EXPERIMENTATION IN SOFTWARE ENGINEERING
An Introduction

Claes Wohlin
Per Runeson
Martin Höst
Magnus C. Ohlsson
Björn Regnell
Anders Wesslén

Foreword by Anneliese von Mayrhauser

Springer Science+Business Media, LLC

EXPERIMENTATION IN SOFTWARE ENGINEERING

An Introduction

THE KLUWER INTERNATIONAL SERIES IN SOFTWARE ENGINEERING

Series Editor

Victor R. Basili
University of Maryland
College Park, MD 20742

Also in the Series:

FORMAL SPECIFICATION TECHNIQUES FOR ENGINEERING MODULAR C PROGRAMS, *by TAN Yang Meng;* ISBN: 0-7923-9653-7

TOOLS AND ENVIRONMENTS FOR PARALLEL AND DISTRIBUTED SYSTEMS, *by Amr Zaky and Ted Lewis;* ISBN: 0-7923-9675-8

CONSTRAINT-BASED DESIGN RECOVERY FOR SOFTWARE REENGINEERING: Theory and Experiments, *by Steven G. Woods, Alexander E. Quilici and Qiang Yang;* ISBN: 0-7923-8067-3

SOFTWARE DEFECT MODELING, *by Kai-Yuan Cai;* ISBN: 0-7923-8259-5

NON-FUNCTIONAL REQUIREMENTS IN SOFTWARE ENGINEERING, *by Lawrence Chung, Brian A. Nixon, Eric Yu and John Mylopoulos;* ISBN: 0-7923-8666-3

The Kluwer International Series in Software Engineering addresses the following goals:

- To coherently and consistently present important research topics and their application(s).
- To present evolved concepts in one place as a coherent whole, updating early versions of the ideas and notations.
- To provide publications which will be used as the ultimate reference on the topic by experts in the area.

With the dynamic growth evident in this field and the need to communicate findings, this series provides a forum for information targeted toward Software Engineers.

EXPERIMENTATION IN SOFTWARE ENGINEERING

An Introduction

by

Claes Wohlin
Per Runeson
Martin Höst
Magnus C. Ohlsson
Björn Regnell
Anders Wesslén
Lund University
Sweden

SPRINGER SCIENCE+BUSINESS MEDIA, LLC

Library of Congress Cataloging-in-Publication Data

Experimentation in software engineering : an introduction / by Claes Wohlin ... [et al.].
p. cm.--(The Kluwer international series in software engineering ; [006])
Includes bibliographical references and index.
ISBN 978-1-4613-7091-8 ISBN 978-1-4615-4625-2 (eBook)
DOI 10.1007/978-1-4615-4625-2
1. Software engineering. 2. Computer software--Evaluation. I. Wohlin, Claes. II. Kluwer international series in software engineering ; 6.

QA76.758 .E995 2000
005.1--dc21

99-048367

Originally published by Kluwer Academic Publishers in 2000
Softcover reprint of the hardcover 1st edition 2000

Printed on acid-free paper.

Contents

FOREWORD

It is my belief that software engineers not only need to know software engineering methods and processes, but that they also should know how to assess them. Consequently, I have taught principles of experimentation and empirical studies as part of the software engineering curriculum. Until now, this meant selecting a text from another discipline, usually psychology, and augmenting it with journal or conference papers that provide students with software engineering examples of experiments and empirical studies.

This book fills an important gap in the software engineering literature: it provides a concise, comprehensive look at an important aspect of software engineering: experimental analysis of how well software engineering methods, methodologies, and processes work. Since all of these change so rapidly in our field, it is important to know how to evaluate new ones. This book teaches how to go about doing this and thus is valuable not only for the software engineering student, but also for the practicing software engineering professional who will be able to

- Evaluate software engineering techniques.
- Determine the value (or lack thereof) of claims made about a software engineering method or process in published studies.

Finally, this book serves as a valuable resource for the software engineering researcher.

Professor Anneliese von Mayrhauser
Colorado State University, USA

PREFACE

Have you ever had a need to evaluate software engineering methods or techniques against each other? This book presents experimentation as one way of evaluating new methods and techniques in software engineering. Experiments are a valuable tool for all software engineers who are involved in evaluating and choosing between different methods, techniques, languages and tools.

It may be that you are a software practitioner, who wants to evaluate methods and techniques before introducing them into your organization. You may also be a researcher, who wants to evaluate new research results against something existing, in order to get a scientific foundation for your new ideas. You may be a teacher, who believes that knowledge of empirical studies in software engineering is essential to your students. Finally, you may be a student in software engineering who wants to learn some methods to turn software engineering into a science and to obtain quantitative data when comparing different methods and techniques. This book provides guidelines and examples of how you should proceed to succeed in your mission.

Software engineering and science

The term "software engineering" was coined 30 years ago, and the area is still maturing. Software engineering has over the years been driven by technology development and advocacy research. The latter referring to that we have invented

and introduced new methods and techniques over the years based on marketing and conviction rather than scientific results. To some extent, it is understandable with the pace the information society has established itself during the last couple of decades. It is, however, not acceptable in the long run if we want to have control of the software we develop. Control comes from being able to evaluate new methods, techniques, languages and tools before using them. Moreover, this would help us turn software engineering into a science. Before looking at the issues we must address to turn software engineering into a science, let us look at the way science is viewed in other areas.

In "Fermat's Last Theorem" by Dr. Simon Singh, [Singh97], science is discussed. The essence of the discussion can be summarized as follows. In science, physical phenomena are addressed by putting forward hypotheses. The phenomenon is observed and if the observations are in line with the hypothesis, this becomes evidence for the hypothesis. The intention is also that the hypothesis should enable prediction of other phenomena. Experiments are important to test the hypothesis and in particular the predictive ability of the hypothesis. If the new experiments support the hypothesis, then we have more evidence in favor of the hypothesis. As the evidence grows and becomes strong, the hypothesis can be accepted as a scientific theory.

The summary is basically aiming at hypothesis testing through empirical research. This is certainly not the way most research is conducted in software engineering today. The need to conduct more empirical studies in software engineering has, however, been stressed more and more lately. Empirical studies include surveys, experiments and case studies. Thus, the objective of this book is to introduce and promote the use of empirical studies in software engineering with a particular emphasis on experimentation.

Purpose

The purpose of the book is to introduce students, teachers, researchers, and practitioners to experimentation and experimental evaluation with a focus on software engineering. The objective is in particular to provide guidelines of how to perform experiments to evaluate methods, techniques and tools in software engineering. The introduction is provided through a process perspective. The focus is on the steps that we have to go through to perform an experiment. The process can be generalized to other types of empirical studies, but the main focus here is on experiments and quasi-experiments.

The motivation for the book comes from the need of support we experienced when turning our software engineering research more experimental. Several books are available which either treat the subject in very general terms or focus on some

specific part of experimentation; most of them focusing on the statistical methods in experimentation. These are important, but there is a lack of books elaborating on experimentation from a process perspective. Moreover, there are no books addressing experimentation in software engineering in particular.

Scope

The scope of the book is primarily experiments in software engineering as a means for evaluating methods, techniques etc. The book provides some information regarding empirical studies in general, including both case studies and surveys. The intention is to provide a brief understanding of these strategies and in particular to relate them to experimentation.

The chapters of the book cover different steps to go through to perform experiments in software engineering. Moreover, examples of empirical studies related to software engineering are provided throughout the book. It is of particular importance to illustrate for software engineers that empirical studies and experimentation can be practiced successfully in software engineering. To stress this issue, one chapter in the book contains a survey of some examples of published experiments in software engineering. Furthermore, two examples of experiments are included in the book. These are introduced to illustrate the experiment process and to exemplify how software engineering experiments can be reported. The intention is that these studies should work as good examples and sources of inspiration for further empirical work in software engineering. The book is mainly focused on experiments, but it should be remembered that other strategies are also available, for example, case studies and surveys. In other words, we do not have to resort to advocacy research and marketing without quantitative data when research strategies as, for example, experiments are available.

Target audience

The target audience of the book can be divided into four categories.

Students may use the book as an introduction to experimentation in software engineering with a particular focus on evaluation. The book is suitable as a course book in undergraduate or graduate studies where the need for empirical studies in software engineering is stressed. Exercises and project assignments are included in the book to combine the more theoretical material with some practical aspects.

Teachers may use the book in their classes if they believe in the need of making software engineering more empirical. The book is suitable as an introduction to the

area. It should be fairly self-contained, although an introductory course in statistics is recommended.

Researchers may use the book to learn more about how to conduct empirical studies and use them as one important ingredient in their research. Moreover, the objective is that it should be fruitful to come back to the book and use it as a checklist when performing empirical research.

Practitioners may use the book as a "cookbook" when evaluating some new methods or techniques before introducing them into their organization. Practitioners are expected to learn how to use empirical studies in their daily work when changing, for example, the development process in the organization they are working.

Outline

The book is divided into thirteen chapters. The outline of the book is summarized in Figure 1. Chapter 1 is a general introduction to the area of empirical studies. It puts empirical studies in general and experiments in particular into a software engineering context. Further, the first chapter emphasizes why empirical studies are needed in software engineering. In Chapter 2, empirical strategies (surveys, case studies and experiments) are discussed in general and the context of empirical studies is elaborated, in particular from a software engineering perspective. Chapter 3 provides a brief introduction to measurement theory. In Chapter 4, the focus is set on experimentation. A general experiment process is introduced, and the following chapters elaborate the process steps in detail. Chapter 5 discusses how to define an experiment, and Chapter 6 focuses on the planning phase. Operation of the experiment is discussed in Chapter 7 and Chapter 8 presents some methods for analyzing and interpreting the results. Chapter 9 discusses presentation and packaging of the experiment. After having introduced the different steps to go through in experimentation, a selective literature survey is presented in Chapter 10. The objective of the survey is to provide some references to software engineering experiments published in the literature. The survey is by no means exhaustive, but the intention is to provide a reference to further reading for those who would like to find examples of experiments in the area. In Chapter 11, an example is presented where the main objective is to illustrate the experiment process, and the example in Chapter 12 is used to illustrate how an experiment in software engineering may be reported in a paper. Finally, some exercises and data are presented in Chapter 13. The book also contains two appendices, one with statistical tables and one that provides an overview of the experiment process. The tables are primarily included to provide support for some of the examples in the book. More comprehensive tables are available in most statistics books.

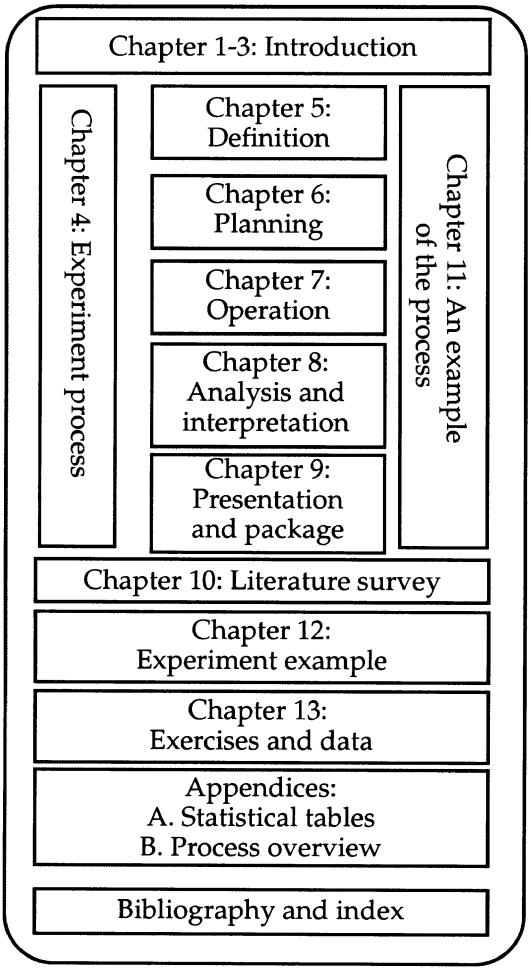


Figure 1. *Structure of the book.*

Exercises

The exercises are divided into four categories:

- **Understanding**. Five questions capturing the most important points for Chapters 1-9 are provided. The objective is to ensure that the reader has understood the most important concepts.
- **Training**. These exercises provide an opportunity to practice experimentation. The exercises are particularly targeted towards analyzing data and answering questions in relation to an experiment.
- **Reviewing**. This exercise is aimed at the examples of experiments presented in Chapters 11-12, and to some extent also the articles referenced in Chapter 10. The objective is to give an opportunity to review some presented experiments. After having read several experiments presented in the literature, it is clear that most experiments suffer from some problems. This is mostly due to the inherit problems of performing experimentation in software engineering. Instead of promoting criticism of work by others, we have provided some examples of studies that we have conducted ourselves. They are, in our opinion, representative of the type of experiments that are published in the literature. This includes that they have their strengths and weaknesses.
- **Assignments**. The objective of these exercises is to illustrate how experiments can be used in evaluation. These assignments are examples of studies that can be carried out within a course, either at a university or in industry. They are deliberately aimed at problems that can be addressed by fairly simple experiments. The assignments can either be done after reading the book or one of the assignments can be carried out as the book is read. The latter provides an opportunity to practice while reading the chapters. As an alternative, we would like to recommend teachers to formulate an assignment, within their area of expertise, that can be used throughout the book to exemplify the concepts presented in each chapter.

Web page

More information about the book can be found through its web page at:
http://www.telecom.lth.se/SERG/ESE/.

References

[Singh97] Singh, Simon, *Fermat's Last Theorem*, pp. 21-22, Fourth Estate, London, UK, 1997.

ACKNOWLEDGMENT

A new book is almost never just an achievement by the authors. Support and help from several persons including families, friends, colleagues, international researchers in the field and funding organizations are often a prerequisite for a new book. This book is certainly no exception. Thus, we would like to thank the following for having contributed to the book.

First, we would like to thank the first main external user of the book Professor Giuseppe Visaggio, University of Bari in Italy for adopting a draft of this book in one of his courses, and for providing valuable feedback. We would also like to express our gratitude to Prof. Anneliese von Mayrhauser, Colorado State University, Fort Collins, USA and Dr. Khaled El Emam, National Research Council, Ottawa, Canada for encouraging us to publish the book and for valuable comments. We also would like to thank Dr. Lionel Briand, Carleton University, Ottawa in Canada, Christian Bunse and Dr. John Daly, Fraunhofer Institute for Experimental Software Engineering, Kaiserslautern, Germany for providing the data for the example on object-oriented design.

The book has been used and evaluated internally within the Software Engineering Research Group at the Department of Communication Systems, Lund University. Thus, we would like to thank the members of the group for providing feedback on different drafts of the book. In particular, we would like to thank Lars Bratthall for taking the time to review the manuscript very thoroughly and providing valuable comments. We would also like to acknowledge the anonymous reviewers for their contribution to the book.

In addition to the above individuals, we would also like to thank all members of ISERN (International Software Engineering Research Network) for interesting and enlightening discussion regarding empirical software engineering research.

Finally, we would like to thank NUTEK (The Swedish National Board for Industrial and Technical Development) for the following research grants: 93-02850, 1K1P-97-09673 and 1K1P-97-09690. The grants have funded research projects where empirical studies have been a cornerstone. This book is partly a result of these research projects.

Lund, Sweden in August 1999

Claes Wohlin,
Per Runeson,
Martin Höst,
Magnus C. Ohlsson,
Björn Regnell, and
Anders Wesslén

1 INTRODUCTION

The information technology revolution has meant, among other things, that software has become a part of more and more products. Software is found in products ranging from toasters to space shuttles. This means that a vast amount of software has been and is being developed. Software development is by no means easy; it is a highly creative process. The rapid growth of the area has also meant that numerous software projects have run into problems in terms of missing functionality, cost overruns, missed deadlines and poor quality. These problems or challenges were identified already in the 1960's, and in 1968 the term "software engineering" was coined with the intention of creating an engineering discipline that focused on the development of software systems.

Software engineering is formally defined by IEEE [IEEE90] as "software engineering means application of a systematic, disciplined, quantifiable approach to development, operation and maintenance of software". Software engineering in general is presented and discussed in books as, for example, [Sommerville95, Pfleeger98]. The objective here is to present how empirical studies and experiments in particular fit into a software engineering context. Three aspects in the definition above are of particular importance here. First, it implies a software process through pointing at different life cycle phases; secondly, it stresses the need for a systematic and disciplined approach; finally, it highlights the importance of quantification. The use of empirical studies is related to all three of them. The software engineering context is further discussed in Section 1.1. The need to turn software engineering

more scientific and how empirical studies play an important role in this is discussed in Section 1.2.

1.1 Software engineering context

A software process model is used to describe the steps to take and the activities to perform when developing software. Examples of software process models are the waterfall model, incremental development, evolutionary development and the spiral model. These and other models are discussed in the general software engineering literature. A simplistic view of the software process is shown in Figure 2. It should be noted that the process is crucial whether we work with development of a new product or maintenance of an existing product.

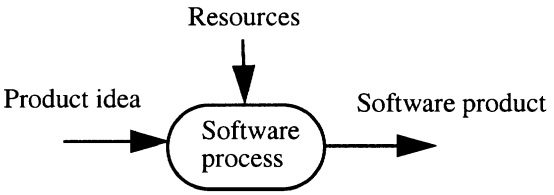


Figure 2. *An illustration of the software process.*

In Figure 2, an idea and resources, primarily in the form of people, are inputs to the software process, and the people develop a software product going through the different steps and performing the different activities in the software process.

The development of software products is many times a complex task. Software projects may run over a long period of time and involve many people, due to the complexity of the software products that are developed. This implies that the software process often also becomes very complex. It consists of many different activities and many documents are written before the final product can be delivered. The complexity of the software process means that it is difficult to optimize it or even find a good enough process. Thus, it is important for companies to strive to improve their way of making business if they intend to stay competitive. This means that most companies are continuously trying to improve their software process in order to improve the products, lower the cost and so forth. The software process stresses the need for a systematic and disciplined approach of working. This approach is also needed when improving the software process, and hence a process improvement process is needed.

An example of an improvement process tailored for software development is the Quality Improvement Paradigm (QIP) [Basili85]. It consists of several steps to support a systematic and disciplined approach to improvement. The QIP is presented briefly in Section 2.6.2. A more general improvement process is the well-known Plan/Do/Check/Act cycle [Demming86]. The improvement processes include two activities, although the same terminology is not always used, that we would like to highlight:

- Assessment of the software process.
- Evaluation of a software process improvement proposal.

The assessment is conducted to identify suitable areas for improvement. Several models exist for assessing the software process. The most well known is probably the Capability Maturity Model (CMM) from Software Engineering Institute at Carnegie-Mellon University, USA [Paulk93]. The assessment models help in pinpointing where improvements are needed. The CMM has five maturity levels with so called key process areas on each level. It is recommended that companies focus on improvement areas according to their maturity level.

Assuming that it is possible to identify areas for improvement through some form of assessment, the next step is to determine how these areas of improvement may be addressed to cope with the identified problems. For example, if too many defects are found in system testing, it may be possible to improve earlier testing, inspections or even specific parts in the development, for example, software design. The objective is that the assessment of the current situation and knowledge about the state-of-the-art should result in that concrete process improvement proposals can be identified. When the improvement proposals have been identified, it is necessary to determine which to introduce, if any. It is often not possible just to change the existing software process without having more information about the actual effect of the improvement proposal. In other words, it is necessary to evaluate the proposals before making any major changes.

One problem that arises is that a process improvement proposal is very hard to evaluate without human involvement directly. For a product, it is possible to first build a prototype to evaluate if it is something to work further with. For a process, it is not possible to build a prototype. It is possible to make simulations and compare different processes, but it should be remembered that this is still an evaluation that is based on a model. The only real evaluation of a process or process improvement proposal is to have people using it, since the process is just a description until it is used by people. Empirical studies are crucial to the evaluation of processes and human-based activities. It is also beneficial to use empirical studies when there is a need to evaluate the use of software products or tools. Experimentation provides a systematic, disciplined, quantifiable and controlled way of evaluating human-based activities. This is one of the main reasons why empirical research is common in social and behavioral sciences, see for example [Robson93].

In addition, empirical studies and experiments in particular are also important for researchers in software engineering. New methods, techniques, languages and tools should not just be suggested, published and marketed. It is crucial to evaluate new inventions and proposals in comparison with existing ones. Experimentation provides this opportunity, and should be used accordingly. In other words, we should use the methods and strategies available when conducting research in software engineering. This is further discussed next.

1.2 Science and software engineering

Software engineering is a cross-disciplinary subject. It stretches from technical issues such as databases and operating systems, through language issues, for example, syntax and semantics, to social issues and psychology. Software development is human-intensive; we are, at least today, unable to manufacture new software. It is a discipline based on creativity and the ingenuity of the people working in the field. Nevertheless, we should, when studying and doing research in software engineering, aim at treating it as a science. This implies using scientific methods for doing research and when making decisions regarding changes in the way we develop software.

In order to perform scientific research in software engineering, we have to understand the methods that are available to us, their limitations and when they can be applied. Software engineering stems from the technical community. Thus, it is natural to look at the methods used for research in, for example, hardware design and coding theory, but based on the nature of software engineering we should look at other disciplines too.

In [Glass94], four research methods in the field of software engineering are summarized. They were initially presented in a software engineering context in [Basili93]. The methods are:

- **The scientific method**. The world is observed and a model is built based on the observation, for example, a simulation model.
- **The engineering method**. The current solutions are studied and changes are proposed, and then evaluated.
- **The empirical method**. A model is proposed and evaluated through empirical studies, for example, case studies or experiments.
- **The analytical method**. A formal theory is proposed and then compared with empirical observations.

The engineering method and the empirical method can be seen as variations of the scientific method [Basili93].

Traditionally, the analytical method is used in the more formal areas of electrical engineering and computer science. The scientific method is used in applied areas, such as simulating a telecommunication network in order to evaluate its performance. It should, however, be noted that simulation as such is not only applied in the scientific method. Simulation may be used as a means for conducting an experiment as well. The engineering method is probably dominating in industry.

The empirical studies have traditionally been used in social sciences and psychology, where we are unable to state any laws of nature, as in physics. In social sciences and psychology, they are concerned with human behavior. The important observation, in this context, is hence that software engineering is very much governed by human behavior through the people developing software. Thus, we cannot expect to find any formal rules or laws in software engineering except perhaps when focusing on specific technical aspects. The focus of this book is on applying and using empirical studies in software engineering. The objective is in particular to emphasize the underlying process when performing empirical studies in general and experimentation in particular. An experiment process is presented, which highlights the basic steps to perform experiments, provides guidelines of what to do and exemplifies the steps using software engineering examples.

It must be noted that it is not claimed that the analytical, scientific and engineering methods are inappropriate for software engineering. They are necessary for some parts of software engineering, for example, we may build mathematical models for software reliability growth [Lyu96]. Moreover, the research methods are not orthogonal, and hence it may, for example, be appropriate to conduct an empirical study within, for example, the engineering method. The important point is that we should make better use of the methods available within empirical research. They are frequently used in other disciplines, for example, behavioral sciences, and the nature of software engineering has much in common with disciplines outside the technical parts of engineering.

The need for experimentation in software engineering was really emphasized for the first time in the middle of the 1980's in [Basili86]. Other articles stressing the need for empiricism in software engineering have since been published, see for example [Basili96b, Fenton94a, Glass94, Potts93, Kitchenham95, Tichy98]. The lack of empirical evidence in software engineering research is stressed in [Tichy95, Zelkowitz98]. The latter publications indicate that the research in software engineering is still too much of advocacy research [Potts93]. A more scientific approach to software engineering is needed.

The focus of this book is on software engineering and the application and use of empirical studies, in particular experimentation, in software engineering. Empirical strategies in software engineering include:

- setting up formal experiments,
- studying real projects in industry, i.e. performing a case study, and

- performing surveys through, for example, interviews.

These strategies are described in some more detail in Chapter 2 before focusing on experimentation. A more general introduction to these research strategies can be found in, for example, [Robson93]. The strategies are neither completely orthogonal nor competing. They provide a convenient classification, but some studies may be viewed as combinations of them or somewhere between two of them. Thus, there are both similarities and differences between the strategies.

The main reason to use experimentation in software engineering is to enable understanding and identification of relationships between different variables. A number of preconceived ideas exist, but are they true? Does object-orientation improve reuse? Are inspections cost-effective? Should we have inspection meetings or is it sufficient to hand-in the remarks to a moderator? This type of questions can be investigated in order to improve our understanding of software engineering. Improved understanding is the basis for changing and improving the way we work, hence empirical studies in general and experimentation in particular are important.

The introduction to the area is based on the introduction of a process for experimentation. The basic steps in the process can be used for other types of empirical studies too. The focus is, however, on providing guidelines and support for performing experiments in software engineering. Furthermore, it should be noted that "true" experiments, i.e. experiments with full randomization, are difficult to perform in software engineering. Software engineering experiments are often quasi-experiments, i.e. experiment in which it, for example, has not been possible to select participants in the experiments by random. Quasi-experiments are important, and they can provide valuable results. The process presented is aimed at both "true" experiments and quasi-experiments. The latter is particularly supported by a thorough discussion of threats to experiments.

Thus, the intention of this book is to provide an introduction to empirical studies and experimentation, in order to highlight the opportunities and benefits of doing experiments in the field of software engineering. The empirical research method can, and should be used more in software engineering. The arguments against empirical studies in software engineering are refuted in [Tichy98]. Hopefully, this introduction to experimentation in software engineering facilitates the use of empirical studies and experimentation both within software engineering research and practice.

2 EMPIRICAL STRATEGIES

There are two types of research paradigms that have different approaches to empirical studies. **Qualitative research** is concerned with studying objects in their natural setting. A qualitative researcher attempts to interpret a phenomenon based on explanations that people bring to them [Denzin94]. Qualitative research begins with accepting that there is a range of different ways of interpretation. It is concerned with discovering causes noticed by the subjects in the study, and understanding their view of the problem at hand. The subject is the person, which is taking part in an experiment in order to evaluate an object.

Quantitative research is mainly concerned with quantifying a relationship or to compare two or more groups [Creswell94]. The aim is to identify a cause-effect relationship. The quantitative research is often conducted through setting up controlled experiments or collecting data through case studies. Quantitative investigations are appropriate when testing the effect of some manipulation or activity. An advantage is that quantitative data promotes comparisons and statistical analysis.

It is possible for qualitative and quantitative research to investigate the same topics but each of them will address a different type of question. For example, a quantitative investigation could be launched to investigate how much a new inspection method decreases the number of faults found in test. To answer questions about the sources of variations between different inspection groups, a qualitative investigation could be launched.

As mentioned earlier quantitative strategies such as controlled experiments are appropriate when testing the effects of a treatment while a qualitative study of

beliefs and understandings are appropriate to find out *why* the results from a quantitative investigation are as they are. The two approaches should be regarded as complementary rather than competitive.

The objective of this chapter is twofold: 1) to introduce empirical research strategies, and 2) to illustrate how the strategies can be used in the context of technology transfer and improvement. To fulfil the first objective, an overview of empirical strategies is provided, see Section 2.1, and then surveys, case studies and experiments are discussed in some more detail. The different empirical strategies are presented briefly in Sections 2.2-2.4, and a comparison of them is provided in Section 2.5. Second, the use of the research strategies within a technology transfer process and as being part of an improvement program is discussed in Section 2.6.

2.1 Overview of empirical strategies

Depending on the purpose of the evaluation, whether it is techniques, methods or tools, and depending on the conditions for the empirical investigation, there are three major different types of investigations (strategies) that may be carried out, [Robson93].

- **Survey**. A survey is often an investigation performed in retrospect, when, for example, a tool or technique, has been in use for a while [Pfleeger94]. The primary means of gathering qualitative or quantitative data are interviews or questionnaires. These are done through taking a sample which is representative from the population to be studied. The results from the survey are then analyzed to derive descriptive and explanatory conclusions. They are then generalized to the population from which the sample was taken. Surveys are discussed further in [Babbie90, Robson93].
- **Case study.** Case studies are used for monitoring projects, activities or assignments. Data is collected for a specific purpose throughout the study. Based on the data collection, statistical analyses can be carried out. The case study is normally aimed at tracking a specific attribute or establishing relationships between different attributes. The level of control is lower in a case study than in an experiment. A case study is an observational study while the experiment is a controlled study [Zelkowitz98]. A case study may, for example, be aimed at building a model to predict the number of faults in testing. Multivariate statistical analysis is often applied in this type of studies. The analysis methods include linear regression and principal component analysis [Manly94]. Case study research is further discussed in [Robson93, Stake95, Pfleeger94, Yin94].

- **Experiment.** Experiments are normally done in a laboratory environment, which provides a high level of control. When experimenting subjects are assigned to different treatments at random. The objective is to manipulate one or more variables and control all other variables at fixed levels. The effect of the manipulation is measured, and based on this a statistical analysis can be performed. In some cases it may be impossible to use true experimentation, we may have to use quasi-experiments. The latter term is often used when it is impossible to perform random assignment of the subjects to the different treatments. An example of an experiment in software engineering is to compare two different methods for inspections. For this type of studies methods for statistical inference are applied with the purpose of showing with statistical significance that one method is better than the other [Montgomery97, Siegel88, Robson93]. The statistical methods are also further discussed in Chapter 8.

Surveys are very common within social sciences where, for example, attitudes are polled to determine how a population will vote in the next election. A survey provides no control of the execution or the measurement, though it is possible to compare it with similar ones, but it is not possible to manipulate variables as in the other investigation methods [Babbie90].

Case study research is a technique where key factors that may have any affect on the outcome are identified and then the activity is documented [Yin94, Stake95]. Case study research is an observational method, i.e. it is done by observation of an on-going project or activity.

An experiment is a formal, rigorous and controlled investigation. In an experiment the key factors are identified and manipulated. The separation between case studies and experiment can be represented by the notion of a state variable [Pfleeger94]. In an experiment, the state variable can assume different values and the objective is normally to distinguish between two situations, for example, a control situation and the situation under investigation. Examples of a state variable could be, for example, the inspection method or experience of the software developers. In a case study, the state variable only assumes one value, which is governed by the actual project under study.

Some of the research strategies could be classified both as qualitative and quantitative depending on the design of the investigation, see Table 1. The classification of a survey depends on the design of the questionnaires, i.e. which data is collected and if it is possible to apply any statistical methods. Also, this is true for case studies but the difference is that a survey is done in retrospect while a case study is done while a project is executed. A survey could also be launched before the execution of a project. In the latter case, the survey is based on previous experiences and hence conducted in retrospect to these experiences although the objective is to get some ideas of the outcome of the forthcoming project.

Experiments are purely quantitative since they have a focus on measuring different variables, change them and measure them again. During these investigations quantitative data is collected and then statistical methods are applied. The following sections give an introduction to each empirical strategy.

Table 1. *Qualitative vs. quantitative in empirical strategies.*

Strategy	Qualitative / Quantitative
Survey	Both
Case study	Both
Experiment	Quantitative

2.2 Surveys

Surveys are conducted when the use of a technique or tool already has taken place [Pfleeger94] or before it is introduced. It could be seen as a snapshot of the situation to capture the current status. Surveys could, for example, be used for opinion polls and market research.

When performing survey research the interest may be, for example, in studying how a new development process has improved the developer's attitudes towards quality assurance. Then a sample of developers is selected from all the developers at the company. A questionnaire is constructed to obtain information needed for the research. The questionnaires are answered by the sample of developers. The information collected are then arranged into a form that can be handled in a quantitative or qualitative manner.

2.2.1 Survey characteristics

Sample surveys are almost never conducted to create an understanding of the particular sample. Instead, the purpose is to understand the population, from which the sample was drawn [Babbie90]. For example, by interviewing 25 developers on what they think about a new process, the opinion of the larger population of 100 developers in the company can be predicted. Surveys aim at the development of generalized suggestions.

Surveys have the ability to provide a large number of variables to evaluate, but it is necessary to aim at obtaining the largest amount of understanding from the fewest number of variables since this reduction also eases the analysis work.

It is not necessary to guess which the most relevant variables in the initial design of the study are. The survey format allows the collection of many variables, which

can be quantified and processed by computers. This makes it is possible to construct a variety of explanatory models and then select the one that best fits the purposes of the investigation.

2.2.2 Survey purposes

The general objectives for conducting a survey is either of the following [Babbie90]:

- Descriptive.
- Explanatory.
- Explorative.

Descriptive surveys can be conducted to enable assertions about some population. This could be determining the distribution of certain characteristics or attributes. The concern is not about why the observed distribution exists, but instead what that distribution is.

Explanatory surveys aim at making explanatory claims about the population. For example, when studying how developers use a certain inspection technique, we might want to explain why some developers prefer one technique while others prefer another. By examining the relationships between different candidate techniques and several explanatory variables, we may try to explain why developers choose one of the techniques.

Finally, explorative surveys are used as a pre-study to a more thorough investigation to assure that important issues are not foreseen. Creating a loosely structured questionnaire and letting a sample from the population answer it could do this. The information is gathered and analyzed, and the results are used to improve the full investigation. In other words, the explorative survey does not answer the basic research question, but it may provide new possibilities that could be analyzed and should therefore be followed up in the more focused or thorough survey.

2.2.3 Data collection

The two most common means for data collection are questionnaires and interviews [Babbie90]. Questionnaires could both be provided in paper form or in some electronic form, for example, e-mail or WWW pages. The basic method for data collection through questionnaires is to send out the questionnaire together with instructions on how to fill it out. The responding person answers the questionnaire and then returns it to the researcher.

Letting interviewers handle the questionnaires (by telephone or face-to-face) instead of the respondents themselves, offers a number of advantages:

- Interview surveys typically achieve higher response rates than, for example, mail surveys.
- An interviewer generally decreases the number of "do not know" and "no answer", because the interviewer can answer questions about the questionnaire.
- It is possible for the interviewer to observe and ask questions.

The disadvantage is the cost and time, which depend on the size of the sample, and they are also related to the intentions of the investigation.

2.3 Case studies

A case study is conducted to investigate a single entity or phenomenon with in a specific time space. The researcher collects detailed information on, for example, one single project during a sustained period of time. During the performance of a case study, a variety of different data collection procedures may be applied [Creswell94].

If we, for example, would like to compare two methods, it might be necessary to organize the study as a case study or an experiment, depending on the scale of the evaluation. An example can be to use a pilot project to evaluate the effects of a change compared to some baseline [Kitchenham95].

Case studies are very suitable for industrial evaluation of software engineering methods and tools because they can avoid scale-up problems. The difference between case studies and experiments is that experiments sample over the variables that are being manipulated, while case studies sample from the variables representing the typical situation. An advantage of case studies is that they are easier to plan but the disadvantages are that the results are difficult to generalize and harder to interpret, i.e. it is possible to show the effects in a typical situation, but it cannot be generalized to every situation [Yin94].

If the effect of a process change is very widespread, a case study is more suitable. The effect of the change can only be assessed at a high level of abstraction because the process change includes smaller and more detailed changes throughout the development process [Kitchenham95]. Also, the effects of the change cannot be identified immediately. For example, if we would like to know if a new design tool increases the reliability, it may be necessary to wait until after delivery of the developed product to assess the effects on operational failures.

Case study research is a standard method used for empirical studies in various sciences such as sociology, medicine and psychology. Within software engineering, case studies should not only be used to evaluate how or why certain phenomena occur, but also to evaluate the differences between, for example, two design methods. This means in other words, to determine "which is best" of the two methods

[Yin94]. An example of a case study in software engineering is an investigation if the use of perspective-based reading increases the quality of requirements specifications. A study like this cannot verify that perspective-based reading reduces the number of faults that reaches test, since this requires a reference group that does not use perspective-based techniques.

2.3.1 Case study arrangements

A case study can be applied as a comparative research strategy, comparing the results of using one method or some form of manipulation, to the results of using another approach. To avoid bias and to ensure internal validity, it is necessary to create a solid base for assessing the results of the case study. There are three ways to arrange the study to facilitate this [Kitchenham95]:

- A comparison of the results of using the new method against a company baseline is one solution. The company should gather data from standard projects and calculate characteristics like average productivity and defect rate. Then it is possible to compare the results from the case study with the figures from the baseline.
- A sister project can be chosen as a baseline. The project under study uses the new method and the sister project the current one. Both projects should have the same characteristics, i.e. the projects must be comparable.
- If the method applies to individual product components, it could be applied at random to some components and not to others. This is very similar to an experiment, but since the projects are not drawn at random from the population of all projects, it is not an experiment.

2.3.2 Confounding factors and other aspects

When performing case studies it is necessary to minimize the effects of confounding factors. A confounding factor is a factor that makes it impossible to distinguish the effects from two factors from each other. This is important since we do not have the same control over a case study as in an experiment. For example, it may be difficult to tell if a better result depends on the tool or the experience of the user of the tool. Confounding effects could involve problems with learning how to use a tool or method when trying to assess its benefits, or using very enthusiastic or skeptical staff.

There are both pros and cons with case studies. Case studies are valuable because they incorporate qualities that an experiment cannot visualize, for example, scale, complexity, unpredictability, and dynamism. Some potential problems with case studies are:

- A small or simplified case study is seldom a good instrument for discovering software engineering principles and techniques. Increases in scale lead to changes in the type of problems that become most indicative. In other words, that the problem may be different in a small case study and in a large case study, although the objective is to study the same issues. For example, in a small case study the main problem may be the actual technique being studied, and in a large case study the major problem may be the amount of people involved and hence also the communication between people.
- Researchers are not completely in control of a case study situation. This is good, from one perspective, because unpredictable changes frequently tell them much about the problems being studied. The problem is that we cannot be sure about the effects due to confounding factors.

2.4 Experiments

Experiments are launched when we want control over the situation and want to manipulate behavior directly, precisely and systematically. Also, experiments involve more than one treatment to compare the outcomes. For example, if it is possible to control who is using one method and who is using another method, and when and where they are used, it is possible to perform an experiment. This type of manipulation can be made in an off-line situation, for example in a laboratory under controlled conditions, where the events are organized to simulate their appearance in the real world. Experiments may alternatively be made on-line, which means that the investigation is executed in the field under normal conditions [Babbie90]. The level of control is more difficult in an on-line situation, but some factors may be possible to control while others may be impossible.

As mentioned earlier, considering the notion of a state variable makes it possible to state the difference between case studies and experiments more rigorously. Examples of state variables could be the application area and the system type. In an experiment, we identify the variables and sample over them. This means that we select objects representing a variety of characteristics that is typical for the organization in which the experiment is conducted and design the research so that more than one value will be measured for each characteristic. An example could be to investigate the effect of an inspection method with respect to the faults found in test. Then the inspection method is the state variable and an experiment will involve objects where, for example, different programming languages are used.

The design of the experiment should be made so that the objects involved represent all the methods we are interested in. Also, it is possible to consider the current situation to be the baseline (control) and the new situation to be the one we want to

evaluate. Then a state variable describes how the evaluated situation differs from the control. Though, the values of all the other state variables should stay the same, e.g. application domain and programming environment.

2.4.1 Characteristics

Experiments are appropriate to investigate different aspects, including:

- Confirm theories, i.e. to test existing theories.
- Confirm conventional wisdom, i.e. to test people's conceptions.
- Explore relationships, i.e. to test that a certain relationship holds.
- Evaluate the accuracy of models, i.e. to test that the accuracy of certain models is as expected.
- Validate measures, i.e. to ensure that a measure actually measure what it is supposed to.

The strength of an experiment is that it can investigate in which situations the claims are true and they can provide a context in which certain standards, methods and tools are recommended for use.

2.4.2 Experiment process

Carrying out an experiment involves several different steps. The different steps are:

- Definition.
- Planning.
- Operation.
- Analysis and interpretation.
- Presentation and package.

The experiment process is presented in Chapter 4, and the different steps are discussed in more detail in Chapters 5-9. Thus, experimentation is not elaborated in this chapter in the same detail as surveys and case studies.

2.5 Empirical strategies comparison

The prerequisites for an investigation limit the choice of research strategy. A comparison of strategies can be based on a number of different factors. Table 2 is an extension of the different factors discussed in [Pfleeger94]. The factors are further described below.

Table 2. *Research strategy 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

Execution control describes how much control the researcher has over the study. For example, in a case study data is collected during the execution of a project. If management decides to stop the studied project due to, for example, economical reasons, the researcher cannot continue carry out the case study. The opposite is the experiment where the researcher is in control of the execution.

Measurement control is the degree to which the researcher can decide upon which measures to be collected, and to include or exclude during execution of the study. An example is how to collect data about requirement volatility. During the execution of a survey we cannot include this kind of measures, but in a case study or in an experiment it is possible to include them. In a survey, we can only collect data regarding people's opinion about requirement volatility.

Closely related, to the factors above, is the **investigation cost**. Depending on which strategy is chosen, the cost differs. This is related to, for example, the size of the investigation and the need for resources. The strategy with the lowest cost is the survey, since it does not require a large amount of resources. The difference between case studies and experiments is that if we choose to investigate a project in a case study, the outcome from the project is some form of product that may be retailed, i.e. it is an on-line investigation. In an off-line experiment the outcome is some form of experience which is not profitable in the same way as a product.

Another important aspect to consider is the possibility to **replicate** the investigation. The purpose of a replication is to show that the result from the original experiment is valid for a larger population. A replication becomes a "true" replication if it is possible to replicate both the design and the results. It is not uncommon that the objective is to perform a replication, but the results, to some extent, turn out differently than the results of the original study.

