Technische Universität Wien

Diplomarbeit

Empirische Untersuchung von Software Inspektionen als Ansatz zur Qualitätsverbesserung durch Fehlererkennung in der Softwareentwicklung

Ausgeführt am Institut für

Softwaretechnik und Interaktive Systeme

Der Technischen Universität Wien

Unter der Anleitung von

Ao.Univ.Prof. Dipl.Ing. Mag.rer.soc.oec. Dr.techn. Stefan Biffl

durch

Dietmar Winkler

Davidgasse 7/22
A 1100 Wien

Wien, Mai 2003

Technische Universität Wien

Thesis

Empirical Evaluation of Inspections as a Quality Management Approach for Defect Detection in the Context of Software Engineering

Carried out at the department of

Software Engineering and Interactive Systems

Vienna University of Technology

Under the guidance of

Ao.Univ.Prof. Dipl.Ing. Mag.rer.soc.oec. Dr.techn. Stefan Biffl

by

Dietmar Winkler

Davidgasse 7/22

A 1100 Vienna

Vienna, May 2003

I. Table of Contents

1 INTRODUCTION 1
1.1 Target Audience 2
1.2 Historical approaches of the basic key issues 2
1.2.1 Software Engineering in an historical overview 2
1.2.2 A brief history of Quality Management 3
1.2.3 Software Engineering Experiments 7
1.3 Topics discussed in this thesis 7
2 APPROACHES IN SOFTWARE ENGINEERING 9
2.1 Definitions 9
2.1.1 What is Software? 9
2.1.2 What Is Quality Software? 10
2.1.3 What Is Software Engineering? 11
2.1.4 What Is a Software Process? 12
2.2 Software Life Cycle Phases 12
2.2.1 Requirements Phase 12
2.2.2 Specification Phase 13
2.2.3 Planning Phase 13
2.2.4 Design Phase 14
2.2.5 Implementation Phase 14
2.2.6 Integration Phase 14
2.2.7 Maintenance Phase 15
2.2.8 Retirement 15
2.3 Software process models 16
2.3.1 Waterfall model 16
2.3.2 V-Model 17
2.3.3 Spiral model 18
2.4 Chapter Summary 19
3 APPROACHES IN QUALITY MANAGEMENT 21
3.1 Quality Management Systems 22
3.1.1 DIN EN ISO 900x series 24
3.1.2 Capability Maturity Model (CMM) 28
3.1.3 Further approaches of quality management systems 30
3.2 Quality Management Methods to support Software Development 30
3.2.1 Assessments / Audits 31
3.2.2 Reviews 34
3.2.3 Inspection 36
3.3 Chapter Summary 37
4 DEFECT DETECTION WITH INSPECTION 38
4.1 Inspection Process approaches 38
4.1.1 Fagan's Inspection Process 38
4.1.2 The Process Sub dimension according to Laitenberger [43] 39
4.1.3 Inspection control 40
4.2 Products 42
4.3 Roles 42
4.4 Reading Technique 43

4.4.1 Ad-hoc reading 43
4.4.2 Checklist based reading (CBR) 43
4.4.3 Scenario-based reading (SBR) 44
4.5 Chapter Summary 45
5 SOFTWARE ENGINEERING EXPERIMENTS 46
5.1 An overview of empirical strategies 47
5.2 Quality Improvement Paradigm 48
5.3 Experiment process approaches 49
5.4 Evaluating Inspections using Experiments 50
5.5 Chapter Summary 51
6 INVESTIGATION OF INSPECTION USING A CONTROLLED EXPERIMENT 52
6.1 Research Questions 52
6.1.1 Investigation of Inspector Qualification 52
6.1.2 Defect Detection Rates 54
6.1.3 Temporal behaviour of defect-finding according to reading-technique 55
6.1.4 Usage and Acceptance of inspections 56
6.2 Experiment description 57
6.2.1 Software artefacts 57
6.2.2 Defect Detection Techniques 59
6.2.3 Feedback Questionnaire 60
6.2.4 Inspection participants 62
6.3 Experiment operation 65
6.4 Data analysis 67
6.4.1 Database structures 67
6.4.2 Volume of data 68
6.5 Data Evaluation Process 69
6.5.1 Data analysis process used for this thesis 69
6.5.2 Model of Basic data 71
6.5.3 Task description 72
6.6 Chapter Summary 73
7 RESULTS 74
7.1 Investigation of Inspector Qualification 74
7.1.1 Supervisor Qualification Assessment (*SD-Skills*) 74
7.1.2 Qualification Rating according to self-estimation (EXP, Q1.1) 75
7.1.3 Qualification Rating according to the PRE-Test (SI-DD, Q1.2) 80
7.1.4 Comparison of qualification assessment models (Q1.3) 83
7.2 Defect Detection Rate 84
7.2.1 Defect Detection Rates according to Inspector Qualification (Q2.1) 84
7.2.2 Defect Detection Rates according to Reading Technique Roles (Q2.2) 88
7.2.3 Defect Detection Rates according to Document Locations (Q2.3) 90
7.3 Temporal Behaviour of Defect-Finding according to reading technique 95
7.3.1 Inspection time according to reading technique and inspector qualification (Q3.1) 95
7.3.2 Average Time to the first defect detection (Q3.2) 96
7.3.3 Average minutes between defect detection (Q3.3) 98
7.4 Acceptance of Inspections 100
7.4.1 Applicability of Reading Techniques (Q4.1) 100
7.5 Chapter Summary 102

8 DISCUSSION 104
8.1 Investigation of Inspector Qualification 104
8.1.1 Supervisor Qualification Assessment (SD-Skills) 104
8.1.2 Self-Estimation Qualification Assessment (EXP, Q1.1) 105
8.1.3 PRE-Test Qualification Assessment (SI-DD, Q1.2) 106
8.1.4 Comparison of Qualification Assessment (Q1.3) 107
8.2 Defect Detection Rate (DDR) 107
8.2.1 Defect Detection and Inspector Qualification (Q2.1) 107
8.2.2 Defect Detection and Reading Technique Roles (Q2.2) 108
8.2.3 Defect Detection and Document Locations (Q2.3) 109
8.3 Temporal Behaviour of Defect Detection 109
8.3.1 Inspection Duration (Q3.1) 110
8.3.2 Average Inspection Time to the first Defect Detection (Q3.2) 110
8.3.3 Average Inspection Time between Defect Detections (Q3.3) 111
8.4 Acceptance of Inspections 111
8.4.1 Usability of Reading Techniques (Q4.1) 111
9 SUMMARY AND OUTLOOK 113
10 REFERENCES 117
10.1 Bibliography 117
10.2 Web-References 120
11 INDICES 121
11.1 Table of Figures 121
11.2 Table of Tables 122
12 APPENDICES 123
Appendix A – ISO 900x series 123
Appendix B – Questionaire according to the SWT-Experiment 2000 123
Appendix C – Task-Description for Reading-Techniques 123

II. Preface

Abstract

Software products influence everybody's life in many areas, solving more or less complex problems, providing support to do the work or simple supply some kind of entertainment. An increasing number of products of daily use don't work without software. While this piece of software meets its requirements, the user is satisfied. Otherwise he denotes the product, as a whole, as a low quality product. Once, developing software products, project-managers and quality managers are responsible for quality software.

This thesis addresses project-managers as well quality managers and engineers in the area of software engineering, who want to improve their project proceeding and quality of their products. A development team, who wants to produce quality software, will have to use a defined process, concerning all phases in the software life cycle, beginning at the first idea and ending at the retirement phase.

To support the production of quality software, several methods exist, which can be summarised as Software Quality Assurance (SQA) activities. Quality Management (QM) is directed towards improvement of product value and service for customers and users. There exist some quality management systems, like the ISO 9000 series, CMM, etc. to provide a well-defined framework.

Nevertheless, the main goal is fulfilment of requirements and, therefore, the reduction of defects in general. Because the repair of defects will spend a lot of resources, defects must be removed as early as possible. One possible approach for this purpose is the software inspection, which will be used for defect detection in early stages of software development, i.e. in the specification phase. The inspection is based on a highly formalised process for the improvement of software documents. Several techniques, like reading techniques exist, to support the defect detection during the inspection process.

This work focus on the practical evaluation of software inspection, using data, collected at an experiment at the *Institute for Software Engineering and Interactive Systems* at *Vienna University of Technology*. The results can support project- and quality managers to implement software inspection within a project team. Following this approach I will investigate

- *Inspector qualification* models to achieve best defect detection results with respect to a practical environment.
- *Defect Detection Rates* in context of qualification, reading technique and document location.
- *Temporal behaviour* of defect finding according to overall inspection duration and preparation time.
- *Acceptance of inspections* as indicator for the usability and simplicity of inspection processes.

Acknowledgment

During my studies in the area of computer science at Vienna University of Technology I got in contact with many people, who support my work. Furthermore the writing of this thesis, which is based on a practical experiment in the area of software engineering, involved a lot of people as well.

I want to thank *Stefan Biffl* for academic advice on this thesis, including discussion, conception and execution, *Michael Halling* for co-ordination of the experiment and his support in the first half of my thesis.

Thanks to the staff of the Institute of *Software Engineering and Interactive Systems* for providing the framework of the experiment. Further I want to thank all members of the *experiment preparation, execution* and *data analysis team* and of course all *students* of the experiment B and their tutors, which provided data for the evaluation of research questions.

I also want to thank everybody, who contributed my life during my studies, first of all my parents for giving me the possibility to study, my girlfriend and all the other friends who participated my life during the last years.

1 Introduction

In everyone's life computers are involved to solve more or less complex problems, support them to do their work and spare time activities or simply provide some kind of entertainment. Computers don't work without software. Therefore software is a very important component of a computer system. We have to distinguish several types of software [68]:

- Administrative systems: The main task of this software is the manipulation of large amounts of data providing a well-designed user-interface and useful tools for input, manipulation, evaluation, analysis, visualisation and finally output of accumulated data.
- Distributed Systems are defined as a collection of autonomous computers linked by a network, with software designed to produce an integrated computing facility [21].
- Real-Time-Systems are computer systems in which the correctness of the system behaviour depends not only on the logical results of the computations, but also on the physical instant at which these results are produced [40].
- System Software is tightly coupled to the characteristics of machines with a corresponding interface for communication between the software and the hardware.
- Expert Systems are used for the representation of stored knowledge. Knowledge has to be presented according to the query and must provide solutions for the expansion of knowledge at all (continuous improvement and expansion of the knowledge base).
- Web Applications are special kinds of distributed systems with special requirements concerning the user interface and security options.

Because of a wide range of different computer systems and software applications, software developers have to use different methods and models to achieve the individual requirements and quality levels.

You can compare software engineering projects, processes and products to each other if you can find similarities between them. These similarities concern project parameters (e.g. project size, project duration, budget, man-power, composition of the software engineering teams according to their skills), process parameters (e.g. approach of a software model, usability of a software process models, etc.), product parameters (e.g. type of software, complexity of the software) and application knowledge (e.g. well known-application area or expeditions to anywhere without any predefined goals), etc. In practice it's always a mixture of all parameters and it is the task of the project management to find a proper solution for the proceeding of a successful project.

In this thesis I will focus on administrative systems but I will provide some information for general purpose as well. These administrative systems are typical median scale software product for 4-6 developers and 6 man-months duration.

The product itself can be considered as an “ordinary” software product in a well-known application area, e.g. distributed ticket-selling system with one central database server.

1.1 Target Audience

This text appeals to quality managers and project managers (and other key people, involved in project decision processes), who want to improve the product quality, reduce defects, and – as a consequence – cost, already in early stages of software development. In the experimental part, I will present some conclusion to support the implementation of inspections within project proceeding.

I also address developer, designers, tester and everyone, who want to understand software engineering processes with respect to quality assurance (QA), quality management (QM) and finally, quality improvement (CI).

Software engineering, the ideas of quality management and empirical software engineering as an evaluation method covers different aspects in scientific life also because of their individual history. The next sections give a rough overview about their individual history, key aspects and tries to find actual interaction between them.

1.2 Historical approaches of the basic key issues

This section gives a rough overview and a selection of key aspects concerning the historical development of software engineering, quality management and experiments in the area of (empirical) software engineering.

1.2.1 Software Engineering in an historical overview

In the 1950s, hardware vendors provided methodologies to their customers as add-on to their components; software has been second-rate. Only talented amateurs were able to implement solutions according to specified machines. Till the mid of the 1960s, different ideas has been formed, “how to create quality computer applications”, which resulted in inhomogeneous products with poor quality and non-standard applications. Nevertheless individual – non-standardized – approaches and proceeding have been developed as well.

At a NATO conference in 1967 the term “software engineering” has been introduced first by conference participants. Reports created at a conference sponsored by the NATO in Garmisch in October 1968 and in Rome in October 1969 captured some ideas giving the implementation of computer programs the “touch” of an engineering-disciplines including analysis and development of applications.

In the mid sixties and seventies the term “software crises” has been formed, because of the poor quality of software products without any hope for improvement. The first papers describing approved software development processes have been published as a reaction on

this crisis. Since the 1980s several software development processes has been formalised, published and distributed to a bigger audience [68].

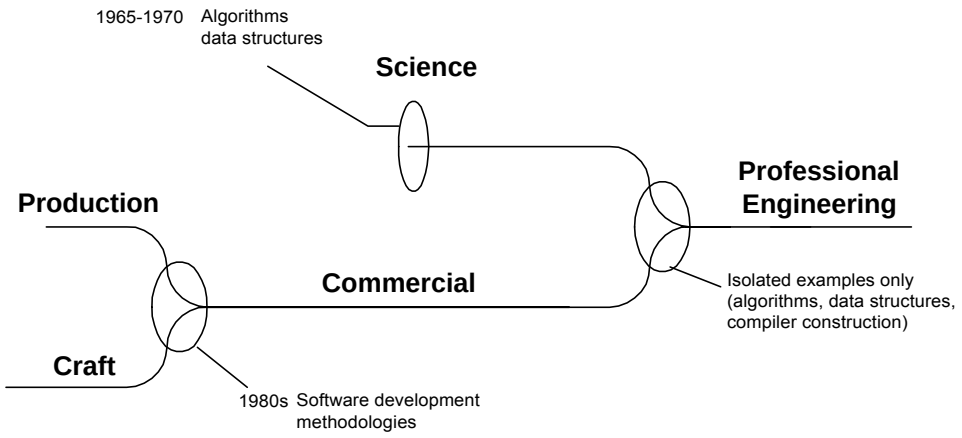


Figure 1.2.1-1: Development of Software Engineering [Shaw et al. 1996]

Figure 1.2.1-1 shows the relationship and the historical development of software engineering in context to the development of an engineering discipline [68].

Up to now, there exist several models and methods, which have been proved in practical environments and only tested in limited areas, like algorithms and data structures, compiler construction, etc. The future's task is the scientific investigation of software engineering, which is an ongoing challenge.

To understand the importance of quality engineering or quality management, we have to look at the historical usage of quality in general.

1.2.2 A brief history of Quality Management

In comparison to the history of software engineering, the area of quality management (QM) and quality assurance (QA) is quite an old discipline.

The first approaches of quality assurance activities appeared at the beginning of the 20th century during the so-called industrial age in the area of automotive production. Regular workflows have been split into small well-organized units with highly specialized staff within a team. Every team-member has been responsible for its own small task without getting an overview about the whole process but their small sub-processes. The consequence of this proceeding was a strict hierarchical structure within a company and a limited influence to the corresponding working areas according to each management level. This proceeding has been introduced to serve the so-called "mass-market" with a high number of unique products.

Because “unique” products are not unique per default, but defects and inhomogeneous products can appear in one charge, additional activities must be implemented to guarantee unique products – the quality assurance has been introduced.

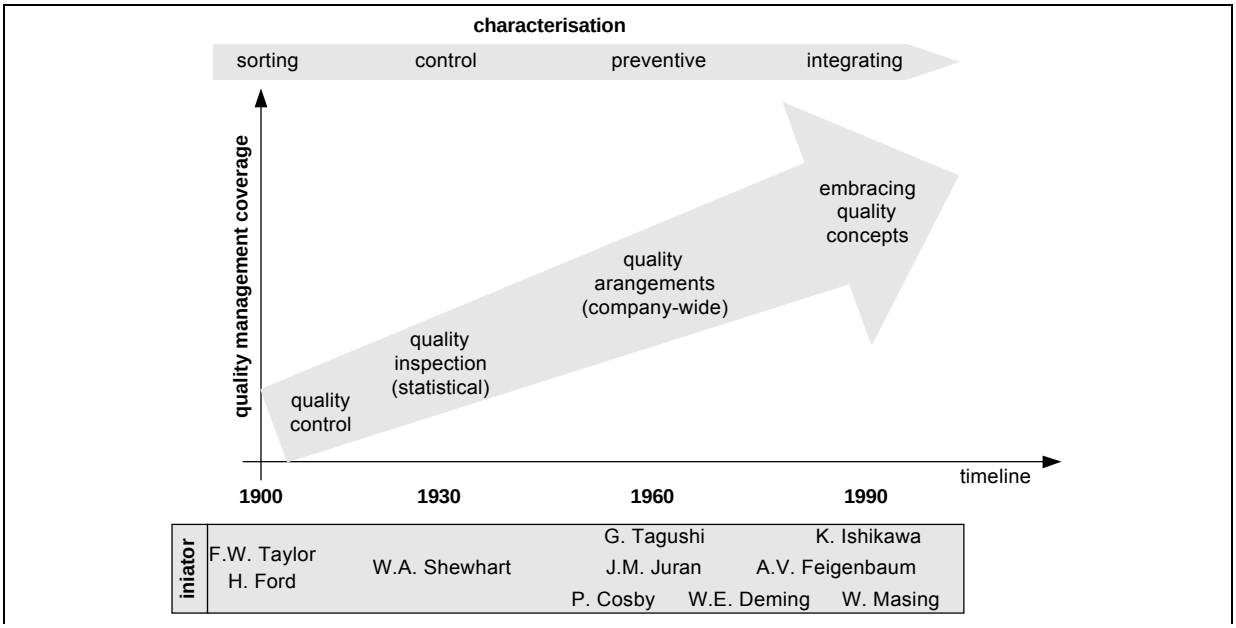


Figure 1.2.2-1: Development of Quality Management

Because there were no process improvements, several controlling mechanisms took place. At the beginning of the 20th century, in 1991, F.W. Taylor published “The Principles of Scientific Management”. His investigation results have been implemented in automotive productions fields by H. Ford, one of the precursors in the area of automotive production and process control (men have been controlled by the process).

In the 1930’s Walter A. Shewart successfully brought together the disciplines of statistics, engineering, and economics at academic level and became known as the father of modern quality control. In 1931 he published his work in “*Economic Control of Quality of Manufactured Product*”, which can be regarded as a complete exposition of the basic principles of quality control [69]. The inspection of products according to their specification has been realized in testing random samples out of the whole lot size using statistical evaluation methods.

A new approach in the area of quality engineering appeared in the 1960’s, because of a more complex composition of products and processes. Before the 1960’s defects corrected at the same process stage as they has been detected. The idea of several pioneers, like Tagushi, Juran, etc. was, that defects must be removed as early as possible, i.e. where they have been generated. They have been triggered by economical approaches, because the cost of defect corrections is cheaper in earlier process-stages.

J.M. Juran's requirements to quality also focus on the management. He claimed for a transparent management system and a customer-oriented quality management system concerning a systematic and permanent quality improvement. Temporary problems have to be solved by the employees themselves, but the management has to solve permanent problems (about 80% of quality-relevant problems are permanent). His concept concerns statistical methods and methods for working-structuring (e.g. job-rotation, enlargement of individual working scopes, self-controlling and education)

In addition to the structural change in a company, P. Crosby defines quality in the following way:

- Accordance to requirements
- Defect prevention as a basic principle
- Zero-Defect-Strategy as a key practice
- Non-compliance or requirements as a benchmark for quality cost

In the 1970's and 1980's the quality assurance has been introduced as a new, independent unit within a company, which supports the others to improve their quality and establish their quality management system. W.A. Deming's basic assumption was the idea of quality as an integrated element within a company. In his opinion, only about 10% of problems, resulted in bad quality, can be corrected by the employees themselves; about 90% only can be corrected by changing the processes, which must be initialised by the management. This proceeding must be integrated in the so-called "company politics", which focus on the continuous improvement of products and services. Processes are primary targets to be improved in co-operation with the customer. This interaction must include everybody in the company, customers, users, products, etc.

The last milestone in the evolution of quality management, I will describe in this text, can be summarized as "value-based quality", which was introduced by A.V. Feigenbaum in the 1990s. His approach includes also the benefit of usages for all participants and of course the cost of development and the price. He points out 4 basic tasks:

- Development of new products in accordance with customer requirements including the analysis of possible defects.
- Permanent monitoring of delivered products and services
- Process studies to investigate cause of defects and improvement of products and processes and, as a consequence, increase of production.
- All activities must agree with quality requirements.

The mainstream of the 1990's leads to a total quality management, away from QSA departments and autonomous task forces to a enclosing quality approach (the well-known

TQM is one of the most important approaches of this trend). The main aspects of TQM are the integration of customers and employees and business-processes into quality management processes. Quality is seen not only as a short-time activity but a permanent activity, to improve quality awareness within the whole company.

The most important goal is the all-embracing control of all processes used in the company (independent of application areas) using TQM, etc. Concerning the area of software engineering, researchers try to adopt techniques out of the area of a well-known application area to software engineering processes.

With special respect to the area of software engineering, Tervonen et al. [62] points out some aspects, which must be considered when introducing a definition to software quality or even quality in general. As part of his results shows individual perspectives, how different people will see quality. Figure 1.2.2-2 shows a summary of this approach:

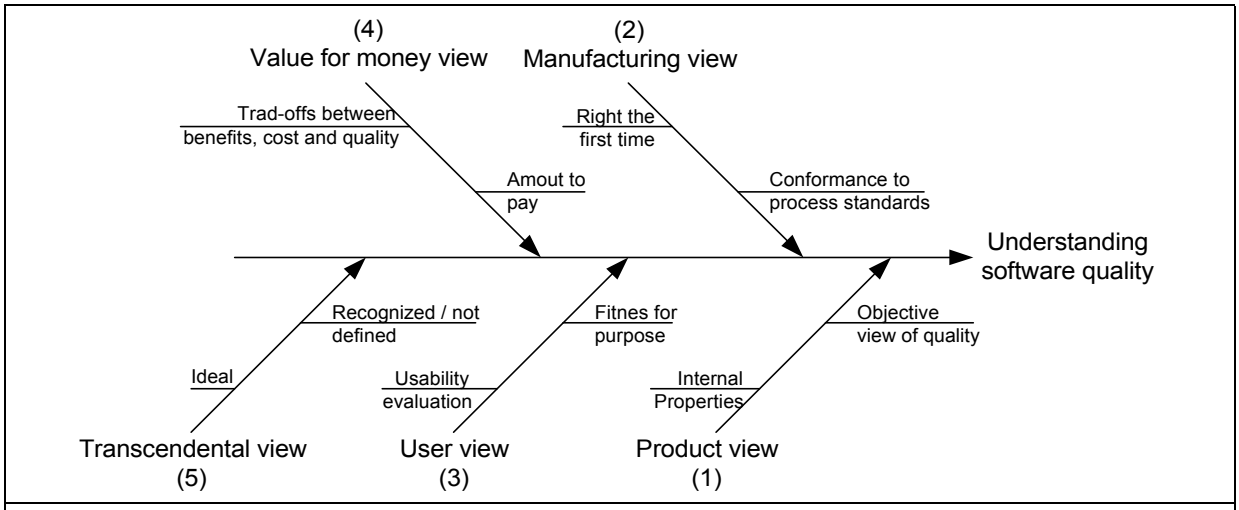


Figure 1.2.2-2: Five perspectives on software quality [62]

Figure 1.2.2-2 describes five different views on quality approaches based on papers from Kitchenham and Garvin [62].

1. Product view: This approach assumes that measuring and controlling internal product properties will result in improved external product behaviour. This offers an objective and context-independent view of quality.
2. Manufacturing view focuses on product quality during production and after delivery, with special respect to processes under the assumption, that conformity to processes will lead to good products.
3. The user view evaluates the product in a task context and can be a highly personalized one. It also presents products usability.
4. Value for money view: This approach evaluates the conflict between user's requirement for a product and manufacturer's goal of minimizing rework.

5. The Transcendental view sees quality as something that can be recognized but not defined. Thus software quality is something towards which we can strive as an ideal, but may never implement completely.

As a consequence several models and quality standards, like the ISO 900x series, CMM, SPICE, BOOTSTRAP, EFQM, etc. have been developed according to their application area and special requirements. Their usage and acceptance will be investigated in the future and, enhanced.

The combination of software engineering and quality management is still a big challenge. Therefore it is necessary to evaluate theoretical approaches, models, methods, and, finally tools according to their usability and usefulness.

One possibility is the implementation of software engineering experiments in the area of empirical software engineering.

1.2.3 Software Engineering Experiments

Software engineering is defined by the IEEE as “software engineering means application of the systematic, disciplined, quantifiable approach to development, operation and maintenance of software” formally [67].

In the context of experiments, there are three aspects important:

- Different life-cycle phases within a software process
- Need for a systematic and disciplined approach
- Quantification of software

Because software engineering is a cross-disciplinary subject, we have to understand the methods themselves, their limitations and their application area. Glass points out four research methods in the field of software engineering, *the scientific method*, where models are built according to a real world observations, *the engineering method*, analysing and adopting current solutions, *the empirical method*, evaluation of models purposed by the researcher, and, *the analytical method*, which is based on a formal theory.

1.3 Topics discussed in this thesis

The text provide a introduction to software engineering (in chapter 2) using a general approaches of the software life-cycle phases and in a more concrete way some widely used software process models like the waterfall-, V- or the spiral-model. Chapter 3 presents some introductory information on quality, quality management and some selected methods for improvement of quality with respect to the software life-cycle model and quality management standards and their key aspects.

One of a very important method for defect detection in early stages of software development, e.g. in the specification phase, is the software inspection. Chapter 4 provides a closer look on software inspection presenting the proceeding of inspection and some techniques out of the practice to improve defect detection rates, etc.

One problem in the practical life of a project or quality manager is the implementation of methods and tools, not used so far, without having detailed knowledge about it. One possibility is some kind of experimental approach (i.e. empirical software engineering) to evaluate the method before using it. I will cover some approaches in empirical software engineering in chapter 5. Because we executed a wide ranged experiment concerning software inspections an different approaches of execution at the *Institute for Software Engineering and Interactive Systems* at the *Vienna University of Technology*, I will describe the proceeding of this experiment in detail (chapter 6) and present results with a closer look at inspector qualification, defect detection rates, efficiency, etc. in chapter 7 and 8.

The whole thesis will give an overview about different topics concerning software engineering in general, quality management systems and methods, experiments in the area of software engineering and software inspections concerning specification documents. Experiences gathered while and after execution of the experiment will support the decision process for the implementation of a software inspection in a software engineering project.

2 Approaches in Software Engineering

There are many challenges in software development to meet a wide range of requirements. On the one hand side, there is the customer (client), who wants to get the very best solution for his own benefits (i.e. cost, schedule, quality, etc). On the other hand there is the developer (or developing team) who has to meet these requirements to achieve highly satisfaction of the client.

The project management has to consider application area, project size, duration, team compilation, establish SE-methods as well as QM-methods (maybe within a quality management system), and find the best solution to prevent defects in early stages of the project. A proper choice will lead to a proper product in most cases (although there is no guarantee but it will be very probably) – errors are expensive in resources because the project-status might return to previous development stages.

2.1 Definitions

2.1.1 What is Software?

While talking about software many people think about programs running on computers doing something. This point of view is very restrictive because the term “software” includes much more than one “simple” program on computers, workstations, servers etc. Software contains programs as well as associated documentation and configuration data that are needed to make these programs operate correctly [60]. Thaller concerns software as a package consisting of computer programs, documents and appropriate data [63].

Sommerville defines two types of software [60]:

- Generic products
- Customized products

Software can be developed for special needs but without any direct contact to the customer while developing a system, e.g. a general-purpose office application like Staroffice™[1] or Microsoft® Office[2]. A customer will buy the product, use it and will be happy with it or even not, e.g. if restrictions in the functionality, defects etc occur. Sommerville call this kind of software “*generic products*” [60].

While “*customized products*” are developed with direct influence or the clients and the developing company’s challenge is to meet the needs of the particular customer very well, e.g. ticket selling system, power-plant control system etc.

---

[1] Sun, Sun Microsystems, the Sun Logo and Staroffice are trademarks or registered trademarks of Sun Microsystems Inc. in the United States and other countries.

[2] Microsoft Office is either registered trademarks or trademarks of Microsoft corporation in the United States and/or other countries.

This second approach is the more difficult because the customer needs special features described in some kind of document and the task of the developing company is to fulfill these requests. The following text covers the latter approach of software. I describe the software process in more detail in chapter 2.2.

2.1.2 What Is Quality Software?

Actually, there exist no general definition concerning quality. Depending on different views of each author, quality definitions may differ. In Deming's opinion quality is directed towards customer needs. Present and future needs (of clients) must be fulfilled. Cosby limits quality to the meeting of requirements only. The ISO 8402 (version 1986) defines quality as a collection of functions and attributes of a product or service which must meet special demands or which are associated with requirements.[63].

Wallmüller [66] refers to an empirical investigation concerning quality to D.A. Garvin. I will summarize the five main approaches pointed out by Garvin:

- Transcendental approach: There is no possibility to measure quality. High experienced people assess the software product, concerning the "feeling" of the solution at computer screen.
- Product related approach: In opposition to the transcendental approach, quality is measurable exactly. Within a type or category of software a rule of precedence according to software quality using classification and attribute lists can be established.
- User related approach: Clients and users define quality in general. If the customer is satisfied with the product concerning functions and appearance the client associate the product with high (or even low) quality.
- Process oriented approach: This approach focus on the accordance of product and specification documents, e.g. guidelines, standards, etc. Following this approach, the other approaches will be fulfilled as well.
- Cost / Effort related approach: The relationship between cost and accordance to specifications is sensible for managers and clients.

As a summary of quality definitions and with respect to software engineering especially to software products, the key aspects can be outlined:

- Functions and attributes must meet requirements exactly
- Contentment of the customers (e.g. caused by minimization of defects remaining in the product)
- Meet all guidelines, e.g. development standards, state of the art, legal conditions, etc.

A further discussion of quality and how quality can be established in software engineering processes as well as in quality management systems is outlined in the following chapters.

2.1.3 What Is Software Engineering?

Software Engineering itself is an "engineering discipline, which is concerned with all aspects of software production" [60]. As an engineering discipline the engineers apply theories as well as methods and develop solutions to predefined problems, if no theoretical approaches are available so far. Even more they must meet constraints like organizational (e.g. time constraints) and financial (e.g. projects budget) milestones as well as quality demands defined by the project management. From this point of view Software Engineering can be see as an integrated engineering activity.

The decision using software engineering methods, which are described in chapter 2.3 depends on the software process used, financial budget, project size and complexness of the resulting product. One of the most difficult problems of a project manager is the estimation of size and cost, because there exist no unambiguous technique to reach proper results.

Sommerville suggests the total cost of developing a median scale system in Figure 2.1.3-1. 100 cost units represent the whole cost for developing a system. Each basic step of the software process (described in 2.3) costs amounted to the corresponding share.

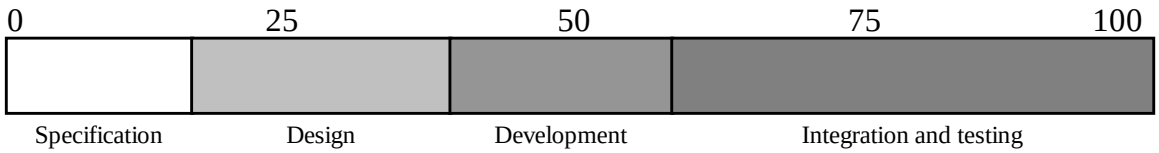


Figure 2.1.3-1: Cost distribution in a software process [60]

Specification and design as well as integration and testing amount to about 80% of the whole developing cost, while implementation (development) only comes to 20%.

Project size is defined in person-hours, person-months or person-years and must be estimated by the project-management according to project environment, resources etc. Due to direct feedback to cost-factors, customers' satisfaction and product quality it is very important to achieve realistic data. Different methods for measuring of cost, complexness etc. exist, e.g. function point method, comparison to similar projects, CoCoMo etc. A further discussion will be out of place for this thesis.

According to the order of magnitude of software products several a project manager must take some measures to achieve product quality (software quality is discussed in chapter 2.1.2. First of all he must introduce a software process and he must use a software engineering model.

2.1.4 What Is a Software Process?

The software process is the way we produce software. It incorporates the software life-cycle model, the tool we use and the individuals building the software [56]. A software process includes a set of activities and associated results, which produce a software product.

The fundamental activities in common software development processes are the (i) software specification for technical and formal demands, (ii) software development and implementation of the product, (iii) software validation to compare software product to customer needs and finally (iv) software evolution to meet changing customers demands [60].

Software processes cannot be considered as kind of a "standard" but only as guidelines defining necessary parts in the software production. A software process has to be adapted to special needs of a company or project.

2.2 Software Life Cycle Phases

According to Sommerville's definition of a software process there exist four fundamental activities in a software engineering project. Schach covers these basic steps too but in a more detail way. He distinguishes 8 phases, which I will cover in this chapter [56].

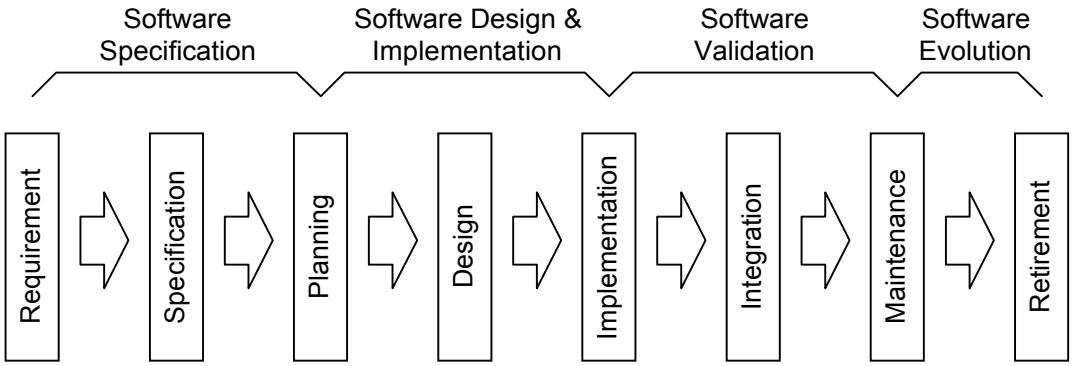


Figure 2.2-1: Software process comparison Schach vs. Sommerville

As tried in Figure 2.2-1, a comparison between both software processes is possible and shows a more detailed view at [56] still containing the [60] basic process steps.

2.2.1 Requirements Phase

The first contact between client (customer) and developer seem to be very diverse. The client's view of the product will be according his own benefit and he doesn't know what he really wants and how this can be realized. The developer owns his opinion and doesn't exactly know what the clients want, but he owns the know-how of some possible solutions according to his ideas. In the requirements phase (or concept exploration) there should be a

compromise, so that its clear, what the client can expect and the developer has to realize. The final document, called requirements analysis describes these facts. Typical constraints are functionality, cost, deadline, reliability etc. [56]. While working at the requirements analysis the team, consisting of client members as well as members of the developing team, creates products like system description, glossary, interfaces and actors, use-case-scenarios and description, prototypes etc [68].

After the final version a quality assurance method should be established to guarantee agreement between client and developer. Reviews, Inspections, described in chapter 3, are possible approaches to do this.

This phase cannot be found at [60] as an own step in the process, but some kind of requirements analysis must be done as a preliminary work for creating a specification.

2.2.2 Specification Phase

After establishing a requirement document a specification document is created, which constitutes a contract between client and developer.

The specification contains detailed information about functionality and non-formal demands to the software product as well as environment, testing and documentation needs. There may no terms in the text indicating different interpretations or unclear information to avoid unwanted changes in later development stages. While writing a specification developers must watch out for ambiguous or even incomplete definitions.

Also the software quality assurance (SQA) group must check the specification because of its very high importance for later development stages.

Sommerville considers the specification phase as the first necessary part in the software-process [60].

2.2.3 Planning Phase

At this stage, which is one necessary process step at [56] this phase must be established for the whole project. [Somm01] doesn't mention a planning phase at all but – in my opinion – it can be related either to the specification or to the design phase.

At this point of the project everybody (client and developer) should know exactly, what has to be done according to the specification document. Now the project management has to plan the further developing process till the end of the project. Some activities, e.g. testing code fragments can only be done, if they has been written before – some activities must be done in sequence so far others might be done in parallel, e.g. coding and writing test. The earliest project milestone a plan can be drawn up is when the specifications have been finalized. [56].

The project manager's task is to find a solution for the project to optimise duration, cost and resources. Also he has to take care, that a SQA can be established to guarantee best project development and make allowances for possible troubles within the project. Detailed

information about project team, tasks of project management etc. can be found in literature, e.g. at [60] [56].

2.2.4 Design Phase

The specifications spell out *what* the product must perform. The aim of the design phase is to determine *how* the product is to do it [56]. The main task during the design phase is the definition of the internal structure, data-flows, algorithms, interfaces to the project (or even the world) etc. still according to the need reported in the specification. Every design decision must be reported to reconstruct ideas of the designers. This can be necessary if one decision leads to a "dead end" and designers has to backtrack to a previous stage of design or for maintenance reasons.

Scalability or openness for future needs has been considered, but is nearly impossible to take care of all possible demands, so the design team has to find a compromise.

Specification phase as well design phase are coupled tightly to each other, so it's difficult to determine the project stage.

The design team creates an *architectural design*, which describes the product in terms of its modules and a *detailed design* for a description of each module [56].

Based on the design, which also must be released by the SQA team, programmers are able to implement the system.

2.2.5 Implementation Phase

The design documents are the basis for the implantation phase. The developer codes the individual modules and, of course, tests them according to the test plan. Further documentation is necessary for later stages in the project, e.g. maintenance phase to reconstruct special realizations of different modules for improvement or defect corrections. Chapter 3 will describe several QM methods to improve or even test software quality.

2.2.6 Integration Phase

Once all modules are generated, tested they must be built together. Schach talks about different strategies to realize integration [56]. Modules can be put together all at once or step by step but also top-down or bottom up (according to the module interconnection diagram). Considering up-down integration, modules, which will be integrated later, will not take effect, because they are not integrated so far. In this approach design errors are visible very soon. Equivalent using bottom-up integration, all modules seem to work properly but design-errors will be detected very late in the integration phase. Both strategies have its advantages and disadvantages but relate to the project plan (according to the planning phase) very tight. Implementation and integration must interoperate to achieve best solutions and early error detection und, as a consequence – correction of e.g. design or even worse specification.

