are expected to have deviations as large as the extreme absolute value.

An asymp sig value of 0.05 is good evidence that the data set is not normally distributed. If the asymp sig value is greater than 0.05, this implies that there is insufficient evidence to suggest that the data set is not normally distributed. On the other hand does this value (asymp sig higher than 0.05) not provide proof that the data set is normally distributed.

Normality test – conclusion:

The data set (number of faults found) is normally distributed. Since the asymp sig value is much higher than 0.05, the confidence that the data set is normally distributed is high.

4.3.1.2 Normality test – Mean time to find a fault

This test verifies that the data set for the mean time to find a fault is normally distributed. The result for the normality test is as follows (see Figure 4-2).

One-Sample Kolmogorov-Smirnov Test

		Total mean time	Diagram 1	Diagram 2	Diagram 3
N		11	11	11	11
Normal Parameters[a,b]	Mean	2,4155	2,5455	3,1673	1,7600
	Std. Deviation	,41741	1,12429	,82681	,39028
Most Extreme Differences	Absolute	,192	,252	,131	,147
	Positive	,192	,252	,131	,147
	Negative	-,168	-,180	-,111	-,083
Kolmogorov-Smirnov Z		,637	,836	,435	,486
Asymp. Sig. (2-tailed)		,811	,487	,992	,972

a. Test distribution is Normal.

b. Calculated from data.

Figure 4-2

Table data:

The table data description can be found in 3.3.1.1 and only the differences are presented here.

The only difference is in the columns. Total mean time is calculated on the mean time that one subject spent to find all his/hers faults in all diagrams.

Normality test – conclusion:

The data set (mean time to find a fault) is normally distributed. Since the asymp sig value is much higher than 0.05, the confidence that the data set is normally distributed is high.

4.3.2 Descriptive statistics

Here the descriptive data is presented and analyzed. The calculations are done with the software SPSS [SPSS] (for a full set of source data, see Appendix A10), and is presented in two parts. One for number of faults found and one for the mean time used to find the faults.

4.3.2.1 Number of faults found – plot

This is a plot of the faults found by each subject (see Figure 4-3).

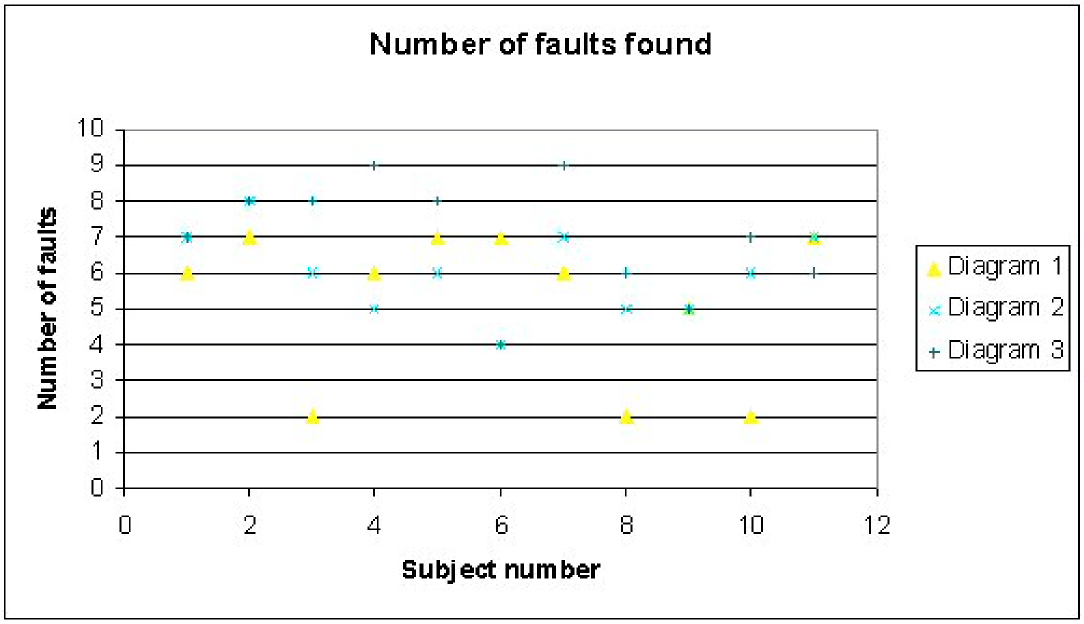


Figure 4-3

Subject 1-3 used perspective based inspection, subject 4-7 used checklist-based inspection and subject 8-11 used technical review. Diagram 1 (class diagram), diagram 2 (collaboration diagram without stereotypes) and diagram 3 (collaboration diagram with stereotypes) are the diagrams used in the study.

The things that are most noticeable in this plot are found in diagram 1. There are three subjects that only have found two faults. Two of those subjects used technical review and that is 50% of the subjects that used technical review in the study.

4.3.2.2 Mean time per fault – plot

This is a plot of the mean time used for each subject to find one fault (see Figure 4-4).

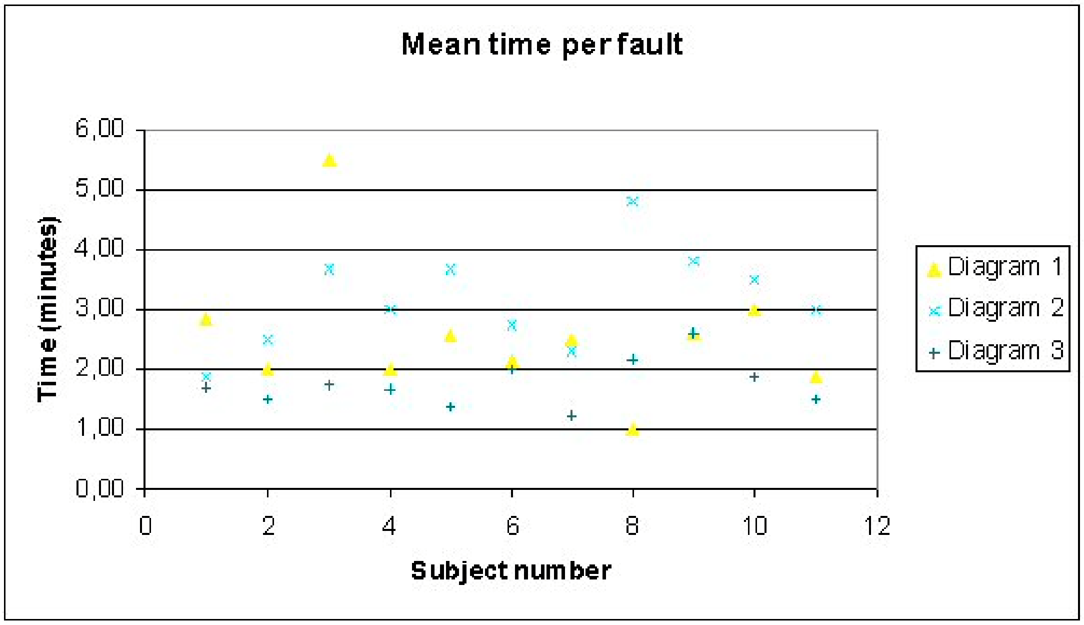


Figure 4-4

Subject 1-3 used perspective based inspection, subject 4-7 used checklist-based inspection and subject 8-11 used technical review.

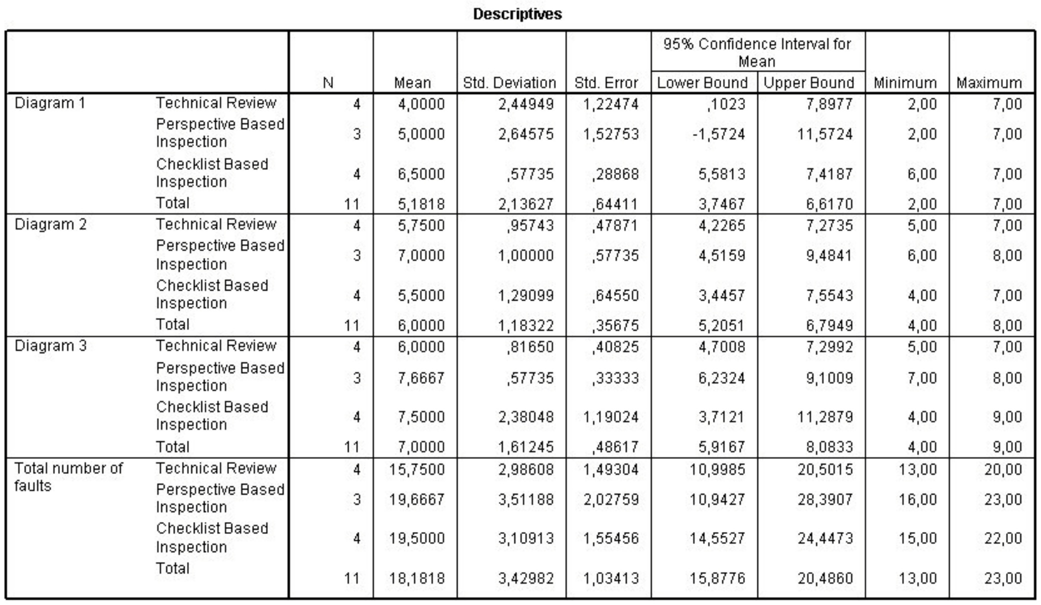
Here are also a few results that are noticeable. Two subjects spent around 5 minutes per fault for one diagram each, which is quite much compared to the other values. By taking a closer look at those unusual events, the reason can be found.

Subject number three found only two faults for diagram 1 (see Figure 4-3), which most likely is the reason for the high amount of time used per fault.

Subject eight also spent a lot of time per fault for diagram number 2, but the number of faults found is not different compared to the other subjects. The reason must be that this subject found faults during the whole inspection time for diagram 2.