The replication of an experiment involves repeating the investigation under identical conditions, but on another population. This helps finding out how much confidence it is possible to place in the results of the experiment. If the assumption of randomization is correct, i.e. the subjects are representative of a larger population, replications within this population show the same results as the previous performed experiment. If we do not get the same results, we have been unable to capture all aspects in the experiment design that affects the result. Even if it is possible to measure a certain variable or to replicate an experiment, it might be difficult and too costly.

Another aspect related to replication, in the sense that it is concerned with studies over time, is longitudinal studies. The main difference between a longitudinal study and a replication is that a longitudinal study is primarily conducted with the same subjects and a replication is mostly a study conducted with new subjects. In other words, replication means several studies and a longitudinal study is a single study. The longitudinal study is conducted over a period of time, for example, a survey can be done at several occasions, experiments can be repeated and the case study may also be longitudinal if it is conducted over a period of time. A longitudinal study is normally conducted to understand, describe or evaluate something that changes over time [Robson93].

The choice of empirical strategy depends on the prerequisites for the investigation, the purpose of it, available resources and how we would like to analyze the collected data.

2.6 Empiricism in a software engineering context

Why should we perform empirical software engineering? The major reasons for carrying out quantitative empirical studies is the opportunity of getting objective and statistically significant results regarding the understanding, controlling, prediction, and improvement of software development. Empirical studies are important input to the decision-making in an improvement seeking organization.

Before introducing new techniques, methods, or other ways of working, an empirical assessment of the virtues of such changes is preferred. In this section, a framework for evaluation of software process changes is presented, where different empirical strategies are suggested in three different contexts: desktop, laboratory, and development projects.

To be successful in software development there are some basic requirements [Basili85, Basili89, Demming86]:

1. Understanding of the software process and product.
2. Definition of process and product qualities.

3. Evaluation of successes and failures.
4. Information feedback for project control.
5. Learning from experience.
6. Packaging and reuse of relevant experience.

Empirical studies are important to support the achievement of these requirements, and fit into the context of industrial and academic software engineering research, as well as in a learning organization, seeking continuous improvement. An example of a learning organization, called Experience Factory, is proposed in conjunction with the Quality Improvement Paradigm [Basili85], as further described in the sequel of this section. This approach also includes a mechanism for defining and evaluating a set of operational goals using measurement. This mechanism is called Goal/Question/Metric (GQM) Paradigm [Basili94b], which is further described below. The GQM Paradigm is described in more detail in [Solingen99]. Before moving into the paradigms and methods, it is useful to understand the scale of the different types of studies.

2.6.1 Empirical evaluation of process changes

An improvement seeking organization wants to assess the impact of process changes (e.g., a new method or tool) before introducing them to improve the way of working. Empirical studies are important in order to get objective and quantifiable information on the impact of changes. In Sections 2.2-2.4, three empirical strategies are described: surveys, case studies and experiments, and they are compared in Section 2.5. This section describes how the strategies may be used when software process changes are evaluated [Wohlin96]. The objective is to discuss the strategies in terms of a suitable way of handling technology transfer from research to industrial use. Technology transfer and some different steps in that process in relation to using empirical strategies are also discussed in [Linkman97].

In Figure 3, the strategies are placed in appropriate research environments. The order of the strategies is based on the "normal" size of the study. In other words, the ordering does not refer to performing a survey, experiment and case study respectively to achieve the same result. The objective is to order the studies based on how they normally may be conducted to enable a controlled way of transferring research results into practice. A survey does not intervene with the normal software development to any large extent hence there is a small risk. An experiment is normally rather limited in comparison with a real project and the case study is normally aimed at one specific project. Furthermore, an experiment may be carried out in a university environment prior to doing a study in industry, hence lowering the cost and risk, see also [Linkman97].

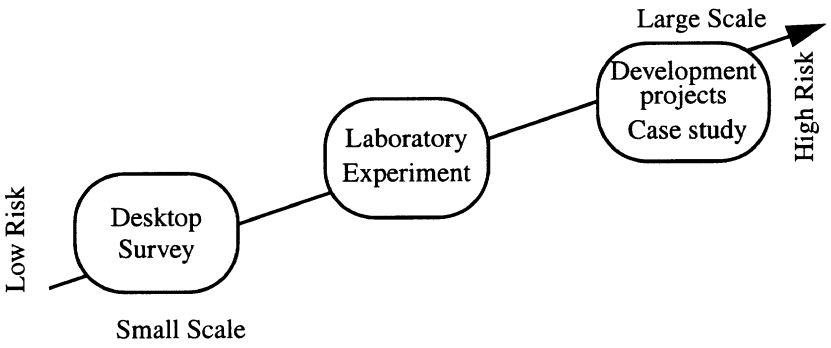


Figure 3. *Surveys, experiments and case studies.*

The research environments are:

- **Desktop**. The change proposal is evaluated off-line without executing the changed process. Hence, this type of evaluation does not involve people that apply the method, tool, etc. In the desktop environment, it is suitable to conduct surveys, for example, through interview-based evaluations and literature studies.
- **Laboratory**. The change proposal is evaluated in a laboratory setting (off-line), where an experiment is conducted and a limited part of the process is executed in a controlled manner.
- **Development projects**. The change proposal is evaluated in a real development situation, i.e. it is observed on-line. This involves, for example, pilot projects. In this environment it is often too expensive to conduct controlled experiments. Instead, case studies are often more appropriate.

In Figure 3, the placement of the different research environments indicates an increase in scale and risk. In order to try out, for example a new design method in a large-scale design project and in a realistic environment, we may apply it in a development project as a pilot study. This is, of course, more risky compared to a laboratory or desktop study, as failure of the process change may, for example, endanger the quality of the delivered product. Furthermore, it is often more expensive to carry out experiments and case studies, compared to desktop evaluation, as a desktop study does not involve the execution of a development process. It should be noted that the costs refer to the cost for investigating the same thing. For example, it is probably less costly to first interview people about the expected impact of a new review method than performing a controlled experiment, which in turn is less costly than actually using the new method in a project with the risks involved in adopting new technology. Generally, it is, of course, not possible to state that one empirical

strategy is more costly than another is, since it depends on the type and size of study.

Before a case study is carried out in a development project, it may be wise to carry out limited studies in either or both desktop and laboratory environments. However, there is no general conclusion on order and cost; for every change proposal, a careful assessment should be made of which empirical strategies are most effective for the specific situation. The key issue is to choose the best strategy based on cost and risk, and in many cases it may be wise to start in a small scale and then as the knowledge increases and the risk decreases the study is scaled up.

Independently of which research strategy we use, there is a need for methodology support in terms of how to work with improvement, how to collect data and to store the information. These issues are further discussed subsequently.

2.6.2 Quality Improvement Paradigm

The Quality Improvement Paradigm (QIP) [Basili85] is a general improvement scheme tailored for the software business. QIP is similar to the Plan/Do/Check/Act cycle [Demming86], and includes six steps as illustrated in Figure 4.

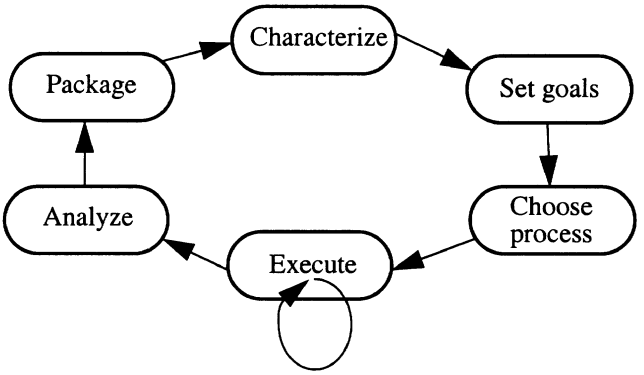


Figure 4. *The six steps of the Quality Improvement Paradigm [Basili85].*

These steps are explained below [Basili94a].

1. **Characterize**. Understand the environment based upon available models, data, intuition, etc. Establish baselines with the existing business processes in the organization and characterize their criticality.

2. **Set goals**. On the basis of the initial characterization and of the capabilities that has a strategic relevance to the organization, set quantifiable goals for successful project and organization performance and improvement. The reasonable expectations are defined based upon the baseline provided by the characterization step.
3. **Choose process**. On the basis of the characterization of the environment and the goals that have been set, choose the appropriate processes for improvement, and supporting methods and tools, making sure that they are consistent with the goals that have been set.
4. **Execute**. Perform the product development and provide project feedback based upon the data on goal achievements that are being collected.
5. **Analyze**. At the end of each specific project, analyze the data and the information gathered to evaluate the current practices, determine problems, record findings, and make recommendations for future project improvements.
6. **Package**. Consolidate the experience gained in the form of new, or updated and refined, models and other forms of structured knowledge gained from this and prior projects.

The QIP implements two feedback cycles [Basili94a], see also Figure 4:

- The *project feedback* cycle (**control cycle**) is the feedback provided to the project during the execution phase. Whatever the goals of the organization, the project used as a pilot should use its resources in the best possible way; therefore quantitative indicators at project and task level are useful in order to prevent and solve problems.
- The *corporate feedback* cycle (**capitalization cycle**) is the feedback loop that is provided to the organization. It has the double purpose of providing analytical information about project performance at project completion time by comparing the project data with the nominal range in the organization and analyzing concordance and discrepancy. Reusable experience is accumulated in a form that is useful and applicable to other projects.

2.6.3 Experience Factory

The QIP is based on that the improvement of software development requires continuous learning. Experience should be packaged into *experience models* that can be effectively understood and modified. Such experience models are stored in a repository, called **experience base**. The models are accessible and can be modified for reuse in current projects.

QIP focuses on a logical separation of project development (performed by the Project Organization) from the systematic learning and packaging of reusable experience (performed by the Experience Factory) [Basili89]. The Experience Factory is

thus a separate organization that supports product development by analyzing and synthesizing all kinds of experience, acting as a repository for such experience, and supplying that experience to various projects on demand, see Figure 5.

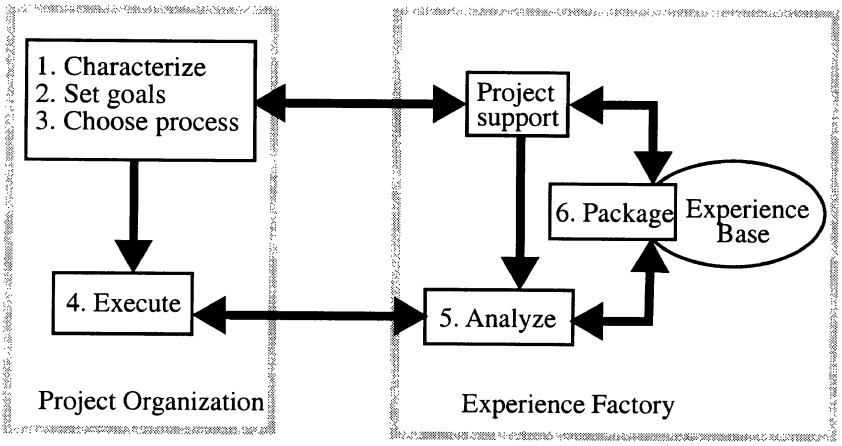


Figure 5. *Experience Factory.*

The Experience Factory packages experience by "building informal, formal, or schematised models and measures of various processes, products, and other forms of knowledge via people, documents, and automated support" [Basili94a].

The goal of the Project Organization is to produce and maintain software. The project organization provides the Experience Factory with project and environment characteristics, development data, resource usage information, quality records, and process information. It also provides feedback on the actual performance of the models processed by the experience factory and utilized by the project.

The Experience Factory processes the information received from the development organization, and returns direct feedback to each project, together with goals and models tailored from similar projects. It also provides baselines, tools, lessons learned, and data, tailored to the specific project.

To be able to improve, a software developing organization needs to introduce new technology. It needs to experiment and record its experiences from development projects and eventually change the current development process. When the technology is substantially different from the current practice, the evaluation may be off-line in order to reduce risks. The change evaluation, as discussed above, may take the form of a controlled experiment (for detailed evaluation in the small) or of a case study (to study the scale effects). In both cases, the Goal/Question/Metric paradigm, as described subsequently, provides a useful framework.

2.6.4 Goal/Question/Metric paradigm

The Goal/Question/Metric (GQM) [Basili94b, Solingen99] approach is based upon the assumption that for an organization to measure in a purposeful way it must:

1. specify the goals for itself and its projects,
2. trace those goals to the data that is intended to define those goals operationally, and
3. provide a framework for interpreting the data with respect to the stated goals.

The result of the application of the GQM approach is a specification of a measurement model targeting a particular set of issues and a set of rules for the interpretation of the measurement data.

The resulting measurement model has three levels:

1. **Conceptual level** (Goal). A goal is defined for an object, for a variety of reasons, with respect to various models of quality, from various points of view, relative to a particular environment. Objects of measurement are products, processes, and resources (see also Chapter 3).
2. **Operational level** (Question). A set of questions is used to characterize the way the assessment/achievement of a specific goal is going to be performed based on some characterization model. Questions try to characterize the objects of measurement (product, process and resource) with respect to a selected quality issue and to determine its quality from the selected viewpoint.
3. **Quantitative level** (Metric). A set of data is associated with every question in order to answer it in a quantitative way (either objectively or subjectively).

In Figure 6, the hierarchical structure of GQM is illustrated.

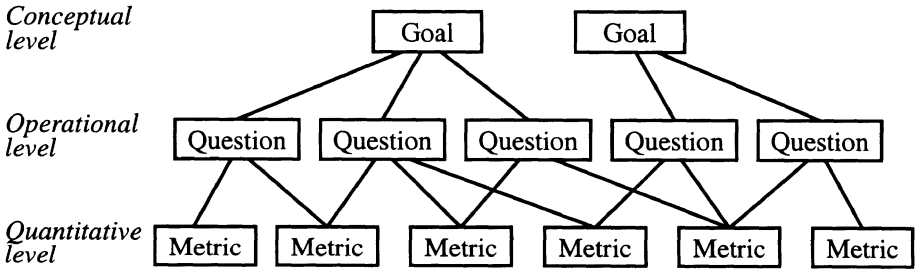


Figure 6. *GQM model hierarchical structure.*

The process of setting goals is critical to the successful application of the GQM approach. Goals are formulated based on (1) policies and strategies of the organiza-

tion, (2) descriptions of processes and products, and (3) organization models. When goals have been formulated, questions are developed based on these goals. Once the questions have been developed, we proceed to associating the questions with appropriate metrics.

Practical guidelines of how to use GQM for measurement-based process improvement are given in [Briand96b, Solingen99]. In Chapter 3, general aspects of measurement are further described.

3 MEASUREMENT

Software measurement is crucial to enable control of projects, products and processes, or as stated by DeMarco: "*You cannot control what you cannot measure*", [DeMarco82]. Moreover, measurement is a central part in empirical studies. Empirical studies are used to investigate the effects of some input to the object under study. To control the study and to see the effects, we must be able to both measure the inputs in order to describe what causes the effect on the output, and to measure the output. Without measurements, it is not possible to have the desired control and therefore an empirical study cannot be conducted.

Measurement and measure are defined as, [Fenton96]:

"***Measurement*** *is a mapping from the empirical world to the formal, relational world. Consequently, a* ***measure*** *is the number or symbol assigned to an entity by this mapping in order to characterize an attribute.*"

Instead of making judgement directly on the real entity, we study the measures and make the judgement on them. The word metric or metrics is also often used in software engineering. Two different meanings can be identified. First of all, software **metrics** is used as a term for denoting the field of measurement in software engineering. The book by Fenton and Pfleeger [Fenton96] and the International Software Metrics Symposium are examples of this. Secondly, the word metric is used to denote an entity which is measured, for example, LOC (lines of code) is a product metric. More precisely, it is a measure of the size of the program. Software measurement is also further discussed in [Shepperd95].

In this chapter, basic measurement theory is presented. Section 3.1 describes the basic concept of measurement theory, and the different scale types of measures. Examples of measures in software engineering and the relation to the statistical analysis are presented in Section 3.2.

3.1 Basic concepts

A measure is a mapping from the attribute of an entity to a measurement value, usually a numerical value. Entities are objects we can observe in the real world. The purpose of mapping the attributes into a measurement value is to characterize and manipulate the attributes in a formal way. One of the basic characteristics of a measure is therefore that it must preserve the empirical observations of the attribute, [Fenton94a]. That is, if object A is longer than object B, the measure of A must be greater than the measure of B.

When we use a measure in empirical studies, we must be certain that the measure is **valid**. To be valid, the measure must not violate any necessary properties of the attribute it measures and it must be a proper mathematical characterization of the attribute.

A valid measure allows different objects to be distinguished from one another, but within the limits of measurement error, objects can have the same measurement value. The measure must also preserve our intuitive notions about the attribute and the way in which it distinguishes different objects, [Kitchenham95].

The mapping from an attribute to a measurement value can be made in many different ways, and each different mapping of an attribute is a **scale**. If the attribute is the length of an object, we can measure it in meters, centimeters or inches, each of which is a different scale of the measure of the length.

As a measure of an attribute can be measured in different scales, we sometimes want to transform the measure into another scale. If this transformation from one measure to another preserves the relationship among the objects, it is called an **admissible transformation**, [Fenton96]. An admissible transformation is also called **rescaling**.

With the measures of the attribute, we make statements about the object or the relation between different objects. If the statements are true even if the measures are rescaled, they are called **meaningful**, otherwise they are **meaningless**, [Briand96a]. For example, if we measure the lengths of objects A and B to 1 meter and 2 meters respectively, we can make the statement that B is twice as long as A. This statement is true even if we rescale the measures to centimeters or inches, and is therefore meaningful. Another example, we measure the temperature in room A and room B to 10°C and 20°C, and make the statement that room B is twice as hot as room A. If we rescale the temperatures to the Fahrenheit scale, we get the tem-

peratures 50°F and 68°F. The statement is no longer true and is therefore meaningless.

Depending on which admissible transformation that can be made on a scale, different scale types can be defined. Scales belonging to a scale type, share the same properties and the scale types are more or less powerful in the sense that more meaningful statements can be made the more powerful the scale is. The most commonly used scale types are described below.

Measures can also be classified in two other ways: (1) if the measure is direct or indirect, or (2) if the measure is objective or subjective. These classifications are further discussed later in this chapter.

3.1.1 Scale types

The most common scale types are the following[1], [Fenton94a, Fenton96, Briand96a]:

- **Nominal scale.** The nominal scale is the least powerful of the scale types. It only maps the attribute of the entity into a name or symbol. This mapping can be seen as a classification of entities according to the attribute.

  Possible transformations for nominal scales are those that preserve the fact that the entities only can be mapped one-to-one.

  Examples of a nominal scale are classification, labeling and defect typing.

- **Ordinal scale.** The ordinal scale ranks the entities after an ordering criterion, and is therefore more powerful than the nominal scale. Examples of ordering criteria are "greater than", "better than", and "more complex".

  The possible transformations for the ordinal scale are those that preserve the order of the entities, i.e. $M'=F(M)$ where M' and M are different measures on the same attribute, and F is a monotonic increasing function.

  Examples of an ordinal scale are grades and software complexity.

- **Interval scale.** The interval scale is used when the difference between two measures are meaningful, but not the value itself. This scale type orders the values in the same way as the ordinal scale but there is a notion of "relative distance" between two entities. The scale is therefore more powerful than the ordinal scale type.

1. In [Fenton94a, Fenton96], a fifth scale type is presented. The scale type is the absolute scale and is a special case of the ratio scale. The absolute scale is used when the value itself is the only meaningful transformation. An example of an absolute scale is counting.

Possible transformations with this scale type are those where the measures are a linear combination of each other, i.e. $M'=\alpha M+\beta$ where M' and M are different measures on the same attribute. Measures on this scale are uncommon in software engineering.

Examples of an interval scale are temperature measured in Celsius or Fahrenheit.

- **Ratio scale.** If there exists a meaningful zero value and the ratio between two measures is meaningful, a ratio scale can be used.

  Possible transformations are those that have the same zero and the scales only differs by a factor, i.e. $M'=\alpha M$ where M' and M are different measures on the same attribute.

  Examples of a ratio scale are length, temperature measured in Kelvin and duration of a development phase.

The measurement scales are related to qualitative and quantitative research. According to [Kachigan86], qualitative research is concerned with measurement on the nominal and ordinal scales, and quantitative research treats measurement on the interval and ratio scales.

3.1.2 Objective and subjective measures

Sometimes, the measurement of an attribute cannot be measured without considering the viewpoint they are taken from. We can divide measures into two classes:

- **Objective measure.** An objective measure is a measure where there is no judgement in the measurement value and is therefore only dependent on the object that is being measured. An objective measure can be measured several times and the same value can be obtained within the measurement error. Examples of objective measures are Lines of Code (LOC), and delivery date.
- **Subjective measure.** A subjective measure is the opposite of the objective measure. The person making the measurement contributes by making some sort of judgement. The measure depends on both the object and the viewpoint from which they are taken. A subjective measure can be different if the object is measured again. A subjective measure is mostly of nominal or ordinal scale type. Examples of subjective measures are Personnel skill, and usability.

3.1.3 Direct or indirect measures

The attributes that we are interested in are sometimes not directly measurable. These measures must be derived through other measures that are directly measurable. To distinguish the direct measurable measures from derived measures, we divide the measures into direct and indirect measures.

- **Direct measure.** A direct measurement of an attribute is directly measurable and does not involve measurements on other attributes. Examples of direct measures are Lines of Code, and the number of defects found in test.
- **Indirect measure.** An indirect measurement involves the measurement of other attributes. The indirect measure is derived from the other measures. Examples of indirect measures are Defect density (Number of defects divided by the number of lines of code), and programmers productivity (lines of code divided by the programmer's effort).

3.2 Measurements in software engineering

The objects that are of interest in software engineering can be divided into three different classes:

- **Process.** The process describes which activities that are needed to produce the software.
- **Product.** The products are the artifacts, deliverables or documents that results from a process activity.
- **Resources.** Resources are the objects, such as personnel, hardware, or software, needed for a process activity.

In each of the classes we also make a distinction between internal and external attributes, [Fenton94b]. An internal attribute is an attribute that can be measured purely in terms of the object. The external attributes can only be measured with respect to how the object relates to other objects. Examples of different software measures are shown in Table 3.

Table 3. *Examples of measures in software engineering.*

Class	Examples of objects	Type of attribute	Examples of measures
Product	Code	Internal	Size
		External	Reliability
Process	Testing	Internal	Effort
		External	Cost
Resource	Personnel	Internal	Age
		External	Productivity

Often in software engineering, software engineers want to make statements of an external attribute of an object. Unfortunately, the external attributes are mostly indirect measures and must be derived from internal attributes of the object. The internal attributes are mostly direct measures.

The measures are often part of a measurement program. Building software measurement programs is discussed in, for example, [Grady94] and [Hetzel93].

Measurements in software engineering are different from measurements in other domains, e.g. physics. It is often clear, which the attributes are, and how they are measured in those domains. In software engineering, however, it is sometimes difficult to define an attribute in a measurable way with which everyone agrees [Fenton96]. Another difference is that it is difficult to prove that the measures are anything else but nominal or ordinal scale types in software engineering. Validation of the indirect measures is more difficult as both the direct measures and the models to derive the external measure have to be validated.

When conducting empirical studies, we are interested in the scale types of the measures as the statistical analysis depends on them. Formally, the statistical analysis methods depend upon the scale type, but the methods are mostly rather robust regarding scale type. The basic rule is that the more powerful scale types we use, the more powerful analysis methods we may use, see Chapter 8.

Many measures in software engineering are often measured with nominal or ordinal scales, or it is not proven that it is a more powerful scale type. This means that we cannot use the most powerful statistical analysis methods, which requires interval or ratio scales, for the empirical studies we conduct.

In [Briand96a], it is argued that we can use the more powerful statistical analysis even if we cannot prove that we have interval or ratio scales. Many of the more powerful statistical methods are robust to non-linear distortions of the interval scale if the distortion is not too extreme. If we take care and carefully consider the risks, we can make use of the more powerful statistical methods and get results that otherwise would be infeasible without a very large sample of measures.

4 EXPERIMENT PROCESS

Experimentation is not simple; we have to prepare, conduct and analyze experiments properly. One of the main advantages of an experiment is the control of, for example, subjects, objects and instrumentation. This ensures that we are able to draw more general conclusions. Other advantages include ability to perform statistical analysis using hypothesis testing methods and opportunities for replication. To ensure that we make use of the advantages, we need a process supporting us in our objectives in doing experiments correctly (the notion of experiments include quasi-experiments, unless clearly stated otherwise). The basic principles behind an experiment are illustrated in Figure 7.

The starting point is that we have an idea of a cause and effect relationship, i.e. we believe that there is a relationship between a cause construct and an effect construct. We have a theory or are able to formulate a hypothesis. A hypothesis means that we have an idea of, for example, a relationship, which we are able to state formally in a hypothesis.

In order to evaluate our beliefs, we may use an experiment. The experiment is created to, for example, test a theory or hypothesis. In the design of the experiment, we have a number of treatments (values that the studied variable can take, see below) over which we have control. The experiment is performed and we are able to observe the outcome. This means that we test the relationship between the treatment and the outcome. If the experiment is properly set up, we should be able to draw conclusions about the relationship between the cause and the effect for which we stated a hypothesis.

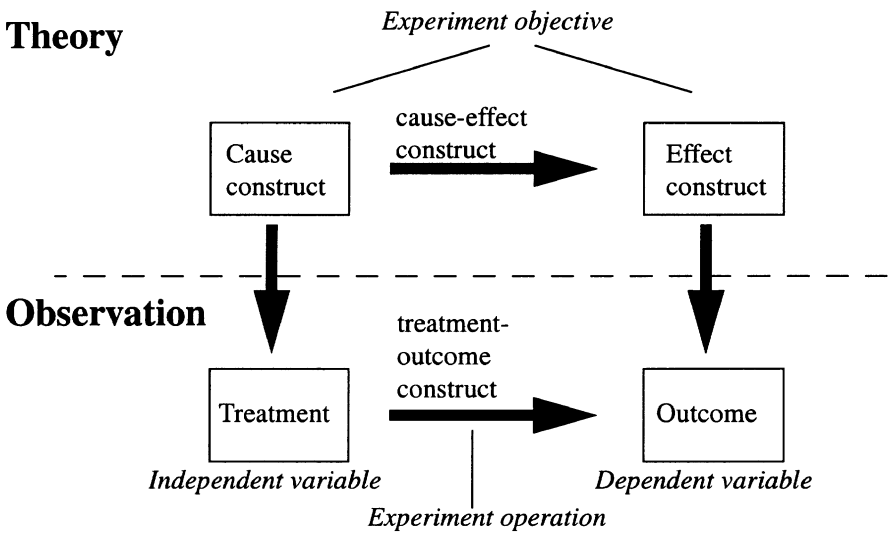


Figure 7. *Experiment principles (adapted from [Trochim99]).*

The main objective of an experiment is mostly to evaluate a hypothesis or relationship, see also Section 2.4.1. Hypothesis testing normally refers to the former and the latter is foremost a matter of building a relational model based on the data collected. The model may be derived using multivariate statistical methods, for example, regression techniques and then we evaluate it in an experiment. The focus in this book is primarily on hypothesis testing. Multivariate statistical methods are treated in, for example, [Kachigan86, Kachigan91, Manly94].

The experiment process presented in this chapter is formulated to make sure that the proper actions are taken to ensure a successful experiment. It is unfortunately not uncommon that some factor is overlooked before the experiment, and the oversight prevents us from doing the planned analysis and hence we are unable to draw valid conclusions. The objective, of having a process, is to provide support in setting up and conducting an experiment. The activities in an experiment are briefly outlined in this chapter and treated in more detail in the following chapters, see Chapters 5-9.

4.1 Variables, treatments, objects and subjects

Before discussing the experiment process, it is necessary to introduce a few definitions in order to have a vocabulary for experimentation. When conducting a formal experiment, see Section 2.4, we want to study the outcome when we vary some of the input variables to a process. There are two kinds of variables in an experiment, independent and dependent variables, see Figure 8.

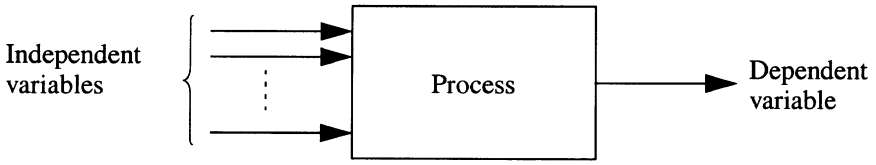


Figure 8. *Illustration of independent and dependent variables.*

Those variables that we want to study to see the effect of the changes in the independent variables are called **dependent variables** (or response variables). Often there is only one dependent variable in an experiment. All variables in a process that are manipulated and controlled are called **independent variables**.

Example: We want to study the effect of a new development method on the productivity of the personnel. For example, we may have chosen to introduce an object-oriented design method instead of a function-oriented approach. The *dependent variable* in the experiment is the productivity. *Independent variables* are, for example, the development method, the experience of the personnel, tool support, and the environment.

An experiment studies the effect of changing one or more independent variables. Those variables are called **factors**. The other independent variables are controlled at a fixed level during the experiment, or else we cannot say if the factor or another variable causes the effect. A **treatment** is one particular value of a factor.

Example: The *factor* for the example experiment above, is the development method since we want to study the effect of changing the method. We use two *treatments* of the factor: the old and the new development method.

The choice of treatment, and at which levels the other independent variable shall have, is part of the experiment design, see Figure 9. Experiment design is described in more detail in Chapter 6.

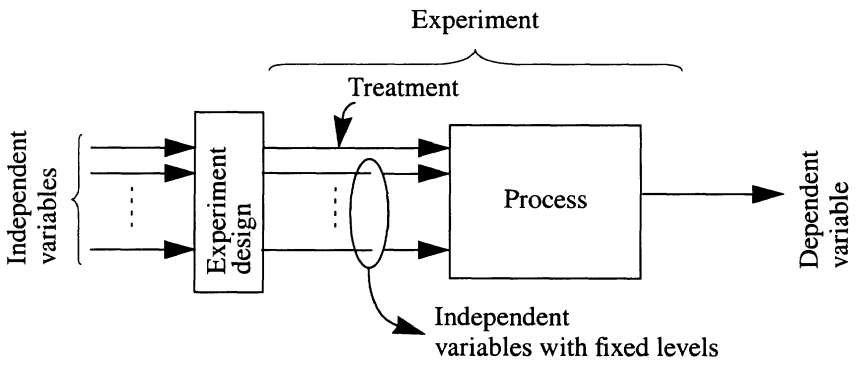


Figure 9. *Illustration of an experiment.*

The treatments are being applied to the combination of **objects** and **subjects**. An object can, for example, be a document that shall be reviewed with different inspection techniques. The people that apply the treatment are called **subjects**[1]. The characteristics of both the objects and the subjects can be independent variables in the experiment.

Example: The *objects* in the example experiment are the programs to be developed and the *subjects* are the personnel.

An experiment consists of a set of **tests** (sometimes called trials) where each test is a combination of treatment, subject and object. It should be observed that this type of test should not be confused with the use of statistical tests, which is further discussed in Chapter 8. The number of tests affects the experimental error, and provides an opportunity to estimate the mean effect of any experimental factor. The experimental error helps us to know how much confidence we can place in the results of the experiment.

Example: A *test* can be that person *N* (*subject*) uses the new development method (*treatment*) for developing program *A* (*object*).

1. Sometimes the term participant is used instead of the term subject. The term subject is mainly used when people are considered with respect to different treatments and with respect to the analysis and the term participant mainly when it deals with how to engage and motivate people in a study.

4.2 Process

A process provides steps that support an activity, for example, software development. Processes are important as they can be used as checklists and guidelines of what to do and how to do it. To perform an experiment, several steps have to be taken and they have to be in a certain order. Thus, a process for how to perform experiments is needed.

The process presented is focused on experimentation, but the same basic steps must be performed in any empirical study. The main difference is the work within a specific activity, for example, the design of a survey, experiment and case study differ, but they all need to be designed. Thus, the basic process may be used for other types of studies than experiments, but it has to be tailored to the specific type of study being conducted, for example, a survey using e-mail or a case study of a large software project. The process is as it is presented, however, suited for both "true" experiments and quasi-experiments. The latter are often used in software engineering when random samples of, for example, subjects (participants) are infeasible.

The starting point for an experiment is insight, and the idea that an experiment would be a possible way of evaluating whatever we are interested in. In other words, we have to realize that an experiment is appropriate for the question we are going to investigate. This is by no means always obvious, in particular since empirical studies are not frequently used within computer science and software engineering [Tichy95, Zelkowitz98]. Some argumentation regarding why computer scientist should experiment more is provided in [Tichy98]. If we assume that we have realized, that an experiment is appropriate then it is important to plan the experiment carefully to avoid unnecessary mistakes, see Section 2.6.

The experiment process can be divided into the following main activities. The **definition** is the first step, where we define the experiment in terms of problem, objective and goals. The **planning** comes next, where the design of the experiment is determined, the instrumentation is considered and the threats to the experiment are evaluated. The **operation** of the experiment follows from the design. In the operational phase, measurements are collected which then are analyzed and evaluated in the **analysis and interpretation**. Finally, the results are presented and packaged in the **presentation and package**. The activities are illustrated in Figure 10, and further elaborated below, and then each of the activities is treated in-depth in Chapters 5-9. An overview of the experiment process including the activities, which are described in more detail in Chapters 5-9 can be found in Appendix B. The appendix, may serve as a quick reference to the process. It also provides references to the chapters and sections presenting the activities.

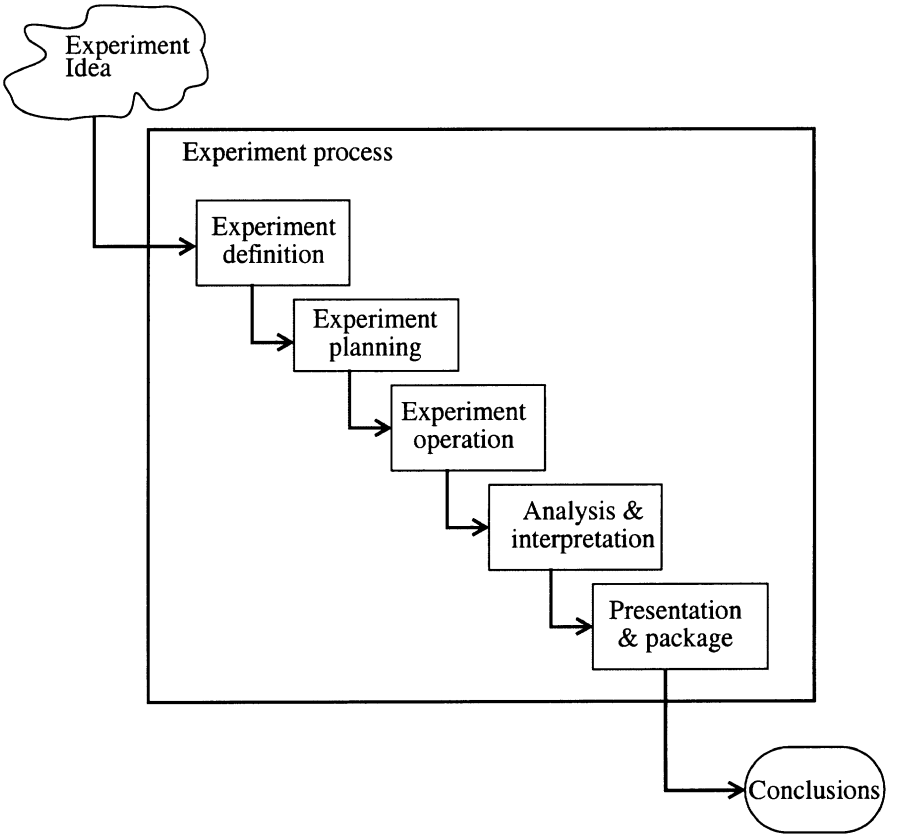


Figure 10. *Overview of the experiment process.*

The process is not supposed to be a "true" waterfall model; it is not assumed that an activity is necessarily finished prior to that the next activity is started. The order of activities in the process primarily indicates the starting order of the activities. In other words, the process is iterative and it may be necessary to go back and refine a previous activity before continuing with the experiment. The main exception is when the operation of the experiment has started, then it is not possible to go back to the definition and planning of the experiment. This is not possible since starting the operation means that the subjects are influenced by the experiment, and

if we go back there is risk that it is impossible to use the same subjects when returning to the operation phase of the experiment process.

Definition. The first activity is definition. The hypothesis has to be stated clearly. It does not have to be stated formally at this stage, but it has to be clear. Furthermore, the objective and goals of the experiment must be defined. The goal is formulated from the problem to be solved. In order to capture the definition, a framework has been suggested [Briand96b, Lott96]. The framework consists of the following constituents:

- object of study (what is studied?),
- purpose (what is the intention?),
- quality focus (which effect is studied?),
- perspective (whose view?), and
- context (where is the study conducted?).

These are further discussed in Chapter 5.

Planning. The planning activity is where the foundation for the experiment is laid. The context of the experiment is determined in detail. This includes personnel and the environment, for example, whether the experiment is run in a university environment with students or in an industrial setting. Moreover, the hypothesis of the experiment is stated formally, including a null hypothesis and an alternative hypothesis.

The next step in the planning activity is to determine variables (both independent variables (inputs) and the dependent variables (outputs). An important issue regarding the variables is to determine the values the variables actually can take. This also includes determining the measurement scale, which puts constraints on the method that we later can apply for statistical analysis. The subjects of the study are identified.

Furthermore, the experiment is designed, which includes choosing a suitable experiment design including, for example, randomization of subjects. An issue closely related to the design is to prepare for the instrumentation of the experiment. We must identify and prepare suitable objects, develop guidelines if necessary and define measurement procedures. These issues are further discussed in Chapter 6.

Finally, it is important to consider the question of validity of the results we can expect. Validity can be divided into four major classes: internal, external, construct and conclusion validity. Internal validity is concerned with the validity within the given environment and the reliability of the results. The external validity is a question of how general the findings are. Many times, we would like to state that the results from an experiment are valid outside the actual context in which the experiment was run. The construct validity is a matter of judging if the treatment reflects

the cause construct and the outcome provides a true picture of the effect construct, see Figure 7. The conclusion validity is concerned with the relationship between the treatment and the outcome of the experiment. We have to judge if there is a relationship between the treatment and the outcome.

The design is a crucial step in an experiment to ensure that the results from the experiment become useful. A poor design may ruin any well-intended study.

Operation. The operation consists in principle of three steps: preparation, execution and data validation. In the preparation step, we are concerned with preparing the subjects as well as the material needed, for example, data collection forms. The participants must be informed about the intention; we must have their consent and they must be committed. The actual execution is normally not a major problem. The main concern is to ensure that the experiment is conducted according to the plan and design of the experiment, which includes data collection. Finally, we must try to make sure that the actually collected data is correct and provide a valid picture of the experiment. The operation activity is discussed in Chapter 7.

Analysis and interpretation. The data collected during operation provide the input to this activity. The data can now be analyzed and interpreted. The first step in the analysis is to try to understand the data by using descriptive statistics. These provide a visualization of the data. The descriptive statistics help us to understand and interpret the data informally.

The next step is to consider whether the data set should be reduced, either by removing data points or by reducing the number of variables by studying if some of the variables provide the same information. Specific methods are available for data reduction.

After having removed data points or reduced the data set, we are able to perform a hypothesis test, where the actual test is chosen based on measurement scales, values on the input data and the type of results we are looking for. The statistical tests together with a more detailed discussion of descriptive statistics and data reduction techniques can be found in Chapter 8.

One important aspect of this activity is the interpretation. That is, we have to determine from the analysis whether the hypothesis was accepted or rejected. This forms the basis for decision-making and conclusions concerning how to use the results from the experiment, which includes motivation for further studies, for example, to conduct an enlarged experiment or a case study.

Two important methods related to analysis of the results are triangulation and meta-analysis. Triangulation is concerned with employing multiple methods to obtain a trustworthy result. Meta-analysis is used to summarize the results from several studies into one analysis, for example, combining several experiments into one analysis. Both these methods are discussed in more detail in, for example, [Robson93].

Presentation and package. The last activity is concerned with presenting and packaging of the findings. This includes primarily documentation of the results, which can be made either through a research paper for publication, a lab package for replication purposes or as part of a company's experience base. This last activity is important to make sure that the lessons learned are taken care of in an appropriate way. Moreover, an experiment will never provide the final answer to a question, and hence it is important to facilitate replication of the experiment. A comprehensive and thorough documentation is a prerequisite to achieve this objective. Thus, we must take some time after the experiment to document and present it in a proper way. The presentation of an experiment is further elaborated in Chapter 9.

The steps in this experiment process are described in more detail subsequently, and to support the understanding of the process, an example is presented in Chapter 11. The objective of the example is to closely follow the defined process in order to illustrate the use of it.

5 DEFINITION

Conducting an experiment is a labor-intensive task. In order to utilize the effort spent, it is important to ensure that the intention with the experiment can be fulfilled through the experiment.

In the definition phase the foundation of the experiment is determined, which is illustrated in Figure 11. If the foundation is not properly laid, rework may be required, or even worse, the experiment can not be used to study what was intended. The purpose of the definition phase is to define the goals of an experiment according to a defined framework. Here we follow the GQM template for goal definition [Briand96b, Lott96, Solingen99].