Once integration is finished, the product itself is ready for clients use. At this point starts the most challenging phase, the maintenance phase.

2.2.7 Maintenance Phase

Maintenance is an integral part of the software process [56] and not only a post-development activity for troubleshooting or repair of a system. Also in the design-phase engineers have to consider maintenance activities with respect to enlargement and modification according to customers needs (demands can change during practical usage).

One critical point during software development is the lack of time. Clients as well as developer give attention to the product more concrete to the program because everybody needs to see the program working. Documentation will be disregarded if there occur any pressure of time. Documentation includes user handbook, developing documentation and source code description as well. Several times later in the stage of maintained nobody really knows why a special solution has been used etc. So it is very difficult to reconstruct special topics and make changes more difficult.

Following this approach already in the stage of design and project planning engineers must think about maintenance problems with special respect to documentation.

It's one of the main tasks of the SQA and project management to guarantee these proceedings.

2.2.8 Retirement

After years of usage it might be necessary to think about different measures of further activities with (or without) the software. There are four main reasons for a decision:

- Too many changes could reason in a complete redesign or recoding because of possible design changes.
- Due to many changes which interdependent activities can effect program status and result in trashing the program. A little modification in one simple module can result in complete wrong results at some other place.
- Through quick & dirty changes without updating documentation will lead to inconsistencies at the whole product so it will be safer to recode the program.
- Changing hardware will probable need a redesign or recode of the program to achieve a proper performance.

Because of this reasons the software will be out of date and its more sensible to trash it an initiate another software-product.

The mentioned stages incorporate the software life cycle. Different problems e.g. backtracking to previous project phases are rarely possible – modifications or different process models must be used which I will describe in the next section.

2.3 Software process models

A Software process model is a simplified description of a software process, which is presented from a particular perspective [60], they may include activities that are part of the software process, software products and the roles of people involved in software engineering. These series of steps during product development is called *life-cycle model*. There exists no software process for general purposes but for individual project approaches according to project types, size and project complexity as well as application area and domain. It's the task of the project manager to apply a process model (or maybe a modified model) to the individual project.

The simplest one is the *build-and-fix model*. Is used for small project or even to get a quick solution. Without requirements analysis or specification a developer writes down lines of code until the client will be satisfied. There is no approach of a software process model with its typical phases.

For more critical projects with higher complexity or involving a larger team involving more than one developer and one client, a process model must be chosen. A measurement of product quality is impossible but using an improved process will improve software as well [63].

Because there are many different models available at the moment, I will describe three important models in this text.

2.3.1 Waterfall model

The first published model of the software development process was derived from other engineering processes (Royce, 1970) [60]. The model consists of all steps of the software life-cycle model including its functionality, like the in Figure 2.3.1-1 shows.

The main point of this model is the strict separation of each process steps or phases. A verification step is included at the end of every phase. It is not possible to access the next phase without finalization (and check) of the actual phase. Although it is possible to backtrack to the previous phase (if the product doesn't meet different requirements) the straightforward strategy avoids backtracking of more than one step. According to this proceeding, the software process is not a simple linear model but involves a sequence of iterations of the development activities. Nevertheless the structure of distinct phases is the most important lack of the model because occurring problems stops further development after the actual problem has been solved. [60]. Following the strict process line modification happen in the maintenance stage and may involve even the design stage.

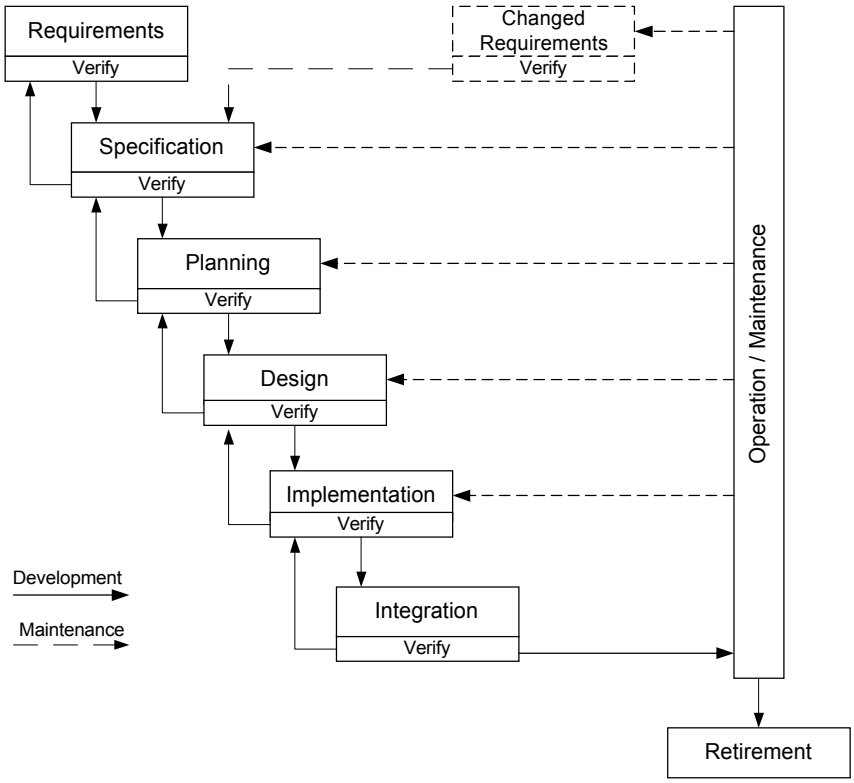


Figure 2.3.1-1: Waterfall model applied from [56]

2.3.2 V-Model

The V-model can be used in companies building small to median software. One of its benefits is the direct view of software verification and validation [63].

Each development stage (according to the software life-cycle) consists of a verification stage at the end of each phase. Looking at the V-model in Figure 2.3.2-1 the validation checks the product of each step to the corresponding requirements.

The proceeding seems to be like the waterfall-model but the context between products and tests is more visible. But also defects in early project stages will be found out very late and will relate in expense maintenance activities, when they must be removed.

The model specially points out the context of the specification part (descending proceeding of analysis) and the related tested products (ascending proceeding of synthesis). Different views with respect to varying levels of detail describe possible testing approaches.

Because of the strict proceeding of individual stages of development, its very hard to fix defects in earlier stages (no backtracking is provided by the model). This model should be used in a well-known or clear defined application area.

To minimize defects in the development process, different methods should be established, e.g. reviews, audits, test-plans and tests etc. at each stage of development.

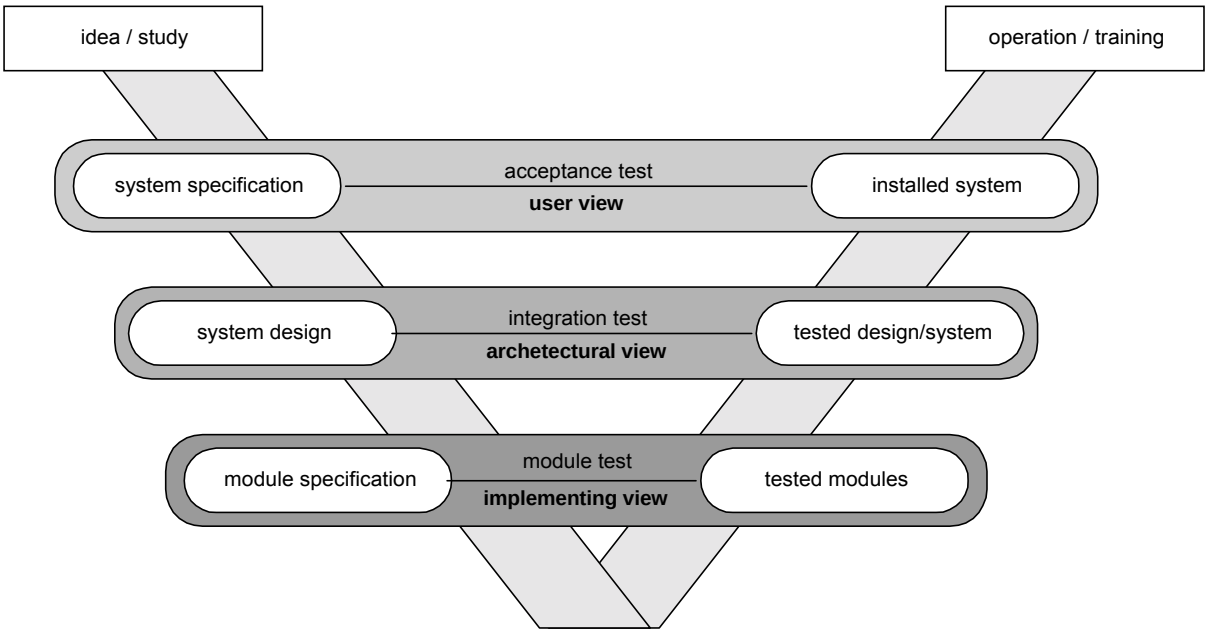


Figure 2.3.2-1: V-model according to [68]

Each software development organization should decide on a life-cycle model that is appropriate for that organization, its management, its employees, and its software process; and vary the model depending on the features of the specific product currently under construction. Such a model will incorporate appropriate features from the various life-cycle models, minimizing their weaknesses and utilizing their strengths [56].

2.3.3 Spiral model

This model concerns project risks and tries to avoid them. The idea of minimizing risk via use of prototypes and other means is the concept underlying the spiral model (Böhm, 1998) [56]. A spiral starting the innermost loop represents each software process phase.

Only particular products will be built in a specific stage, depending on pervious (existing) products. Figure 2.3.3-1 shows the full spiral model by Böhm.

Each loop is split in four sectors [60]:

a) Objective setting: definition (identification and planning) of specific objectives of the project. Identification of risk and planning of alternative strategies.

b) Risk assessment and reduction: All identified risks are analysed and steps introduced to reduce them (e.g. using a prototype)

c) Development and validation: According to the type of risk (e.g. integration problems etc) the best process-model is used to support developing.

d) Planning: The final step of each spiral is the review of the project step. Either the next step (spiral) will be initiated or the project stage must be improved, e.g. using another different strategies.

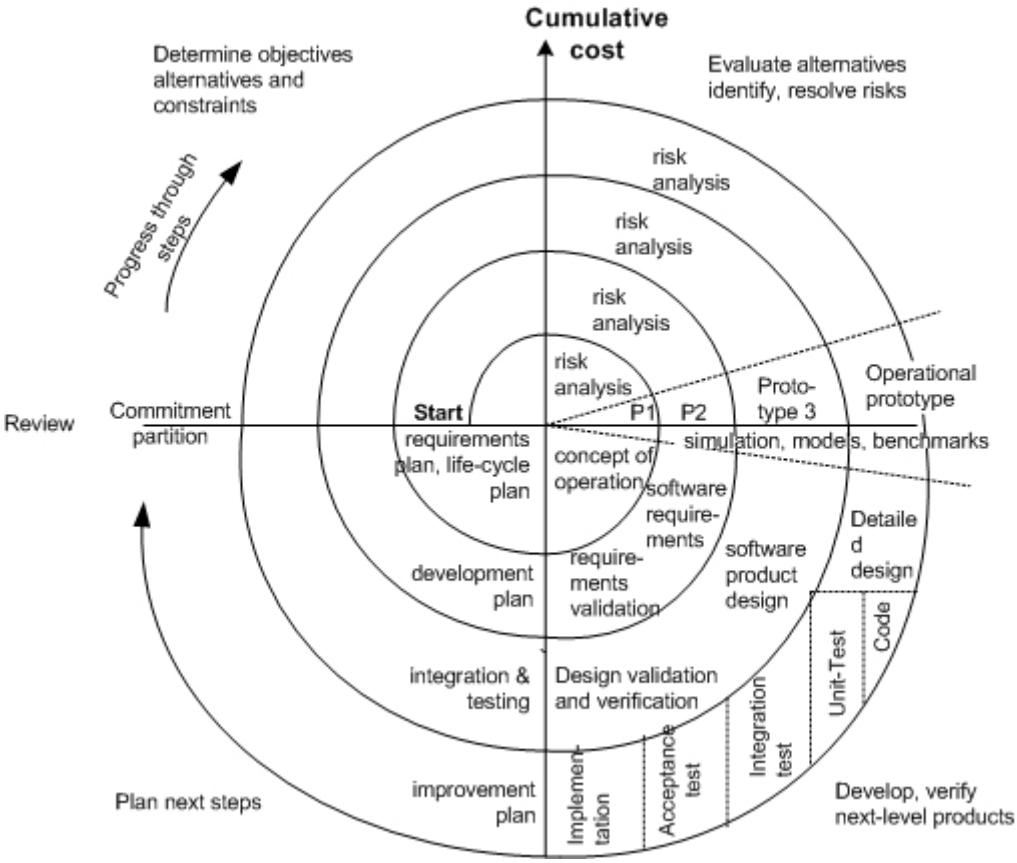


Figure 2.3.3-1: Spiral model by Böhm [56]

The model is used for large-scale projects with a high level of complexity and a long duration until initial delivery. The life-cycle phases don't approach in a sequential way but occur within the spiral, using other models according to the risk, analysed.

2.4 Chapter Summary

To achieve high-qualified software products, project managers must use defined models and methods during the whole development cycle.

The first section points out some important goals, i.e. definitions for quality software, (software) engineering processes, etc.

A closer view on software processes with respect to the life-cycle model present main steps (phases) in software development and describe them in more detail. Because of the general

approach of life-cycle models as a meta-proceeding, project managers need concrete methods and proceeding for sequence of events during project proceedings.

The section also presents some examples for software development processes with methods and practical relevance. After reading this section, projects managers should own an overview about different software development process models and their practical relevance.

3 Approaches in Quality Management

As outlined in Software Engineering section several mechanism and SQA activities etc. are placed to support project development, guarantee requirements-meetings and improve quality in all phases of the software life cycle. In this chapter I will perform a closer look at quality, quality management systems and methods and models to achieve "quality".

Quality management is directed towards improvement of product value and service for customers and users. The ISO 8402 (design 1992) defines quality as a collection of functions and attributes of a product or service which must meet special demands or which are associated with requirements [66]. Quality management, as defined in the DIN ISO 8402, concerns all activities as a whole, to define quality policy, goals and responsibilities and resources of quality planning, quality control, quality assurance and improvement to realize a proper quality management system.

According to [51] quality policy provides a framework in a company with respect to quality products defining goals and environment. This preconditions must concern *personal aspects*, e.g. training, including employees into quality processes and improving motivation, organizational aspects, e.g. definitions of workflows, responsibilities and interfaces between different areas and *technical aspects* concerning, e.g. tools and their maintenance.

Sommerville structures quality management into three principal activities [60]:

- Quality assurance
- Quality planning
- Quality control

Quality assurance (or *Software Quality Assurance* (SQA)) can be considered as a collection or even a framework of organizational proceedings and standards, e.g. ISO900x (*International Organization for Standardization*) and CMM (Capability Maturity Model). These SQA standards arose in the last 25 years [54] and still gain in importance.

To implement a proper quality management within a project to achieve a high-qualified product, different procedures, techniques and standards are selected and adopted to meet the corresponding requirements. This basic proceeding is called *quality planning*.

Quality control represents the inspection of the used procedures and standards as well as the observation of their guidelines, defined in the quality plan.

Quality management (QM) should be separated from project management (PM) so that quality is not compromised by management responsibilities for budget and time schedule [60].

Due to different approaches of quality management (involving the mentioned topics) different QM-system arose, some of them are described in more detail in the following chapter.

3.1 Quality Management Systems

A quality management system is embedded as an independent area within the organizational hierarchy (horizontal as well as vertical) of a company [51]. This QM-System has not only been written down in different handbooks but must be established in the mind of every person working in the company. According to the ISO9000-1, which describes the most general approach of QM-systems) the QM-system is defined as "organizational structures, proceedings, processes and methods for the realization of quality management". (Original title: "Zur Verwirklichung des Qualitätsmanagements erforderliche Organisationsstruktur, Verfahren, Prozesse und Mittel" [51]. The IEEE defines QM-systems for the area of software engineering in more details [30].

"The developer shall maintain an effective system for quality management, planned and developed in conjunction with other functions, which shall be documented. Requirements shall be met by the establishment and implantation of procedures which have the specific purpose of ensuring that only software conforming to contractual requirements is delivered." To follow this approach, different tasks must be established in the software engineering process (which will be described later in the text).

- Quality affecting factors must be recognized, monitored and, if the don't meet its requirements, revised.
- Quality requirement, including e.g. development, inspection, testing, shipping, installation and maintenance have to be determined as well as standards or procedures must be established to satisfy the requirements.
- Early error detection and corrections methods, which must be effective as well as timely.

These aspects seem to be very restrictive containing regulative mechanisms and a lot of management and documentation overhead. Because of very diverse areas there could not be one single solution but only a framework or even basic requirements, *what* must be done to establish a QM-system not *how* it is done [51]. These requirements are summarized and are the basic for different standards of QM-systems, e.g. ISO series, CMM, SPICE, Bootstrap and quality awards like EFQM (European Quality Award), etc. I will describe ISO 9000 and CMM in the next section in more detail.

Feedback using Deming's Cycle: One of the key-topics of Deming's theory is a control-mechanism for continuous improvement of products quality – the quality circle with the stages "Plan-Do-Check-Act" [63]. It provides feedback within a sequence of projects or even a project part with regard to products improvement. According to [9] the Figure 3.1-1 shows the context between Deming's quality cycle and feedback-mechanism for software development.

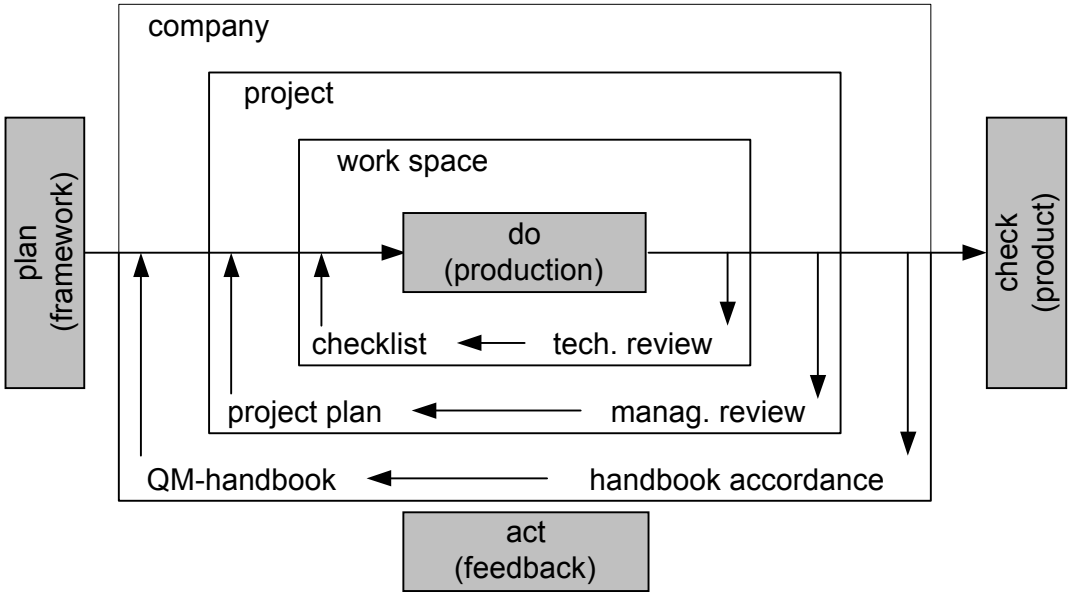


Figure 3.1-1: Plan-Do-Check-Act (PDCA-cycle) according to [9]

Thaller [63] explain the PDCA-cycle in the following way:

- Plan: A product must be planned and designed according to needs, requirements and desires of the client.
- Do: product production and analysis
- Check: Sale of the product as well as analysis for further product improvement.
- Act: Products investigation after selling it, according to functions, quality, pricing etc. with special respect to customers and user. This phase includes questioning possible customers in future.

To meet all requirements there must exist a standard or proceeding model according to quality as well as to software engineering. Some approaches with respect to software engineering have been described in chapter 2.

The Quality Assurance (QA) process involves *defining* or *selecting* standards that should be applied to the software development process or software product [60]. Sommerville identifies two types of standards:

- Product standard
- Process standard

Product standards apply to the software product itself. It contains *document standards*, e.g. the structure of a requirements document, *documentation standards*, e.g. comment headers within object class definitions and *coding standards*, e.g. how to use a programming

language. But also the verification of the product is implemented within a product norm [37], ***Process standards*** define the processes which should be followed during software development. An underlying assumption of quality management is that the quality of development process directly affects the quality of the products. One simple controlling circle can be found at [60] which describes production and quality improvement of a product.

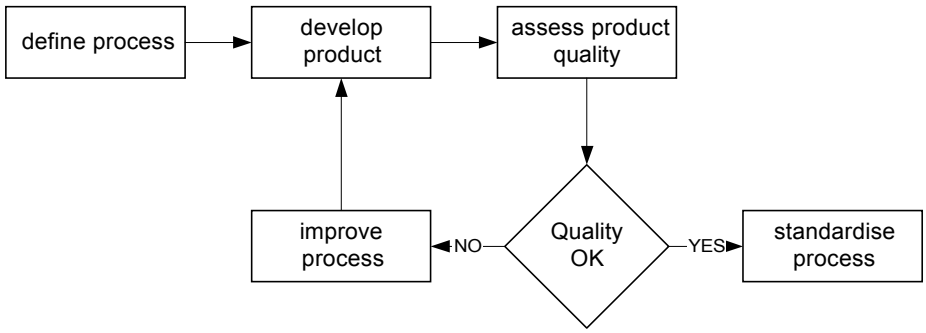


Figure 3.1-2: Process based quality according to [60]

There are two main reasons for the establishment of norms, each developed by different institutions following other approaches.

- Software quality grows to key factors of business competition, caused to customer's quality understanding.
- Corrections of defects in the product are very expensive, so they must be reduced as soon as possible using quality management systems.

Norms help companies to find the best solution for their own business-area, which is not limited to software at all, but can adapted to many different fields [37]. Some of them, especially the ISO900x series and CMM will be described in the following section.

3.1.1 DIN EN ISO[3] 900x series

The ISO 9000 family is one of the most important standards in the field of software quality management in Europe [61]. The ISO9000 series is aimed to provide criteria for the company and customers to assess quality skills of the company itself and to give an overview about the quality levels of its products and services [37].

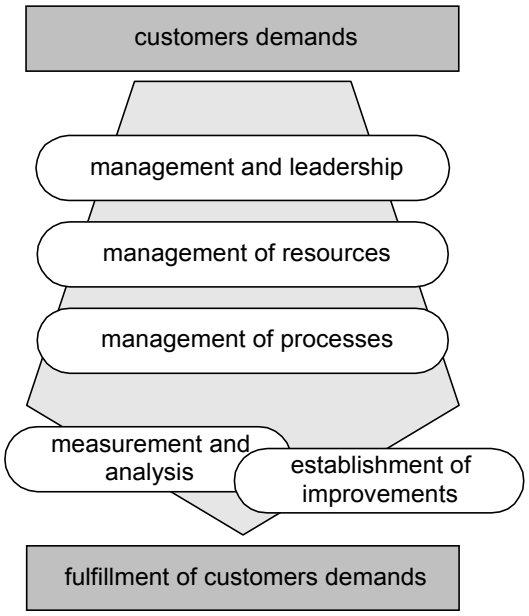
In the publication of ISO 9000 in the year 1994 the following brief structure could be found: The ISO 9000-x series provides an overview about the whole family of norms and supports its selection. The most important "real" norm is the ISO 9001, which describes a "model for quality assurance in design, development, production installing and servicing". The ISO 9002 and 9003 limit the scope of ISO 9001 to different business-areas. ISO 9002 concerns

[3] ISO will be used as a short-cut in the text

production, installing and servicing and ISO 9003 is limited to final checks only. Because they are not relevant in practices, they were called in again. The most interesting norm is the ISO 9004, which concerns all aspects, which are not documented in the ISO 9001 – it is called "guidelines to quality improvement" [63].

The norm consists of 20 chapters[4] concerning nearly every step in the business of a company, e.g. development, inspection, defect handling and maintenance. Although the norm is valid also for software development, no detailed information can be found there (reading "between the lines" is necessary to find out), a new norm – the ISO 9000-3 – has been developed, concerning software development and its specials needs (Original title: "Qualitätsmanagement- und Qualitätssicherungsnormen: Leitfaden für die Anwendung von ISO 9001 auf die Entwicklung, Lieferung und Wartung von Software"). The standard is separated in 3 major parts, one concerning the framework of the QM-system, the second describing life-cycle activities and the third one provides supporting activities in establishing a QM-system. Though its structure differs[5] from the ISO 9001 because of a better understanding and usage by software engineers.

Due to many modification of the norm and the check of its "state of the art" the norm, especially the ISO 9001 has been revised at the end of 1999. This revision leads to more process oriented structure (concerning Total Quality Management, see at [55] for a practical view of TQM)



The main changes with respect to the previous version of the ISO series can be summarized in the following facts:

- A more process-oriented view on workflows and proceedings
- Emphasis on customer relationship
- Explicit resource management of personal, infrastructure and equipment
- The need for internal communication
- Continuous improvement
- Defect detection and prevention

Figure 3.1.1-1: ISO9001:2000 – an overview according to [63]

[4] An overview of the 20 chapters of ISO 9001, release 1994 can be found at appendix A.1

[5] A brief summary and a comparison to ISO 9000 can be found at appendix A.2

This new approach of the ISO 9001:2000 is structured in a different way containing a more process oriented view concerning additional requirements to quality management systems.

Figure 3.1.1-1 shows the five main levels of quality in top-down design, (i) management and leadership, (ii) management of resources, (iii) management of processes and two components closely engaged to each other, (iv) measurement and analysis and finally (v) establishment of improvements. In this text I will only show some important facts with respect to software engineering[6].

The *management level* includes definitions of quality policy and targets as well as their verification and review. They must also establish an orientation towards customer needs and take care about the QM-system as a whole. The QM-System must be checked at predefined time-intervals to guarantee usefulness and actuality to find out how to improve and revise quality politics and goals. *Management of resources* concerns people as well as information and all kind of equipment to achieve quality conformity of products. *Management of processes* fixes demands on processes with special respect to product development and manufacturing, customer relations, defect handling and prevention and management of those processes itself. The basic elements of the new ISO norm can be summarized with *measurement and analysis* and after evaluation the *improvement* of product, processes etc. Special topics within the section 8 of ISO9001:2000 concern aspects for customer's satisfaction, internal audits, process- and product evaluation and improvement.

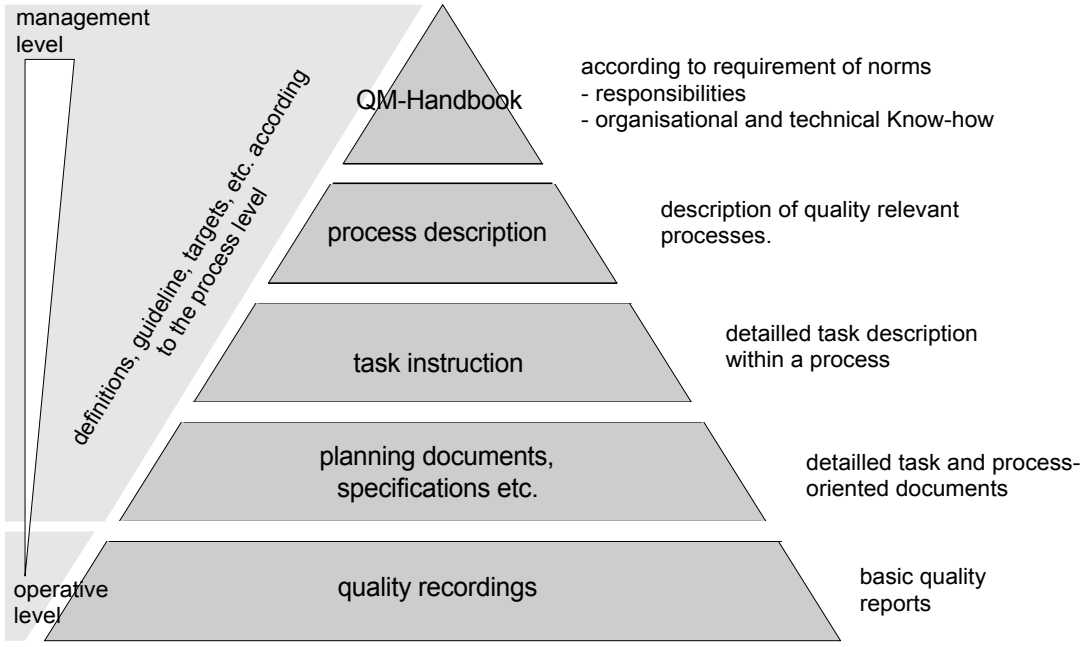


Figure 3.1.1-2: QM-systems documentation according to [37]

[6] An overview about the revised ISO9001:2000 can be found at appendix A.3

Every company who is interested in using a quality management system must select proper norms and standards, (introduced in the ISO9000-X) which fits best to its business-area. There must also be documentation of the system for every level of responsibility and all key processes of the company. Figure 3.1.1-2 describes the context of possible documentation mechanisms according to the norms structure.

The pyramid of documentation includes basic leadership information (at the highest level) such as quality policy, responsibilities, workflow management for general purpose within the whole company and other managerial information. Every "lower" level in this schema presents a more detailed view on the processes, describing concrete approaches of specific workflows. At the most concrete level of documentation, quality recordings are used to check process outputs to their defined targets, e.g. defect reports found at reviews, etc.

Although the ISO standard is a wide spread and good approach for establishing a quality management system, there exists several critics on this approach; I will cover only a few important one:

a) ISO 9000:1994 and ISO9000:2000 is certainly not just a software standard, but covers a wide range of application areas [56]. Because of the importance of software development and its quality relevance the standard ISO9001-3 has been released interpreting and advancing the original standard with special respect to software.

b) ISO is not just a standalone standard, because every company will have to use different standards (out of the ISO-series) to achieve predefined goals like product improvement etc.

c) ISO depend on top-down centralized quality management. It depends on the implantation of the quality system. ISO does neither prevent a bottom-up strategy no does it prohibit active participation of all employees [61].

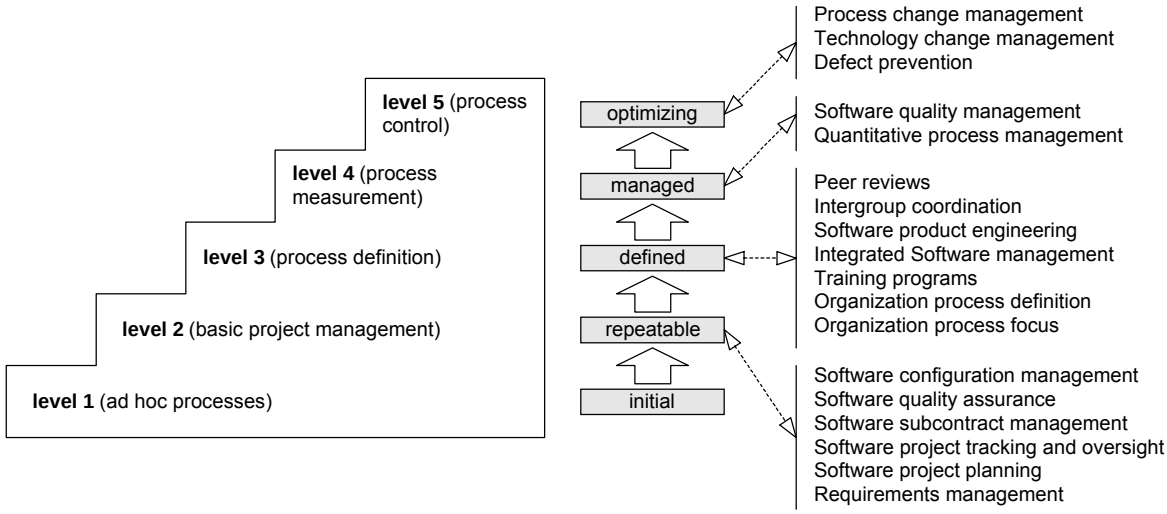
d) ISO promotes documentation because "any aspect that has not been documented virtually does not exist in an ISO 9000 environment" as a highly important element [61].

Once the ISO norm has been established, a certificate documents its implementation. The assessment according to ISO shows, that all minimal requirements are met, according to the text of the norm. The individual chapters must be documented in a so-called quality management handbook, they must be established in the whole company and they must fit together.

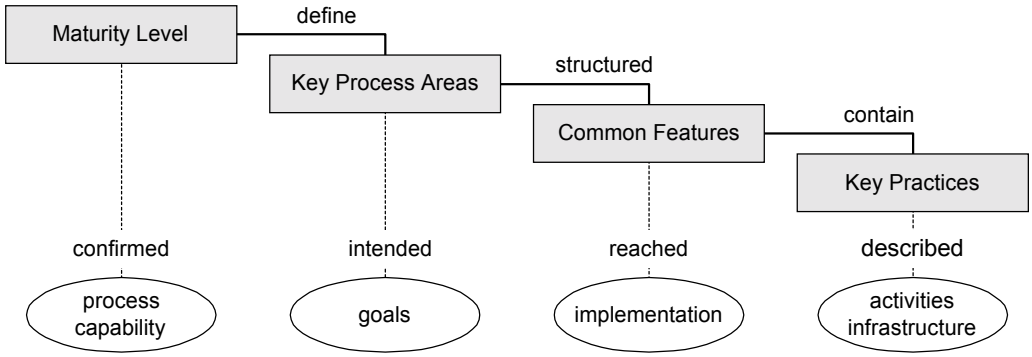
Another important approach of Quality management systems is CMM (capability maturity model), which focus on the definition of proceedings for the improvement of processes and shows its strength and weakness.

3.1.2 Capability Maturity Model (CMM)

The first applicable version of CMM has been introduced by Watts in 1986 at the Software Engineering Institute (SEI) at Carnegie-Mellon University. It is built on ideas of Crosby, which invented its basics in the 1960s [63]. The basic strategy of the CMM is to improve management of software process in the belief that improvements in technique will be a natural consequence [56].



(a) CMM assessment levels



(b) concept of key process areas

Figure 3.1.2-1: SEI capability maturity model according to [60]

The CMM model defines five basic stages of "maturity", each equipped with predefined key-process areas and a range of common features and key practices. "A cluster of related activities that, when performed collectively, achieve a set of goals considered important for establishing process capability. The key process areas have been identified by the SEI to be the principal building blocks to help determine the software process capability of an

organization and understand the improvements needed to advance to higher maturity levels." [69][7]

There is a strict separation between the levels and one (higher) level can only be achieved on condition that all lower levels have been fulfilled so far. Figure 3.1.2-1 shows the basic elements of the CMM model.

The *initial level* (maturity level 1) no management practices with respect to processes and quality take place, everything is done in an *ad hoc* manner. Project success depends mainly on the skills of engineers. Characteristics of software and software processes will be unpredictable [56].

The second maturity level (*repeatable)* allows a project proceeding of similar types (repeat it) due to organizational guidelines (basic project management). There exist a formal management, quality assurance and configuration control mechanisms. One premises is the usage of measurements to get process results and according to them, different activities can be proceeded [56][60].

If one organization owns clearly defined processes, which are the basic "material" for process measurement (e.g. reviews), it is classified at level three (*defined*). The process for software production is fully documented and management concerning technical aspects) [60]. Schach describes that there are a lot of companies, which have attained level 2 or 3, but at the moment (1996) no one has reached level 4 or even level 5 [56].

A company who reached level 4 (managed) must have a defined software process including formal programs of quantitative data collection. Process and product metrics are collected and fed into process improvement activities [60]. Quality and productivity goals are defined for every project and both are continually measured e.g. using statistical quality control mechanisms.

Finally, at the 5th level (*optimising*) companies must implement continuous process improvement, which is planned as an integrated part of the organization's process [60]. Results from earlier projects must influence all future projects. Statistical quality and process control techniques are used to guide the organization [56].

To mention again, a company, who wants to reach CMM level 4 has to fulfil all lower levels, that is level 1 to 3) and – of course – level 4.

For the majority of areas, there is a clear correlation between the key processes and the ISO 9000 standard. The CMM is more detailed and prescriptive and includes a framework for process improvement [60]. ISO doesn't provide this so far. Nevertheless organization whose process maturity is rated at level 2 or 3 are likely to be ISO 9000 compliant, but also some companies at CMM level 1 can fulfil requirement of the ISO standard.

[7] SEI: CMM based Assessment Method description

But using CMM some problems occur, which can be found at [60]:

a) The model focuses exclusively on project management rather that product development.

b) It excludes risk analysis and resolution as a key process area.

c) The domain of applicability of the model is not defined.

Like at the ISO standards, assessment also occurs at CMM. It is based on a standard questionnaire to identify several key process areas. According to this key processes, different people are interviewed – including feedback cycles – to find improvements for processes. The assessment finishes with a presentation of assessment reports and a score for CMM classification. Not like at ISO assessments, there is no strict regulative norm but individual process descriptions.

3.1.3 Further approaches of quality management systems

Because ISO 9000 as well as CMM owns its individual weak points, SPICE (Software Process Improvement and Capability DEtermination) has been developed to reduce them. The quality model has grown to an ISO standard, named ISO/IEC TR 15504. It is based on three major blocks, process assessment, process improvement and process capability determination each containing key-process areas with 5 elements. The main difference to CMM is at level 1 “initial” (at CMM) and “performed” at SPICE. While companies are a priori at CMM level 1 (without performing an assessment), they have to perform basic workflows including elements of quality management to achieve SPICE level 1. Another difference is the strict separation of maturity levels at CMM (to achieve e.g. CMM level 3, all key process areas for level 1, 2 and 3 must be fulfilled). The assessment of SPICE focus on processes, so its possible that some elements are at SPICE level 3 others only at spice level 1.

Another approach of quality management systems special with respect to software engineering is BOOTSTRAP. The model contains elements from ISO 9000-3, CMM, SPICE and the Software-Standard ISO 12207, which define a framework for software-life-cycle processes.

As outlined in section “Software Engineering” and “Quality Management” there must be some methods, tools and models to (i) get quality data, (ii) analyse them and (iii) find improvements to achieve products with higher quality.

3.2 Quality Management Methods to support Software Development

This section provides some important methods for product quality improvement with special respect to early development stages. The main reason minimize defects in very early development stages, i.e. requirements, planning or specification phase within the software life cycle are time and cost. Obviously it will be more time and cost consuming to repair defects, which are identified at later life-cycle activities (in worst case a defect is identified in

maintenance phase at the customer). In this way I will cover audits (quality management) and reviews (software engineering), although there exist many more methods, like testing etc.

3.2.1 Assessments / Audits

"An activity to determine through investigation the adequacy of, and adherence to, established procedures, instructions, specifications, codes and standards or other applicable contractual and licensing requirements, and the effectiveness of implementation [ANSI N45.2.10-1973].