4.3.2.3 Number of faults found – Descriptive data

These are the descriptive data returned from the SPSS calculation for number of faults found (see Table 4-5).



Descriptives

		N	Mean	Std. Deviation	Std. Error	95% Confidence Interval for Mean		Minimum	Maximum
						Lower Bound	Upper Bound		
Diagram 1	Technical Review	4	4,0000	2,44949	1,22474	,1023	7,8977	2,00	7,00
	Perspective Based Inspection	3	5,0000	2,64575	1,52753	-1,5724	11,5724	2,00	7,00
	Checklist Based Inspection	4	6,5000	,57735	,28868	5,5813	7,4187	6,00	7,00
	Total	11	5,1818	2,13627	,64411	3,7467	6,6170	2,00	7,00
Diagram 2	Technical Review	4	5,7500	,95743	,47871	4,2265	7,2735	5,00	7,00
	Perspective Based Inspection	3	7,0000	1,00000	,57735	4,5159	9,4841	6,00	8,00
	Checklist Based Inspection	4	5,5000	1,29099	,64550	3,4457	7,5543	4,00	7,00
	Total	11	6,0000	1,18322	,35675	5,2051	6,7949	4,00	8,00
Diagram 3	Technical Review	4	6,0000	,81650	,40825	4,7008	7,2992	5,00	7,00
	Perspective Based Inspection	3	7,6667	,57735	,33333	6,2324	9,1009	7,00	8,00
	Checklist Based Inspection	4	7,5000	2,38048	1,19024	3,7121	11,2879	4,00	9,00
	Total	11	7,0000	1,61245	,48617	5,9167	8,0833	4,00	9,00
Total number of faults	Technical Review	4	15,7500	2,98608	1,49304	10,9985	20,5015	13,00	20,00
	Perspective Based Inspection	3	19,6667	3,51188	2,02759	10,9427	28,3907	16,00	23,00
	Checklist Based Inspection	4	19,5000	3,10913	1,55456	14,5527	24,4473	15,00	22,00
	Total	11	18,1818	3,42982	1,03413	15,8776	20,4860	13,00	23,00

Table 4-5

Table data:

The first column: Diagram 1 (class diagram), diagram 2 (collaboration diagram - without stereotypes) and diagram 3 (collaboration diagram - with stereotypes) are the different design diagrams. Total number of faults is the total number of faults found during the entire experiment.

The verification methods are also listed in this column and Total is the result for all the three methods together.

N: This is the number of subjects in the study.

Mean: The mean value for number of faults found.

Std. Deviation: This is the standard deviation for number of faults. It tells how far away each subject's number of faults found is from the mean value of number of faults found (It is a measure of dispersion around the mean value). Assuming that the number of faults is normally distributed (as in this experiment), the following can be stated [Trochim]:

- Approximately 69% of the faults in the sample fall within one standard deviation of the mean.
- Approximately 95% of the faults in the sample fall within one standard deviation of the mean.
- Approximately 99% of the faults in the sample fall within one standard deviation of the mean.

For this experiment, this means for example that in 69% of the cases, the total number of faults found (mean = 18.1818) is between 14.75198 and 21.61162.

Std. Error: This value indicates the uncertainty of the mean value.

95 % confidence interval for Mean: This interval contains the mean value in 95% of the cases. The width of this interval gives an indication on how uncertain we are about the mean value. A very wide interval indicates that more data should be collected before a definite conclusion can be drawn about the mean value [Easton and McColl 1997].

Minimum: This is the lowest number of faults found for a certain method.

Maximum: This is the highest number of faults found for a certain method.

Descriptive data conclusions:

There is a difference in the mean value for number of faults found. Method 1 (technical review) has found almost 16 faults in average, while the inspections (method 2 and 3) has found almost 20 faults. From that the conclusion that inspections find more faults can be drawn. But there is another factor to consider. The standard deviation and standard error is a little bit higher for inspections, and the 95% confidence interval is also a little wider than for the technical review. Therefore the uncertainty about the mean for inspections value is a little higher. But the uncertainty for the mean value for technical review is also quite high.

Histogram

To get a graphical overview of the number of faults found, the data is also presented in histograms. There is one histogram for each verification method. Total number of faults is all the faults found by a subject in all the design diagrams together.

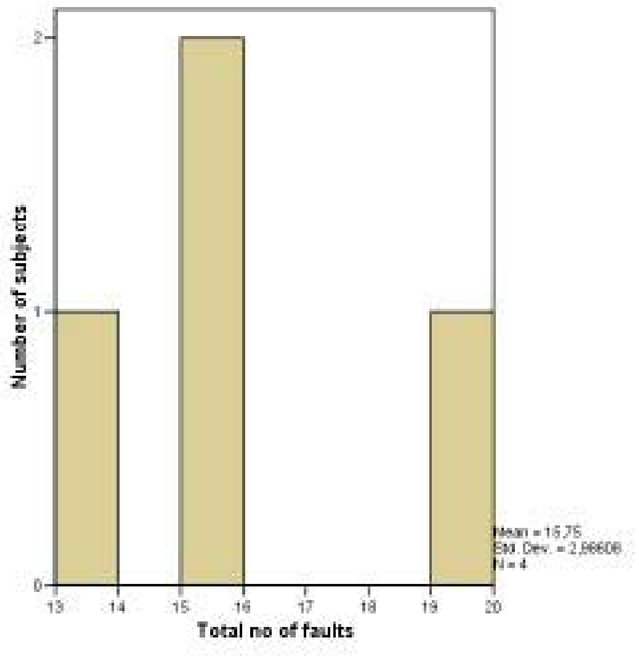


Figure 4-5

The first method presented is Technical review (see Figure 4-5). Three of the subjects have found between 13-16 faults and one subject has found between 19-20 faults.

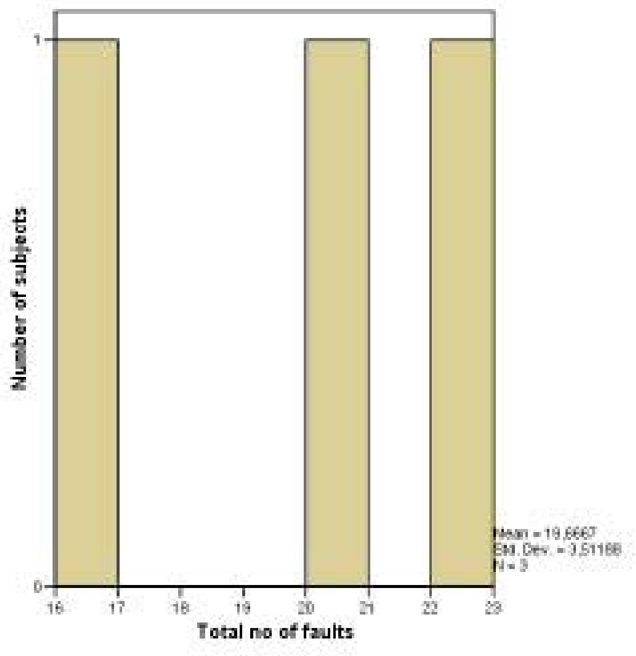


Figure 4-6

The second histogram is from the perspective based inspection method (see Figure 4-6). This method had only three subjects and the number of found faults is spread out in this case too. But the result is better than for technical review. Only one subject found between 16-17 faults and the other subjects found between 20 and 23 faults.

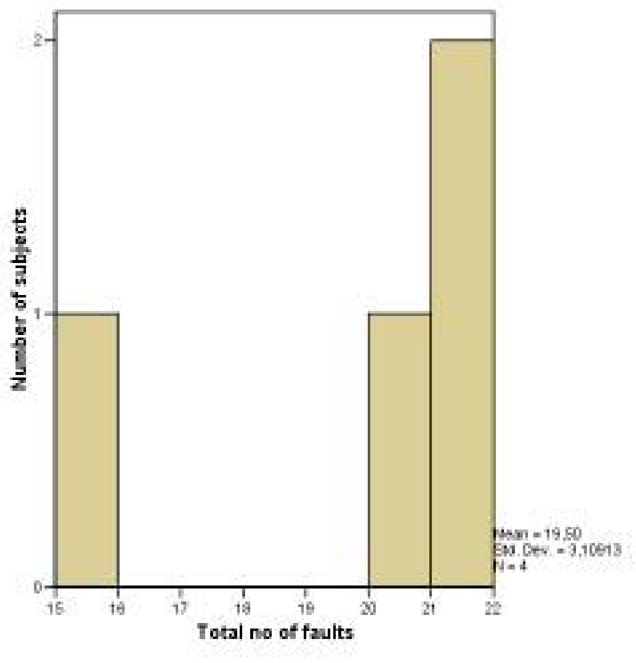


Figure 4-7

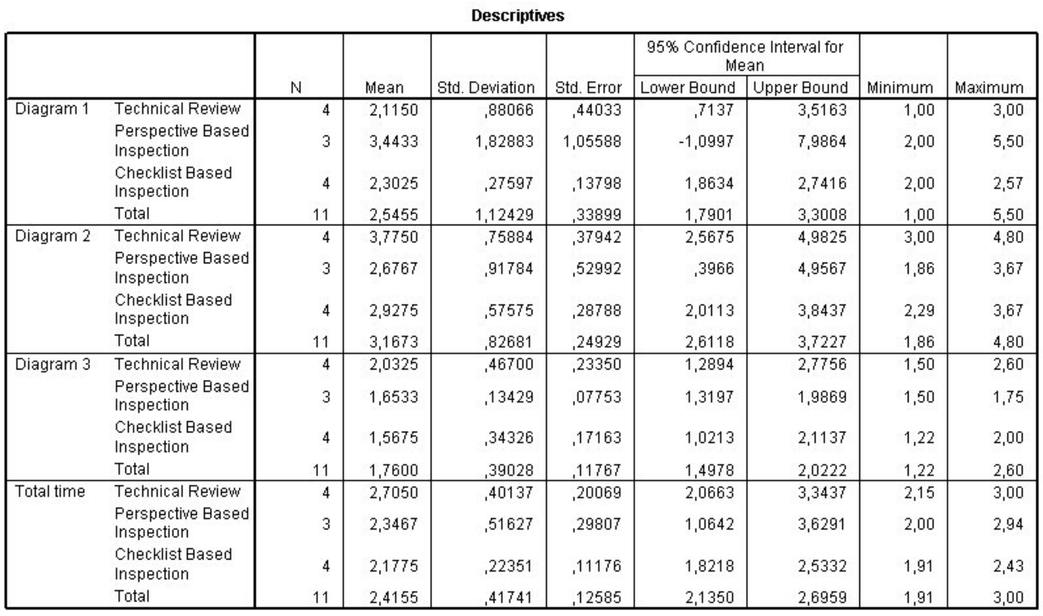
The third histogram is from the checklist based inspection method (see Figure 4-7). As for the perspective based inspection, here is also one subject that has found few faults (between 15-16 faults). Three subjects have found between 20-22 faults, which is almost as good as for the perspective based inspection.