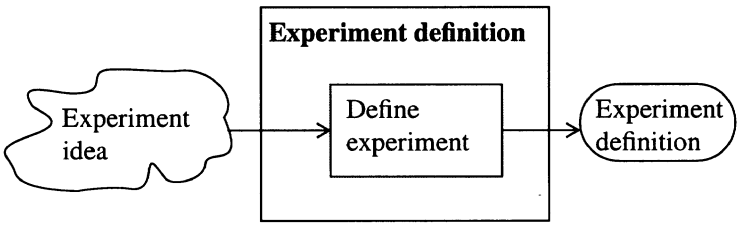


Figure 11. *Definition phase overview.*

The definition of an experiment is discussed in Section 5.1. An experiment goal definition example is presented in Section 5.2.

5.1 Define experiment

5.1.1 Goal definition template

The purpose of a goal definition template is to ensure that important aspects of an experiment are defined before the planning and execution take place. By defining the goal of the experiment according to this template, the foundation is properly laid. The goal template is:

Analyze <*Object(s) of study*>
for the purpose of <*Purpose*>
with respect to their <*Quality focus*>
from the point of view of the <*Perspective*>
in the context of <*Context*>.

Object of study. The object of study is the entity that is studied in the experiment. The object of study can be products, processes, resources, models, metrics or theories. Examples are the final product, the development or inspection process, or a reliability growth model.

Purpose. The purpose defines what the intention of the experiment is. It may be to evaluate the impact of two different techniques, or to characterize the learning curve of an organization.

Quality focus. This is the primary effect under study in the experiment. Quality focus may be effectiveness, cost, reliability etc.

Perspective. The perspective tells the viewpoint from which the experiment results are interpreted. Examples of perspectives are developer, project manager, customer and researcher.

Context. The context is the "environment" in which the experiment is run. The context briefly defines which personnel is involved in the experiment (subjects) and which software artifacts (objects[1]) are used in the experiment. Subjects can be characterized by experience, team size, workload etc. Objects can be characterized by size, complexity, priority, application domain etc.

The experiment context can be characterized in terms of the number of subjects and objects involved in the study, see Table 4.

Table 4. *Experiment context characterization*

		# Objects	
		One	**More than one**
# Subjects per object	**One**	Single object study	Multi-object variation study
	More than one	Multi-test within object study	Blocked subject-object study

Single object studies are conducted on a single subject and a single object. Multi-object variation studies are conducted on a single subject across a set of objects. Multi-test within object studies examines a single object across a set of subjects. Blocked subject-object studies examine a set of subjects and a set of objects. All these experiment types can be run either as an experiment or a quasi-experiment. In a quasi-experiment there is a lack of randomization of either subjects or objects. The single-object study is a quasi-experiment if the single subject and object are not selected by random, but it is an experiment if the subject and object are chosen by random. The difference between experiments and quasi-experiments is discussed further in [Robson93].

Examples of the different experiment types are given by the series of experiments conducted at NASA-SEL, aimed at evaluation of Cleanroom principles and techniques. Cleanroom is a collection of engineering methods and techniques assembled with the objective to produce high-quality software. A brief introduction to Cleanroom can be found in [Linger94]. The series consists of four distinct steps. First, a reading versus unit test experiment was conducted in a blocked subject-object study [Basili87], see 1 in Table 5. Secondly, a development project applying Cleanroom techniques was conducted in a student environment [Selby87]. The

1. Note that the "objects" here are generally different from the "objects of study" defined above.

experiment was a multi-test within object variation experiment, see 2 in Table 5. Thirdly, a project using Cleanroom was conducted at NASA-SEL [Basili94c] as a single object experiment, see 3 in Table 5. Fourthly, three Cleanroom projects were conducted in the same environment, constituting a multi-object variation study [Basili94c], see 4 in Table 5. The next round is a new reading experiment where different techniques are analyzed [Basili96a], see 5 in Table 5. This series of experiments is also discussed in [Linkman97].

Table 5. *Example experiment context characterization*

		# Objects	
		One	**More than one**
# Subjects per object	**One**	3. Cleanroom project no. 1 at SEL [Basili94c]	4. Cleanroom projects no. 2–4 at SEL [Basili94c]
	More than one	2. Cleanroom experiment at University of Maryland [Selby87]	1. Reading versus test [Basili87] 5. Scenario based reading vs. checklist [Basili96a]

The example, in Table 5, illustrates how experiments (see 1 and 2) can be conducted as prestudies prior to case studies (see 3 and 4). This is in line with the discussion regarding technology transfer and a suitable ordering based on cost and risk as discussed in Section 2.6.

5.2 Example

The goal definition framework can be filled out with different objects of study, purposes etc. In Table 6, examples of elements are given.

Table 6. *Definition framework.*

Object of study	Purpose	Quality focus	Perspective	Context
Product	Characterize	Effectiveness	Developer	Subjects
Process	Monitor	Cost	Modifier	Objects
Model	Evaluate	Reliability	Maintainer	
Metric	Predict	Maintainability	Project manager	
Theory	Control	Portability	Corporate manager	
	Change		Customer	
			User	
			Researcher	

A study definition example is constructed by composing the elements of the framework and is presented below. The example defines an inspection experiment where different inspection techniques were evaluated. The experiment is reported in [Regnell99].

Object of study. The objects studied are the Perspective-Based Reading (PBR) technique and a checklist-based technique.

Purpose. The purpose is to evaluate the reading techniques, in particular with respect to differences between perspectives in the PBR.

Quality focus. The quality focus is the effectiveness and efficiency of the reading techniques.

Perspective. The perspective is from the researcher's point of view.

Context. The experiment is run using M.Sc. and Ph.D. students as subjects based on a defined lab package with textual requirements documents. The study is conducted as a blocked subject-object study, see Table 4.

The example is summarized as:
Analyze *the PBR and checklist techniques*
for the purpose of *evaluation*
with respect to *effectiveness and efficiency*
from the point of view of *the researcher*
in the context of *M.Sc. and Ph.D. students reading requirements documents.*

5.3 Summary

The goal definition gives a target for the experiment. The object of study is defined to narrow its scope. The purpose and the quality focus are the basis for the hypothesis in the experiment. The perspective defines the view (normally in terms of role, for example, project manager) and the context relates to the validity of the results.

6 PLANNING

After the definition of the experiment, the planning takes place. The definition determines the foundation for the experiment – *why* the experiment is conducted – while the planning prepares for *how* the experiment is conducted.

As in all types of engineering activities, the experiment must be planned and the plans must be followed-up in order to control the experiment. The result of the experiment can be disturbed, or even destroyed if not planned properly.

The planning phase of an experiment can be divided into six steps. The input to the phase is the experiment definition, see Chapter 5. Based on the experiment definition, the **context selection** selects the environment in which the experiment will be executed. Next, the **hypothesis formulation** and the **variable selection** of independent and dependent variables take place. The **selection of subjects** is carried out. The **experiment design** is chosen based on the hypothesis and variables selected. Next the **instrumentation** prepares for the practical implementation of the experiment. Finally the **validity evaluation** aims at checking the validity of the experiment. The planning process is iterated until a complete experiment design is ready. An overview of the planning phase is given in Figure 12.

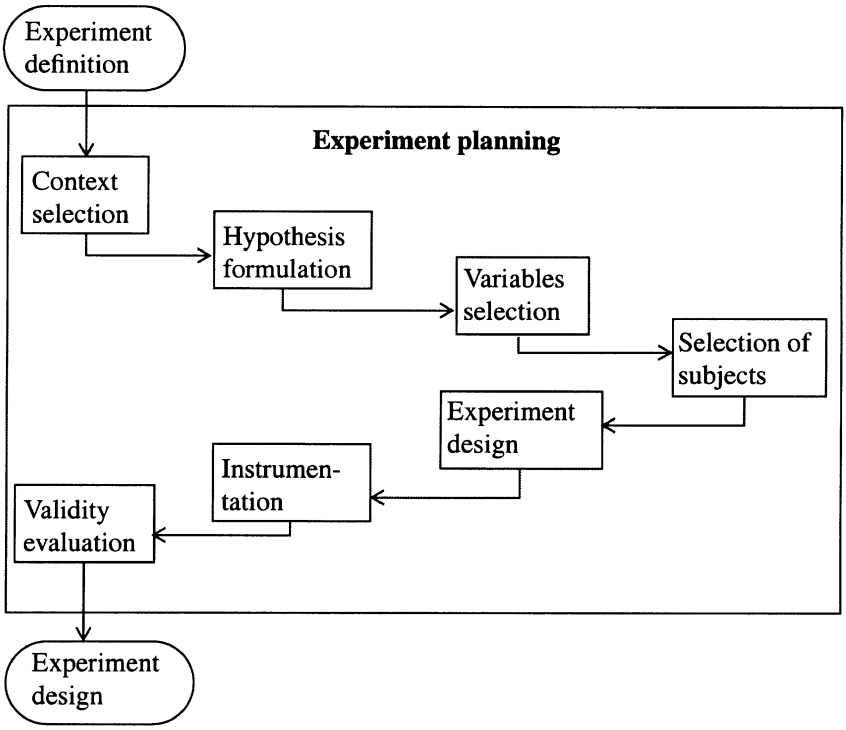


Figure 12. *Planning phase overview*

6.1 Context selection

In order to achieve the most general results in an experiment, it should be executed in large, real software projects, with professional staff. However, conducting an experiment involves risks, for example that the new method to be examined is not as good as expected and causes delays. An alternative is to run off-line projects in parallel with the real projects. This reduces the risks but causes extra costs. A cheaper alternative is to run projects staffed by students. Such projects are cheaper, easier to control, but more directed to a certain context than projects staffed by professionals with more and various experience. Furthermore these projects do seldom address real problems, but problems more of toy size due to constraints in cost and time. This trade-off involves a balance between making studies valid to a specific context or valid to the general software engineering domain.

Hence, the context of the experiment can be characterized according to four dimensions:

- Off-line vs. on-line.
- Student vs. professional.
- Toy vs. real problems.
- Specific vs. general.

A common situation in an experiment is that something existing is compared to something new, for example an existing inspection method is compared to a new one [Basili96a, Porter94, Porter97b]. There are two problems related to this type of studies. Firstly, what is the existing method? It has been applied for some period of time, but it is rarely well documented and there is no consistent application of the method. Secondly, learning a new method may influence how the old one is applied.

This and other issues related to that we are concerned with people have to be taken into account when planning for an experiment in order to make the results valid.

6.2 Hypothesis formulation

The basis for the statistical analysis of an experiment is hypothesis testing. A hypothesis is stated formally and the data collected during the course of the experiment is used to, if possible, reject the hypothesis. If the hypothesis can be rejected then conclusions can be drawn, based on the hypothesis testing under given risks.

6.2.1 Hypothesis statement

In the planning phase, the experiment definition is formalized into hypotheses. Two hypotheses have to be formulated:

- **A null hypothesis, H_0:** This hypothesis states that there are no real underlying trends or patterns in the experiment setting; the only reasons for differences in our observations are coincidental. This is the hypothesis that the experimenter wants to reject with as high significance as possible. An example hypothesis is that a new inspection method finds on average the same number of faults as the old one. H_0: $\mu_{N_{old}} = \mu_{N_{new}}$.
- **An alternative hypothesis, H_a, H_1, etc.:** This is the hypothesis in favor of which the null hypothesis is rejected. An example hypothesis is that a new inspection method on average finds more faults than the old one. H_1: $\mu_{N_{old}} < \mu_{N_{new}}$.

There are a number of different statistical tests described in the literature that can be used to evaluate the outcome of an experiment. They are all based on that the above hypotheses are formulated before the statistical tests are chosen and performed. The statistical tests are elaborated in Section 8.3.

Testing hypotheses involves different types of risks. Either the test rejects a true hypothesis or the test does not reject a false hypothesis. These risks are referred to as type-I-error and type-II-error.

- **Type-I-error**: A type-I-error has occurred when a statistical test has indicated a pattern or relationship even if there actually is no real pattern. That is, the probability of committing a type-I-error can be expressed as:

  $P\ (type\text{-}I\text{-}error) = P\ (reject\ H_0 \mid H_0\ true).$

  In the example hypothesis above, the type-I-error is the probability of rejecting H_0 even though the two methods on average find the same number of faults.

- **Type-II-error**: A type-II-error has occurred when a statistical test has not indicated a pattern or relationship even if there actually is a real pattern. That is, the probability of committing a type-II-error can be expressed as:

  $P\ (type\text{-}II\text{-}error) = P\ (not\ reject\ H_0 \mid H_0 false)$

  In the example hypothesis above, the type-II-error is the probability of not rejecting H_0 even though the two methods on average have different means.

The size of the errors depends on different factors. One example is the ability of the statistical test to reveal a true pattern in the collected data. This is referred to as the power of a test.

- **Power**: The power of a statistical test is the probability that the test will reveal a true pattern if H_0 is false. An experimenter should choose a test with as high power as possible. The power can be expressed as:

  $Power = P\ (reject\ H_0 \mid H_0 false) = 1 - P\ (type\text{-}II\text{-}error)$

  All these factors have to be considered when planning an experiment.

6.3 Variables selection

Before any design can start we have to choose the dependent and independent variables.

6.3.1 Choice of independent variables

The independent variables are those variables that we can control and change in the experiment. Choosing the right variables is not easy and it usually requires domain knowledge. The variables should have some effect on the dependent variable and must be controllable. The choices of the independent and dependent variables are often done simultaneously or in reverse order. The choice of independent variables also includes choosing the measurement scales, the range for the variables and the specific levels at which tests will be made.

6.3.2 Choice of dependent variables

The effect of the treatments is measured in the dependent variable or variables. Often there is only one dependent variable and it should therefore be derived directly from the hypothesis. The variable is mostly not directly measurable and we have to measure it via an indirect measure instead. This indirect measure must be carefully validated, because it affects the result of the experiment. The hypothesis can be refined when we have chosen the dependent variable. The choice of dependent variable also means that the measurement scale and range of the variables are determined.

6.4 Selection of subjects

The selection of subjects is important when conducting an experiment [Robson93]. The selection is closely connected to the generalization of the results from the experiment. In order to generalize the results to the desired population, the selection must be representative for that population. The selection of subjects is also called a sample from a population.

The sampling of the population can be either a probability or a non-probability sample. The difference between the two is that in the probability sampling, the probability of selecting each subject is known and in the non-probability sampling it is unknown. Examples of probability sampling techniques are:

- *Simple random sampling:* Subjects are selected from a list of the population at random.

- *Systematic sampling:* The first subject is selected from the list of the population at random and then every *n:th* person is selected from the list.
- *Stratified random sampling:* The population is divided into a number of groups or strata with a known distribution between the groups. Random sampling is then applied within the strata.

Examples of non-probability sampling techniques are:

- *Convenience sampling:* The nearest and most convenient persons are selected as subjects.
- *Quota sampling:* This type of sampling is used to get subjects from various elements of a population. Convenience sampling is normally used for each element.

The size of the sample also impacts the results when generalizing. The larger the sample is, the lower the error becomes when generalizing the results. The sample size is also closely related to the power of the statistical test. There are some general principles for choosing the sample size:

- If there is large variability in the population, a larger sample size is needed.
- The analysis of the data may influence the choice of the sample size. It is therefore needed to consider how the data shall be analyzed already at the design stage of the experiment.

6.5 Experiment design

To draw meaningful conclusions from an experiment, we apply statistical analysis methods on the collected data to interpret the results, as further described in Chapter 8. To get the most out of the experiment, it must be carefully planned and designed. Which statistical analyses we can apply depend on the chosen design, and the used measurement scales, see Chapter 3. Therefore design and interpretation are closely related.

Before any design is started, we have to recognize which problem the experiment shall investigate, see Sections 5.1 and 6.2. It is necessary to develop all ideas about the objectives of the experiment. To achieve a better understanding and to help the final solution, we have to make a clear definition.

6.5.1 Choice of experiment design

An experiment consists of a series of tests of the treatments. To get the most out of the experiment, the series of tests must be carefully planned and designed. A design of an experiment describes how the tests are organized and run. More formally, we can define an experiment as a set of tests.

As described above, the design and the statistical analysis are closely related. The choice of design affects the analysis and vice versa. To design the experiment, we have to look at the hypothesis to see which statistical analysis we have to perform to reject the null hypothesis. Based on the statistical assumptions, e.g. the measurement scales, and on which objects and subjects we are able to use, we make the experiment design. During the design we determine how many tests the experiment shall have to make sure that the effect of the treatment is visible. A proper design also forms the basis to allow for replication.

In the following two sections, general design principles and some standard design types are presented.

6.5.2 General design principles

When designing an experiment, many aspects must be considered. The general design principles are randomization, blocking and balancing, and most experiment designs use some combination of these. To illustrate the general design principles, we use the following example:

A company will conduct an experiment to investigate the effect on the reliability of a program when using object-oriented design instead of the standard company design principle. The experiment will use program A as the experiment object. The experiment design is of type "multi-test within object study", see Chapter 5.

The three general design principles can be described as:

- **Randomization**. One of the most important design principles is randomization. All statistical methods used for analyzing the data require that the observations be from independent random variables. To meet this requirement, randomization is used. The randomization applies on the allocation of the objects, subjects and in which order the tests are performed. Randomization is used to average out the effect of a factor that may otherwise be present. Randomization is also used to select subjects that is representative of the population of interest.

  Example: The selection of the persons (subjects) will be representative of the designers in the company, by random selection of the available designers. The assignment to each treatment (object-oriented design or the standard company design principle) is selected randomly.

- **Blocking**. Sometimes we have a factor that probably has an effect on the response, but we are not interested in that effect. If the effect of the factor is known and controllable, we can use a design technique called blocking. Blocking is used to systematically eliminate the undesired effect in the comparison among the treatments. Within one block, the undesired effect is the same and we can study the effect of the treatments on that block. Blocking is used to eliminate the undesired effect in the study and therefore the effects between the blocks are not studied. This technique increases the precision of the experiment.

  Example: The persons (subjects) used, for this experiment, have different experience. Some of them have used object-oriented design before and some have not. To minimize the effect of the experience, the persons are grouped into two groups (blocks), one with experience of object-oriented design and one without.
- **Balancing**. If we assign the treatments so that each treatment has equal number of subjects, we have a balanced design. Balancing is desirable because it both simplifies and strengthens the statistical analysis of the data, but it is not necessary.

  Example: The experiment uses a balanced design, which means that there is the same number of persons in each group (block).

6.5.3 Standard design types

In this section some of the most frequently used experiment designs are presented. The designs range from simple experiments with a single factor to more complex experiments with many factors. Experiment design is also discussed in, for example [Montgomery97]. For most of the designs, an example hypothesis is formulated and statistical analysis methods are suggested for each design. The design types presented in this section are suitable for experiments with:

- One factor with two treatments.
- One factor with more than two treatments.
- Two factors with two treatments.
- More than two factors each with two treatments.

One factor with two treatments. With these experiments, we want to compare the two treatments against each other. The most common is to compare the means of the dependent variable for each treatment.

The following notations are used:

μ_i	The mean of the dependent variable for treatment i.
y_{ij}	The j:th measure of the dependent variable for treatment i.

Example of an experiment: To investigate if a new design method produces software with higher quality than the previously used design method. The factor in this experiment is the design method and the treatments are the new and the old design method. The independent variable can be the number of faults found in development.

- Completely randomized design:

  This is a simple experiment design for comparing two treatment means. The design setup uses the same objects for both treatments and assigns the subjects randomly to each treatment, see Table 7. Each subject uses only one treatment on one object. If we have the same number of subjects per treatment the design is balanced.

  Table 7. *Example of assigning subjects to the treatments for a randomized design.*

Subjects	Treatment 1	Treatment 2
1	X	
2		X
3		X
4	X	
5		X
6	X	

  Example of hypothesis: H_0: $\mu_1 = \mu_2$
  H_1: $\mu_1 \neq \mu_2$, $\mu_1 < \mu_2$ or $\mu_1 > \mu_2$

  Examples of analysis: t-test,
  Mann-Whitney, see Section 8.3.

- Paired comparison design:

  We can sometimes improve the precision of the experiment by making comparisons within matched pairs of experiment material. In this design, each subject uses both treatments on the same object. To minimize the effect of the order, in which the subjects apply the treatments, the order is assigned randomly to each subject, see Table 8. This design cannot be applied in every case of comparison as the subject can gain too much information from the first treatment to perform the experiment with the second treatment. The comparison for the experiment

can be to see if the difference between the paired measures is zero. If we have the same number of subjects starting with the first treatment as with the second, we have a balanced design.

Table 8. *Example of assigning the treatments for a paired design.*

Subjects	Treatment 1	Treatment 2
1	2	1
2	1	2
3	2	1
4	2	1
5	1	2
6	1	2

Example of hypothesis: $d_j = y_{1j} - y_{2j}$ and μ_d is the mean of the difference.
H_0: $\mu_d = 0$
H_1: $\mu_d \neq 0$, $\mu_d < 0$ or $\mu_d > 0$

Examples of analysis: Paired t-test,
Sign test,
Wilcoxon, see Section 8.3.

One factor with more than two treatments. As with experiments with only two treatments, we want to compare the treatments with each other. The comparison is often performed on the treatment means.

Example of an experiment: The experiment investigates the quality of the software when using different programming languages. The factor in the experiment is the programming language and the treatments can be C, C++, and Java.

- Completely randomized design:

  A completely randomized design requires that the experiment is performed in random order so that the treatments are used in an environment as uniform as possible. The design uses one object to all treatments and the subjects are assigned randomly to the treatments, see Table 9.

Table 9. *Example of assigning the treatments to the subjects.*

Subjects	Treatment 1	Treatment 2	Treatment 3
1		X	
2			X
3	X		
4	X		
5		X	
6			X

Example of hypothesis: H_0: $\mu_1 = \mu_2 = \mu_3 = ... = \mu_a$
H_1: $\mu_i \neq \mu_j$ for at least one pair (i, j)

Example of analysis: ANOVA (ANalysis Of VAriance), see Section 8.3.
Kruskal-Wallis, see Section 8.3.

- Randomized complete block design:

  If the variability between the subjects is large, we can minimize this effect on the result by using a randomized complete block design. With this design, each subject uses all treatments and the subjects form a more homogeneous experiment unit, i.e. we block the experiment on the subjects, see Table 10. The blocks represent a restriction on randomization. The experiment design uses one object to all treatments and the order in which the subjects use the treatments are assigned randomly. The paired comparison design above is a special case of this design with only two treatments. The randomized complete block design is one of the most used experiment designs.

Table 10. *Example of assigning the treatments for a randomized complete block design.*

Subjects	Treatment 1	Treatment 2	Treatment 3
1	1	3	2
2	3	1	2
3	2	3	1
4	2	1	3
5	3	2	1
6	1	2	3

Example of hypothesis: H_0: $\mu_1 = \mu_2 = \mu_3 = ... = \mu_a$
H_1: $\mu_i \neq \mu_j$ for at least one pair *(i,j)*

Example of analysis: ANOVA (ANalysis Of VAriance), see Section 8.3.
Kruskal-Wallis, see Section 8.3.

Two factors. The experiment gets more complex when we increase from one factor to two. The single hypothesis for the experiments with one factor will split into three hypotheses: one hypothesis for the effect from one of the factors, one for the other and one for the interaction between the two factors. We use the following notations:

τ_i The effect of treatment *i* on factor A.

β_j The effect of treatment *j* on factor B.

$(\tau\beta)_{ij}$ The effect of the interaction between τ_i and β_j.

- 2*2 factorial design:

  This design has two factors, each with two treatments. In this experiment design, we randomly assign subjects to each combination of the treatments, see Table 11.

  Example of an experiment: The experiment investigates the understandability of the design document when using structured or object-oriented design based on one 'good' and one 'bad' requirements documents. The first factor, A, is the design method and the second factor, B, is the requirements document. The experiment design is a 2*2 factorial design as both factors have two treatments and every combination of the treatments are possible.

Table 11. *Example of a 2*2 factorial design.*

		Factor A	
		Treatment A1	**Treatment A2**
Factor B	**Treatment B1**	Subject: 4,6	Subject: 1,7
	Treatment B2	Subject: 2,3	Subject: 5,8

Example of hypothesis: H_0: $\tau_1 = \tau_2 = 0$
H_1: *at least one* $\tau_i \neq 0$

H_0: $\beta_1 = \beta_2 = 0$
H_1: *at least one* $\beta_j \neq 0$

H_0: $(\tau\beta)_{ij} = 0$ *for all i, j*
H_1: *at least one* $(\tau\beta)_{ij} \neq 0$

Example of analysis: ANOVA (ANalysis Of VAriance), see Section 8.3.

- Two-stage nested design:

  If one of the factors, for example B, in the experiment is similar but not identical for different treatments of the other factor, for example A, we have a design that is called nested or hierarchical design. Factor B is said to be nested under factor A. The two-stage nested design has two factors, each with two or more treatments. The experiment design and analysis are the same as for the 2*2 factorial design, see Table 12.

  Example of an experiment: The experiment investigates the test efficiency of unit testing of a program when using function or object-oriented programming and if the programs are 'defect-prone' or 'non-defect-prone'. The first factor, A, is the programming language and the second factor, B, is the defect-proneness of the program. The experiment design has to be nested, as a 'defect-prone/non-defect-prone' functional program is not the same as a 'defect-prone/non-defect-prone' object-oriented program.

Table 12. *Example of a two-stage nested design where B is nested under A.*

Factor A			
Treatment A1		**Treatment A2**	
Factor B		**Factor B**	
Treatment B1'	**Treatment B2'**	**Treatment B1"**	**Treatment B2"**
Subject: 1,3	Subject: 6,2	Subject: 7,8	Subject 5,4

More than two factors. In many cases, the experiment has to consider more than two factors. The effect in the dependent variable can therefore be dependent not only on each factor separately but also on the interactions between the factors. These interactions can be between two or more factors. This type of designs is called factorial designs. This section gives an introduction to designs where each factor has only two treatments each. Designs where the factors have more than two treatments can be found in [Montgomery97].

- 2^k factorial design:

  The 2*2 factorial design is a special case of the 2^k factorial design, i.e. when k=2. The 2^k factorial design has k factors where each factor has two treatments. This means that there are 2^k different combinations of the treatments. To evaluate the effects of the k factors, all combinations have to be tested. The subjects are randomly assigned to the different combinations. An example of a 2^3 factorial design is shown in Table 13.

  The hypotheses and the analyses for this type of design are of the same type as for the 2*2 factorial design. More details about the 2^k factorial design can be found in [Montgomery97].

Table 13. *Example of a 2^3 factorial design.*

Factor A	Factor B	Factor C	Subjects
A1	B1	C1	2, 3
A2	B1	C1	1, 13
A1	B2	C1	5, 6
A2	B2	C1	10, 16
A1	B1	C2	7, 15
A2	B1	C2	8, 11
A1	B2	C2	4, 9
A2	B2	C2	12, 14

- 2^k fractional factorial design:

  When the number of factor grows in a 2^k factorial design, the number of factor combinations grows rapidly, for example, there are 8 combinations for a 2^3 factorial design and 16 for a 2^4 factorial design. Often, it can be assumed that the effects of certain high-order interactions are negligible and that the main effects and the low-order interaction effects can be obtained by running a fraction of the complete factorial experiment. This type of design is therefore called fractional factorial design.

  The fractional factorial design is based on three ideas:

  The sparsity of effect principal: It is likely that the system is primarily driven by some of the main and low-order interaction effects.

  The projection property: A stronger design can be obtained by taking a subset of significant factors from the fractional factorial design.

Sequential experimentation: A stronger design can be obtained by combining sequential runs of two or more fractional factorial designs.

The major use of these fractional factorial designs is in screening experiments, where the purpose of the experiment is to identify the factors that have large effects on the system. Examples of fractional factorial designs are:

One-half fractional factorial design of the 2^k factorial design: Half of the combinations of a full 2^k factorial design is chosen. The combinations are selected so that if one factor is removed the remaining design is a full 2^{k-1} factorial design, see Table 14. The subjects are randomly assigned to the selected combinations. There are two alternative fractions in this design and if both fractions are used in sequence, the resulting design is a full 2^k factorial design.

Table 14. *Example of an one-half fraction of the 2^3 factorial design.*

Factor A	Factor B	Factor C	Subjects
A1	B1	C2	2, 3
A2	B1	C1	1, 8
A1	B2	C1	5, 6
A2	B2	C2	4, 7

One-quarter fractional factorial design of the 2^k factorial design: One quarter of the combinations of the full 2^k factorial design is chosen. The combinations are selected so that if two factors are removed the remaining design is a full 2^{k-2} factorial design, see Table 15. There are however dependencies between the factors in the one-quarter design due to that it is not a full factorial design.

For example, in Table 15, factor D is dependent on a combination of factor A and B. It can, for example, be seen that for all combinations of A1 and B1, we have D2, and so forth. In a similar way, factor E is dependent on a combination of factor A and C. Thus, if factor C and E (or B and D) are removed, the resulting design becomes two replications of a 2^{3-1} fractional factorial design and not a 2^3 factorial design. The latter design is obtained if D and E are removed. The two replications can be identified in Table 15 by noticing that the first four rows are equivalent to the four last rows in the table, when C and E are removed, and hence it becomes two replications of a 2^2 factorial design.

The subjects are randomly assigned to the selected combinations. There are four alternative fractions in this design and if all four fractions are used in sequence, the resulting design is a full 2^k factorial design. If two of the fractions are used in sequence a one-half fractional design is achieved.

More details on the fractional factorial designs can be found in [Montgomery97].

Table 15. *Example of an one-quarter fraction of the 2^5 factorial design.*

Factor A	Factor B	Factor C	Factor D	Factor E	Subjects
A1	B1	C1	D2	E2	3, 16
A2	B1	C1	D1	E1	7, 9
A1	B2	C1	D1	E2	1, 4
A2	B2	C1	D2	E1	8, 10
A1	B1	C2	D2	E1	5, 12
A2	B1	C2	D1	E2	2, 6
A1	B2	C2	D1	E1	11, 15
A2	B2	C2	D2	E2	13, 14

In summary, the choice of the correct experimental design is crucial, since a poor design will undoubtedly affect the possibility of being able to draw the correct conclusions after the study. Furthermore, the design puts constraints on the statistical methods that can be applied. Finally, it should be stressed that it is important to try to use a simple design if possible and try to make the best possible use of the available subjects.

6.6 Instrumentation

The instruments for an experiment are of three types, namely objects, guidelines and measurement instruments. In the planning of an experiment, the instruments are chosen. Before execution, the instruments are developed for the specific experiment.

Experiment objects may be, for example, specification or code documents. When planning for an experiment, it is important to choose objects that are appropriate. For example, in an inspection experiment, the number of defects must be known in the inspection objects. This can be achieved by seeding defects or by using a document with a known number of defects. Using a true early version of a document in which the defects are identified can do the latter.

Guidelines are needed to guide the participants in the experiment. Guidelines include, for example, process descriptions and checklists. If different methods are compared in the experiment, guidelines for the methods have to be prepared for the

experiment. In addition to the guidelines, the participants also need training in the methods to be used.

Measurements in an experiment are conducted via data collection. In human-intensive experiments, data is generally collected via manual forms or in interviews. The planning task to be performed is to prepare forms and interview questions and to validate the forms and questions with some people having similar background and skills as the experiment participants. An example of a form used to collect information about the experience of subjects is shown among the exercises, see Table 37 in Chapter 13.

The overall goal of the instrumentation is to provide means for performing the experiment and to monitor it, without affecting the control of the experiment. The results of the experiment shall be the same independently of how the experiment is instrumented. If the instrumentation affects the outcome of the experiment, the results are invalid.

The validity of an experiment is elaborated in Section 6.7 and more about the preparation of instruments can be found in Sections 7.1.2 and 7.2.2.

6.7 Validity evaluation

A fundamental question concerning results from an experiment is how valid the results are. It is important to consider the question of validity already in the planning phase in order to plan for adequate validity of the experiment results. Adequate validity refers to that the results should be valid for the population of interest. First of all, the results should be valid for the population from which the sample is drawn. Secondly, it may be of interest to generalize the results to a broader population. The results are said to have adequate validity if they are valid for the population to which we would like to generalize.

Adequate validity does not necessarily imply most general validity. An experiment conducted within an organization may be designed to answer some questions for that organization exclusively, and it is sufficient if the results are valid within that specific organization. On the other hand, if more general conclusions shall be drawn, the validity must cover a more general scope as well.

There are different classification schemes for different types of threats to the validity of an experiment. Campbell and Stanley define two types, threats to internal and external validity [Campbell63]. Cook and Campbell extend the list to four types of threats to the validity of experimental results. The four threats are *conclusion, internal, construct* and *external validity* [Cook79]. The former categorization is sometimes referred to in the literature, but the latter is preferable since it is easily mapped to the different steps involved when conducting an experiment, see Figure 13.

Each of the four categories presented in [Cook79] is related to a methodological question in experimentation. The basic principles of an experiment are presented in Figure 13.

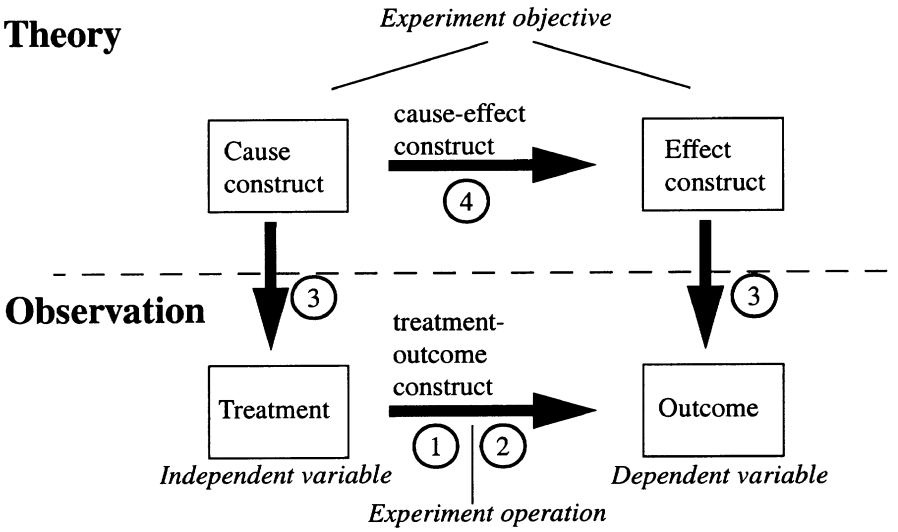


Figure 13. *Experiment principles (adapted from [Trochim99]).*

On the top, we have the *theory* area, and on the bottom, the *observation* area. We want to draw conclusions about the theory defined in the hypotheses, based on our observations. In drawing conclusions we have four steps, in each of which there is one type of threat to the validity of the results.

1. **Conclusion validity.** This validity is concerned with the relationship between the treatment and the outcome. We want to make sure that there is a statistical relationship, i.e. with a given significance.
2. **Internal validity.** If a relationship is observed between the treatment and the outcome, we must make sure that it is a causal relationship, and that it is not a result of a factor of which we have no control or have not measured. In other words that the treatment causes the outcome (the effect).
3. **Construct validity.** This validity is concerned with the relation between theory and observation. If the relationship between cause and effect is causal, we must ensure two things: 1) that the treatment reflects the construct of the cause well (see left part of Figure 13) and 2) that the outcome reflects the construct of the effect well (see right part of Figure 13).

4. **External validity.** The external validity is concerned with generalization. If there is a causal relationship between the construct of the cause, and the effect, can the result of the study be generalized outside the scope of our study? Is there a relation between the treatment and the outcome?

Conclusion validity is sometimes referred to as statistical conclusion validity [Cook79]. Threats to conclusion validity are concerned with issues that affect the ability to draw the correct conclusion about relations between the treatment and the outcome of an experiment. These issues include, for example, choice of statistical tests, choice of sample sizes, care taken in the implementation and measurement of an experiment.

Threats to internal validity concern issues that may indicate a causal relationship, although there is none. Factors that impact on the internal validity are how the subjects are selected and divided into different classes, how the subjects are treated and compensated during the experiment, if special events occur during the experiment etc. All these factors can make the experiment show a behavior that is not due to the treatment but to the disturbing factor.

Threats to construct validity refer to the extent to which the experiment setting actually reflects the construct under study. For example, the number of courses taken at the university in computer science may be a poor measure of the subject's experience in a programming language, i.e. has poor construct validity. The number of years of practical use may be a better measure, i.e. has better construct validity.

Threats to external validity concern the ability to generalize experiment results outside the experiment setting. External validity is affected by the experiment design chosen, but also by the objects in the experiment and the subjects chosen. There are three main risks: having wrong participants as subjects, conducting the experiment in the wrong environment and performing it with a timing that affects the results.

A detailed list of threats to the validity is presented in Section 6.8. This list can be used as a checklist for an experiment design. In the validity evaluation, each of the items is checked to see if there are any threats. If there are any, they have to be addressed or accepted, since sometimes some threat to validity has to be accepted. It may even be impossible to carry out an experiment without certain threats and hence they have to be accepted and then addressed when interpreting the results. The priority between different types of threats is further discussed in Section 6.9.

6.8 Detailed description of validity threats

Below, a list of threats to the validity of experiments is discussed based on [Cook79]. All threats are not applicable to all experiments, but this list can be seen as a checklist. The threats are summarized in Table 16 and the alternative and limited classification scheme [Campbell63] is summarized in Table 17.

Table 16. *Threats to validity according to [Cook79]*

Conclusion validity	**Internal validity**
Low statistical power	History
Violated assumption of statistical tests	Maturation
Fishing and the error rate	Testing
Reliability of measures	Instrumentation
Reliability of treatment implementation	Statistical regression
Random irrelevancies in experimental setting	Selection
Random heterogeneity of subjects	Mortality
	Ambiguity about direction of causal influence
	Interactions with selection
	Diffusion of imitation of treatments
	Compensatory equalization of treatments
	Compensatory rivalry
	Resentful demoralization

Construct validity	**External validity**
Inadequate preoperational explication of constructs	Interaction of selection and treatment
Mono-operation bias	Interaction of setting and treatment
Mono-method bias	Interaction of history and treatment
Confounding constructs and levels of constructs	
Interaction of different treatments	
Interaction of testing and treatment	
Restricted generalizability across constructs	
Hypothesis guessing	
Evaluation apprehension	
Experimenter expectancies	

Table 17. *Threats to validity according to [Campbell63]*

Internal validity	External validity
History	Interaction of selection and treatment
Maturation	Interaction of history and treatment
Testing	Interaction of setting and treatment
Instrumentation	Interaction of different treatments
Statistical regression	
Selection	

6.8.1 Conclusion validity

Threats to the conclusion validity are concerned with issues that affect the ability to draw the correct conclusion about relations between the treatment and the outcome of an experiment.

- **Low statistical power.** The power of a statistical test is the ability of the test to reveal a true pattern in the data. If the power is low, there is a high risk that an erroneous conclusion is drawn, see further Section 6.2.1, or more specifically we are unable to reject an erroneous hypothesis.
- **Violated assumptions of statistical tests.** Certain tests have assumptions on, for example, normally distributed and independent samples. Violating the assumptions may lead to wrong conclusions. Some statistical tests are more robust to violated assumptions than others are, see Chapter 8.
- **Fishing and the error rate.** This threat contains two separate parts. Searching or "fishing" for a specific result is a threat, since the analyses are no longer independent and the researchers may influence the result by looking for a specific outcome. The error rate is concerned with the actual significance level. For example, conducting three investigations with a significance level of 0.05 means that the total significance level is $(1–0.05)^3$, which equals 0.14. The error rate (i.e. significance level) should thus be adjusted when conducting multiple analyses.
- **Reliability of measures.** The validity of an experiment is highly dependent on the reliability of the measures. This in turn may depend on many different factors, like poor question wording, bad instrumentation or bad instrument layout. The basic principle is that when you measure a phenomenon twice, the outcome shall be the same. For example, lines of code are more reliable than function points since it does not involve human judgement. In other words, objective measures, that can be repeated with the same outcome, are more reliable than subjective measures, see also Chapter 3.

- **Reliability of treatment implementation.** The implementation of the treatment means the application of treatments to subjects. There is a risk that the implementation is not similar between different persons applying the treatment or between different occasions. The implementation should hence be as standard as possible over different subjects and occasions.
- **Random irrelevancies in experimental setting.** Elements outside the experimental setting may disturb the results, such as noise outside the room or a sudden interrupt in the experiment.
- **Random heterogeneity of subjects.** There is always heterogeneity in a study group. If the group is very heterogeneous, there is a risk that the variation due to individual differences is larger than due to the treatment. Choosing more homogeneous groups will on the other hand affect the external validity, see below. For example an experiment with undergraduate students reduces the heterogeneity, since they have more similar knowledge and background, but also reduces the external validity of the experiment, since the subjects are not selected from a general enough population.

6.8.2 Internal validity

Threats to internal validity are influences that can affect the independent variable with respect to causality, without the researcher's knowledge. Thus they threat the conclusion about a possible causal relationship between treatment and outcome. The internal validity threats are sometimes sorted into three categories, *single group threats, multiple group threats* and *social threats*.

Single group threats. These threats apply to experiments with single groups. We have no control group to which we do not apply the treatment. Hence, there are problems in determining if the treatment or another factor caused the observed effect.

- **History**. In an experiment, different treatments may be applied to the same object at different times. Then there is a risk that the history affects the experimental results, since the circumstances are not the same on both occasions. For example if one of the experiment occasions is on the first day after a holiday or on a day when a very rare event takes place, and the other occasion is on a normal day.
- **Maturation**. This is the effect of that the subjects react differently as time passes. Examples are when the subjects are affected negatively (tired or bored) during the experiment, or positively (learning) during the course of the experiment.