Assessments (*assessment audits*), as described in the previous chapters are some kind of audits. The term "*assessment*" has been used in the context of QM-systems to achieve a certificate for ISO standards or CMM levels according to corresponding norms and standards. In general "to assess" means proving or checking something against something else. E.g. at a assessment according to ISO standards (the norm itself), the quality framework of company is checked based on the quality documentation – the QM-handbook – of the corresponding area and the norm. Assessments are classified at management level. They will be performed by the management or even by a third party (*certification authority*) in case of a certification. Thaller describes assessments as a rating or review of the organization and processes for developing software [63].

Subject	Audit (DIN EN ISO 9001)	CMM
Base	Standard, norm	Description of key processes
Coverage	Not limited to subject of companies	Software norm / standard
Method of revision	Audit	Assessment
Auditors	2-4 persons (not only software specialists required)	3-5 persons (software specialists
Duration	1-2 days	About 1 week
Result	Rating	Certificate
Goals	- Certificate, if a company fulfil the requirements of a standard, namely the ISO 900x standards - No suggestion how to achieve continuous improvement - Gives an idea of quality capability of the company, its products and processes	- Classification of a company (which level of some process model (e.g. CMM) - Analysing and understanding of processes and techniques used. - Finding of strength, weakness and possibilities for improvement - Suggestions for improvements (!) - Description of (sub)processes for continuous improvement

Table 3.2.1-1: Comparison of ISO vs. CMM and audits vs. assessments according to [63]

Both methods (audits and assessments) aim to find weak points, get information about some subjects or check those subjects according to requirements, standards or even specifications. Product, process or even the whole (or parts of a) quality management system can be the subject of an audit or assessment. Some differences between audits and assessments can be seen in Table 3.2.1-1.

Audits are used in many different areas of a company or even projects, with the aim to find weak points, get initial stages for improvement and supervision of quality measures [51]. At audit level several different kinds of audits can be found, there also exist several special proceedings exist. Pfeiffer distinguishes between “product”, “process” and “system-audit” [51].

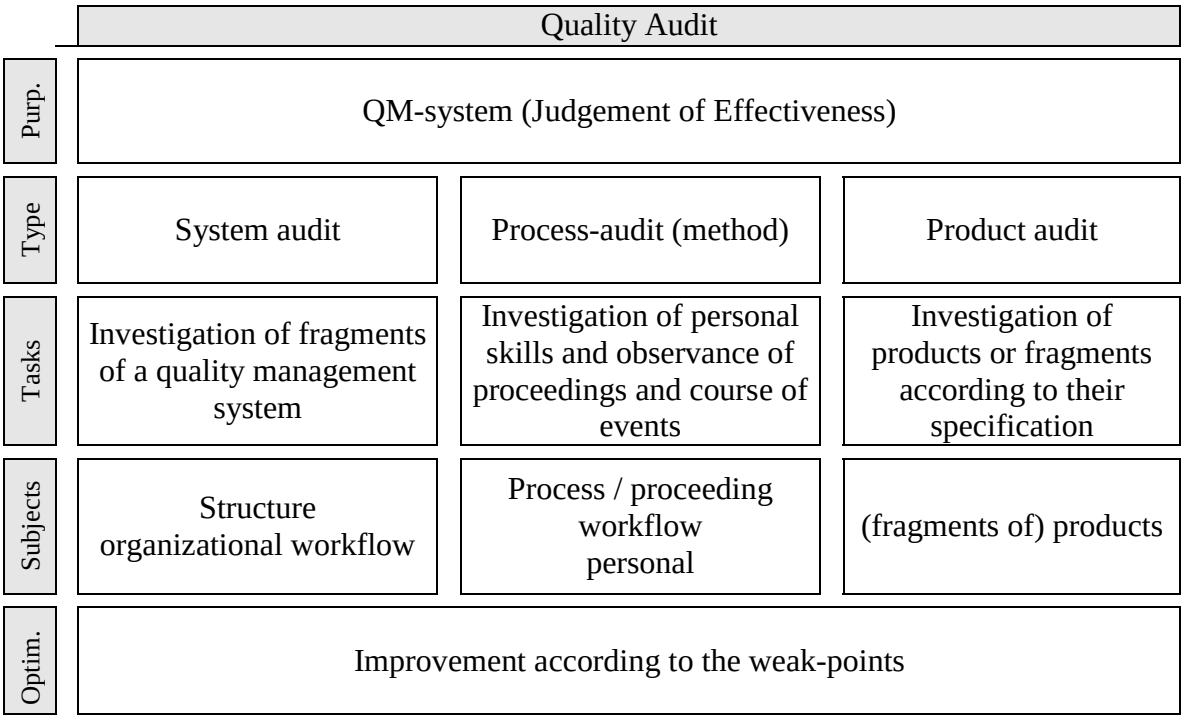


Figure 3.2.1-1: Different kinds of audits according to [51]

The subject of *product-audits* is either the final product or even a product at a defined phase at the life-cycle-model (concerning software engineering) or any temporary product for further proceeding (concerning production companies). The main goals at both approaches are checking for *quality characteristic* (and if the meet requirements), *investigations of defects* (e.g. systematic defects, development errors) and their *reason* and finally investigations for *product improvement* [51]. While product-audits concern products themselves, *process-audits* aim to investigate workflows and methods and focus on optimisation and improvement. In contrast to product- and process-audit, the *system-audit* is used to check the whole quality management system (or even critical areas within in the QM-system) to verify its correctness

and efficiency. If requirements are not met or even errors are detected, corrective measures have to take place. The base for system-audits may be the QM-handbook and may be considered as some kind of an assessment for certification.

Another differentiation concerning audits is the kind of implementation. Pfeiffer [51] distinguishes between *internal* and *external* audits. While customers or certification authorities to investigate the proper working of the quality system establish external audits, internal audits are focused on improvement within the company itself. Internal audits are an effective management method or tool to assess quality capability, improve workflows etc.

Table 3.2.1-2 shows the proceeding of a typical audit, which e.g. can be found in a Quality Management Handbook, according to ISO 9001, chapter 17 (internal quality audits).

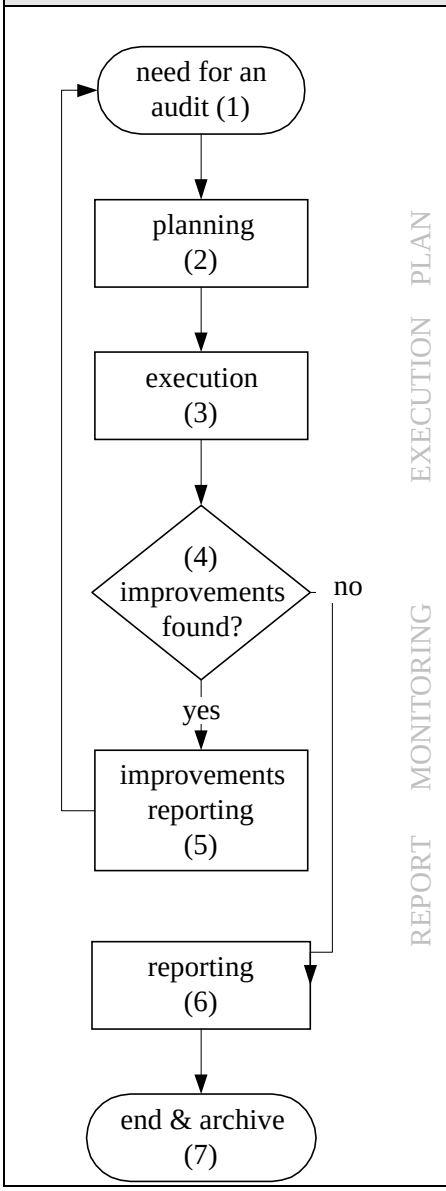
Workflow	No	Activity
	1	*Need for an audit* - Milestone with a software development process - Claim for an audit due to a project- or QM-plan - External Claims (certification, customer, etc.) - Internal claims (project management, QSA, etc.)
	2	*Audit planning* - Topic (which area to be revised) - Definition of auditing team and contact persons - Formal basis (norm, checklists, etc.) - Resource co-ordination (time, people, etc.)
	3	*Audit execution* - Execution of the audits according to audit plans - Documentation of results - Discussion of the results within a team
	4	*Improvement / weakness found* - Discussion of strength, weakness and improvements found at the audit - Maybe arrangement for repeating an audit (in case corrections must take place) - Audit report by the audit team
	5	*Establishment of improvements* - Implementation of improvements (corrections due to registered problems, or optional corrections for continuous improvement)
	6	*Audit reporting* - Written paper by the audit team according to the results and agreed measures

Table 3.2.1-2: Planning and execution of audits

Audits and assessments are classified mainly at management level but also in the whole life-cycle model (e.g. product or process audit). Once, a software document has written, it must be tested to guarantee functionality of the product. A well-know technique is “reviewing something”, which I will cover in the next chapter.

3.2.2 Reviews

Because testing is placed very late in the life-cycle model (in or even after the implementation phase) it is very expensive in money and time to fix defects or even (in worst case) to redesign the product. Table 3.2.1-1 presents the relative cost for removing defects in different selected development stages [63].

Development Phase	Relative Cost
Requirements analysis	1,6
Design	3,5
Implementation	10
Testing (also implementation phase)	25
Test of acceptance (also integration and maintenance phase)	65
Operation (also maintenance and retirement phase)	180

Table 3.2.2-1: Relative cost for defect removal

It’s easy to see that defect prevention must be established as early as possible in the development stages also defects are unavoidable. The main goal is, to reduce them to a minimum. This is the approach, where reviews take place.

IEEE 729-1983 defines reviews as a “a formal meeting at which a product or document is presented to the user, customer, or other interested parties for comment an approval. It can be a review of the management and technical progress of hardware/software development project” [9]. Thaller [63] differentiates between basic approaches of reviews:

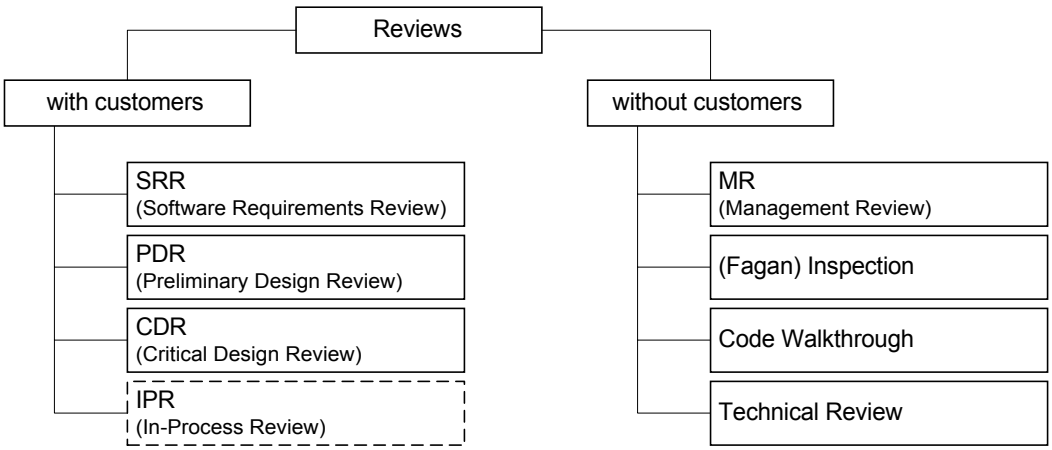


Figure 3.2.2-1: Differentiation of reviews [63] [9]

On the one hand side there exist reviews, made in cooperation with the customer, on the other hand side there exist a lot of internal review-types, without consideration of the customer. The inclusion of the customer at the first stages within the software-life-cycle is very important for him, to get the right product. First I cover some of reviews concerning the customer:

Once the requirements analysis has been finished, but not released as the final version a ***Software Requirements Review (SRR)*** takes place. Usually the check of requirements documents is performed before the design stage, because any lack of clarity on clients' side as well on developers' side must be eliminated at this stage of developing.

The second review (***Preliminary Design Review (PDR)***) is done, when the design stage is in process, to check architectural design of the software and the rough structure of the product before designing it in detail.

Because of a very important milestone (after design stage and before implementation of the program) there is a "last" review (***Critical Design Review (CDR)***), which is strongly recommended, because this will be the last point for the customer to stop the project (without "wasting" too much money). Depending on the size and complexity of the software product, PDR and CDR can be summarized. In this case it will be placed at CDR stage. CDRs purpose is the discussion of the detailed specification because afterwards the very time- and money consuming phase – the implementation of the program – starts.

Concerning very large projects with long implementation phases, it can be necessary to establish a so-called ***In-Process Review (IPR)*** to present implementation results, prototypes or even test cases to the customer [63]. I think it is very important, that the customers know about the projects progress.

To achieve proper results, it is necessary to use additional methods to improve quality and to check software products against their requirements. Some mechanisms of internal reviews (without customers) are management reviews and technical reviews, inspection and code walkthrough.

Management Review is considered as a formal assessment of the project-plan or of the project-status according to such a plan [9]. Usually management reviews are planned within a project plan at several stages of the software life cycle, e.g. requirement phase, design phase etc. But on special need additional management reviews can be established if there occur serious problems or unpredictable facts, e.g. change of responsibilities or change of important project goals.

To assess one special software artefact, ***technical reviews*** are used to see the correspondence between specification or standards and the software product itself and even to find defects [68].

3.2.3 Inspection

One kind of reviews, which is similar to technical reviews, is a ***Code Walkthrough*** (code-inspection). It is a method for defect detection and comparison to standards and requirement documents with the aim for a discussion of potential alternatives and coding details (e.g. programming style etc.). This method provides feedback to the authors without need to change something in the code. Code-Walkthrough is a mechanism for training of developer and support communication and discussion within a team. Some piece of code is checked using different test cases.

One more general approach of a Code-Walkthrough is ***Software Inspection***, which is considered as a more formal way to reduce and prevent defects in all software products. Following this approach, inspections are a quite useful method to meet requirement in software engineering and quality management as well:

a) Inspections support defect prevention in early development stages in the software life-cycle model in the area of *software engineering*.

b) Considering the area of *quality management*, there is a challenge to reduce defects, costs and improve customers satisfaction. Also inspections can support this approach.

Thaller [63] describes Fagan's inspection, which is the first appearance of inspection in literature, in 6 different steps.

1. Planning Phase: After finishing a document by its author, an inspection team is established. One team member represents a moderator who is responsible for organizational tasks (leadership). All other members represent special roles, which are discussed below.
2. Tutorial (optional introduction to inspection methods): New team members or teams, who are unfamiliar with inspection, learn basic rules, techniques with respect to inspection.
3. Preparation Phase: In a typical time-interval of about two hours (might be more if teams are unfamiliar with inspection processes) the inspectors prepare for inspection using tools, e.g. checklists etc. with the propose to get familiar with the application area and the inspection subject in general.
4. Operation: Under direction of the moderator the team inspect the specification document. The main task in this stage is the finding of defects. All defects are documented concerning sort and severity of the defect. There is no need to find any improvements or corrections but only the establishment of defects. The decision about the usability of the document (rework necessary or release) complete the operation phase.

5. Rework: Once the software document must be reworked it's the task of the author to perform this task.
6. Verification: After fixing the problems, the moderator checks the modification and is able – if another inspection is necessary – to start the inspection process a second time.

Nevertheless Fagan's approach isn't a straightforward proceeding but there are relations between each step. Since Fagan introduced inspection in 1976 as a method for defect detection several extensions and also a more detailed inspection process has been developed.

3.3 Chapter Summary

Software Engineering proceedings support project progress, presenting several structured approaches of phases and process steps. The frameworks presented in chapter 2 don*t guarantee high-quality software products, but the chance to achieve good results is quite higher than without structural approaches.

Normally project managers, quality managers, etc. use several methods and approaches to measure quality, to find defects and to manage software development, software engineering processes and, at least, management approaches for company-wide business cases. Control cycles like the PDCA-cycle exit to support continuous improvement of products, projects, etc. at all levels of a company. One step in this cycle is the assessment of products and processes using reviews, audits, inspections etc. at different process levels.

Using different approaches for quality improvement are usually managed within a quality management system. I described actual approaches, like the ISO-series, CMM and SPICE and their frameworks in more detail, to give an overview.

The main purpose and goals of project- and quality managers is to achieve proper products (independent of application areas). One of the approaches for good products is the reduction of defects in the products. I will cover inspection as a special method to achieve these goals in early stages of software development in the next section.

4 Defect Detection with Inspection

Inspections are a *formal*, *efficient*, and *economical* method of finding errors in design and code [24]. Inspection is performed in early stages of software development concerning design phase, especially generating a specification document [63] but according to Fagan, to code as well. I will follow Thaller's approach concerning specification documents primary. Gilb et al. describes inspection as a look at a software product "through a microscope" so that the defects can be discovered [27]. Therefore Gilb et al. see inspection as a method for identifying defects in software documents in all stages of the software life-cycle model as well [27].

Laitenberger et al. describe the technical dimensions of Software Inspection as framework for inspections concerning the *inspection process*, *product artefacts*, *reading techniques* and *roles* within an inspection team as shown in Figure 4-1 [42] [41].

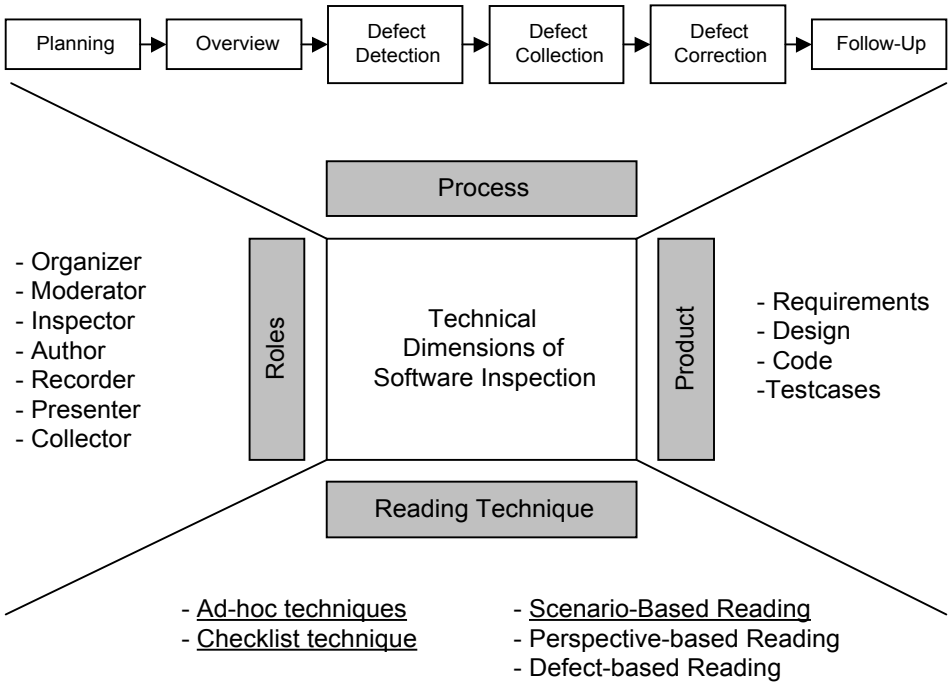


Figure 4-1: Technical Dimension of Software Inspection adopted to Laitenberger et al. [42].

4.1 Inspection Process approaches

4.1.1 Fagan's Inspection Process

Following Fagan, who introduced inspections in the 1970s, he identifies 6 basic steps in the inspection process [63]:

1. Planning Phase: After finishing a document by its author, an inspection team is established. One team member represents a moderator who is responsible for organisational tasks (leadership). All other members represent special roles, which are discussed in chapter 4.3.

2. Tutorial (optional introduction to inspection methods): New team members or teams, who are unfamiliar with inspection, learn basic rules, techniques with respect to inspection.
3. Preparation Phase: In a typical time-interval of about two hours (might be more if teams are unfamiliar with inspection processes) the inspectors prepare for inspection using tools, e.g. checklists etc. with the propose to get familiar with the application area and the inspection subject in general.
4. Operation: Under direction of the moderator the team inspect the specification document. The main task in this stage is the finding of defects. All defects are documented concerning sort and severity of the defect. There is no need to find any improvements or corrections but only the establishment of defects. The decision about the usability of the document (rework necessary or release) complete the operation phase.
5. Rework: Once the software document must be reworked it's the task of the author to perform this task.
6. Verification: After fixing the problems, the moderator checks the modification and is able – if another inspection is necessary – to start the inspection process a second time.

4.1.2 The Process Sub dimension according to Laitenberger [42]

1. Planning: The planning phase aims to organize the execution part of the inspection process. After passing entry criteria the right inspection team, regarding inspectors capability and role assignment within the team, must be selected, a proper scheduling must be defined and inspection material has to be split up and distributed to the team members.
2. Overview: The overview phase represents the first meeting ("kick-off-meeting" [27]), where the author gives a first overview about the inspection product. According to [42] it is sensible to proceed this phase if (i) the inspected product is very complex and difficult to understand and (ii) it is part of a large software system. In both cases the author is able to explain context as well relationship to other parts of the system. Fagan doesn't support such a kick-off meeting because it may prohibit an independent inspection process but he supports a kind of "*Tutorial*" or "*Preparation Phase*" for team members to get familiar with inspection processes in general [24].
3. Defect Detection: The aim of the defect detection phase is the most important core of the inspection process. In literature there are different approaches concerning the approach of defect detection either as an individual activity or a group (team) activity. In Fagan's opinion there are synergy effects performing defect detection, as a group activity and the team will find more defects than the summary of all individuals. On the other side, Votta (1993) didn't find significant synergies in his empirical

evaluation [15]. Another approach is performing an individual defect finding process and afterwards generating a team defect list using group synergies.

4. Defect Collection: The aim for a defect collection phase is to filter out real defects and to document them. In most cases this is done in a team (group) meeting. Also the decision to perform a second inspection cycle, a so-called Re-Inspection, is discussed.
5. Defect Correction: Once a defect list containing real defects has been generated, the author has to improve the software document according to the reported defects.
6. Follow-Up: This – in many cases optional – phase represents the check of the author's modification in the software document by the inspection team.

4.1.3 Inspection control

Additional to Fagan's basic process steps and Laitenberger's sub dimensions, Biffl describes an inspection model embedded within project and quality management containing also an inspection management [15].

The boxes shown in Figure 4-2 describes two management approaches: *the quality management in the project*, which decides on the execution of an inspection process and the *inspection management*, which is responsible for the implementation of the inspection. In the model of [15] there occur three different layers, which represent different views on the inspection process according to the inspection level.

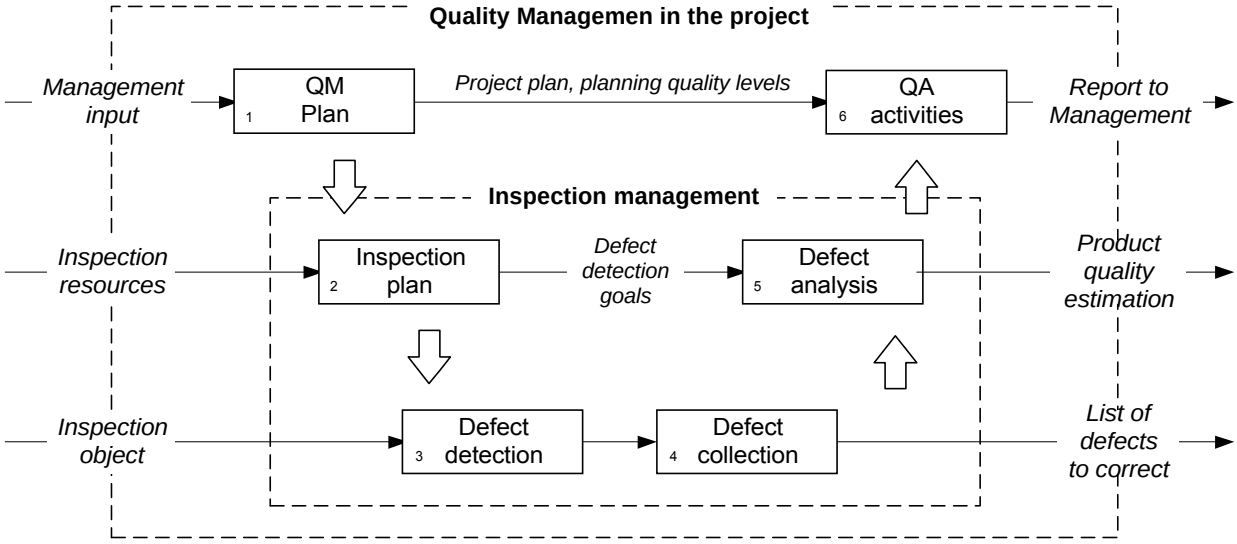


Figure 4-2: Framework for inspection planning and control according to Biffl [15]

1. Project use of inspection results: This layer regards the interface of risk management in a project to inspections and quality concerns in the whole company. Reuse of inspection data (e.g. building a knowledge base) and the management's decision to establish inspection processes (according to cost-benefits, inspection design, etc.) at a specific project represent some key aspects at this management level.
2. Inspection management: Once an inspection process is established by the management, it's the challenge of the project- and the local quality management to build up the whole environment and set up proceedings at the beginning and to perform several data and product quality estimations after processing.
3. (Technical) inspection conduct: According to the inspection plan, defect detection and the inspection team does collection.

The following subsections discuss the individual boxes in the Figure 4-2 will be proceeded sequentially.

Process 1: Quality management (QM) plan, economic model: Input from the management level for the inspection process is defined in the QM-plan. Also defect classification and their value (related to the cost to repair factors in different development stages) are part of the QM-plan and the economical model to achieve a most efficient and effective inspection process.

Process 2: Inspection plan, reading techniques: Once an inspection process is established, it's the project managers task to set up the inspection environment, including products, inspection staff (e.g. concerning availability, capability and team composition) and guidelines (e.g. reading techniques, checklists) according to the type of products.

Process 3: Defect detection: During defect detection the inspectors apply their defined role within the inspection team and generate individual defects list in parallel. Additional information about the inspection process can be achieved by getting feedback from the inspectors. The investigation of feedback results is basic material for process improvement.

Process 4: Defect collection: After finishing the individual inspection process all defect lists are balanced by the inspection teams to select real defects and classify them for defect correction by the author. Defect collection can be done either at a team-meeting process (including discussion, gain or loss of defects within a team) or as a formal team operation (summary of defects-classes or individual defects according to the inspection plan).

Process 5: Defect analysis: The analysis of the team defect list result in statistical information to estimate the remaining defects in the inspected software document and on feedback on the inspection process itself. Additional project managers are able to draw conclusion from defect list about product quality. Using defect content estimation techniques (see [16] for more detailed information) is some kind for an in-process control, which can result in a second inspection cycle [10]. Analysing product quality after developing with respect to inspection results are interesting for future projects and process improvement strategies (a posteriori information).

Process 6: Quality Assurance activities: Based on the original project plan, QM-plan, economical model and results form defect analysis (e.g. defect content estimation) further quality assurance activities can be established.

4.2 Products

As described in chapter 2, which shows the cost distribution defect removal, it is more expensive to remove a defect in later stages of development rather than in earlier project stages. Laitenberger et al. presents the distribution of the use of software inspections on various document types [42]. In a study, carried out by Briand, Emam, Fußbroich and Laitenberger in 1998 titled "Using Simulation to build inspection Efficiency Benchmarks for Development Processes" he found out, that the introduction of code inspections saved 39% of defect costs compared to testing alone. The introduction of design inspections saved 44% of defect costs compared to testing alone.

Following these results it will be the best approach, using inspection in early development stages but also in all other phases of software development as well.

4.3 Roles

A team consisting of several persons (usually software developers) carry out the inspection process. Each team member is assigned to a different role. Laitenberger et al. distinguish between 7 different roles, as outlined in Figure 4-1 [42]. In comparison to Fagan, which introduces only 4-5 roles according to the roles in the software life-cycle model [24].

The *Organizer (1)* is responsible for the planning of the inspection process (inspection plan) and, in most cases as the project manager, also for the development of the whole software product. The *Moderator (2)* represents the *key person* in a successful inspection process. Although he must have special skills depending on the inspected area, he but should be integrated in an unrelated project because of independence [24] [42]. The *inspectors (3)* are responsible for defect detection as team members and – if a meeting is done – for the discussion and documentation. If a team meeting is done, there are two additional roles: A *Reader / Presenter (4)* is responsible for the leading through the inspection document in order for a discussion, and a *Recorder (5)* whose task is to document the "real defects". If there is no meeting, a *Collector (6)* summarizes the individual defect list for further proceeding.

Once a defect list has been generated, the *Author (7)* has do modify the document according to the defect list. Laitenberger et al state that the author must not be a moderator, reader or recorder but must support the inspection process explaining specific questions in case of a lack of clarity. In opposite, Fagan doesn't allow the author's presence because of independence of the inspection team [24]. After choosing team members and assigning them to specific roles, there's still the problem, how the document must be read. There exist several methods for reading a software document. Some of them I will discuss in the next section.

4.4 Reading Technique

To increase the effectiveness of an inspection team, methods must be established to support inspection participants in handling the document. Reading techniques, i.e. how to read a software document, can be used. A reading technique can be defined as a series of steps or procedures whose purpose is to guide an inspector in acquiring a deep understanding of inspected software entity [42]. It's up to the inspector to use the advantage of reading techniques.

4.4.1 Ad-hoc reading

Inspectors don't use any special approaches of reading technique it is called ad-hoc reading, i.e. no predefined support for defect detection, Using this technique, defects will be found anyway even more by more experienced inspectors.

4.4.2 Checklist based reading (CBR)

Reading support is realized using the form of questions, which inspections must answer while reading the document [42]. Appendix B.3 (Questionnaire) and C.1 (Task-Description) shows the basic material for checklist-based reading according to the SWT[8]-Experiment.

Although checklist based reading is more accurate for defect detection, Laitenberger et al. identifies four principle weaknesses:

a) Checklists represent the past: Checklists are generated before the inspection process is started and therefore the checklist is based upon past defect information (in worst case, this information doesn't fit to the application area at all or there may be a lack of defect classes, if they didn't appear in the past).

b) Range of questionnaire: Because of reading techniques structure, as a collection of questions there might be a very high volume of questions. It is quite difficult for an inspector to decide how he should answer one specific question (relation between questionnaire and defects detected).

c) Documentation problem: Once an inspector found one defect there seems to be no necessity to document especially but on the questionnaire. This will result in a lack or reconstruction for other inspectors (e.g. for collector role).

d) Coverage: The questionnaire covers all parts of the documents so there will be an overhead with unnecessary details.

[8] SWT is a short term for "Institute for Software Engineering and Interactive Systems" at Vienna Universitiy of Technology

4.4.3 Scenario-based reading (SBR)

Figure 4.4.3-1 adopted from [4] shows a fragment of the family of reading techniques concerning scenario-based reading. Appendix B.2 and C.2-C.4 (Task-Descriptions) shows the basic material for scenario-based reading according to the SWT-Experiment.

For the problem space "defect detection", the specific goal of reading processes, he focuses on requirement documents (as the software artefact being inspected), written in English, which presents the notion of the document.

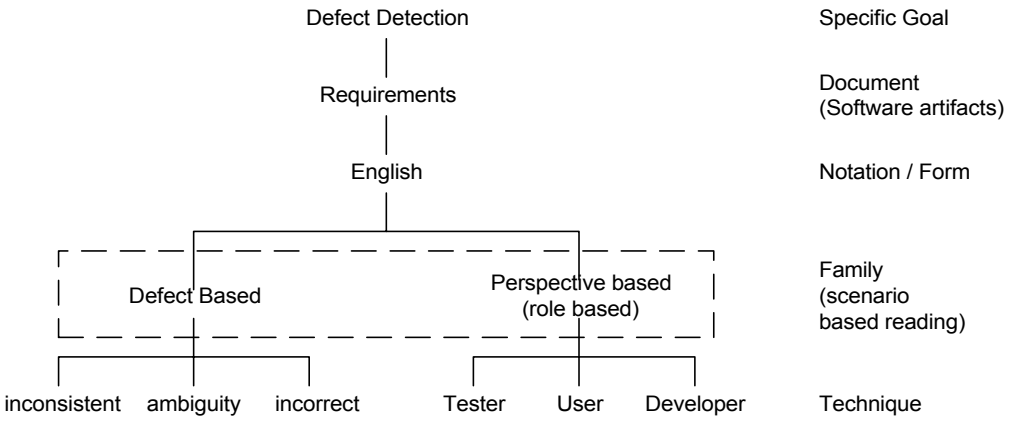


Figure 4.4.3-1: Families of Reading technique according to [4]

This scenario-based reading technique focus on the notion of scenarios that provide custom guidance for inspectors how to find defects, using a set of questions or a more detailed description for proceeding. The main goal of scenarios is the limitation of inspector's attention to a predefined set of defects. Basili et al. describes two approaches for scenario based reading, (i) defect-based reading and (ii) perspective based reading [4].

Defect-based reading (DBR) focuses on particular classes of defects (e.g. inconsistency, incorrect functions, ambiguity or missing information [4]), supported by characteristic scenarios containing a set of questions. Answering these questions helps an inspector primarily detect defects of the particular class [42].

Perspective based reading (PBR) focus on different product perspectives, e.g. reading from the perspective of the software designer, the tester, the user, etc. [4]. Using a role-specific questionnaire and according to the specific point of view different defects can be found, e.g. incorrect facts, ambiguity, inconsistency, etc.

Software inspection, using specific reading techniques, is a useful instrument for defect detection in requirements documents. Because there exists a wide range of different reading techniques (see also [4] [42] [15], it is quite interesting to find out, which approach is the best

one to find as many defects as possible. Because inspection itself is a team activity, it might be useful to combine different reading techniques within a team to achieve better results. Do team members prefer one technique to another? To find answers to these questions, it might be useful, to set up and perform an experiment. The basic steps of introducing experiments in the area of software engineering are outlined in the next chapter.

4.5 Chapter Summary

Software inspections are a useful method for defect detection in early phases of software development, e.g. a specification document. The usage of inspections will lead to better software products and less time and cost during maintenance and error correction.

This chapter presents a basic framework for the technical dimension of inspections, a brief historical review and a inspection process including inspection control. The text and the experiment, described later in this thesis, focus on the usage of reading techniques. Some basic theoretical information with respect to reading techniques has been introduced in the chapter too.

A more detailed view on inspection processes and their practical relevance with respect to the experimental part and experiment design will be presented in chapter 6.

5 Software Engineering Experiments

At project initiation a project manager has to decide, which software development model will be used, which kind of methods for improving products quality should be established etc. It depends on the skills and experience of the project management to make the right decision using a well-known model. But it is also possible to "try" some new or even modified methods, which might fit exactly to the requirements. Because there is less information about a new approach, he can do this in an experimental way. The following section gives an overview about experimental approaches in the area of software engineering.

The major reason for carrying out quantitative empirical studies is the opportunity of getting objective and statistically results regarding the understanding, controlling, prediction, and improvement of software development. They are important input to the decision-making in an improvement seeking organization [67].

According to different software engineering processes as well as quality improvement processes, there must be any possibility to test modified or even new models. One possible approach is the comparison of similar models or simulating them. But the main problem using this approach is, that this comparison or simulation is still based on models and cannot be applied to reality in the same way.

The only real evaluation of a process or process improvement is, seeing it in action while people are using it [67]. He also says, that experimentation provides a systematic, disciplined, quantifiable and controlled way of evaluating human-based activities.

According to Basili and Glass, Wohlin et al. mention four research methods in the context of software engineering [67]:

- Scientific method: This method aims to generate a model, e.g. a simulation model, based on observation of the real world and is used for simulation of e.g. a telecommunication network.
- Engineering method: The central point is an existing method or model, which is studied in detail. After studying the model and adoption to special needs it will be evaluated. This approach is widely spread in industrial usage.
- Empirical method: One selected model will be evaluated using empirical studies or experiments. This approach will be used originally in areas of psychology or even social science; it is where human factors should be analysed. Because of a high dependence to humans in the area of software engineering, Wohlin applies those theoretical approaches to software engineering.
- Analytical method: Based on a formal theory (of some method) these theoretical results are compared with empirical evaluations. Because of the premises this method is applied in more formal approaches of software engineering.

5.1 An overview of empirical strategies

The first distinction concerning experimental paradigms is *qualitative* versus *quantitative* approaches. Wohlin et al. regards both as complementary rather than competitive. *Qualitative research* is concerned with studying objects in their nature using e.g. interviews to users to evaluate them. The researchers task is to find interpretations of the results. This can be very difficult because there exist a wide range of different interpretations. As a complementary method quantitative research is used to quantify a relationship or a comparison of different groups to identify a cause-effect relationship. To meet this approach controlled experiments or case studies are set up, executed and analysed using e.g. statistical investigation. It depends on the area of investigation as well as on the basic goals of evaluation, which approach to use.

Wohlin et al. identify – according to Robson – three major types of investigation strategies, depending on the (i) purpose of the evaluation and on (ii) conditions for the empirical investigations [67]:

1. Survey: It is used as a review of the object, e.g. tool, method, model, etc. after it has been used for a while. Based on interviews of a sample of representative people out of the population to be studied, researchers try to find a generalized model to describe it.
2. Case study: The goal of case studies, which can be considered as observational studies, is the collection of data and analysing them in statistical ways.
3. Experiments are carried out using controlled studies in a laboratory environment providing a high level of control. The objective is to manipulate one or more variables and control all other variables at fixed levels, measuring and analysing their influences using statistical methods.

Table 5.1-1 shows a comparison of these approaches concerning several important factors

Factor	Survey	Case-Study	Experiment
Execution control	No	No	Yes
Measurement control	No	Yes	Yes
Investigation cost	Low	Medium	High
Ease of replication	High	Low	High

Table 5.1-1: Comparison of empirical strategies according to Wohlin et al. [67]

Regarding the *focus of control* during investigation experiments allows control of the researcher only. The other methods will be influenced by several control mechanisms, such as management of a company because of economical reasons, etc. *Measurement control* concerns factors, which can be included during execution of the investigation process. Using the survey approach it is not possible, the main proceeding is limited to data collection of the people's interview. *Investigation* cost is a major aspect for choosing a strategy. It is related to (i) the size of investigation and (ii) the need for resources. Case Studies aims to observe a

product, which may be retailed afterwards while experiments are a kind of artificial investigations where no product but some form of experience will be achieved. *Replication*, either of design or results is possible using surveys and experiments.

5.2 Quality Improvement Paradigm

One reason establishing empirical research in context with software engineering is the idea of continuous improvement of process, product and models.

Wohlin et al. describe Basili's *Quality Improvement Paradigm* (QIP), which is very similar to Deming's Plan-Do-Check-Act (PDCA-cycle) (see chapter 3 for details) concerning 6 steps to meet this goal using an empirical approach in software engineering.