The big difference is found between technical review and the inspection methods. The lowest number of found faults is found in the technical review result and the best result for technical review is only as good as the middle values for the inspection methods. There is very little difference between the inspection methods.

4.3.2.4 Mean time to find faults – Descriptive data

This part shows the descriptive data for the mean time used to find the faults. The time used is the time it took for the subjects to find the one actual fault. No false negatives are counted or timed.

These are the descriptive data for the mean time to finds fault calculation returned from SPSS (see Table 4-6).



Descriptives

		N	Mean	Std. Deviation	Std. Error	95% Confidence Interval for Mean		Minimum	Maximum
						Lower Bound	Upper Bound		
Diagram 1	Technical Review	4	2,1150	,88066	,44033	,7137	3,5163	1,00	3,00
	Perspective Based Inspection	3	3,4433	1,82883	1,05588	-1,0997	7,9864	2,00	5,50
	Checklist Based Inspection	4	2,3025	,27597	,13798	1,8634	2,7416	2,00	2,57
	Total	11	2,5455	1,12429	,33899	1,7901	3,3008	1,00	5,50
Diagram 2	Technical Review	4	3,7750	,75884	,37942	2,5675	4,9825	3,00	4,80
	Perspective Based Inspection	3	2,6767	,91784	,52992	,3966	4,9567	1,86	3,67
	Checklist Based Inspection	4	2,9275	,57575	,28788	2,0113	3,8437	2,29	3,67
	Total	11	3,1673	,82681	,24929	2,6118	3,7227	1,86	4,80
Diagram 3	Technical Review	4	2,0325	,46700	,23350	1,2894	2,7756	1,50	2,60
	Perspective Based Inspection	3	1,6533	,13429	,07753	1,3197	1,9869	1,50	1,75
	Checklist Based Inspection	4	1,5675	,34326	,17163	1,0213	2,1137	1,22	2,00
	Total	11	1,7600	,39028	,11767	1,4978	2,0222	1,22	2,60
Total time	Technical Review	4	2,7050	,40137	,20069	2,0663	3,3437	2,15	3,00
	Perspective Based Inspection	3	2,3467	,51627	,29807	1,0642	3,6291	2,00	2,94
	Checklist Based Inspection	4	2,1775	,22351	,11176	1,8218	2,5332	1,91	2,43
	Total	11	2,4155	,41741	,12585	2,1350	2,6959	1,91	3,00

Table 4-6

Table data:

The descriptions for the values can be found in 4.3.2.3, and only the differences are mentioned here.

The first column: The diagrams are the same as in 4.3.2.3. Total time is the total mean time used to find faults in the design diagrams (only the time to find actual faults is used).

Mean: The mean value for time used to find one fault.

Std. Deviation: For this experiment, this means for example that in 69% of the cases, the total mean time to find a fault (mean = 2.4155) is between 1.99809 minutes and 2.83291 minutes.

Descriptive data conclusions:

In most cases, the subjects spend more time to find a fault when using technical review than when using an inspection method. Unless for diagram 1 (class diagram), but that is most likely due to the low number of faults found by this method. By looking at the total time, a ranking can be made of the methods according to effectiveness. The best method is checklist based inspection followed by perspective based inspection and the least efficient method is technical review.

The standard deviation and the standard error are lower than for the number of fault calculation. Therefore it can be assumed that the mean values for time usage is more accurate and a better way to assess the verification methods. The 95% confidence interval for mean is also narrower.

Histogram

To get a graphical overview of the mean time to find a fault, the data is also presented in a histogram. There is one histogram for each verification method. The mean time is calculated on the total time that each subject spent to find all the real faults in all the design diagrams together.

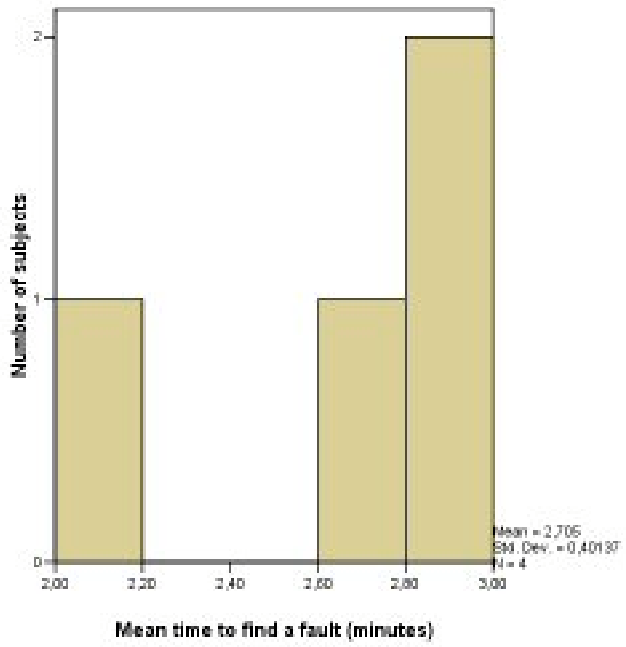


Figure 4-8

The first histogram is for the technical review. Most of the subjects spent between 2.60-3 minutes per fault. One subject spent only between 2-2.20 minutes.

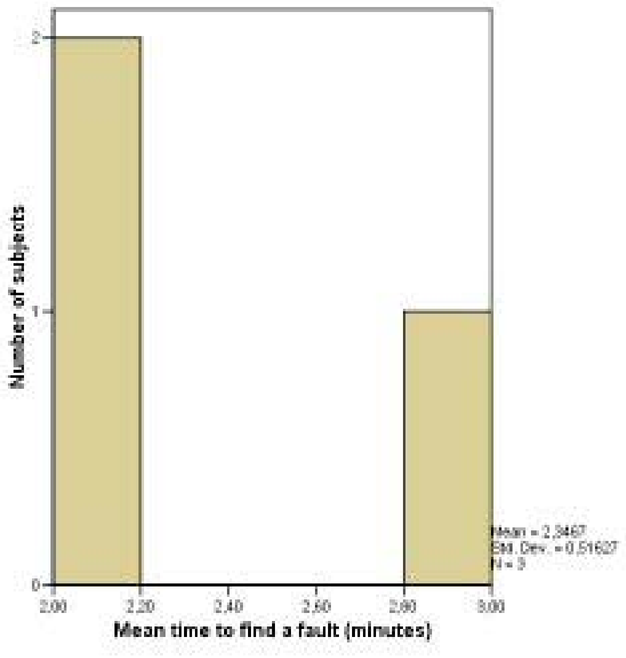


Figure 4-9

The second histogram is for the perspective based inspection. Here the result is almost opposite to the technical review. Most of the subjects spent between 2-2.20 minutes per fault and only one subject spent between 2.80-3 minutes per fault.

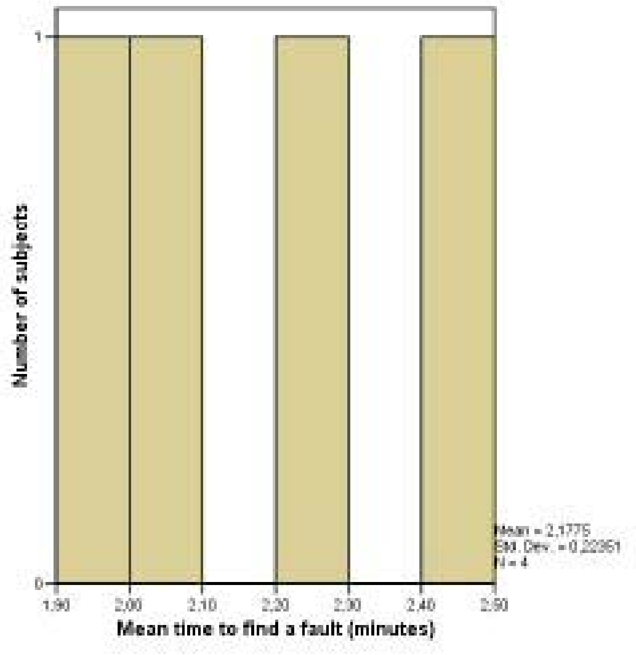


Figure 4-10

The final histogram is for the checklist-based inspection. The values are quite spread out for this verification method. But the highest value is between 2.40-2.50 minutes per fault, which is better than for the other methods. The lowest value is between 1.90-2 minutes, which is also better.

A pattern can be found here. The subjects using technical review uses the most time to find a fault. Most of the subjects are using quite a lot of time. For the perspective based inspection, most of the subjects are using less time, which puts this method in second place. For the most effective method, checklist based inspection the results are spread out. But since the subjects using this method uses less time in general, checklist based inspection is the most effective method (considering time to find a fault).