- **Testing**. If the test is repeated, the subjects may respond differently at different times since they know how the test is conducted. If there is a need for familiarization to the tests, it is important that the results of the test are not fed back to the subject, in order not to support unintended learning.
- **Instrumentation**. This is the effect caused by the artifacts used for experiment execution, such as data collection forms, document to be inspected in an inspection experiment etc. If these are badly designed, the experiment is affected negatively.
- **Statistical regression**. This is a threat when the subjects are classified into experimental groups based on a previous experiment or case study, for example top-ten or bottom-ten. In this case there might be an increase or improvement, even if no treatment is applied at all. For example if the bottom-ten in an experiment are selected as subjects based on a previous experiment, all of them will probably *not* be among the bottom-ten in the new experiment due to pure random variation. The bottom-ten cannot be worse than remain among the bottom-ten, and hence the only possible change is to the better, relatively the larger population from which they are selected.
- **Selection**. This is the effect of natural variation in human performance. Depending on how the subjects are selected from a larger group, the selection effects can vary. Furthermore, the effect of letting volunteers take part in an experiment may influence the results. Volunteers are generally more motivated and suited for a new task than the whole population. Hence the selected group is not representative for the whole population.
- **Mortality**. This effect is due to the different kinds of persons who drop out from the experiment. It is important to characterize the dropouts in order to check if they are representative of the total sample. If subjects of a specific category drop out, for example, all the senior reviewers in an inspection experiment, the validity of the experiment is highly affected.
- **Ambiguity about direction of causal influence**. This is the question of whether A causes B, B causes A or even X causes A and B. An example is if a correlation between program complexity and error rate is observed. The question is if high program complexity causes high error rate, or vice versa, or if high complexity of the problem to be solved causes both.

Most of the threats to internal validity can be addressed through the experiment design. For example, by introducing a control group many of the internal threats can be controlled. On the other hand, multiple group threats are introduced instead.

Multiple groups threats. In a multiple groups experiment, different groups are studied. The threat to such studies is that the control group and the selected experi-

ment groups may be affected differently by the single group threats as defined above. Thus there are interactions with the selection.

- **Interactions with selection.** The interactions with selection are due to different behavior in different groups. For example, the selection-maturation interaction means that different groups mature at different speed, for example if two groups apply one new method each. If one group learns its new method faster than the other, due to its learning ability, does, the selected groups mature differently. Selection-history means that different groups are affected by history differently, etc.

Social threats to internal validity. These threats are applicable to single group and multiple group experiments. Examples are given below from an inspection experiment where a new method (perspective-based reading) is compared to an old one (checklist-based reading).

- **Diffusion or imitation of treatments.** This effect occurs when a control group learns about the treatment from the group in the experiment study or they try to imitate the behavior of the group in the study. For example, if a control group uses a checklist-based inspection method and the experiment group uses perspective-based methods, the former group may hear about the perspective-based method and perform their inspections influenced by their own perspective. The latter may be the case if the reviewer is an expert in a certain area.
- **Compensatory equalization of treatments.** If a control group is given compensation for being a control group, as a substitute for that they do not get treatments; this may affect the outcome of the experiment. If the control group is taught another new method as a compensation for not being taught the perspective-based method, their performance may be affected by that method.
- **Compensatory rivalry.** A subject receiving less desirable treatments may, as the natural underdog, be motivated to reduce or reverse the expected outcome of the experiment. The group using the traditional method may do their very best to show that the old method is competitive.
- **Resentful demoralization.** This is the opposite of the previous threat. A subject receiving less desirable treatments may give up and not perform as good as it generally does. The group using the traditional method is not motivated to do a good job, while learning something new inspires the group using the new method.

6.8.3 Construct validity

Construct validity concerns generalizing the result of the experiment to the concept or theory behind the experiment. Some threats relate to the design of the experiment, others to social factors.

Design threats. The design threats to construct validity cover issues that are related to the design of the experiment and its ability to reflect the construct to be studied.

- **Inadequate preoperational explication of constructs**. This threat, despite its extensive title, is rather simple. It means that the constructs are not sufficiently defined, before they are translated into measures or treatments. The theory is not clear enough, and hence the experiment cannot be sufficiently clear. For example, if two inspection methods are compared and it is not clearly enough stated what being "better" means. Does it mean to find most faults, most faults per hour or most serious faults?
- **Mono-operation bias**. If the experiment includes a single independent variable, case, subject or treatment, the experiment may under-represent the construct and thus not give the full picture of the theory. For example, if an inspection experiment is conducted with a single document as object, the cause construct is under-represented.
- **Mono-method bias**. Using a single type of measures or observations involves a risk that if this measure or observation gives a measurement bias, then the experiment will be misleading. By involving different types of measures and observations they can be cross-checked against each other. For example, if the number of faults found is measured in an inspection experiment, where fault classification is based on subjective judgement, the relations cannot be sufficiently explained. The experimenter may bias the measures.
- **Confounding constructs and levels of constructs**. In some relations it is not primarily the presence or absence of a construct, but the level of the construct which is of importance to the outcome. The effect of the presence of the construct is confounded with the effect of the level of the construct. For example, the presence or absence of prior knowledge in a programming language may not explain the causes in an experiment, but the difference may depend on if the subjects have 1, 3 or 5 years of experience with the current language.
- **Interaction of different treatments**. If the subject is involved in more than one study, treatments from the different studies may interact. Then you cannot conclude whether the effect is due to either of the treatments or of a combination of treatments.

- **Interaction of testing and treatment**. The testing itself, i.e. the application of treatments, may make the subjects more sensitive or receptive to the treatment. Then the testing is a part of the treatment. For example, if the testing involves measuring the number of errors made in coding, then the subjects will be more aware of their errors made, and thus try to reduce them.
- **Restricted generalizability across constructs.** The treatment may affect the studied construct positively, but unintendedly affect other constructs negatively. This threat makes the result hard to generalize into other potential outcomes. For example, a comparative study concludes that improved productivity is achieved with a new method. On the other hand, it can be observed that it reduces the maintainability, which is an unintended side effect. If the maintainability is not measured or observed, there is a risk that conclusions are drawn based on the productivity attribute, ignoring the maintainability.

Social threats to construct validity. These threats are concerned with issues related to behavior of the subjects and the experimenters. They may, based on the fact that they are part of an experiment, act differently than they do otherwise, which gives false results from the experiment.

- **Hypothesis guessing**. When people take part in an experiment they might try to figure out what the purpose and intended result of the experiment is. Then they are likely to base their behavior on their guesses about the hypotheses, either positively or negatively, depending on their attitude to the anticipated hypothesis.
- **Evaluation apprehension**. Some people are afraid of being evaluated. A form of human tendency is to try to look better when being evaluated which is confounded to the outcome of the experiment. For example, if different estimation models are compared, people may not report their true deviations between estimate and outcome, but some false but "better" values.
- **Experimenter expectancies**. The experimenters can bias the results of a study both consciously and unconsciously based on what they expect from the experiment. The threat can be reduced by involving different people which have no or different expectations to the experiment. For example, questions can be raised in different ways in order to give the answers you want.

6.8.4 External validity

Threats to external validity are conditions that limit our ability to generalize the results of our experiment to industrial practice. There are three types of interactions with the treatment: people, place and time:

- **Interaction of selection and treatment.** This is an affect of having a subject population, not representative of the population we want to generalize to, i.e. the wrong people participate in the experiment. An example of this threat is to select only programmers in an inspection experiment when programmers as well as testers and system engineers generally take part in the inspections.
- **Interaction of setting and treatment.** This is the effect of not having the experimental setting or material representative of, for example, industrial practice. An example is using old-fashioned tools in an experiment when up-to-date tools are common in industry. Another example is conducting experiment on toy problems. This means wrong "place" or environment.
- **Interaction of history and treatment.** This is the effect of that the experiment is conducted on a special time or day which affects the results. If, for example, a questionnaire is conducted on safety-critical systems a few days after a big software-related crash, people tend to answer differently than a few days before, or some weeks or months later.

The threats to external validity are reduced by making the experimental environment as realistic as possible. On the other hand, reality is not homogenous. Most important is to characterize and report the characteristics of the environment, such as staff experience, tools, methods in order to evaluate the applicability in a specific context.

6.9 Priority among types of validity threats

There is a conflict between some of the types of validity threats. The four types considered are internal validity, external validity, conclusion validity and construct validity. When increasing one type, another type may decrease. Prioritizing among the validity types is hence an optimization problem, given a certain purpose of the experiment.

For example, using undergraduate students in an inspection experiment will probably enable larger study groups, reduce heterogeneity within the group and give reliable treatment implementation. This results in high conclusion validity, while the external validity is reduced, since the selection is not representative if we want to generalize the results to the software industry.

Another example is to have the subjects measure several factors by filling out schemes in order to make sure that the treatments and outcomes really represent the constructs under study. This action will increase the construct validity, but there is a risk that the conclusion validity is reduced since more, tedious measurements have a tendency to reduce the reliability of the measures.

In different experiments, different types of validity can be prioritized differently, depending on the purpose of the experiment. Cook and Campbell [Cook79] propose the following priorities for theory testing and applied research:

- In theory testing, it is most important to show that there is a casual relationship (internal validity) and that the variables in the experiment represent the constructs of the theory (construct validity). Adding to the experiment size can generally solve the issues of statistical significance (conclusion validity). Theories are seldom related to specific settings, population or times to which the results should be generalized. Hence there is little need for external validity issues. The priorities for experiments in theory testing are in decreasing order: internal, construct, conclusion and external.
- In applied research, which is the target area for most of the software engineering experiments, the priorities are different. Again, the relationships under study are of highest priority (internal validity) since the key goal of the experiment is to study relationships between causes and effects. In applied research, the generalization - from the context in which the experiment is conducted to a wider context - is of high priority (external validity). For a researcher, it is not so interesting to show a particular result for company X, but rather that the result is valid for companies of a particular size or application domain. Third, the applied researcher is relatively less interested in which of the components in a complex treatment that really causes the effect (construct validity). For example, in a reading experiment, it is not so interesting to know if it is the increased understanding in general by the reviewer, or it is the specific reading procedure that helps the readers to find more defects. The main interest is in the effect itself. Finally, in practical settings it is hard to get sufficient size of data sets, hence the statistical conclusions may be drawn with less significance (conclusion validity). The priorities for experiments in applied research are in decreasing order: internal, external, construct and conclusions.

It can be concluded that the threats to validity of experimental results are important to evaluate and balance during planning of an experiment. Depending on the purpose of the experiment, different validity types are given different priority. The threats to an experiment are also closely related to the practical importance of the results. We may, for example, be able to show a statistical significance, but the difference is of no practical importance. This issue is further elaborated at the end of Chapter 8.

7 OPERATION

When an experiment has been designed and planned it must be carried out in order to collect the data that should be analyzed. This is what we mean with the operation of an experiment. In the operational phase of an experiment, the treatments are applied to the subjects. This means that this part of the experiment is the part where the experimenter actually meets the subjects. In most experiments in software engineering there are only a few other times when the subjects actually are involved. These occasions can, for example, be in a briefing before subjects commit to participate in the experiment and after the experiment when the results of the experiment are presented to the subjects. Since experiments in software engineering almost always deal with humans, this chapter deals to some extent with how to motivate people to participate and take part in experiments.

Even if an experiment has been perfectly designed and the collected data is analyzed with the appropriate analysis methods, the result will be invalid if the subjects have not participated seriously in the experiment. Since the field of experimental psychology also deals with experiments involving humans, guidelines for conducting experiments from that field [APA92, BPS93] are to some extent applicable alsc in software engineering.

The operational phase of an experiment consists of three steps: **preparatio** where subjects are chosen and forms etc. are prepared, **execution** where the sut jects perform their tasks according to different treatments and data is collected, ar **data validation** where the collected data is validated. The three steps are displaye in Figure 14 and they are further described in the sequel of this chapter.

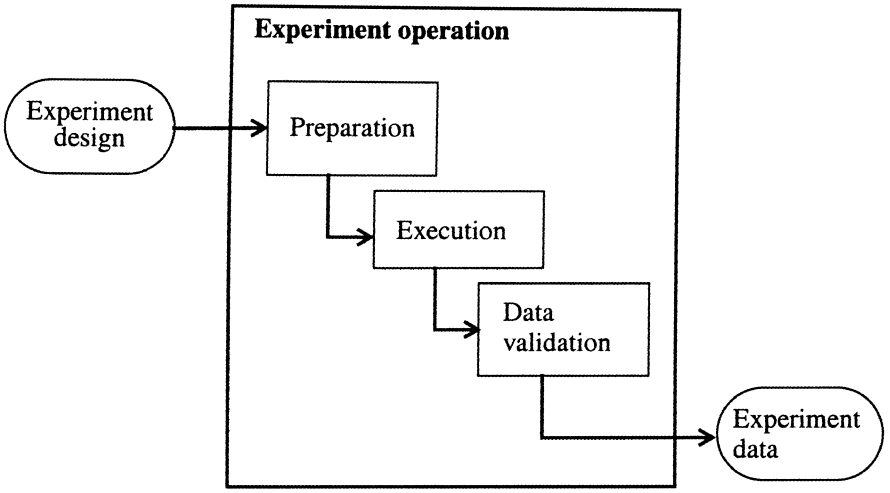


Figure 14. *Three steps in experiment operation.*

7.1 Preparation

Before the experiment is actually executed there are some preparations that have to be made. The better these preparations are performed the easier it will be to execute the experiment. There are two important aspects in the preparation. The first is to select and inform participants, and the second is to prepare material such as forms and tools.

7.1.1 Commit participants

Before an experiment can be started, people who are willing to act as subjects have to be found. It is essential that the people are motivated and willing to participate throughout the whole experiment.

In many cases it is important to find people who work with tasks in the experiment that are not too far from what they do when they are not participating in the experiment. For example, if an experiment involves writing C-code with different kinds of tools, it would probably make sense to involve persons who are used to write C-code, and not to involve Pascal-programmers. If people are chosen that are not a representative set of the people that we want to be able to make statements

about, this will be a threat to the external validity of the experiment, see Chapter 6. The selection of subjects, in terms of sampling technique, is discussed in Section 6.4.

As the right people are found and it is necessary to convince these people to participate in the experiment. The following aspects could be considered when people are participating as subjects:

- **Obtain consent**. The participants have to agree to the research objectives. If the participants do not know the intention of the work or the work does not comply with what they thought they should do when they agreed to participate, there is a risk that they will not perform the experiment according to the objectives and their personal ability. This could result in that the data becomes invalid. It is important to describe how the result of the experiment will be used and published. It should be made clear to the participants that they are free to withdraw from the experiment. Sometimes a trade-off must be made between this aspect and the design with respect to validity. If the participants are affected by the experiment as such, this will affect the validity of the experiment.
- **Sensitive results**. If the results obtained in the experiment are sensitive for the participants, it is important to assure the participants that the results of their personal performance in the experiment will be kept confidential. It is sometimes hard to judge if the result is sensitive or not, but generally it can be said that if the result would have a meaning for the participants outside the experiment it is in some way sensitive. For example, if the experiment measures the productivity of a programmer, the result would indicate how skilled the programmer is as a programmer and the result would be sensitive. On the other hand if participants are asked to use a method for acceptance testing and they normally never deal with this type of testing, the result of the experiment would probably not be that sensitive.
- **Inducements**. One way to attract people to an experiment is to offer some kind of inducement. The value of it should however not be too large, since this could cause people to participate merely to receive the inducement. This would not motivate people to seriously participate in the experiment.
- **Deception**. Deception, i.e. to deceive or betray the participants, is generally not to prefer. If alternative ways of conducting the experiment are available these methods should be used instead. If deception is the only alternative it should only be applied if it concerns aspects that are insignificant to the participants and do not affect their willingness to participate in the experiment. Deception with respect to aspects that are more significant, such as sensitive data etc., should never be applied. If deception is applied it should be explained and revealed to the participants as early as possible.

7.1.2 Instrumentation concerns

Before the experiment can be executed, all experiment instruments must be ready, see Section 6.6. This may include the experiment objects, guidelines for the experiment and measurement forms and tools. The required instruments are determined by the design of the experiment and the method that will be used for data collection.

If the subjects themselves should collect data, this means in most cases that some kind of forms must be handed out to the participants. One thing to determine when forms are constructed is whether they should be personal or the participants should fill them out anonymously. If there should be no additional studies and there hence is no real need for the experimenter to distinguish between different participants, it may be appropriate to use anonymous forms. This will however mean that there is no possibility to contact the participant if something is filled out in an unclear way.

In many cases it is appropriate to prepare one personal set of instruments for every participant. This is because many designs deal with randomization and repeated tests, such that different participants should be subject to different treatments. This can be done also when the participants are anonymous.

If data should be collected in interviews, questions should be prepared before the execution of the experiment. Here it may also be appropriate to prepare different questions for different participants.

7.2 Execution

The experiment can be executed in a number of different ways. Some experiments, such as simple inspection experiments can be carried out at one occasion when all participants are gathered at, for example, a meeting. The advantage of this is that the result of the data collection can be obtained directly at the meeting and there is no need to contact the participants and later on ask for their respective results. Another advantage is that the experimenter is present during the meeting and if questions arise they can be resolved directly.

Some experiments are, however, executed during a much longer time span, and it is impossible for the experimenter to participate in every detail of the experiment and the data collection. This is, for example, the case when the experiment is performed in relation to one or several large projects, where different methods for development are evaluated. An example of such an experiment is presented in [Ohlsson98], where a course in large-scale software development was studied during two years. Each year, seven projects were run in parallel with a total of approximately 120 students. The objective of the experiment in [Ohlsson98] was to evaluate different levels of formality when collecting effort data.

7.2.1 Data collection

Data can be collected either manually by the participants that fill out forms, manually supported by tools, in interviews, or automatically by tools. The first alternative is probably the most common and the last is probably the least common.

An advantage of using forms is that it does not require so much effort for the experimenter, since the experimenter does not have to actively take part in the collection. A drawback is that there is no possibility for the experimenter to directly reveal inconsistencies, uncertainties and flaws in the forms etc. This type of faults cannot be revealed until after the data collection or if the participants raise attention to faults or have questions. An advantage with interviews is that the experimenter has the possibility to communicate better with the participants during the data collection. A drawback is of course that it requires more effort from the experimenter.

7.2.2 Experimental environment

If an experiment is performed within a regular development project, the experiment should not affect the project more than necessary. This is because the reason for performing the experiment within the project is to see the effects of different treatments in an environment such as the one in the project. If the project environment is changed too much because of the experiment that effect will be lost.

There are however some cases where it is appropriate with some interaction between the experiment and the project. If the experimenter, for example, reveals that some parts of the project could be performed better or that estimations are not correct, it would be appropriate for the experimenter to tell the project leader. This type of direct feedback from the experiment to the project can help to motivate project personnel to participate in the experiment.

7.3 Data validation

When data has been collected, the experimenter must check that the data is reasonable and that it has been collected correctly. This deals with aspects such as if the participants have understood the forms and therefore filled them out correctly. Another source of error is that some participants may not have participated in the experiment seriously and some data therefore should be removed before the analysis. Outlier analysis is further discussed in Section 8.2.

It is important to review that the experiment has actually been conducted in the way that was intended. It is, for example, important that the subjects have applied the correct treatments in the correct order. If this type of misunderstandings have occurred, the data is of course invalid.

One way to check that the participants have not misunderstood the intentions of the experimenter, is to give a seminar, or in some other way present the results of the data collection. This will give the participants the possibility to reflect on results that they do not agree with.

8 ANALYSIS AND INTERPRETATION

After collecting experimental data in the operation phase, we want to be able to draw conclusions based on this data. To be able to draw valid conclusions, we must interpret the experiment data. Quantitative interpretation may be carried out in three steps, as depicted in Figure 15.

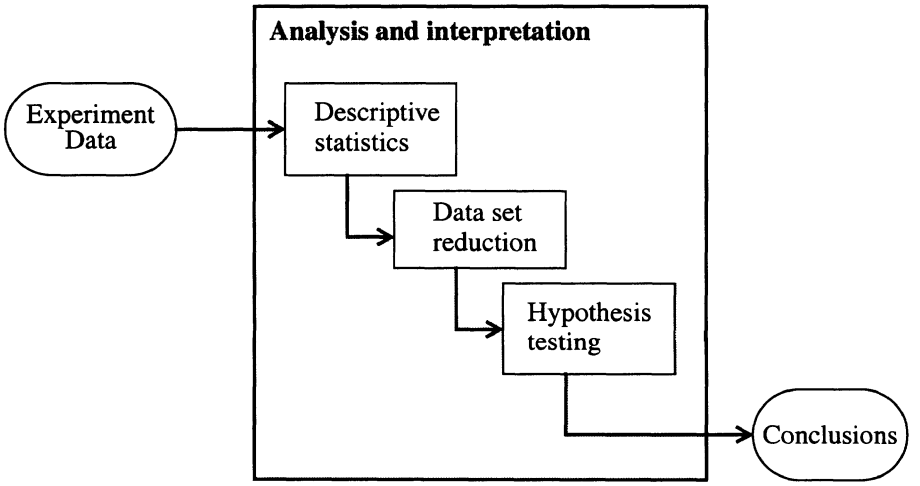


Figure 15. *Three steps in quantitative interpretation.*

In the first step, the data is characterized using **descriptive statistics**, which visualize central tendency, dispersion, etc. In step two, abnormal or false data points are excluded, thus **reducing the data set** to a set of valid data points. In the third step, the data is analyzed by **hypothesis testing**, where the hypotheses of the experiment are evaluated statistically, at a given level of significance. These steps are described in more detail in the sequel of this chapter.

8.1 Descriptive statistics

Descriptive statistics deal with the presentation and numerical processing of a data set. After collecting experimental data, descriptive statistics may be used to describe and graphically present interesting aspects of the data set. Such aspects include measures indicating, for example, where on some scale the data is positioned and how concentrated or spread out the data set is. The goal of descriptive statistics is to get a feeling for how the data set is distributed. Descriptive statistics may be used before carrying out hypothesis testing, in order to better understand the nature of the data and to identify abnormal or false data points (so called *outliers*).

In this section, we present a number of descriptive statistics and plotting techniques that may help to get a general view of a data set. The scale of measurement (see Chapter 3) restricts the type of statistics that are meaningful to compute. Table 18 shows a summary of some of these statistics in relation to the scales under which they are admissible. It should, however, be noted that measures of one scale type can be applied to the more powerful scales, for example, mode can be used for all four scales in Table 18.

Table 18. *Some relevant statistics for each scale.*

	Measure of		
Scale Type	**central tendency**	**dispersion**	**dependency**
Nominal	mode	frequency	
Ordinal	median, percentile	interval of variation	Spearman corr. coeff. Kendall corr. coeff.
Interval	mean	standard deviation, variance, and range	Pearson corr. coeff.
Ratio	geometric mean	coefficient of variation	

8.1.1 Measures of central tendency

Measures of central tendency, such as mean, median, and mode, indicate a "middle" of a data set. This "midpoint" is often called *average* and may be interpreted as an estimation of the *expectation* of the stochastic variable from which the data points in the data set are sampled.

When describing the measures of central tendency, we assume that we have n data points $x_1 ... x_n$, sampled from some stochastic variable.

The (arithmetic) **mean**, denoted $\bar{x}$, is calculated as:

$$\bar{x} = \frac{1}{n} \cdot \sum_{i=1}^{n} x_i$$

The mean value is meaningful for the interval and ratio scales.

For example, we may compute the mean for the data set (1, 1, 2, 4) resulting in $\bar{x} = 2.0$.

The **median**, denoted $\tilde{x}$, represents the middle value of a data set, following that the number of samples that are higher than the median is the same as the number of samples that are lower than the median. The median is calculated by sorting the samples in ascending (or descending) order and picking the middle sample. This is well defined if n is odd. If n is even, the median may be defined as the arithmetic mean of the two middle values. (The latter operation requires that the scale is at least interval. If the scale is ordinal, one of the two middle values may be selected by random choice, or the median may be represented as a pair of values)

The median value is meaningful for the ordinal, interval, and ratio scales.

As an example, we may compute the median for the data set (1, 1, 2, 4) resulting in $\tilde{x} = 1.5$.

The median is a special case of the **percentile**, namely the 50%-percentile, denoted $x_{50\%}$, indicating that 50% of the samples lies below $x_{50\%}$. In general x_p denotes the percentile where $p\%$ of the samples lies below this value. The percentile value is meaningful for the ordinal, interval, and ratio scales.

The **mode** represents the most commonly occurring sample. The mode is calculated by counting the number of samples for each unique value and selecting the value with the highest count. The mode is well defined if there is only one value that is more common than all others are. If an odd number of samples have the same occurrence count, the mode may be selected as the middle value of the most common samples. (The latter operation requires that the scale is at least ordinal. If the scale is nominal, the mode may be selected among the most common samples by random choice or represented as a pair of the most common values).

The mode value is meaningful for the nominal, ordinal, interval and ratio scales. As an example, we may compute the mode for the data set (1, 1, 2, 4) giving a mode of 1.

A less common measure of central tendency is the **geometric mean**, which is calculated as the n:th root of the product of all samples, as shown below.

$$\sqrt[n]{\prod_{i=1}^{n} x_i}$$

The geometric mean is well defined if all samples are non-negative and meaningful for the ratio scale.

The (arithmetic) mean and median are equal if the distribution of samples is symmetric. If the distribution is both symmetric and has one unique maximum, all these three measures of central tendency are equal. However, if the distribution of samples is skewed, the values of the mean, median and mode may differ, see Figure 16.

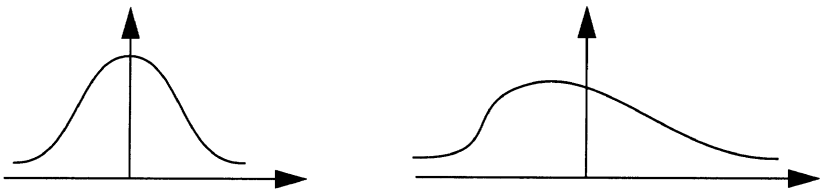


Figure 16. *A symmetric distribution has the same values of mean, median, and mode, while they may differ if the distribution is asymmetric.*

If, for example, the higher tail of the distribution is outstretched, the mean is increased, while the median and mode is unaffected. This indicates that the mean is a more sensitive measure. However, it requires at least an interval scale, and hence may not always be meaningful.

8.1.2 Measures of dispersion

The measures of central tendency do not convey information of the dispersion of the data set. Thus, it is necessary to measure the level of variation from the central tendency, i.e. to see how outspread or concentrated the data is.

The (sample) **variance**, denoted s^2, is a common measure of dispersion, and is calculated as:

$$s^2 = \frac{1}{n-1} \cdot \sum_{i=1}^{n} (x_i - \bar{x})^2$$

Hence, the variance is the mean of the square distance from the sample mean. It may seem odd that the dividend is $n-1$ and not just n, but by dividing with $n-1$, the variance gets some desirable properties. In particular, the sample variance is an unbiased and consistent estimation of the variance of the stochastic variable. The variance is meaningful for the interval and ratio scales.

The **standard deviation**, denoted s, is defined as the square root of the variance:

$$s = \sqrt{\frac{1}{n-1} \cdot \sum_{i=1}^{n} (x_i - \bar{x})^2}$$

The standard deviation is often preferred over the variance as it has the same dimension (unit of measure) as the data values themselves. The standard deviation is meaningful for the interval and ratio scales.

The **range** of a data set is the distance between the maximum and minimum data value:

$$range = x_{max} - x_{min}$$

The range value is meaningful for the interval and ratio scales.

The **variation interval** is represented by the pair $(x_{\min}, x_{\max})$ including the minimum and maximum of the data values. This measure is meaningful for ordinal, interval and ratio scales.

Sometimes the dispersion is expressed in percentages of the mean. This value is called the **coefficient of variation** and is calculated as:

$$100 \cdot \frac{s}{\bar{x}}$$

The coefficient of variation measure has no dimension and is meaningful for the ratio scale.

A general view of dispersion is given by the **frequency** of each data value. A *frequency table* is constructed by tabulating each unique value and the count of occurrence for each value. The **relative frequency** is calculated by dividing each frequency by the total number of samples. For the data set (1, 1, 1, 2, 2, 3, 4, 4, 4, 5, 6, 6, 7) with 13 samples we can construct the frequency table shown in Table 19. The frequency is meaningful for all scales.

Table 19. *A frequency table example.*

Value	Frequency	Relative frequency
1	3	23%
2	2	15%
3	1	8%
4	3	23%
5	1	8%
6	2	15%
7	1	8%

8.1.3 Measures of dependency

When the data set consists of related samples in pairs (x_i, y_i) from two stochastic variables, X and Y, it is often interesting to examine the dependency between these variables.

If X and Y are related through some function, $y=f(x)$, we want to estimate this function. If we suspect that the function $y=f(x)$ is linear and could be written on the form $y=\alpha+\beta x$, we could apply **linear regression**. Regression means fitting the data points to a curve, and in our case we will show how fitting a line that minimizes the sum of the quadratic distances to each data point makes linear regression.

Before we present the formulas we define the following shorthands for some commonly occurring sums:

$$S_{xx} = \sum_{i=1}^{n} (x_i - \bar{x})^2$$

$$S_{yy} = \sum_{i=1}^{n} (y_i - \bar{y})^2$$

$$S_{xy} = \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y}) = \left(\sum_{i=1}^{n} x_i y_i\right) - \frac{1}{n}\left(\sum_{i=1}^{n} x_i\right)\left(\sum_{i=1}^{n} y_i\right)$$

The sums can be used to compute the regression line $y = \bar{y} + \beta(x - \bar{x})$ where the slope of the line is:

$$\beta = \frac{S_{xy}}{S_{xx}}$$

and the line crosses the y-axis at $\alpha = \bar{y} - \beta\bar{x}$.

If the dependency is non-linear, it may be possible to find a *transformation* of data, so that the relation becomes linear, and linear regression can be used. If, for example, the relation is exponential, $y=\alpha x^{\beta}$, this implicates that a logarithmic transformation of the data results in the linear relation $\log(y) = \log(\alpha) + \beta \log(x)$. After the logarithmic transformation we can use linear regression to calculate the parameters of the line.

For a single number that quantifies how much two data sets, x_i and y_i, vary together, we can use the **covariance**. This measure of dependency, denoted c_{xy}, is defined as:

$$c_{xy} = \frac{S_{xy}}{n-1}$$

The covariance is meaningful for interval and ratio scales. The covariance is dependent on the variance of each variable, and to be able to compare dependencies between different related variables, the covariance may be normalized with the standard deviations of x_i and y_i. If we do this we get the **correlation coefficient** r (also called *Pearson correlation coefficient*), which is calculated as:

$$r = \frac{c_{xy}}{s_x \cdot s_y} = \frac{S_{xy}}{\sqrt{S_{xx} \cdot S_{yy}}} = \frac{\left(n \cdot \sum_{i=1}^{n} x_i y_i\right) - \left(\sum_{i=1}^{n} x_i\right)\left(\sum_{i=1}^{n} y_i\right)}{\sqrt{\left(n \cdot \sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2\right) \cdot \left(n \cdot \sum_{i=1}^{n} y_i^2 - \left(\sum_{i=1}^{n} y_i\right)^2\right)}}$$

The r-value is between -1 and $+1$, and if there is no correlation r equals zero. The opposite is however not true. The x_i and y_i values may be strongly correlated in a non-linear manner even if $r = 0$. The (Pearson) correlation coefficient measures only linear dependency and is meaningful if the scales of x_i and y_i are interval or ratio, and works good for data that is normally distributed.

If the scale is ordinal or if the data is far from normally distributed, the **Spearman rank-order correlation coefficient**, denoted r_s, can be used. The Spearman correlation is calculated in the same manner as the Pearson correlation except that

the ranks (i.e., the order numbers when the samples are sorted) are used instead of the sample values, see for example [Siegel88].

Another measure of dependency is the **Kendall rank-order correlation coefficient**, denoted *T*. The Kendall correlation is suitable as a measure for the same sort of data as the Spearman correlation, i.e. at least ordinal samples in pair. The Kendall correlation differs, however, in the underlying theory as it focuses on counting agreements and disagreements in ranks between samples, see for example [Siegel88, pp. 245-262].

If we have more than two variables, we can apply **multivariate analysis**, including techniques such as *multiple regression*, *principal component analysis* (PCA), *cluster analysis*, and *discriminant analysis*. These techniques are described in, for example, [Manly94, Kachigan86, Kachigan91].

8.1.4 Graphical visualization

When describing a data set, quantitative measures of central tendency, dispersion, and dependency, should be combined with graphical visualization techniques. Graphs are very illustrative and give a good overview of the data set.

One simple but effective graph is the **scatter plot**, where pairwise samples (x_i, y_i) are plotted in two dimensions, as shown in Figure 17.

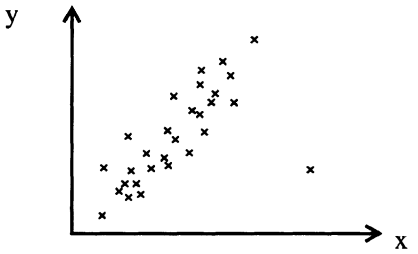


Figure 17. *A scatter plot.*

The scatter plot is good for assessing dependencies between variables. By examining the scatter plot, it can be seen how outspread or concentrated the data points are, and if there is a tendency of linear relation. Atypical values (outliers) may be identified, and the correlation may be observed. In Figure 17, there is a linear tendency with a positive correlation, and we may observe potential outliers. In this particular case there is one candidate outlier.

The **box plot** is good for visualizing the dispersion and skewedness of samples. The box plot is constructed by indicating different percentiles graphically, as shown in Figure 18. Box plots can be made in different ways. We have chosen an approach advocated by, for example, [Fenton96, Frigge89]. The main difference between the

approaches is how to handle the whiskers. Some literature proposes that the whiskers should go to the lowest and highest values respectively, see for example [Montgomery97]. In [Fenton96], it is proposed to use a value, which is the length of the box, multiplied with 1.5 and added or subtracted from the upper and lower quartiles respectively.

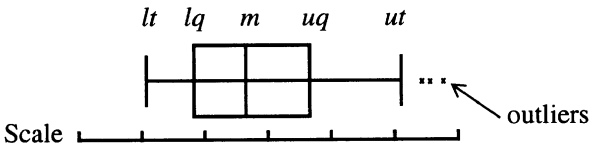


Figure 18. *A box plot.*

The middle bar in the box m, is the median. The lower quartile lq, is the 25% percentile (the median of the values that are less than m), and the upper quartile uq is the 75% percentile (the median of the values that are greater than m). The length of the box is $d = uq - lq$.

The tails of the box represent the theoretical bound within which it is likely to find all data points if the distribution is normal. The upper tail ut is $uq+1.5d$ and the lower tail lt is $lq–1.5d$ [Frigge89]. The tail values are truncated to the nearest actual data point, in order to avoid meaningless values (such as negative lines of code).

Values outside the lower and upper tails are called **outliers**, and are shown explicitly in the box plot. In Figure 18, there are three outliers.

The **histogram** can be used to give an overview of the distribution density of the samples from one variable. A histogram consists of bars with heights that represent the frequency (or the relative frequency) of a value or an interval of values, as shown in Figure 19. The histogram is thus a graphical representation of a frequency table. One distribution of particular interest is the normal distribution, since it is one aspect that should be taken into account when analyzing the data. Thus, a plot could provide a first indication whether the data resembles a normal distribution or not. It is also possible to test the data for normality. This is further discussed in Section 8.3 when introducing the Chi-2 test.

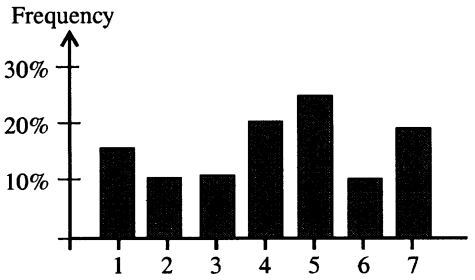


Figure 19. *A histogram.*

The **cumulative histogram,** illustrated in Figure 20, may be used to give a picture of the probability distribution function of the samples from one variable. Each bar is the cumulative sum of frequencies up to the current class of values.

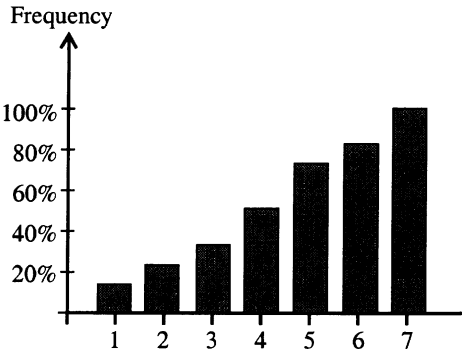


Figure 20. *A cumulative histogram.*

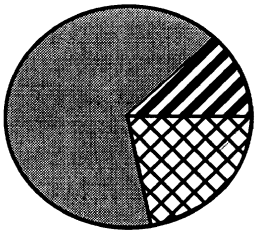


Figure 21. *A pie chart.*

A **pie chart**, as illustrated in Figure 21, shows the relative frequency of the data values divided into a specific number of distinct classes, by constructing segments in a circle with angles proportional to the relative frequency.

8.2 Data set reduction

In Section 8.3, a number of statistical methods are described. All methods have in common that the result of using them, depend much on the quality of the input data. If the data, that the statistical methods are applied on, does not represent what we think it represents, then the conclusions that we draw from the results of the methods are of course not correct.

Errors in the data set can occur either as systematic errors, or they can occur as outliers, which means that the data point is much larger or much smaller than one could expect looking at the other data points, see Figure 22.

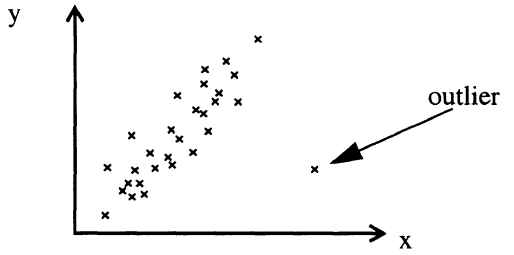


Figure 22. *An outlier detected in a scatter plot.*

One very effective way to identify outliers is to draw scatter plots, as shown in Figure 22. Another way is to draw box plots, as described in Section 8.1. There are some statistical methods available for detecting outliers. These methods can, for example, be based on that the data comes from a normal distribution and determining the probability of finding a value such as the largest or smallest value from this distribution. This can, for example, be done by looking at the difference between possible outliers and the mean of all values or at the difference between the outlier and its closest value, and then determining the probability of finding as large difference as was found. This study is conducted to evaluate if it is possible that the outlier found can come from the normal distribution, although it seems like an extreme value.

Notice that data reduction as discussed here is related to data validation as discussed in Chapter 7. Data validation deals with identifying false data points based on the execution of the experiment, such as determining if people have participated seriously in the experiment. The type of data reduction discussed in this section is concerned with identifying outliers not only based on the experiment execution, but instead looking at the results from the execution in the form of collected data and taking into account, for example, descriptive statistics.

When outliers have been identified, it is important to decide what to do with them. This should not only be based on the coordinates in the diagram, here it is important to analyze the reasons for the outliers. If the outlier is due to a strange or rare event that never will happen again, the point could be excluded. This can, for example, be the case if the point is completely wrong or misunderstood.

If the outlier is due to a rare event that may occur again, for example, if a module was implemented by inexperienced staff, it is not advisable to exclude the value from the analysis, because there is much relevant information in the outlier. If the outlier is due to a variable that was not considered before, such as the staff experi-

ence, it may be considered to base the calculations and models also on this variable. It is also possible to derive two separate models. In the case with staff experience it means one model based on normal staff (with the outlier removed) and one separate model for inexperienced staff. How to do this must be decided for every special case.

It is not only invalid data that can be removed from the data set. It is sometimes ineffective to analyze redundant data if the redundancy is too large. One way to identify redundant data is through factor analysis and principal component analysis (PCA). These techniques identify orthogonal factors that can be used instead of the original factors. It would lead too far to describe this type of techniques in this book, refer instead to for example [Kachigan86, Kachigan91, Manly94].

8.3 Hypothesis testing

8.3.1 Basic concept

The objective of hypothesis testing is to see if it is possible to reject a certain null hypothesis, H_0, based on a sample from some statistical distribution. That is, the null hypothesis describes some properties of the distribution from which the sample is drawn and the experimenter wants to reject that these properties are true with a given significance. The null hypothesis is also discussed in Chapter 6.

A common case is that the distribution depends on a single parameter. Setting up H_0 then means formulating the distribution and assigning a value to the parameter, which will be tested.

For example, if an experimenter observes a vehicle and wants to show that the vehicle is not a car. The experimenter knows that all cars have four wheels, but also that there are other vehicles than cars that have four wheels. A very simple example of a null hypothesis can be formulated as "H_0: the observed vehicle is a car".

To test H_0, a test unit, t, is defined and a critical area, C, is given as well which is a part of the area over which t varies. This means that the significance test can be formulated as:

- If $t \in C$, reject H_0
- If $t \notin C$, do not reject H_0

In our example, the test unit t is the number of wheels and the critical area is $C = \{1, 2, 3, 5, 6, \ldots\}$. The test is: if $t \leq 3$ or $t \geq 5$, reject H_0, otherwise do not reject H_0.

If it is observed that $t = 4$, it means that the hypothesis cannot be rejected and no conclusion can be drawn. This is because there may be other vehicles than cars with four wheels.

The null hypothesis should hence be formulated negatively, i.e. the intention of the test is to reject the hypothesis. If the null hypothesis is not rejected, nothing can be said about the outcome, while if the hypothesis is rejected, it can be stated that the hypothesis is false with a given significance (α), see below. When a test is carried out it is in many cases possible to calculate the lowest possible significance (often denoted the p-value) with which it is possible to reject the null hypothesis. This value is often reported from statistical analysis packages.