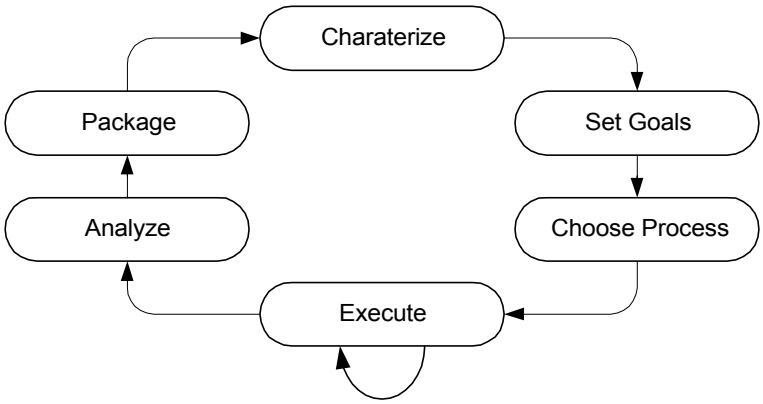


Figure 5.2-1: Quality Improvement Paradigm [67]

The main steps of the Quality Improvement Paradigm includes:

- *Characterization*: Understanding of the environment (models, data, etc.) and establishing baselines according to business processes.
- *Definition of goals*: Fix quantifiable goals for the project, organization performance and improvement according to the initial characterization.
- *Choose Process*: Selection of a proper process according to previous definitions. The process must be consistent with the process.
- *Execution*: Performing product development and providing project feedback according to the goals.
- *Analysis:* Once the project has finished, data and information collected during developing the product must be analysed to evaluate current practice, determine problems and recommendations for improvement of future projects.
- *Packaging:* Consolidation of experience and information achieved during product development as well as integrating this knowledge with respect to prior projects.

This model provides two forms of feedback: a *project cycle* (control cycle) which is established in the stage of execution in order to prevent and solve problems at task and project level using control loops. The second a *corporate feedback cycle* (capitalization cycle) performs feedback to the organization. The benefit is twofold: on the one hand side the organization achieves information about the current project and its proceeding, on the other hand side, accumulated information using prior project information can be reused in future projects.

5.3 Experiment process approaches

As described in the previous section and according to the process-oriented view of either software engineering or quality management and method, also for experimentation exist a process to guarantee a controlled proceeding or the experiment. A more general process for performing experiments is discussed in this section [67].

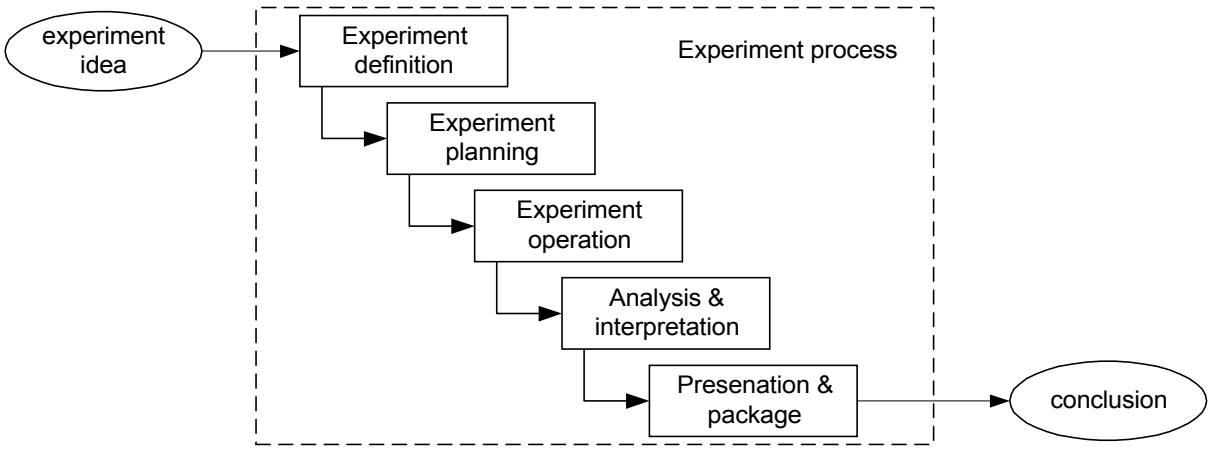


Figure 5.3-1: Overview of experiment process [67]

The process model shown in Figure 5.3-1 doesn't represent a straightforward waterfall model but there may be some reasons to go back to previous stages. Once executing the experiment it is not possible to return to the definition stage to redesign the experiment because modified goals will influence several tasks in the execution stage.

Experiment definition: In this stage of experimentation proceeding there must exist an idea of hypothesis, which should be accepted or rejected during and after the experimental run as well as in the analysis stage. Also goals and objectives must be clear at this stage. It must be clear, which entity (*object of study*) is studied in the experiment, e.g. products, processes, models, etc and what's the intention (*purpose*) to start the experiment at all, e.g. comparison of two methods, etc. Additional it must be defined, which environment (*context*) will be used and how the results will be interpreted (*perspective*) afterwards.

Experiment planning: At this stage of the experimental process the context of the experiment itself is determined in detail, including personnel and environmental approaches, e.g. students at a university course. All hypotheses must be defined formally including independent variables and dependent variables as well. The difference between these kinds of variables is the manipulation level, independent variables are basic variables defined at definition stage while dependent variables can be achieved afterwards using independent variables. Furthermore an experimentation design must be chosen, that is how the tests are organized and run. According to the experimentation design there must exist a suitable instrumentation, e.g. guidelines, measurement instruments, etc. Also in the planning phase, validity evaluation must be performed. Wohlin et al. distinguish *internal* (validity within the environment according to the results), *external* (generalization findings), *construct* (theory reflects observation) and *conclusion* (relationship between treatment[9] and outcome) validity [67].

Experiment operation: Once the planning phase has finished, the experiment will be performed. According to Wohlin et al., three sub processes must be realized, *preparation*, *execution* and *data validation*. The preparation includes activities before the experiment starts, providing material, instruction participants, etc. The execution part must be realized as defined in the planning phase, including data collection. And finally all collected data must be correct and have to show a valid picture of the experiment.

Analysis and interpretation: Based on valid data, investigations of the data (or probably parts of it) are performed using different statistical methods and tools. Depending on these statistical results, interpretations can take place to accept or reject hypothesis.

Presentation and Package: The last step in the experimentation process model is the presentation of the results, which includes results and experiment description as well.

5.4 Evaluating Inspections using Experiments

Concerning inspection as a quality management tool with the intention to reduce defects in early stages of software development, an experiment to investigate factors like usability, efficiency, efficiency, etc is set up by a team at research group for Industrial Software Engineering at Vienna University of Technology.

The following chapter describes a part of the experimental process with focus the data analysis phase with special respect to

- Defect detection and inspector feedback
- Team structure
- Economic aspects concerning inspection

[9] A treatment describes one particular value of a factor (one or more independent variables).

5.5 Chapter Summary

This chapter presented a brief overview of software engineering experiments including some strategic approaches. Reasons to perform empirical research in the area of software engineering is the idea of continuous improvement of products and processes or knowledge collection and evaluation of methods, models and tools.

This general approach of an experiment proceeding leads to the “inspection experiment” carried out at the ifs-institute. This proceeding consists of experiment design, execution, reworking, analysis and finally presentation of the results.

6 Investigation of Inspection using a Controlled Experiment

Project and quality managers are responsible for the product and the proceedings of the project and its success. Biffl points out the difference between an *economic* and a *technical* project manager and a *quality* manager at project organisation level [11]. But as common features of these roles, they both have to solve problems concerning cost, time, results and performance.

Customers and company management want a low price solution but still providing a very high performance. The project-team needs resources (infrastructure, time, etc.) to achieve proper results and other project teams are also trying to get resources as well (resources are very rare all the time). Project managers have to find the best possible solution within the whole company. Because the technical project manager is very close to the technical software engineering process, and he must also consider economical aspects, he might be the most important person in a company. Quality managers provide project-support concerning quality-relevant aspects. He is responsible for the "quality" of the final product and – according to this fact – for the success of the project (see chapter 2.1).

6.1 Research Questions

As outlined in chapter 3.2.3 one possibility to improve the software product and support project proceedings is software inspection, which must be examined at early stages of product development. This thesis might provide project- and quality managers some assistance with inspection and reaching the decision to establish inspections in the project or company.

a) Investigation of inspector qualification: I will focus on different qualification rating models and real-team composition in section.

b) Defect detection rates: What kind of defects and how many defects (at various severity levels) will be detected using software inspections, either by individuals?

c) Temporal proceeding of defect-finding: According to different reading technique approaches as declared in the feedback-questionnaire by individual inspectors, the defect detection proceedings must agree to this classification. I will select various inspectors (with respect to defined criteria) and show their time-proceeding and their total effectiveness.

d) Feedback How easy it is to establish inspections according to participants' estimation of acceptance and usability of the method, etc.?

6.1.1 Investigation of Inspector Qualification

Once a project- or quality manager wants to perform an inspection within a project for the first time, he has to form an inspection team. Because there has been no experience so far he has to decide which people must participate to guarantee best results.

Without additional information, like practical experience or theoretical knowledge of the participants, the simplest way to get information about SE- and inspection skills will be a qualification-questionnaire before getting started.

In order to investigate different qualification ratings, we used data, collected at the experiment, building up three different models concerning

- Supervisor qualification assessment: This rating has been established as a pre-condition for the software-engineering workshop by lab-assistants. The inspectors had to examine a small software-program, with respect to their software-development skills (*SD-Skills*), to practice SE techniques and to get an impression or further proceedings of the workshop.
- Self-assessment: Individual feedback-questionnaires concerning inspector experience (*Exp*) are the basic material for the second qualification rating. These questionnaires concern questions according to general skills, software-engineering skills and inspection skills. Single results have been aggregated to an overall experience for each individual inspector. This proceeding might be a realistic starting-point for the choice of a project manager.
- Pre-Test: Performing some kind of “mini-inspection” (small sample specification and a 2 hour overall inspection time) the number of defects detected is a useful method to find out inspector qualification according to their inspection skills. The aggregation of matched defects provides a better view on inspector capabilities. I summarize classified number of all, minor and major matched defects (SI-DD).

I describe a more detailed view on each model including investigation results in chapter 7. According to inspector qualification, acquired during supervisor qualification assessment, inspection teams (so-called *real teams*) have been set up. The second approach of inspection team composition is based on so-called *virtual teams*. The team-size has been limited to 4-6 team members.

Research Questions:

Q1.1: Identification of a model for qualification evaluation according to experience and feedback questionnaires (*self-estimation*, *Exp*). I expect a realistic view on inspector qualification because of an extensive questionnaire and a defined application context.

Q1.2: Equivalent to Q1.1 but using an inspection pre-test (*SI-DD*) as data sources. I expect the most realistic method for qualification assessment because the focus of the pre-test covers an equal application area and requires inspection as well as software engineering skills.

Q1.3: Comparison of qualification ratings according to *SD-Skills*, *Exp* and *SI-DD* and selection of the most realistic qualification evaluation method in practical environment

for further investigations. In general, I expect an increasing number of level C inspectors because the evaluation topics cover not only SE-skills (e.g. programming skills) but also project and inspection experience. The inspectors are more familiar to SE-Skills but to project management skills.

6.1.2 Defect Detection Rates

One of the main goals of inspection is the reduction of defects in the software artefact, in our case a software specification for a large-scale software product.

This section gives an overview of defect detection rates, using data collected at our experiment after the 1st inspection cycle, concerning defect severity classes and the location of the individual defects with respect to inspector qualification and reading technique roles. To see the advantage of software-inspection I will focus on the *effectiveness* of inspections according to *qualification levels* and *reading-technique roles* concerning *document location* and *defect severity*.

1. Inspector Qualification: As described in the previous chapter and chapter 7, I will use one out of three different types of inspector qualification, concerning *supervisor qualification assessment, self assessment* and a qualification rating according to a *pre-test* (i.e. mini-inspection).
2. Defect severity: The defects have been classified into four initial severity levels, 0 (*trivial*), 1 (*minor*), 2 (*major*) and 3 (*critical*). In this text I will summarise level 0 and 1 to minor defects and severity level 2 and 3 to major defects. Chapter 6.2.1.2 shows the distribution of seeded defects with respect to defect severity.
3. Reading technique / reading technique roles: One specified reading-technique-role has been assigned to individual inspectors, who form an inspection team with 4-6 inspectors. The usage of reading techniques has been described in section 6.2.2.
4. Document location: Defects at various severity levels have been seeded in the inspection document in four different locations: *introduction, business functions, object oriented class models* and *class descriptions*. Section 6.2.1.2 presents the distribution of seeded defects with respect to document location.

Effectiveness is defined as the total number of defects found in a predefined defect class in relation to all seeded defects in the same defect class.

$$Effectiveness_{defect\_class} = \frac{\sum found\_defects_{defect\_class}}{\sum all\_defects_{defect\_class}}$$

Research Questions:

Q2.1: Investigation of defect detection rates according to one selected qualification-rating model with respect to defect severity classes. Usually higher qualified inspectors will find more defects in comparison to lower qualified inspectors.

Concerning reading technique roles, I assume, that CBR readers will find more defects with respect to every qualification level.

Q2.2: Equivalent to Q2.1 with focus on reading technique and roles. Because of the internal structure of our reading technique methods, each individual approach focus on defined types of defects, locations, etc. Because CBR reading techniques cover the whole document per default, therefore CBR inspectors might find most defects.

Q2.3: Equivalent to Q2.1 with focus on document location. Reading techniques focus on predefined document locations. Therefore I expect a clear representation of reading technique roles concerning the defect detection rate.

Further investigations on defect detection with focus on combined attributes including different team defect finding results will be out of range of this thesis.

6.1.3 Temporal behaviour of defect-finding according to reading-technique

During the proceeding of inspection we provide a task-description for the usage of the individual reading techniques. Appendix C shows the task-descriptions used during the experiment. Inspectors have to give feedback to their individual reading approach.

0) "I read *document at once*, thought about defects and noted them. Afterwards, I reviewed the checklist / scenario instructions for completeness of defect classes."
1) "I picked *each question* from the checklist / scenario instructions, and inspected the whole document according to this topic."
2) "I read the *whole* checklist / scenario instruction and remembered most of them. Afterwards I inspected the document as a whole. "
3) Any other reading technique approach has to be defined.

Independent of this reading technique approach inspectors search for defects for a defined time (with the lower limit of about 2h and an upper limit of about 6h). Within this range, the inspectors can finish their work, if all defects have been found (subjective estimations).

Now it is quite interesting to observe the defect detection rate in relationship to the overall time of inspection and the average time between two defects detected. Obviously I have to distinguish between noted and matched defects and different inspector qualification levels.

Research Questions:

Q3.1: Investigation of inspection duration according to reading-technique roles and inspector qualification. Because of the focus of reading technique approaches, which focus to limited documentation locations and they have not to cover the whole document, the overall inspection duration will be shorter using SBR-reading techniques. Concerning inspector qualification, I expect shorter inspection duration for higher-qualified inspectors.

Q3.1: Investigation of inspection pre-work, i.e. the distance between start of inspection and the first defect detection. Concerning the SBR reading technique, inspectors have to finish pre-work, e.g. creating models, before starting the defect detection process. Following those instructions, SBR inspectors will start later and, therefore, the duration until the first defect match will be later as well. I expect a clear representation in the data collected during inspection.

Q3.3: Average minutes between two defects found during inspection in a linear way, ignoring task-specific delay (concerning matched and noted defects). Because of the scope of reading techniques, SBR inspectors will take longer to find defects (and matched defects) in contrast to CBR inspectors. Additional SBR techniques focus on a more concrete model (they have to construct it) while CBR inspectors use checklists during the reading process. Therefore I expect a shorter duration (in mean) between two defects using CBR reading technique approaches. Furthermore higher-qualified inspectors will find more defects in a shorter duration than lower qualified inspectors.

6.1.4 Usage and Acceptance of inspections

Inspectors have to answer feedback questionnaires after each inspection activity. One of these questions is the personal reading approach (as described in the previous section). The answers are *not anonymous* because they must allow investigation of the context between feedback-results and defect detection rate, etc. The complete questionnaires are described in appendix B

Research Questions:

A project or quality manager is interested in the applicability and usability of inspection and reading-techniques at the starting point of establishing inspection processes in a practical environment. Following this approach, I will investigate *applicability* (FB01) in the following section.

Q4.1: Applicability of reading techniques according to inspector reading technique and inspector qualification. Because lower qualified inspectors will use the checklist more frequently (as guidelines), I expect a higher acceptance by lower qualified inspectors. Concerning reading techniques, I expect a greater acceptance at SBR reading techniques because the method is more clearly arranged and focus on specific parts only.

6.2 Experiment description

This section describes the experiment environment for the experiment conducted at *Vienna University of Technology* in 2000/2001. It introduces to the software artefacts, inspection participants, processes and volume of data and dependent as well as independent variables in context with the experiment and special focus on qualification, defect detection (individual and team) and feedback questionnaire. The experiment description can also be found in publications, e.g. at [11] and [15].

6.2.1 Software artefacts

6.2.1.1 Requirements document

The key document is a *requirements document* (specification document) with seeded defects out of a well-known application area, in our case a distributed administrative information system for managing ticket sales, administration and location management. The system consists of four basic parts, as shown in Figure 6.2-1. (i) *Context information* in natural language containing text and illustrating diagrams, (ii) *business functions* and non-functional requirements, an (iii) *object oriented class* model in UML notation and (iv) a *class description*.

The inspection document in the experiment contains about 47 pages with about 13.000 words, 16 diagrams and 97 seeded defects. Biffl provides an estimation of development cost at about 5000 person-hours (after performing an inspection) within about 5 months and 4-6 developers involved [15].

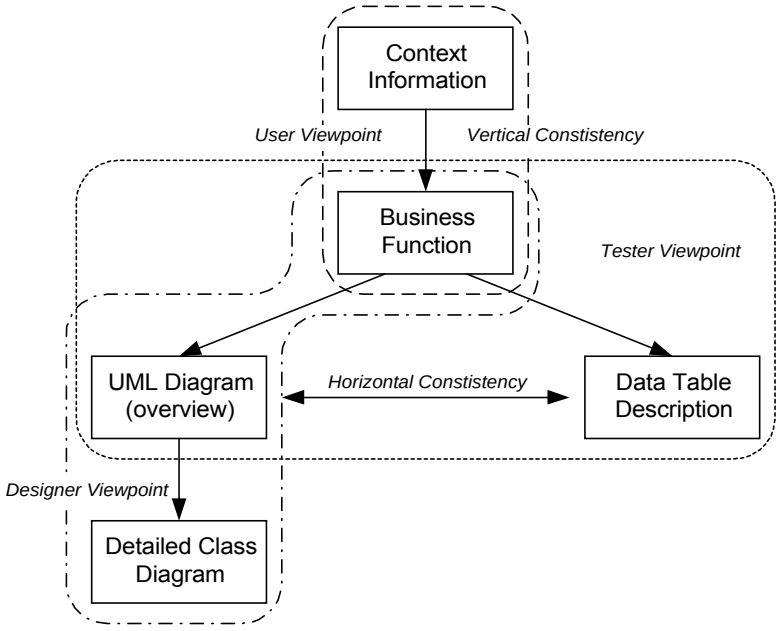


Figure 6.2-1: Related parts in the requirement document according to Biffl [15]

Figure 6.2-1 also shows different viewpoints (user, designer and tester), which represent the focus of the scenario-based reading techniques. I provide an overview of reading-techniques in context to the experiment in chapter 6.2.2.1. Inspectors use these reading-techniques to find appropriate defect classes [15].

6.2.1.2 Reference defects (seeded defects)

The software artefact, the specification, has been seeded with 97 artificial defects, which might occur in software engineering practices. The defects were seeded with respect to document location (introduction (INTRO), business functions (BUSI), objects (OBJ) and data description (DATA)) and defect type (unclear, missing, wrong and inconsistent information), each equipped with one severity level out of trivial (SEV0), minor (SEV1), major (SEV2) and critical (SEV3) defects).

Defect classification has been introduced by Fagan, which is described at [23]. He differs *minor* defects (severity level 0 at [15]), *major* defects (severity level 1 and 2 at [15]) and *super-major* (severity level 3 at [15]) defects.

The inspection preparation team (EPT) seeded 97 defects considering all defect classification areas across the specification document. The following figures give a rough overview about the basic specification document used for inspection, according to the reference-defects.

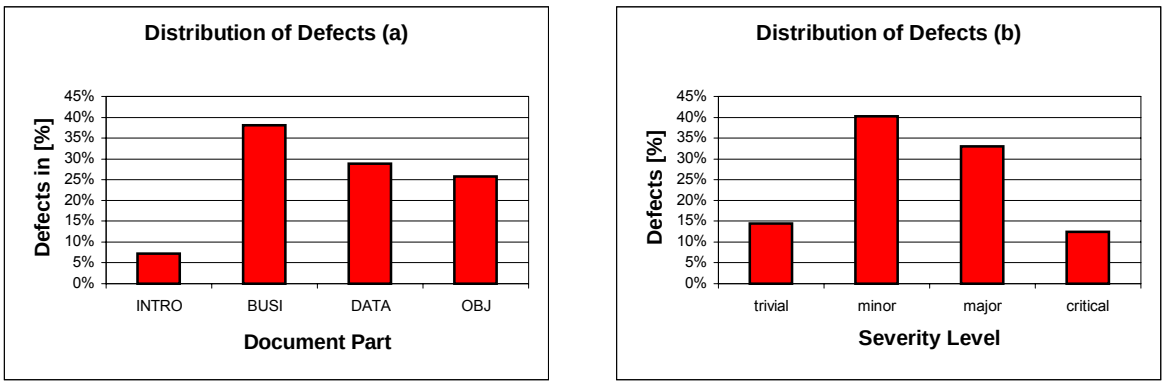


Figure 6.2-2, Distribution of seeded defects according to document location (a) and severity (b)

Figure 6.2-2a presents the distribution of seeded defect in different document locations, the introduction, which contains some information about the project in general, the application area and the products goals. The introduction is seeded with 7 defects (7,2 % of all defects) – in relation to the technical description in BUSI (37 defects; 38,1%), OBJ (25 defects, 25,8%) and DATA (28 defects, 28,9%).

Figure 6.2-2b shows the distribution of seeded defects according to the corresponding severity level. 14 defects (14,4%) has been classified as trivial, 39 defects (40,2%) as minor, 32

defects (33%) as major and 12 defects (12,4%) as critical. For further investigations in this text, I use 2 severity levels only, minor and major. *Minor* includes trivial and minor defects and *major* defects, which include major and critical defects. According to this restriction, I achieve 53 minor defects (54,6%) and 44 major defects (45,4%).

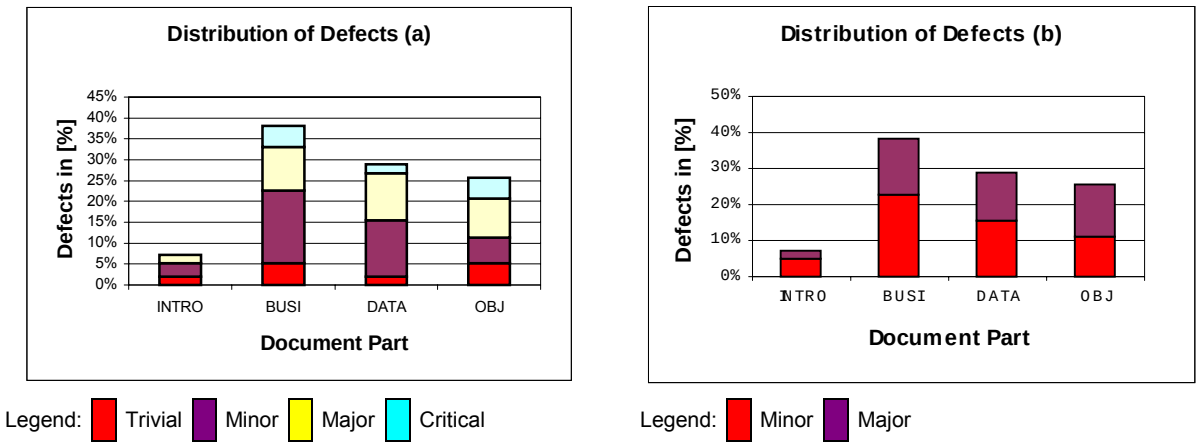


Figure 6.2-3, Distribution of seeded defects according to document location and (a) initial severity level, (b) evaluation severity level

	Defect distribution according to initial severity level (a)										compressed severity level (b)					
	trivial	[%]	minor	[%]	major	[%]	critical	[%]	sum	[%]	minor	[%]	major	[%]	sum	[%]
INTRO	2	2,1%	3	3,1%	2	2,1%	0	0,0%	7	7,2%	5	5,2%	2	2,1%	7	7,2%
BUSI	5	5,2%	17	17,5%	10	10,3%	5	5,2%	37	38,1%	22	22,7%	15	15,5%	37	38,1%
DATA	2	2,1%	13	13,4%	11	11,3%	2	2,1%	28	28,9%	15	15,5%	13	13,4%	28	28,9%
OBJ	5	5,2%	6	6,2%	9	9,3%	5	5,2%	25	25,8%	11	11,3%	14	14,4%	25	25,8%
SUM	14	14,4%	39	40,2%	32	33,0%	12	12,4%	97	100,0%	53	54,6%	44	45,4%	97	100,0%

Table 6.2-1: Distribution of seeded Defects

Figure 6.2-3 and Table 6.2-1 shows the distribution of the initial four severity levels in (a) and the compressed severity levels (b) according to document locations. Reading techniques and their roles focus on different document locations and should support defect detection.

6.2.2 Defect Detection Techniques

6.2.2.1 Guidelines for defect detection

Defect detection is the main reason for introducing a software inspection. Reading techniques support the defect finding process by assigning individual roles to the inspectors.

In our experiment we use checklist based reading (CBR) and scenario-based reading (SBR). SBR reading techniques are structured in *designer*, *user* and *tester* point of view. Each role is documented using checklists and task-descriptions (see appendix C for details).

1. Task description: Reading technique CBR (appendix C.1)
   CBR uses a straightforward proceeding concerning all document parts sequentially each equipped with a range of questions according to the document part. Therefore we introduce 5 blocks of questions, concerning 4 document parts and one all-embracing chapter concerning general questions.
2. *Task description: Reading technique SBR-Designer* (appendix C.2)
   The focus of SBR-D reading techniques covers *business functions* in order to achieve a matrix of objects, *UML diagrams* and *data tables*. The matrix of objects shows all objects and the relation to each other and it is generated at the beginning as a preliminary work (without finding defects). Based on this matrix the UML model and Data Tables are checked for defects.
3. *Task description: Reading technique SBR-User* (appendix C.3)
   SBR-U focus on USE-case and therefore on context information and business-functions. In the preparation phase the inspectors generate a flowchart representing the proceeding of tasks according to the business-functions. Based on these charts use-cases and actor-descriptions will be analyzed with respect to defects.
4. *Task description: Reading technique SBR-Tester* (appendix C.4)
   SBR-T focuses on test cases in general. Based on business-functions the inspectors generate a table of objects consisting of attributes and relating test-scenarios and test cases. Based on the preliminary preparation phase, UML diagram and Data Table description is checked for defects.

All reading techniques are applied to the requirements document to support defect detection. Once finding a defect it is documented in a defect detection logbook.

6.2.2.2 Defect Detection Logbook

All defects found by inspectors using different roles (according to reading technique approaches), at different sessions (tutorial, inspection, re-inspection) are documented in a logbook and electronically in our database. The logbook contains (i) defect specific (e.g. defect description, estimation of severity, etc.), (ii) location specific (e.g. defect location, etc.) and (iii) time-specific (e.g. found at time, start time of session and tasks, etc.) information.

6.2.3 Feedback Questionnaire

Feedback reports have an important role in subject-based experiments. It can be used to verify the quantitative data, to provide insights into subject's performance [52]. In our experiment we use additional information, which is provided by inspectors before inspection (experience) and after each inspection cycle (see appendix B for details).

1. *Experience questionnaire* (appendix B.1)
   The experience survey has been performed before proceeding the inspection process in order to get information about inspector's capability (inspectors view). The questionnaire covers topics like motivation, experience in software development and aspects in relation to the inspection processes (e.g. experience in reading requirements documents, etc.). It contains the basic information for qualification assessment "*Exp*".

2. *Reading Technique Questionnaire (Feedback)*:
   After inspecting the specification another questionnaire (according to the reading technique) has been established to get information about the reading technique itself (e.g. usage and usability), time-management (e.g. document coverage, time requirements, etc.), defect detection and estimation of remaining defects after inspection. The feedback-forms according to each reading-technique are closely resembled to each other to enable comparability between Scenario Based Reading (appendix B.3) and Checklist Based Reading (appendix B.2).

   a) Usability of the reading-technique and its instruction: This section covers *applicability* (FB01), *usability* (FB02) of the reading-technique and *personal approach of reading* (FB03)

   b) Inspection artefact: The *clarity of the inspection artefact* is collected with question FB05.

   c) Time-management during the inspection process and document coverage: This block of questions defines the percentage of document inspected per document part (FB06-FB09) and – if necessary – additional inspection time for reading (FB04).

   d) Number of defects detected during the inspection process: The inspector notes their individual estimation about their defects found during the inspection process (FB10-FB17) according to document locations and severity levels.

   e) Estimation of remaining defects: The last section of questions concerns subjective estimations of inspectors according to the remaining defects in the specification document with respect to document location and severity levels (FB18-FB41).

6.2.4 Inspection participants

The planning, execution and evaluation of large-scale experiments require a set of roles and responsibilities. Biffl proposes the following roles in the experiment [15].

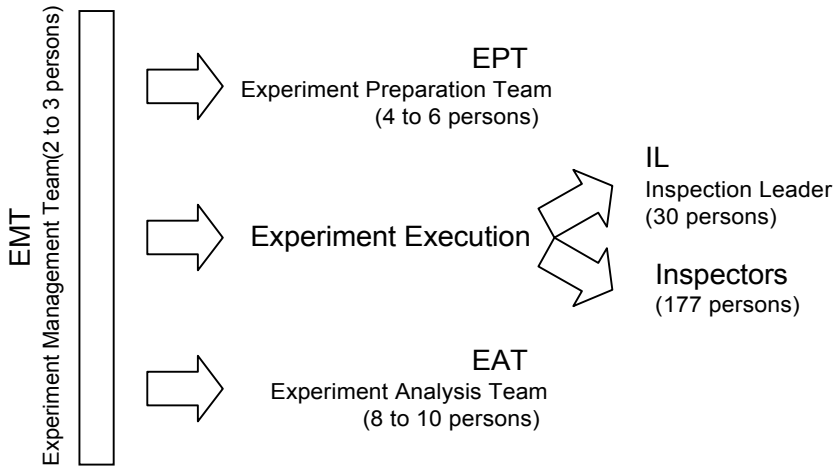


Figure 6.2-4: Inspection participants

- *Experiment Management Team – EMT*: Overall planning of the experiment.
- *Experiment Preparation Team – EPM*: Preparation of inspection and supporting documents. Development of experiment environment including data collection tool.
- *Experiment Analysis Team – EAT*: Preparation and Execution of the analysis of collected data including consistency checks and statistical analysis.
- *Inspection Execution – Inspection Leader (IL) and Inspectors*: The inspection leaders are assigned to the inspection team and helps them to execute inspection and control their performance. In our experiment 30 IL support 30 teams with 177 inspectors in summary. I will provide a closer look to team-composition at 6.2.4.3 and to inspectors at 6.2.4.2.

6.2.4.1 Inspectors and their assigned Reading Technique

According to the defect detection guidelines, different reading techniques have been developed to focus on different defect locations, types, etc. In our experiment we used *checklist based reading (CBR)* and *scenario based reading (SBR)*. As sub-topics of SBR we used different point of view on the specification document, i.e. user (SBR-U), developer (SBR-D) and tester (SBR-T).

In summary we got 4 different reading techniques, which has been assigned to the inspectors in a uniform way, i.e. in our large-scale experiment 177 participants (in the role as inspectors) has been assigned to reading technique roles according to their qualification and perform the inspection process as in a team structure.

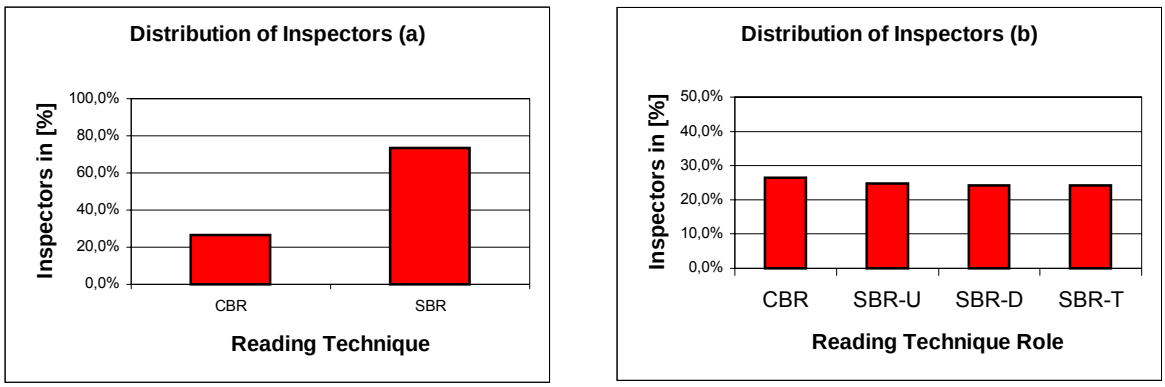


Figure 6.2-5, Distribution of inspectors according to
(a) reading technique, (b) reading technique roles

	reading techniques (a)			reading technique roles (b)				
	CBR	SBR	Sum	CBR	SBR-U	SBR-D	SBR-T	Sum
# of inspectors	47	130	177	47	44	43	43	177
[%]	26,6 %	73,4 %	100 %	16,6 %	24,9 %	24,3 %	24,3 %	100 %

Table 6.2-2: Distribution of reading technique (roles)

Figure 6.2-5 (a) and (b) and Table 6.2-2 show the assignment of inspectors to reading technique roles. It is easy to see the uniform distribution of inspectors with respect to the reading technique roles and, because of the granularity of SBR reading approaches, about 75% of all inspectors are at SBR and only 25% at CBR. Further investigations concerning individual inspectors will focus on reading technique roles.

6.2.4.2 Inspector

As described in the previous section, all inspectors has initially assigned to reading technique and reading technique roles by the EMT in an almost uniform way. Another important question concerns the qualification of inspectors, who participate an inspection.

To investigate individual qualification in relation to their defect findings and the reading technique the qualification has been assigned according to *SD-Skills* by the EMT. The rating of SD-Skills is based on a small programming example concerning software engineering methods. The inspectors have to implement a small database application program to learn the programming language, usage of user interfaces and database handling as well as

documentation of the product. During a verbal delivery session together with EMT the product has been rated to achieve qualification levels.

The distribution of inspectors to reading technique roles and the predefined qualification is shown in Figure 6.2-6 and Table 6.2-3.

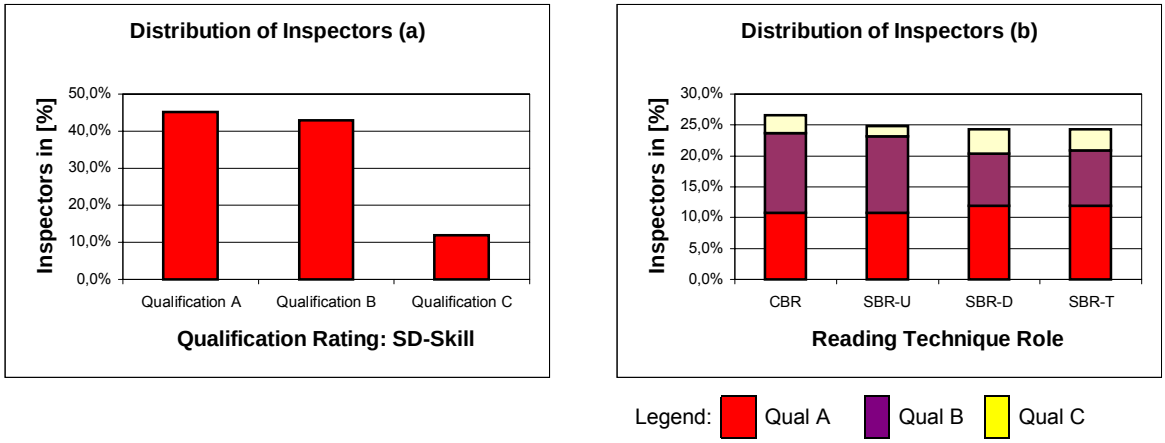


Figure 6.2-6, Distribution of inspectors according to
(a) initial qualification, (b) reading technique roles

According to SD-Skills most of the inspectors had been assigned to qualification level A and B (each >40%), only 12% of all inspectors has been rated level C. All inspectors have been assigned to all 4 reading technique roles in a uniform way. The Distribution of reading technique roles with respect to initial qualification level is presented in Figure 6.2-6 (b).

	Inspector distribution according to reading technique role (a)								compressed reading technique (b)					
	CBR	[%]	SBR-U	[%]	SBR-D	[%]	SBR-T	[%]	CBR	[%]	SBR	[%]	sum	[%]
Qualification A	19	10,7%	19	10,7%	21	11,9%	21	11,9%	19	10,7%	61	34,5%	80	45,2%
Qualification B	23	13,0%	22	12,4%	15	8,5%	16	9,0%	23	13,0%	53	29,9%	76	42,9%
Qualification C	5	2,8%	3	1,7%	7	4,0%	6	3,4%	5	2,8%	16	9,0%	21	11,9%
SUM	47	26,6%	44	24,9%	43	24,3%	43	24,3%	47	26,6%	130	73,4%	177	100,0%

Table 6.2-3: Distribution of Inspectors according to initial qualification

Because it might be useful to base qualification related data on other point of views. We set up an experience questionnaire (see chapter 4.4), which provides subjective information of inspector, how they estimate their skills themselves.

The most suitable and realistic consideration of inspector qualification according to software inspections can be investigated according to the pre-test and their defect detection rating. The purpose of this pre-test is the introduction of all participants within a real inspection environment. These qualification rating has been built by a model described in 6.1.1 and 7 in more detail.

6.2.4.3 Team composition

All 177 inspectors have been assigned to inspection teams, who perform the inspection process according to the reading technique roles. Because of most efficient teams the team-size varies from 4 to 6 participants (about 93% of the teams consist of 6 team members) shows. The assignment of the inspectors has been established randomly but with respect to comparability of teams according to reading-technique roles and qualification.

After team composition, which was executed randomly, but with respect to inspector qualification by EMT every team has been assigned to one inspection leader. The inspection Leader helps the teams to perform inspection.