4.3.3 Data set reduction

In this part a scatter plot (for both numbers of faults and mean time to find faults) is drawn to find any outliers in the data set received from the experiment. Because the methods used for statistically testing the hypotheses depends on the quality of the input data, a scatter plot is an effective way to find subjects that have returned values that are very different from the rest of the subjects. When the scatter plot is done and the outliers are found, a decision has to be taken on what to do with the outliers. If the reason for the outlier is a rare and random event, then the outlier can be removed.

The scatter plot for this experiment is like this (see Figure 4-11):

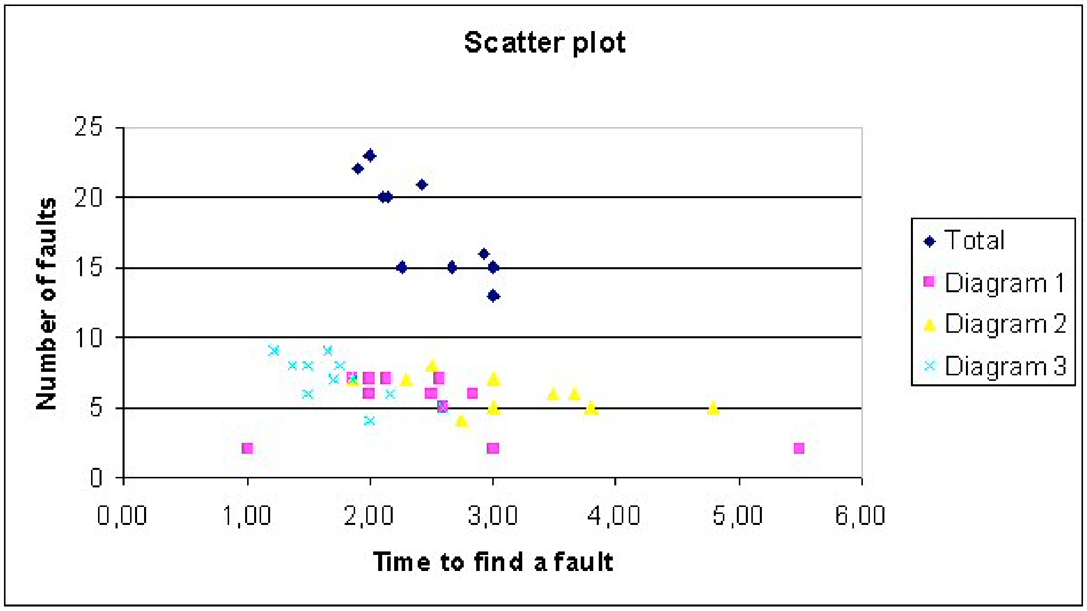


Figure 4-11

Diagram 1 (class diagram), diagram 2 (collaboration diagram without stereotypes) and diagram 3 (collaboration diagram with stereotypes) are the diagrams used in the study.

There are two outliers that are quite easy to spot and one that might not be considered an outlier. One subject has found very few faults and spent a long time (about 5.5 minutes) for each fault when verifying diagram 1. Another subject found few faults in the same diagram and spent only one minute per fault. In diagram two, there is also one outlier, namely the subject that spent almost five minutes per fault. By looking at the experiment data (see Appendix A10) and do a comparison between data and the outliers gives the result that the outliers might not be strange events and therefore the outliers must remain in the analysis. Two outliers (diagram 1 – one minute and diagram 2 – almost 5 minutes) are both from the technical review. The outlier where more than five minutes are spent on each fault is the result of perspective based inspection. The reason for the outliers could be that a more general verification method causes more outliers due to its generality.

Due to this the outliers is not removed. The outliers are not the result of a strange and random event. The reason is that the methods are not equally generic

4.3.4 Hypothesis testing:

The objective for this testing is to see if it is possible to reject the null hypothesis with as high significance as possible, based on the data from the experiment. The objective is also to verify the results from the descriptive analysis and test if those results are significant.

ANOVA calculation: The hypothesis is tested with a mathematical method called ANOVA and an example on how to make calculations according to the ANOVA test can be found in [Wohlin et al. p105]. In this case SPSS [SPSS] is used to make the calculations. ANOVA is based on the variability of the data according to different components. In this case the test compares the variability according to the treatments.

Two calculations are needed since there are two hypotheses, one for number of faults found and one for the mean time to find faults.

4.3.4.1 Number of faults found

The input is the data gathered during the experiment, in this case the number of faults found by each subject for each verification method (see Appendix A10).

The result of the One-way ANOVA calculation is as follows (see Table 4-7):

ANOVA

		Sum of Squares	df	Mean Square	F	Sig.
Diagram 1	Between Groups	12,636	2	6,318	1,532	,273
	Within Groups	33,000	8	4,125		
	Total	45,636	10			
Diagram 2	Between Groups	4,250	2	2,125	1,744	,235
	Within Groups	9,750	8	1,219		
	Total	14,000	10			
Diagram 3	Between Groups	6,333	2	3,167	1,288	,327
	Within Groups	19,667	8	2,458		
	Total	26,000	10			
Total number of faults	Between Groups	37,220	2	18,610	1,851	,218
	Within Groups	80,417	8	10,052		
	Total	117,636	10			

Table 4-7

Table data:

Sum of squares: This is the total amount of variability in the response, the sum of the squared differences between each observation and the overall mean.

Df: The total degrees of freedom is one less than the number of subjects. One is the number of levels and the other is the difference between number of levels and total degree of freedom.

Mean square: These are the sum of squares divided by the corresponding degrees of freedom.

F: This is the test statistic used to decide if the sample means are within sampling variability of each other. F is the ratio between the model mean square and the error mean square. If the F ratio is large, the null hypothesis is rejected.

Sig: If this value is low (at least below 0.1), then the null hypothesis can be rejected with high significance. The higher this value is, the higher is the risk that the hypothesis is wrongly rejected.

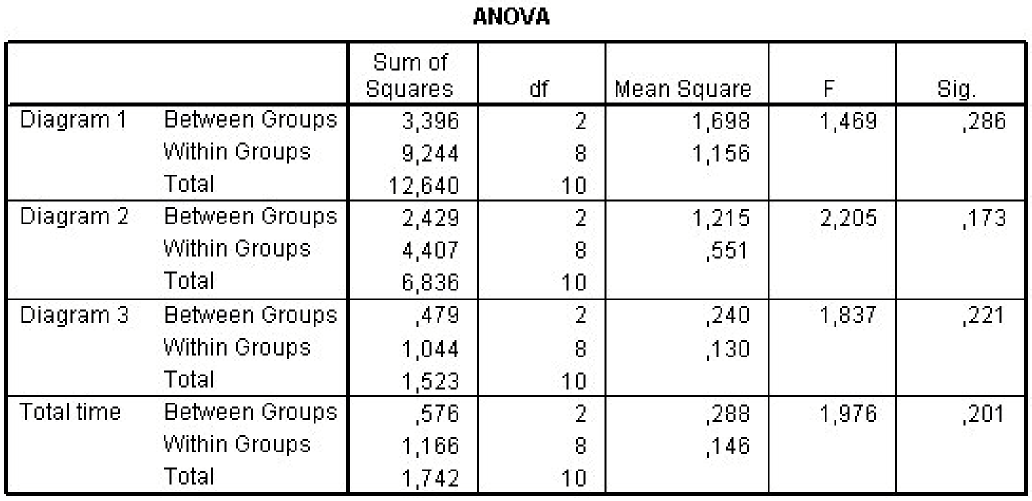
Analysis conclusion:

The sig value is high (higher than 0.1), due to a large overlap between the methods. This gives a large uncertainty when rejecting the hypothesis (that the methods find equally many faults). By looking at the descriptive statistics, a difference is clearly noticeable. But due to the large variability between the values, the uncertainty when rejecting the hypothesis is high. Therefore, the null hypothesis cannot be rejected. It seems that the verification methods are equally good at finding faults. One reason for this is probably the low number of subjects used in the experiment, which gives a large variability in the input data.

4.3.4.2 Time to find faults

The input data is the mean times used to find a fault in the diagrams. The mean time is calculated for each subject and diagram. The number of faults used in this calculation is the total number of real faults found per subject and diagram.

The result of the One-way ANOVA calculation is as follows (see Table 4-8):



ANOVA

		Sum of Squares	df	Mean Square	F	Sig.
Diagram 1	Between Groups	3,396	2	1,698	1,469	,286
	Within Groups	9,244	8	1,156		
	Total	12,640	10			
Diagram 2	Between Groups	2,429	2	1,215	2,205	,173
	Within Groups	4,407	8	,551		
	Total	6,836	10			
Diagram 3	Between Groups	,479	2	,240	1,837	,221
	Within Groups	1,044	8	,130		
	Total	1,523	10			
Total time	Between Groups	,576	2	,288	1,976	,201
	Within Groups	1,166	8	,146		
	Total	1,742	10			

Table 4-8

Table data:
The descriptions for the table data can be found in 4.3.4.1. Only the differences are described here.

Analysis conclusion:
The sig value is high (higher than 0.1) for this calculation too, due to a large overlap between the methods. The value is a little lower than for the previous calculation (number of faults). But the sig value is still too high to be able to reject the hypothesis with a good significance. It seems that the methods require the same mean time to find one fault. One reason for this is probably the low number of subjects used in the experiment, which gives a large variability in the input data.

4.4 Conclusions

This part contains the conclusions draw during the experiment.

The experiment operation went well. The number of subjects was quite good considering that the experiment was voluntary. Though, to get a high statistical power, the number of subjects should have been much higher. The subjects showed up in time and took the experiment seriously.

The result of experiment analysis is not as good as expected. The goal was to find a difference between the different analysis methods. The verification methods should be ranked considering number of faults found and the mean time to find a fault.