The critical area, C, may have different shapes, but it is very common that it has some form of intervals, for example $t \leq a$ or $t \geq b$. If C consists of one such interval it is one-sided. If it consists of two intervals ($t \leq a$, $t \geq b$, where $a < b$), it is two-sided.

Three important probabilities concerning hypothesis testing are:

α = P(type-I-error) = P(reject H_0 | H_0 is true)

β = P(type-II-error) = P(not reject H_0 | H_0 is false)

Power = 1–β = P(reject H_0|H_0 is false)

These probabilities are also discussed in Chapter 6.

We try here to illustrate the concepts in a small example describing a simple but illustrative test called the **binomial test**. An experimenter has measured a number of failures during operation for a product and classified them as corruptive (faults that do corrupt the program's data) and non-corruptive (faults that do not corrupt the program's data). The experimenter's theory is that the non-corruptive faults are more common than the corruptive faults. The experimenter therefore wants to perform a test in order to see if the difference in the number of faults of the different types is due to coincident or if it reveals a systematic difference.

The null hypothesis is that there is no difference in the probability of receiving a corruptive fault and receiving a non-corruptive fault. That is, the null hypothesis can be formulated as:

H_0: P(corruptive fault) = P(non-corruptive fault) = 1/2.

It is decided that α should be less than 0.10. The experimenter has received the following data:

- There are 11 faults that are non-corruptive.
- There are 4 faults that are corruptive.

If the null hypothesis is true, the probability of obtaining as few as four (i.e., four or less) corruptive faults out of 15 is

$$P(\text{0-4 corruptive faults})) = \sum_{i=0}^{4} \binom{15}{i} \left(\frac{1}{2}\right)^{i} \left(\frac{1}{2}\right)^{15-i} = \frac{1}{2^{15}} \sum_{i=0}^{4} \binom{15}{i} = 0.059$$

That is, if the experimenter concludes that the data that has been received shows that the non-corruptive faults are more common than the corruptive faults, the probability of committing a type-I-fault is 0.059. In this case the experimenter can reject H_0 because $0.059 < 0.10$.

The probability of receiving five or less corruptive faults, if the null hypothesis is true, can be computed to be 0.1509. This is larger than 0.10, which means that 5 corruptive faults out of 15 would not be sufficient to reject H_0. The experimenter can therefore decide more formally to interpret the data in an experiment with 15 received faults as:

- If four or less faults are corruptive, reject H_0.
- If more than four faults are corruptive, do not reject H_0.

To summarize, the number of received corruptive faults (out of 15 faults) is the test unit and the critical area is 0, 1, 2, 3 and 4 (corruptive faults).

Based on this, it is interesting to determine the power of the formulated test. Since the power is the probability of rejecting H_0 if H_0 is not true, we have to formulate what we mean with that H_0 is not true. In our example this can be formulated as

P(corruptive fault) < P(non-corruptive fault).

Since the sum of the two probabilities equals 1, this can also be formulated as:

P(corruptive fault) = $a < 1/2$

The probability of receiving four or less corruptive faults out of 15 faults (i.e., the probability of rejecting H_0 if H_0 is false) is:

$$p = \sum_{i=0}^{4} \binom{15}{i} a^i (1-a)^{15-i}$$

This probability is plotted for different values of a in Figure 23.

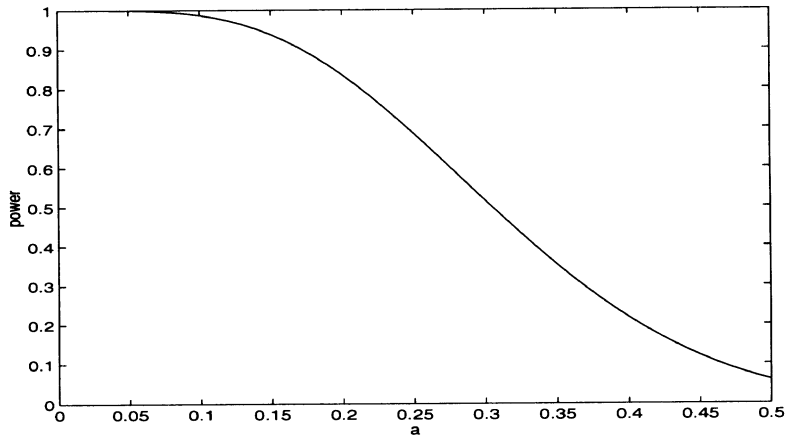


Figure 23. *Power of a one sided binomial test.*

It can be seen that the power of the test is high if the difference between the probabilities of receiving a corruptive fault and a non-corruptive fault is large. If, for example, $a = 0.05$ there is a very great chance that there will be four or fewer corruptive faults. On the other hand, if the difference is very small, the power will be smaller. If, for example, $a = 0.45$ there is a great chance that there will be more than four corruptive faults.

There are a number of factors affecting the power of a test. First, the test itself can be more or less effective. Second, the sample size affects the power. A larger sample size means higher power. Another aspect that affects the power is the choice of a one sided or two sided alternative hypothesis. A one sided hypothesis gives a higher power than a two sided hypothesis.

8.3.2 Parametric and non-parametric tests

Tests can be classified into parametric tests and non-parametric tests. Parametric tests are based on a model that involves a specific distribution. In most cases, it is assumed that some of the parameters, involved in a parametric test, are normally distributed. One test for normality is the Chi-2 test, which is further described below when discussing the different types of test. Parametric tests also require that parameters can be measured at least on an interval scale. If parameters cannot be measured on at least an interval scale this means generally that parametric tests cannot be used. In these cases there are a wide range of non-parametric tests available.

Non-parametric tests do not make the same type of assumptions concerning the distribution of parameters as parametric tests do. Only very general assumptions are made to derive non-parametric tests. The binomial test, described in the previous

subsection is an example of a non-parametric test. Non-parametric tests are more general than parametric tests. This means that non-parametric tests, if they are available, can be used instead of parametric tests, but parametric tests cannot generally be used when non-parametric tests can be used.

With respect to the choice of parametric or non-parametric test, there are two factors to consider:

- **Applicability**. What are the assumptions made by the different tests? It is important that assumptions regarding distributions of parameters and assumptions concerning scales are realistic.
- **Power**. The power of parametric tests is generally higher than for non-parametric tests. Therefore, parametric tests require fewer data points, and therefore smaller experiments, than non-parametric test if the assumptions are true.

The choice between parametric and non-parametric statistical methods is also discussed in [Briand96a]. There it is described that even if it is a risk using parametric methods when the required conditions are not fulfilled, it is in some cases worth taking that risk. Simulations have shown that parametric methods, such as the t-test, described below, are fairly robust to deviations from the preconditions (interval scale) as long as the deviations are not too large.

8.3.3 Overview of tests

In addition to the binomial test introduced above, the following tests are described in this section.

- **t-test**. One of the most often used parametric tests. The test is used to compare two sample means. That is, the design is one factor with two treatments.
- **Mann-Whitney**. This is a non-parametric alternative to the t-test.
- **F-test**. This is a parametric test that can be used to compare two sample distributions.
- **Paired t-test**. A t-test for a paired comparison design.
- **Wilcoxon**. This is a non-parametric alternative to the paired t-test.
- **Sign test**. This is a non-parametric alternative to the paired t-test. The sign test is a simpler alternative to the Wilcoxon test.
- **ANOVA** (ANalysis Of VAriance). A family of parametric tests that can be used for designs with more than two levels of a factor. ANOVA tests can, for example, be used in the following designs: One factor with more than two levels, one factor and blocking variable, factorial design, and nested design.
- **Kruskal-Wallis**. This is a non-parametric alternative to ANOVA in the case of one factor with more than two treatments.

- **Chi-2**. This is a family of non-parametric tests that can be used when data are in the form of frequencies.

The different tests can be sorted with respect to design type and with respect to whether they are parametric or non-parametric as in Table 20.

Table 20. *Overview of parametric / non-parametric tests for different designs.*

Design	Parametric	Non-parametric
One factor, one treatment		Chi-2 Binomial test
One factor, two treatments, completely randomized design	t-test F-test	Mann-Whitney Chi-2
One factor, two treatments, paired comparison	Paired t-test	Wilcoxon Sign test
One factor, more than two treatments	ANOVA	Kruskal-Wallis Chi-2
More than one factor	ANOVA (not described in this book)	

For all tests that are described, the following items are presented in separate tables for each test:

- **Input**. The type of measurements needed to make the test applicable describes the input to the test. That is, this describes what requirements there are on the experiment design if the test should be applicable.
- **Null hypothesis**. A formulation of the null-hypothesis is provided.
- **Calculations**. It describes what to calculate based on the measured data.
- **Criterion**. The criterion for rejecting the null hypothesis. This often involves using statistical tables and it is described which table to use from Appendix A of this book. In this book tables are only provided for one level of significance, but for many tests references are given to other sources where more comprehensive tables are provided.

All tests are not described completely here. For more information concerning the tests refer to the references given in the text. For example, the Mann-Whitney test, the Wilcoxon test, the sign test and the Kruskal-Wallis test are described for the most straightforward case with few samples. If there are many samples (for example, more than about 35 for the sign test) it is in many cases hard to do the cal-

culations and decisions as described below. In these cases it is possible to do certain approximations because there are so many samples. How to do this is described in, for example, [Siegel88]. In that reference it is also described how to do when ties (two or more equal values) occur for those tests.

The objective of the descriptions of the tests is that it should be possible to use the tests based on the descriptions and the examples. The intention is not to provide all details behind the derivations of the formulas.

Using the type of description outlined above, our simple example test, see Section 8.3.1, can be summarized as:

Binomial test	
Input	Number of events counted for two different kind of events (event 1 and event 2).
H_0	P(event 1) = P(event 2)
Calculations	Calculate $p = \frac{1}{2^N}\sum_{i=0}^{n}\binom{N}{i}$, where N is the total number of events, and n is the number of events of the most rare event.
Criterion	Two sided (H_1: P(event 1) ≠ P(event 2)): reject H_0 if $p < \alpha/2$. One sided (H_1: P(event 1) < P(event 2)): reject H_0 if $p < \alpha$ and event 1 is the most rare event in the sample.

In the above table the Binomial test is described for the null hypothesis that the two events are equally probable. It is possible to state other null hypotheses, such as P(event 1) = 0.3 and P(event 2) = 0.7. For a description of how to perform the test in those cases, see for example [Siegel88].

For most of the tests in this chapter examples of usage are presented. The examples are based on fictitious data. Moreover, the tests are primarily presented for a significance level of 5% for which tables are provided in Appendix A. More comprehensive tables are available in books on statistics, for example, [Marascuilo88, Montgomery97].

8.3.4 t-test

The t-test is a parametric test used to compare two independent samples. That is, the design should be one factor with two levels. The t-test can be performed based on a number of different assumptions, but here an often-used alternative is described. For more information, refer for example to [Montgomery97, Siegel88, Marascuilo88]. The test is performed as:

t-test	
Input	Two independent samples: $x_1, x_2,...x_n$ and $y_1, y_2,... y_m$.
H_0	$\mu_x = \mu_y$, i.e. the expected mean values are the same.
Calculations	Calculate $t_0 = \dfrac{\bar{x} - \bar{y}}{S_p\sqrt{\dfrac{1}{n} + \dfrac{1}{m}}}$, where $S_p = \sqrt{\dfrac{(n-1)S_x^2 + (m-1)S_y^2}{n+m-2}}$, and, S_x^2 and S_y^2 are the individual sample variances.
Criterion	Two sided (H_1: $\mu_x \neq \mu_y$): reject H_0 if $\lvert t_0 \rvert > t_{\alpha/2,\, n+m-2}$. Here, $t_{\alpha,\, f}$ is the upper α percentage point of the t distribution with f degrees of freedom, which is equal to $n+m-2$. The distribution is tabulated in, for example, Table A1 and [Montgomery97, Marascuilo88]. One sided (H_1: $\mu_x > \mu_y$): reject H_0 if $t_0 > t_{\alpha,\, n+m-2}$.

Example of t-test. The defect densities in different programs have been compared in two projects. In one of the projects the result is x = {3.42, 2.71, 2.84, 1.85, 3.22, 3.48, 2.68, 4.30, 2.49, 1.54} and in the other project the result is y = {3.44, 4.97, 4.76, 4.96, 4.10, 3.05, 4.09, 3.69, 4.21, 4.40, 3.49}. The null hypothesis is that the defect density is the same in both projects, and the alternative hypothesis that it is not. Based on the data it can be seen that n = 10 and m = 11. The mean values are $\bar{x}$ = 2.853 and $\bar{y}$ = 4.1055.

It can be found that S_x^2 = 0.6506, S_y^2 = 0.4112, S_p = 0.7243 and t_0 = –3.96.

The number of degrees of freedom is $f = n+m-2$ = 10+11–2 = 19. In Table A1, it can be seen that $t_{0.025,\, 19}$ = 2.093. Since $|t_0| > t_{0.025,\, 19}$ it is possible to reject the null hypothesis with a two tailed test at the 0.05 level.

8.3.5 Mann-Whitney

The Mann-Whitney test is a non-parametric alternative to the t-test. It is always possible to use this test instead of the t-test if the assumptions made by the t-test seem uncertain. The test, which is based on ranks, is not described completely here. More details can be found in, for example, [Siegel88][1] or [Marascuilo88]. The test can be summarized as:

Mann-Whitney	
Input	Two independent samples: $x_1, x_2, \ldots x_n$ and $y_1, y_2, \ldots y_m$.
H_0	The two samples come from the same distribution.
Calculations	Rank all samples and calculate $U = N_A N_B + \frac{N_A(N_A+1)}{2} - T$ and $U' = N_A N_B - U$, where $N_A = \min(n, m)$, $N_B = \max(n, m)$, and T is the sum of the ranks of the smallest sample.
Criterion	Tables providing criterion for rejection of the null hypothesis based on the calculations are available in, for example, Table A3 and [Marascuilo88.]. Reject H_0 if $\min(U, U')$ is less than or equal to the value in Table A3.

Example of Mann-Whitney. When the same data is used, as in the example with the t-test, it can be seen that $N_A = \min(10, 11) = 10$ and $N_B = \max(10, 11) = 11$.

The ranks of the smallest sample (x) are {9, 5, 6, 2, 8, 11, 4, 17, 3, 1} and the ranks of the largest sample (y) are {10, 21, 19, 20, 15, 7, 14, 13, 16, 18, 12}. Based on the ranks it can be found that $T = 66$, $U = 99$ and $U' = 11$.

Since the smallest of U and U' is smaller than 26, see Table A3, it is possible to reject the null hypothesis with a two tailed test at the 0.05 level.

1. In [Siegel88] the Wilcoxon-Mann-Whitney test is described instead of the Mann-Whitney test. The two tests are, however, essentially the same.

8.3.6 F-test

The F-test is a parametric test that can be used to compare the variance of two independent samples. More details about the test can be found in, for example, [Montgomery97, Robson94, Marascuilo88]. The test is performed as:

F-test	
Input	Two independent samples: $x_1, x_2, ...x_n$ and $y_1, y_2, ... y_m$.
H_0	$\sigma_x^2 = \sigma_y^2$, i.e. the variances are equal
Calculations	Calculate $F_0 = \frac{max(S_x^2, S_y^2)}{min(S_x^2, S_y^2)}$, where S_x^2 and S_y^2 are the individual sample variances.
Criterion	Two sided (H_1: $\sigma_x^2 \neq \sigma_y^2$): reject H_0 if $F_0 > F_{\alpha/2, n_{max}-1, n_{min}-1}$, where n_{max} is the number of scores in the sample with maximum sample variance and n_{min} is the number of scores in the sample with minimum sample variance. F_{α, f_1, f_2} is the upper α percentage point of the F distribution with f_1 and f_2 degrees of freedom, which is tabulated in, for example, Table A5.1, Table A5.2 and [Montgomery97, Marascuilo88]. One sided (H_1: $\sigma_x^2 > \sigma_y^2$): reject H_0 if $F_0 > F_{\alpha, n_{max}-1, n_{min}-1}$ and $S_x^2 > S_y^2$.

Example of F-test. When the same data is used, as in the example with the t-test, it can be found that $S_x^2 = 0.6506$ and $S_y^2 = 0.4112$, which means that $F_0 = 1.58$. It can also be seen that $n_{max} = 10$ and $n_{min} = 11$.

From Table A5.1 it can be seen that $F_{0.025, 9, 10} = 3.78$. Since $F_0 < F_{0.025, 9, 10}$ it is impossible to reject the null hypothesis with a two tailed test at the 0.05 level. That is, the test does not reject that the two samples have the same variance.

8.3.7 Paired t-test

The paired t-test is used when two samples resulting from repeated measures are compared. This means that measurements are made with respect to, for example, a subject more than once. An example of this is if two tools are compared. If two groups independently use the two different tools, the result would be two indepen-

dent samples and the ordinary t-test could be applied. If instead only one group would be used and every person used both tools, we would have repeated measures. In this case the test examines the difference in performance for every person with the different tools.

The test, which is described in more detail in, for example, [Montgomery97, Marascuilo88] is performed as:

Paired t-test	
Input	Paired samples: (x_1, y_1), (x_2,y_2)... *and* (x_n,y_n)
H_0	$\mu_d = 0$, where $d_i = x_i - y_i$, i.e. the expected mean of the differences is 0.
Calculations	Calculate $t_0 = \dfrac{\bar{d}}{S_d/(\sqrt{n})}$, where $S_d = \sqrt{\dfrac{\sum_{i=1}^{n}(d_i-\bar{d})^2}{n-1}}$
Criterion	Two sided (H_1: $\mu_d \neq 0$): reject H_0 if $\lvert t_0 \rvert > t_{\alpha/2,\, n-1}$. Here, $t_{\alpha,\, f}$ is the upper α percentage point of the t distribution with f degrees of freedom. The distribution is tabulated in, for example, Table A1 and [Montgomery97, Marascuilo88]. One sided (H_1: $\mu_d > 0$): reject H_0 if $t_0 > t_{\alpha,\, n-1}$.

Example of paired t-test. 10 programmers have independently developed two different programs. They have measured the effort that was required and the result is displayed in Table 21.

Table 21. *Required effort.*

Programmer	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Program 1	105	137	124	111	151	150	168	159	104	102
Program 2	86.1	115	175	94.9	174	120	153	178	71.3	110

The null hypothesis is that the required effort to develop program 1 is the same as the required effort to develop program 2. The alternative hypothesis is that it is not. In order to carry out the test the following are calculated:

$d = \{18.9, 22, -51, 16.1, -23, 30, 15, -19, 32.7, -9\}$

$S_d = 27.358$

$t_0 = 0.39$

The number of degrees of freedom is $f = n–1 = 10–1 = 9$. In Table A1, it can be seen that $t_{0.025,\ 9} = 2.262$.

Since $t_0 < t_{0.025,\ 9}$ it is impossible to reject the null hypothesis with a two sided test at the 0.05 level.

8.3.8 Wilcoxon

The Wilcoxon test is a non-parametric alternative to the paired t-test. The only requirements are that it is possible to determine which of the measures in a pair is the greatest and that it is possible to rank the differences.

The test, which is based on ranks, is not described in detail here. A more detailed description can be found in, for example, [Siegel88, Marascuilo88]. The test can be summarized as:

Wilcoxon	
Input	Paired samples: (x_1, y_1), (x_2,y_2)... *and* (x_n,y_n)
H_0	If all differences ($d_i = x_i - y_i$) are ranked (1, 2, 3...) without considering the sign, then the sum of the ranks of the positive differences equals the sum of the ranks of the negative differences.
Calculations	Calculate T^+ as the sum of the ranks of the positive d_i:s and T^- as the sum of the ranks of the negative d_i:s.
Criterion	Tables that can be used to determine if H_0 can be rejected based on T^+, T^- and the number of pairs, n, are available. See for example Table A4 or [Siegel88, Marascuilo88]. Reject H_0 if $\min(T^+, T^-)$ is less than or equal to the value in Table A4

Example of Wilcoxon. When the same data is used, as in the example with the paired t-test, it is found that the ranks of the absolute values of the difference (d) are {4, 6, 10, 3, 7, 8, 2, 5, 9, 1}. Based on this T^+ and T^- can be calculated to be 32 and 23.

Since the smallest of T^+ and T^- is larger than 8 (see Table A4) it is impossible to reject the null hypothesis with a two-tailed test at the 0.05 level.

8.3.9 Sign test

The sign test is, as the Wilcoxon test, a non-parametric alternative to the paired t-test. The sign test can be used instead of the Wilcoxon test when it is not possible or necessary to rank the differences, since it is only based on the sign of the difference of the values in each pair. For example, it is not necessary to use Wilcoxon when it is possible to show significance with the sign test. This is due to that the sign test has a lower power. Moreover, the sign test is easier to perform.

The test is further described in for example [Siegel88, Robson94] and can be summarized as:

Sign test	
Input	Paired samples: (x_1, y_1), (x_2,y_2)... *and* (x_N,y_N)
H_0	$P(+) = P(-)$, where + and – represents the two events that $x_i > y_i$ and $x_i < y_i$.
Calculations	Represent every positive differences ($d_i = x_i - y_i$) by a + and every negative difference by a –. Calculate $p = \frac{1}{2^N}\sum_{i=0}^{n}\binom{N}{i}$, where N is the total number of signs, and n is number of signs of the most rare signs.
Criterion	Two sided (H_1: $P(+) \neq P(-)$): reject H_0 if $p < \alpha/2$. One sided (H_1: $P(+) < P(-)$): reject H_0 if $p < \alpha$ and the + event is the most rare event in the sample.

The reader may recognize that this test is a binomial test where the two events are + and – respectively.

Example of sign test. When the same data is used, as in the example with the paired t-test, it is found that there are 6 positive differences and 4 negative. This means that

$$p = \frac{1}{2^{10}}\sum_{i=0}^{4}\binom{10}{i} = \frac{193}{512} \approx 0.3770$$

Since $p > 0.025$ it is impossible to reject the null hypothesis with a two tailed test at the 0.05 level.

8.3.10 ANOVA (ANalysis Of VAriance)

Analysis of variance can be used to analyze experiments from a number of different designs. The name, analysis of variance, is used because the method is based on looking at the total variability of the data and the variability partition according to different components. In its simplest form the test compares the variability due to treatment and the variability due to random error.

In this section, it is described how to use ANOVA in its simplest form. The test can be used to compare if a number of samples has the same mean value. That is, the design is one factor with more than two treatments. The test can be summarized as:

ANOVA, one factor, more than two treatments	
Input	a samples: $x_{11}, x_{12}, \ldots x_{1n_1}$; $x_{21}, x_{22}, \ldots x_{2n_2}$;...; $x_{a1}, x_{a2}, \ldots x_{an_a}$
H_0	$\mu_{x_1} = \mu_{x_2} = \ldots = \mu_{x_a}$, i.e. all expected means are equal
Calculations	Calculate: $SS_T = \sum_{i=1}^{a}\sum_{j=1}^{n_i} x_{ij}^2 - \frac{x_{..}^2}{N}$ $SS_{Treatment} = \sum_{i=1}^{a} \frac{x_{i.}^2}{n_i} - \frac{x_{..}^2}{N}$ $SS_{Error} = SS_T - SS_{Treatment}$ $MS_{Treatment} = SS_{Treatment}/(a-1)$ $MS_{Error} = SS_{Error}/(N-a)$ $F_0 = MS_{Treatment}/MS_{Error}$ where N is the total number of measurements and a dot index denotes a summation over the dotted index, e.g. $x_{i.} = \sum_j x_{ij}$
Criterion	Reject H_0 if $F_0 > F_{\alpha,a-1,N-a}$. Here, F_{α,f_1,f_2} is the upper α percentage point of the F distribution with f_1 and f_2 degrees of freedom, which is tabulated in, for example, Table A5.1, Table A5.2 and [Montgomery97, Marascuilo88].

The results of an ANOVA test are often summarized in an ANOVA table. The results of a test for one factor with multiple levels can, for example, be summarized as in Table 22.

Table 22. *ANOVA table for the ANOVA test described above.*

Source of variation	Sum of squares	Degrees of freedom	Mean square	F_0
Between treatments	$SS_{Treatment}$	a–1	$MS_{Treatment}$	$F_0 = \frac{MS_{Treatment}}{MS_{Error}}$
Error[a]	SS_{Error}	N–a	MS_{Error}	
Total	SS_T	N–1		

a. This is sometimes denoted within treatments.

Notice that the described ANOVA test is just one variant of ANOVA tests. ANOVA tests can be used for a number of different designs, involving many different factors, blocking variables, etc. It would lead too far to describe these tests in detail here. Refer instead to, for example, [Montgomery97, Marascuilo88].

Example of ANOVA. The module sizes in three different programs have been measured. The result is:

Program 1: 221, 159, 191, 194, 156, 238, 220, 197, 197, 194

Program 2: 173, 171, 168, 286, 206, 140, 226, 248, 189, 208, 213

Program 3: 234, 188, 181, 207, 266, 153, 190, 195, 181, 238, 191, 260

The null hypothesis is that the mean module size is the same in all three programs. The alternative hypothesis is that it is not. Based on the data above the ANOVA table in Table 23 can be calculated.

Table 23. *ANOVA table.*

Source of variation	Sum of squares	Degrees of freedom	Mean square	F_0
Between treatment	579.0515	2	289.5258	0.24
Error	36151	30	1205	
Total	36730	32		

The number of degrees of freedom are $f_1 = a-1 = 3-1 = 2$ and $f_2 = N-a = 33-3 = 30$. In Table A5.1, it can be seen that $F_{0.025, 2, 30} = 4.18$. Since $F_0 < F_{0.025, 2, 30}$ it is impossible to reject the null hypothesis at the 0.025 level.

8.3.11 Kruskal-Wallis

The Kruskal-Wallis one way analysis of variance by ranks is a non-parametric alternative to the parametric one factor analysis of variance described above. This test can always be used instead of the parametric ANOVA if it is not sure that the assumptions of ANOVA are met. The test, which is based on ranks, is not described in detail here. More details can be found in, for example, [Siegel88, Marascuilo88].

The test can be summarized as:

Kruskal-Wallis	
Input	a samples: $x_{11}, x_{12}, \ldots x_{1n_1}$; $x_{21}, x_{22}, \ldots x_{2n_2}$;...; $x_{a1}, x_{a2}, \ldots x_{an_a}$
H_0	The population medians of the a samples are equal.
Calculations	All measures are ranked in one series (1, 2,... $n_1+n_2+\ldots+n_a$), and the calculations are based on these ranks. See for example [Siegel88, Marascuilo88].
Criterion	See for example [Siegel88, Marascuilo88].

8.3.12 Chi-2

Chi-2 (sometimes denoted χ^2) tests can be performed in a number of different ways. All Chi-2 tests are, however, based on that data is in the form of frequencies. An example of frequencies for two systems with a number of modules can be that for system 1 there are 15 small modules, 20 medium modules and 25 large modules, while for system 2 there are 10 small modules, 19 medium modules and 28 large modules. This is summarized in Table 24.

Table 24. *Frequency table for module size (variables) of two systems (groups).*

Module size	**System 1**	**System 2**
small	15	10
medium	20	19
large	25	28

In this case a Chi-2 test could be performed to investigate if the distribution of small, medium and large modules are the same for the two systems.

Chi-2 tests can also be performed with one group of data, in order to see, if one measured frequency distribution is the same as a theoretical distribution. This test can, for example, be performed in order to check if samples can be seen as normally distributed.

Below, a Chi-2 test, which can be used to compare if measurements from two or more groups come from the same distribution, is summarized:

Chi-2, k independent samples (groups)

Input

Data as frequencies for k groups.

H_0

Measurements from the k groups are from the same distribution.

Calculations

Create a contingency table. An example of a contingency table for two groups and three variables (i.e., the same dimensions as the data in Table 24) can be constructed as:

variable	group 1	group 2	combined
1	n_{11}	n_{12}	R_1
2	n_{21}	n_{22}	R_2
3	n_{31}	n_{32}	R_3
Total	C_1	C_2	N

In this table n_{ij} denotes the frequency for variable i and group j, C_i denotes the sum of the frequencies for group i and R_i denotes the sum for variable i. N is the sum of all frequencies.

Compute $X^2 = \sum_{i=1}^{r} \sum_{j=1}^{k} \frac{(n_{ij} - E_{ij})^2}{E_{ij}}$, where $E_{ij} = \frac{R_i C_j}{N}$

(the expected frequency if H_0 is true), r is the number of variables and k is the number of groups.

Criterion

Reject H_0 if $X^2 > \chi^2_{\alpha, f}$, where f is the number of degrees of freedom determined as $f = (r-1)(k-1)$. $\chi^2_{\alpha, f}$ is the upper α percentage point of the Chi-2 distribution with f degrees of freedom, which is tabulated in, for example, Table A2 and [Siegel88].

Example of Chi-2 test. If a Chi-2 test is performed on the data in Table 24 then Table 25 can be constructed.

Table 25. *Calculations for Chi-2 test. (Expected values, E_{ij}, are displayed in parenthesis.)*

Module size	System 1	System 2	Combined
small	15 (12.8205)	10 (12.1795)	$R_1 = 25$
medium	20 (20)	19 (19)	$R_2 = 39$
large	25 (27.1795)	28 (25.8205)	$R_3 = 53$
Total	$C_1 = 60$	$C_2 = 57$	$N = 117$

The null hypothesis is that the size distribution is the same in both systems, and the alternative hypothesis is that the distributions are different. Based on the data the test statistic can be calculated to $X^2 = 1.12$. The number of degrees of freedom is $(r-1)(k-1) = 2*1 = 2$. In Table A2, it can be seen that $\chi^2_{0.05,2} = 5.99$. Since $X^2 < \chi^2_{0.05,2}$ it is impossible to reject the null hypothesis at the 0.05 level.

Chi-2 Goodness of fit test. A Chi-2 test can also be carried out in order to check if measurements are taken from a certain distribution, e.g., the normal distribution. In this case a goodness of fit test is performed according to:

Chi-2, goodness of fit	
Input	Data as frequencies for one group (i.e., O_1, O_2,... O_n, where O_i represents the number of observations in category i). Compare with Table 19.
H_0	Measurements are from a certain distribution.
Calculations	Compute $X^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i}$, where E_i is the expected number of observations in category i if H_0 is true and n is the number of categories.
Criterion	Reject H_0 if $X^2 > \chi^2_{\alpha,f}$, where f is the number of degrees of freedom determined as $f = n-e-1$, and e is the number of parameters that must be estimated from the original data (see below). $\chi^2_{\alpha,f}$ is the upper α percentage point of the Chi-2 distribution with f degrees of freedom, which is tabulated in, for example, Table A2 and [Siegel88]. This is a one sided test.

If the goodness of fit test is performed for a continuous distribution, the possible values that can be measured must be divided into intervals so that each interval can represent one value. This must, for example, be done for the normal distribution.

If the distribution of H_0 is completely specified (for example, $P(X = 1) = 2/3$, $P(X = 2) = 1/3$) then no parameters must be estimated from the measured data (i.e. $e = 0$). On the other hand, for example, if the null hypothesis only specifies that the values comply with a normal distribution, two parameters must be estimated. Both the mean value and the standard deviation of the normal distribution must be estimated, otherwise it is not possible to determine the values of the different expected values, E_i, for the intervals. Therefore, in this case, $e = 2$.

Example: Chi-2 Goodness of fit test for normal distribution. 60 students have developed the same program and the measured size is displayed in Table 26.

Table 26. *Measured size*

757	758	892	734	800	979	938	866	690	877	773	778
679	888	799	811	657	750	891	724	775	810	940	854
784	843	867	743	816	813	618	715	706	906	679	845
708	855	777	660	870	843	790	741	766	677	801	850
821	877	713	680	667	752	875	811	999	808	771	832

The null hypothesis is that the data is normally distributed, and the alternative hypothesis that it is not. Based on the data, the mean and the standard deviation can be estimated:

$\bar{x} = 794.9833$, and $s = 83.9751$.

The range can be divided into segments which have the same probability of including a value if the data actually is normally distributed with mean $\bar{x}$ and standard deviation s. In this example the range is divided into 10 segments. In order to find the upper limit (x) of the first segment the following equation should be solved:

$P(X < x) = 1/10$ where X is $N(\bar{x}, s)$, which in terms of the standard normal distribution corresponds to

$P(X_S < (x-\bar{x})/s) = 1/10$ where X_S is N(0, 1), which is the same as

$$P(X_S < z) = \int_{-\infty}^{z} \frac{1}{\sqrt{2\pi}} e^{(-y^2)/2} dy = 1/10 \text{ where } X_S \text{ is N(0, 1) and } x = sz + \bar{x}.$$

These equations can be solved in a number of different ways. One way is to iterate and use a computer to help find z or x. Another way is to use a table of the standard normal distribution (which shows $P(X_S < z)$ for different values of z). This type of table is available in most books on statistics. It is also possible to use a special-

ized table that directly shows the limit values for the segments, i.e. the values of z. This type of table is available in [Humphrey95].

The resulting segment boundaries and the number of values that fall into each segment are shown in Table 27.

Table 27. *Segments.*

Segment	Lower boundary	Upper boundary	Number of values
1		687.3	8
2	687.3	724.3	6
3	724.3	750.9	4
4	750.9	773.7	6
5	773.7	795	5
6	795	816.3	9
7	816.3	839	2
8	839	865.7	6
9	865.7	902.6	9
10	902.6		5

The expected number of values (E_i) in each segment is 60/10 = 6. This means that $X^2 = 7.3$. The number of degrees of freedom is 10–2–1 = 7. In Table A2, it can be seen that $\chi^2_{0.05,\,7} = 14.07$. Since $X^2 < \chi^2_{0.05,\,2}$ it is impossible to reject the null hypothesis at the 0.05 level. If we look at a histogram, see Figure 24, of the data, we can see that it seems to be fairly normally distributed.

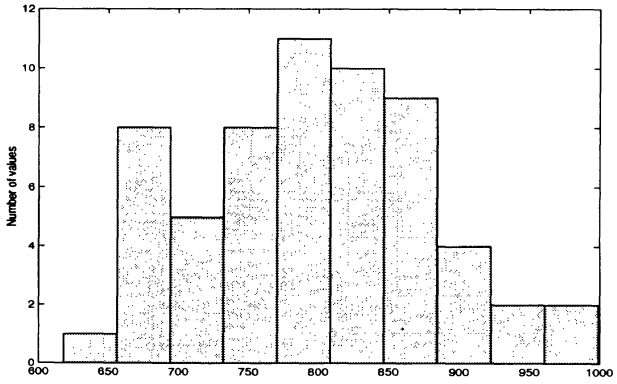


Figure 24. *Histogram.*

Chi-2 test final remarks. The Chi-2 test is based on certain assumptions, which are likely to be fulfilled if the expected values, E_i, are not too small. A rule of thumb is, if the number of degrees of freedom (f) is equal to 1, the Chi-2 test should not be used if any of the expected frequencies are less than 5. If $f>1$ the Chi-2 test should not be used if more than 20% of the expected frequencies are less than 5 or any of them is less than 1. It should be observed that sometimes the test is used although the expected frequencies are not fulfilled. In these cases, this is a calculated risk.

One way to obtain larger expected frequencies is to combine related categories to new categories. However, the new categories must be meaningful. For more information concerning the Chi-2 test refer, for example, to [Siegel88].

8.3.13 Model adequacy checking

Every statistical model is relying on specific assumption regarding, e.g., distribution, independence and scales. If the assumptions are invalidated by the data set, the results of the hypothesis testing are invalid. Hence, it is crucial to check that all assumptions are fulfilled.

The checking of model adequacy is made depending on the assumptions. Below we describe three cases:

- **Normal distribution**. If a test assumes that the data is normally distributed, a Chi-2 test can be made to assess to which degree the assumption is fulfilled. The Chi-2 test is described above.
- **Independence**. If the test assumes that the data is a sample from several independent stochastic variables, it is necessary to check that there is no correlation between the sample sets. This may be checked with scatter plots and by calculating correlation coefficients as discussed in the beginning of this section.
- **Residuals**. In many statistical models, there is a term that represents the residuals (statistical error). It is often assumed that the residuals are normally distributed. A common way to check this property is to plot the residuals in a scatter plot and see that there is no specific trends in the data (the distribution looks random).

8.3.14 Drawing conclusions

When the experiment data has been analyzed and interpreted, we need to draw conclusions regarding the outcome of the experiment. If the hypotheses are rejected we may draw conclusions regarding the influence of the independent variables on the dependent variables, given that the experiment is *valid*, see Chapter 6.

If, on the other hand, the experiment cannot reject the null hypothesis, we cannot draw any conclusions on the independent variables influence on the dependent

variable. The only thing we have shown, in this case, is that there is no statistically significant difference between the treatments.

If we have found statistically significant differences, we want to make general conclusions about the relation between independent and dependent variables. Before this can be done we need to consider the *external validity* of the experiment, see Chapter 6. We can only generalize the result to environments that are similar to the experimental setting.

Although the result of the experiment may be statistically significant, it is not necessarily that the result is of any practical importance. Assume, for example, that method X was shown with a high statistical significance to be 2% more cost effective than method Y, although there is a high statistical significance, the improvement of changing from method Y to method X might not be cost-effective.

It may also be vice versa; although the experiment results may not be statistically significant or have a low statistical significance, the lessons learned from the experiment may still be of practical importance. The fact that a null hypothesis cannot be rejected with a certain level of significance does not mean that the null hypothesis is true. There may be problems with the design of the experiment, such as real threats to validity or too few data samples. Furthermore, depending on the situation and objective of the study, we may settle for a lower statistical significance since the results are of high practical importance. This issue is also related to the discussion regarding threats to the validity, see Section 6.9.

When finding a significant correlation between a variable A and a variable B, we cannot, in general, draw the conclusion that there is a **causal relation** between A and B. There may be a third factor C, that causes the measurable effects on A and B.

The conclusions drawn based on the outcome of the experiment, are input to a decision, e.g. that a new method will be applied in future projects, or that further experimentation is needed.

9 PRESENTATION AND PACKAGE

When an experiment has been carried out, the intention is often to present the findings. This could for example be done in a paper for a conference, a report for decision-making, a package for replication of the experiment or as educational material. The packaging could also be done within companies to improve and understand different processes. In this case, it is appropriate to store the experiences in an experience base according to the concepts discussed in [Basili94a].

During the presentation and packaging of an experiment there are issues that are necessary to consider. It is essential not to forget important aspects or necessary information, needed in order to enable others to replicate or take advantage of the experiment, and the knowledge gained through the experiment. Therefore, an outline for an experiment report to be used when preparing, for example, an article for publication or lab package is presented in this chapter.

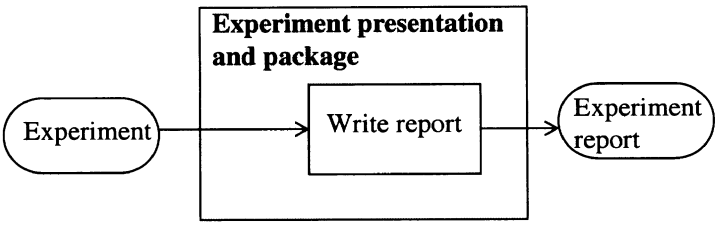


Figure 25. *Overview of presentation and package.*

9.1 An experiment report outline

The report outline consists of a number of different headings, which are presented below.

9.1.1 Introduction

An introduction to the research area and the objective of the research is the first part of a report. A short descriptive text providing an overview and motivation for the experiment should be provided. Information about the intentions of the work can also be included to clarify and capture the readers' interest.

9.1.2 Problem statement

After the introduction a deeper description of the background to the problem should be presented. This includes information about how the problem has been observed and the reasons for the investigation. This provides the reader with an understanding of why the research has been carried out and why there is a need for it. It is also suitable to provide an overview of related work at this point of time. The related work is important to provide a picture of how the current experiment is related to work conducted previously.

In this section, an experiment definition should be provided, describing the purpose of the experiment, i.e. the question we seek an answer to and its boundaries, see Chapter 5. Specifying the definition in detail helps to better focus on the purpose of the research and eases replication of the experiment.

9.1.3 Experiment planning

The context of the experiment including the hypotheses, which are derived from the problem statement, is described in detail in this section. It includes defining the variables necessary to measure, both the independent and the dependent, and to show that they are measured properly, and thus explaining the strategy for measurement and data analysis.

A description of how the data is analyzed should be included and when applicable also an alternative analysis strategy. A presentation of the subjects should be provided and a full description of the experiment design, for example, the type of design, see Chapter 6. The discussion about the experiment's conclusion, internal, construct and external validity should be provided here together with the possible threats against the plans.

The purpose for describing these items is to enable other persons to both understand the design so that it is visible to the reader that the results are trustworthy and

to enable replication of the study. In short, it should help the reader to get deeper a understanding of what has been done.

9.1.4 Experiment operation

The first part to describe is how the operation is prepared, see Chapter 7. It is important to include descriptions of aspects that will ease replication of the experiment and to give insight into how activities have been carried out. The preparation of the subjects has to be presented. Information such as whether they attended some lessons or not is important to provide. The execution of the experiment should also be presented and how data was collected during the experiment.

Validation procedures of the data collection are another issue that has to be stressed and it has to be reported if sidesteps have been taken from the plans. All information is aimed to provide a case for the validity of the data and to highlight problems.