6.3 Experiment operation

The experiment has been performed in the year 2000 as a part of a software engineering workshop within a period of about 3 weeks. In every week a special part of the experiment is done: The first week has been used as training sessions (*pre-test*) for the inspectors to get familiar with the area of software engineering and inspections. This preparation phase consists of a theoretical introduction, performed by the inspection leaders and a practical exercise for evaluation including a feedback self-estimation questionnaire according inspector skills. Results from this practical evaluation and the feedback questionnaires are used for estimation of inspector qualification (see chapter 6.1.1 and 7).

During the following two weeks inspection (1st cycle) and re-inspection (2nd cycle) are performed. A schematic overview about the operational part of the experiment can be found at Figure 6.3-1. The proceeding of one inspection cycle is described in Figure 6.3-2.

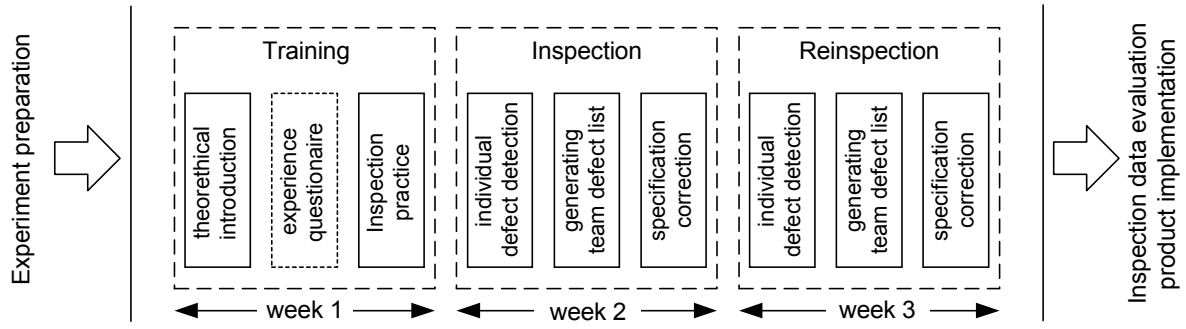


Figure 6.3-1: Experiment proceeding adapted to Biffl [15]

The defect seeded requirements document is the inspection object for inspection (cycle 1). After searching for defects and their documentation by individual inspectors, a team defect list has been generated automatically based on the individual defect lists (according to each inspection team). All defects, listed in the team defect list has been corrected by the

inspection leader and represent the input requirements document for the second inspection cycle (re-inspection).

After the second inspection cycle (including generation of the team defect list and correction of the requirements document) the inspection teams implement the system and the evaluation of inspection data has been executed by the EAT.

Figure 6.3-1 shows the proceeding of inspection, re-inspection and presents the corresponding data-flows.

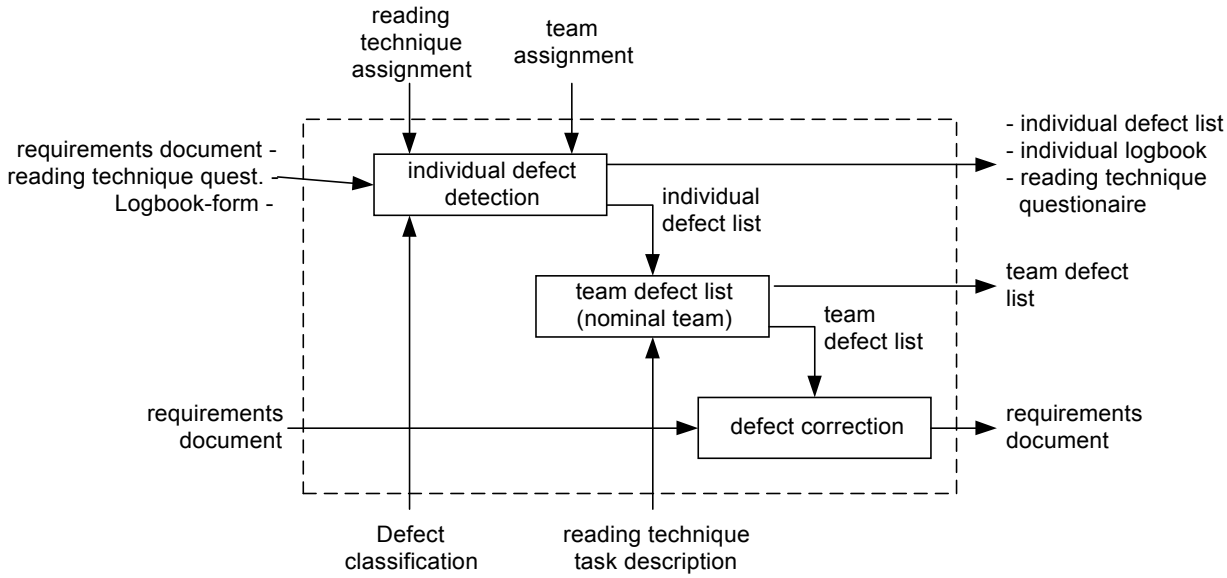


Figure 6.3-2: Inspection process – sequence of events adapted to Biffl [15]

The inspection process in our case consists of three main stages:

- Individual defect detection: The inspection teams are formed with respect to qualification (SD-Skills) and reading technique assignment. The individual inspectors use a table for defect classification as well as a task description according to the reading technique.
  The inspection object is the specification with 97 seeded defects, prepared by the EPT. All found defects must be documented using a logbook form (for task, time and defect relational information). Once the individual defect detection phase is finished, the inspectors must provide feedback to their reading technique. Both the defect list and the feedback reports must be registered at our MySQL database using a Web-Front-End.
- Team defect list: In our experiment we do not establish a team meeting. To achieve team defects, all individual defect lists are merged team wide to a team defect list considering overlap of matched defects. If two inspectors found identical defects (an

overlap occur), it will appear only once in the team defect list. This task is done automatically via tool support.

- Defect correction: Based on the team defect list the inspection leaders modify the requirements document with respect to the matched defects. This document will be the input for the 2nd inspection cycle and, after processing re-inspection, the basic document for implementation phase.

On further proceeding to the software engineering workshop this second (corrected) version of the specification has to be implemented according to the life-cycle model.

The operational part of the inspection experiment starts at the training session of inspection and stops at completion of data collection at the end of the second inspection cycle.

6.4 Data analysis

Defect detection data as well as feedback results are registered in a MySQL database by the individual inspectors using a Web-Interface for easy handling. Although the input form provided by EPT considering basic integrity rules as well as using select boxes for predefined values (and very less text-boxes for comments) to reduce input defects, the inspection leader must perform checks to improve data quality (correctness and completeness).

I will provide the basic database structure, which will be relevant for data analysis, and some interesting facts about volume of inspection data in the following sections.

6.4.1 Database structures

Figure 6.4-1 shows the basic database structure, which is used for evaluation and inspection data analysis phase. Organizational overhead, which is necessary for management of the course and additional internal information, will not be covered in the text because there is no context with the inspection experiment and this text.

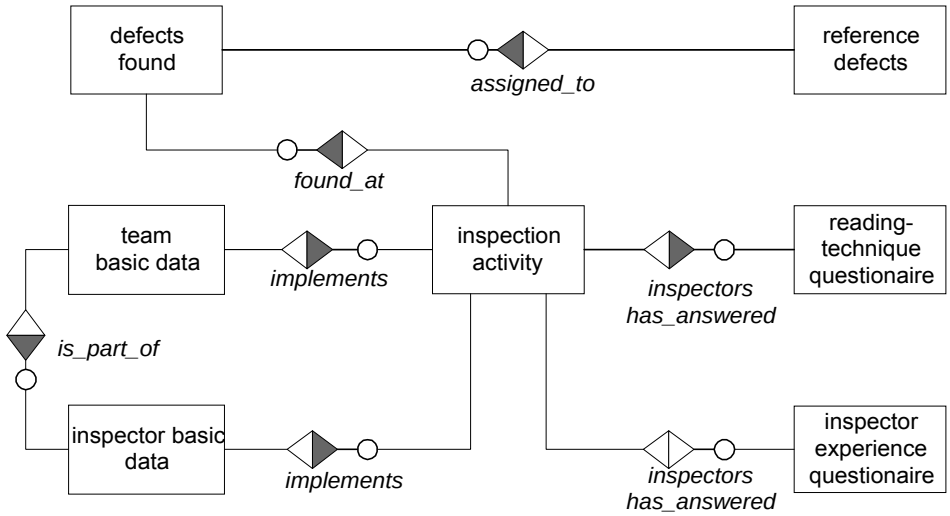


Figure 6.4-1: Database model for inspection experiment evaluation (simplified)

One inspection team consists of 4-6 inspectors who execute the whole inspection-process, team members do not change during the process. In every session, i.e. inspection activity, defects are detected by the individuals and are matched to reference defects automatically. Additional information is provided at different session stages, i.e. experience questionnaire at the beginning of the training session and two reading-technique questionnaires after both inspection cycles.

This context has been modelled in the basic database model using seven major entities related to each other.

a) Inspector basic data: contains inspector relevant information like personal information, qualification, reading technique role, etc.

b) Team basic data: contains team relevant information like team identifier, team reading technique, team size, etc.

c) Inspection activity: provides data relations to training session, inspection and re-inspection, etc.

d) Defects found: describes all defects (matched or noted) during all session by all inspectors.

e) Reference defects: defines the set of seeded defects

f) Reading technique questionnaire: questionnaire about reading technique usage and usability (performed 2 times coupled with activity “inspection” and “re-inspection”)

g) Inspection experience questionnaire: questionnaire about inspectors estimation about own skills (coupled with activity “training”)

6.4.2 Volume of data

Performing large-scale experiments, like it is done in our case, will result in a very large volume of data, which cost time (e.g. preparation, inspection time) and money. Biffl records for a similar experiment performed a year ago 530 staff hours by EMT (management team), 470 by EPT (preparation team) and 420 by EAT (analysis team) [15].

About 2000 staff hours by individual inspectors have been recorded in our database for inspection reading at all activities. This doesn’t include registering of data via Web-Interface, so it will be a very low estimation about this cost factors.

The 47 pages specification document contains about 13000 word and 16 diagrams. It has been seeded with 97 artificial defects including 53 minor and 44 major defects.

177 inspectors perform in summary 3 sessions (pre-test, inspection and re-inspection each with one questionnaire) and form 30 real teams. During inspection (which is the main part of this thesis) they found 6198 defects, including 2288 matched defects (according to our seeded defects). 2288 matched defects include 1540 major defects.

Because of the high volume of data and a lot of possibilities for combination of variables there must exist a structured approach for evaluation of the specific tasks. Our EAT developed several models, which aim to be comfortable for different needs because of a modular structure and a step-by-step proceeding during the investigation and evaluation-process. I will describe this structure in the following chapters.

6.5 Data Evaluation Process

After performing the experiment, we (the EAT) developed a proceeding for evaluation and easy reuse of data, accumulated data and a basic structure for model evaluation.

6.5.1 Data analysis process used for this thesis

The data analysis process is set up in modules, which can be used and evaluated automatically. Therefore only a small set of documents and data sets must be modified (e.g. according to the requirements of the EMT) to achieve a new set of results, because most of the tasks were realized with respect to automatic operations. The workflow of data for the evaluation process is shown in Figure 6.5-1.

a) *MySQL Database*: inspection data are registered in a MySQL database using a web interface. Individual inspectors have performed this task at the end of each inspection cycle. The design of the database has been described in an overview in chapter 6.4.1 and includes some basic verification of data for internal consistency (less text boxes for comments but predefined select boxes to avoid typing errors, a predefined workflow for input of data to avoid missing values, etc.)

b) *Access Database*: To achieve a higher availability and portability of data, we transferred all tables of the MySQL database to an Access database. This step includes some consistency checks as well, e.g. range of allowed data, number of sessions (no duplicated session, etc).

c) *Basic matrices*: These matrices present a first summary of basic data for later evaluation tasks and were created using simple SQL statements at the Access database (e.g. team and inspector basic data, defect detection matrix for inspection, etc.). I will present a closer look at these matrices in chapter 6.5.2.

d) *Input matrices*: According to task-specific need of data the input matrices were built to get a focus on the investigation topics as input for specific task oriented calculation models. I will use MS Office tools (e.g. MS Excel) to achieve some kind of data aggregation for tasks (e.g. defect detection matrix with respect to severity levels, etc.). Input matrices are an integrated component of the "*working space*".

e) *Calculation of tasks and models*: Based on task-specific evaluation models, I will use tools, like MS Excel and SPSS to calculate these models.

f) *Presentation:* this final step document the individual results of the analysis process and provide feedback to the participants. This step will not be provided in this text.

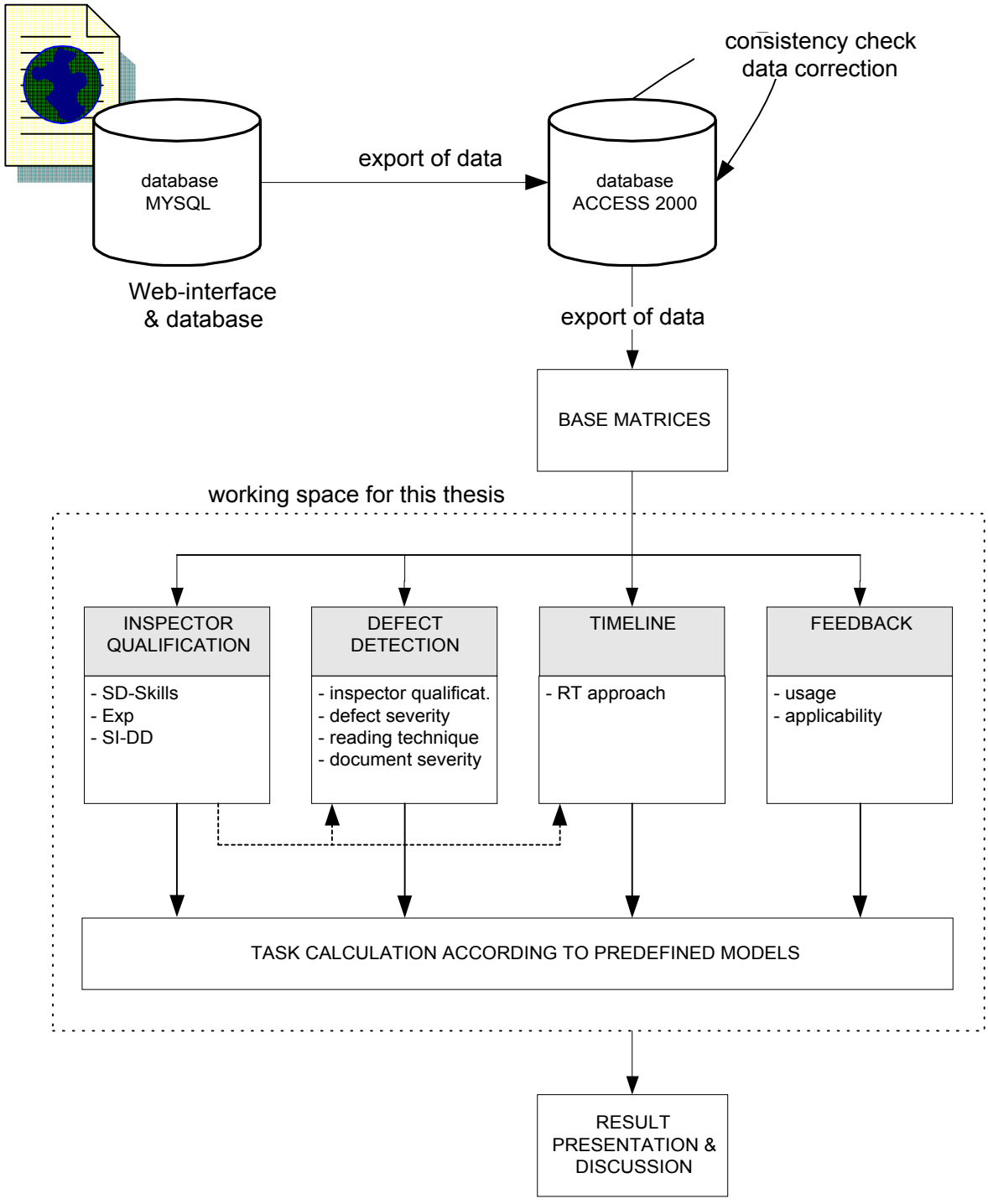


Figure 6.5-1: Database model for inspection experiment

I will use plain text (ASCII-format) to provide data exchange with the corresponding process stages and programs. We also use EER notation to describe information (for basic and input

matrices) and the relation to each other for a structured view on the data. Nevertheless the data are not stored in database structure.

6.5.2 Model of Basic data

As described in the previous chapter, base matrices represent a first summary of inspection data and are generated by simple SQL-statements dropped at the Access database. To provide a solid presentation, we choose the EER notation (see Figure 6.5-2 for details).

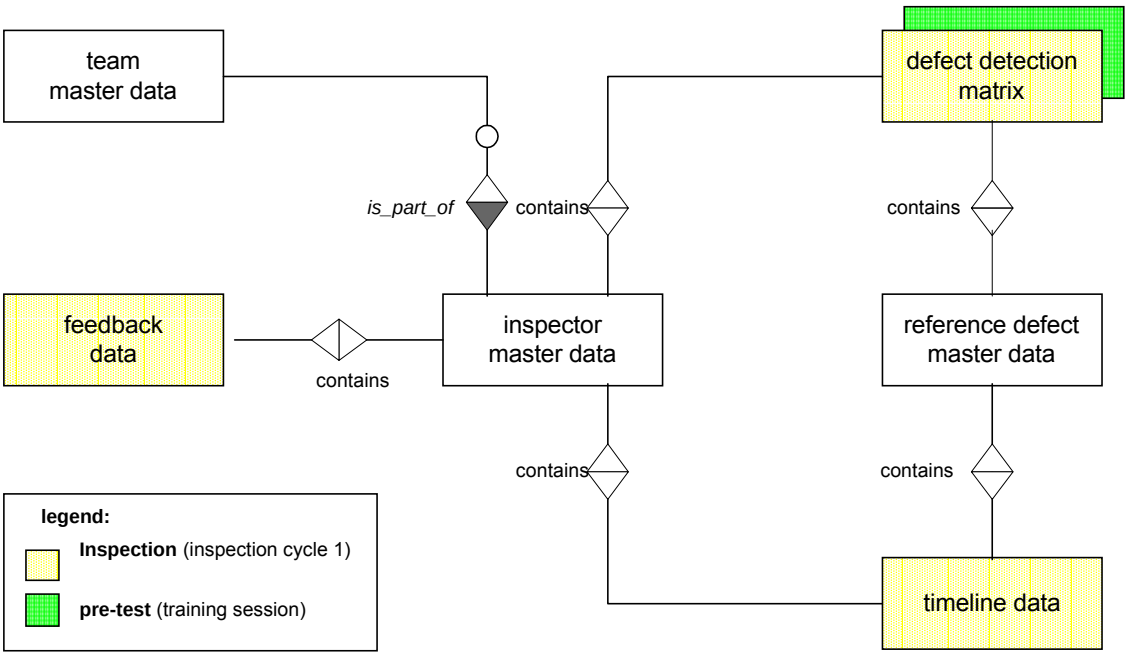


Figure 6.5-2: Database model for basic data

The database model for basic data contains two different blocks of entities: (i) activity independent information and (ii) activity dependent information.

Activity independent information:

a) Team master data: this table consists of basic information of each team (30 data sets) containing team-identification, team reading technique (SBR, CBR), team size, etc.

b) Inspector master data: basic information of inspectors (177 data sets relevant for inspection evaluation) containing inspector identification, individual reading technique role, initial qualification, etc.

c) Reference defect master data: basic information about reference defects (97 data sets) containing defect identification, severity and location

Activity dependent information: This subset of data depends on the session (inspection and re-inspection) containing identical types of information but with respect to the session. This thesis focuses on inspection data only (1st inspection cycle).

a) Defect detection matrix (0-1 matrix): This matrix describes inspector in relation to reference defects found during the session, containing 1s (defect found) and 0s (defect not found).

b) Timeline data: The data are represented in a list containing time relevant information about defect detection, e.g. inspector and team identification, inspection effort, starting time, matched reference defect identifier and the time when the corresponding has been found.

c) Feedback data: The table reflects the feedback questionnaire including inspector estimation for remaining defects.

These basic matrices are used to generate task specific data for model calculation according to hypothesis and theories.

6.5.3 Task description

Figure 6.5-3 presents an overview about the 4 major evaluation tasks, described in the section "*research questions*" in chapter 6.1, and their dependencies to each other.

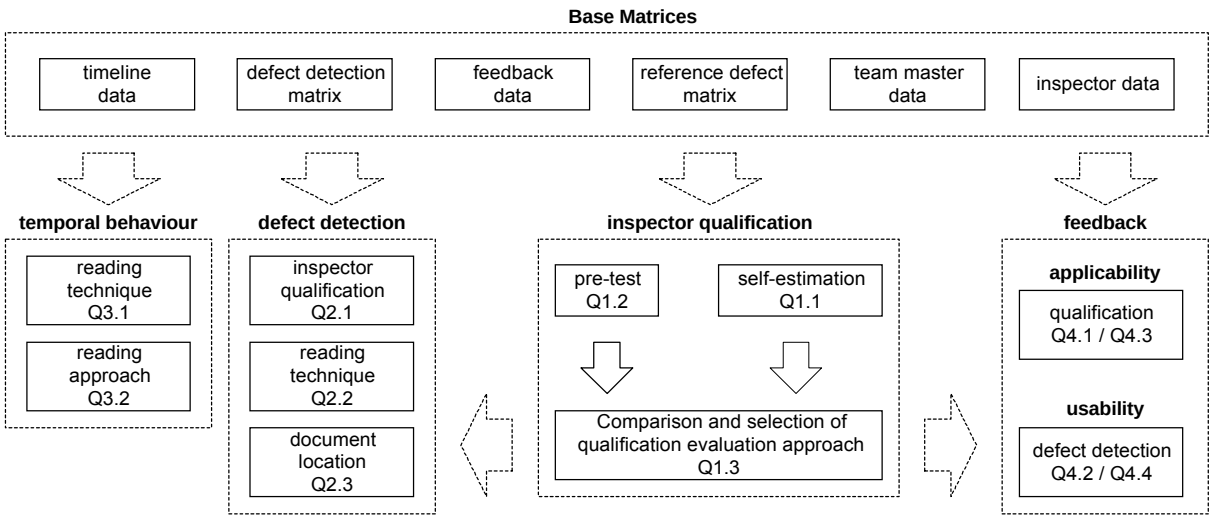


Figure 6.5-3: Task Description

Investigation of inspector qualification (Task Q1): The results of the first part of evaluation can be summarized as qualification investigation according to the individual models (*self-estimation* and *pre-test*). In comparison to the pre-defined supervisor qualification assessment, I will choose one most realistic (and practical) approach for further investigation.

Defect Detection (Task Q2): This section covers some aspects of defect detection rates, i.e. the number of defects found during the inspection process with special focus on defect severity classes, reading techniques and document location with respect to qualification levels.

Temporal behaviour (Task Q3): Because we use different reading techniques approaches (checklist and scenario-based reading) during the experiment, it is interesting, how this approaches influence defect detection rates during the inspection process. Concerning these approaches, we must also distinguish between method-oriented (following the method-description) and inspector-related (following individual preferences in reading documents) differences.

Feedback (Task Q4): Because of the experiment specification, this reading technique approaches have to be used. Therefore the acceptance and usability of these techniques in the inspectors view is a very interesting topic (useful techniques in practical environment, etc.)

6.6 Chapter Summary

This chapter described the experimental environment as well as research questions and the proceeding of investigation and evaluation.

The research questions focus on (i) the definition of qualification calculation, using different models and covers aspects of (ii) defect detection ratings, (iii) temporal behaviour and (iv) subjective feedback factors for further investigations.

The experiment description gives a detailed overview of software artefacts and the preparation with seeded defects (including some visualization of predetermined key facts). Inspectors use pre-defined reading techniques (i.e. guidelines) to proceed and provide feedback to the experiment, methods and proceedings.

Once, the experiment has finished, data has to be collected, checked, and prepared for further investigation. I described the operational experimental environment, structure and volume of data and evaluation process, used in this thesis.

7 Results

This section provides results of the inspection experiment according to the research questions (chapter 6.1) using the artefacts and the environment described in the previous chapter.

I will cover

- Inspector qualification
- Defect detection rates
- Temporal behaviour of defect-finding processes according to reading technique approaches
- Feedback-Results with respect to applicability and usability of reading techniques.

Due to a wide range of investigation possibilities, it is not possible to cover all relevant aspects but only a selection of the most interesting facts.

7.1 Investigation of Inspector Qualification

To get a realistic view on inspector qualification, we introduced a primary qualification rating according to a qualification-test before the experiment starts at all. The inspection leaders (IL) graded all inspectors with respect to the results of this qualification test. This test introduced the students to the whole workshop and showed them, what they had to expect during the course. This test contained the implementation of a small software product with database access using software development techniques and a final presentation of the product.

Because this test doesn't refer especially to software inspections but to software engineering basics and programming knowledge, we introduced additional qualification levels (just with special focus on the inspection process) according to a feedback questionnaire and training session of inspection techniques.

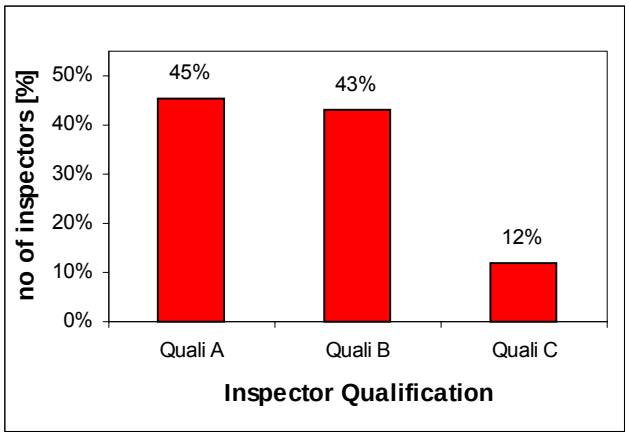
7.1.1 Supervisor Qualification Assessment (*SD-Skills*)

According to the entry test, a small software product, performed by the inspectors, has been assessed by the lab-assistants. The "software product" has been classified to

- Functionality
- User Interface Design
- Understanding of product and software engineering tools and methods

According to the subjective impressions of the lab-assistants the inspectors got a rating A for high qualification, B for medium qualification and C for low qualification. In our terminology

of qualification levels, “A” students are the highest qualified inspectors and level “C” students own the lowest qualification level.



There are 177 inspectors consisting of 80 (45%) inspectors with qualification A, 76 (43%) inspectors with qualification B and 21 (12%) inspectors with qualification C. Figure 7.1-1 shows the overall distribution of inspector qualification.

According to SD-Skills about 88% of all inspectors are A or B qualified (almost equal distribution) and only 12% own qualification C. Obviously there is a overhead with well-qualified inspectors.

Figure 7.1-1: SD-Skills: Distribution of Inspectors

Because of the subjectivity of IL-assessment and the small causal context to software inspections (primary focus on program development), I use the feedback questionnaire (self-estimation of skills) to get a more realistic view on the qualification ratings. Nevertheless there is still the problem of subjectivity, now in estimating own skills. This might result in over- and underestimation of the participants.

7.1.2 Qualification Rating according to self-estimation (EXP, Q1.1)

To improve the weakness of the supervisor qualification classification we use some kind of self-estimation by the inspectors, which has been gathered during examination of the inspection process using questionnaires (see appendix B for the complete questionnaires).

The rating concerns four sections

a) General skills (e.g. reading and understanding of technical documents)

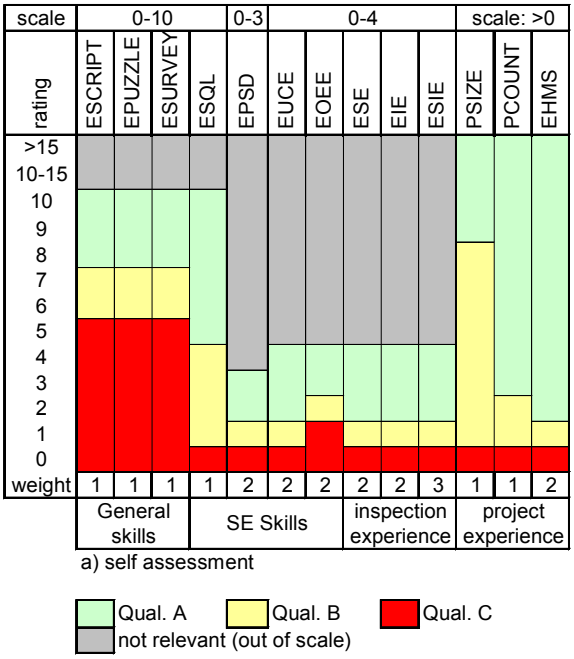
b) Software Engineering methods (e.g. knowledge in using UML, EER, etc)

c) Inspection relevant questions (e.g. practical usage of inspections in the past, etc.)

d) Project experience in general (number and size of project, etc.)

Following this approach, all individual questions has been classified to numerical values, which represent a possible range of answers and a relationship to qualification ranges. Additional all questions has been equipped with a weight (1..normal, 2..high and 3..very high importance), which show the relevance for inspection processes and several limits to achieve qualification level C, B or even A.

The summary of points for all questions has been assigned to the final self-assessment qualification level for each inspector. (Figure 7.1-2) presents the rating-scale in a short summary.

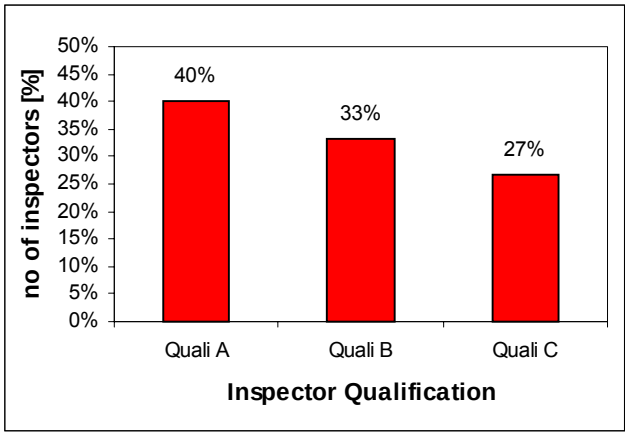


Short-Cut	Description
ESCRIPT:	Script Experience (ability to understand explanations, e.g. in specifications)
EPUZZLE:	Puzzle Experience (ability to solve problems efficiently)
ESURVEY:	Survey Experience (ability to solve more complex problems)
ESQL:	SQL Experience
EPSD:	Experience in practical Software Development (Experience questionnaire, FB 03)
EUCE:	Use-Case Experience (Experience questionnaire, FB 04)
EOOE:	OOA, EER and UML experience (Experience questionnaire, FB 05)
ESE:	Experience to handle Specification documents (Experience questionnaire, FB 06)
EIE:	Experience to handle Inspections (Experience questionnaire, FB 08)
ESIE:	Experience to handle Software Inspections (Experience questionnaire, FB 09)
PSIZE:	Number of person-weeks for project teamwork
PCOUNT:	Number of projects an inspector participated
EHMS:	Number of Specifications

Figure 7.1-2: EXP: Qualification Rating Scale for Self-Assessment

The marked areas in Figure 7.1-2 shows the corresponding qualification levels for each question. Because of different question classifications another bar shows the “out-of-range” areas. An inspector has no or even very less knowledge in a defined area, he is rated at level “C”. If he heard about the subject mentioned in the question and already used it, he has been rated at level “B”. Otherwise if he defined himself as “specialist” or used the methods frequently he has been rated at “A” level. The corresponding weights in a scale from 1 to 3 are described at the bottom of each question. Basic, maybe general skills are equipped with the weight of factor 1, important skills own factor 2 and very important skills for performing inspections are rated at 3.

After applying this rating schema to all individuals, all points has been summarized. The possible ratings are in the range of at least 0 (if all questions has been rated 21 points) and at most 63 points. The results after evaluation show a range between 20 and 63. This range has been used to calculate the final qualification level; lower reached limit + 30% of the range until lower limit + 50% of the range are finally rated at B, below C and above A.



Concerning self-estimation qualification ratings, the distribution is quite more balanced but the ranking is still the same (most of inspectors are A, followed by B and C qualification). Figure 7.1-3 shows 71 high-qualified inspectors (40%), 59 level B inspectors (33%) and at last 47 level C inspectors (27%).

In comparison to SD-Skills the number of inspectors for qualification A and B decreases and the number of qualification C inspectors increases.

Figure 7.1-3: EXP: Distribution of inspectors

Figure 1.2.-4 presents inspector distribution according to experience qualification sections (*EXP*) within our rating scale and our predefined qualification-relevant sections in more detail.

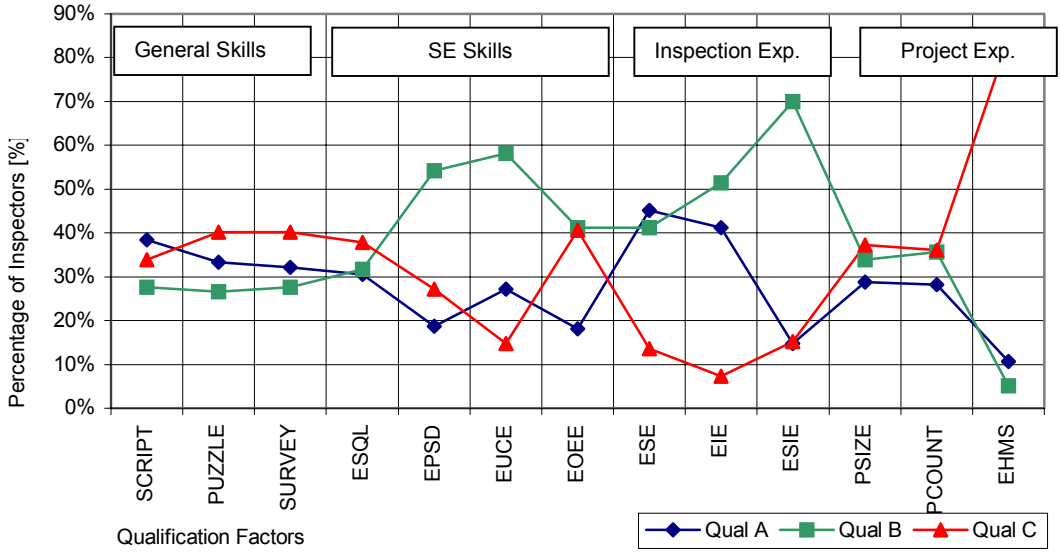


Figure 7.1-4: EXP: Distribution of inspectors according questionnaire sections

A closer look at the qualification ratings according to the four sections will allow some facts, which might be interesting for further investigations. Therefore I will cover some aspects, which are most interesting in my opinion.

Concerning general skills, there is a balanced distribution of the number of inspectors at each qualification level. About 60% of the inspectors are B qualified, i.e. they are experienced in

the area of software engineering (SE-Skills) at the beginning of the experiment. Because the participants are students of computer science and the course focus on software engineering topics (tools and methods) they own basic knowledge (also due to the qualification-test at the beginning of the course) in this area. Most students work as programmers in practical environments and have to use specifications for their work. Therefore their skills concerning the usage of *specifications, inspection,* etc is quite good. Only a few inspectors are novices in the area of inspection and project management. Another reason for the high number of level B inspectors especially in *ESIE* (estimation software inspection experience) may be the introductory training session at the beginning of the experiment, where they got in contact with software inspection techniques.

The following tables show the distribution of inspection qualification in more detail:

(a) The section *general skills* describes the inspectors ability to solve (sophisticated) problems and read technical documents (i.e. specifications).

Qualification factor	Qualification A		Qualification B		Qualification C		Summary	
	[no.]	[%]	[no.]	[%]	[no.]	[%]	[no.]	[%]
SCRIPT	68	38%	49	28%	60	34%	177	100%
PUZZLE	59	33%	47	27%	71	40%	177	100%
SURVEY	57	32%	49	28%	71	40%	177	100%
Qual. Exp.	71	40%	59	33%	47	27%	177	100%

Table 7.1-1: EXP: Qualification Distribution according to General Skills

According to the rating scale in this text, the majority of inspectors (about 38%) are specialists in reading technical documents (*SCRIPT*) but less experienced in solving sophisticated problems (about 40% of all inspectors concerning *PUZZLE* and *SURVEY*).

(b) *Project experience* refers to the size and number of project an inspector participated in the past and number of specifications (which indicates the usage of a software model).

Qualification factor	Qualification A		Qualification B		Qualification C		Summary	
	[no.]	[%]	[no.]	[%]	[no.]	[%]	[no.]	[%]
PSIZE	51	29 %	60	34 %	66	37 %	177	100 %
PCOUNT	50	28 %	63	36 %	64	36 %	177	100 %
EHMS	19	11 %	9	5 %	149	84 %	177	100 %
Qual. Exp.	71	40 %	59	33 %	47	27 %	177	100 %

Table 7.1-2: EXP: Qualification Distribution according to project experience

Considering qualification parameter *PSIZE* (project size) the majority of inspectors (37% = 66 inspectors) did never participate in a software development team, about 34% (60 inspectors) are team members up to 10 person-weeks and about 29% participated in

projects with 10 or more person-weeks so far. *PCOUNT* describes the number of projects for each inspector. 64 inspectors (36%) have never been involved in a project, 63 inspectors (36%) participated in up to 3 projects and 50 inspectors (28%) participated in 10 or more projects.

Considering *EHMS*, which defines the number of specifications, the inspectors participated in the past (execution or development), 19 inspectors (11%) defined themselves as specialists, 28 inspectors as less or un-experienced. But there are only 47 inspectors (27% of all inspectors) who rated this question, i.e. 130 inspectors, who did not rate the question, had been considered as un-experienced as well.

c) *SE-Skills* themselves refer to the main-topic of the SE-workshop and is the main point of interest of the inspectors. The summarized results is shown in Figure 1.2.-6.

Qualification factor	Qualification A		Qualification B		Qualification C		Summary	
	[no.]	[%]	[no.]	[%]	[no.]	[%]	[no.]	[%]
ESQL	54	31 %	56	32 %	67	37 %	177	100 %
EPSD	33	19 %	96	54 %	48	27 %	177	100 %
EUCE	48	27 %	103	58 %	26	15 %	177	100 %
EOEE	32	18 %	73	41 %	72	41 %	177	100 %
Qual. Exp.	71	40 %	59	33 %	47	27 %	177	100 %

Table 7.1-3: EXP: Qualification Distribution according to SE Skills

SE-Skills accumulate database knowledge (*ESQL*), experience in practical software development (*EPSD*), use-case experience (*EUCE*) and object-oriented data modelling (*EOEE*), i.e. main topics of the SE-workshop. Table 7.1-3 shows the distribution to each qualification factor. There is a rather uniform distribution of inspector qualification in the area of SQL and database knowledge, but significant differences between the other factors. Concerning *EPSD* (experience in practical software development), there are 33 (19%) high-experienced inspectors and 48 inspectors (27%) with less or no experience.