By using descriptive statistic the preferred result is achieved, which is to see that the verification methods are not equally effective. By looking at the descriptive statistic, the conclusion that the inspections find more faults and are more time-efficient can be drawn. The difference between the inspection methods is very small.

The hypothesis testing did not give the desired result. According to the significance test (ANOVA) the hypotheses cannot be rejected with a satisfactory significance. The sig value (table 4-7 and table 4-8) should at least be below 0.1, but in this study, none of the sig values are that low. The reason for this is the large variability in the input data, which probably depends on the low number of subjects in the experiment.

The aim of this experiment is the compare three different design verification methods in the context of OO software design. The goal is also to be able to rank the verification methods after the study. The descriptive statistic gives data to do this ranking, but the hypotheses testing do not support rejecting the hypotheses due to the high sig values. So the methods are compared but the result has too low significance to be really useful.

5 METHOD GUIDELINES AND ASSESSMENT

In this chapter guidelines are given and the verification methods are assessed. The methods are ranked according to different factors, like time to find faults and number of faults found. The result from another experiment concerning perspective based inspection and checklist based inspection is also presented and compared with the result of this experiment. The methods are also assessed according to factors found in the literature study, like complexity and reusability. The assessment and ranking is based on the literature studies and the descriptive data from the experiment. Since the hypotheses could not be rejected, the significance of the ranking is quite low.

5.1 Method ranking

The experiment is not able to statistically (ANOVA) reject the hypothesis (that the methods compared are equally good at finding faults, and equally effective at finding faults) with high enough confidence. To be able to rank the methods, other factors also have to be considered. Time to find a fault, number of faults found, number of outliers, complexity and reusability are considered.

5.1.1 Time to find faults

This table (see Table 5-1) shows the mean time spent per fault found in the experiment. The value is the mean value for all the subjects using a particular verification method, and is calculated on the time each subject has spent to find all the valid faults in one diagram.

Method	Total	Diagram 1	Diagram 2	Diagram 3	Mean
Technical review	2.71	2.12	3.78	2.03	2.66
Perspective based inspection	2.35	3.44	2.68	1.65	2.53
Checklist based inspection	2.18	2.30	2.93	1.57	2.25

Table 5-1

By using the mean value from this table (which is the mean value of total, diagram 1,2 and 3), the methods should be ranked like this. The best method is checklist based inspections, followed by perspective based inspections and the last method is technical review. The inspections are quite equal and better than technical review in most cases. Diagram 1 is an exception, but that result is most likely due to the high number of outliers.

5.1.2 Number of faults found

This table (see Table 5-2) shows how many faults each method found during the experiment (see 4.3.2.3.). The values are the mean value for all the subjects using a particular verification method.

Method	Total	Diagram 1	Diagram 2	Diagram 3	Mean
Technical review	15.75	4	5.75	6	7.875
Perspective based inspection	19.67	5	7	7.67	9,835
Checklist based inspection	19.5	6.5	5.5	7.5	9.75

Table 5-2

By using this factor the methods should be ranked in the following order (according to the mean value. Which is the mean of total, diagram 1, 2 and 3): Perspective based inspection, checklist based inspection and technical review. In this case too, the difference between the two inspection methods is marginal, while technical review is falling behind a bit.

5.1.3 Similar study

This part describes a similar experiment [Sabaliauskaitea et al. 2003]. In this experiment perspective based inspection (PBR) is compared with checklist based inspection (CBR). The design used is UML based and object-oriented, just like in this experiment. But this experiment is much larger with many more subjects, which is not a bad thing. Because it is so much larger, the confidence in the conclusions is much more accurate.

Experiment setup:

The variables measured in both experiments are number of faults found and time spent per fault.

Hypotheses (for the single person inspection) [Sabaliauskaitea et al. 2003]:

"H01: Subjects spend more time on inspection using PBR than using CBR.

H02: Cost per defect of subjects who use PBR is lower than the cost per defect of subjects who use CBR.

H03: There is no difference in defect detection effectiveness of subjects who use PBR inspection technique as compared to subjects who use CBR."

For the simulated 3 person inspection teams, the following hypothesis is stated [Sabaliauskaitea et al. 2003]:

“H04: There is no difference in defect detection effectiveness of the 3-person simulated teams, which use PBR inspection technique as compared to the 3-person simulated teams, which use CBR inspection technique.”

Design, subjects and objects:

The experiment type is 'one factor with two treatments'. The factor is the design and the treatments PBR and CBR. There are 59 subject with computer science education, and they are divided into two groups (one for PBR and one for CBR, depending on their marks from a program design class). There were also more objects, namely one set of 24 pages and one set of 18 pages. The objects were use cases, activity diagrams, class diagrams, component diagrams and sequence diagrams. The time period was quite large. It took three weeks to complete the study.

At the beginning of the experiment a training session was held to improve the subjects’ understanding of the software systems. Subjects using CBR had to inspect all diagrams, while the subjects using PBR only had to inspect those relevant to their perspective.

Experiment result:

This is a part of the result found in the experiment [Sabaliauskaitea et al. 2003: 581-582].

1. Subjects using PBR spent less time on inspection, because the subjects only had to inspect the documents for their perspective.
2. The cost per defect was lower for the subjects using CBR.
3. When using an inspection team, the CBR technique was more effective.

Compare experiment result with this experiment

The results are compared number for number. Number one for the similar experiment versus number one for this experiment.

1. The times spent on inspection in this experiment were the same for all methods and therefore this conclusion is not comparable.
2. Both experiments have the conclusion that CBR is more effective when it comes to finding faults. This is applicable both for single person inspection teams (both experiments) and for small inspection teams [Sabaliauskaitea et al. 2003: 581].
3. By looking at the descriptive statistic, the same result is found in this experiment. But if ANOVA is used then the same result is not achieved in this study due to high sig value.

5.1.4 Number of outliers

A large number of outliers could indicate that the verification method is hard to understand and the result is not very accurate. There are three outliers in the experiment data. Two of them occurred when verifying diagram 1 and one when verifying diagram 2. One outlier for the perspective based inspection and two outliers for the technical review. This gives the following ranking; checklist based inspection, perspective based inspection and technical review.

5.1.5 Complexity and reusability

In this ranking FTA is also included, because this part not dependent on the experiment data. This part is dependent on literature studies.

Complexity:

This factor focuses on how much preparation that is required and how much training (if any) the subjects need to be able to understand the method. It also considers how many steps that have to be performed during the inspection and if any tools are required. The ranking is as this (from least complex to most complex) and is based on literature studies:

- FTA. This method is very complex and a lot of preparation is needed before the design verification (see 2.3). Software primitive objects (SPO) must be defined and modes and methods must be defined for each SPO. Safety assertions and a reachability graph must be generated. When all this is done it is time to look for faults.
- Technical review, perspective based inspection and checklist-based inspection is basically equally complex. Checklist based inspection is the easiest method to use, but it requires most preparation. Perspective based inspection and technical review are basically equally easy to use, but perspective based inspection requires a little more preparation.

Reusability:

This factor focuses on how easy it is to reuse the method, the material and instructions in another design inspection. A more general method requires less rework before it can be used in another design verification. Therefore the ranking is as follows (from most reusable to least reusable) and is based on literature studies:

- Technical review. This is the most generic method, since the verification instructions and questions are generic. Therefore, this method can be used without modification at least throughout the whole project or even in many projects.
- Perspective based inspection. This method is less generic, since the instructions and questions have to be adapted to a certain stakeholders' aspect. It can be used throughout the whole project, but it has to be adapted to the different roles used in the inspection.
- Checklist based inspection. This method is the least generic, since it has to be adapted to every type of document used in the project. It is also bound to that project, and often quite a lot of modification is required to use the checklists in another project.
- FTA. This is the least reusable method of the ones in this thesis. The software primitive objects and the product behavior specification are connected to a specific product. Also the safety assertions are product specific.

5.1.6 Discussion of guidelines

This part discusses and summarizes the guidelines and assessment for the design verification methods.

The hypotheses testing could not reject the hypotheses with good enough significance. Therefore, the hypotheses could not be used for ranking the methods. Factors that have been used are the number of faults found, mean time to find a fault, outliers and complexity/reusability.

FTA: This method seems to be very good at finding faults. But this comes with a high price. It requires a lot of training before using this method and it also requires quite a lot of time and preparation during the analysis.

This method could be applicable when verifying design for safety critical systems.

Technical review: This method finds least faults of the methods studied in the experiment (according to the descriptive statistics), and it takes a little more time to find the faults. It is also the most general method and is therefore easier to reuse. The preparation time is the lowest.

This method could be applicable when a general verification method is required and when there is no time to prepare material for specific design diagrams or projects. This method can be developed and prepared by the quality department and then used for many projects and inspections.

Perspective based inspection: This method finds the most faults of the methods studied in the experiment (according to the descriptive statistics). It is second best when it comes to time per fault.

This method could be applicable if a quite generic method is desirable, which also is effective and finds many faults. This method requires preparation for the different perspectives, but it is quite reusable.

Checklist based inspection: This method finds almost as many faults as the perspective based inspection, and this method is also a little more effective (according to the descriptive statistics). The disadvantage is that this method requires more preparation time and is not especially reusable (compared to technical review and perspective based inspection). The checklist's has to be prepared and they are specific for each document type.

6 FURTHER WORK

This chapter presents suggestions and ideas on how to move on from this thesis, how to improve the results in this thesis and how to derive new theses from this work.

6.1 Effect of stereotypes

This effect has only been touched in this thesis. In this case, stereotypes are used to get a higher statistical power through a higher effect size. Due to the fact that some faults are the same in the collaboration diagram without stereotypes and the collaboration diagram with stereotypes, it is not valid to draw conclusions from this study according the effect of stereotypes.

6.2 Effect of different factors and treatments

In this thesis the factor is the verification method and the treatments are technical review, perspective based inspection and checklist based inspection.