9.1.5 Data analysis

A presentation of the data analysis, where the calculations are described together with the assumptions for using some specific analysis model, should be provided. Information about sample sizes, significance levels and application of tests must also be included so that the reader will know the prerequisites for the analysis. The reasons for the actions taken, for example outlier removal, should be described to avoid misunderstandings in the interpretation of the results. For more information see Chapter 8.

9.1.6 Interpretation of results

Raw results from the analysis are not enough to provide an understanding of the results and conclusions from the experiment. An interpretation must also be provided, see Chapter 8. It includes the rejection of the hypothesis or the inability to reject the null hypothesis. The interpretation summarizes how the results from the experiment may be used.

The interpretation should be done with references to validity, see Chapter 6. Factors that might have had an impact on the results should be described.

9.1.7 Discussion and conclusions

Finally, the discussions about the findings and the conclusions are presented as a summary of the whole experiment together with the outcomes, problems, deviations from the plans and so forth. The results should also be related to work reported previously. It is important to address similarities and differences in the findings.

Ideas for future work might also be included in this section and information about where more information can be found to get a deeper insight to the experiment and to ease replication of the experiment.

9.1.8 Appendix

Information that is not vital for the presentation could be included in appendices. This could, for example, be the collected data and more information about the subjects and objects. If the intention is to produce a lab package, the material used in the experiment could be provided here.

10 LITERATURE SURVEY

The objective of this chapter is to provide material for further studies within the area of software engineering experimentation. The intention is to provide references to examples of experiments in the software engineering domain and present some resources for further work in the area. The experiments show examples of different areas in the software engineering domain in which experiments are conducted, and thereby which areas are *not* covered yet. Furthermore, two experiments are included in this book. In Chapter 11, an experiment is presented concerning programmers' abilities related to their background. The purpose of that experiment is to illustrate the experiment process. In Chapter 12, an experiment concerning the use of C and C++ is presented, which is intended as an example of how an experiment may be reported.

The purpose of the survey in this chapter is to give pointers to published articles. Thus, the objective is *not* to provide a comprehensive treatment of all articles, including experiment design, all results and so forth. The articles can be used for further studies in the application of the methodology presented in this book. More can be learned by critical review of the referenced articles or the experiments in Chapters 11 and 12, for example, by use of the checklist in Section 13.3. In addition, the experiments can inspire researchers to replicate studies and to conduct new experiments that are not yet performed.

The survey is not exhaustive in terms of presented experiments; on the contrary, it is intentionally selective. Primarily, archival journals in software engineering are

surveyed. Furthermore, one area is covered in depth, while other areas are surveyed more briefly.

In the area of inspections and inspection techniques, many experiments are presented. Among the experiments there are studies aiming at replicating earlier experiments. Some experiments confirm earlier results, and others contradict what was concluded in an earlier experiment. Therefore, a sequence of experiments related to inspections is presented in Section 10.1. There are also experiments conducted in other areas within the software engineering domain, for example, programming languages, design techniques and software maintenance. In Section 10.2, a selection of such experiments is presented. Section 10.3 presents resources for further studies and practice of software engineering experimentation.

10.1 Inspection experiments

The existing literature on empirical software engineering includes a number of studies related to inspections, where experimentation has shown to be a relevant research strategy. In particular, there is a sequence of experiments which addresses inspection of requirements documents presented in Section 10.1.1.

The reported experiments represent different types of experiments. There are original experiments and replicated experiments, experiments conducted in educational environments and those performed in an industrial setting.

In addition to the sequence presented in Section 10.1.1, experiments concerning effectiveness of code inspections and experiments investigating the value and role of review meetings are surveyed in Section 10.1.2.

10.1.1 Requirements inspection experiments

The experiments presented in this section compares different reading techniques, such as Defect-Based Reading (DBR), Checklist-Based Reading (Checklist) and reading without any specified technique (Ad Hoc). DBR is a scenario-based approach, where different reviewers have different responsibilities and are guided in their reading by specific scenarios. In DBR, the scenarios concentrate on specific defect classes. The Checklist approach uses a list of items that shall be checked by the reviewers. The checklists are generally tailored for specific documents and specific environments.

This sequence of experiments uses two general requirements documents as objects, a water level monitoring system (WLMS, 24 pages) and an automobile cruise control system (CRUISE, 31 pages) with 42 and 26 defects respectively. The specifications were written using the SCR tabular notation [Heninger80]. The experiments are summarized below.

Maryland-95. The first study in the sequence is the *Maryland-95* study [Porter95], which compared DBR with Ad Hoc and Checklist in an academic environment. The experiment was run twice with 24 subjects on each occasion. The requirements documents used were the WLMS and the CRUISE. The reported results are the following.

- *Result 1*: DBR reviewers had significantly higher defect detection rates than either Ad Hoc or Checklist reviewers.
- *Result 2*: DBR reviewers had significantly higher defect detection rates for those defects that the scenarios were designed to uncover, while all three methods had similar defect detection rates for other defects.
- *Result 3*: Checklist reviewers did *not* have significantly higher defect detection rates than Ad Hoc reviewers.
- *Result 4*: Review meetings produced *no* net improvement in the defect detection rate – meeting losses offset meeting gains.

Bari. The *Bari* study [Fusaro97] is a replication of the Maryland-95 study. It compared DBR with Ad Hoc and Checklist in an academic environment using the WLMS and CRUISE documents. The experiment had 30 subjects. The results are as follows.

- *Result 1*: DBR did *not* have significantly higher defect detection rates than either Ad Hoc or Checklist.
- *Result 2*: DBR reviewers did *not* have significantly higher defect detection rates for those defects that the scenarios were designed to uncover, while all three methods had similar defect detection rates for other defects.
- *Result 3*: Checklist reviewers did *not* have significantly higher defect detection rates than Ad Hoc reviewers.
- *Result 4*: Review meetings produced *no* net improvement in the defect detection rate – meeting losses offset meeting gains.

The replicated study gives hence other results than the original one in the first two items. The authors discuss a set of factors that may have impacted the experiment operation.

Another three replications with minor changes in the experiment design are presented: the *Strathclyde* study, the *Linköping* study and the *Lucent* study.

Strathclyde. The *Strathclyde* study [Miller98] compared DBR with Checklist in an academic environment using the WLMS and CRUISE documents. The experiment had 50 subjects.

- *Result 1*: In the WLMS document, DBR did *not* have significantly higher defect detection rates than Checklist.

- *Result 2*: In the CRUISE document, DBR had significantly higher defect detection rates than Checklist.
- *Result 3*: Review meetings produced *no* net improvement in the defect detection rate – meeting losses offset meeting gains.

In this replication, the results were interacting with the documents. With one document, the results were the same as in the original study, with the other document it was not.

Linköping. The *Linköping* study [Sandahl98] compared DBR with Checklist in an academic environment using the WLMS and CRUISE documents. More defects were added to the list of total defects. The experiment had 24 subjects.

- *Result*: DBR reviewers did *not* have significantly higher defect detection rates than Checklist reviewers.

This experiment is based on the same lab package, but an additional set of defects or anomalies was identified, adding up to 76 defects for CRUISE and 73 defects for the WLMS specification. Furthermore, the design and analysis techniques were different from the earlier studies.

Lucent. The *Lucent* study [Porter98b] replicated the Maryland-95 study in an industrial environment using 18 professional developers at Lucent Technologies. The replication was successful and completely corroborated the results from the Maryland-95 study.

This fact highlights an interesting question concerning replication of experiments. Among this sequence of replications, the same researchers conducted the only one that corroborated the results, as did the original experiment. Perhaps it is not enough to use the same lab package in order to replicate an experiment. Too much knowledge about the experiment may be tacit. Possible solutions to the problem is to define the lab packages in more detail, for example, via a well defined experiment process, or to do experiments with staff from different research groups.

Summary. The Maryland-95 and Lucent studies indicate that a defect-based approach gives a higher defect detection rate. The Bari, Strathclyde and Linköping studies could, however, not corroborate this with statistically significant results, which motivates further studies to increase the understanding of defect-based reading. The sequence of studies summarized above is a salient example of how independently conducted replications of experiments in different environments are used to strengthen or question theories and hypotheses.

Another series of experiments, with requirements documents, evaluates the Perspective-Based Reading (PBR) compared to Checklist or Ad Hoc inspections. PBR is a scenario-based technique, like the DBR, but in this case different roles or stakeholders are represented in the reading scenarios. The *NASA* study [Basili96a] com-

pares PBR with Ad Hoc in an industrial environment. The experiment consisted of a pilot study with 12 subjects and a second experiment with 13 subjects. There were two groups of requirements documents used; general requirements documents: an automatic teller machine (ATM, 17 pages), a parking garage control system (PG, 16 pages), and two flight dynamics requirements documents (27 pages each). A lab package for this experiment as well as reports from replications can be found on the Internet. Some links to resources in experimentation are provided in Section 10.3.

10.1.2 Code inspection experiments

A question that has been debated for many years is the efficiency of code inspections versus other means for defect detection. Many arguments are raised for or against the use of inspections. To base the discussion on empirical evidence, Basili and Selby defined an early experiment to investigate three different techniques [Basili87], 1) code reading by stepwise abstraction, 2), functional testing using equivalence partitioning and boundary value analysis, and 3) structural testing with coverage criteria. The study was conducted in academic and industrial environments, involving a total of 74 subjects in three phases, categorized in three levels of expertise. The experiment used a fractional factorial design. Among the results, it can be noted that experienced reviewers were more effective with code reading than testing and that university students did not differ in efficiency among the three techniques. Furthermore, different types of defects were identified with the three techniques.

Lott and Rombach present a survey of experiments comparing different defect-detection techniques [Lott96]. In particular, a study by Kamsties and Lott, extending the Basili and Selby study [Basili87], is presented in detail. The study compares functional testing, structural testing and code reading. It was operated at two different occasions with university students, the first with 27 and the second with 23 students. The results show that the techniques did not differ significantly with respect to effectiveness. However functional testing identified defects rapidly, but it took longer time to isolate the faults. Code reading on the other hand took longer time to identify the defects, but the isolation took very little additional time.

Another question that is debated concerning inspections is the size of the review team. Porter et al have conducted an industrial on-line experiment that ranged over 18 months [Porter97b]. In an ongoing development project, 130 inspection occasions were scheduled, each was assigned a particular number of reviewers and it was decided whether one or two sequential inspections should be conducted. These characteristics of the inspections were decided, based on random assignment according to the experiment design. The data did not give direct answers to the hypotheses, despite the large data set. Just a few conclusions concerning team size and sessions could be drawn.

The researchers therefore continued the analysis of the experiment to find sources of variation in software inspection results [Porter98b]. They tried to build a prediction model that is applicable in the particular experiment context, hence it should not be generalized to larger domains. It can be noted from the variation study that the presence or absence of two individual reviewers in the inspection team "strongly affects the outcome of the inspection" [Porter98b]. Hence, it is not easy to generalize the experiment results, but the experiments are a step of further understanding of causes and effects in software inspections.

Finally, a question discussed in application of software inspections is the role of the review meeting. Are the meetings only for collection of data or do they contribute to the defect detection? Many of the above referenced experiments present meeting gains in terms of share of the identified defects which were found during the review meeting, for example, 3.9% and 4.7% for two different documents in the study by Porter et al [Porter95]. In addition, meeting losses are measured, i.e. the defects that are found by reviewers before the meeting, but not reported at the meeting. The question of meeting gains was raised and investigated informally by Votta [Votta93]. In order to study the issue in depth, an experiment was launched at University of Hawaii [Johnson98] with students as subjects and C++ programs as objects. The experiment design is one factor with two treatments (with and without review meeting). The study did not find any significant differences between reviews with and without meeting, with respect to the number of discovered defects. However, there was a difference in cost in that meeting-based reviews were more costly per defect found. On the other hand, meetings were effective in filtering out false positives and they made the reviewers feel more confident in their reviews.

The results indicate that there are still much to understand concerning reviewers's performance in inspections. An observational study with both qualitative and quantitative issues has been conducted to investigate what happens in the discussions at a review meeting [Seaman98], but this type of studies is outside the scope of this book.

10.2 Other experiments in Software Engineering

Software maintenance is an area in which some experiments have been conducted. Software maintenance is very much related to understanding of software artifacts. Therefore, different notations and technologies are investigated with respect to the time taken to identify and perform the needed changes for a certain maintenance task.

An early study of maintainability compared the effort involved in changing two different systems, implemented in two different languages [Rombach87]. The

experiment design hence had two implementations of each system, one in each language.

More recently, the difference is investigated between performance in the maintenance of object-oriented versus structured design documents [Briand97]. Furthermore, the impact of the document's quality on the maintenance results was investigated. Students were assigned to maintenance tasks in a 2*2 factorial design, with the factors development approach (levels OO/SD) and document quality (levels good/bad). It is interesting, however, that due to educational constraints, all combinations of the factors were not possible, which the authors point out. The students were promised to see examples of both levels of both factors, hence the repeated measures could not be performed for some of the hypotheses in the study.

Within the domain of object-oriented programming, Daly et al [Daly96] studied the issue of inheritance depth. This article presents the results from three different experiment occasions: a first experiment with an internal replication and a second experiment. The experiments used two different programs written in two versions, with and without inheritance. In the first experiment and the replication, programs with a maximum inheritance level of 3 were used and in the second experiment, programs with a maximum inheritance level of 5 were used.

The results show that for the first experiment and the replication, the program version with inheritance took the shortest time to maintain. In the second experiment with the inheritance level of 5, the null hypothesis could not be rejected. However, the statistical methods applied did not allow for a small effect to be observed. Hence there may be a small effect, which could not be observed in the analysis.

Another two example experiments concerning language issues are represented by an experiment on visual or textual representations by Kiper et al [Kiper97] and an experiment on the gains of type checking by Prechelt and Tichy [Prechelt98]. In the first case, an experiment is used to investigate if a graphical or a textual representation of program logic best supports comprehension. The second experiment investigated whether the use of type checking impacts on productivity and defects. The experiment could successfully reject all the hypotheses and hence conclude that type checking improves productivity and reduces defects.

Finally, another area in which some experiments have been conducted is concerned with on-line or off-line guidance for the development process [Lott97]. The subjects were given a task to test a program, using a particular testing process. The process guidance was given either on-line or off-line, and different quantitative and qualitative aspects were evaluated. The experiment contributed to an initial understanding of some variation factors with respect to process guideline support although the hypotheses could not be rejected with statistical significance.

10.3 Resources

During the last decade, there has been a growing interest in empirical studies and experimentation in software engineering. This is can be highlighted by the following:

- ISERN (International Software Engineering Research Network) was established in 1993. The network is focused on empirical studies as a means for improvement in software engineering.
- Kluwer Academic Publishers launched an international journal devoted to empirical studies in 1996. The journal is entitled "Empirical Software Engineering: An International Journal".
- A conference devoted to empirical studies has been created at Keele University, UK. The conference is called "Empirical Assessment in Software Engineering (EASE)", and it was held for the first time in 1997.

More information about the above items can be found via the web page of this book, see Preface.

These three examples show that empiricism and experimentation are essential elements in software engineering, and that the interest is growing.

11 EXAMPLE: EXPERIMENT PROCESS

The primary objective of the presentation of this experiment is to illustrate experimentation and the steps in the experiment process introduced in the previous chapters. The presentation of the experiment in this chapter is more detailed than described in Chapter 9, and the structure proposed in Chapter 9 is hence not followed faithfully. The experience report outline is closer to the experiment described in Chapter 12.

The objective of the presented experiment is to investigate the performance in using the Personal Software Process (PSP) [Humphrey95, Humphrey97] based on the individual background of the people taking the PSP course. The experiment, as reported here, is part of a larger investigation of the individual differences in performance within the PSP. The larger study is reported in [Wohlin98b].

The PSP is an individual process for a systematic approach to software development. The process includes, for example, measurement, estimation, planning and tracking. Furthermore, reuse is a key issue, and in particular the reuse of individual experiences and data. The PSP course introduces the process in seven incremental steps adding new features to the process using templates, forms and process scripts.

For the sake of simplicity, only two hypotheses are evaluated here. The data set for the larger study [Wohlin98b] can be found in Section 13.2. The experiment presented in this chapter uses a subset of the data.

11.1 Definition

11.1.1 Goal definition

The first step is to become aware that an experiment is a suitable way to analyze the problem at hand. In this particular case, the objective of the empirical study is to determine the differences in individual performance for people using the PSP given their background.

The experiment is motivated by a need to understand the differences in individual performance within the PSP. It is well known, and accepted, that software engineers perform differently. One objective of introducing a personal process is to provide support for the individuals to improve their performance. In order to support improvement in the best possible way, it is important to understand what differences we still can expect within the PSP and if it is possible to explain and thus understand the individual differences.

Object of study. The object of study is the participants in the Personal Software Process (PSP) course and their ability in terms of performance based on their background and experience. Watts Humphrey in his two books on the subject [Humphrey95, Humphrey97] defines the Personal Software Process.

Purpose. The purpose of the experiment is to evaluate the individual performance based on the background of the people taking the PSP course. The experiment provides insight in to what we can expect in terms of individual performance when using the PSP.

Perspective. The perspective is from the point of view of the researchers and teachers, i.e. the researcher or teacher would like to know if there is any systematic difference in the performance in the course based on the background of the individuals entering the PSP course. This also includes people who may want to take the course in the future or to introduce the PSP in industry.

Quality focus. The main effect studied in the experiment is the individual performance in the PSP course. Here, two specific aspects are emphasized. We have chosen to focus on Productivity (KLOC / development time) and Defect density (faults / KLOC), where KLOC stands for thousands of lines of code.

Context. The experiment is run within the context of the PSP. Moreover, the experiment is conducted within a PSP course given at the Department of Communication Systems, Lund University in Sweden. This study is from the course given in 1996-97, and the main difference from the PSP as presented in [Humphrey95] is that we

provided a coding standard and a line counting standard. Moreover, the course was run with C as a mandatory programming language independently of the background of the students. The experimental context characterization is "multi-test within object study". The study is focused on the PSP or more specifically the 10 programs in [Humphrey95] denoted 1A-10A (the PSP is the object), hence one object. The PSP course is taken by a large number of individuals (this particular year, we had 65 students finishing the course). Thus, we have 65 subjects in the study, see Section 5.1. Thus, from this definition the study can be judged as being a controlled experiment. The lack of randomization of students, i.e. the students signed up for the course, means that we all the same lack one important ingredient to make it fully into a controlled experiment. This is hence classified as a quasi-experiment, see Section 5.1.

11.1.2 Summary of definition

The summary is made according to Section 5.2.

Analyze *the outcome of the PSP*
for the purpose *of evaluation*
with respect to *the background of the individuals*
from the point of view of the *researchers and teachers*
in the context of *the PSP course.*

11.2 Planning

11.2.1 Context selection

The context of the experiment is a PSP course at the university, and hence the experiment is run off-line (not industrial software development), it is conducted by graduate students (normally students in their fourth year at the university), and the experiment is specific since it is focused on the PSP. The ability to generalize from this specific context is further elaborated below when discussing threats to the experiment. The experiment addresses a real problem, i.e. the differences in individual performance and our understanding of the differences.

The use of the PSP as an experimental context provides other researchers with excellent opportunities to replicate the experiment. Furthermore, it means that we do not have to spend much effort in setting up the experiment in terms of defining and creating the environment in which the experiment is run. [Humphrey95] defines the experimental context, and hence there is no need to prepare forms for data collection and so forth.

11.2.2 Hypothesis formulation

An important aspect of experiments is to know and to formally state clearly what we intend to evaluate in the experiment. This leads us to the formulation of a hypothesis (or several hypotheses). Here, we have chosen to focus on two hypotheses. Informally, they are:

1. Students both from the Computer Science and Engineering program and the Electrical Engineering program have taken the course. We know that students from the Computer Science and Engineering program normally have taken more courses in computer science and software engineering, and hence we believe that they have a higher productivity than students from the Electrical Engineering program will.
2. As part of the first lecture, the students were asked to fill out a survey regarding their background in terms of experiences from issues related to the course. This can be exemplified with, for example, knowledge in C. The students were required to use C in the course independently of their prior experience of the language. Thus, we did not require that the students had taken a C-course prior to entering the PSP course, which meant that some students learned C within the PSP course. This is not according to the recommendation by Humphrey, see [Humphrey95]. The hypothesis based on the C experience is that students with more experience in C make fewer faults per lines of code.

Based on this informal statement of the hypotheses, we are now able to state them formally and also to define what measures we need to evaluate the hypotheses.

1. Null hypothesis, H_0: There is no difference in productivity (measured as lines of code per total development time) between students from the Computer Science and Engineering program (CSE) and the Electrical Engineering program (EE).

   H_0: Prod(CSE) = Prod(EE)

   Alternative hypothesis, H_1: Prod(CSE) $\neq$ Prod(EE)

   Measures needed: student program (CSE or EE) and productivity (LOC/hour).
2. Null hypothesis, H_0: There is no difference between the students in terms of number of faults per KLOC (1000 lines of code) based on the prior knowledge in C.

   H_0: Number of faults per KLOC is independent of C experience.

   Alternative hypothesis, H_1: Number of faults per KLOC changes with C experience.

   Measures needed: C experience and Faults/KLOC.

   The hypotheses mean that we have to collect the following data:

- Student program: measured by CSE or EE (nominal scale)

- Productivity is equal to LOC / Development time. Thus, we have to measure program size (lines of code according to the coding standard and the counting standard) and development time (minutes spent developing the program). The development time is translated to hours when the productivity is calculated. It should be noted that we have chosen to study the total program size (the sum of the 10 programming assignments) and the development time for all 10 programs. Thus, we do not study the individual assignments.

  Lines of code are measured by counting the lines of code using a line counter program (ratio scale). The lines counted are new and changed lines of code.

  Development time is measured in minutes (ratio scale).

  Productivity is hence measured on a ratio scale.
- C experience is measured by introducing a classification into four classes based on prior experience of C (ordinal scale). The classes are:

  1. No prior experience.
  2. Read a book or followed a course.
  3. Some industrial experience (less than 6 months).
  4. Industrial experience (more than 6 months).

  The experience in C is hence measured on an ordinal scale.
- Faults/KLOC is measured as of the number of faults divided by the number of lines of code.

The hypotheses and measures put constraints on the type of statistical test we may perform, at least formally. The measurement scales formally determine the application of specific statistical methods, but we may want to relax these requirements for other reasons. We will return to this issue below, when discussing the actual type of design we have in our experiment.

11.2.3 Variables selection

The independent variables are student program and experience in C. The dependent variables are productivity and faults/KLOC.

11.2.4 Selection of subjects

The subjects are chosen based on convenience, i.e. the subjects are students taking the PSP course. The students are a sample from all students at the two programs, but not a random sample.

11.2.5 Experiment design

The problem has been stated, and we have chosen our independent and dependent variables. Furthermore, we have determined the measurement scales for the variables. Thus, we are now able to design the experiment. Let us start by addressing the general design principles:

Randomization. The object is not assigned randomly to the subjects. All students use the PSP and its ten assignments. The objective of the study is not to evaluate the PSP vs. something else. The subjects are, as stated above, not selected randomly; they are the students that have chosen to take the course. Moreover, the assignments are not made in random order. The order is, however, not important, since the measures used in the evaluation are the results of developing 10 programs.

Blocking. No systematic approach to blocking is applied. The decision to measure the 10 programs and evaluate based on this, rather than looking at each individual program, can be viewed as providing blocking for differences between the 10 programs. Thus, blocking the impact of the differences between individual programs.

Balancing. It would have been preferable to have a balanced data set, but the experimental study is based on a course where the participants have signed up for the course, and hence we have been unable to influence the background of people and consequently unable to balance the data set.

Standard design types. The information available is compared with the standard type of designs outlined in Chapter 6. Both designs can be found among the standard types, and the statistical tests are available in this book.

1. The definition, hypotheses and measures for the first evaluation means that the design is: one factor with two treatments. The factor is the program and the treatments are CSE or EE. The dependent variable is measured on a ratio scale, and hence a parametric test is suitable. In this particular case, the t-test will be used.
2. The second design is of the type "one factor with more than two treatments". The factor is the experience in C, and we have four treatments, see the experience grading above. The dependent variable is measured on a ratio scale and we can use a parametric test for this hypothesis too. The ANOVA test is hence suitable to use for evaluation.

11.2.6 Instrumentation

The background and experience of the individuals is found through a survey handed out at the first lecture, see Table 37 in Chapter 13. This data provides the input to the characterization of the students, and hence are the independent variables in the experiment. The objects are the programs developed within the PSP course. The guidelines and measurements are provided through the PSP [Humphrey95].

11.2.7 Validity evaluation

In our particular case, we have several levels of validity to consider. *Internal validity* is primarily focused on the validity of the actual study. *External validity* can be divided into PSP students at Lund University forthcoming years, students at Lund University (or more realistically to students at the CSE and EE programs), the PSP in general, and for software development in general. The *conclusion validity* is concerned with relationship between treatment and outcome, and our ability to draw conclusions. *Construct validity* is about generalizing the result to the theory behind the experiment.

The internal validity within the course is probably not a problem. The large number of tests (equal to the number of students) ensures that we have a good internal validity.

Concerning the external threats, it is highly probable that similar results should be obtained when running the course in a similar way at Lund University. It is more difficult to generalize the results to other students, i.e. students not taking the course. They are probably not as interested in software development and hence they come from a different population. The results from the analysis can probably be generalized to other PSP courses, where it is feasible to compare participants based on their background in terms of computer science or electrical engineering or experience of a particular programming language.

The major threat regarding the conclusion validity is the quality of the data collected during the PSP course. The students are expected to deliver a lot of data as part of their work with the course. Thus, there is a risk that the data is faked or simply not correct due to mistakes. The data inconsistencies are, however, not believed to be particularly related to any specific background, hence the problem is likely the same independent of the background of the individuals. The conclusion validity is hence not considered to be critical.

The construct validity includes two major threats. The first threat is that the measurements as defined may not be appropriate measures of the entities we want to measure, for example, is "LOC/Development time" a good measure of productivity? The second major threat to the construct validity is that it is part of a course, where the students are graded. This implies that the students may bias their data, as they believe that it will give them a better grade. It was, however, in the beginning

of the course emphasized that the grade did not depend on the actual data. The grade was based on timely and proper delivery, and the understanding expressed in the reports handed in during the course.

The results are found for the PSP, but they are likely to hold for software development in general. There is no reason that people coming from different study programs or having different background experience from a particular programming language perform differently between the PSP and software development in general. This is probably valid when talking about differences in background, although the actual size of the difference may vary. The important issue is that there is a difference, and the actual size of the difference is of minor importance.

11.3 Operation

11.3.1 Preparation

The subjects (students) were not aware of what aspects we intended to study. They were informed that we wanted to study the outcome of the PSP course in comparison with the background of the participants. They were, however, not aware of the actual hypotheses stated. The students, from their point of view, did not primarily participate in an experiment; they were taking a course. All students were guaranteed anonymity.

The survey material was prepared in advance. Most of the other material was, however, provided through the PSP book [Humphrey95].

11.3.2 Execution

The experiment was executed over 14 weeks, during which the 10 programming assignments were handed in regularly. The data was primarily collected through forms. Interviews were used at the end of the course, primarily to evaluate the course and the PSP as such.

The experiment was, as stated earlier, run within a PSP course and in a university environment. The experiment has not been allowed to affect the course objectives. The main differences between running the PSP solely as a course has been the initial survey of the students' background.

11.3.3 Data validation

Data was collected for 65 students. After the course, the achievements of the students were discussed among the people involved in the course. Data from six students was removed, due to that the data was regarded as invalid or at least questionable. Students have not been removed (at this stage) from the evaluation based on the actual figures, but due to our trust in the delivered data and whether or not the data is believed to be representative. The six students were removed due to:

- Data from two students was not filled in properly.
- One student finished the course much later than the rest, and that student had a long period when no work was done with the PSP. This may have affected the data.
- Data from two students was removed based on that they delivered their assignments late and required considerably more support than the other students did, hence it was judged that the extra advice may have affected their data.
- Finally, one student was removed based on that the background was completely different than the others.

This means removing six students out of the 65, hence leaving 59 students for statistical analysis and interpretation of the results.

11.4 Analysis and interpretation

11.4.1 Descriptive statistics

As a first step in analyzing the data, we use descriptive statistics to visualize the data collected.

Study program vs. productivity. Figure 26 shows the productivity for the two study programs, when dividing the population into classes based on productivity. The first class includes those with a productivity between 5-10 lines of code per hour. Thus, the eighth class includes those with a productivity between 40-45 lines of code per hour. From Figure 26, we see that students from the Electrical Engineering program (EE) seem to have a lower productivity. Moreover, it is noticeable to see that the variation of the distribution seems to be larger among the students from the Computer Science and Engineering program (CSE).

In total, we have 32 CSE students and 27 EE students. The mean value for CSE students is 23.0 with a standard deviation of 8.7, and for the EE students the mean value is 16.4 with a standard deviation of 6.3. To gain an even better understanding of the data, a box plot is done, see Figure 27.

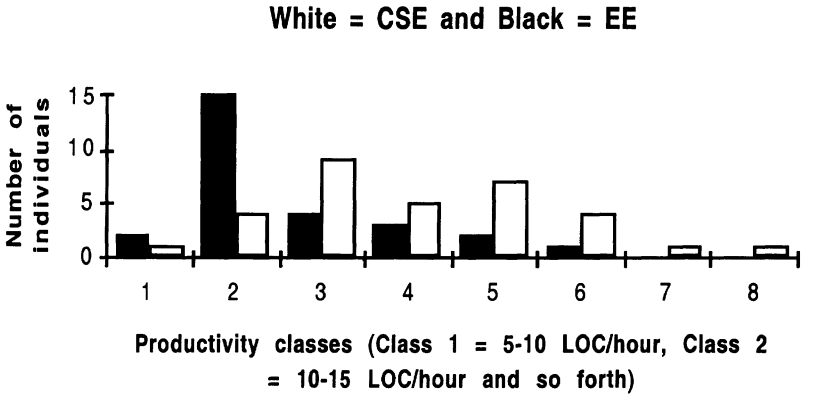


Figure 26. *Frequency distribution for the productivity (in classes).*

The whiskers in the box plot are constructed as proposed in [Frigge89] and discussed in Chapter 8. We use a value that is the length of the box multiplied with 1.5 and added or subtracted from the upper and lower quartiles respectively. For example, for CSE we have (see Figure 27): median = 22.7, box length = 29.4 – 17.6 = 11.8, the upper tail becomes: 29.4 + 1.5 * 11.8 = 47.1. There is, however, an exception to this rule and that is that the upper and lower tail should never be higher or lower than the highest and lowest value in the data set, hence the upper tail becomes 42.5, which is the highest value. This exception is introduced to avoid negative values or other types of unrealistic values. The other values in Figure 27 are found in a similar way.

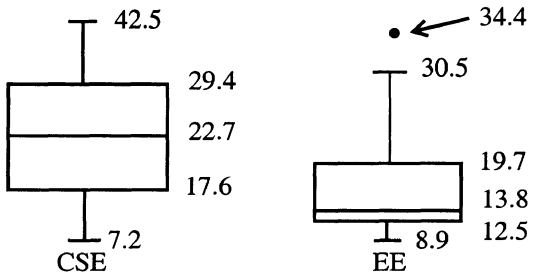


Figure 27. *Box plot of productivity for the two study programs.*

From Figure 27, we can see that there is a clear pattern that the EE students have a lower productivity. Thus, it may be possible to prove the difference in a hypothesis test. The t-test is used below.

It is also important to look at outliers in comparison to the upper and lower tails. For the CSE-students, we have no outliers. For the EE students, we have one outlier, i.e. 34.4. This outlier is not removed later since it is not judged as an extreme value. It is an unusual value, but it is determined to keep the value in the analysis.

C experience vs. faults/KLOC. The number of students for each class of C experience is shown in Table 28 together with the mean and median values, and standard deviation for respective class.

Table 28. *Faults/KLOC for the different C experience classes.*

Class[a]	Number of students	Median value of faults/ KLOC	Mean value of faults/KLOC	Standard deviation of faults/KLOC
1	32	66.8	82.9	64.2
2	19	69.7	68.0	22.9
3	6	63.6	67.6	20.6
4	2	63	63.0	17.3

a. The different classes are explained in Section 11.2.2.

From Table 28, we can see that the distribution is skewed towards no or little experience of C. If we look at the mean values of faults/KLOC, there seems to be a tendency that the more experienced students create fewer faults. The standard deviation is, however, extremely large, and the median varies unexpectedly in comparison with the mean value and the underlying hypothesis. The standard deviation for the first class is very high and a further investigation of the data is recommended. Thus, we use the box plot for this data set too.

Box plots are constructed for all four experience classes. The plots for classes 2, 3 and 4 reveal nothing, all values are within the boundaries of the whiskers, and hence the upper and lower tails become equal to the highest and lowest value respectively.

The box plot for the first class is more interesting, and it is shown in Figure 28.

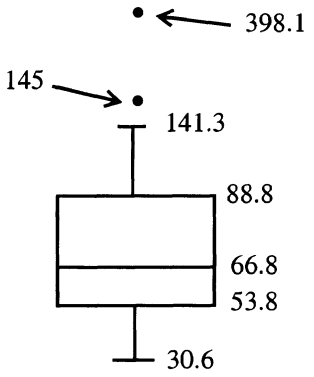


Figure 28. *Box plot for faults/KLOC for class 1.*

From Figure 28, we see that the lower tail is equal to the lowest value of faults/KLOC. The upper tail on the other hand is not equal to the highest value, and hence we have one or more outliers. We actually have two outliers, namely 145 and 398.1. The latter outlier is an extreme outlier; it is more than ten times higher than the lowest value. It is also almost three times as high as the second highest value. Thus, we can conclude that the high standard deviation can be explained with the extreme outlier. For the second hypothesis, the ANOVA test is used.

The descriptive statistics have provided a better insight into the data, both in terms of what we can expect from the hypothesis testing and to potential problems caused by outliers.

11.4.2 Data reduction

Data reduction can always be debated, since as soon as we remove data points or in other ways reduce the data set we loose some information. Two separate ways of reducing data can be identified:

- single data points can be removed, for example, outliers, or
- we may analyze the data and come to the conclusion that due to high inter-correlation between some variables, we should combine measures into some more abstract measure.

This means that we can either remove data points or reduce the number of variables. In the case of removing data points, the main candidates are the outliers. It is by no means obvious that we should remove all outliers, but they are certainly candidates for removal. It is important to remember that we should not remove data

points just because they do not fit with our belief or hypothesis. On the other hand, it is important to remove data points which may make a completely valid relationship invalid, due to that we include, for example, an extreme outlier, which we do not expect if replicating the study.

To reduce the number of variables we need statistical methods for data reduction. Some examples are principal component analysis and factor analysis [Kachigan86, Kachigan91, Manly94]. These types of methods are not considered here, as the objective is not to reduce the number of variables.

It is probably better to be restrictive in reducing a data set, as there is always a risk that we aim for a certain result. Thus, for the data presented above, we choose to only remove the extreme outlier for the number of faults/KLOC. After removing the extreme outlier, the data for class 1 is summarized in Table 29.

Table 29. *Faults/KLOC for C experience class 1.*

Class	Number of students	Median value of faults/ KLOC	Mean value of faults/KLOC	Standard deviation of faults/KLOC
1	31	66	72.7	29.0

The removal of the outlier decreased the mean value and standard deviation considerably. The mean number of faults/KLOC is still highest for class 1. However, the differences between the classes are not that large. After reducing the second data set with one data point, we are now able to perform the statistical test. This is where the hypotheses are evaluated.

11.4.3 Hypothesis testing

The first hypothesis regarding higher productivity for students following the Computer Science and Engineering program is evaluated using a t-test. An ANOVA test is applied to evaluate the hypothesis that more experience in C means fewer faults/ KLOC.

Study program vs. productivity. The results from the t-test (unpaired, two-tailed) are shown in Table 30.

Table 30. *Results from the t-test.*

Factor	Mean diff.	Degrees of freedom (DF)	t-value	p-value
CSE vs. EE	6.1617	57	3.283	0.0018

From Table 30, we can conclude that H_0 is rejected. There is a significant difference in productivity for students coming from different study programs. The *p*-value is very low so the results are highly significant. The actual reason for the difference has to be further evaluated.

C experience vs. faults/KLOC. This hypothesis is evaluated with an ANOVA test (factorial). The results of the analysis are shown in Table 31.

Table 31. *Results from the ANOVA-test.*

Factor: C vs. Faults/KLOC	Degrees of freedom (DF)	Sum of squares	Mean square	F-value	p-value
Between treatments	3	3483	1160.9	0.442	0.7236
Error	55	144304	2623.7		

The results from the analysis are not significant, although we observe some difference in terms of mean value, see above, we are unable to show that there is a significant difference in terms of number of faults/KLOC based on C experience.

Since the number of students in class 3 and 4 is very limited, class 2, 3 and 4 are grouped together to study the difference between class 1 and the rest. A t-test was performed to evaluate if it was possible to differentiate between class 1 and the grouping of class 2-4 into one class. No significant results were obtained.

11.5 Summary and conclusions

We have investigated two hypotheses:

1. Study program vs. productivity
2. C experience vs. faults/KLOC

We are able to show that students from the Computer Science and Engineering program are more productive. This is in accordance with the expectation, although not formally stated in the hypothesis. The expectation was based on the knowledge that most students from CSE has taken more computer science and software engineering and courses than those from the Electrical Engineering program.

It is not possible to show with any statistical significance that experience in C influences the number of faults/KLOC. This is interesting in relation to that in [Humphrey95], it is recommended that one should follow the PSP course with a well-known language in order to focus on the PSP and not the programming lan-

guage as such. The results obtained may indicate one or several of the following results:

- The difference will become significant with more students.
- The number of faults introduced is not significantly affected by the prior experience. We have a tendency to make a certain number of faults when developing software. The type of faults may vary, but the total number of faults introduced is about the same.
- The inexperienced students may write larger programs, which directly affects the number of faults/KLOC. This is further evaluated in [Wohlin98b].

Other explanations can probably also be found, but they all have one thing in common that is the need for replication. Thus, we come back to the important issue of replication to enable us to understand, and hence control and improve the way software is developed. Furthermore, other factors must be studied as well.

From a validity point of view, it is reasonable to believe that students (in general) from a computer science program have higher productivity than students coming from other disciplines have. This is more or less inherent from the educational background and it is no surprise.

Since, we were unable to show any statistically significant relation between experience in a programming language and the number of faults/KLOC; we have no conclusions to generalize. Further studies are needed, either through replication of our PSP experiment or similar studies in other environments.

12 EXAMPLE: C VERSUS C++

Abstract
Different programming languages are assumed to be differently effective, and in this chapter a comparison of C and C++ is presented. The study is performed within the context of a defined software process, the Personal Software Process, and the comparison is made with respect to eight different measures. The results of the study are presented and the validity of the results is investigated and discussed. The results of the study are not statistically significant, although they indicate that more defects are introduced if C++ is used than if C is used.

12.1 Introduction and problem statement

New programming languages are regularly developed in order to address some problem with the existing languages. Object-oriented languages were developed based on the notions of encapsulation and inheritance, and the assumption is that it is a better way of developing software. An important question to answer is, of course, if a specific language actually is better than another is? In [Tichy98], the theories concerning object orientation, functional orientation, and formal logic are listed as theories that have not yet been tested systematically. That is, the choice of programming language is one of a number of computer science theories that have not been tested empirically.

In this chapter C and C++, i.e., one functional programming language and one object oriented language, are compared based on a number of evaluation criteria. Earlier empirical investigations have been reported. In [Harrison96], C++ is compared to SML (a functional language) with respect to 16 defined measures. In that study, a professional developer develops the same product (12 sets of algorithms and a set of general-purpose library functions for image analysis) in both SML and C++. Significant differences are found for the time taken to test the programs (SML takes longer time), the number of known errors per KLOC (higher for SML), and some other measures dealing with function calls and reuse. In the study presented in this chapter, C and C++ are compared instead of SML and C++, the development is carried out by many developers (students). The participants in this study have less experience and the measures are not the same.

The study is carried out within the context of the Personal Software Process (PSP) [Humphrey95]. In other words, we have used the PSP course as the environment for the study. That is, the technical question regarding the choice of programming language is empirically investigated within the context of a defined software development process, the PSP. The PSP provides opportunities for different empirical studies of software development, and the study presented here is one example. The PSP is briefly described below and the use of the PSP as context for empirical studies is further discussed in [Wohlin98a].

The study has been set up as a quasi-experiment performed in retrospect according to the terminology in [Robson93]. We are in control of the experiment except for the choice of programming language, which means that the subjects are not assigned to the treatments (C or C++) at random. Instead the persons choose programming languages themselves, and the study is performed based on the data that is collected during the course. If the language was randomly assigned to the students, this would mean that many of the students would use a language they have never used before. The objective of the study is not to compare the languages for people who are using them for their first time, but to compare the languages for people who know the languages. The lack of randomization does, however, mean that there is a risk that other unknown and unwanted factors affect the outcome of the comparison. The objective has been to identify and make such effects as small as possible.

The first year the course was given every student was forced to use the same programming language irrespective of his or her knowledge of it. The data from this year is not used in the study. This is further discussed in Section 12.2.4.

The study is based on the initial hypothesis that C++ is better than C due to the benefits of object-orientation. The study results in no clearly significant results although it indicates that more faults are introduced with C++ than C. One conclusion of the study is that further studies are necessary, and the intention is that it should be possible to replicate the study, or perform a modified study, based on the description.