Only a few inspectors, 48 inspectors (27 %) of all inspectors are familiar with use-case-scenarios (*EUCE*), the majority, 103 inspectors (58 %) did hear about use-cases and 26 inspectors (15%) are un-experienced at all. Concerning OOA/EER/UML notation techniques (*EOEE*), only 32 inspectors (18%) declare themselves as high-experienced or even specialists, 73 inspectors are less experienced and 72 inspectors are unfamiliar with OOA/EER/UML notations so far (each 41%).

d) The qualification section *Inspection Skills* summarizes the experience of using specification usage (*ESE*), inspection knowledge (*EIE*) and software-inspection skills (*ESIE*). The summarized results are shown in Figure 1.2.-7.

Qualification factor	Qualification A		Qualification B		Qualification C		Summary	
	[no.]	[%]	[no.]	[%]	[no.]	[%]	[no.]	[%]
ESE	80	45 %	73	40 %	24	15 %	177	100%
EIE	73	41 %	91	52 %	13	7 %	177	100%
ESIE	26	14 %	124	71 %	27	15 %	177	100%
Qual. Exp.	71	40 %	59	33 %	47	27 %	177	100%

Table 7.1-4: EXP: Qualification Distribution according to SE Skills

Concerning *ESE* (specification experience), 80 inspectors (45%) are experts, 73 inspectors (40%) are familiar with the area, but with less practical experience and 24 inspectors (15%) are new to specifications at all. 164 inspectors (93%) know about inspections (qualification A or B) in general (not limited to the area of software engineering) and 13 inspectors (7%) are new to inspections techniques.

The Knowledge of the specific method “software inspection” decreases experts to 26 inspectors (14 %) and increases the number of un-experienced inspectors to 27 (15%).

Concerning self-estimation, there also exist some points to remember:

- Over/under-estimation of inspector feedback: Although there is unreliability in the self-estimation because some inspectors will over-estimate their knowledge, others will under-estimate their skills we expect a more realistic view in sum.
- Completeness of Experience questionnaire: The number and main focus of questions varies in the range of application, projects, resources, etc. Because of the individual project-requirements, managers have to use different configurations for qualification ratings to get useful results.
- Project managers have to configure well-defined questionnaires according to the project environment and can use the results to find the best team composition with respect to qualification levels.

7.1.3 Qualification Rating according to the PRE-Test (SI-DD, Q1.2)

The analysis of data collected during past inspection activities, e.g. defect detection rates, etc. will be the best solution for actual qualification assessments. Nevertheless managers will perform inspection for the first time (as a defined pre-condition of this thesis) and do not have any relevant data. To solve this problem, they can establish some kind of PRE-test, i.e. a mini-inspection, which is limited in inspection time and number of “known” defects, but in a useful environment to get data for qualification assessment.

We performed a tutorial for the inspectors to get familiar with inspections techniques in a time-slot of about 2-3 hours and included some training session as “mini-inspection”. The inspectors had to collect defects, categorized them as minor and major. Based on this data we set up another qualification rating model, the SI-DD.

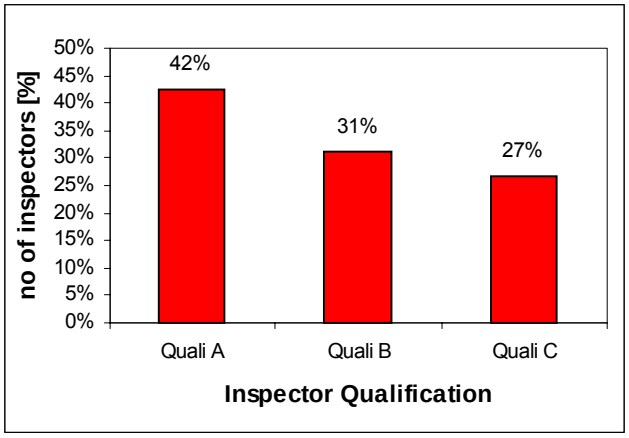
The defect detection rates for the parameters *minor*, *major* and *all defects* has been noted and rated by the inspectors. According to the range of found defects, we set up the limits for qualification level C, B and A.

	Minor	Major	All
Qualification C	0	0	≤ 2
Qualification B	1-2	1-2	2-4
Qualification A	>2	>2	>4
Weight	1	2	1

Table 7.1-5: SI-DD: Qualification Rating

Table 7.1-5 shows the rating schema for the defect detection process during the PRE-test. Again, every parameter has been equipped with a weight, in our case, the detection of major defects is more important than minor defects.

Figure 7.1-5 shows the distribution of inspector qualification according to SI-DD, after applying the qualification assessment model to all individuals.



Concerning qualification ratings of the pre-test with respect to self-estimation the number of level A inspectors increases from 71 to 75 (i.e. about 6%), the number of level B-inspectors decreases from 58 to 55 (i.e. about 5%) and the number of level C inspectors doesn't change in sum (i.e. 47 inspectors).

Figure 7.1-5: SI-DD: Distribution of Inspectors

As described above, the overall qualification rating is a summary of three defect-detection rates, all, minor and major defects). Figure 7.1-6 shows the distribution according to each defect detection rate.

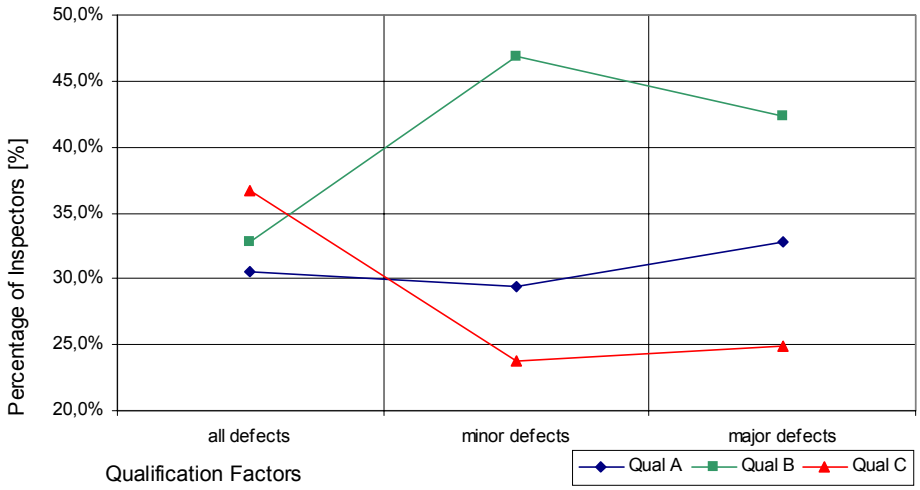


Figure 7.1-6: SI-DD: Distribution of Inspectors according to Defect Detection

Concerning all matched defects, 30% (54 inspectors) of all inspectors are classified as experts, 33% (58 inspectors) own level B and even 37% (65 inspectors) are classified level C.

The number of level B inspectors increases to 47% (83 inspectors) concerning defect detection rate of minor defects, and increases to 42% (75 inspectors) with respect to major defects. The number of level C inspectors decreases in about the same range while the number of level A inspectors doesn't change very much

Qualification factor	Qualification A		Qualification B		Qualification C		Summary	
	[no.]	[%]	[no.]	[%]	[no.]	[%]	[no.]	[%]
ALL	54	30 %	58	33 %	65	37 %	177	100%
Minor	52	29 %	83	47 %	42	24 %	177	100%
Major	58	33 %	75	42 %	44	25 %	177	100%
Qual. SI-DD	75	42 %	55	31 %	47	27 %	177	100%

Table 7.1-6: SI-DD: Distribution of Inspectors according to Defect Detection

Concerning a pre-test, there also exist some points to remember:

- Using (real) inspection results from the past, the highest reliability of data will be guaranteed, but this data are hardly comparable at all because of different environment, specifications, application areas, real defects etc.
- To get a "perfect" inspection team, i.e. at a high qualification level it's difficult to establish a mini-inspection because of a lack of resources. Some kind of testing environment has to be developed to get comparable results. These results can be used for evaluation only, because there is no real context to projects performed in the

company at all. Additionally only a subset of possible inspectors (out of the testing environment) perform a real inspection at all.

It's the project managers' task to find out a proper evaluation method for qualification rating, considering advantages and disadvantages of the methods.

7.1.4 Comparison of qualification assessment models (Q1.3)

I introduced some qualification rating schemes in the previous chapters. To support the decision to get the best evaluation model, the results has to be comparable. The comparability is achieved using schemes, which maps inspector capabilities to a range of A-C, as described.

Figure 7.1-7 describes differences of the number of inspectors according to different evaluation models. Introducing a rating of qualification distribution, most inspectors are level "A" followed by level "B" and level "C" with the least number of inspectors.

There are only few differences between the three different qualification ratings (40% - 45%) with respect to qualification level "A". Considering qualification level "B" EXP and SI-DD are in the range of 33% to 31%; only the number of inspectors according to SD-Skills are 43%. This might reason in the focus of the evaluation method (SD-Skills are calculated according to a software engineering qualification test).

Equivalent to qualification level B the distribution of qualification level C is quite different. 27% of all inspectors have been rated at "C" concerning *EXP* and *SI-DD*; but we have only a few inspectors (12%) rated at level C concerning SD-Skills. The configuration of the experiment participant team might be a reason for this difference. They are students of computer science and the course focus on software engineering topics, so the own basic knowledge in this area.

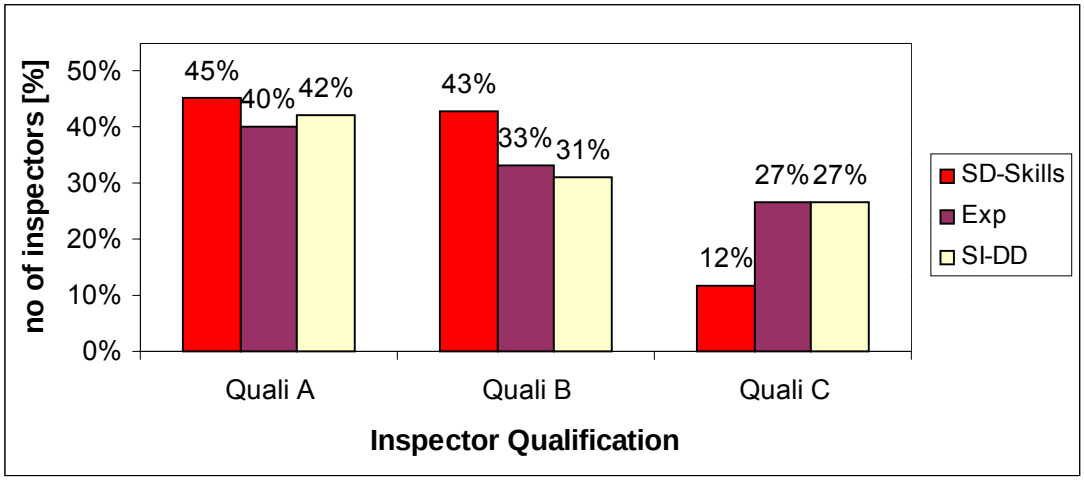


Figure 7.1-7: Qualification: Comparison of Qualification Distribution

Table 7.1-7 gives a more detailed, numerical overview about qualification configuration within the experiment participant team.

Qualification rating	Qualification A		Qualification B		Qualification C		Summary	
	[no.]	[%]	[no.]	[%]	[no.]	[%]	[no.]	[%]
SD-Skills	80	45 %	76	43 %	21	12 %	177	100%
Exp	71	40 %	59	33 %	47	27 %	177	100%
SI-DD	75	42 %	55	31 %	47	27 %	177	100%

Table 7.1-7: Qualification: Comparison of Qualification Distribution

Because of the distribution of the number of inspectors according to qualification rating models and the focus of the experiment itself, I will choose the *EXP*-qualification model for further investigations. Another reason for this choice is the practical relevance of qualification ratings. It is quite impossible within a company to establish "qualification-tests" because there is always a lack of resources and there might be no reason for a qualification test (acceptance by the employees). The most realistic approach is inspection knowledge in the past, but - as described at the beginning of this section - there are no experiences so far.

The next sections show results out of the experiment itself and the defect detection rates.

7.2 Defect Detection Rate

The purpose of inspection is the reduction of defects in software artefacts, in our case a software specification document for a large-scale software product.

Inspections are usually used in early stages of software production to reduce cost for maintenance and repair in further software engineering proceedings or even in practical environment at the customer.

This chapter gives an overview of defect detection rates (*DDR*), using data collected at our experiment after the 1st inspection cycle, concerning inspector qualification, reading technique roles, defect severity classes and the location of defects within the software document.

7.2.1 Defect Detection Rates according to Inspector Qualification (Q2.1)

As outlined in chapter 7.1.4, I will use the inspector qualification model *EXP* (self assessment) for further investigations. I will cover the number of defects noted during the inspection and the number of matched defects (i.e. defects seeded by *EPT*).

Concerning all noted defects (6198 defects found in sum), about 41% has been found by "A" inspectors, 31% has been detected by level "B" inspectors and about 28% by level "C" inspectors. Looking at "real" defects, i.e. matched defects (2288 defects found in sum), the range of defect detection rates is wider (42% level A, 32% level B and, 26% level C).

Because of a non-uniform distribution of inspectors to the corresponding qualification level, a comparison of the absolute numbers of defects detected is not possible. Table 7.2-2 shows the distribution of matched and noted defects in summary, to get a impression about the amount of data, collected during the inspection process.

Defect Detection Rate (DDR)	Qualification A		Qualification B		Qualification C		Summary	
	[no.]	[%]	[no.]	[%]	[no.]	[%]	[no.]	[%]
Noted defects	2529	41%	1931	31%	1738	28%	6198	100%
Matched defects	956	42%	727	32%	605	26%	2288	100%

Table 7.2-1: Absolute number of defects found during inspection

Including inspector distribution, according to qualification and regardless of inspector reading technique, the average number of defects found during inspection is about 14 defects (deviation of 7,7) concerning high-qualified inspectors, about 12 defects (deviation of 7,9 defects) concerning medium-qualified inspectors and about 12 defects (deviation of 6,8 defects) concerning low qualified inspectors. Therefore higher qualified inspectors find more defects with respect to lower qualified inspectors.

Figure 7.2-1 gives a closer look at defect detection distribution according to inspector qualification and reading techniques.

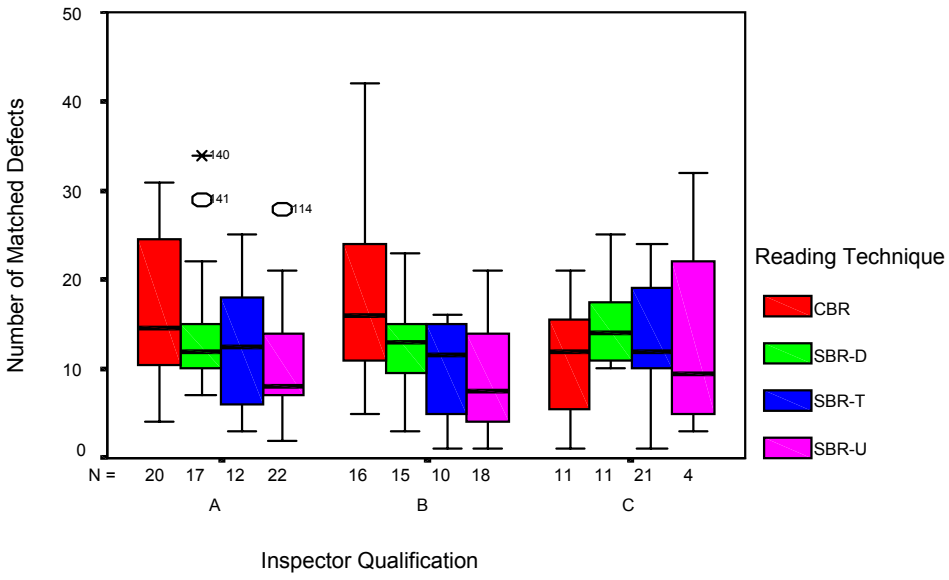


Figure 7.2-1: DDR: Qualification vs. Reading Technique

Concerning qualification level A, CBR inspectors find more defects (in average about 16.6 defects) followed by SBR-D (14.3 defects), SBR-T (12.8 defects) and SBR-U (10.4 defects). With respect to medium qualified inspectors, the number of defects, found by inspectors increases at CBR and decreases at SBR reading technique approaches (the ranking doesn't change). Concerning low-qualified inspectors, there is variation, because in average CBR

inspectors find 10.6 defects, while SBR inspectors find 12.7 till 15.2 defects (thus, low-qualified CBR inspectors find less defects than low-qualified SBR-inspectors). Regarding high and medium qualified inspectors, CBR will be the best choice, but lower qualified inspectors will achieve better results using SBR reading techniques. Table 7.2-2 shows the results in more detail.

Defect Detection	Qualification A				Qualification B				Qualification C			
		SBR				SBR				SBR		
	CBR	D	T	U	CBR	D	T	U	CBR	D	T	U
Min	4	7	3	2	5	3	1	1	1	10	1	3
Max	31	34	25	28	42	23	16	21	21	25	24	32
Mean	16,6	14,3	12,8	10,4	18,1	12,4	10,0	8,4	10,6	15,2	12,7	13,5
Std-Dev.	8,61	7,39	7,53	6,16	10,06	5,65	5,60	5,80	6,86	5,25	6,39	12,87
# SD[10]	97	97	97	97	97	97	97	97	97	97	97	97
Av. DDR[11]	17%	15%	13%	11%	19%	13%	10%	9%	11%	16%	13%	14%

Table 7.2-2: DDR: Qualification vs. Reading Technique

Another basic evaluation concerns defect severity levels, which can be found during inspection by the participants. Because reference-defects have been equipped with severity levels *trivial* (0), *minor* (1), *major* (2) and *critical* (3) only matched defects can be analysed for this investigation task.

Due to an unequally distributed number of seeded defects at every severity level (see Chapter 6.2.1.2), I must use some form of "normalisation". Therefore I use

$$number\ of\ defects_{defect\ class} := \frac{real\ number\ of\ defects\ \det ected_{defect\ class}}{number\ of\ seeded\ defects_{defect\ class}} * 100\,[\%].$$

For example, an inspector (high-qualified) found 20 minor defects within the whole document. Because 39 defects have been seeded within the whole document he achieve a defect detection rate (DDR) of 51% of all possible defects. Figure 7.2-2 shows this distribution of defect detection according to inspector qualification and defect severity classes.

[10] Number of seeded defects according to the evaluation task

[11] Average Defect detection rate according to the evaluation task

Concerning trivial defects (severity level 0), low qualified inspectors find most defects (16% of all seeded trivial defects). High- and medium-qualified inspectors achieve best results according to minor defects (16% each). Low qualified inspectors detect most major defects (18% of all seeded major defects) and, finally, analysing critical defects (severity level 3), the detection rate of medium- and low-qualified inspectors is the highest one (18% each).

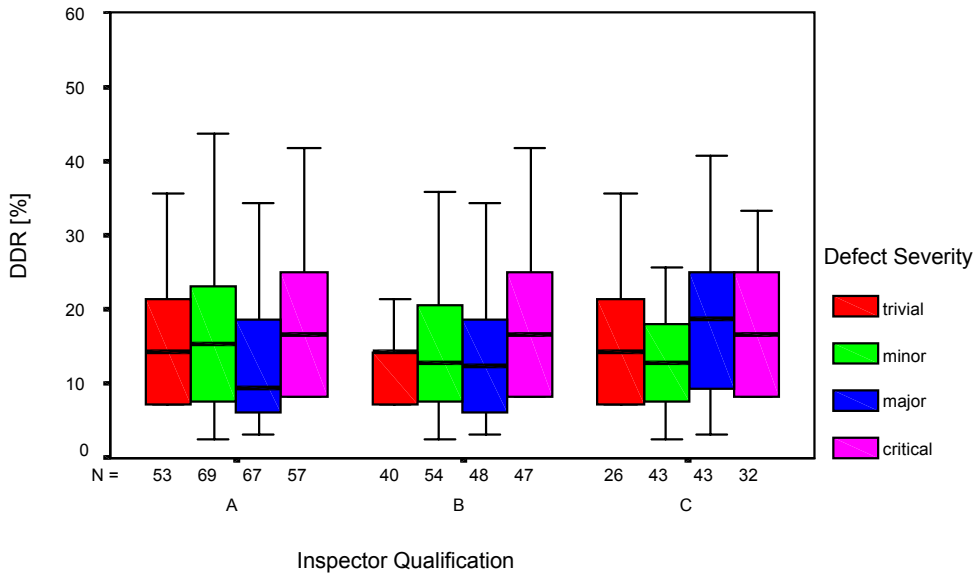


Figure 7.2-2: DDR: Inspector Qualification vs. Defect Severity

Because of this distribution, further investigations will be necessary to achieve significant findings. Table 7.2-3 shows the defect detection rates. Summarising all trivial defects, found by all high-qualified inspectors (qualification A), they detected at least 1 defect and at most 6 defects with respect to one individual inspector. Minor defects will find easier with qualification level A and B, while lower qualified inspectors find more major and critical defects.

Defect Detection	Qualification A				Qualification B				Qualification C			
	0[12]	1	2	3	0	1	2	3	0	1	2	3
Min	1	1	1	1	1	1	1	1	1	1	1	1
Max	6	17	16	6	5	19	15	5	6	14	13	4
Mean	2,1	6,4	4,4	2,0	1,9	6,2	4,5	2,2	2,3	5,5	5,7	2,1
Std-Dev.	1,16	3,76	3,49	1,21	1,05	3,97	3,01	1,06	1,44	2,97	3,17	1,01
# SD	14	39	32	12	14	39	32	12	14	39	32	12
Av. DDR	15%	16%	14%	17%	14%	16%	14%	18%	16%	14%	18%	18%

Table 7.2-3: DDR: Qualification vs. Defect Severity

[12] Defect Severity Classes: Trivial (0), Minor (1), Major (2), Critical (3)

Another interesting fact is the defect detection rate with respect to the reading technique for choosing the best reading technique approach.

7.2.2 Defect Detection Rates according to Reading Technique Roles (Q2.2)

One specified reading-technique-role (*CBR*, *SBR-U*, *SBR-D*, *SBR-T*) has been assigned to individual inspectors, who form an inspection team with 4-6 inspectors (team composition will be a topic for further investigations). The usage of reading techniques has been described in chapter 6.2.2.

Figure 7.2-3 shows the defect detection distribution with respect to reading techniques. The overall defect detection rate of CBR inspectors is the highest (32%), followed by the technical designer (SBR-D) (26%), tester view (23%) and user view (19%) One reason for the distribution of defects found might be the non-uniform distribution of overall defects to all locations and reading technique focus on document location.

Concerning CBR reading technique, which covers the whole document for defect detection, high- and medium qualified inspector achieve the best results, i.e. they find more defects than lower qualified inspectors (18 and 17 defects found in average). Low qualified inspectors, using SBR designer view, find more defects in our experiment than higher qualified inspectors. The average deviation is quite smaller than at CBR users. Concerning SBR-T reading technique high- and low-qualified inspectors find almost an equal number of defects. Finally, the inspectors, using the tester view achieve better results, although the are less qualified.

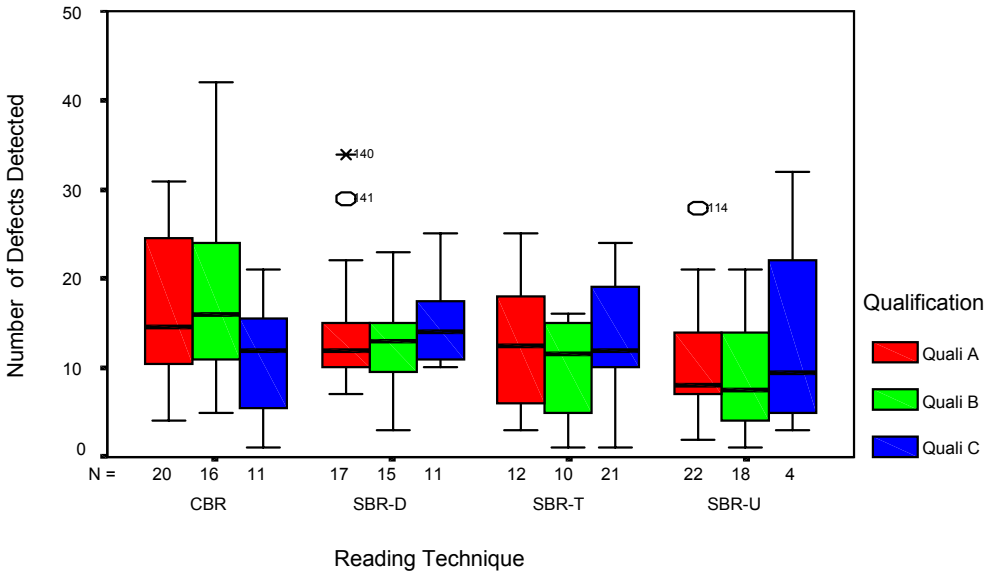


Figure 7.2-3: DDR: Reading Technique vs. Inspector Qualification

Because reading techniques focus to different topics, which map to different document locations, e.g. a user-view covers primary introduction (*INTRO*) and business function (*BUSI*)

and designer view focus on the object oriented class model (*OBJ*) and the description (*DATA*), I have to consider reading technique with respect to document location as well in chapter 7.2.3.

Defect Detection	CBR Qualification			SBR-D Qualification			SBR-T Qualification			SBR-U Qualification		
	A	B	C	A	B	C	A	B	C	A	B	C
Min	4	5	1	7	3	10	3	1	1	2	1	3
Max	31	42	21	34	23	25	25	16	24	28	21	32
Mean	16,6	18,1	10,6	14,3	12,4	15,2	12,8	10,0	12,7	10,4	8,4	13,5
Std-Dev.	8,61	10,06	6,86	7,39	5,65	5,25	7,53	5,60	6,39	6,16	5,80	12,87
# SD	97	97	97	97	97	97	97	97	97	97	97	97
Av.DDR	17%	19%	11%	15%	13%	16%	13%	10%	13%	11%	9%	14%

Table 7.2-4: DDR: Reading Technique vs. Inspector Qualification

Table 7.2-4 shows the individual defect detection rates of all qualification levels according to the reading technique roles. Considering the evaluation data, higher- and medium qualified inspectors achieve higher defect detection rates, using CBR reading technique approaches. SBR-D inspectors find more defects, even with lower inspector qualification.

Another interesting point of view is the defect detection rate according to defect severity classes. Because *critical defects* are harder to repair in comparison to *trivial defects* is will be quite interesting which reading technique will prefer critical defects for example.

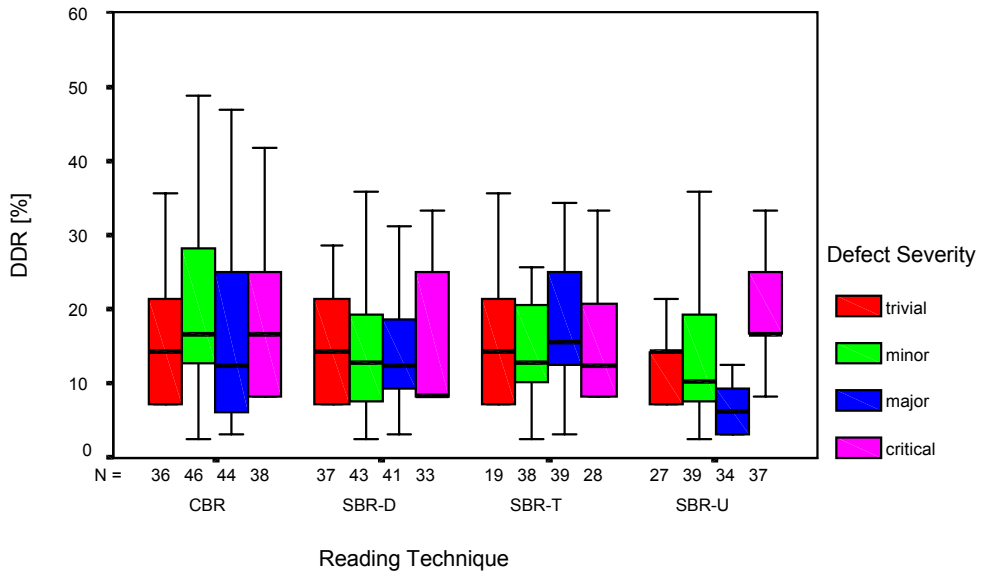


Figure 7.2-4: DDR: Reading technique vs. Defect Severity

Due to an unequally distributed number of seeded defects at every severity level (see Chapter 6.2.1.2), I must use some form of “normalisation” again (percentage of overall seeded defect

severity classes). This evaluation task covers reading technique roles with respect to defect severity classes. Figure 7.2-4 presents the results.

Defect Detection	CBR Defect Severity				SBR-D Defect Severity				SBR-T Defect Severity				SBR-U Defect Severity			
	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3
Min	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Max	6	19	15	6	4	16	16	4	5	10	11	4	6	17	8	4
Mean	2,1	7,6	5,2	2,2	2,1	5,7	5,2	1,8	2,0	5,5	5,7	1,8	2,0	5,4	2,5	2,4
Std-Dev.	1,3	4,4	3,8	1,3	1,1	3,3	3,4	1,1	1,2	2, 5	2,9	0,9	1,3	3,6	1,5	1,0
# SD	14	39	32	12	14	39	32	12	14	39	32	12	14	39	32	12
Av.DDR	15%	19%	16%	18%	15%	15%	16%	15%	14%	14%	18%	15%	14%	14%	8%	20%

Table 7.2-5: DDR: Reading technique vs. Defect Severity

I want to point out some interesting facts (regarding average defect detection rate with respect to distribution of defect severity): The finding of trivial defects (severity level 0) doesn't depend on the reading technique (about 14-15% at each reading technique role). Looking at minor defects, CBR inspection techniques seems to fit better to those defect types (19%, a decreasing number of DDR using the other reading techniques). With respect to major defects, the tester-view found about 18% in average, but SBR-U found only 8% of major defects at the experiment. Finally, concerning critical defects, the scenario approach for users found about 20% of the seeded defects, followed by CBR (18%) and SBR-T/D (15% each). Table 7.2-5 shows the complete results of the real defect detection rate.

As described in the previous section the number of seeded defects at different severity levels vary according to the document location. In the next chapter I will try to analyse the influence of document location with respect to severity levels, reading technique and inspector qualifications.

7.2.3 Defect Detection Rates according to Document Locations (Q2.3)

Defects at various severity levels have been seeded in the inspection document in four different locations: *introduction*, *business functions*, *object oriented class models* and *class descriptions*. Chapter 6.2 presents the distribution of seeded defects with respect to document location. Most of the defects have been seeded in the document location *business function* (BUSI, 38%), followed by *class descriptions* (DATA, 29%), *object oriented class model* (OBJ, 26%) and, finally *introduction* (only about 7%). Therefore, to achieve comparability, I concern the total number of defects according to a defined document location.

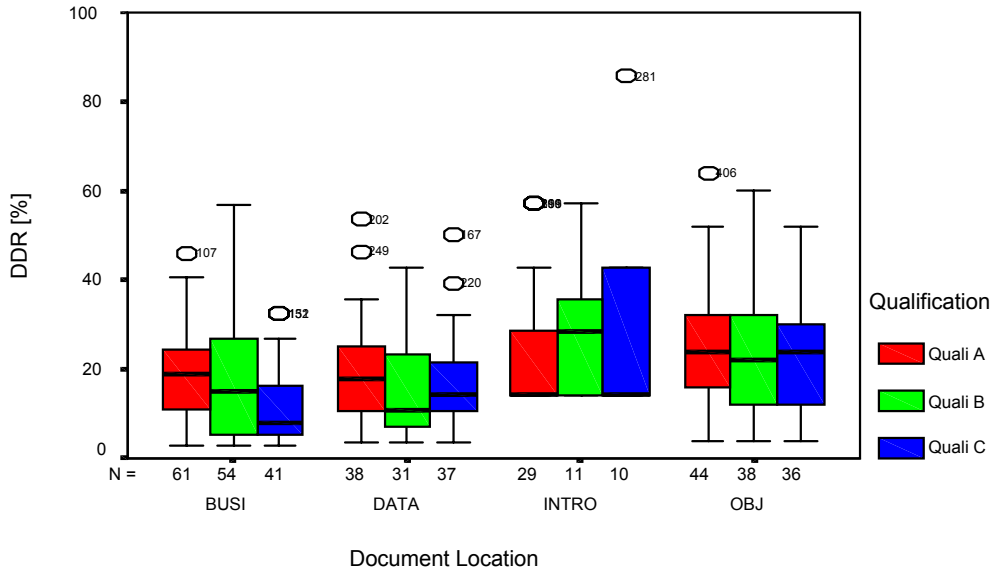


Figure 7.2-5: DDR: Inspector Qualification vs. Document Location

Considering all inspectors according to their qualification rating and the number of defects at every severity level, I normalize with the possible number of defects at every severity class (initial distribution of severity classes). Figure 7.2-5 shows the results of defect detection rate according to inspector qualification. Concerning high-qualified inspectors, they will find more defects in every document location, except the introduction. Analysing introduction (with is affected most by the normalisation), lower qualified inspectors find more defects, but because there are only seeded a small number of defects within the introduction; this fact influences the results.

Table 7.2-6 shows detailed information of real defects found during inspection with respect to the document locations. Concerning *business functions, data descriptions and the object-model*, higher qualified inspectors find most defects (7 defects, 5.2 and 6.4 defects in average). Low qualified inspectors find most defects in the *introduction* and an almost similar number of defects concerning *data description* and *object models*, with respect to higher qualified inspectors.

Defect Detection	BUSI Qualification			DATA Qualification			INTRO Qualification			OBJ Qualification		
	A	B	C	A	B	C	A	B	C	A	B	C
Min	1	1	1	1	1	1	1	1	1	1	1	1
Max	17	21	26	15	12	14	5	4	6	16	15	13
Mean	7,0	6,5	4,6	5,2	4,3	5,1	1,8	1,9	2,0	6,4	5,8	5,8
Std-Dev.	3,98	4,82	4,43	3,22	2,57	2,87	1,17	1,04	1,63	3,55	3,50	3,10
# SD	37	37	37	28	28	28	7	7	7	25	25	25
Av.DDR	19%	18%	12%	19%	15%	18%	26%	27%	29%	26%	23%	23%

Table 7.2-6: DDR: Inspector Qualification vs. Document Location

Not only the number of seeded defects vary at different document locations, but also the assigned severity classes (Table 6.2-1 shows initial defect distribution). E.g. the introduction (*INTRO*) contains no critical defect but we seeded 39 minor defects (40%) and 32 major defects (33%) within the whole software specification, mostly distributed in the document parts "*business functions*" (27 defects, 28% of all defects) and "*data description*" (24 defects, 25% of all defects).

Because the introduction contains 7 defects in sum, with 2, 3, 2 and 0 defects according to the severity levels trivial, minor, major and critical, there is a high defect detection rate concerning all severity levels with a small deviation. There are 37 seeded defects in the business function section (BUSI: 5, 17, 10 and 5 defects according to different severity classes). The defect detection rate for *trivial* and *critical* is quite higher (because of the small number of seeded defects) with respect to *minor* as well as *major* defects. Concerning the object model section, there are 5, 6, 9 and 5 seeded defects. The data show the best defect detection rate for *minor* and *major* defects. Finally, the *data description* area contains 28 seeded defects in sum (2, 13, 11 and 2 defects according to the severity classes *trivial*, *minor*, *major* and *critical*). Because there are only 2 defects with respect to trivial and critical errors, the defect detection rate is quite high, even if inspectors find at least one defect.

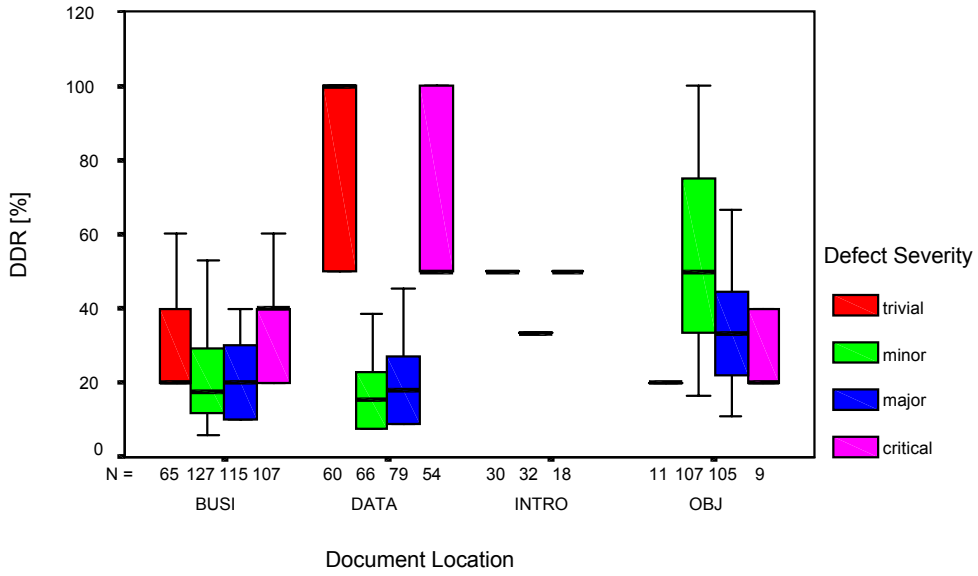


Figure 7.2-6: DDR: Defect Severity vs. Document Location

Table 7.2-7 show the detailed data of the number of defects detected during inspection. With respect to document locations and regardless of the distribution of seeded defects, the inspectors found at most 3.7 defects (BUSI), 2.5 defects (DATA), 1.3 defects (INTRO) and 3.3 defects (OBJ) in average. A more detailed view on different locations shows, that 36% of all critical defects have been found in the business chapter (but there is no separation between qualification and reading technique). 80 % of trivial defects in the data section, 55% of trivial and major defects in the introduction and 55% of minor defects in the object model have been

found. But there are only a few seeded defects in some document locations at defined severity class (this might be a reason for a quite high defect detection rate).

Defect Detection	BUSI Defect Severity				DATA Defect Severity				INTRO Defect Severity				OBJ Defect Severity			
	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3
Min	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1
Max	4	12	6	4	2	7	8	2	2	3	2	0	1	6	9	2
Mean	1,3	3,7	2,0	1,8	1,6	2,2	2,5	1,4	1,1	1,3	1,1	0,0	1,0	3,3	3,0	1,4
Std-Dev.	0,6	2,5	1,1	0,9	0,5	1,4	1,4	0,5	0,3	0,6	0,3	0,0	0,0	1,6	1,7	0,5
# SD	5	17	10	5	2	13	11	2	2	3	2	0	5	6	9	5
Av.DDR	26%	22%	20%	36%	80%	17%	23%	70%	55%	43%	55%	N/A	20%	55%	33%	28%

Table 7.2-7: DDR: Defect Severity vs. Document Location

Because reading techniques focus on different document locations, e.g. user-view focus primary on business-functions, etc. the defect detection rates might represent these facts. Figure 7.2-7 tries to visualise the defect detection rate according to document locations.