For further work it could be interesting to have the design diagrams as a factor too (or instead of the verification method). This also includes the effect of stereotypes.

6.3 Replication of this experiment

A replication of this experiment with more subjects would give more significant results. The statistical power would also be higher. At least 15-20 subjects per verification method would be a good approach. In a similar study [Sabaliauskaitea et al. 2003: 581], was the number of subjects per method were 29 and 30.

6.4 Replication and capture-recapture

This experiment can be replicated as it is and the capture-recapture method can be applied to calculate the number of faults remaining in the design.

7 THESIS CONCLUSION

The earlier in the development process a fault is discovered, the cheaper it is to correct the fault. Verification and validation methods are a way of finding faults in the software design.

The methods presented in this thesis are review (focus on technical review), inspections (focus on perspective based inspection and checklist based inspection) and FTA.

Technical review, perspective based inspection and checklist-based inspection is studied further in an experiment. FTA is not included in the experiment due to the large difference in complexity compared to the other methods. In the experiment, three design diagrams (one class diagram, one collaboration diagram without stereotypes and one collaboration diagram with stereotypes) are verified by 11 subjects. The subjects were divided into three groups; one for each verification method and each subject verified all three diagrams.

The result of the experiment is divided into two parts. The descriptive statistics indicates that there is a difference in the number of faults found and time spent per fault between the review and the inspection methods. The difference between the two inspection methods is very small. The data is also tested statistically with ANOVA, and the result of this test indicates that there is no significant difference between the verification methods.

The verification methods have also been assessed, ranked and recommendations are presented. The descriptive statistics and literature studies have mainly been used for this ranking.

FTA: Finds many faults, but requires a lot of preparation and training before the verification.

Technical review: This method finds the least faults and requires most time per fault. This method is the most reusable and requires little training and preparation.

Perspective based inspection: This method finds the most faults and is second best when it comes to time per fault. It requires preparation time and training due to the different perspectives, but it is quite reusable.

Checklist based inspection: This method is almost as good as perspective based inspection at finding faults and this method finds the faults faster. The preparation time is higher and the prepared material is not so easy to reuse.

8 REFERENCES

[Archambault 2000] Susan Archambault, Psychology Department, Wellesley College, < http://www.wellesley.edu/Psychology/Psych205/anova.html > (040518)

[Aurum et al. 2001] Aybuke Aurum, Håkan Petersson and Claes Wohlin (2001) 'State-of-the-Art: Software Inspections after 25 Years', October 12, 2001, pages 1-24

[Basili et al. 1996] Basili V. R, Green S, Laitenberger O, Lanubile F, Shull F, Sörumgård S, Zelkowitz M, (1996): 'The Empirical Investigation of Perspective-Based Reading'. International Journal on Empirical Software Engineering, 1(12), pages 133-144

[Bisant and Lyle 1989] Bisant, D. B. and Lyle, J. R, 'A Two Person Inspection Method to Improve Programming Productivity'. IEEE Transactions on Software Engineering, 15(10), pages 1294-1304

[Burnham and Overton 1978] K.P. Burnham and W.S. Overton, 'Estimation of the size of a closed population when capture probabilities vary among animals', Biometrica, vol.65, 1978, pages 625-633

[Easton and McColl 1997] Valerie J. Easton and John H. McColl, 'Statistics Glossary', 1997, <http://www.stats.gla.ac.uk/steps/glossary/confidence_intervals.html#confinterval> (2004-04-28)

[Fagan 1976] Fagan, M. E, 'Design and Code Inspections to Reduce Errors in Program Development', IBM Systems Journal, 15(3), pages 182-211.

[Fagan 1986] Fagan M. E, (July 1986): 'Advances in Software Inspections'. IEEE Transactions on Software Engineering, 12(7)

[Fukaya et al. 1994] Tetsuji Fukaya, Masayuki Hirayama and Yukihiro Mihara (1994) 'Software Design Verification using FTA', IEEE, pages 208-213

[Gilb and Graham 1993] Gilb, T. and Graham, D, 'Software Inspection', Addison Wesley Publishing Company, ISBN 0-201-63181-4

[IEEE 1998] Software Engineering Standards Committee (1997) 'IEEE Standard for Software Reviews', IEEE Std 1028-1997, ISBN 1-55937-987-1, pages 1-37

[Knight and Myers 1993] Knight, J. C. and Myers, A. E, 'An Improved Inspection Technique'. Communications of ACM, 36(11), pages 50-69

[Kusumoto et al. 1996] Shinji Kusumoto, Tohru Kikuno, Ken-ichi Matsumoto and Koji Torii, 'Experimental Evaluation of Time Allocation Procedure for Technical Reviews', Elsevier Science Inc, 1996, pages 119-126

[Körner, Wahlgren 2000] Svante Körner and Lars Wahlgren (2000), *Statistisk data-analys*, Studentlitteratur, Sweden, pages 408

[Laitenberger and DeBaud, 1997] Laitenberger, O. and DeBaud, J. N, 'Perspective-based Reading of Code Documents at Robert Bosch GmbH', Information and Software Technology, 39, pages 781-791

[Linger et al. 1979] Linger, R. C, Mills, H. D, Witt, B. I, (1979): Structured Programming: Theory and Practice. Addison Wesley

[Martin and Tsai 1990] Martin J, and Tsai W. T, (February 1992): 'N-Fold Inspection: A Requirements Analysis Technique'. Communications of ACM, 33(2), pages 225-232

[OMG] 'UML™ Resource Page', Object Management Group Inc, <http://www.omg.org/technology/uml/index.htm> (040520)

[One-Way ANOVA] Dr Ebenge Usip, Toungstown State University, 1996, <http://cc.ysu.edu/~eeusip/anova.htm> (2004-04-19)

[Parnas and Weiss 1985] Parnas, D. L. and Weiss, D. M, 'Active Design Reviews: Principles and Practices'. Proceedings of ICSE'85, (London, England, Aug 28-30), IEEE Computer Society, pages 132-136

[Porter and Votta 1994] Porter, A. A. and Votta, L. G. (1994): 'An Experiment to Asses Different Defect Detection Methods for Software Requirements Inspections'.

Proceedings on 16th International Conference on Software Engineering, ICSE-16, pages 103-112

[Porter et al. 1995] Porter, A. A., Votta, L. G., Basili, V. (1995): 'Comparing Detection Methods for Software Requirements Inspection: A Replicated Experiment'. IEEE Transaction on Software Engineering, 21(6), pages 563-575

[Sabaliauskaitea et al. 2003] Giedre Sabaliauskaitea, Fumikazu Matsukawab, Shinji Kusumotob, Katsuro Inoue, 'Further investigations of reading techniques for object-oriented design inspection', Osaka University, 2003, Elsevier Science B.V, pages 571-585

[Scott et al. 1993] Scott A, Vander Wiel and Lawrence G Votta (1993) 'Assessing Software Designs Using Capture-Recapture Methods', IEEE Vol 10, no 11, pages 1045-1054

[Shull et al. 2000] Shull F, Rus I, Basili V, (July 2000): 'How Perspective-Based Reading Can Improve Requirements Inspection', IEEE, Computer, pages 73-79

[SPSS] SPSS Inc, <http://www.spss.com/> (2004-05-05)

[Trochim] William M.K. Trochim, Cornell University, 2002, <http://trochim.human.cornell.edu/kb/intval.htm> (2004-04-22)

[Votta 1993] Votta, L. G, 'Does Every Inspection Need a Meeting?' Proceedings of ACM Symposium on Software Development Engineering, Reprinted in Software Inspection: An Industry Best Practice, IEEE, Computer Society Press, USA, (1996). ISBN 0-8186-7340-0

[Wohlin et al. 2000] Claes Wohlin, Per Runesson, Martin Höst, Magnus C Ohlson, Björn Regnell. Anders Wesslén, 'Experimentation in Software Engineering. An Introduction', Kluwer Academic Publishers, 2000, pages 204

[Yourdon 1989] Yourdon, E, 'Structured Walkthroughs', 4th Edition, Prentice-Hall, Englewood Cliffs, N.J, 1989.

APPENDIX A1 (BACKGROUND QUESTIONS)

You shall answer these questions by giving a grade between 1-5.

The scale:
1 = very bad.
2 = Bad
3 = Average
4 = Good
5 = Very good

1: How is your knowledge about UML?

2: How is your knowledge about stereotypes?

3: How is your knowledge about technical reviews?