The chapter is organized as follows. In Section 12.2 the design of the study is presented and the data it is based on is described. In Section 12.3 the results of the analysis are presented and in Section 12.4, some conclusions are presented.

12.2 Experiment planning

The PSP is further described in the next section and how the PSP can be used as context for empirical studies is further discussed in [Wohlin98a].

12.2.1 The study environment, PSP

The study is performed within the context of the PSP. The PSP course involves developing ten different programs (#1A-#10A), and the course participants develop the programs independently of each other. All development is carried out following a defined process that is the same for every participant. The process is enhanced during the assignments, from a basic process in the first assignment to a more advanced process in the last assignment. There are, however, some phases which are included in all processes:

- Planning deals with planning the development of the program in the assignment.
- Design means producing a high level design.
- Code is concerned with the implementation based on the design.
- Compile means compiling the program until no additional faults can be found.
- Test means executing the program and correcting it until all faults that have found are removed.
- Postmortem deals with summarizing and reporting experience from development. In this phase, the measurements used in the experiment are summarized.

In addition to the above tasks, some additional tasks are performed in some processes. These tasks include a design review (in assignment #7A-#10A) after the design, and a code review (in assignment #7A-#10A) after the coding. The assignments each require about 100 - 500 minutes to perform. Assignment #10A requires more time than the others do, while the other assignments require about the same time.

The choice of programming language is not crucial for the PSP course. This means that different people can use different programming languages, and hence that the PSP course is suitable as an environment to the study explained in this chapter.

The study is performed with seven assignments: #4A-#10A. These assignments include development of procedures for different types of statistical analysis. For

further information concerning the assignments, refer to [Humphrey95]. Assignment #1A is not included since it is the first assignment, and the participants have no experience of the course when they perform it. Not including the first assignment in the study means that some confounding effects due to learning could be avoided. Assignments #2A and #3A are not included in the study because they deal with counting the number of lines of code according to a coding standard that should be defined for the language. It would not be possible to distinguish the effect of the different individual coding standards, and hence different definitions of how to count lines of code, from the effect of implementing the program in different programming languages. It is not equally hard to implement a line of code counter for different definitions of lines of code in different programming languages.

The measurements required in the study can be derived from the measurements made according to the course. This means that the participants do not have to perform any additional measurements because of the study. During the course, the participants carry out measurements, and the measures of interest in the study are based on these measurements. The participants have been told that their data will be analyzed, but they have not been told the exact nature of the analysis. If the participants are told the exact nature of the analysis this could affect the result.

12.2.2 Variables in the study

Independent variables. The independent variables describe the treatments and are thus the variables for which the effects should be evaluated. The independent variable of interest in this study is the choice of programming language. It is nominal and can take two values: C and C++.

It is important to consider effects of other potential variables as well. The PSP course has been given in a number of classes, which has been identified as a potential confounding factor in the study. The class can be seen as an additional independent variable in the study, which means that analysis can be performed in order to investigate if the performance is different for different classes. In this type of study it is in many cases impossible to completely remove confounding factors, such as the class. It is therefore important to identify them and perform the analysis in such a way that the effects of them are as small as possible.

Other types of analyses can be performed as a complement and in this study some complementary statistical tests have been performed.

Dependent variables. The dependent variables are the response variables that describe the effects of the treatments described by the independent variables. The measured values of the dependent variables represent mainly the performance of the different programming languages, and hence describe how good the different languages are. Since the performance of a programming language can be measured

in a number of different ways, a number of dependent variables are defined and evaluated in the study. The following dependent variables have been defined:

- **Total time:** This is a measure of the total time required to develop the programs, and it is assumed that it is representative of the total cost developing the programs.
- **Planning time:** This is a measure of the time required to plan for the assignment. Since the programming language does not generally have to affect the planning time, a difference in performance of different programming languages does not have to be seen in this variable. However, there may be some differences of the planning time since the time includes time for developing a conceptual design, which is the basis for time estimation. There is a possibility that the time for developing the conceptual design depends on whether a functional or object oriented language should be used. Our assumption is, however, that the conceptual design is very informal and that the choice of programming language does not affect the planning time in this way.

  The main reason to include the planning time in the study is to use it as an indication of that something else but the programming language affects the results of different groups of people participating in the study. If a difference in planning time could be seen this indicates that there are confounding factors affecting the result of the study.
- **Code to test:** This is a measure of the time required to perform the work from code to test. This time includes, in addition to coding and testing also compilation and code review. This measure represents the costs of the phases, which are directly affected by the programming language.
- **Code to test relative:** This is a measure of the relative time required for activities from code to test, derived as the code to test time divided by the total time. If all participants require equal effort in phases which are not affected by the programming language, a difference in performance in phases which are affected by the programming language should be seen in the relative time from code to test. However, if groups with different programming languages not only require different effort in the affected phases but also spend different effort in phases not affected by the programming language, this kind of threat could be revealed through this variable.

  For example, if measurements show that the group using programming language X needs less time than the group using programming language Y in the time from code to test, this may mean that programming language X requires less time than programming language Y. On the other hand, if the difference is only due to that the group that uses programming language X generally spends less work on the assignments, then they spend equally less work on the phases that are not affected by the programming language. This means that the relative

time spent in phases directly affected by the programming language is the same as for the group using programming language Y. That is, if a difference is found between two groups in the time from code to test, but no difference can be found for the relative time from code to test, care must be taken to prevent false conclusions from being drawn.

- **Total defects:** This is a measure of the total number of defects introduced during development of the programs.
- **Code defects:** This is a measure of the number of defects introduced in the coding phase.
- **Review efficiency:** This is a measure of how efficient code reviews are, and it is calculated as the number of faults found in code review divided by the number of faults found in code review, compile and test. This measure can show if there is a difference in the ability to find defects in reviews for the different programming languages. A high review efficiency indicates that the code is easy to understand, while a low review efficiency indicates that the code is hard to understand.
- **Total quality cost:** This is a measure of the total time used to remove faults after coding, i.e., code review, compile and test. This measure could indicate that the cost of removing defects is different for different programming languages.
- **Reuse:** When the programs are developed it is encouraged to reuse code from earlier assignments. This measure is the relative amount of reused code in all seven programs. For some groups of students the measure is, however, unavailable because of unavailable data.

Measures based on time are often referred to as effort or cost, and a low value is generally considered better than a high value. The measures refer to the effective time, i.e. ineffective time, such as breaks, is excluded. Concerning the measures that deal with the number of faults that are found, a low value is considered better than a high value, since the assignments are small and all faults in most cases can be found.

The dependent variables are measured for every participant for the whole course, i.e., for assignments #4A-#10A. For example, the total time for a certain person is the sum of times that the person has spent on the seven assignments.

12.2.3 Analysis

The results of the measurements can, for every dependent variable, be analyzed with standard procedures for hypothesis testing. When the effects of two independent variables on one depending variable is analyzed, the standard procedure is to perform an analysis of variance (ANOVA). In this analysis, it is possible to detect

significant effects of each of the independent variables and also of their interaction [Montgomery97].

12.2.4 Data used in the study

The study is performed with data from the Personal Software Process (PSP) in two courses held at Lund University (Lund97) and Linköping University (Link96) in Sweden. The study is conducted in retrospect and hence there is no real operation phase solely for the experiment.

The two classes can be characterized according to:

- Lund97: This course was held with M.Sc. students at Lund University in Sweden. The participants chose programming language themselves in the beginning of the course. Some students used languages other than C and C++. Their data is not included in the study.
- Link96: This course was held with Ph.D. students at Linköping University in Sweden. The participants chose programming language themselves in the beginning of the course. Some students used languages other than C and C++. Data from them is not included in the study. From this class there is no data available concerning reuse.

The course has been held in one additional class, Lund96[1]. This course was held with 65 M.Sc. students at Lund University in Sweden, but only 59 of the students did deliver data that was judged to be trustworthy. Thus, only the data from the 59 students is used in the analysis. The participants did not choose programming language; instead everyone had to use C, irrespective of their prior knowledge and experience of the language. That is, the main difference between Lund96 and the other two classes is that the students could not choose programming language themselves. It would be too hard to separate the effect of this from the effects of the choice of programming languages in an analysis with this threat. In addition to this, the data set would be highly unbalanced if data from this class are included. Therefore, data from Lund96 is not included in the study.

Another reason to exclude data points is that they represent outliers in the data. Potential outliers can, for example, be identified through box plots and scatter plots. The data from Link96 and Lund97 has been analyzed in order to reveal outliers. Data from two people show very high values with respect to some of the measures. Our analysis does not indicate that the high values have to do with the choice of programming language. Instead, it depends probably more on the people participating in the study. One of the persons used C and one of them used C++. Data from two other persons shows very low values with respect to planning time. This indi-

1. The course has since this study been run one more time, i.e. Lund98.

cates that there have been some problems in the reporting or that the persons have not performed the planning as they were supposed to. One of the persons used C and one of them used C++.

A decision must be taken whether or not to use the identified potential outliers. In this case the four identified points (one person using C and one person using C++ from each class (Lund97 and Link96), i.e. in total four persons) are considered so unreliable that they are excluded from the study. The resulting number of students from which data is used in the study is cross tabulated for class and programming language in Table 32.

Table 32. *Students and their choice of programming language.*

Class	C	C++	Total
Link96	5	12	17
Lund97	4	12	16
Total	9	24	33

12.2.5 Validity of results

Introduction. In this section the validity of the findings is discussed with respect to internal and external validity (see for example [Robson93, Basili96a, Porter94]). Internal threats to validity are threats to conclusions about the cause and effects in the study and deal with the validity of the specific study. External validity is not concerned with the validity of the specific study. Instead, it describes the ability to generalize the results.

Internal validity. The following threats to the internal validity have been considered:

- **History:** When treatments are applied at different occasions there is a possibility that the circumstances are not the same for every occasion. Since the participants perform the PSP assignments independently, the assignments are performed at many different occasions. If special events and disruptions occur, this affects only a few assignments for a few participants in a random fashion. While no systematic trends have been identified, this is not considered an important threat to this study. This threat also includes that the process changes between the assignments in the PSP course.
- **Maturation:** During the execution of the experiment, the subjects may react differently over time.

One effect to consider is that the participants can become bored or tired, or they may become more positive to the tasks in the experiment. Since the PSP course is performed during the time corresponding to a whole semester, this threat cannot be disregarded. No serious effects have, however, been noticed, even if some of the participants during the course may have increased their interest in the assignments and some participants of course may have decreased their interest. If these effects have occurred, nothing indicates that the effects are more severe for any specific programming language. Instead, if the effects have occurred, they have probably occurred randomly to participants with different programming languages. Therefore, this effect is not considered important.

Another effect to consider is the learning effect. It is possible that the participants learn for example the programming language during the course. The effect is reduced by not including the first three assignments in the study. Learning effects are further discussed in below.

- **Testing:** When participants perform assignments repeatedly it is possible that they learn the procedures of the study. Since the participants only knew that their data should be analyzed, but were not aware of the exact nature of the analysis or the objectives of the analysis, they have not changed because of it.
- **Instrumentation:** If the material, such as forms, is badly designed, this can cause lower validity. Since the study is run within the PSP, without any extra instrumentation, this is not considered an important threat.
- **Selection:** In a controlled experiment, participants from a larger population are randomly sampled for the study and randomly grouped into groups that receive different treatments. In this study the participants chose the programming language themselves. The course objective has not been to randomize with respect to subjects and treatments. If people were assigned to different programming languages by random, this would mean that they would have to learn the language during the study. Since the objective is not to investigate how people learn the different languages, this is not considered positive, and this is one of the reasons that data from Lund96 was discarded from the study. For the study it would of course be better to choose people randomly from a larger population of people. This is, however, not possible due to obvious practical reasons.
- **Mortality:** If too many persons leave the study, this can be a threat to the validity. We have not seen any systematic trend in people leaving the course. To our knowledge, the people who left the course did this independently of their choice of programming language. Therefore, this threat is not considered important for the experiment.

External validity. As describe above, the external validity deals with the ability to generalize the results. The following aspects of external validity have been considered:

- **Generalizability of setting:** The setting is one aspect of generalizability. The assignments are conducted with relatively small assignments, which to a large extent deal with implementation of mathematical functions. The assignments are performed individually by single persons. This means that the results of the study at least are valid for small assignments developed individually. Whether the result is valid for larger assignments developed by teams of developers must be elaborated for different cases.
- **Generalizability of subjects:** Another aspect of generalizability is to be able to generalize from the participants in the study. The study is conducted with M.Sc. students and Ph.D. students who are familiar with software development. Whether the results are valid for industrial software developers must be elaborated for different cases.

12.3 Analysis and interpretation

12.3.1 Factor analysis

Before the measured data is analyzed further, it is interesting to compare the different dependent variables. The objective is to determine if the variables address completely different aspects, or they address aspects that are in some way related. This can, for example, be analyzed with factor analysis based on principal component analysis [Kachigan91]. A factor analysis results in orthogonal factors derived as linear combinations of the variables. The factors can be rotated in order to obtain a solution that is easier to interpret. The result of a factor analysis is shown in Table 33, where the correlation of the variables with the derived factors (Components) is tabulated.

Table 33. *A factor analysis.*

Variable	Factor 1	Factor 2
Total time	**0.875**	−0.138
Planning time	0.582	−0.629
Code to test	**0.901**	0.292
Code to test relative	0.141	**0.903**
Total quality cost	**0.889**	0.315
Total defects	**0.876**	−0.014
Code defects	**0.887**	0.0465
Review efficiency	0.0363	0.202

The two derived factors are:

- Factor 1, which is most related to the total time, the time from code to test, the total quality cost, the total number of defects, and the number of code defects. One explanation to that effort measures and defects measures are related to the same factor may be that much effort is required if many faults are introduced.
- Factor 2, which is most related to the relative time from code to test.

The review efficiency and the planning time are not included in the two factors.

The experiment's dependent variables are not independent of each other. All dependent variables related to Factor 1 are correlated to each other. A decision could be taken to reduce the number of variables by removing some of the dependent variables related to Factor 1. All dependent variables are, however, retained in this study.

It is interesting to note that the planning time is not highly related to the same factor as the total time and the time from code to test. This indicates that the time spent in planning is not directly related to the time in later phases.

The reuse measure is not included in the factor analysis, since too few data points are available for this measure.

12.3.2 Main analysis

In Table 34, the result of one analysis of variance test for every dependent variable is shown.

Table 34. *An analysis of variance (ANOVA).*

Dependent variable	p(language)	p(class)	p(interaction)
Total time	0.940	0.014	0.463
Planning time	0.873	0.745	0.386
Code to test	0.906	0.008	0.732
Code to test relative	0.993	0.268	0.610
Total quality cost	0.870	0.005	0.877
Total defects	0.140	0.348	0.542
Code defects	0.108	0.348	0.436
Review efficiency	0.507	0.943	0.518
Reuse[a]	0.396	-	-

a. Since reuse data is only available from one of the classes, a Mann Whitney test (see below) has been performed instead of an ANOVA test.

It can be seen that there is a significant difference (e.g. if a significance level of 0.05 is chosen) between the classes for a number of the variables (total time, code to test, and total quality cost). The differences may be due to that one of the classes consists of Ph.D. students and one consists of M.Sc. students. When investigating the mean values it can be seen that Ph.D. students spend less time in the phases measured by these variables. Since this is not related to the objectives of this study, this is not further analyzed. The advantage of the analysis of variance test is that it can distinguish between the effect of the class and the language, and therefore investigate the effects of the language, irrespective of the effects of the class.

Even if the differences between the programming languages are not statistically significant, the analysis results in low p-values for the differences between C and C++ for the total number of defects and the number of defects introduced in coding. The results indicate that there might be an effect of the programming language on these variables. If the mean values for the other values are investigated, it is not possible to see any major differences between the two languages. Either there are very small effects, or the effects are different for different classes (although there is no significant interaction effect). For the total number of defects and the number of defects introduced in code, there are however differences in mean values that indicate a difference between the languages.

It is, however, important to notice that the significance of the difference is low, i.e., there is a risk that it has appeared by chance. It is also important to notice that there are many dependent variables in the study, which means that there, by chance, is a risk of observing a difference in at least one of them. This is always a potential risk when multiple variables are analyzed. With this data, a multivariate analysis of variance (e.g., Pillai's Trace, Wilks' Lambda) does not succeed do detect a significant difference between the set of means for C and the set of means for C++. However, this does not generally have to mean that there are no differences. Here we choose to look at the two identified variables, but again, it is important to notice that the statistical significance is low and the results are uncertain.

The mean values of the two variables for each combination of programming language and class are displayed in Figure 29.

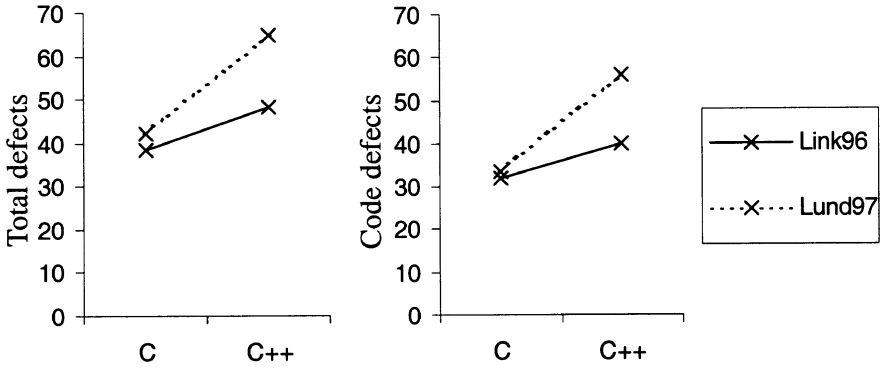


Figure 29. *Mean values for each combination of programming language and class for the two variables.*

The mean values indicate that the programming language effects the number of introduced defects for both classes. The effect is larger for Lund97 than for Link96.

Concerning the planning time, it could be noticed that there is no large difference between either the two classes or between the two groups of students using C and C++. In this case it is not so important, but the latter would be considered positive if there would have been a difference with respect to the time-based measures for the two groups using different programming languages. If there would have been a large difference in the planning time, it may have meant that there was another difference than the programming language between the groups using C and C++.

There is a logical similarity between the two variables and there is also a logical difference between the identified variables and the other because both variables are related to the number of introduced defects. It could also be noticed that both variables are related to the same factor from the factor analysis. In the analysis pre-

sented in Section 12.3.1, the two variables did not form a factor of their own. But when a factor analysis, as the one presented, is performed with the original data (i.e., all available data from 3 classes, 96 students including outliers) it results in three factors where one of them is related to only the two variables, see Table 35. This may indicate that there is an underlying commonality for the two variables, which also is intuitive and easy to explain. Concerning the large data set, there are, however, problems as pointed out previously, and conclusions should not be drawn solely on it.

Table 35. *A factor analysis for the large data set.*

Variable	Factor 1	Factor 2	Factor 3
Total time	**0.927**	0.135	– 0.304
Planning time	0.647	0.027	– 0.620
Code to test	**0.967**	0.214	0.0384
Code to test relative	– 0.0156	0.107	**0.883**
Total quality cost	**0.922**	0.287	0.0821
Total defects	0.278	**0.923**	0.0459
Code defects	0.210	**0.937**	0.125
Review efficiency	– 0.0178	– 0.437	0.377

12.3.3 Non-parametric analysis

Since the analysis presented above is based on analysis of variance, a number of assumptions have been made concerning the nature of the data. With this kind of data, there is always a risk that these assumptions are not fulfilled. The solution is in many cases to use non-parametric tests instead. In [Harrison96] where data of a similar type is handled, non-parametric tests were performed. In order to see if the two identified variables are identified with a non-parametric approach too, Mann-Whitney tests [Siegel88] have been performed for each variable to test the difference between C and C++. This test does not consider the differences between the classes, but it could be seen from Table 32 that the data is matched with respect to classes for each programming language. That is, for C, about half of the students belong to Link96 and about half of them belong to Lund97, and for C++ the number of students that belong to the different classes are equal. This type of design reduces the unwanted bias resulting from the differences between the classes when the programming languages are compared without considering the classes. The result of one Mann-Whitney test for each variable is shown in Table 36.

Table 36. *Results of a Mann-Whitney test.*

Dependent variable	p(language)
Total time	0.492
Planning time	0.585
Code to test	0.599
Code to test relative	0.686
Total quality cost	0.903
Total defects	0.129
Code defects	0.066
Review efficiency	0.793
Reuse	0.396

It can be seen that the results support the results from the ANOVA tests. The analysis indicates that the programming language affects the total number of defects and the number of defects introduced in coding. This gives some support for performing parametric analysis in the study.

12.3.4 Learning effects and different knowledge

As it is said in Section 12.2.5, validity threats of the study are that the different groups in the experiment have different knowledge of their programming language at the beginning of the course and that the learning effects are different for different groups. Learning effects in the PSP course are also discussed in [Wohlin98a] where it is concluded that there is no significant effect of the knowledge of the programming language before the course on the combined result of all ten assignments (#1A-#10A), see Chapter 11. This means that the learning effects should not be given too much attention in the study, since it is not certain that they result in large effects on the overall performance. The learning effects can, however, reveal if persons in a specific group have less knowledge before the course than another group.

If a learning effect can be found for one programming language this indicates that the persons which use the language in the beginning of the course do not know it as well as a group of people with no learning effect. One threat is already removed by removing one of the classes (Lund96) from the original data, and by not using data from the three first assignments (#1A-#3A). The objective of this section is to discuss whether learning effects remain in the study.

In Figure 30, the mean number of defects per LOC is shown for the 7 assignments. Notice that it is not meaningful to compare the number of defects per KLOC

for the two programming languages for a certain assignment, since the meaning of a LOC is not the same in different programming languages. The trends (i.e. the learning effects) of the curves can, however, be compared.

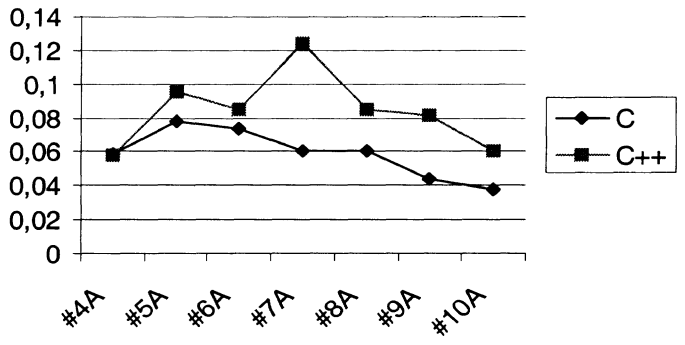


Figure 30. *Learning curves.*

If one of the languages would show a very large number of defects per LOC in the first assignments, we would have a dangerous learning effect, since it would not affect both languages equally. In this case there may be some learning during the course (decreasing number defect density after assignment #6A), which is natural since the objective of the course is that the work of the students should mature throughout the course. There is, however, no large difference between the two languages. We conclude that the learning threat is not crucial for this study.

12.4 Conclusions and further work

It could be noticed that there is a difference between the two classes. There is a significant difference between the two classes with respect to the total time, the time from code to test, and the total quality cost. In all three cases, the Ph.D. students in Link96 spent less effort than the M.Sc. students in Lund97 did.

There are no large differences between C and C++ where a clear statistical significance can be shown. This is mainly because the data set is too small. The data does, however, give indications of some differences. There seems to be a difference between the two programming languages with respect to the number of introduced defects. It also seems like more defects are introduced if C++ is used in comparison with using C.

It is not possible to show that the difference is statistically significant, but in the data there is a difference in mean values for C and C++. The difference between C and C++ should be subject to further studies. One approach is to perform similar studies on larger data sets or to perform a meta-analysis. The latter refers to combining results from several studies. Another approach is to focus on the defect injection and perform further studies on different data sets where C and C++ are compared with respect to only this aspect. If further studies can show the same difference, it would be necessary to investigate *why* more defects are introduced with C++ than with C.

13 EXERCISES

Explanations of the different types of exercises can be found in the Preface. In summary, the objective is to provide four types of exercises:

- **Understanding**. These exercises aim at highlighting the most important issues from each chapter. Exercises are available for Chapters 1-9.
- **Training**. The objective of these exercises is to encourage practicing experimentation. This includes setting up hypotheses and performing the statistical analysis.
- **Reviewing**. Two chapters include examples of experiments and one chapter provides a survey of some published experiments. The intention of this part is to provide help in reviewing and reading published experiments.
- **Assignments**. These exercises are formulated to promote an understanding of how experiments can be used in software engineering to evaluate methods and techniques.

The four types of exercises can be found in the subsequent sections.

The data sets used are available electronically from the home page of the book, see Preface.

13.1 Understanding

13.1.1 Introduction

1. Why can experiments be viewed as prototyping for process changes?
2. How can experiments be used in improvement activities?
3. Why are empirical studies important in software engineering?
4. When is the empirical research method best suited in software engineering in comparison with the scientific, engineering and analytic methods respectively?
5. Which three strategies are empirical methods divided into?

13.1.2 Empirical strategies

1. What is the difference between qualitative and quantitative research?
2. What is a survey? Give examples of different types of surveys in software engineering.
3. What is a case study? Discuss different ways of conducting a case study.
4. What is the Quality Improvement Paradigm and how does this relate to performing empirical studies?
5. How can the Experience Factory be combined with the Goal/Question/Metrics paradigm and empirical studies?

13.1.3 Measurement

1. What are measure, measurement and metric and how do they relate?
2. Which are the four main measurement scale types?
3. What are the difference between a direct and an indirect measure?
4. Which three classes are measurements in software engineering divided into?
5. What are internal and external attributes and how are they mostly related to direct and indirect measures?

13.1.4 Experiment process

1. What is a cause and effect relationship?
2. What is a treatment, and why is it sometime necessary to apply treatments in a random order?
3. What are dependent and independent variables respectively?

4. What are quasi-experiments? Explain why these are common in software engineering.
5. Which are the main steps in the experiment process, and why is it important to have distinct steps?

13.1.5 Definition

1. Why is it important to have set up clear goals with an experiment from the beginning?
2. Write an example of a goal definition for an experiment you would like to conduct.
3. Why is the context in an experiment important?
4. How can the context be characterized?
5. Explain how a series of studies can be used for technology transfer.

13.1.6 Planning

1. What are a null hypothesis and an alternative hypothesis?
2. What is type-I-error and type-II-error respectively, which is worst and why?
3. In which different ways may subjects be sampled?
4. What different types of experiment designs are available, and how do the design relate to the statistical methods to apply in the analysis?
5. Which four types of validity must be addressed in a study and why, what is the trade-off between different validity types?

13.1.7 Operation

1. Which factors should be considered when selecting subjects?
2. Why are ethical issues important in experimentation?
3. Why is it necessary to prepare the instrumentation carefully before an experiment?
4. What is data validation and why should it be done before the statistical analysis?
5. How should we handle subjects that have a personal interest in the outcome of the experiment?

13.1.8 Analysis and interpretation

1. What is descriptive statistics and what is it used for?

2. What is a parametric and non-parametric test respectively, and when can they be applied?
3. What is the power of a test?
4. What is a paired comparison?
5. Explain the ANOVA test briefly.

13.1.9 Presentation and package

1. Why is it important to document an experiment thoroughly?
2. What is a lab package? Can you find any lab packages on the Internet?
3. What do you think is a suitable introduction to an experiment? What should be included in the introduction?
4. Why is it important to report related work?
5. Why is it not enough just to provide the results from the analysis? In other words, why is a special interpretation of the results important?

13.2 Training

The exercises are preferably solved either using a statistical program package or tables from books in statistics. The tables in Appendix A may be used, but the tables provided are only for the 5% significance level, so if other significance levels are used then other sources must be used. It should be remembered that Appendix A has primarily been provided to explain the examples in Chapter 8.

13.2.1 Normally distributed data

The probably most complicated example of the statistical methods in Chapter 8 is the goodness of fit test for the normal distribution, see Section 8.3.12. Thus, it is appropriate to ensure a good understanding of that test. Carry out the same test, on the same data, see Table 26, using 12 segments instead.

13.2.2 Experience

In Chapter 11, the outcome of the Personal Software Process course is compared with the background of the students taking the course. The analysis conducted in Chapter 11 is only partial. The full data set is provided in Table 38 and Table 39.

In Table 37, the survey material handed out at the first lecture is presented. The outcome of the survey is presented in Table 38. The outcome of the PSP course is

presented in Table 39, where the following seven measures have been used to measure the outcome of the course:

- Size: The number of new and changed lines of code for the ten programs.
- Time: The total development time for the ten programs.
- Prod.: The productivity measured as number of lines of code per development hour.
- Faults: The number of faults logged for the ten programs. This includes all faults found, for example, including compilation faults.
- Faults/KLOC: The number of faults for each 1000 lines of code.
- Pred. Size: The absolute relative error in predicting program size. The figures show the error in absolute percentages, for example, both over- and underestimates with 20% are shown as 20 percent without any sign indicating the direction of the estimation error.
- Pred. Time: The absolute relative error in predicting the development time.

Based on the presentation in Chapter 11 and the data in Table 38 and Table 39 answer the following questions.

1. How can the survey be improved? Think about what constitutes good measures of background, experience and ability.
2. Define hypotheses, additional to those in Chapter 11, based on the available data. Motivate why these hypotheses are interesting.
3. What type of sampling has been used?
4. Analyze the hypotheses you have stated. What are the results?
5. Discuss the external validity of your findings. Can the results be generalized outside the PSP? Can the results be generalized to industrial software engineers?

Table 37. *Student characterization*

Area	Description	Answer
Study program (denoted Line)	Answer: Computer Science and Engineering or Electrical Engineering	
General knowledge in computer science and software engineering (denoted SE)	1. Little, but curious about the new course 2. Not my speciality (focus on other subjects) 3. Rather good, but not my main focus (one of a couple of areas) 4. Main focus of my studies	
General knowledge in programming (denoted Prog.)	1. Only 1-2 courses 2. 3 or more courses, no industrial experience 3. A few courses and some industrial experience 4. More than 3 courses and more than 1 year industrial experience	
Knowledge about the PSP (denoted PSP)	1. What is it? 2. I have heard about it 3. A general understanding of what it is 4. I have read some material	
Knowledge in C (denoted C)	1. No prior knowledge 2. Read a book or followed a course 3. Some industrial experience (less than 6 months) 4. Industrial experience	
Knowledge in C++ (denoted C++)	1. No prior knowledge 2. Read a book or followed a course 3. Some industrial experience (less than 6 months) 4. Industrial experience	
Number of courses (denoted Courses)	A list of courses was provided and the students were asked to put down a yes or no whether they had taken the course or not. Moreover, they were asked to complement the list of courses if they had read something else they thought was a particularly relevant course.	

Table 38. *Information from background survey.*

Subject	Line	SE	Prog.	PSP	C	C++	Courses
1	1	2	1	2	1	1	2
2	1	3	2	1	2	1	4
3	2	3	2	2	2	2	7
4	1	3	2	3	2	1	3
5	1	3	2	3	2	1	5
6	2	4	3	2	1	1	7
7	2	3	2	2	1	2	7
8	1	3	2	2	1	1	4
9	2	4	3	2	1	1	9
10	2	4	2	1	1	1	7
11	1	2	2	1	2	1	3
12	2	4	3	2	1	1	9
13	2	4	3	2	3	3	8
14	2	3	2	2	1	1	6
15	1	3	2	2	1	1	5
16	2	4	2	1	1	1	10
17	1	3	3	1	1	1	5
18	2	4	3	2	1	3	6
19	2	4	3	3	3	3	8
20	1	1	1	1	1	1	2
21	2	3	3	2	2	2	10
22	2	3	2	3	1	1	5
23	1	3	2	2	1	1	4
24	1	2	1	1	1	1	3
25	2	4	3	1	2	2	7
26	1	3	2	2	1	1	5
27	2	4	3	2	3	2	7
28	1	3	2	3	1	1	2
29	2	4	2	3	1	1	7
30	2	3	3	1	2	3	6
31	1	3	2	2	2	2	5

Table 38. *Information from background survey.*

Subject	Line	SE	Prog.	PSP	C	C++	Courses
32	2	3	3	1	2	2	10
33	2	4	3	1	1	1	5
34	1	2	2	1	2	2	3
35	1	2	1	1	1	1	2
36	1	2	1	2	1	1	2
37	1	2	2	2	2	2	2
38	2	4	2	2	2	1	6
39	1	2	1	2	1	1	2
40	2	4	3	1	4	4	7
41	2	3	3	2	2	2	8
41	2	4	3	2	2	2	9
43	1	3	2	1	1	1	3
44	1	4	3	2	3	2	7
45	2	4	2	2	2	1	6
46	2	2	4	2	4	4	7
47	2	4	3	2	3	2	7
48	1	2	2	2	1	1	2
49	1	3	3	1	1	1	3
50	2	3	2	3	1	1	8
51	2	4	2	4	2	2	8
52	2	4	3	3	3	2	8
53	2	4	3	3	2	2	10
54	1	2	1	2	1	1	2
55	1	2	2	2	1	1	4
56	2	3	2	1	1	1	8
57	1	2	3	1	1	1	4
58	2	4	3	3	1	1	6
59	1	2	2	2	2	1	4

Table 39. *Outcome from the PSP course.*

Subject	Size	Time	Prod.	Faults	Faults/ KLOC	Pred. Size	Pred. Time
1	839	3657	13.8	53	63.2	39.7	20.2
2	1249	3799	19.7	56	44.8	44.1	21.2
3	968	1680	34.6	71	73.3	29.1	25.1
4	996	4357	13.7	35	35.1	24.3	18.0
5	794	2011	23.7	32	40.3	26.0	13.2
6	849	2505	20.3	26	30.6	61.1	48.2
7	1455	4017	21.7	118	81.1	36.5	34.7
8	1177	2673	26.4	61	51.8	34.6	32.5
9	747	1552	28.9	41	54.9	51.0	18.2
10	1107	2479	26.8	59	53.3	22.6	14.0
11	729	3449	12.7	27	37.0	26.9	52.0
12	999	3105	19.3	63	63.1	26.0	19.8
13	881	2224	23.8	44	49.9	47.9	39.9
14	730	2395	18.3	94	128.8	63.0	20.3
15	1145	3632	18.9	70	61.1	33.3	34.8
16	1803	3193	33.9	98	54.4	52.9	21.8
17	800	2702	17.8	60	75.0	34.3	26.7
18	1042	2089	29.9	64	61.4	49.3	41.5
19	918	3648	15.1	43	46.8	49.7	71.5
20	1115	6807	9.8	26	23.3	34.1	22.4
21	890	4096	13.0	108	121.3	19.3	34.8
22	1038	3609	17.3	98	94.4	21.4	52.0
23	1251	6925	10.8	498	398.1	21.8	34.1
24	623	4216	8.9	53	85.1	40.5	36.3
25	1319	1864	42.5	92	69.7	43.7	45.0
26	800	4088	11.7	74	92.5	42.6	36.2
27	1267	2553	29.8	88	69.5	53.0	30.1
28	945	1648	34.4	42	44.4	33.3	17.9
29	724	4144	10.5	49	67.7	32.8	17.8
30	1131	2869	23.7	102	90.2	29.2	15.5

Table 39. *Outcome from the PSP course.*

Subject	Size	Time	Prod.	Faults	Faults/ KLOC	Pred. Size	Pred. Time
31	1021	2235	27.4	49	48.0	18.0	25.0
32	840	3215	15.7	69	82.1	85.6	54.0
33	985	5643	10.5	133	135.0	27.3	31.0
34	590	2678	13.2	33	55.9	83.0	20.0
35	727	4321	10.1	48	66.0	17.0	22.7
36	955	3836	14.9	76	79.6	33.3	36.8
37	803	4470	10.8	56	69.7	18.2	27.7
38	684	1592	25.8	28	40.9	35.0	34.1
39	913	4188	13.1	45	49.3	25.3	27.5
40	1200	1827	39.4	61	50.8	31.6	20.9
41	894	2777	19.3	64	71.6	21.3	22.4
41	1545	3281	28.3	136	88.0	35.0	16.1
43	995	2806	21.3	71	71.4	15.6	38.3
44	807	2464	19.7	65	80.5	43.3	26.4
45	1078	2462	26.3	55	51.0	49.1	51.6
46	944	3154	18.0	71	75.2	59.0	39.2
47	868	1564	33.3	50	57.6	50.4	45.2
48	701	3188	13.2	31	44.2	21.2	49.7
49	1107	4823	13.8	86	77.7	19.3	28.4
50	1535	2938	31.3	71	46.3	29.6	20.7
51	858	7163	7.2	97	113.1	58.4	32.9
52	832	2033	24.6	84	101.0	48.4	25.6
53	975	3160	18.5	115	117.9	29.5	31.5
54	715	3337	12.9	40	55.9	41.7	26.6
55	947	4583	12.4	99	104.5	41.0	22.3
56	926	2924	19.0	77	83.2	32.5	34.7
57	711	3053	14.0	78	109.7	22.8	14.3
58	1283	7063	10.9	186	145.0	46.5	26.6
59	1261	3092	24.5	54	42.8	27.4	45.3

13.2.3 Programming

In an experiment, 20 programmers have developed the same program, where 10 of them have used programming language A and 10 have used language B. Language A is newer and the company is planning to change to language A if it is better than language B. During the development, the size of the program, the development time, the total number of removed defects and the number of defects removed in test have been measured.

The programmers have been randomly assigned a programming language and the objective of the experiment is to evaluate if the language has any effect on the four measured variables. The collected data can be found in Table 40. The data is fictitious.

1. Which design has been used in the experiment?
2. Define the hypotheses for the evaluation.
3. Use box plots to investigate the differences between the languages in terms of central tendency and dispersion with respect to all four factors. Is there any outlier and if so should it be removed?
4. Assume that parametric tests can be used. Evaluate the effect of the programming language on the four measured variables. Which conclusions can be drawn from the results?
5. Evaluate the effect of the programming language on the four measured variables using a non-parametric test. Which conclusions can be drawn from the results? Compare the results to those achieved when using parametric tests.
6. Discuss the validity of the results and if it is appropriate to use a parametric test.
7. Assume that the participating programmers have chosen the programming language themselves. What consequences does this have on the validity of the results? Do the conclusions still hold?

Table 40. *Data for programming exercise.*

Prog. language	Program size (LOC)	Development time (minutes)	Total number of defects	Number of test defects
A	1408	3949	89	23
A	1529	2061	69	16
A	946	3869	170	41
A	1141	5562	271	55
A	696	5028	103	39
A	775	2296	75	29
A	1205	2980	79	11
A	1159	2991	194	28
A	862	2701	67	27
A	1206	2592	77	15
B	1316	3986	68	20
B	1787	4477	54	10
B	1105	3789	130	23
B	1583	4371	48	13
B	1381	3325	133	29
B	944	5234	80	25
B	1492	4901	64	21
B	1217	3897	89	29
B	936	3825	57	20
B	1441	4015	79	18

13.2.4 Design

This exercise is based on data obtained from an experiment carried out by Briand, Bunse and Daly. The experiment is further described in [Briand99].

An experiment is designed in order to evaluate the impact of quality object-oriented design principles when intending to modify a given design. The quality design principles evaluated are the principles provided by Coad and Yourdon, [Coad91]. In the experiment two systems are used with one design for each system. One of the designs is a "good" design made using the design principles and the other is a "bad" design not using the principles. The two designs are documented in the same way in terms of layout and content and are of the same size, i.e. they are developed to be as similar as possible except for following or not following the

design principles. The objective of the experiment is to evaluate if the quality design principles ease impact analysis when identifying changes in the design.

The task for each participant is to undertake two separate impact analyses, one for each system design. Marking all places in the design that have to be changed but not actually change them makes the impact analyses. The first impact analysis is for a changed customer requirement and the second is for an enhancement in the systems functionality. Four measures are collected during the task:

- Time for identification: Time spent on identifying places for modification.
- Completeness of impact analysis: Represents the completeness of the impact analysis and is defined as $\frac{\textit{Number of correct places found}}{\textit{Total number of places to be found}}$.
- Correctness of impact analysis: Represents the correctness of the impact analysis and is defined as $\frac{\textit{Number of correct places found}}{\textit{Number of places indicated as found}}$.
- Identification rate: The number of correct places found per time unit, that is $\frac{\textit{Number of correct places found}}{\textit{Time for identification}}$.

The experiment is conducted at two occasions, in order to let each participant work with both the good design and the bad design. The subjects were randomly assigned to one of two groups, A or B. Group A worked with the good design at the first occasion and the bad design in the second. Group B studied the bad design first and then the good design. The collected data can be found in Table 41.

1. Which design has been used in the experiment?
2. Define the hypotheses for the evaluation.
3. How should the missing values in Table 41 be treated?
4. Assume that parametric tests can be used. Evaluate the effect of the quality design principles on the four measured variables. Which conclusions can be drawn from the results?
5. Evaluate the effect of the quality design principles on the four measured variables using non-parametric tests. Which conclusions can be drawn from the results? Compare the results to those achieved when using parametric tests.
6. Discuss the validity of the results and if it is appropriate to use parametric tests.
7. The participants in the experiment are students taking a software engineering course that have volunteered to be subjects. From which population is the sample taken from? Discuss how this type of sampling will affect the external validity of the experiment? How can the sampling be made differently?