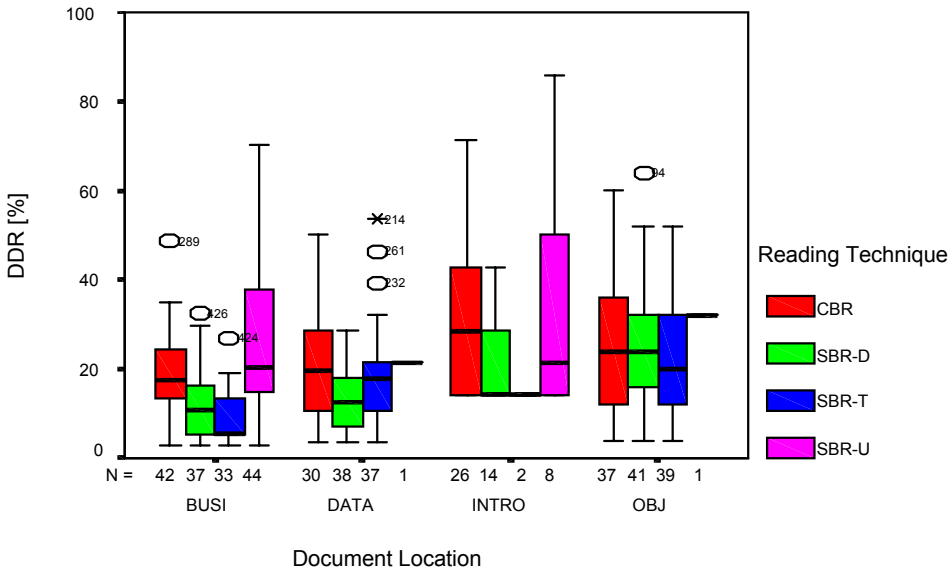


Figure 7.2-7: DDR: Reading Technique vs. Document Location

Checklist-based reading covers the whole document using pre-defined checklists. Therefore this approach supports defect detection in the whole document (Figure 7.2-7 confirm this theory). Scenario-Based reading focus on different points of view and support the defect detection process in only in the corresponding document parts. While *SBR-U* (user- view) focus primary on the introduction and the business-functions, the defect detection rate is quite

high. In contrast to the user-view, the tester-view (SBR-T) focuses on the technical parts (internal testing proceeding), of the document (i.e. *DATA*, *OBJ* and partly *BUSI).*

DDR	BUSI				DATA				OBJ				INTRO			
	CBR	SBR			CBR	SBR			CBR	SBR			CBR	SBR		
		D	T	U		D	T	U		D	T	U		D	T	U
Number	286	169	110	401	171	149	193	6	227	258	216	8	53	20	2	19
Min	1	1	1	1	1	1	1	6	1	1	1	8	1	1	1	1
Max	18	12	10	26	14	8	15	6	15	16	13	8	5	3	1	6
Mean	6,8	4,6	3,3	9,1	5,7	3,9	5,2	6,0	6,1	6,3	5,5	8,0	2,0	1,4	1,0	2,4
Std-Dev.	3,6	3,1	2,3	5,5	3,3	2,0	3,2	0,0	3,5	3,6	3,1	0,0	1,2	0,6	0,0	1,8
# SD	37	37	37	37	28	28	28	28	25	25	25	25	7	7	7	7
Av.DDR	18%	12%	9%	25%	20%	14%	19%	21%	24%	25%	22%	32%	29%	20%	14%	34%

Table 7.2-8: DDR: Reading Technique vs. Document Location

Because designers have to understand the user requirements (*INTRO, BUSI*) and to develop the product itself, he will find defects in the whole document also. Nevertheless the main point of interests cover the object oriented class model, where SBR-D found most defects.

Once, implementing an inspection process within a development team, project- and quality managers have to consider *inspector qualification levels*, *defect severity classes*, *document locations* and, finally, *reading techniques*, because every different approach focus on various attributes and the composition inspection environment must take care of this. Especially regarding reading-technique roles, inspection teams must consider document coverage and team composition.

During project proceeding, there is always a lack of time, because customers as well managers looking forward to achieve proper results in a cheap and less-time-consuming way. Once, establishing inspection processes, it is quite interesting, how defect detection rates change during inspection time.

7.3 Temporal Behaviour of Defect-Finding according to reading technique

During the inspection process the inspectors search for defects within a defined time-slot. The inspection time has been limited to at least 2h and at most about 6 hours. The finish of the inspection within this range of time is the choice of the inspector. If he has found all defects (in his own opinion) or if he will not be able to find more defects spending more time with inspection, he can finish at his own risk.

7.3.1 Inspection time according to reading technique and inspector qualification (Q3.1)

Because of the inspection proceeding, it will be quite interesting, how long it will take – subjective view of the inspectors – to find "all" defects within the specification document. Figure 7.3-1 describes the distribution of inspection duration according to individual reading techniques grouped by inspector qualification.

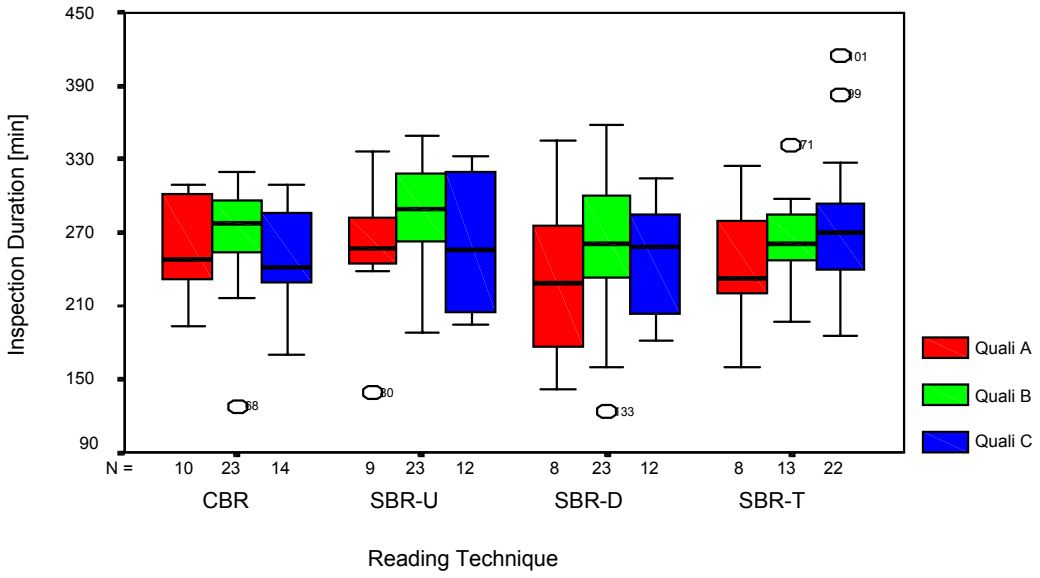


Figure 7.3-1: Timeline: Inspection duration

The overall mean duration is about 263 min (4h 23min). Concerning CBR reading-technique Level C, mean inspection duration is almost equal to qualification A but shorter than qualification B. Equivalent to CBR, SBR-U shows the same scenario. Qualification A and C are at the same level, while B is quite higher, i.e. a longer inspection time. The range of level C inspectors spread from about 200 min to 330 min, which is a greater range with respect to the other qualification levels and reading techniques. Looking at SBR-D level A inspectors inspect the shortest time but with a range from 150 min to 270 min. The Tester-Scenario shows a rating scale, like I expected before the evaluation, because qualification A inspectors inspect shortest, followed by level B and level C inspectors, each in about the same range of

data. This means, that lower qualified inspectors use more time to find all defects (concerning their individual view).

Table 7.3-1 presents the duration distribution results according to the reading-technique approaches and qualification levels.

Defect Detection	CBR Qualification			SBR-D Qualification			SBR-T Qualification			SBR-U Qualification		
	A	B	C	A	B	C	A	B	C	A	B	C
# Inspect.	10	23	14	8	23	12	8	13	22	9	23	12
Min	193	128	170	142	123	182	160	197	185	139	188	195
Max	310	320	310	345	358	314	325	342	415	336	349	332
Mean	254,7	273,3	249,6	231,3	260,3	248,9	243,9	263,2	272,9	260,2	282,7	263,2
Std-Dev.	41,15	42,38	40,90	67,02	56,97	44,94	50,51	38,40	53,68	57,44	44,60	54,59

Table 7.3-1: Timeline: Inspection duration

Concerning one reading technique approach, high-qualified inspectors need less time than inspectors, using other reading technique approaches. SBR-D inspectors work shortest, followed by SBR-T, CBR and SBR-U inspections.

7.3.2 Average Time to the first defect detection (Q3.2)

Because of the task-descriptions, which are used for pre-defined reading techniques, it will take some time to get familiar with the scenarios and checklists before searching for defects effectively. Some graphs and models have to be built in dependency to the scenario reading technique roles.

Another point to remember is the preparation time with the specification document because the inspectors first have to read it (the subject and the application area is completely new to all inspectors).

Remembering these basic facts, the distribution of time-slots until the first defect match (in this evaluation task I will cover noted defects so far). Figure 7.3-4 visualises the results of this evaluation task.

Starting again with CBR reading-technique, it will take about 30 minutes (in average) to read specification document to get familiar with the environment. The range of qualification A is quit larger (up to about 1h) for the first defect detection. SBR-U (user view) and SBR-D (designer view) starts first finding defects because they focus on introduction and business cases first, which are located at the first pages of the specification document.

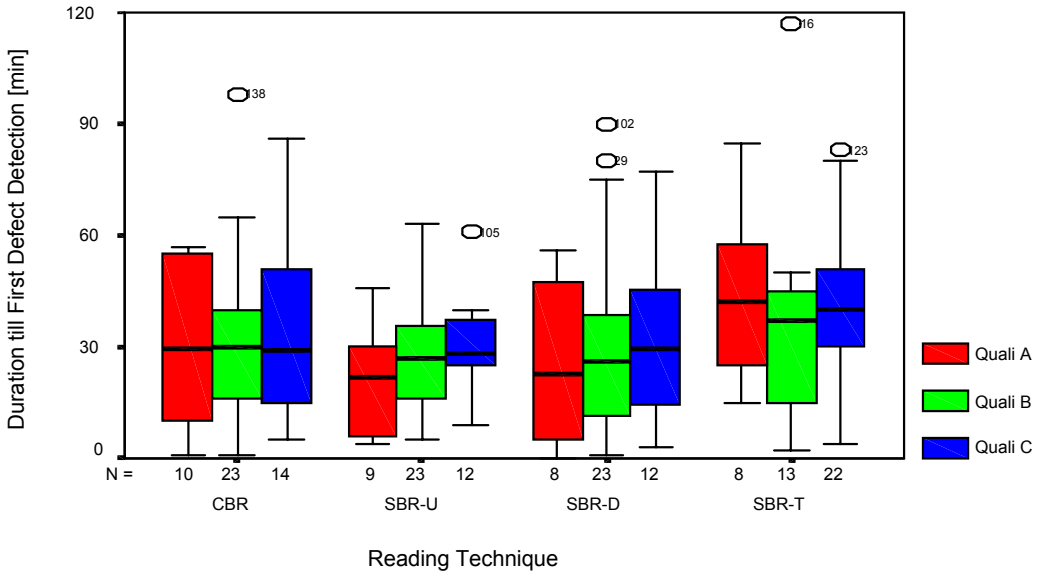


Figure 7.3-2: Timeline: Duration until the First Defect Detected

In contrast, SBR-T (tester view) finds defects after about 45 min (average value) at all qualification levels. One reason is the focus of the reading technique (object model and data description) and the inspectors have to get information of the environment (introduction and business cases), have to visualise their knowledge into models and can afterwards find defects in their assigned document parts. One interesting fact at SBR-U inspectors is the distribution at different qualification levels. Level B starts finding defects, followed by C and by A.

Defect Detection	CBR Qualification			SBR-D Qualification			SBR-T Qualification			SBR-U Qualification		
	A	B	C	A	B	C	A	B	C	A	B	C
# Inspect.	10	23	14	8	23	12	A	B	C	9	23	12
Min	1	1	5	0	1	3	8	13	22	4	5	9
Max	146	98	86	56	90	77	15	2	4	46	63	61
Mean	40,7	31,4	34,5	25,8	31,0	33,3	85	134	83	21,3	26,6	30,8
Std-Dev.	42,20	21,84	23,28	23,09	24,76	24,23	43,6	41,8	41,0	14,30	14,51	13,32

Table 7.3-2: Timeline: Duration until the First Defect Detected

Additional the results will depend on the reading technique approach of the individual inspector. Trying to analyse the real timeline of defect detection according to the reading approach finished without results, because inspectors don't use clear attributable approaches but some mixtures.

7.3.3 Average minutes between defect detection (Q3.3)

Another interesting evaluation task is the defect detection over time. Because there is a large amount of influence factors, covering many aspects of inspection parameters and data, I will choose a evaluation task, trying to find a relationship between inspection duration, qualification and reading techniques.

Therefore, I consider the inspection process as a “linear model”, starting at the beginning of inspection (without any pre-work) with a continuous defect detection rate until the end of inspection. This approach isn’t a realistic one because there is a need for pre-work (practice time) and there is no guarantee for a linear defect detection rate (reading technique approaches, “quality” of seeded defects, etc). But it is possible to find a relationship between overall inspection duration, qualification and reading technique approach.

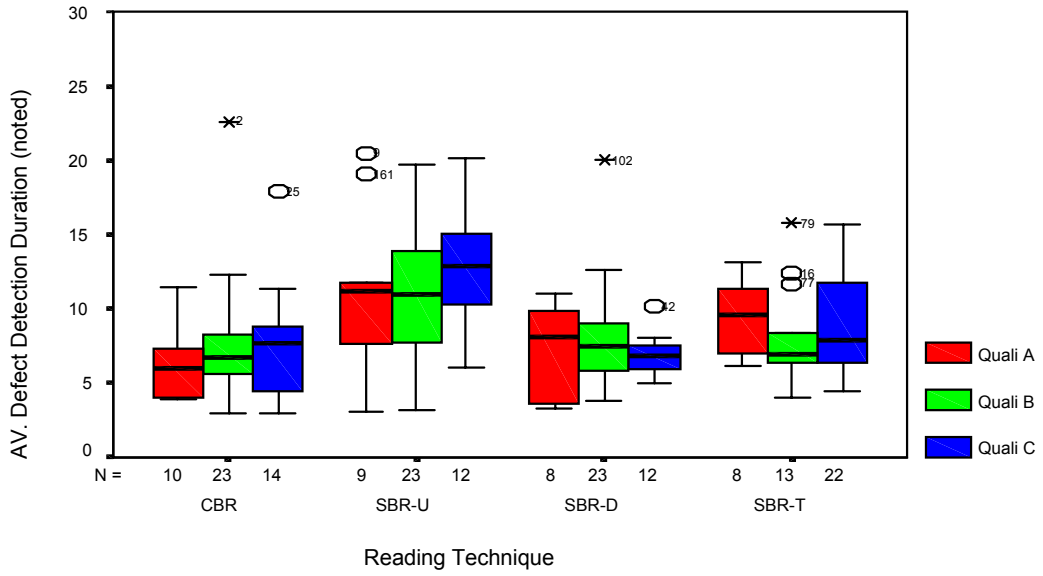


Figure 7.3-3: Timeline: Average minutes between Defect Detection (noted defects)

The top goal is a very short inspection-time including a large number of “real” defects, listed in the defect list, i.e. in this linear approach a very short time between two single defect detections. In general, I expect a short mean duration between the defect detections for high-qualified inspector and the longest duration concerning low-qualified inspectors. Figure 7.3-3 shows the results for noted defects.

In this model CBR inspectors (all qualification levels) find defects frequently, i.e. between 5 and 10 minutes per defect. As expected, level A inspectors are more successful that level B and level C inspectors. Concerning SBR-U the ranking of qualification levels doesn’t change, but the frequency of defect detection grows to about 10 to 15 minutes (average value). Defect detection frequency, using SBR-D and SBR-T reading techniques are in the range of 5 to 10 minutes; i.e. quite higher than SBR-U but lower than CBR. Another distinctive feature is the change of the qualification level ranking. It takes longer for high-qualified inspectors than for lower qualified inspectors. Analysing these effects will result a wide range of investigation.

Defect Detection	CBR Qualification			SBR-D Qualification			SBR-T Qualification			SBR-U Qualification		
	A	B	C	A	B	C	A	B	C	A	B	C
# Inspect.	10	23	14	8	23	12	8	13	22	9	23	12
Min	4	3	3	3	4	5	6	4	4	3	3	6
Max	11	23	18	11	20	10	13	16	33	21	36	20
Mean	6,3	7,8	7,6	7,2	8,0	6,9	9,4	8,1	9,8	10,9	12,0	12,8
Std-Dev.	2,52	4,05	3,83	3,32	3,38	1,40	2,54	3,28	6,00	5,90	6,63	3,84

Table 7.3-3: Timeline: Average minutes between Defect Detection (noted defects)

One point of view is the number of defects and the frequency of defect detection for all defects, found by the inspectors. The next section presents the results for the matched defects in a similar way.

First of all, I expect a lower frequency of defect detection, i.e. an increasing duration for finding the next matched defect.

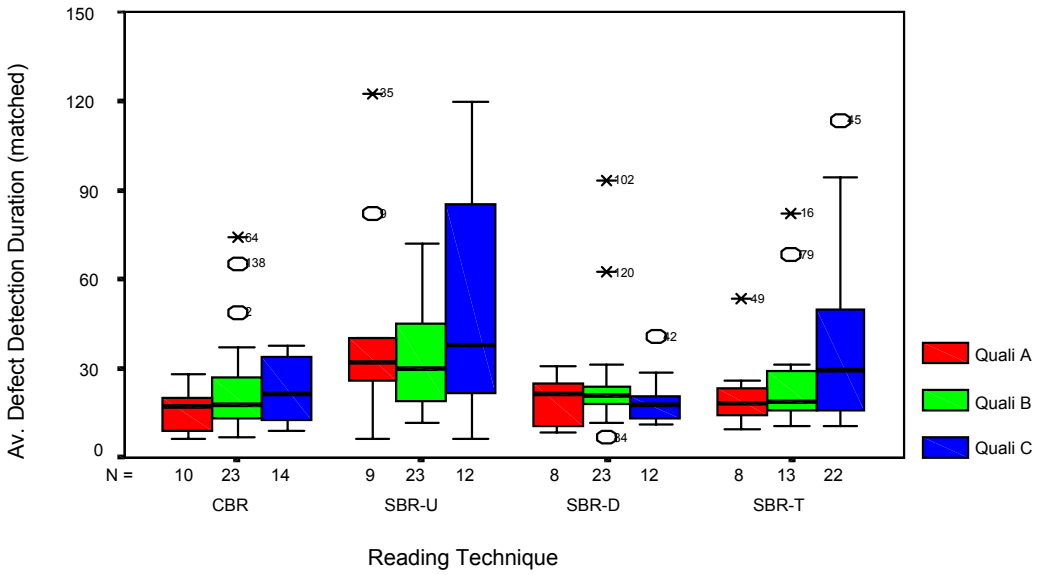


Figure 7.3-4: Timeline: Average minutes between Defect Detection (matched defects)

In summary, the frequency of defect findings increases to about 15 until 50 minutes (SBR-U, level C even to 90 minutes). The basic ranking of frequencies doesn't change very much, but there are some interesting facts. Less qualified inspectors need quite longer (e.g. SBR-U, SBR-T, qualification level C) to find matched defects. In contrast, high-qualified inspectors improve their defect detection frequency (e.g. CBR, SBR-T) but need – of course – more time as well.

Defect Detection	CBR Qualification			SBR-D Qualification			SBR-T Qualification			SBR-U Qualification		
	A	B	C	A	B	C	A	B	C	A	B	C
# Inspect.	10	23	14	8	23	12	8	13	22	9	23	12
Min	6	7	9	8	7	11	9	10	11	7	12	6
Max	28	74	286	31	93	41	53	82	328	123	330	205
Mean	16,2	24,0	55,2	19,2	24,8	19,2	21,8	28,1	61,6	44,1	57,2	59,2
Std-Dev.	6,68	17,33	88,00	8,39	18,19	8,30	13,82	21,96	81,61	35,86	86,18	57,26

Table 7.3-4: Timeline: Average minutes between Defect Detection (matched defects)

Additional to inspector qualification, reading-technique, defect severity levels, which are described in the previous sections in a short way, the usage and acceptance of inspections in general and the assigned reading techniques is important for the practical implementation of inspection techniques within a software project.

7.4 Acceptance of Inspections

Methods and Tools must be easy to handle and achieve proper results to establish techniques like software inspection within a project team. Another fact is the acceptance of tools and methods, because if the “user” doesn’t accept them, it will become more difficult to implement it permanently.

In the experiment we try to get feedback to different topics of interest in the area of software inspection. In this thesis I will point out inspection *acceptance* of reading techniques approaches as examples.

All inspectors have to answer a feedback questionnaire after each inspection activity (i.e. the inspection). The following sections describe the results with respect to reading technique roles.

7.4.1 Applicability of Reading Techniques (Q4.1)

We implemented a questionnaire (FB01) to achieve subjective estimations of the inspectors about the difficulty of applicability of reading technique and task descriptions.

We determine, that the scenario / checklist, assigned to the reading technique is clear, understandable and easy to use. Therefore we ask the inspectors to rate this statement in a range of 0 (completely disagree) to 4 (full agreement). Figure 7.4-1 presents an overview of reading-technique role and the assessment of RT acceptance.

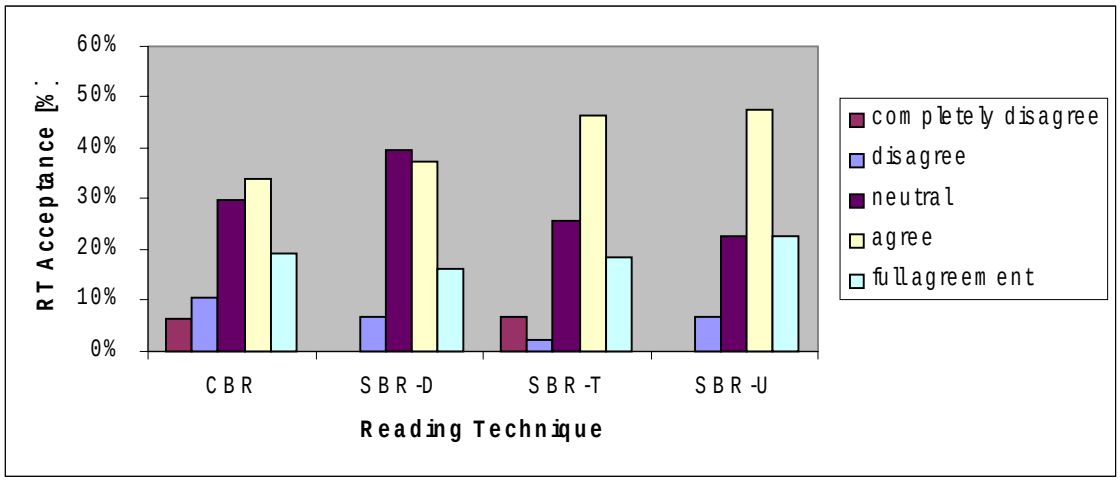


Figure 7.4-1: Feedback: RT Acceptance according to Reading Technique

Concerning individual inspector reading technique, most of the inspectors declare this inspection approach as clear, understandable and easy to use, i.e. a neutral rating or even better. Lower than 10% of inspectors (at all reading techniques) had problems with the reading technique, so they considered the method as completely useless or even less useful.

Between 15 and 25% of the inspectors rates, that reading techniques support the defect detection process very well (full agreement) and about a third and more of all inspectors rate this question positive, i.e. it helps defect finding. About 25 to 40% doesn't see whether advantages nor disadvantages (i.e. neutral rating). Looking SBR reading technique approaches, the rating is very high, concerning neutral and agreement (each above 30%), and highest at SBR-U (nearly 50% of SBR-U inspectors). The number of inspectors for "disagree", "completely disagree" and "full agreement" is at about the same level with respect to all reading techniques. Table 7.4-1 shows the distribution of acceptance rating in more detail.

RT Acceptance	CBR		SBR-D		SBR-T		SBR-U		Summary	
	[no.]	[%]	[no.]	[%]	[no.]	[%]	[no.]	[%]	[no.]	[%]
Completely disagree	3	6%	0	0%	3	7%	0	0%	6	3%
Disagree	5	11%	3	7%	1	2%	3	7%	12	7%
Neutral	14	30%	17	40%	11	26%	10	23%	52	29%
Agreement	16	34%	16	37%	20	47%	21	48%	73	41%
Full agreement	9	19%	7	16%	8	19%	10	23%	34	19%
Total:	47	100%	43	100%	43	100%	44	100%	177	100%

Table 7.4-1: Feedback: RT Acceptance according to Reading Technique

In summary, reading technique and roles achieve a high acceptance in relation to all inspectors, especially SBR-reading technique approaches. It will be quite interesting, how the

acceptance of reading technique distributes with respect to inspector qualification. Figure 7.4-2 presents the results in a short summary.

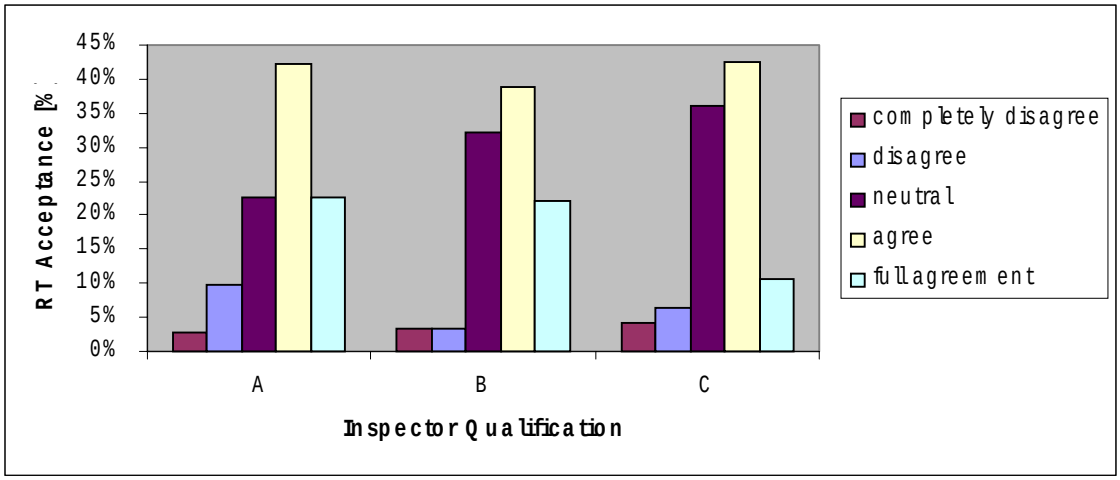


Figure 7.4-2: Feedback: RT Acceptance according to Inspector Qualification

Concerning full agreement, the acceptance of reading technique is quite high (about 20%) of qualification A and B and only at 10% of qualification C. The acceptance rating for "agreement" is quite high for all qualification levels (about 40% each). The neutral rating scale increases from about 20% (qualification A) to about 35% (qualification C). Additional the number of high-qualified inspectors (qualification rating A) the disagreement to the acceptance of reading technique is about at 10%.

RT Acceptance	Quali A		Quali B		Quali C		Summary	
	[no.]	[%]	[no.]	[%]	[no.]	[%]	[no.]	[%]
Completely disagree	2	3%	2	3%	2	4%	6	3%
Disagree	7	10%	2	3%	3	6%	12	7%
Neutral	16	23%	19	32%	17	36%	52	29%
Agreement	30	42%	23	39%	20	43%	73	41%
Full agreement	16	23%	13	22%	5	11%	34	19%
Total:	71	100%	59	100%	47	100%	177	100%

Table 7.4-2: Feedback: RT Acceptance according to Reading Technique

7.5 Chapter Summary

Project and quality manager, who want to set up an inspection technique, have to consider several topics to implement software inspection within a project team or even in a software development department. He has to include *inspector qualification* to set up an inspection team (chapter 7.1). Because of the influence of qualification, I introduced three models for qualification assessment. After selecting a most realistic (and realizable) approach, the self-

estimation qualification, I showed *defect detection* (chapter 7.2) rates according to inspector qualification, reading technique, defect severity and defect location. Because may inspectors are involved in the inspection process, it is quite important, considering the temporal behaviour of defect finding processes. I tried to give an overview of inspection duration and temporal defect density (i.e. the duration between two defects detected by the inspectors) in chapter (chapter 7.3). Because of the dependencies of software inspections to humans, the acceptance, usability is quite important. I covered this approach in a short way in chapter (chapter 7.4).

Every section presents an overview, but there will be a large number of interesting further investigations to achieve results in more detail and to find underlying dependencies. Nevertheless these investigations will go beyond the scope of this thesis.

8 Discussion

This chapter shows the results with respect to the research questions (chapter 6.1), defines some evaluation problems with data and the specific environment, concerning dependent and independent variables, and show possible starting points for further investigations.

8.1 Investigation of Inspector Qualification

Because software inspections depend on the participants at a very high level, the inspection team must be well qualified to achieve the goal of defect detection.

In the evaluation phase of the experiment, I used three types of inspector qualification, the supervisor qualification assessment (*SE-Skills*), a self-estimation qualification model (*EXP*) and a pre-test (SI-DD) in the area of software inspection.

8.1.1 Supervisor Qualification Assessment (SD-Skills)

Before the experiment starts at all, the participants had to implement a small software product using state-of the art techniques in the area of software engineering to get familiar with the topic “software engineering” itself and to practice their skills. Because this qualification test is a precondition for the software engineering course, the inspectors have to finish these tests in positive way (the inspection experiment has been embedded within a software engineering course at university level). The lab assistants (inspection leaders) assessed the software product according to functionality, user interface design and understanding of product and software engineering methods and tools. According to the results of this qualification test, the inspectors have been classified to qualification levels (“A” for a high qualification, “B” for medium qualification and “C” for low qualification). Therefore, software engineering is the main focus of this qualification assessment.

The usage of supervisor assessment (assessing a software product, i.e. a program, with focus on SE-Skills) suffer from different points of critics:

- Subjectivity of Supervisors: As humans assess the software product, there is a subjective view on the software product. It will be quite strange within a practical development environment to rate products of different team members in some kind of “school ratings”. Nevertheless lab-assistants use checklists and proceedings to rate the programs in a defined way to achieve comparable and objective results.
  Another possibility to solve this problem may be some kind of external assessment. External experts (no team members) can assess and rate the program in a more objective way.
- Focus on software engineering Skills: The qualification test focuses on software engineering skills rather than on software inspection skills. Therefore it is quite difficult to get a proper rating for inspector qualification because there is no guarantee to perform an inspection at the same high level.

On the other side inspection processes contain a detailed set of instructions to perform inspection. Therefore it is quite more important for the inspectors to get familiar with SE-techniques rather than inspection techniques.

- Project managers are not able to perform some kind of tests in a practical environment, but he has to use other methods to investigate inspector qualification.

In summary, project managers will not achieve realistic qualification rating assessing software programs out of the history. They have to find some other methods to assess inspector qualification, e.g. self-estimation (*EXP*)

8.1.2 Self-Estimation Qualification Assessment (EXP, Q1.1)

To improve the weakness of supervisor qualification assessment, I used a qualification model, which is based on a feedback questionnaire, concerning the inspectors' subjective experiences.

This qualification rating covers *general skills* (e.g. reading capability with respect to technical documents), *software engineering skills* (e.g. knowledge of UML), *inspection skills* (e.g. practical usage of software inspections in the past) and *project experience* (e.g. project participation). Different questions have been assigned to each area of interest, equipped with a weight, which defines the importance for software inspection) and finally accumulated to one qualification rating.

Concerning research question, I will expect a realistic view on inspector qualification because of an extensive questionnaire and a defined application context. As the results show, there exist some points to remember, when introducing the self-estimation model.

- Over- and under-estimation of inspector: Although there is unreliability in the self-estimation because some inspectors will over-estimate their knowledge, others will under-estimate their skills we expect a more realistic view in sum.
  Therefore this approach, including the weight of individual qualification factors will allow a quite good rating without much additional waste of time and money.

- Completeness of Experience questionnaire: The number and main focus of questions varies in the range of application, projects, resources, etc. Because of the individual project-requirements, managers have to use different configurations for qualification ratings to get useful results.

- Project managers have to install a proper questionnaire according to the project environment and can use the results to find the best team composition with respect to qualification levels. This kind of questionnaire can also be used for project team assembling.

- Non-anonymous questionnaires: Because of our structure and the handling within the software engineering course the questionnaire is a non-anonymous one. This might

result in an overestimation of the inspectors (i.e. the estimate their skills at higher qualification level to achieve a better certification of the course – but this ranking of skills doesn't influence the certification at all.

- Partial experience: Some questions cover wide ranges of experiences. For example, one qualification factor within the topic project experience is the number of projects an inspector participated in the past. Usually projects are segmented into different phases and an inspector will participate only in a subset of phases, achieving different skills within this phase. We didn't survey this segmentation.

In summary, this qualification rating is quite useful, if there is agreement between the questionnaire and the project environment.

8.1.3 PRE-Test Qualification Assessment (SI-DD, Q1.2)

Using inspection data based on real inspections in the past is the most realistic view on inspector qualification. Because this thesis focus on the first implementation of inspection within a project team or even a company, there exist no data of previous inspections. Therefore we use a training session to introduce the inspection techniques and get a view on inspector qualification and capability.

Research question *Q1.2* covers the evaluation of SI-DD. Using SI-DD, I will expect the most realistic view on inspector qualification because the pre-test focus an equal application area and requires inspection skills as well as software engineering skills. As the results show, there exist some points to remember, when introducing a pre-test.

- Real inspection results: SI-DD covers real inspection results, concerning application area as well as necessary skills for project proceeding and real defects. Therefore, SI-DD will be the best solution for qualification assessment.
- Application area: Because the pre-test covers only one defined application area, the qualification rating corresponds to this area (or maybe similar application areas) only. Considering the software engineering approach, results of pre-tests must consider similar proceedings (e.g. software process models, notations, etc.).
- Feasibility of pre-tests: The implementation of a pre-test (for inspection purposes) might be difficult because of a lack of resources and, additionally, it will be some kind of examination within a project team. Further, the environment has to be adapted to the "real" project as well. But a pre-test should be implemented as preparatory training sessions.

In summary, this qualification rating is the most realistic one, if there is agreement between the pre-test and the project environment, but it will suffer from acceptance because of their nature of examination. It's difficult to set up a training session in every project.

8.1.4 Comparison of Qualification Assessment (Q1.3)

The main goal of the different qualification assessment models is the selection of a most realistic view on inspector capability and a simple evaluation model in a practical environment. I investigated the different approaches and compared them to each other in the results chapter.

According to research question, I compare all three models and I expect an increasing number of lower qualified inspectors because of the scope of qualification models. While *SE-Skills* focus on software engineering skills (i.e. the main topic of the course), *EXP* focuses on project experience and *SI-DD* addresses additional inspection skills.

Figure 7.1-7 shows the results of this evaluation task. The distribution of inspector qualification corresponds to the expectations. There are more lower-qualified inspectors concerning EXP and SI-DD qualification models. A closer look at the results (concerning the number of inspectors) show more differences between qualification evaluation models at level "B" rather than at level "A". In summary, the results agree to the research question.

8.2 Defect Detection Rate (DDR)

The main purpose of inspection is the reduction of defects in software artefacts, in our case a software specification for a large-scale software product. Defect detection depends on many environmental factors, e.g. inspector qualification, reading techniques, application area, etc. The scope of this thesis covers different defect severity classes and document location.

8.2.1 Defect Detection and Inspector Qualification (Q2.1)

The first evaluation task covers inspector qualification and reading technique with respect to the defect detection rate of matched defects. Following this approach, I expect a higher defect detection rate, concerning higher qualified inspectors. High-qualified inspectors found 42% of all matched defects, medium qualified inspectors found 32% and lower qualified inspectors found 26% of all defects. *The hypothesis, high qualified inspectors will find most defects, applies*.

I also assume, that CBR readers will find more defects because of the straight forward proceeding and the quite simple approach of this reading technique. Concerning qualification level "A", CBR inspectors find more defects than scenario-based reading technique approaches (SBR-D, SBR-U and SBR-T in order of their defect detection rate at qualification A). The medium qualification level shows a similar distribution of defect detection rates, i.e. inspector using CBR reading techniques find most defects, followed by SBR-D, SBR-U and finally SBR-T. The results of low qualified inspectors differ, because SBR-T inspectors find most defects followed by SBR-D, CBR and SBR-U inspectors. *Therefore the hypothesis, CBR inspectors will find most defects at every defect severity level, doesn't fit.*

Every seeded defect is equipped with one defined severity level. We distinguish *trivial*, *minor*, *major* and *critical* defects. The distribution of 97 seeded defects has already been described in chapter 6.2.1.2. In a short summary, we seeded 14 trivial defects, 39 minor defects, 32 major defects and 12 critical defects.

High-qualified inspectors will find more defects, independent of their severity level, but with special focus on major and critical defects. Analysing the data, low-qualified inspectors find most of the *trivial defects*. Concerning minor defects, high- and medium qualified inspectors find – in average – about 16% of all seeded minor defects within the document. Concerning major defects, the low-qualified inspectors achieve the highest defect detection rate (18%) and, finally, concerning critical defect, medium and low-qualified inspectors achieve 18% as well, but the defect detection rate of high-qualified inspectors is quite lower.

8.2.2 Defect Detection and Reading Technique Roles (Q2.2)

Every inspector uses one predefined reading technique approach (checklist-based reading (*CBR*) or scenario-based reading approaches (*SBR*)). Concerning scenario-based reading techniques, we use different points of view to inspect the specification document: user-view (SBR-U), designer-view (SBR-D) and tester-view (SBR-T). All reading technique approaches are equipped with a task-description, which can be considered as guidelines to perform the defect detection. Because reading techniques as well task descriptions focus on different document locations, defect types, etc. I expect a clear representation of the defects detection rates according to their individual focus.

CBR inspectors will find more defects, independent of their individual qualification level because CBR covers the whole document. As the results (chapter 7.2.2) show, the overall defect detection rate of CBR inspectors is about 32%, followed by SBR-D inspectors (26%), testers (23%) and user (19%).