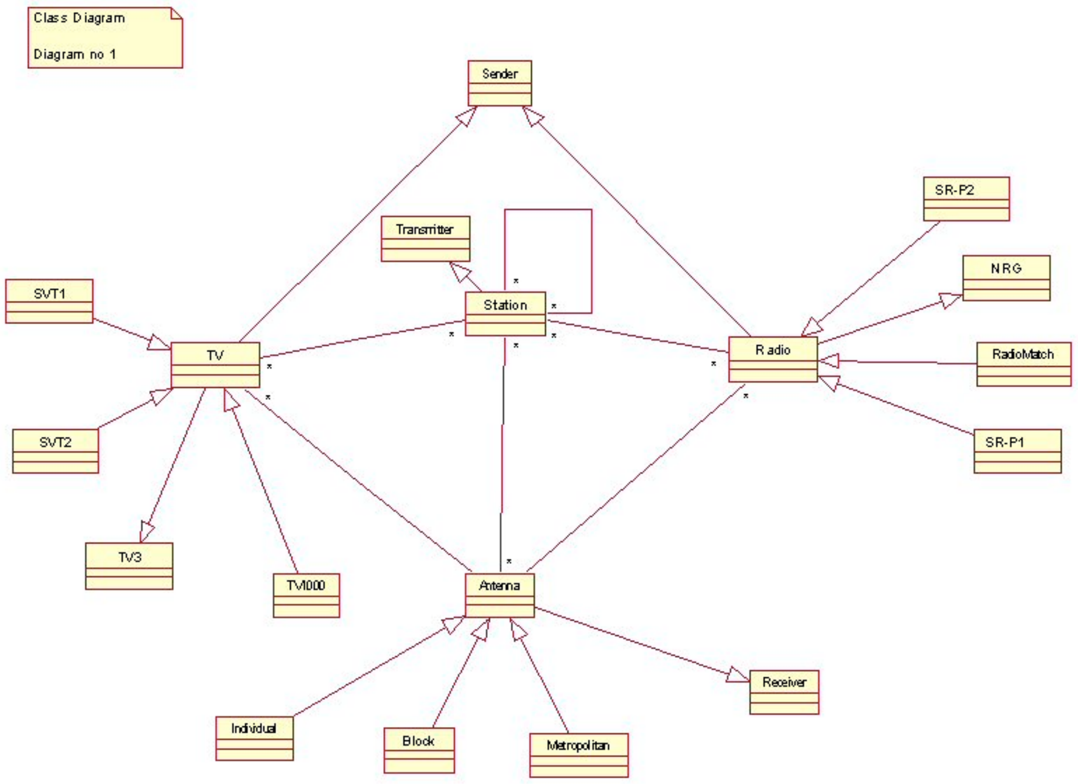
4: How is your knowledge about perspective based inspections?

5: How is your knowledge about checklist based inspections?

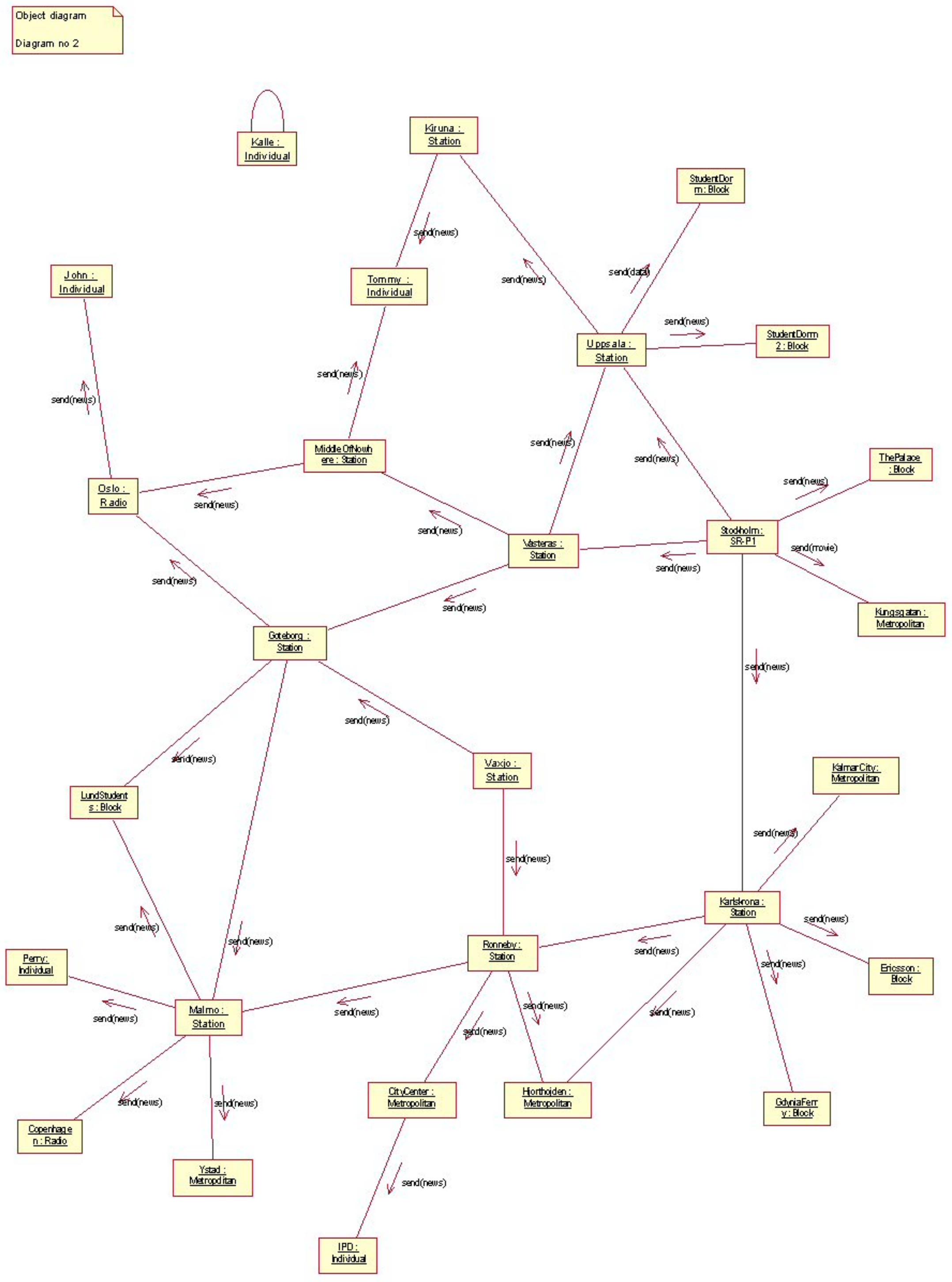
The result of the background questionnaire

Subject number	1. UML	2. ST	3. TR	4. PBI	5. CBI
1	5	5	3	2	2
2	3	1	4	5	5
3	4	3	3	2	3
4	2	2	3	2	3
5	3	2	3	2	4
6	3	1	1	1	2
7	3	2	4	3	4
8	3	2	1	1	1
9	3	2	3	2	3
10	4	4	3	3	3
11	3	1	4	3	4

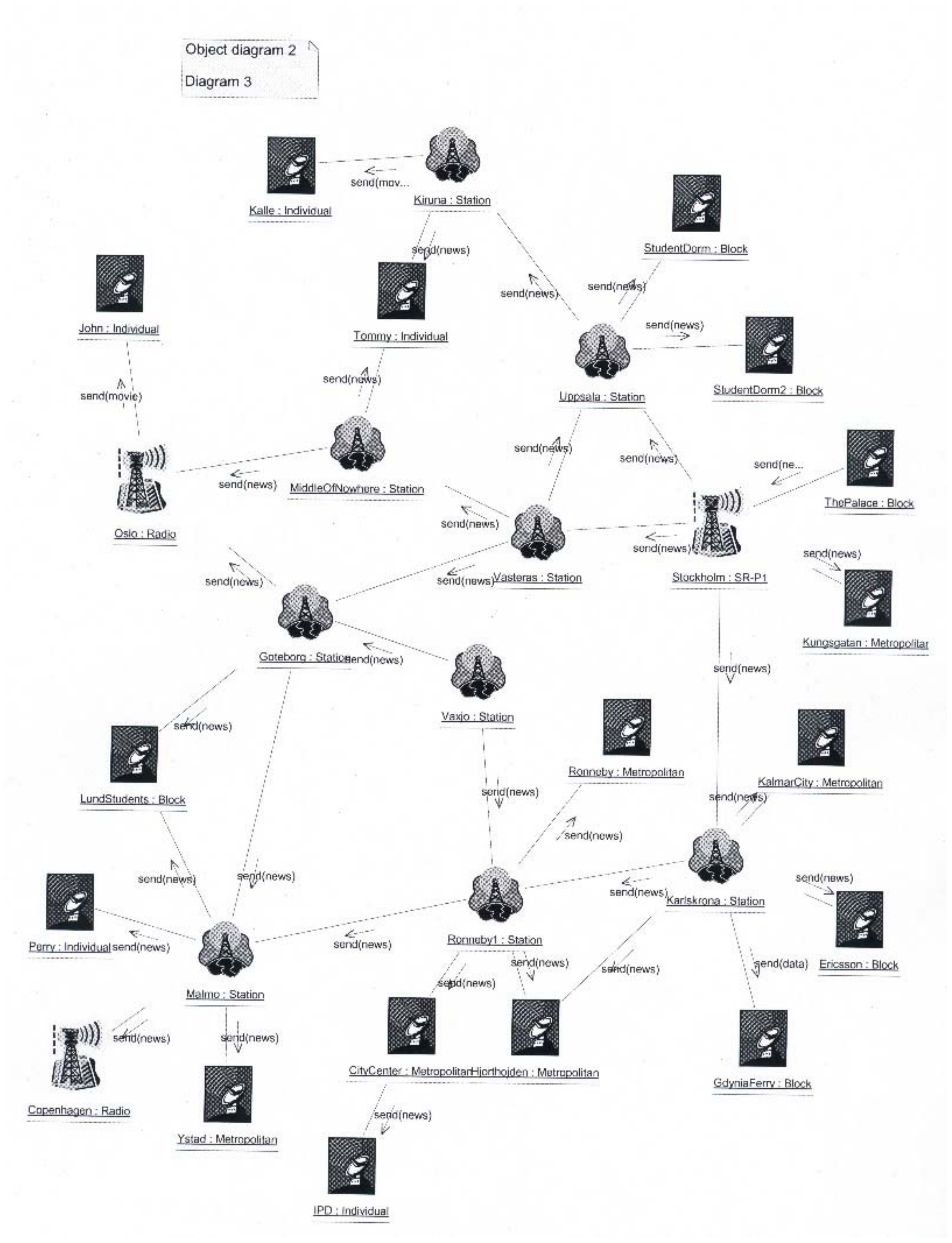
APPENDIX A2 (DESIGN DIAGRAM NO 1)



APPENDIX A3 (DESIGN DIAGRAM NO 2)



APPENDIX A4 (DESIGN DIAGRAM NO 3)



APPENDIX A5 (FAULT REPORT FORM)

Fault description form

Instructions:
The fields shall be filled in like this.