Table 41. *Data for design exercise.*

		Good Object-Oriented design				Bad Object-Oriented design			
Participant	Group	Time for identification	Completeness of impact analysis	Correctness of impact analysis	Identification rate	Time for Identification	Completeness of impact analysis	Correctness of impact analysis	Identification rate
P01	B	-	0.545	0.75	-	-	0.238	0.714	-
P02	B	-	0.818	1	-	-	0.095	1	-
P03	A	20	0.409	1	0.45	25	0.19	1	0.16
P04	B	22	0.818	1	0.818	25	0.238	1	0.2
P05	B	30	0.909	1	0.667	35	0.476	0.909	0.286
P07	A	-	0	-	-	38	0.476	1	0.263
P09	A	-	0.455	1	-	-	0.476	1	-
P10	B	-	0.409	0.9	-	-	0.381	1	-
P11	A	45	0.545	0.923	0.267	50	0.714	1	0.3
P12	B	-	0.773	1	-	-	0.714	1	-
P13	A	40	0.773	1	0.425	40	0.762	1	0.4
P14	B	30	0.909	1	0.667	30	0.333	0.875	0.233
P15	B	-	0.864	1	-	40	0.238	1	0.125
P16	B	30	0.773	1	0.567	-	-	-	-
P17	B	-	0.955	1	-	-	0.286	0.75	-
P18	B	-	0	-	-	-	0.19	1	-
P19	A	29	0.818	1	0.621	27	0.667	1	0.519
P20	A	9	0.591	1	1,444	15	0.19	0.8	0.267
P21	B	20	0.591	1	0.65	35	0.19	1	0.114
P22	B	30	0.682	1	0.5	20	0.714	1	0.75
P23	B	-	0.818	1	-	-	0.476	1	-
P24	A	30	0.773	1	0.567	40	0.762	1	0.4
P25	A	-	0.955	1	-	-	0.667	0.875	-
P26	B	25	0	0	0	25	0.095	0.5	0.08
P27	A	27	0.773	0.944	0.63	36	0.389	0.7	0.194
P28	A	25	0.773	1	0.68	30	0.667	1	0.467

Table 41. *Data for design exercise.*

		Good Object-Oriented design				Bad Object-Oriented design			
Participant	**Group**	**Time for identification**	**Completeness of impact analysis**	**Correctness of impact analysis**	**Identification rate**	**Time for Identification**	**Completeness of impact analysis**	**Correctness of impact analysis**	**Identification rate**
P29	B	44	0.773	1	0.386	23	0.762	1	0.696
P31	A	-	0.409	1	-	-	0.286	0.75	-
P32	A	30	0.909	1	0.667	-	0.5	1	-
P33	A	65	0.818	1	0.277	-	0.619	1	-
P34	A	50	0.636	0.933	0.28	30	0.4	0.889	0.267
P35	A	10	0.591	1	1,3	10	0.667	1	1,4
P36	A	13	1	1	1,692	-	0.619	1	-

13.2.5 Inspections

In this exercise, an inspection experiment should be evaluated. The experiment has been conducted as follows. The objective was to evaluate the difference between three different perspectives when performing an inspection of a requirements specification. A perspective means providing a view for the reader to use when reading the document. This is often referred to as perspective-based reading (PBR) [Basili96a]. In PBR, the reader produces different documents during the inspection depending on the perspective. The objective of this is to have an active reading. The three perspectives used in the experiment are user, tester and designer. Two different requirements specifications were used in the experiment. The first is for a parking garage control system (PG) and it consists of 16 pages and it has 30 defects. The second specification is for an automatic teller machine (ATM), which is 17 pages long and has 29 defects. It should be noted that the defects for the two documents are not the same, although they are shown in the same table, see Table 42. The experiment was conducted with 30 students. They were assigned by random to one document and one perspective, i.e. five students were assigned to each document and each perspective. The data in Table 42 shows the number of reviewers who found a specific defect. It should be observed that the maximum number is five reviewers. The data for the individual students is shown in Table 43. The data in Table 43 is:

- Id - This is an id number for the subjects or reviewers.
- Perspective - This indicates which perspective the reviewer used (U - User, T - Tester and D - Designer)
- Document - This column shows which document the subject reviewed.
- Time - All subjects record the time spent in individual preparation. The time unit used is minutes.
- Defects - The number of defects found by each reviewer is recorded, excluding false positives. The experimenters remove the false positives, in order to ensure that all defect candidates are treated equally.
- Efficiency - The fault finding efficiency, i.e. the number of defects found per hour, calculated as (Defects*60)/Time.
- Rate - The fault finding effectiveness, i.e. the fraction of found defects by total number of defects (also called detection rate) is calculated as Defects divided by the total number of known defects contained in the inspected documents.

The objective of the experiment was to evaluate if there was any difference between the perspectives in terms of defect detection ability. Moreover, it is important to investigate that the results obtained are not just a result of the document. In other words, that we obtain one result for one document and another result for other documents.

The following should be evaluated:

1. How can assumed differences between the perspectives be analyzed?
2. How can assumed differences between the documents be analyzed? What are the implications if there happens to be a difference between the documents?
3. Perform a complete experiment for the analyses that can be carried out. This includes carrying out all the steps in the experiment process as described in Chapter 4, with the exception that the actual operation of the experiment has already been conducted. You do not need to carry out instrumentation (part of planning) and you can also skip the operation step; instead assume that these steps have resulted in Tables 42 and 43.

Table 42. *Defect data for each document.*

	PG document			ATM document		
Defect number	**User Perspective**	**Tester Perspective**	**Designer Perspective**	**User Perspective**	**Tester Perspective**	**Designer Perspective**
1	2	1	2	1	1	3
2	4	2	3	2	4	4
3	0	3	1	0	0	2
4	0	1	1	3	2	4
5	3	2	2	0	0	0
6	2	2	3	0	1	2
7	0	1	1	1	0	2
8	5	4	3	1	3	3
9	4	3	1	0	0	2
10	0	1	0	2	0	1
11	4	0	2	1	3	2
12	1	1	0	3	3	1
13	0	0	0	2	0	3
14	1	4	4	0	0	0
15	1	2	1	1	2	0
16	1	0	2	2	3	2
17	2	1	1	0	1	1
18	0	1	0	1	0	1
19	0	0	0	1	2	1
20	0	0	0	1	2	0
21	1	1	3	0	0	0
22	1	0	1	0	0	1
23	1	0	1	0	2	0
24	2	1	0	0	0	0
25	0	0	0	0	0	0
26	0	0	0	0	2	0

Table 42. *Defect data for each document.*

	PG document			ATM document		
Defect number	User Perspective	Tester Perspective	Designer Perspective	User Perspective	Tester Perspective	Designer Perspective
27	1	0	0	1	0	3
28	2	0	1	2	1	3
29	0	0	0	3	2	1
30	2	1	0	-	-	-

Table 43. *Data for each subject.*

Id	Perspective	Document	Time	Defects	Efficiency	Rate
1	U	ATM	187	8	2.567	0.276
2	D	PG	150	8	3.200	0.267
3	T	ATM	165	9	3.273	0.310
4	U	PG	185	11	3.568	0.367
5	D	ATM	155	8	3.097	0.276
6	T	PG	121	8	3.967	0.267
7	U	ATM	190	7	2.211	0.241
8	D	PG	260	7	1.615	0.233
9	T	ATM	123	6	2.927	0.207
10	U	PG	155	6	2.323	0.200
11	D	ATM	210	11	3.143	0.379
12	T	PG	88	9	6.136	0.300
13	U	ATM	280	11	2.357	0.379
14	D	PG	145	11	4.552	0.367
15	T	ATM	170	5	1.765	0.172
16	U	PG	120	6	3.000	0.200
17	D	ATM	190	9	2.842	0.310
18	T	PG	97	5	3.093	0.167
19	U	ATM	295	2	0.407	0.069
20	D	PG	180	7	2.333	0.233

Table 43. *Data for each subject.*

Id	Perspective	Document	Time	Defects	Efficiency	Rate
21	T	ATM	306	7	1.373	0.241
22	U	PG	223	4	1.076	0.133
23	D	ATM	157	6	2.293	0.207
24	T	PG	130	6	2.769	0.200
25	U	ATM	195	13	4.000	0.448
26	D	PG	200	7	2.100	0.233
27	T	ATM	195	8	2.462	0.276
28	U	PG	125	5	2.400	0.167
29	D	ATM	200	8	2.400	0.276
30	T	PG	150	5	2.000	0.167

13.3 Reviewing

This section includes a list of questions, which are important to consider when reading or reviewing an article presenting an experiment. Use the list below and review the examples presented in Chapters 11-12, and also some experiment presented in the literature. Some examples of possible experiments to choose to review are presented in Chapter 10.

The list below should be seen as a checklist in addition to normal questions when reading an article. An example of a normal question may be; is the abstract a good description of the content of the paper? Some specific aspects to consider when reading an experiment article are:

- Is the experiment understandable and interesting in general?
- Does the experiment have any practical value?
- Are other experiments addressing the problem summarized and referenced?
- What is the population in the experiment?
- Is the sample used representative of the population?
- Are the dependent and independent variables clearly defined?
- Are the hypotheses clearly formulated?
- Is the type of design clearly stated?
- Is the design correct?
- Is the instrumentation described properly?
- Is the validity of the experiment treated carefully and convincing?

- Are different types of validity threats addressed properly?
- Has the data been validated?
- Is the statistical power sufficient, are there enough subjects in the experiment?
- Are the appropriate statistical tests applied? Are Parametric or non-parametric tests used and are they used correctly?
- Is the significance level used appropriate?
- Is the data interpreted correctly?
- Are the conclusions correct?
- Are the results not overstated?
- Is it possible to replicate the study?
- Is data provided?
- Is it possible to use the results for performing a meta-analysis?
- Is further work and experimentation in the area outlined?

13.4 Assignments

The assignments presented in this section are based on the following general scenario. A company would like to improve their way of working by changing the software process. You are consulted as an expert in evaluating new techniques and methods in relation to the existing process. The company would like to know whether or not to change their software process.

You are expected to search for appropriate literature, review the existing literature on the subject, apply the experiment process and write a report containing a recommendation for the company. The recommendation should discuss both the results of the experiment and other relevant issues for taking the decision whether or not to change the process. Other relevant issues include costs and benefits for making the change. If you are unable to find the correct costs, you are expected to make estimates. The latter may be in terms of relative costs.

The assignments are intentionally fairly open-ended to allow for interpretation and discussion. Each assignment is described in terms of prerequisites needed to perform the assignment and then the actual task is briefly described.

It should be noted that the assignments below are examples of possible experiments that can be conducted. The important issue to hold in mind is that the main objective is that the assignments should provide practice in using experiments as part of an evaluation procedure.

Finally, it should be noted that some organizations provide what is called lab packages that can be used to replicate experiments. Lab packages are important as

they allow us to build upon work by others and hence hopefully come to more generally valid results by replication. Some lab packages can be found by a search on the Internet. It may also be beneficial to contact the original experimenter to get support and maybe also a non-published lab package.

13.4.1 Unit test and code reviews

The company wants to evaluate if it is cost-effective to introduce code reviews. Unit testing is done today, although on non-reviewed code. Is this the best way to do it?

Prerequisites.

- Suitable programs with defects that can be found during either reviews or testing.
- A review method, which may be ad hoc, but preferable it should be something more realistic, for example, a checklist-based approach. In this case, a checklist is needed.
- A testing method, which also may be ad hoc, but preferably it is based on, for example, usage or equivalence partitioning.

Task. Evaluate if it is cost-effective to introduce code reviews.

13.4.2 Inspection methods

Several different ways of conducting reviews are available. The company intends to introduce the best inspection method out of two possible choices. Which of the two methods is the best to introduce for the company?

Prerequisites.

- Suitable software artifacts to review should be available.
- Two review methods with appropriate support in terms of, for example, checklists or description of different reading perspectives, see also Section 13.2.5.

Task. Assume that the company intends to introduce reviews of the chosen software artifacts, which method should they introduce? Determine which of the inspection methods that is best in finding defects. Is the best method also cost-effective?

13.4.3 Requirements notation

It is important to write requirements specification so that all readers interpret them easily and in the same way. The company has several different notations to choose from. Which is the best way of representing requirements?

Prerequisites.

- A requirements specification written in several different notations, for example, natural language and different graphical representations.

Task. Evaluate if it is beneficial to change the company's notation for requirements specifications. Assume that the company uses natural language today.

APPENDIX A: STATISTICAL TABLES

This appendix contains statistical tables for a significance level of 5%. More elaborated tables can be found in most books on statistics, for example [Marascuilo88], and tables are also available on the Internet. Some links to resources in experimentation and in particular on experimentation in software engineering are provided on the home page for the book, see Preface.

The main objective here is to provide some information, so that the tests that are explained in Chapter 8 become understandable and so that the examples provided can be followed. This is important even if statistical packages are used for the calculations, since it is important to understand the underlying calculations before just applying the different statistical tests. It is also worth noting the tables are a short-cut, for example, the values for the t-test, F-test and Chi-2 can be calculated from the respective distributions.

The following statistical tables are included:

- t-test (see Section 8.3.4, 8.3.7, and Table A1)
- Chi-2 (see Section 8.3.12 and Table A2)
- Mann-Whitney (see Section 8.3.5 and Table A3)
- Wilcoxon (see Section 8.3.8 and Table A4)
- F-test (see Section 8.3.6, 8.3.10, Table A5.1 and Table A5.2)

Table A1. *Critical values two-tailed t-test (5%), see Section 8.3.4 and 8.3.7.*

Degrees of freedom	t-value
1	12.706
2	4.303
3	3.182
4	2.776
5	2.571
6	2.447
7	2.365
8	2.306
9	2.262
10	2.228
11	2.201
12	2.179
13	2.160
14	2.145
15	2.131
16	2.120
17	2.110
18	2.101
19	2.093
20	2.086
21	2.080
22	2.074
23	2.069
24	2.064
25	2.060
26	2.056
27	2.052
28	2.048
29	2.045
30	2.042
40	2.021
60	2.000
120	1.980
∞	1.960

Table A2. *Critical values one-tailed Chi-2 test (5%), see Section 8.3.12.*

Degrees of freedom	χ^2
1	3.84
2	5.99
3	7.81
4	9.49
5	11.07
6	12.59
7	14.07
8	15.51
9	16.92
10	18.31
11	19.68
12	21.03
13	22.36
14	23.68
15	25.00
16	26.30
17	27.59
18	28.87
19	30.14
20	31.41
21	32.67
22	33.92
23	35.17
24	36.42
25	37.65
26	38.88
27	40.11
28	41.34
29	42.56
30	43.77
40	55.76
60	79.08
80	101.88
100	124.34

Table A3. *Critical values two-tailed Mann-Whitney (5%), see Section 8.3.5.*

N_A \ N_B	5	6	7	8	9	10	11	12
3	0	1	1	2	2	3	3	4
4	1	2	3	4	4	5	6	7
5	2	3	5	6	7	8	9	11
6		5	6	8	10	11	13	14
7			8	10	12	14	16	18
8				13	15	17	19	22
9					17	20	23	26
10						23	26	29
11							30	33
12								37

Please note that in Table A3, N_A is for the smaller sample and N_B for the larger sample.

Table A4. *Critical values two-tailed matched-pair Wilcoxon test (5%), see Section 8.3.8.*

n	T
6	0
7	2
8	3
9	5
10	8
11	10
12	13
13	17
14	21
15	25

n	T
16	29
17	34
18	40
19	46
20	52
22	66
25	89

Please note that Tables A5.1 and A5.2 provide the upper 0.025 percentage point of the F distribution with f_1 and f_2 being the degrees of freedom. This is equivalent to $F_{0.025, f_1, f_2}$.

Table A5.1. *Critical values two-tailed F-test (5%), see Section 8.3.6. For ANOVA, this is equivalent to a significance level of 2.5%, see Section 8.3.10.*

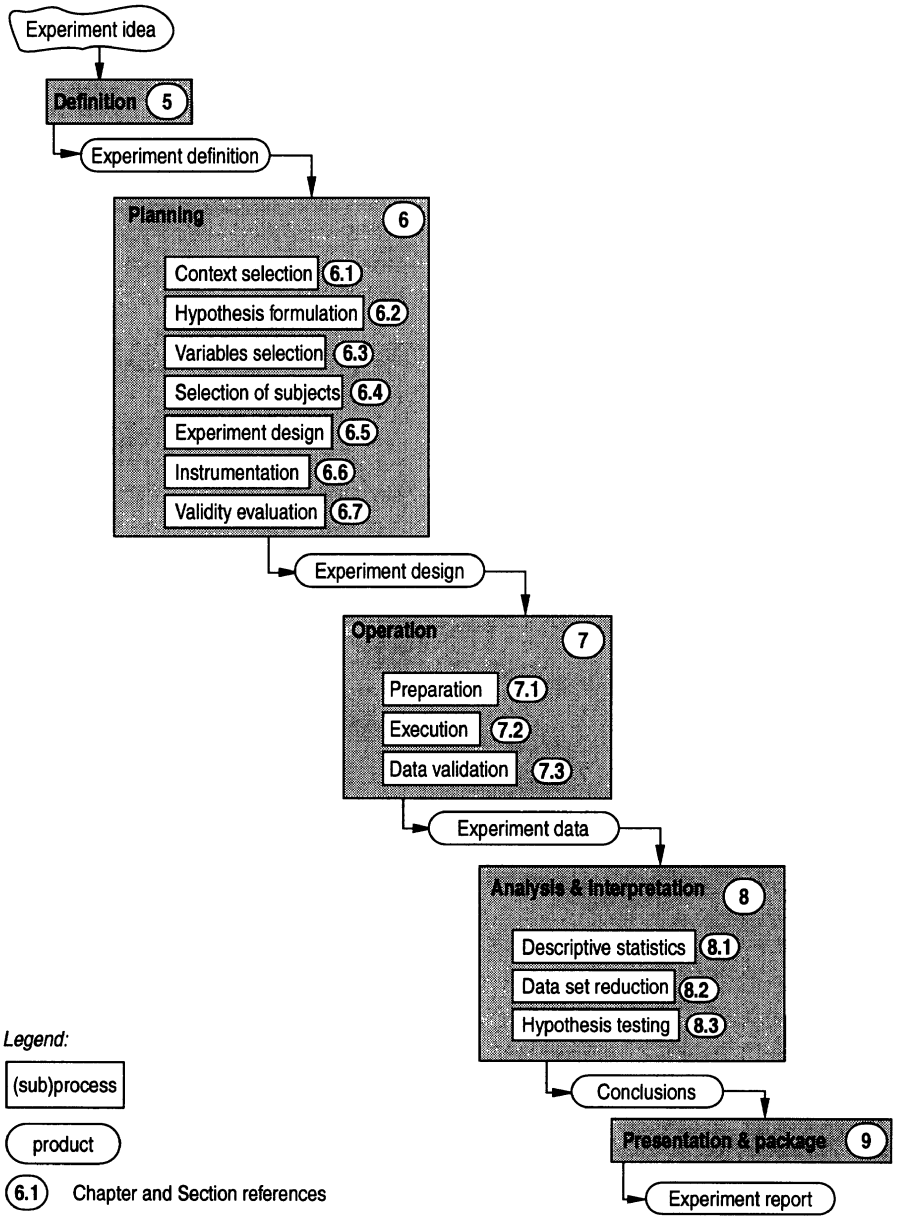
f_2 \ f_1	1	2	3	4	5	6	7	8	9	10
1	648	800	864	900	922	937	948	957	963	969
2	38.5	39.0	39.2	39.2	39.3	39.3	39.4	39.4	39.4	39.4
3	17.4	16.0	15.4	15.1	14.9	14.7	14.6	14.5	14.5	14.4
4	12.2	10.6	9.98	9.60	9.36	9.20	9.07	8.98	8.90	8.84
5	10.0	8.43	7.76	7.39	7.15	6.98	6.85	6.76	6.68	6.62
6	8.81	7.26	6.60	6.23	5.99	5.82	5.70	5.60	5.52	5.46
7	8.07	6.54	5.89	5.52	5.29	5.12	4.99	4.90	4.82	4.76
8	7.57	6.06	5.42	5.05	4.82	4.65	4.53	4.43	4.36	4.30
9	7.21	5.71	5.08	4.72	4.48	4.32	4.20	4.10	4.03	3.96
10	6.94	5.46	4.83	4.47	4.24	4.07	3.95	3.85	3.78	3.72
12	6.55	5.10	4.47	4.12	3.89	3.73	3.61	3.51	3.44	3.37
15	6.20	4.76	4.15	3.80	3.58	3.41	3.29	3.20	3.12	3.06
20	5.87	4.46	3.86	3.51	3.29	3.13	3.01	2.91	2.84	2.77
30	5.57	4.18	3.59	3.25	3.03	2.87	2.75	2.65	2.57	2.51
40	5.42	4.05	3.46	3.13	2.90	2.74	2.62	2.53	2.45	2.39
60	5.29	3.93	3.34	3.01	2.79	2.63	2.51	2.41	2.33	2.27
120	5.15	3.80	3.23	2.89	2.67	2.52	2.39	2.30	2.22	2.16
∞	5.02	3.69	3.12	2.79	2.57	2.41	2.29	2.19	2.11	2.05

Please note that Tables A5.1 and A5.2 provide the upper 0.025 percentage point of the F distribution with f_1 and f_2 being the degrees of freedom. This is equivalent to $F_{0.025, f_1, f_2}$.

Table A5.2. *Critical values two-tailed F-test (5%), see Section 8.3.6. For ANOVA, this is equivalent to a significance level of 2.5%, see Section 8.3.10.*

f_2 \ f_1	12	15	20	30	40	60	120	∞
1	977	985	993	1001	1006	1010	1014	1018
2	39.4	39.4	39.4	39.5	39.5	39.5	39.5	39.5
3	14.3	14.2	14.2	14.1	14.0	14.0	14.0	13.9
4	8.75	8.66	8.56	8.46	8.41	8.36	8.31	8.26
5	6.52	6.43	6.33	6.23	6.18	6.12	6.07	6.02
6	5.37	5.27	5.17	5.07	5.01	4.96	4.90	4.85
7	4.67	4.57	4.47	4.36	4.31	4.25	4.20	4.14
8	4.20	4.10	4.00	3.89	3.84	3.78	3.73	3.67
9	3.87	3.77	3.67	3.56	3.51	3.45	3.39	3.33
10	3.62	3.52	3.42	3.31	3.26	3.20	3.14	3.08
12	3.28	3.18	3.07	2.96	2.91	2.85	2.79	2.72
15	2.96	2.86	2.76	2.64	2.59	2.52	2.46	2.40
20	2.68	2.57	2.46	2.35	2.29	2.22	2.16	2.09
30	2.41	2.31	2.20	2.07	2.01	1.94	1.87	1.79
40	2.29	2.18	2.07	1.94	1.88	1.80	1.72	1.64
60	2.17	2.06	1.94	1.82	1.74	1.67	1.58	1.48
120	2.05	1.94	1.82	1.69	1.61	1.53	1.43	1.31
∞	1.94	1.83	1.71	1.57	1.48	1.39	1.27	1.00

APPENDIX B: EXPERIMENT PROCESS OVERVIEW



REFERENCES

APA92 American Psychological Association, "Ethical Principles of Psychologists and Code of Conduct", *American Psychologist*, pp. 1597-1611, Dec. 1992.

Babbie90 E. Babbie, *Survey Research Methods*, Wadsworth, ISBN 0-524-12672-3, 1990.

Basili85 V. R. Basili, "Quantitative Evaluation of Software Engineering Methodology", *Proceedings of the First Pan Pacific Computer Conference*, Melbourne, Australia, Vol. 1, pp. 379-398, 1985.

Basili86 V. R. Basili, R. W. Selby and D. H. Hutchens, "Experimentation in Software Engineering", *IEEE Transactions on Software Engineering*, 12 (7), pp. 733-743, 1986.

Basili87 V. R. Basili and R. W. Selby, "Comparing the Effectiveness of Software Testing Strategies", *IEEE Transactions on Software Engineering*, 13 (12), pp. 1278-1298, 1987.

Basili89 V. R. Basili, "Software Development: A Paradigm for the Future", *Proceedings of the 13th Annual International Computer Software & Applications Conference, COMPSAC'89*, pp. 471-485, Orlando, Florida, USA, 1989.

Basili93 V. R. Basili, "The Experimental Paradigm in Software Engineering", *Experimental Software Engineering Issues: Critical Assessment and Future Directives,* editors: H. D. Rombach, V. R. Basili and R. W. Selby, Springer-Verlag, #706, Lecture Notes in Computer Science, 1993.

Basili94a V. R. Basili, G. Caldiera and H. D. Rombach, "Experience Factory", *Encyclopedia of Software Engineering,* ed. J. J. Marciniak, Vol. I, pp. 469-476, Wiley, 1994.

Basili94b V. R. Basili, G. Caldiera and H. D. Rombach, "Goal Question Metrics Paradigm", *Encyclopedia of Software Engineering*, ed. J. J. Marciniak, Vol. I, pp. 528-532, Wiley, 1994.

Basili94c V. R. Basili and S. Green, "Software Process Evaluation at the SEL", *IEEE Software*, pp. 58-66, July 1994.

Basili96a V. R. Basili, S. Green, O. Laitenberger, F. Lanubile, F. Shull, S. Sørumgård and M.V. Zelkowitz, "The Empirical Investigation of Perspective-Based Reading", *Journal of Empirical Software Engineering*, 1(2), pp. 133-164, 1996.

Basili96b V. R. Basili, "The Role of Experimentation in Software Engineering: Past, Current and Future", *Proceedings 18th International Conference on Software Engineering*, Berlin, Germany, pp. 442-449, 1996.

BPS93 British Psychological Society, "Ethical Principles for Conducting Research with Human Participants", *The Psychologist*, pp. 33-35, Jan. 1993.

Briand96a L. C. Briand, K. El Emam and S. Morasca, "On the Application of Measurement Theory in Software Engineering", *Journal of Empirical Software Engineering*, 1(1), pp. 61-88, 1996.

Briand96b L. C. Briand, C. M. Differding and H. D. Rombach, "Practical Guidelines for Measurement-Based Process Improvement", *Software Process Improvement and Practice*, 2(4), pp. 253-280, 1996.

Briand97 L. C. Briand, C. Bunse, J. W. Daly and C. M. Differding, "An Experimental Comparison of the Maintainability of Object-Oriented and Structured Design Documents", *Journal of Empirical Software Engineering*, 2(3), pp. 291-312, 1997.

Briand99 L. C. Briand, C. Bunse, and J. W. Daly, "A Controlled Experiment for Evaluating Quality Guidelines on the Maintainability of Ob-

ject-Oriented Designs", To appear in IEEE Transactions on Software Engineering, 1999.

Campbell63 D. T. Campbell and J. C. Stanley, *Experimental and Quasi-Experimental Designs for Research*, Houghton Mifflin Company, Boston, MA, USA, 1963.

Coad91 P. Coad and E. Yourdon, *Object-Oriented Design*, Prentice-Hall, first edition, 1991.

Cook79 T. D. Cook and D. T. Campbell, *Quasi-Experimentation – Design and Analysis Issues for Field Settings*, Houghton Mifflin Company, 1979.

Creswell94 J. W. Creswell, *Research Design, Qualitative and Quantitative Approaches*, Sage Publications, 1994.

Daly96 J. W. Daly, A. Brooks, J. Miller, M. Roper and M. Wood, "Evaluating Inheritance Depth on the Maintainability of Object-Oriented Software", *Journal of Empirical Software Engineering*, 1(2), pp. 109-132, 1996.

DeMarco82 T. DeMarco, *Controlling Software Projects*, Yourdon Press, New York, USA, 1982.

Demming86 E. Demming, *Out of the Crisis*, MIT Centre for Advanced Engineering Study, MIT Press, Cambridge, Massachusetts, 1986.

Denzin94 N. K. Denzin and Y.S. Lincoln, *Handbook of Qualitative Research*, Sage Publications, London, UK, 1994.

Fenton94a N. Fenton, S. L. Pfleeger and R. Glass, "Science and Substance: A Challenge to Software Engineers", *IEEE Software*, pp. 86-95, July 1994.

Fenton94b N. Fenton, "Software Measurement: A Necessary Scientific Basis", *IEEE Transactions on Software Engineering*, 3 (20), pp. 199-206, 1994.

Fenton96 N. Fenton, and S. L. Pfleeger, *Software Metrics: A Rigorous & Practical Approach*, 2nd edition, International Thomson Computer Press, 1996.

Frigge89 M. Frigge, D. C. Hoaglin, and B. Iglewicz, "Some Implementations of the Boxplot", *The American Statistician*, 43(1), pp. 50-54, Feb. 1989.

Fusaro97 — P. Fusaro, F. Lanubile and G. Visaggio, "A Replicated Experiment to Assess Requirements Inspection Techniques", *Journal of Empirical Software Engineering*, 2(1), pp. 39-57, 1997.

Glass94 — R. Glass, "The Software Research Crisis", *IEEE Software*, pp. 42-47, Nov. 1994.

Grady94 — R. B. Grady and D. L. Caswell, *Software Metrics: Establishing a Company-Wide Program*, Prentice-Hall, 1994.

Harrison96 — R. Harrison, L. G. Samaraweera, M. R. Dobie and P. H. Lewis, "Comparing Programming Paradigms: an Evaluation of Functional and Object-Oriented Programs", *Software Engineering Journal*, pp. 247-254, July 1996.

Heninger80 — K. L. Heninger, "Specifying Software Requirements for Complex Systems: New Technologies and their Application", *IEEE Transactions on Software Engineering*, 6(1), pp. 2-13, 1980.

Hetzel93 — B. Hetzel, *Making Software Measurement Work: Building an Effective Measurement Program*, John Wiley and Sons, 1993.

Humphrey95 — W. S. Humphrey, *A Discipline for Software Engineering*, Addison Wesley, 1995.

Humphrey97 — W. S. Humphrey, *Introduction to the Personal Software Process*, Addison Wesley, 1997.

IEEE90 — IEEE, "IEEE Standard Glossary of Software Engineering Terminology", IEEE Std 610.12-1990, 1990.

Johnson98 — P. M. Johnson and D. Tjahjono, "Does Every Inspection Really Need a Meeting?", *Journal of Empirical Software Engineering*, 3(1), pp. 9-35, 1998.

Kachigan86 — S. K. Kachigan, *Statistical Analysis: An Interdisciplinary Introduction to Univariate & Multivariate Methods*, Radius Press, New York, 1986.

Kachigan91 — S. K. Kachigan, *Multivariate Statistical Analysis - A Conceptual Introduction*, 2nd Edition, Radius Press, New York, 1991.

Kiper97 — J. Kiper, B. Auernheimer and C. K. Ames, "Visual Depiction of Decision Statements: What is Best for Programmers and Non-Programmers?", *Journal of Empirical Software Engineering*, 2(4), pp. 361-379, 1997.

Kitchenham95 B. Kitchenham, L. Pickard and S. L. Pfleeger, "Case Studies for Method and Tool Evaluation", *IEEE Software*, pp. 52-62, July 1995.

Linger94 R. Linger, "Cleanroom Process Model", *IEEE Software*, pp. 50-58, March 1994.

Linkman97 S. Linkman and H. D. Rombach, "Experimentation as a Vehicle for Software Technology Transfer - A Family of Software Reading Techniques", *Journal of Information and Software Technology*, 39(11), pp. 777-780, 1997.

Lott96 C. M. Lott and H. D. Rombach, "Repeatable Software Engineering Experiments for Comparing Defect-Detection Techniques", *Journal of Empirical Software Engineering*, 1(3), pp. 241-277, 1996.

Lott97 C. M. Lott, "A Controlled Experiment to Evaluate On-Line Process Guidance", *Journal of Empirical Software Engineering*, 2(3), pp. 269-289, 1997.

Lyu96 M. R. Lyu (ed.), *Handbook of Software Reliability Engineering*, McGraw-Hill, 1996.

Manly94 B. F. J. Manly, *Multivariate Statistical Methods - A Primer,* Second Edition, Chapman & Hall, London, 1994.

Marascuilo88 L. A. Marascuilo and R. C. Serlin, *Statistical Methods for the Social and Behavioral Sciences*, W. H. Freeman and Company, New York, USA, 1988.

Miller98 J. Miller, M. Wood and M. Roper, "Further Experiences with Scenarios and Checklists", *Journal of Empirical Software Engineering*, 3(1), pp. 37-64, 1998.

Montgomery97 D. C. Montgomery, *Design and Analysis of Experiments*, 4th edition, John Wiley& Sons, 1997.

Ohlsson98 M. C. Ohlsson and C. Wohlin, "A Project Effort Estimation Study", *Journal of Software and Information Technology*, 40(14), pp. 831-839, 1998.

Paulk93 M. C. Paulk, B. Curtis, M. B. Chrissis and C. V. Weber, "Capability Maturity Model for Software, Version 1.1", Software Engineering Institute, CMU/SEI-93-TR-24, February 1993.

Pfleeger94 S. Pfleeger, "Experimental Design and Analysis in Software Engineering Part 1-5", *ACM Sigsoft, Software Engineering Notes*,

Vol. 19, No. 4, pp. 16-20; Vol. 20, No. 1, pp. 22-26; Vol. 20, No. 2, pp. 14-16; Vol. 20, No. 3, pp. 13-15; and Vol. 20, No. 4, pp. 14-17, 1994-1995.

Pfleeger98 S. Pfleeger, *Software Engineering: Theory and Practice*, Prentice-Hall, 1998.

Porter94 A. A. Porter and L. G. Votta, "An Experiment to Assess Different Defect Detection Methods for Software Requirements Inspections", *Proceedings 16th International Conference on Software Engineering*, Sorrento, Italy, pp. 103-112, 1994.

Porter95 A. A. Porter, L. G. Votta and V. R. Basili, "Comparing Detection Methods for Software Requirements Inspection: A Replicated Experiment", *IEEE Transactions on Software Engineering*, 21(6), pp.563-575, 1995.

Porter97a A. A. Porter and P. M. Johnson, "Assessing Software Review Meetings: Results of a Comparative Analysis of Two Experimental Studies", *IEEE Transactions on Software Engineering*, 23(3), pp. 129-145, 1997.

Porter97b A. A. Porter, H. P. Siy, C. A. Toman and L. G. Votta, "An Experiment to Assess the Cost-Benefits of Code Inspections in Large Scale Software Development", *IEEE Transactions on Software Engineering*, 23(6), pp. 329-346, 1997.

Porter98a A. A. Porter, H. P. Siy, A. Mockus and L. G. Votta, "Understanding the Sources of Variation in Software Inspections", *ACM Transactions on Software Engineering and Methodology*, 7(1), pp. 41-79, 1998.

Porter98b A. A. Porter, and L. G. Votta, "Comparing Detection Methods for Software Requirements Inspection: A Replication Using Professional Subjects", *Journal of Empirical Software Engineering*, 3(4), pp. 355-380, 1998.

Potts93 C. Potts, "Software Engineering Research Revisited", *IEEE Software*, pp. 19-28, Sept. 1993.

Prechelt98 L. Prechelt and W. F. Tichy, "A Controlled Experiment to Assess the Benefits of Procedure Argument Type Checking", *IEEE Transactions on Software Engineering*, 24(4), pp. 302-312, 1998.

Regnell99 B. Regnell, P. Runeson and T. Thelin, "Are the Perspectives Really Different? - Further Experimentation on Scenario-Based Reading of Requirements", Technical Report CODEN: LUT-

EDX(TETS-7172) / 1-40 / 1999 & local 4, Dept. of Communication Systems, Lund University, 1999.

Robson93 C. Robson, *Real World Research: A Resource for Social Scientists and Practitioners-Researchers*, Blackwell, 1993.

Robson94 C. Robson, *Experiment, Design and Statistics in Psychology*, 3rd edition, Penguin Books, London, England, 1994.

Rombach87 H. D. Rombach, "A Controlled Experiment on the Impact of Software Structure on Maintainability", *IEEE Transactions on Software Engineering*, 13(3), pp. 344-354, 1987.

Sandahl98 K. Sandahl, O. Blomkvist, J. Karlsson, C. Krysander, M. Lindvall and N. Ohlsson, "An Extended Replication of an Experiment for Assessing Methods for Software Requirements", *Journal of Empirical Software Engineering*, 3(4), pp. 381-406, 1998.

Seaman98 C. B. Seaman and V. R. Basili, "Communication and Organization: An Empirical Study of Discussion in Inspection Meeting", *IEEE Transactions on Software Engineering*, 24(7), pp. 559-572, 1998.

Selby87 R. W. Selby, V. R. Basili and F. T. Baker, "Cleanroom Software Development: An Empirical Evaluation", *IEEE Transactions on Software Engineering*, 13(9), pp. 1027-1037, 1987.

Shepperd95 M. Shepperd, *Foundations of Software Measurement*, Prentice-Hall, 1995.

Siegel88 S. Siegel and J. Castellan, *Nonparametric Statistics for the Behavioral Sciences*, 2nd edition, McGraw-Hill International Editions, 1988.

Solingen99 R. van Solingen and E. Berghout, *The Goal/Question/Metric Method: A Practical Guide for Quality Improvement and Software Development"*, McGraw-Hill International, 1999.

Sommerville95 I. Sommerville, *Software Engineering*, Addison-Wesley, 1995.

Stake95 R. E. Stake, *The Art of Case Study Research*, SAGE Publications, 1995.

Tichy95 W. F. Tichy, P. Lukowicz, L. Prechelt and E. A. Heinz, "Experimental Evaluation in Computer Science: A Quantitative Study", *Journal of Systems and Software*, 28 (1), pp. 9-18, 1995.

Tichy98 W. F. Tichy, "Should Computer Scientists Experiment More?", *IEEE Computer*, 31 (5), pp. 32-39, 1998.

Trochim99 W. Trochim, *The Research Methods Knowledge Base*, 2nd Edition, Cornell Custom Publishing, Cornell University, Ithaca, New York, 1999.

Votta93 L. G. Votta, "Does Every Inspection Need a Meeting?", *Proceedings of the ACM SIGSOFT 1993 Symposium on Foundations of Software Engineering, ACM Software Engineering Notes*, 18(5), pp. 107-114, 1993.

Wohlin96 C. Wohlin, A. Gustavsson, M. Höst and C. Mattsson, "A Framework for Technology Introduction in Software Organizations", *Proceedings Conference on Software Process Improvement*, Brighton, United Kingdom, pp. 167-176, 1996.

Wohlin98a C. Wohlin, "The Personal Software Process as a Context for Empirical Studies", *Software Process Newsletter*, No. 12, pp. 7-12, 1998.

Wohlin98b C. Wohlin, "Evaluation of the Performance in the PSP Course based on Student Background and Experience", Technical report, Dept. of Communication Systems, Lund University, 1998.

Yin94 R. K. Yin, *Case Study Research Design and Methods*, Sage Publications, Beverly Hills, California, 1994.

Zelkowitz98 M. V. Zelkowitz and D. R. Wallace, "Experimental Models for Validating Technology", *IEEE Computer*, 31(5), pp. 23-31, 1998.

ABOUT THE AUTHORS

All of the authors have a strong focus on empirical software engineering research. The authors are all members of the Software Engineering Research Group at the Department of Communication Systems, Lund University in Sweden. The research group is focused on issues related to large-scale software development. More information about the research group can be found at http://www.telecom.lth.se/SERG/. The research group is a member of the international network ISERN (International Software Engineering Research Network) which has empirical software engineering research as an important issue on the agenda.

Prof. Claes Wohlin. Prof. Wohlin is a professor of software engineering at the Department of Communication Systems, Lund University. Prior to this, he was professor of software engineering at Linköping University. He has a Ph.D. in Communication Systems from Lund University. He is director of the research group in software engineering consisting of 11 people in Lund and responsible for the education and research in the area of software engineering in telecommunications. He is also vice head of the department. His research interests include empirical methods in software engineering, software metrics, software quality and process improvement. Claes Wohlin is on the editorial board of the journal of Information and Software Technology.

Dr. Per Runeson. Dr. Runeson is an assistant professor of software engineering at the Department of Communication Systems, Lund University. He has a Master of

Science in Computer Science and Engineering and a Ph.D. of Software Engineering from Lund University, Sweden. He has industrial experience from working as a consulting expert in software engineering for six years. He is now responsible for a Bachelor's program in Software Engineering and is heading a research project with focus on requirements engineering and verification & validation. His research interests cover software quality, empirical methods and software process improvement.

Dr. Martin Höst. Dr. Höst is an assistant professor of software engineering at the Department of Communication Systems, Lund University. He has a Master of Science in Computer Science and Engineering and a Ph.D. of Software Engineering from Lund University. His main research interests is early evaluation of software process improvement proposals, carried out through interviews or experimentation.

Mr. Magnus C. Ohlsson. Mr. Ohlsson is a doctoral student at the Department of Communication Systems, Lund University. He received his Master of Science in Software Engineering from University of Karlskrona/Ronneby in Sweden and his Licentiate in Engineering from the Department of Communication Systems, Lund University. His main research areas are the use of experience to build prediction models and models for process improvement. Most of the work has been carried out within effort prediction and maintenance.

Dr. Björn Regnell. Dr. Regnell is an assistant professor of software engineering at the Department of Communication Systems, Lund University. He has a Master of Science in Computer Science and Engineering and a Ph.D. of Software Engineering from Lund University. He is now heading a research project with focus on market-driven requirements engineering. His research interests include scenarios, prioritization, usability, as well as empirical methods and software process improvement within requirements engineering.

Dr. Anders Wesslén. Dr. Wesslén is a postdoctoral fellow in software engineering at the Department of Communication Systems, Lund University. Prior to this, he has taken a Master of Science in Computer Science and Engineering and a Ph.D. of Software Engineering from Lund University. He is currently working in a research project with focus on requirements engineering and verification & validation. His research interests cover software verification & validation, software quality and empirical methods.

Index