Concerning the different qualification levels, there the results differ with respect to the overall defect detection rate. High- and medium-qualified inspectors found most defects using the CBR reaching technique approach, but concerning low-qualified inspectors are more successful using SBR-D reading techniques. *In summary, the hypothesis fits to high- and medium-qualified inspectors and doesn't fit to low-qualified inspectors.*

SBR reading technique approaches focus on different defect types, i.e. more severe defect. Therefore I expect a higher defect detection rate using SBR-reading techniques. Furthermore, the SBR reading technique approach covers pre-defined document locations, e.g. the object model or data description. Because of this specialization, I expect a higher defect detection rate at more severe defects. Analysing critical defects (severity level 3) SBR-U inspectors found 20% of all seeded defects (regarding the number of critical defects in the specification document), followed by SBR-T and CBR inspectors (18% each) and SBR-T inspectors (15%). Concerning major defects (severity level 2), SBR-T inspectors detected most defects (18%), followed by SBR-D and CBR (16% each) and SBR-U (only 8%). Furthermore, CBR inspectors found most minor (19%) and trivial (15%) defects. Because CBR inspectors read

the whole document and SBR-inspectors focus on different defect types, the defect detection rate at CBR is at high level, but they do not find specific defects. Specific defects can be found by the corresponding SBR reading technique approaches with a higher defect detection rate.

8.2.3 Defect Detection and Document Locations (Q2.3)

As described before, defects has been seeded at different document locations, i.e. the *introduction*, *business functions*, *object models* and *data description*. The document locations (chapters) are not seeded with defects in a uniform way (neither the number of defects nor the severity of defects). Therefore, I have to consider the defect detection rate according to the overall number of defects, seeded in the document locations. CBR reading techniques cover the whole specification document and SBR reading technique focus, according to the corresponding task-description, to specific parts of the document.

I expect a clear representation of the defect detection rate according to the document location. E.g. a SBR-D inspector will find most defects in the chapter "object model" and "data description" but less defects in the chapter, covering "business functions". Concerning CBR inspectors, who cover the whole document, achieve an overall high defect detection rate (29% INTRO, 18% BUSI, 24% OBJ and 29% DATA). The inspectors, using the designer-view, find 25% OBJ defects (which is his main part), 20% INTRO, 14% DATA and 12% BUSI. SBR-T inspectors focus on OBJ (22%), INTRO (20%), DATA (19%) and BUSI (only 9%). Finally, SBR-U inspectors find 34% (INTRO), 32% (OBJ), 25% (BUSI) and 21% DATA defects. Following these results, the individual inspectors focus on different document locations and find a corresponding number of defects.

Concerning qualification, I expect, that higher qualified inspectors will find more defects in every document location, independent of their reading technique approach. The results present a higher defect detection rate, according to my assumption, for high-qualified inspectors, concerning the *business functions* (19%), *object model* (26%) and *data description* (19%). But concerning the introduction low-qualified inspectors found about 29% of all seeded defects. The reason might be the structure of the introduction, which is very simple (all other inspectors achieved a high defect detection rate as well).

8.3 Temporal Behaviour of Defect Detection

Software inspection requires resources, like inspectors (usually organized in an inspection team) and inspection time, which corresponds to the type of the inspection document, the reading techniques, used during inspection and, in some cases a team meeting for a discussion. There exist several papers, covering aspect of team-meeting, benefits and possible problems establishing a team meeting [16][25][28].

Another approach according to inspection duration (in summary) is the decision of establishing a second inspection cycle (a so-called re-inspection). A second inspection cycle

may be necessary, if the specification document still contains a lot of defects. Concerning the re-inspection there also exist a wide range of publications [10][12][14].

In this thesis I focus on one inspection cycle without a team meeting. The basic guideline for the duration of inspection has been between 2h at least and about 6h at most. The finish of the inspection within this time-range has been the choice of the inspector. If he has found all defects (in his opinion) or if he will find no further defects spending more time with inspection, he can finish at his own risk.

8.3.1 Inspection Duration (Q3.1)

Concerning the inspection duration within the pre-defined time-range is quite interesting to find dependencies between defect detection rates and inspection duration. The main goal of inspection, in the view of project- and quality managers is a high defect detection rate at a very short time, to save time and money.

I expect a shorter average duration, concerning SBR inspectors because they cover segments of the specification document and will finish earlier, with respect to CBR inspectors, who have to search the whole document. In this evaluation task, I will not include defect detection rate at all. Concerning the average duration of four different reading technique approaches, SBR-U inspectors search 272 min, SBR-T 264 min, CBR 262 min and SBR-D 251 min for defects. The results show, that CBR inspectors need more time than SBR-D inspectors but less time than SBR-T and SBR-U inspectors. Regarding the hypothesis, there is no clear agreement or disagreement, so there must be some more investigation.

With respect to qualification levels, I expect a shorter duration for high-qualified inspectors, rather than for low-qualified inspectors. Table 7.3-1 shows, that high-qualified inspectors (qualification level a) need less time to finish inspection, concerning SBR-U, SBR-D and SBR-T reading technique approach. With respect to CBR reading technique, the low-qualified inspectors (qualification level C) need less time to inspect. But there is no strict ranking according to the qualification levels within one reading technique.

8.3.2 Average Inspection Time to the first Defect Detection (Q3.2)

Concerning the SBR reading technique, inspectors have to finish pre-work, e.g. creating models, before starting the defect detection process. Following those instructions, SBR inspectors will start later and, therefore, the duration until the first defect match will be later as well.

I expect the average duration from the beginning of inspection to the first defect detection, regarding noted defects only, will be shorter using the CBR, and longer at SBR reading technique approaches. After evaluation, the results present a completely different view. Concerning high-qualified inspectors, SBR-U starts after 21min with inspection, i.e. the first defect found, followed by SBR-D, CBR and finally, SBR-T. Independent of all qualification levels, SBR-U starts first and SBR-T starts at least. In composite to my expectation, the CBR

inspection technique need longer preparation time until the first defect detection, followed be SBR-T users.

8.3.3 Average Inspection Time between Defect Detections (Q3.3)

This evaluation task covers the average minutes between two defects found during inspection in a linear way, ignoring task-specific delay (concerning matched and noted defects). Because of the scope of reading techniques, SBR inspectors will take longer to find defects (and matched defects) in contrast to CBR inspectors. Additional SBR techniques focus on a more concrete model (they have to construct it) while CBR inspectors use checklists during the reading process.

Therefore I expect a shorter duration (in mean) between two defects using CBR reading technique approaches. Concerning noted defects first, the average duration between two defect detections (in this linear model), is smallest at CBR (7.4 min), followed by SBR-D (7.6 min), SBR-T (9.2 min) and finally, SBR-A (12 min). Following this approach, the results fit to my expectations.

Higher qualified inspector will find more defects in a shorter duration, i.e. the duration between two defect findings will be smaller with respect to lower-qualified inspectors. The results show an agreement to this hypothesis, regarding CBR and SBR-U reading techniques but don't agree with SBR-D (C qualification) and SBR-T (B qualification).

Analysing the average inspection time between two defect findings with respect to matched defects, will show the same facts, but – of course – the average duration for finding matched defects are quite longer (at least 16 min using a CBR reading technique approach (qualification A) and at most 61 min using a SBR-T reading technique approach (inspector qualification C). Therefore, the results fit to my expectation.

8.4 Acceptance of Inspections

8.4.1 Usability of Reading Techniques (Q4.1)

The usability and simplicity of tool or method is a pre-condition for the acceptance and the successful implementation within a practical environment, i.e. within a project or even company-wide. A project or quality manager is interested in the applicability and usability of inspection and reading-techniques at the starting point of establishing inspection processes in a practical environment.

To investigate the usability and as a consequence, the acceptance of inspections and reading techniques, we introduced a questionnaire after inspection cycle. The inspectors have to answer those feedback questionnaires after each inspection activity.

We determine, that the scenario / checklist, assigned to the reading technique is clear, understandable and easy to use. Therefore, we ask the inspectors to rate this statement in a range of o (completely disagree) to 4 (full agreement).

Concerning reading techniques, I expect a greater acceptance at SBR reading techniques because the method is more clearly arranged and focus on specific parts only. Most of the CBR inspectors (34%) agree to this statement, 30% rate this statement in a neutral way. Concerning SBR-D reading technique approach, 40% of all inspectors rate neutral and 37% agree. Analysing the results of SBR-T (47% agreement) and SBR-U (48% agreement) inspectors, the statement has been rated in a very positive way. In summary, inspection using reading technique approaches, are quite useful to support defect detection and easy to handle. Furthermore, the acceptance of scenario-based inspector is quite higher with respect to checklist-based inspectors.

Due to strict guidelines, lower qualified inspectors will use the checklist more frequently. Therefore I expect a higher acceptance by lower qualified inspectors. The results shows an agreement (or even fully agreement) of low-qualified inspectors (53%), followed by medium qualified inspectors (61%) and high-qualified inspectors (64%). Following these results, inspection techniques and reading-technique approaches, as support-methods for defect detection, are more important for high-qualified inspectors than for low qualified inspectors. Nevertheless the acceptance of inspections and reading techniques are quite high.

9 Summary and Outlook

This thesis addresses project-managers as well quality managers and engineers in the area of software engineering, who want to improve their project proceeding and quality of their products. Software production must grow as an engineering discipline, using structured approaches during the whole software production process.

In chapter 2 “Approaches in Software Engineering” I introduce the so-called life-cycle model, as a basic proceeding for software development, concerning the different phases form the first idea to the retirement of the software product. Furthermore, I introduce some important and basic software development processes, like the waterfall-model, V-model, etc. Every model focus on different scopes, i.e. application area, project size, duration, etc. The choice of the best solution is the task of the project-manager in co-operation with the project team and – if required – with the customer.

Chapter 3 “Approaches in Quality Management” focus on wide spread quality management systems, like the ISO 9000 series and CMM, discussing advantages, disadvantages and details of the internal structure. To improve product-, process- and management system quality, I summarise important quality management tools and methods, like audits and assessments, reviews and inspections.

One definition of quality refers to the fulfilment of requirements and customer satisfaction. If there are no or even less defects in the product it is associated with “high quality”. Because the correction of defects is quite expensive in time, resources and money, it is necessary to fix the problems as soon as possible. The cost for repair increases in the timeline of software development rapidly. Because there is a wide range of methods and tools in the area of quality management and software engineering, one big challenge is the right choice of good, well-defined models. One approach of quality improvement method is the software inspection. Inspections focus on defect detection in early phases in software development. Therefore, the main goal of software inspection is the defect detection as early as possible, i.e. in the specification phase. I describe the inspection process in chapter 4.

It’s the project or quality manager’s task to implement an inspection process as a useful tool for defect detection. One possibility to evaluate tools and methods are experiments in the area of software engineering, the so-called empirical software engineering approach. Chapter 5 gives a rough overview about empirical software engineering with special focus on software engineering experiments.

To evaluate a software inspection using reading technique approaches we set up an experiment at the *Institute of Software Engineering and Interactive Systems* at *Vienna University of Technology* during a software engineering workshop. In this experiment we use a specification document for a well-know administrative system (a ticket-selling-system). This software specification has been seeded with defects at different document locations, i.e. the *introduction*, *business-functions*, *object model* and *data description*. Every defect is equipped with one pre-defined severity level (*trivial*, *minor*, *major* and *critical defect*). The inspectors provide information about their skills in the area of software engineering, project-

experience, etc. I use this feedback questionnaire to calculate a qualification model to investigate the influence of inspector qualification to defect detection rates.

We also provide some guidelines for the defect detection process, i.e. checklists and scenario-guidelines according to pre-defined reading techniques. Reading techniques approaches help the inspector to find defects within the document. We support checklist-based reading (CBR), and three scenario-based reading approaches, regarding different view (the user-, tester-, and designer view). I describe the experiment description in chapter 6.2. Once the collection of defects is finished, we set up a data analysis process, which prepares the data for further investigations. This process is a general approach for data evaluation processes, using a base-data model and task-descriptions for individual purposes. The proceeding of this evaluation process is described in chapter 6.

This thesis covers four subjects of software inspections:

- Investigation of Inspector Qualification: In this section I use (and design) models for inspector qualification assessment using feedback data and a so-called pre-test (training session for the inspection process). After evaluation of the qualification models I choose the best (most realistic) qualification model for further investigation. Therefore I use the so-called self-estimation (*EXP*), which is based on feedback data.
- Defect Detection Rates: Because the main goal of inspection is a most effective defect detection rate, I investigate inspector capability (i.e. inspector qualification), the influence of reading technique roles and the effectiveness of inspectors with respect to defect detection rates.
- Temporal behaviour of defect detection rates: Because of a lack of resources, project managers need a high efficient solution, requiring less time and inspectors, regarding a very high defect detection rate. Therefore I investigate the temporal behaviour of defect detection according to the overall inspection duration, the time-delay to the first defect detected and the average duration between two defect matches with respect to inspector qualification and reading technique.
- Acceptance of Inspections: If a method or tool is easy to use, simple to understand and helpful finding defects, it will be accepted within a project team or even a whole company. Therefore I investigate the feedback questionnaire according to these parameters to find out, if inspections fulfil these requirements.

Inspector Qualification

I choose the self-estimation qualification model (*EXP*) because it will be easy to implement into a software development team using simple questionnaires. It's the project manager's task to set up a useful questionnaire according to the application area. Other qualification rating models (e.g. qualification test, pre-test) are not realisable, because of their need of resources. Furthermore is it quite difficult to establish a qualification test with a company, using some kind of school ratings because of the acceptance of the team members.

Although there are some points of critics using the self-estimation approach: inspectors might over- or underestimate their skills. Because of the distribution of answers, according to the questionnaire, I expect a more realistic view in sum.

The main focus and the number of questions depend on the range of application, projects, resources, etc. Because of individual project-requirements, managers have to use different configurations for qualification ratings to get useful results. Therefore, the project-managers have to configure a well-defined questionnaire according to the project environment. Additionally they can use the results for a team composition.

Defect Detection Rates

The defects, equipped with defined severity levels, have been seeded within the specification document at different locations. Because the reading technique focus on specific defect types and document locations, the defect detection rate differs according to the reading technique approach. While CBR reading techniques cover the whole document, SBR-D focus on the object model, SBR-T includes object model and the intro and SBR-U covers business-functions and data description as well.

Once, installing an inspection process, managers must recognise the focus of the different reading techniques for overall document coverage.

Temporal Behaviour of Defect Detection

Concerning inspection duration, high-qualified inspectors need less time to inspect the specification document. Additionally, it depends on the reading technique, and the design of the guidelines, to achieve proper results.

Following the results, the preparation time of CBR inspectors is quite longer than individual inspectors, using SBR-U and SBR-D reading technique approaches. SBR-T need more than CBR to prepare for inspection. In summary, the usage of reading techniques doesn’t influence temporal behaviour of defect detection.

Acceptance of Inspection

Due to strict guidelines the proceeding of inspection is well defined, and – even lower qualified inspectors – will achieve proper results. The investigation show, that high-experiences inspectors prefer guidelines more than lower qualified inspectors. The acceptance of reading technique approaches and their corresponding guidelines depend on the design of those documents and must be adapted to the application area and the project team environment.

Thus, the acceptance of the inspection method itself and the reading technique approaches as well is very high in sum.

Some ideas for further investigations

- Implementation of quality management system with respect to quality software engineering.
- Further investigation of inspection with respect to qualification rating models to find general-purpose qualification rating models.
- Adaptation of inspection and reading techniques to different application areas and project types, e.g. distributed systems, real-time systems, hardware design and development etc.
- Generalisation of feedback questionnaires to support project- and quality managers in designing inspection processes according to their application area.
- Design, creation and evaluation of tools for inspection support, concerning reading techniques as well.

10 References

10.1 Bibliography

[1] Ackerman, A.F.; Buchwald, L.S.; Lewsky, F.H.: "Software Inspections: An Effective Verification Process"; IEEE Software, 6(3):31–36; 1989

[2] Balzert, Helmut: "Lehrbuch der Software-Technik"; Band 1; Spektrum Akademischer Verlag; ISBN: 3827404800; 2000

[3] Balzert, Helmut: "Lehrbuch der Software-Technik"; Band 2; Spektrum Akademischer Verlag; ISBN: 3827400651; 1997

[4] Basili, Victor R.; Briand, Lionel C.; Melo, Walcelio L.: „*A Validation of Object-Oriented Design Metrics as Quality Indicators*", 1996, IEEE Transactions on Software Engineering, Vol 22, No. 10, 751-761

[5] Basili. V.R.: "Evolving and Packaging Reading Technologies"; Journal of Systems and Software, 38(1); 1997

[6] Basili, V.R.; Green, S.: "Software Process Evolution at the SEL"; IEEE Software, 11(4):58–66; 1994

[7] Basili, V.R.; Shull, F.; Lanubile, F,: "Building Knowledge through Families of Experiments"; IEEE Transactions on Software Engineering, 25(4); 1999

[8] Basili, V.R.; Green, S.; Laitenberger, O.; Lanubile, F.; Shull, F.; Sorumgard, S.; Zelkowitz, M.V.: "The Empirical Investigation of Perspective-based Reading"; Journal of Empirical Software Engineering; 2(1):133–164; 1996

[9] Biffl, Stefan; Brem, Christian; Horak, Klaus: „*Methoden der Software Qualitätssicherung*"; Technische Universität Wien; 1996; Skriptum zur Vorlesung

[10] Biffl, Stefan; Freimut, Bernd; Laitenberger, Oliver: „*Investigating the Cost-Effectiveness of Reinspection in Software Development*", 2000, Technical Report 00-25; Dept. Software Engineering, Vienna

[11] Biffl, Stefan; Halling, Michael: "*Software Product Improvement with Inspection*"; Proceeding Euromicro 2000 workshop on Software Product and Process Improvement, Maastricht, IEEE Comp. Soc. Press, 2000

[12] Biffl, Stefan; Halling, Michael: "*Investigating Reinspection Decision Accuracy regarding Product-Quality and Cost-Benefit Estimates*"; Proc. of Compsac'01; Chicago, Ill, IEEE Comp. Soc. Press; 2001

[13] Biffl, Stefan: "*Analysis of the Impact of Reading Technique and Inspector Capability on Individual Inspection Performance*"; Proc. of APSEC 2000 Asia-Pacific Software Engineering Conference; Singapore; IEEE Comp. Soc. Press; 2000

[14] Biffl, Stefan: "*Using Inspection Data for Defect Estimation*"; IEEE Software special issue on recent project estimation methods; 2000

[15] Biffl, Stefan: „*Software Inspection Techniques to support Project and Quality Management*", Shaker Verlag, 2001, ISBN: 3-8265-8512-7

[16] Biffl, Stefan; Gutjahr, Walter: „*Influence of Team Size and Defect Detection Technique on Inspection Effectiveness*", 2001

[17] Booth, Wayne C; Colomb, Gregory G.; Williams, Joseph M.: "*The Craft of Research*", The University of Chicago Press, 1995, ISBN: 0-226-06584-7

[18] Bröhl, A.; Dröschel, W.: "Das V-Modell"; Oldenbourg; 1995

[19] Ciolkowski, M.; Differding, C.; Laitenberger, O.; Münch, and J.: "Empirical Investigation of Perspective-based Reading: A Replicated Experiment. Technical Report ISERN-97-13, Fraunhofer Institute for Experimental Software Engineering"; 1997

[20] Ciolkowski, M.; Shull, F., Biffl, Stefan: "*A Family of Inspection Experiments*"; Int. Conf. on Empirical Assessment of Software Engineering (EASE); 2002

[21] Coulouris, George; Dollimore, Jean; Kindberg, Tim: "*Distributed Systems – Concepts and Design*", Second Edition, Addison-Wesley, 1996, ISBN: 0-201-62433-8

[22] Curtis, B.: "Measurement and experimentation in software engineering"; Proceedings of the IEEE, 68(9):1144–1157; 1980

[23] Doolan, E.P.: „*Experience with Fagan's Inspection Method*", Software Practice and Experience, Vol 22(2), 173-182, February 1992

[24] Fagan, M.E: „*Design and Code Inspections to Reduce Errors in Program Development*", 1976, IBM System Journal, No 3; p182-211

[25] Fagan, M.E.: "Advances in Software Inspections"; IEEE Transactions on Software Engineering, 12(7):744–751; 1986

[26] Freimut, Bernd; Laitenberger, Oliver; Biffl, Stefan: "*Investigating the Impact of Reading Techniques on the Accuracy of Different Defect Content Estimation Techniques*"; Proc. of IEEE Int. Software Metrics Symposium; London; 2001

[27] Gilb, Tom; Graham, Dorothy: „*Software Inspection*", Addison-Wesley, 1993, ISBN: 0-201-63181-4

[28] Halling, Michael, Biffl, Stefan: "*Investigating the Influence of Software Inspection process Parameters on Inspection Meeting Performance*"; Int. Conf. on Empirical Assessment of Software Engineering (EASE); Keele; 2002

[29] Humphrey, W.H.: "A Discipline for Software Engineering"; Addison-Wesley; 1995

[30] IEEE Computer Society: „*IEEE Standard: Software Quality Managment System, Part 1: Requirements*", IEEE Std 1298-1992

[31] Jacobson, I.; Rumbaugh, J.; Booch, B.: "The Unified Software Development Process"; Object Technology Series; Addison-Wesley; 1999

[32] Jalote, Pankaj; Jalote, P.: "CMM in Practice: Processes for Executing Software Projects at Infosys"; SEI Series in Software Engineering; Addison-Wesley Professional; ISBN: 0201616262; 1999

[33] Johnson, M.P.; Tjahjono, D.: "Does Every Inspection Really Need a Meeting"; Journal of Empirical Software Engineering; 3(1):9–35; 1998

[34] Jones,C.: "Applied Software Measurement - Assuring Productivity and Quality"; McGraw Hill; 2nd edition; 1997

[35] Kelly, J.C., Sherif, J.S.; Hops, J.; "An analysis of defect densities found during software inspections"; Journal of Systems and Software; 17:111–117; 1992

[36] Kitchenham, B.A.; Kitchenham, A.P.; Fellows, J.P.: "The effects of inspections on software quality and productivity"; Technical Report 1; ICL Technical Journal; May 1986

[37] Kneuper, Ralph; Sollman, Frank: „*Normen zum Qualitätsmanagement bei der Softwareentwicklung*"; 1995; Informatik Spektrum 18, Iss 6; 314-323

[38] Knight, J.C.; Myers, E.A.: "Phased Inspections and their Implementation"; ACM SIGSOFT Software Engineering; Notes, 16(3):29–35; 1991

[39] Knight, J.C.; Myers, E.A.: "An Improved Inspection Technique"; Communications of the ACM, 36(11):51–61; 1993.

[40] Kopetz, Hermann: "*Real-Time Systems – Design Principles for Distribued Embedded Applications*"; Kluver Academic Publishers, 1997, ISBN: 0-7923-9894-7

[41] Laitenberger, Oliver: „*Studying the Effects of Code Inspection and Structural Testing on Software Quality*", 1998, IESE-Report 024.98/E, May 1998; Fraunhofer Institute for Experimental Software Engineering, Germany, ISERN-98-10

[42] Laitenberger, Oliver; Atkinson, Colin; Schlich, Maud; El Emam, Khaled: „*An Experimental Comparison of Reading Techniques for Defect detection in UML design Documents*", 2000, IESE-Report 080.99/E, December 1999; INSERN, 2000-01

[43] Laitenberger, O.; DeBaud, J.-M.; "An Encompassing Life-Cycle Centric Survey of Software Inspection"; Journal of Systems and Software; 2000

[44] Masing, Walter (Hrsg): "Handbuch Qualitätsmanagement"; Hanser Fachbuch; ISBN: 3446193979; 1999

[45] Miller, J.; Wood, M.; Roper, M.: "Further Experiences with Scenarios and Checklists"; Journal of Empirical Software Engineering, 3(3):37–64; 1998

[46] Naur, P.; Randell, B.: "Software Engineering - Proceedings of the NATO Working Conference", Garmisch-Partenkirchen; October 1968

[47] Port, D.; Halling, M.; Kazman, R.; Biffl, St.: "*Strategic Quality Assurance Planning*"; Proc. of the 4th Int. Workshop on Economics Driven Software Engineering Research (EDSER-4) at the Int. Conf. on Software Engineering; Orlando; Florida; 2002

[48] Porter, A.A.; Votta, L.G.: "Comparing Detection Methods for Software Requirements Inspection: A Replication using Professional Subjects"; Journal of Empirical Software Engineering, 3(4):355–378; 1998

[49] Porter, A.A.; Votta, L.G.: "What Makes Inspections Work?"; IEEE Software, 14(6):99–102; 1997

[50] Porter, A.A.; Votta, L.G.; Basili, V.R.: "Comparing Detection Methods for Software Requirements Inspections: A Replicated Experiment"; IEEE Transactions on Software Engineering, 21(6):563–575; June 1995

[51] Pfeiffer, Tilo: „*Qualitätsmanagement, Strategien, Methoden, Techniken, 2. Auflage*", München, Wien – Hanser; 1996, ISBN 3-446-18579-8

[52] Roper, Marc; Wood, Murray; Miller, James: „ *An empirical evaluation of defect detection techniques*", Information and Software Technology, 39(11), 793-775, 1997

[53] Rombach; H.D.: "Skript zur Vorlesung Software Engineering I. AG Software Engineering"; Universität Kaiserslautern; 1995

[54] Rubey, Raymond J.: „*Software Quality Assuance Standards - A Comparison and an Integration*", 1991, IEEE TH0365-7/90/0000/0064, 64-71

[55] Runge, Joachim H.: „*Total Quality Management - in die Praxis umgesetzt*"; 1994; IBM Nachrichten 44 Heft 316, 71-75

[56] Schach, Stephen R.: „*Classical and Object-Oriented Software Engineering*", 3rd Edition, Vanderbilt University, 1996, ISBN: 0-256-18298-1

[57] Schneider, G.M.; Martin, J.; Tsai, W.T.: "An experimental study of fault detection in user requirements documents"; ACM Transactions on Software Engineering and Methodology, 1(2):188–204; April 1992

[58] Shull, F.: "Developing Techniques for Using Software Documents: A Series of Empirical Studies"; PhD thesis, University of Maryland; 1998

[59] Strauss, S.H.; Ebenau, R.G.: "Software Inspection Process"; McGraw Hill Systems Design & Implementation Series; 1993

[60] Sommerville, Ian: „*Software Engineering*“, 6th Edition, Addison Wessley, 2001, ISBN: 0-201-39815-X

[61] Stelzer, Dirk; Mellis, Werner; Herzwurm, Georg: „*A critical look at ISO 9000 for software quality management*”; 1997; Software Quality Journal 6, Issue 2

[62] Tervonen I.; Kerola P.: “*Towards deeper co-understanding of software quality*”; 1998; Information and Software Technology 39, pp. 995-1003

[63] Thaller, Georg Erwin: „*Software Qualitat*“, VDE Verlag, 2000, ISBN: 3-8007-2494-4

[64] Wagner, Karl Werner (Hrsg): "PQM - Prozessorientiertes Qualitätsmanagement"; Hanser Fachbuch; ISBN: 3446212299; 2001

[65] Wallmüller, Ernest: „*Qualitätsmanagement in der Informationsverarbeitung*“, 1996, Wirtschaftsinformatik 38, 2; 137-146

[66] Wallmüller, Ernest: „*Software-Qualitätsmanagement in der Praxis*“, 2. völlig überarbeitete Auflage, Hanser Fachbuch, 2001, ISBN: 3-446-21367-8

[67] Wohlin, Claes; Runeson, Per; Höst, Martin; Ohlsson, Magnus C; Regnell, Björn; Wesslen, Anders: „*Experimentation in Software Engineering - An Introduction*”, Kluver Academic Press, 2000, ISBN: 0-7923-8682-5

[68] Zuser, Wolfgang; Biffl, Stefan; Grechenig, Thomas; Köhle, Monika: „*Software Engineering mit UML und dem Unified Process*“, Pearson Studium, 2001, ISBN: 3-8273-7927-2

10.2 Web-References

[69] American Society for Quality (ASQ)
Weblink: http://www.asq.org/join/about/history/shewhart.html

[70] Bootstrap Institute: Methodology & tools for software process assessment & improvement
Weblink: http://www.bootstrap-institute.com/

[71] ISO – International Organization for Standardization
Weblink: http://www.iso.ch

[72] „*Glossary for CMM Models*“; Carnegie Mellon University, Software Engineering Institute (SEI); [Weblink: http://www.sei.cmu.edu]

[73] Software Process Improvement and Capability dEtermination
Weblink: http://www.sqi.gu.edu.au/spice/

11 Indices

11.1 Table of Figures

FIGURE 1.2.1-1: DEVELOPMENT OF SOFTWARE ENGINEERING [SHAW ET AL. 1996] 3
FIGURE 1.2.2-1: DEVELOPMENT OF QUALITY MANAGEMENT 4
FIGURE 1.2.2-2: FIVE PERSPECTIVES ON SOFTWARE QUALITY [62] 6
FIGURE 2.1.3-1: COST DISTRIBUTION IN A SOFTWARE PROCESS [60] 11
FIGURE 2.2-1: SOFTWARE PROCESS COMPARISON SCHACH VS. SOMMERVILLE 12
FIGURE 2.3.1-1: WATERFALL MODEL APPLIED FROM [56] 17
FIGURE 2.3.2-1: V-MODEL ACCORDING TO [68] 18
FIGURE 2.3.3-1: SPIRAL MODEL BY BÖHM [56] 19
FIGURE 3.1-1: PLAN-DO-CHECK-ACT (PDCA-CYCLE) ACCORDING TO [9] 23
FIGURE 3.1-2: PROCESS BASED QUALITY ACCORDING TO [60] 24
FIGURE 3.1.1-1: ISO9001:2000 – AN OVERVIEW ACCORDING TO [63] 25
FIGURE 3.1.1-2: QM-SYSTEMS DOCUMENTATION ACCORDING TO [37] 26
FIGURE 3.1.2-1: SEI CAPABILITY MATURITY MODEL ACCORDING TO [60] 28
FIGURE 3.2.1-1: DIFFERENT KINDS OF AUDITS ACCORDING TO [51] 32
FIGURE 3.2.2-1: DIFFERENTIATION OF REVIEWS [63] [9] 34
FIGURE 4-1: TECHNICAL DIMENSION OF SOFTWARE INSPECTION ADOPTED TO LAITENBERGER ET AL. [42]. . 38
FIGURE 4-2: FRAMEWORK FOR INSPECTION PLANNING AND CONTROL ACCORDING TO BIFFL [15] 40
FIGURE 4.4.3-1: FAMILIES OF READING TECHNIQUE ACCORDING TO [4] 44
FIGURE 5.2-1: QUALITY IMPROVEMENT PARADIGM [67] 48
FIGURE 5.3-1: OVERVIEW OF EXPERIMENT PROCESS [67] 49
FIGURE 6.2-1: RELATED PARTS IN THE REQUIREMENT DOCUMENT ACCORDING TO BIFFL [15] 57
FIGURE 6.2-2, DISTRIBUTION OF SEEDED DEFECTS ACCORDING TO DOCUMENT LOCATION (A) AND SEVERITY (B) 58
FIGURE 6.2-3, DISTRIBUTION OF SEEDED DEFECTS ACCORDING TO DOCUMENT LOCATION AND (A) INITIAL SEVERITY LEVEL, (B) EVALUATION SEVERITY LEVEL 59
FIGURE 6.2-4: INSPECTION PARTICIPANTS 62
FIGURE 6.2-5, DISTRIBUTION OF INSPECTORS ACCORDING TO (A) READING TECHNIQUE, (B) READING TECHNIQUE ROLES 63
FIGURE 6.2-6, DISTRIBUTION OF INSPECTORS ACCORDING TO (A) INITIAL QUALIFICATION, (B) READING TECHNIQUE ROLES 64
FIGURE 6.3-1: EXPERIMENT PROCEEDING ADAPTED TO BIFFL [15] 65
FIGURE 6.3-2: INSPECTION PROCESS – SEQUENCE OF EVENTS ADAPTED TO BIFFL [15] 66
FIGURE 6.4-1: DATABASE MODEL FOR INSPECTION EXPERIMENT EVALUATION (SIMPLIFIED) 67
FIGURE 6.5-1: DATABASE MODEL FOR INSPECTION EXPERIMENT 70
FIGURE 6.5-2: DATABASE MODEL FOR BASIC DATA 71
FIGURE 6.5-3: TASK DESCRIPTION 72
FIGURE 7.1-1: SD-SKILLS: DISTRIBUTION OF INSPECTORS 75
FIGURE 7.1-2: EXP: QUALIFICATION RATING SCALE FOR SELF-ASSESSMENT 76
FIGURE 7.1-3: EXP: DISTRIBUTION OF INSPECTORS 77
FIGURE 7.1-4: EXP: DISTRIBUTION OF INSPECTORS ACCORDING QUESTIONNAIRE SECTIONS 77
FIGURE 7.1-5: SI-DD: DISTRIBUTION OF INSPECTORS 81
FIGURE 7.1-6: SI-DD: DISTRIBUTION OF INSPECTORS ACCORDING TO DEFECT DETECTION 82
FIGURE 7.1-7: QUALIFICATION: COMPARISON OF QUALIFICATION DISTRIBUTION 83
FIGURE 7.2-1: DDR: QUALIFICATION VS. READING TECHNIQUE 85
FIGURE 7.2-2: DDR: INSPECTOR QUALIFICATION VS. DEFECT SEVERITY 87
FIGURE 7.2-3: DDR: READING TECHNIQUE VS. INSPECTOR QUALIFICATION 88
FIGURE 7.2-4: DDR: READING TECHNIQUE VS. DEFECT SEVERITY 89
FIGURE 7.2-5: DDR: INSPECTOR QUALIFICATION VS. DOCUMENT LOCATION 91
FIGURE 7.2-6: DDR: DEFECT SEVERITY VS. DOCUMENT LOCATION 92
FIGURE 7.2-7: DDR: READING TECHNIQUE VS. DOCUMENT LOCATION 93
FIGURE 7.3-1: TIMELINE: INSPECTION DURATION 95
FIGURE 7.3-2: TIMELINE: DURATION UNTIL THE FIRST DEFECT DETECTED 97
FIGURE 7.3-3: TIMELINE: AVERAGE MINUTES BETWEEN DEFECT DETECTION (NOTED DEFECTS) 98

FIGURE 7.3-4: TIMELINE: AVERAGE MINUTES BETWEEN DEFECT DETECTION (MATCHED DEFECTS) 99
FIGURE 7.4-1: FEEDBACK: RT ACCEPTANCE ACCORDING TO READING TECHNIQUE 101
FIGURE 7.4-2: FEEDBACK: RT ACCEPTANCE ACCORDING TO INSPECTOR QUALIFICATION 102

11.2 Table of Tables

TABLE 3.2.1-1: COMPARISON OF ISO VS. CMM AND AUDITS VS. ASSESSMENTS ACCORDING TO [63] 31
TABLE 3.2.1-2: PLANNING AND EXECUTION OF AUDITS 33
TABLE 3.2.2-1: RELATIVE COST FOR DEFECT REMOVAL 34
TABLE 5.1-1: COMPARISON OF EMPIRICAL STRATEGIES ACCORDING TO WOHLIN ET AL. [67] 47
TABLE 6.2-1: DISTRIBUTION OF SEEDED DEFECTS 59
TABLE 6.2-2: DISTRIBUTION OF READING TECHNIQUE (ROLES) 63
TABLE 6.2-3: DISTRIBUTION OF INSPECTORS ACCORDING TO INITIAL QUALIFICATION 64
TABLE 7.1-1: EXP: QUALIFICATION DISTRIBUTION ACCORDING TO GENERAL SKILLS 78
TABLE 7.1-2: EXP: QUALIFICATION DISTRIBUTION ACCORDING TO PROJECT EXPERIENCE 78
TABLE 7.1-3: EXP: QUALIFICATION DISTRIBUTION ACCORDING TO SE SKILLS 79
TABLE 7.1-4: EXP: QUALIFICATION DISTRIBUTION ACCORDING TO SE SKILLS 80
TABLE 7.1-5: SI-DD: QUALIFICATION RATING 81
TABLE 7.1-6: SI-DD: DISTRIBUTION OF INSPECTORS ACCORDING TO DEFECT DETECTION 82
TABLE 7.1-7: QUALIFICATION: COMPARISON OF QUALIFICATION DISTRIBUTION 84
TABLE 7.2-1: ABSOLUTE NUMBER OF DEFECTS FOUND DURING INSPECTION 85
TABLE 7.2-2: DDR: QUALIFICATION VS. READING TECHNIQUE 86
TABLE 7.2-3: DDR: QUALIFICATION VS. DEFECT SEVERITY 87
TABLE 7.2-4: DDR: READING TECHNIQUE VS. INSPECTOR QUALIFICATION 89
TABLE 7.2-5: DDR: READING TECHNIQUE VS. DEFECT SEVERITY 90
TABLE 7.2-6: DDR: INSPECTOR QUALIFICATION VS. DOCUMENT LOCATION 91
TABLE 7.2-7: DDR: DEFECT SEVERITY VS. DOCUMENT LOCATION 93
TABLE 7.2-8: DDR: READING TECHNIQUE VS. DOCUMENT LOCATION 94
TABLE 7.3-1: TIMELINE: INSPECTION DURATION 96
TABLE 7.3-2: TIMELINE: DURATION UNTIL THE FIRST DEFECT DETECTED 97
TABLE 7.3-3: TIMELINE: AVERAGE MINUTES BETWEEN DEFECT DETECTION (NOTED DEFECTS) 99
TABLE 7.3-4: TIMELINE: AVERAGE MINUTES BETWEEN DEFECT DETECTION (MATCHED DEFECTS) 100
TABLE 7.4-1: FEEDBACK: RT ACCEPTANCE ACCORDING TO READING TECHNIQUE 101
TABLE 7.4-2: FEEDBACK: RT ACCEPTANCE ACCORDING TO READING TECHNIQUE 102

12 Appendices

Appendix A – ISO 900x series

Appendix A contains tables according to the structure of ISO norm series:

A.1: ISO 9001:1994

A.2: ISO 9003

A.3: ISO 9001:2000

Appendix B – Questionaire according to the SWT-Experiment 2000

Appendix B contains the complete questionaire, used during the experiement. They contain subjective estimates concerning reading technique roles and inspector experience. The summarized information is used for inspector qualification calculation in chapter 7.

B.1: Erhebung Erfahrung

B.2: Subjektive Beurteilung von checklistenbasiertem Lesen

B.3: Subjektive Beurteilung von szenariobasiertem Lesen

Appendix C – Task-Description for Reading-Techniques

Appendix C contains basic material for inspection execution. Taskdescriptions explain the usage of the specified reading-technique and lead the inspectors through the inspection process according to their reading-technique role.

C.1: Taskdescription: Readingtechnique CBR

C.2: Taskdescription: Readingtechnique SBR; Designer

C.3: Taskdescription: Readingtechnique SBR; User

C.4: Taskdescription: Readingtechnique SBR; Tester