Fault no: This is the number of the fault. This number is already added to the fault description form.
Time: Here you shall enter the time when the fault was found (HH-MM).
Diagram no: The number of the diagram where the fault was found (1-3).
Type of fault: The kind of fault found. There are two categories (one fault can be part of more that one category. If so, then fill in all categories in the report form).

- Spec: The diagram is not compliant with the requirement specification. Examples are missing classes, misspelled class/object names.
- Logical: There are some relationships or classes/objects that are not logical. A logical fault is also often not compliant with the requirement specification. Examples are senders receiving, receivers sending or stations transmitting different data type than received.

Description of the fault: Here you shall describe the fault in your own words.

Fault no	Time	Diagram no	Type of fault	Fault Description
1				
2				
3				
4				
5				
6				
7				

APPENDIX A6 (REQUIREMENT SPECIFICATION)

Requirement specification for Radio and TV transmission system

1. Non-functional requirements

1.1. General requirements:
1.1.1. The requirements for a class also apply to the subclasses of that class.

1.2. Sender:
1.2.1. The sender can be a TV station or radio station.
1.2.2. The sender can only send messages.
1.2.3. The sender cannot receive any messages.
1.2.4. A sender cannot send messages to it self.

1.3. Receiver:
1.3.1. The receiver can be an antenna.
1.3.2. The receiver can only receive messages.
1.3.3. The receiver cannot send any messages.
1.3.4. A receiver cannot receive messages from it self.

1.4. Transmitter:
1.4.1. The transmitter can be a station.
1.4.2. The transmitter can receive messages.
1.4.3. The transmitter can send messages.
1.4.4. The transmitter can only transmit the same message that it received.
1.4.5. A transmitter cannot transmit a message to it self.

2. Functional requirements

2.1. Radio:

2.1.1. There are five types of radio.
- NRJ
- RadioMatch
- SR-P1
- SR-P2
- HIT-FM

2.1.2. A radio can only send radio messages like news and music.
2.1.3. A radio can send to many stations.
2.1.4. A radio can send to many antennas.

2.2. TV:
2.2.1. There are four types of TV.
- SVT1
- SVT2
- TV3
- TV4

2.2.2. A TV can only send TV messages like movies and sport.

2.2.3. A TV can send to many stations.
2.2.4. A TV can send to many antennas.

2.3. Antenna:

2.3.1. There are three types of antenna.

- Individual
- Block
- Metropolitan

2.3.2. An antenna can receive from many stations.
2.3.3. An antenna can receive from many TVs.
2.3.4. An antenna can receive from many radios.

2.4. Station:

2.4.1. There is one type of station.
2.4.2. A station can send to many other stations.
2.4.3. A station can send to many antennas.
2.4.4. A station can receive from many radios.
2.4.5. A station can receive from many TVs.

APPENDIX A7 (TECHNICAL REVIEW INSTRUCTIONS)

Introduction

The software artifacts shall be evaluated by a team of qualified personnel (you) to make sure they adheres to regulations, guidelines, plans and procedures that are applicable to the project. The software artifact shall also conform to its specification, which often is the main issue.
The input to this review includes the following:

- Review objectives
- The software artifacts
- Documented review procedure and instructions

Review objectives:
The objectives are to answer these questions:

- Does the design comply with the requirements specified in the requirement specification?
- Does the object diagrams comply with the class diagram and the requirement specification?

The software artifacts:
The software artifacts to examine are the following diagrams:

- Class diagram (diagram no: 1)
- Object diagram (diagram no: 2)
- Object diagram (diagram no: 3)

Instructions/Review procedure

This is how the review shall be conducted for each diagram.
Class diagram (diagram no 1):
Verify if the class diagram complies with the requirement specification.
Verify that class names, the relationships, multiplicity and inheritance are correct.

Object diagram (diagram no 2):
Verify that the object diagram complies with the class diagram, concerning object type and relationship.
Verify if the object diagram complies with the requirement specification.
Verify the message passing between the objects are correct and logical concerning parameters and direction according to the requirement specification.

Object diagram (diagram no 3):
Verify that the object diagram complies with the class diagram, concerning object type and relationship.
Verify if the object diagram complies with the requirement specification.
Verify the message passing between the stereotype objects are correct and logical concerning parameters and direction according to the requirement specification.

APPENDIX A8 (PERSPECTIVE BASED INSPECTION INSTRUCTIONS)

Introduction

This inspection has three stages. Preparation, collection (without a meeting) and rework. Some of the preparation is already completed and the rest takes place now. A part of the collection takes part when you fill in the fault report form and the rest is performed after this session, by the experiment leader. Usually a meeting is used in inspections to gather the information from the participants, but a meeting is not used in this experiment. The fault report form is used instead to collect the inspection data. Rework is not used in this experiment, since it is outside a development project.

Perspective based inspection:
This approach focuses on the needs and the point of the specific stakeholder and is an enhanced version of scenarios. Each scenario consists of a set of questions and is also based on the viewpoint of the stakeholder. The reviewers (you) shall read the document from the specified viewpoint.

Instructions

You shall read the diagrams from the designer's perspective and follow the steps described in the designer's scenario on the next page.

Designers scenario

Assume that you are one of the designers of the system. The concern for the designer is to make sure that the design is consistent with the requirement specification and to define a static structure of the system (class diagram). The designer must also verify that the object diagrams are consistent with the class diagram and don't violate the requirements.

Step 1	**Inspect the class diagram (diagram no 1)**
1.1	Are all the classes necessary to realize the functionality in the requirement specification defined in the class diagram?
1.2	Are the names of the classes correct according to the requirement specification?
1.3	Are all the necessary associations and inheritance between classes present?
1.4	Are the associations and inheritance between the classes correct?

Step 2	**Inspect the object diagram no 1 (diagram no 2)**
2.1	Are all the objects in the object diagram present in the class diagram?
2.2	Are the necessary relationships between the components defined?
2.3	Are the relationships correct and logical according to the class diagram?
2.4	Are the messages sent between the components correct according to the requirement specification?

Step 3	**Inspect the object diagram no 2 (diagram no 3)**
3.1	Are all the stereotype objects in the stereotype diagram present in the class diagram?
3.2	Are the necessary relationships between the components defined?
3.3	Are the relationships correct and logical according to the class diagram?
3.4	Are the messages sent between the components correct according to the requirement specification?

Appendix A9 (Checklist based inspection instructions)

Introduction

This inspection has three stages. Preparation, collection (without a meeting) and rework. Some of the preparation is already completed and the rest takes place now. A part of the collection takes part when you fill in the fault report form and the rest is performed after this session, by the experiment leader. Usually a meeting is used in inspections to gather the information from the participants, but a meeting is not used in this experiment. The fault report form is used instead to collect the inspection data. Rework is not used in this experiment, since it is outside a development project.

Checklist.

This is a systematic approach, which is based on that the reviewer (you) answer the questions in the checklist. The checklists will guide you through the review.

Instructions

There are some questions in the checklist. These questions shall be answered during the inspection of the design documents. You do not have to write down the answer to the questions, but the fault found must be written down in the fault report form.

Question explanation:

- Specification errors: Are there any inconsistencies between the requirement specification and the diagrams? Examples are spelling mistakes and missing classes.
- Logical errors: Example is a sender receiving, or a receiver sending. The logical errors are also inconsistent with the requirements.

The checklist

Class diagram (diagram no 1)

1	Are all classes present according to the requirement specification?	Yes	No
2	Are the names on the classes correct according to the requirement specification (no spaces, starts with a letter)?	Yes	No
3	Are the relationships and inheritance between the classes correct (no circular inheritance)?	Yes	No
4	Is the multiplicity between the classes defined correctly?	Yes	No
5	Are the classes inheriting from a more generic class (receiver, sender, transmitter) directly or indirectly?	Yes	No

Object diagram (diagram no 2)

1	Are the objects of a correct type according to the class diagram?	Yes	No
2	Are any senders (like a radio) receiving any messages?	Yes	No
3	Are any receivers sending any messages?	Yes	No
4	Does every station receive at least one message?	Yes	No
5	Is the parameter correct in the send function, according to the requirement specification? For example, is a radio sending movies?	Yes	No
6	Are there any connections missing?	Yes	No

Object diagram (diagram no 3)

1	Are the objects of correct type according to the class diagram?	Yes	No
2	Are any sender stereotypes (like a radio) receiving any messages?	Yes	No
3	Are any receiver stereotypes sending any messages?	Yes	No
4	Does every station stereotype receive at least one message?	Yes	No
5	Is the parameter correct in the send function, according to the requirement specification? For example, is a radio sending movies?	Yes	No
6	Are there any connections missing?	Yes	No

APPENDIX A10 (EXPERIMENT DATA)

Participants	Verification method	Total no of faults	Total time (min)	Diagram 1	D1 Time (min)	Diagram 2	D2 Time (min)	Diagram 3	D3 Time (min)
Subject 1	Perspective based inspection	20	42	6	17	7	13	7	12
Subject 2	Perspective based inspection	23	46	7	14	8	20	8	12
Subject 3	Perspective based inspection	16	47	2	11	6	22	8	14
Subject 4	Checklist based inspection	20	42	6	12	5	15	9	15
Subject 5	Checklist based inspection	21	51	7	18	6	22	8	11
Subject 6	Checklist based inspection	15	34	7	15	4	11	4	8
Subject 7	Checklist based inspection	22	42	6	15	7	16	9	11
Subject 8	Technical review	13	39	2	2	5	24	6	13
Subject 9	Technical review	15	45	5	13	5	19	5	13
Subject 10	Technical review	15	40	2	6	6	21	7	13
Subject 11	Technical review	20	43	7	13	7	21	6	9
Total		26	70	7	20	9	25	10	25

Total is the numbers set for the experiment during the experiment design phase.